Assignment 3: Time-Series Data

April 7, 2024

[1]:
```
!pip install tensorflow==2.12
```

```
Collecting tensorflow==2.12
  Downloading
tensorflow-2.12.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(585.9 MB)
                                        585.9/585.9
MB 2.0 MB/s eta 0:00:00
Requirement already satisfied: absl-py>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (1.6.3)
Requirement already satisfied: flatbuffers>=2.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (24.3.25)
Collecting gast<=0.4.0,>=0.2.1 (from tensorflow==2.12)
  Downloading gast-0.4.0-py3-none-any.whl (9.8 kB)
Requirement already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (0.2.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (1.62.1)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-
packages (from tensorflow==2.12) (3.9.0)
Requirement already satisfied: jax>=0.3.15 in /usr/local/lib/python3.10/dist-
packages (from tensorflow==2.12) (0.4.23)
Collecting keras<2.13,>=2.12.0 (from tensorflow==2.12)
  Downloading keras-2.12.0-py2.py3-none-any.whl (1.7 MB)
                                        1.7/1.7 MB
43.8 MB/s eta 0:00:00
Requirement already satisfied: libclang>=13.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (18.1.1)
Collecting numpy<1.24,>=1.22 (from tensorflow==2.12)
  Downloading
numpy-1.23.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.1
MB)
                                        17.1/17.1 MB
38.3 MB/s eta 0:00:00
Requirement already satisfied: opt-einsum>=2.3.2 in
```

```
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (3.3.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-
packages (from tensorflow==2.12) (24.0)
Requirement already satisfied:
protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3
in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (3.20.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-
packages (from tensorflow==2.12) (67.7.2)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-
packages (from tensorflow==2.12) (1.16.0)
Collecting tensorboard<2.13,>=2.12 (from tensorflow==2.12)
  Downloading tensorboard-2.12.3-py3-none-any.whl (5.6 MB)
                                 5.6/5.6 MB
56.5 MB/s eta 0:00:00
Collecting tensorflow-estimator<2.13,>=2.12.0 (from tensorflow==2.12)
  Downloading tensorflow_estimator-2.12.0-py2.py3-none-any.whl (440 kB)
                                 440.7/440.7

kB 16.7 MB/s eta 0:00:00
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (4.10.0)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (0.36.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.10/dist-packages (from
astunparse>=1.6.0->tensorflow==2.12) (0.43.0)
Requirement already satisfied: ml-dtypes>=0.2.0 in
/usr/local/lib/python3.10/dist-packages (from jax>=0.3.15->tensorflow==2.12)
(0.2.0)
Requirement already satisfied: scipy>=1.9 in /usr/local/lib/python3.10/dist-
packages (from jax>=0.3.15->tensorflow==2.12) (1.11.4)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.13,>=2.12->tensorflow==2.12) (2.27.0)
Collecting google-auth-oauthlib<1.1,>=0.5 (from
tensorboard<2.13,>=2.12->tensorflow==2.12)
  Downloading google_auth_oauthlib-1.0.0-py2.py3-none-any.whl (18 kB)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.13,>=2.12->tensorflow==2.12) (3.6)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.13,>=2.12->tensorflow==2.12) (2.31.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in
```

```
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.13,>=2.12->tensorflow==2.12) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.13,>=2.12->tensorflow==2.12) (3.0.2)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from google-
auth<3,>=1.6.3->tensorboard<2.13,>=2.12->tensorflow==2.12) (5.3.3)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from google-
auth<3,>=1.6.3->tensorboard<2.13,>=2.12->tensorflow==2.12) (0.4.0)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-
packages (from google-auth<3,>=1.6.3->tensorboard<2.13,>=2.12->tensorflow==2.12)
(4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/usr/local/lib/python3.10/dist-packages (from google-auth-
oauthlib<1.1,>=0.5->tensorboard<2.13,>=2.12->tensorflow==2.12) (1.3.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from
requests<3,>=2.21.0->tensorboard<2.13,>=2.12->tensorflow==2.12) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests<3,>=2.21.0->tensorboard<2.13,>=2.12->tensorflow==2.12)
(3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from
requests<3,>=2.21.0->tensorboard<2.13,>=2.12->tensorflow==2.12) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from
requests<3,>=2.21.0->tensorboard<2.13,>=2.12->tensorflow==2.12) (2024.2.2)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/usr/local/lib/python3.10/dist-packages (from
werkzeug>=1.0.1->tensorboard<2.13,>=2.12->tensorflow==2.12) (2.1.5)
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in
/usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-
auth<3,>=1.6.3->tensorboard<2.13,>=2.12->tensorflow==2.12) (0.6.0)
Requirement already satisfied: oauthlib>=3.0.0 in
/usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->google-
auth-oauthlib<1.1,>=0.5->tensorboard<2.13,>=2.12->tensorflow==2.12) (3.2.2)
Installing collected packages: tensorflow-estimator, numpy, keras, gast, google-
auth-oauthlib, tensorboard, tensorflow
  Attempting uninstall: tensorflow-estimator
    Found existing installation: tensorflow-estimator 2.15.0
    Uninstalling tensorflow-estimator-2.15.0:
      Successfully uninstalled tensorflow-estimator-2.15.0
  Attempting uninstall: numpy
    Found existing installation: numpy 1.25.2
    Uninstalling numpy-1.25.2:
      Successfully uninstalled numpy-1.25.2
```

```
  Attempting uninstall: keras
    Found existing installation: keras 2.15.0
    Uninstalling keras-2.15.0:
      Successfully uninstalled keras-2.15.0
  Attempting uninstall: gast
    Found existing installation: gast 0.5.4
    Uninstalling gast-0.5.4:
      Successfully uninstalled gast-0.5.4
  Attempting uninstall: google-auth-oauthlib
    Found existing installation: google-auth-oauthlib 1.2.0
    Uninstalling google-auth-oauthlib-1.2.0:
      Successfully uninstalled google-auth-oauthlib-1.2.0
  Attempting uninstall: tensorboard
    Found existing installation: tensorboard 2.15.2
    Uninstalling tensorboard-2.15.2:
      Successfully uninstalled tensorboard-2.15.2
  Attempting uninstall: tensorflow
    Found existing installation: tensorflow 2.15.0
    Uninstalling tensorflow-2.15.0:
      Successfully uninstalled tensorflow-2.15.0
ERROR: pip's dependency resolver does not currently take into account all
the packages that are installed. This behaviour is the source of the following
dependency conflicts.
chex 0.1.86 requires numpy>=1.24.1, but you have numpy 1.23.5 which is
incompatible.
pandas-stubs 2.0.3.230814 requires numpy>=1.25.0; python_version >= "3.9", but
you have numpy 1.23.5 which is incompatible.
tf-keras 2.15.1 requires tensorflow<2.16,>=2.15, but you have tensorflow 2.12.0
which is incompatible.
Successfully installed gast-0.4.0 google-auth-oauthlib-1.0.0 keras-2.12.0
numpy-1.23.5 tensorboard-2.12.3 tensorflow-2.12.0 tensorflow-estimator-2.12.0
```

[3]:

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

```
--2024-04-07 00:18:57-- https://s3.amazonaws.com/keras-
datasets/jena_climate_2009_2016.csv.zip
Resolving s3.amazonaws.com (s3.amazonaws.com)... 54.231.140.40, 54.231.227.168,
52.217.122.56, ...
Connecting to s3.amazonaws.com (s3.amazonaws.com)|54.231.140.40|: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 57.7MB/s    in 0.2s

2024-04-07 00:18:58 (57.7 MB/s) - 'jena_climate_2009_2016.csv.zip' saved
[13565642/13565642]

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

Inspecting the data of the Jena weather dataset - 420451 rows and 15 Features

[4]:
```python
import os
fname = os.path.join("jena_climate_2009_2016.csv")

with open(fname) as f:
    data = f.read()

lines = data.split("\n")
header = lines[0].split(",")
lines = lines[1:]
print(header)
print(len(lines))

num_variables = len(header)
print("Number of variables:", num_variables)
num_rows = len(lines)
print("Number of rows:", num_rows)
```

```
['"Date Time"', '"p (mbar)"', '"T (degC)"', '"Tpot (K)"', '"Tdew (degC)"', '"rh
(%)"', '"VPmax (mbar)"', '"VPact (mbar)"', '"VPdef (mbar)"', '"sh (g/kg)"',
'"H2OC (mmol/mol)"', '"rho (g/m**3)"', '"wv (m/s)"', '"max. wv (m/s)"', '"wd
(deg)"']
420451
Number of variables: 15
Number of rows: 420451
```

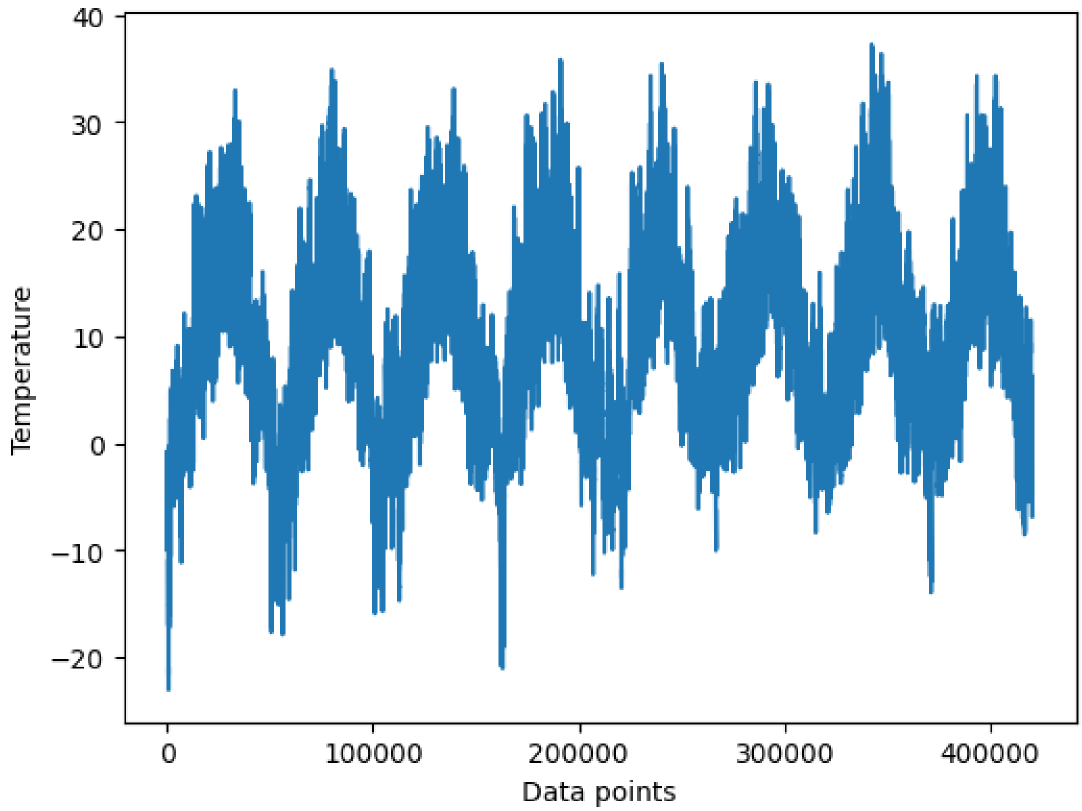
Parsing the data- converting the comma-separated values into floating-point numbers, and then storing specific values in the temperature and raw_data arrays for further processing or analysis.

[5]:
```python
import numpy as np
temperature = np.zeros((len(lines),))
raw_data = np.zeros((len(lines), len(header) - 1))
for i, line in enumerate(lines):
    values = [float(x) for x in line.split(",")[1:]]
    temperature[i] = values[1]
    raw_data[i, :] = values[:]
```

Plotting the temperature timeseries

[6]:
```
from matplotlib import pyplot as plt
plt.plot(range(len(temperature)), temperature)
plt.xlabel('Data points')
plt.ylabel('Temperature')
```

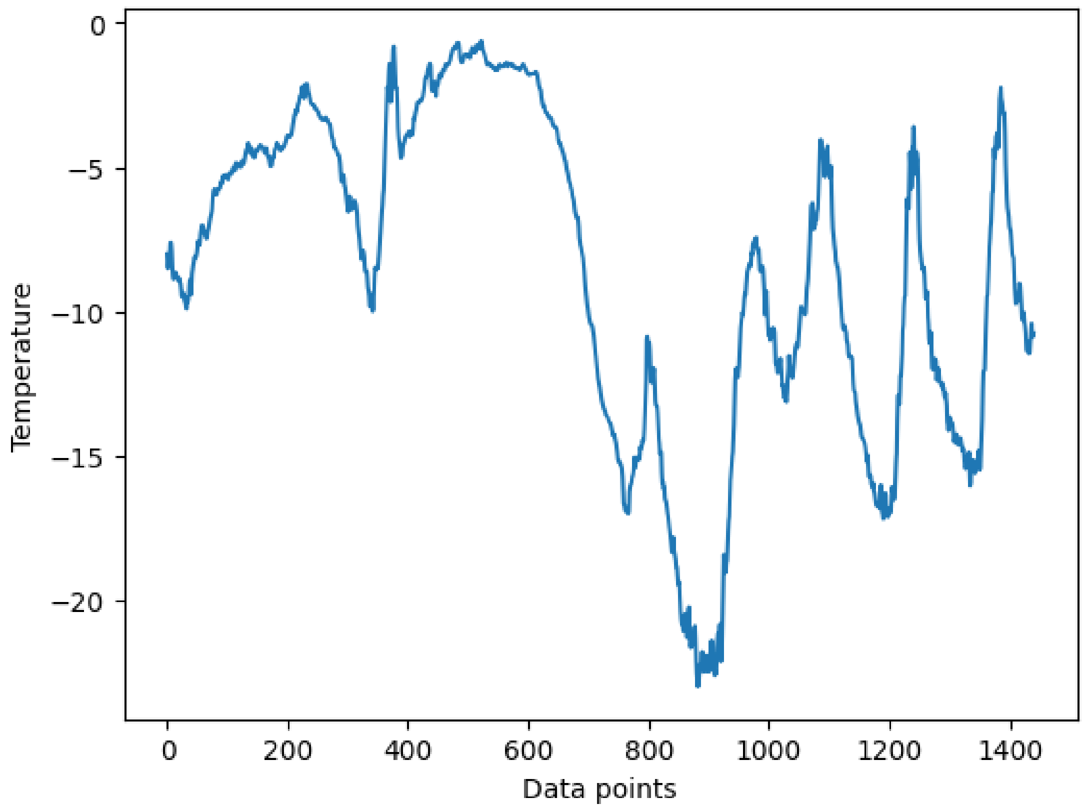
[6] : Text(0, 0.5, 'Temperature')



Plotting the first 10 days of the temperature timeseries- As given that one day data has 144 data points hence 10days will have 1440 data points

[7] :
```
plt.plot(range(1440), temperature[:1440])
plt.xlabel('Data points')
plt.ylabel('Temperature')
```

[7] : Text(0, 0.5, 'Temperature')



Computing the number of samples we'll use for each data split- 50% for Train, 25%-validation

[8] :
```
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("num_train_samples:", num_train_samples)
print("num_val_samples:", num_val_samples)
print("num_test_samples:", num_test_samples)
```

```
num_train_samples: 210225
num_val_samples: 105112
num_test_samples: 105114
```

1 Preparing the data

Normalizing the data- Since the data is already in a numerical format, vectorization is unnecessary. However, given that the data scales differ across variables, with temperature ranging from -20 to +30 and pressure measured in millibars, it is advisable to standardize all variables.

[9] :
```
mean = raw_data[:num_train_samples].mean(axis=0)
raw_data -= mean
std = raw_data[:num_train_samples].std(axis=0)
raw_data /= std
```

[11]:
```
import numpy as np
from tensorflow import keras
int_sequence = np.arange(10)
dummy_dataset = keras.utils.timeseries_dataset_from_array(
    data=int_sequence[:-3],
    targets=int_sequence[3:],
    sequence_length=3,
    batch_size=2,
)

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
```

Instantiating datasets for training, validation, and testing - it is required because the samples in the dataset are highly redundant Hence, it would be inefficient to allocate memory for each sample explicitly. Instead, we will generate the samples dynamically.

[12]:
```
sampling_rate = 6
sequence_length = 120
delay = sampling_rate * (sequence_length + 24 - 1)
batch_size = 256

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)

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)

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)
```

Inspecting the output of one of our datasets

[13]:
```
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,)
```

2 A common-sense, non-machine-learning baseline

Computing the common-sense baseline MAE - This defined function "evaluate_naive_method" provides a baseline for evaluating the performance of a simple forecasting approach, where the last value in the input sequence is used as a prediction for the next value.

[14]:
```
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

print(f"Validation MAE: {evaluate_naive_method(val_dataset):.2f}")
print(f"Test MAE: {evaluate_naive_method(test_dataset):.2f}")
```

```
Validation MAE: 2.44
Test MAE: 2.62
```

Common-sense baseline approach is to predict that the temperature 24 hours ahead will be identical to the current temperature. By using this straightforward baseline, the validation MAE (Mean Absolute Error) is 2.44 degrees Celsius, while the test MAE is 2.62 degrees Celsius. In other words, assuming that the temperature in the future remains the same as the current temperature would result in an average deviation of approximately two and a half degrees.

3 A basic machine-learning model - Dense Layer

Training and evaluating a densely connected model

[15]:
```
from tensorflow import keras
from tensorflow.keras import layers

inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.Flatten()(inputs)
x = layers.Dense(16, activation="relu")(x)
outputs = layers.Dense(1)(x)
model = keras.Model(inputs, outputs)
```

[16]:
```
callbacks = [
    keras.callbacks.ModelCheckpoint("jena_dense.keras",
                                    save_best_only=True)]
```

[17]:
```
model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
```

[19]:
```
history = model.fit(train_dataset, epochs=10,
                    validation_data = val_dataset, callbacks=callbacks)
```

```
Epoch 1/10
819/819 [==============================] - 49s 58ms/step - loss: 12.0966 - mae:
2.6877 - val_loss: 11.2644 - val_mae: 2.6574
Epoch 2/10
819/819 [==============================] - 48s 58ms/step - loss: 8.7591 - mae:
2.3288 - val_loss: 11.3583 - val_mae: 2.6713
Epoch 3/10
819/819 [==============================] - 57s 69ms/step - loss: 8.0575 - mae:
2.2348 - val_loss: 10.3659 - val_mae: 2.5448
Epoch 4/10
819/819 [==============================] - 55s 66ms/step - loss: 7.6087 - mae:
2.1739 - val_loss: 12.2755 - val_mae: 2.7762
Epoch 5/10
819/819 [==============================] - 52s 63ms/step - loss: 7.2928 - mae:
2.1289 - val_loss: 11.0961 - val_mae: 2.6300
Epoch 6/10
819/819 [==============================] - 49s 59ms/step - loss: 7.0350 - mae:
2.0931 - val_loss: 11.1941 - val_mae: 2.6441
Epoch 7/10
819/819 [==============================] - 49s 59ms/step - loss: 6.8404 - mae:
```

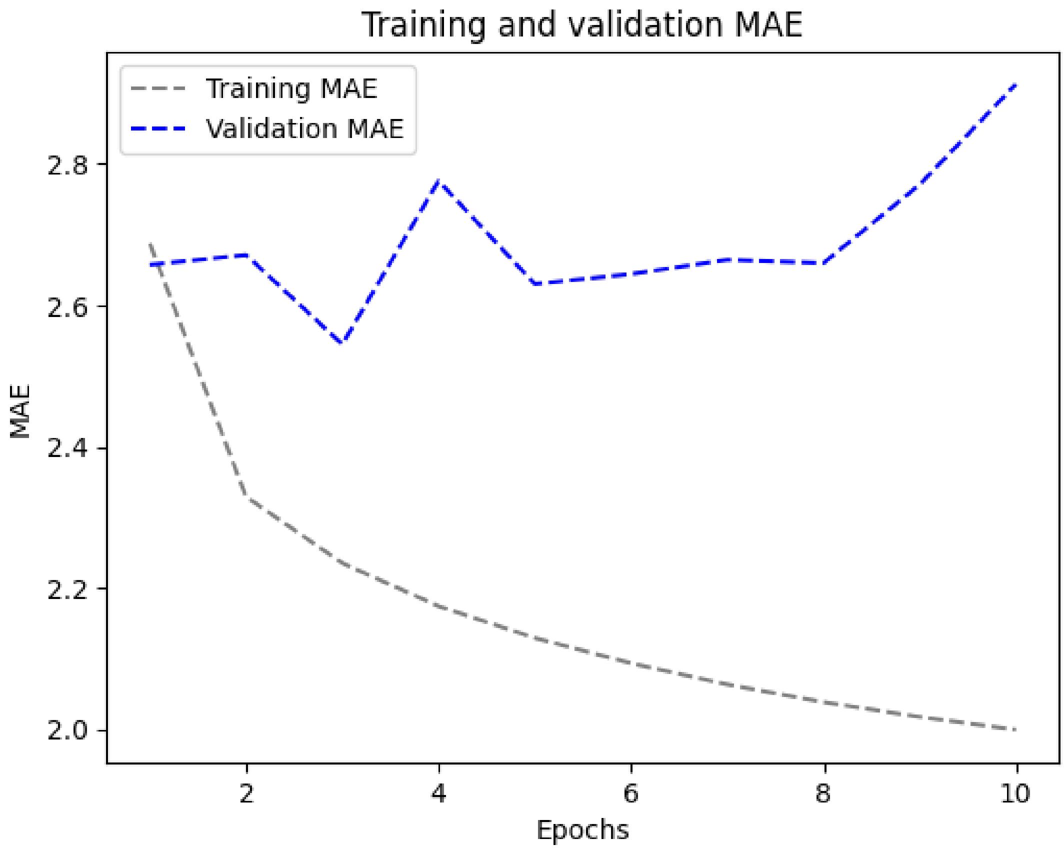
```
2.0629 - val_loss: 11.4156 - val_mae: 2.6644
Epoch 8/10
819/819 [==============================] - 50s 61ms/step - loss: 6.6839 - mae:
2.0381 - val_loss: 11.3781 - val_mae: 2.6597
Epoch 9/10
819/819 [==============================] - 58s 70ms/step - loss: 6.5367 - mae:
2.0170 - val_loss: 12.3677 - val_mae: 2.7709
Epoch 10/10
819/819 [==============================] - 50s 61ms/step - loss: 6.4264 - mae:
1.9990 - val_loss: 13.5889 - val_mae: 2.9130
```

[20]:
```python
model = keras.models.load_model("jena_dense.keras")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
405/405 [==============================] - 16s 39ms/step - loss: 11.4217 - mae:
2.6437
Test MAE: 2.64
```

Plotting results

[21]:
```python
import matplotlib.pyplot as plt
loss = history.history["mae"]
val_loss = history.history["val_mae"]

epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, color="grey", linestyle="dashed", label="Training MAE")
plt.plot(epochs, val_loss, color="blue",linestyle="dashed", label="Validation MAE")
plt.title("Training and validation MAE")
plt.xlabel("Epochs")
plt.ylabel("MAE")
plt.legend()
plt.show()
```



4 Let's try a 1D convolutional model

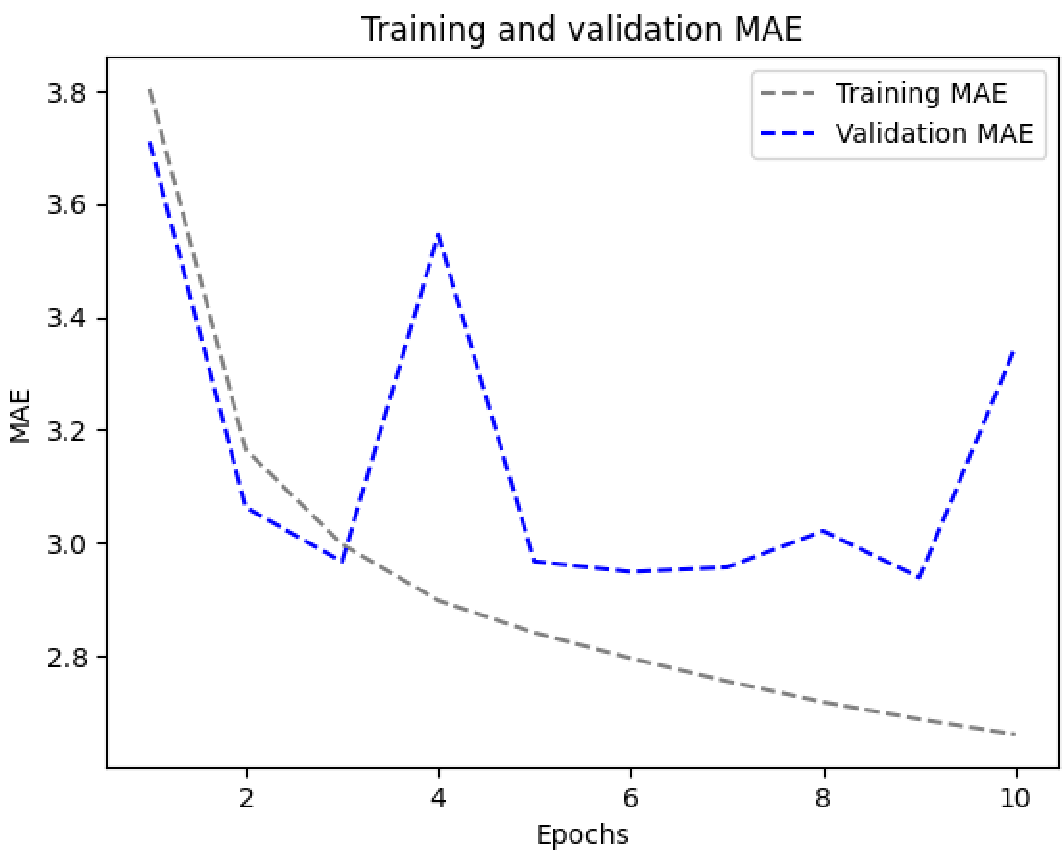
[22]:
```
inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.Conv1D(8, 24, activation="relu")(inputs)
x = layers.MaxPooling1D(2)(x)
x = layers.Conv1D(8, 12, activation="relu")(x)
x = layers.MaxPooling1D(2)(x)
x = layers.Conv1D(8, 6, activation="relu")(x)
x = layers.GlobalAveragePooling1D()(x)
outputs = layers.Dense(1)(x)
model = keras.Model(inputs, outputs)

callbacks = [
    keras.callbacks.ModelCheckpoint("jena_conv.keras",
                                    save_best_only=True)
]
model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
history = model.fit(train_dataset,
                    epochs=10,
```

```
                           validation_data=val_dataset,
                           callbacks=callbacks)

model = keras.models.load_model("jena_conv.keras")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
Epoch 1/10
819/819 [==============================] - 86s 103ms/step - loss: 23.7746 - mae:
3.8034 - val_loss: 22.4870 - val_mae: 3.7098
Epoch 2/10
819/819 [==============================] - 82s 99ms/step - loss: 15.8125 - mae:
3.1647 - val_loss: 14.8130 - val_mae: 3.0623
Epoch 3/10
819/819 [==============================] - 83s 101ms/step - loss: 14.2417 - mae:
2.9972 - val_loss: 14.3072 - val_mae: 2.9666
Epoch 4/10
819/819 [==============================] - 88s 107ms/step - loss: 13.3576 - mae:
2.8977 - val_loss: 20.1665 - val_mae: 3.5454
Epoch 5/10
819/819 [==============================] - 89s 108ms/step - loss: 12.8322 - mae:
2.8407 - val_loss: 14.2670 - val_mae: 2.9668
Epoch 6/10
819/819 [==============================] - 89s 109ms/step - loss: 12.4519 - mae:
2.7954 - val_loss: 14.0571 - val_mae: 2.9485
Epoch 7/10
819/819 [==============================] - 123s 150ms/step - loss: 12.0873 -
mae: 2.7547 - val_loss: 13.9501 - val_mae: 2.9565
Epoch 8/10
819/819 [==============================] - 95s 115ms/step - loss: 11.7606 - mae:
2.7177 - val_loss: 14.8918 - val_mae: 3.0209
Epoch 9/10
819/819 [==============================] - 88s 107ms/step - loss: 11.5121 - mae:
2.6875 - val_loss: 13.8599 - val_mae: 2.9384
Epoch 10/10
819/819 [==============================] - 86s 105ms/step - loss: 11.2766 - mae:
2.6607 - val_loss: 18.1669 - val_mae: 3.3476
405/405 [==============================] - 21s 49ms/step - loss: 14.8259 - mae:
3.0379
Test MAE: 3.04
```

[23]:
```
import matplotlib.pyplot as plt
loss = history.history["mae"]
val_loss = history.history["val_mae"]

epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, color="grey", linestyle="dashed", label="Training MAE")
```

```
plt.plot(epochs,    val_loss,  color="blue",linestyle="dashed",  label="Validation␣
↪MAE")
plt.title("Training and validation MAE")
plt.xlabel("Epochs")
plt.ylabel("MAE")
plt.legend()
plt.show()
```



It seem that the convolutional data perform worse compared to common sense or dense model. it could be because

- The assumption of translation invariance does not hold well for weather data.
- The order of the data is crucial. Recent past data is significantly more informative for predicting the temperature of the following day compared to data from several days ago. Unfortunately, a 1D convolutional neural network is unable to effectively capture this critical temporal order.

5 A Simple RNN

1.An RNN layer that can process sequences of any length

[24]:
```python
num_features = 14
inputs = keras.Input(shape=(None, num_features))
outputs = layers.SimpleRNN(16)(inputs)

model = keras.Model(inputs, outputs)

callbacks = [
    keras.callbacks.ModelCheckpoint("jena_SimRNN.keras",
                                    save_best_only=True)
]
model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
history = model.fit(train_dataset,
                    epochs=10,
                    validation_data=val_dataset,
                    callbacks=callbacks)

model = keras.models.load_model("jena_SimRNN.keras")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
Epoch 1/10
819/819 [==============================] - 77s 93ms/step - loss: 138.7432 - mae:
9.6939 - val_loss: 144.0067 - val_mae: 9.9023
Epoch 2/10
819/819 [==============================] - 75s 91ms/step - loss: 136.3165 - mae:
9.5548 - val_loss: 143.8013 - val_mae: 9.8819
Epoch 3/10
819/819 [==============================] - 78s 94ms/step - loss: 136.2390 - mae:
9.5478 - val_loss: 143.6788 - val_mae: 9.8670
Epoch 4/10
819/819 [==============================] - 74s 90ms/step - loss: 136.2022 - mae:
9.5432 - val_loss: 143.6592 - val_mae: 9.8659
Epoch 5/10
819/819 [==============================] - 73s 88ms/step - loss: 136.1734 - mae:
9.5386 - val_loss: 143.5966 - val_mae: 9.8574
Epoch 6/10
819/819 [==============================] - 75s 91ms/step - loss: 136.1472 - mae:
9.5352 - val_loss: 143.5659 - val_mae: 9.8513
Epoch 7/10
819/819 [==============================] - 73s 89ms/step - loss: 136.1489 - mae:
9.5348 - val_loss: 143.5657 - val_mae: 9.8522
Epoch 8/10
819/819 [==============================] - 69s 84ms/step - loss: 136.1194 - mae:
9.5329 - val_loss: 143.5380 - val_mae: 9.8501
Epoch 9/10
819/819 [==============================] - 73s 89ms/step - loss: 136.2053 - mae:
9.5411 - val_loss: 143.7877 - val_mae: 9.8736
Epoch 10/10
```

```
819/819 [==============================] - 72s 87ms/step - loss: 136.2590 - mae:
9.5465 - val_loss: 143.5481 - val_mae: 9.8515
405/405 [==============================] - 19s 45ms/step - loss: 151.2728 - mae:
9.9161
Test MAE: 9.92
```

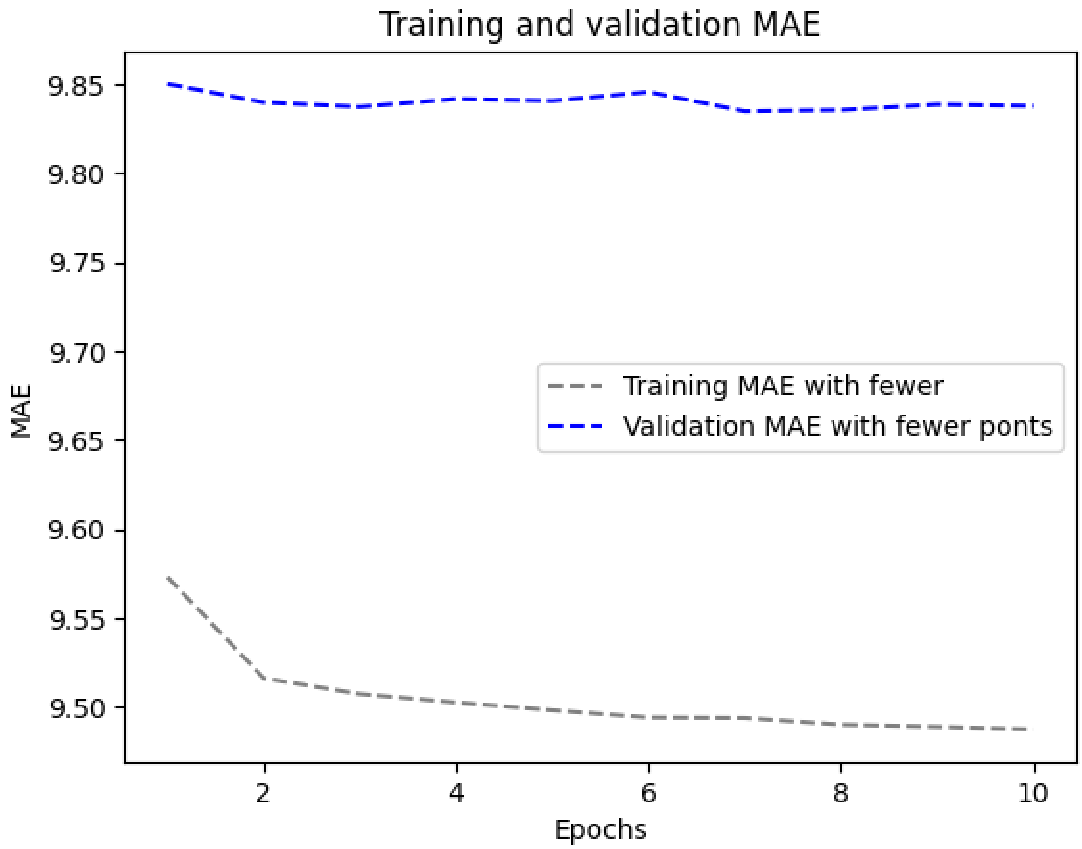
6 2.Simple RNN - Stacking RNN layers

[25]:
```python
num_features = 14
steps = 120
inputs = keras.Input(shape=(steps, num_features))
x = layers.SimpleRNN(16, return_sequences=True)(inputs)
x = layers.SimpleRNN(16, return_sequences=True)(x)
outputs = layers.SimpleRNN(16)(x)
model = keras.Model(inputs, outputs)

callbacks = [
    keras.callbacks.ModelCheckpoint("jena_SRNN2.keras",
                                    save_best_only=True)
]
model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
history = model.fit(train_dataset,
                    epochs=10,
                    validation_data=val_dataset,
                    callbacks=callbacks)

model = keras.models.load_model("jena_SRNN2.keras")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
Epoch 1/10
819/819 [==============================] - 135s 161ms/step - loss: 136.9767 -
mae: 9.5732 - val_loss: 143.5129 - val_mae: 9.8500
Epoch 2/10
819/819 [==============================] - 130s 159ms/step - loss: 135.9816 -
mae: 9.5164 - val_loss: 143.4397 - val_mae: 9.8397
Epoch 3/10
819/819 [==============================] - 127s 155ms/step - loss: 135.9150 -
mae: 9.5075 - val_loss: 143.4221 - val_mae: 9.8372
Epoch 4/10
819/819 [==============================] - 128s 155ms/step - loss: 135.8847 -
mae: 9.5028 - val_loss: 143.4524 - val_mae: 9.8416
Epoch 5/10
819/819 [==============================] - 145s 177ms/step - loss: 135.8538 -
mae: 9.4984 - val_loss: 143.4391 - val_mae: 9.8406
Epoch 6/10
819/819 [==============================] - 143s 175ms/step - loss: 135.8293 -
mae: 9.4944 - val_loss: 143.4548 - val_mae: 9.8455
```

```
Epoch 7/10
819/819 [==============================] - 128s 156ms/step - loss: 135.8316 -
mae: 9.4941 - val_loss: 143.4133 - val_mae: 9.8347
Epoch 8/10
819/819 [==============================] - 128s 155ms/step - loss: 135.8039 -
mae: 9.4903 - val_loss: 143.4056 - val_mae: 9.8355
Epoch 9/10
819/819 [==============================] - 128s 155ms/step - loss: 135.7974 -
mae: 9.4892 - val_loss: 143.4248 - val_mae: 9.8385
Epoch 10/10
819/819 [==============================] - 143s 174ms/step - loss: 135.7887 -
mae: 9.4877 - val_loss: 143.4221 - val_mae: 9.8377
405/405 [==============================] - 28s 66ms/step - loss: 151.1893 - mae:
9.9191
Test MAE: 9.92
```

[27]:
```
import matplotlib.pyplot as plt
loss = history.history["mae"]
val_loss = history.history["val_mae"]

epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, color="grey", linestyle="dashed", label="Training MAE␣
↪with fewer")
plt.plot(epochs, val_loss, color="blue",linestyle="dashed", label="Validation␣
↪MAE with fewer ponts")
plt.title("Training and validation MAE")
plt.xlabel("Epochs")
plt.ylabel("MAE")
plt.legend()
plt.show()
```



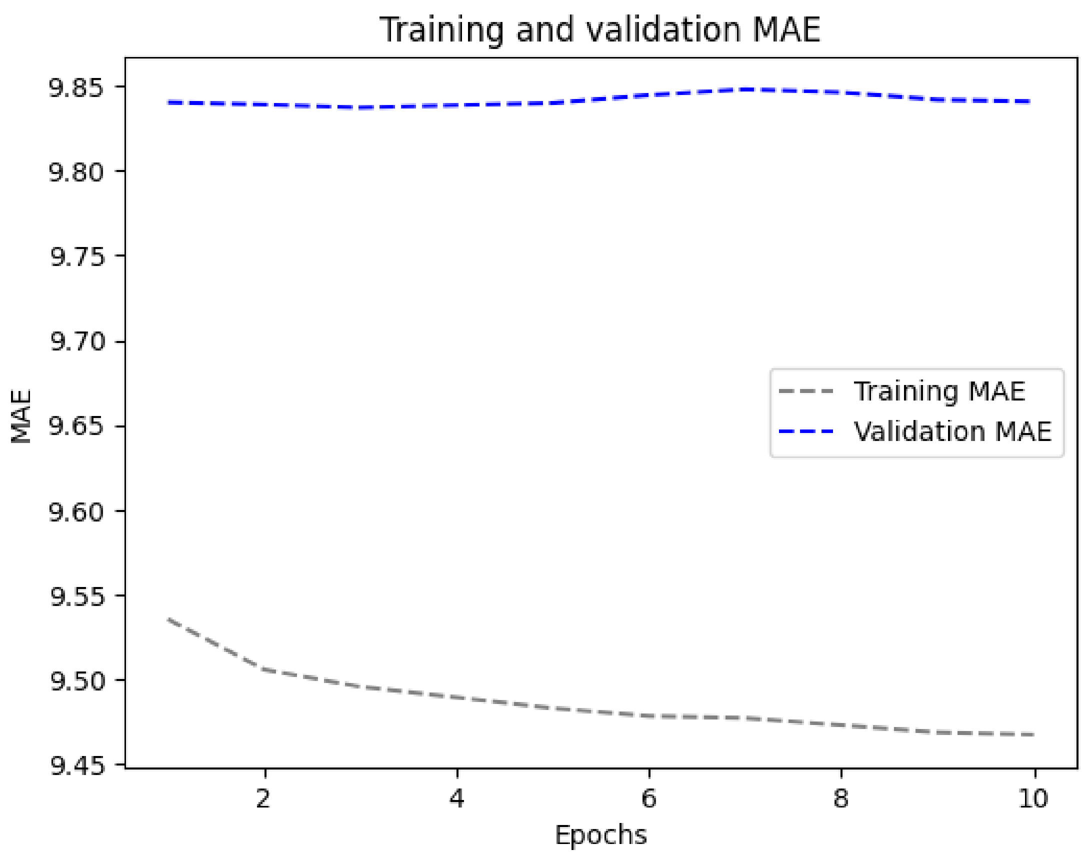
[28]:
```
num_features = 14
steps = 120
inputs = keras.Input(shape=(steps, num_features))
x = layers.SimpleRNN(64, return_sequences=True)(inputs)
x = layers.SimpleRNN(64, return_sequences=True)(x)
outputs = layers.SimpleRNN(64)(x)
model = keras.Model(inputs, outputs)

callbacks = [
    keras.callbacks.ModelCheckpoint("jena_SRNN2.keras",
                                    save_best_only=True)
]
model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
history = model.fit(train_dataset,
                    epochs=10,
                    validation_data=val_dataset,
                    callbacks=callbacks)

model = keras.models.load_model("jena_SRNN2.keras")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
Epoch 1/10
819/819 [==============================] - 284s 344ms/step - loss: 136.2690 -
mae: 9.5354 - val_loss: 143.4331 - val_mae: 9.8405
Epoch 2/10
819/819 [==============================] - 239s 292ms/step - loss: 135.9033 -
mae: 9.5057 - val_loss: 143.4382 - val_mae: 9.8391
Epoch 3/10
819/819 [==============================] - 280s 341ms/step - loss: 135.8371 -
mae: 9.4955 - val_loss: 143.4353 - val_mae: 9.8373
Epoch 4/10
819/819 [==============================] - 281s 343ms/step - loss: 135.7985 -
mae: 9.4893 - val_loss: 143.4571 - val_mae: 9.8387
Epoch 5/10
819/819 [==============================] - 281s 342ms/step - loss: 135.7591 -
mae: 9.4829 - val_loss: 143.4392 - val_mae: 9.8401
Epoch 6/10
819/819 [==============================] - 278s 339ms/step - loss: 135.7342 -
mae: 9.4783 - val_loss: 143.4498 - val_mae: 9.8446
Epoch 7/10
819/819 [==============================] - 279s 340ms/step - loss: 135.7322 -
mae: 9.4771 - val_loss: 143.4912 - val_mae: 9.8481
Epoch 8/10
819/819 [==============================] - 282s 344ms/step - loss: 135.7082 -
mae: 9.4729 - val_loss: 143.4852 - val_mae: 9.8463
Epoch 9/10
819/819 [==============================] - 281s 342ms/step - loss: 135.6851 -
mae: 9.4687 - val_loss: 143.4914 - val_mae: 9.8420
Epoch 10/10
819/819 [==============================] - 282s 344ms/step - loss: 135.6797 -
mae: 9.4673 - val_loss: 143.4500 - val_mae: 9.8409
405/405 [==============================] - 51s 121ms/step - loss: 151.1381 -
mae: 9.9063
Test MAE: 9.91
```

[29]:

```python
import matplotlib.pyplot as plt
loss = history.history["mae"]
val_loss = history.history["val_mae"]

epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, color="grey", linestyle="dashed", label="Training MAE")
plt.plot(epochs, val_loss, color="blue",linestyle="dashed", label="Validation MAE")
plt.title("Training and validation MAE")
plt.xlabel("Epochs")
plt.ylabel("MAE")
plt.legend()
```

```
plt.show()
```



[30]:

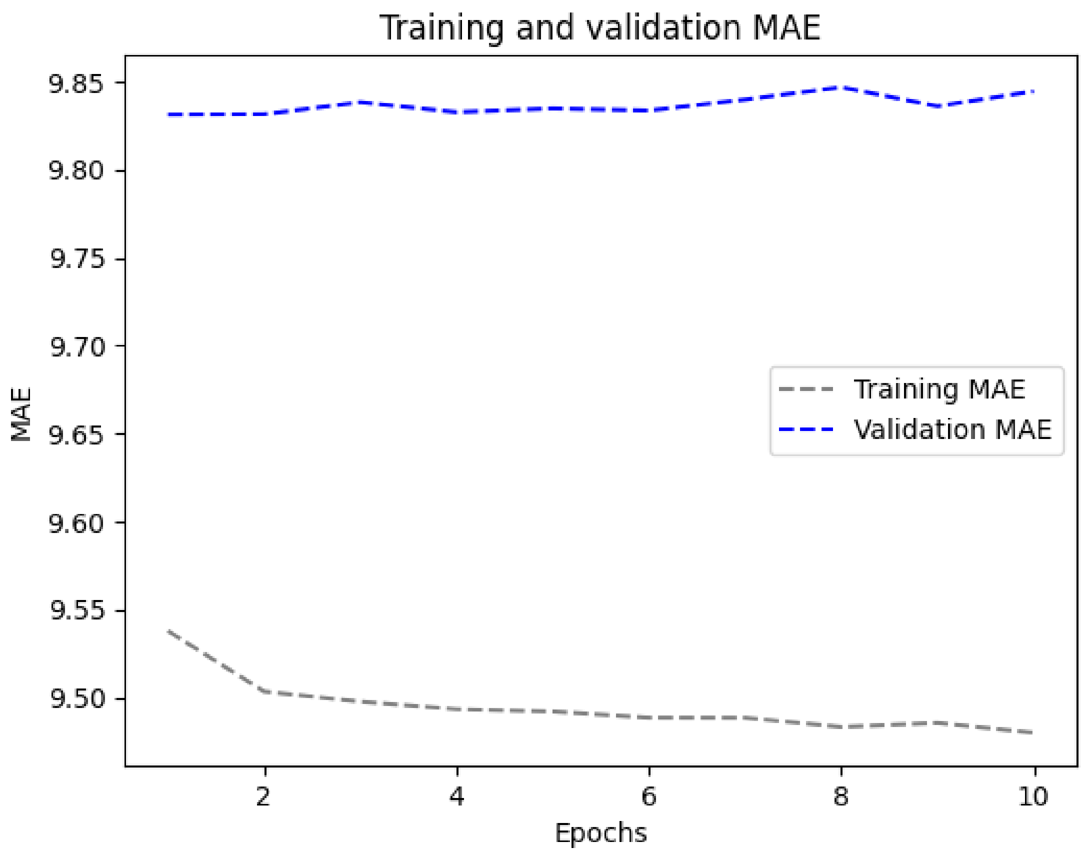
```
num_features = 14
steps = 120
inputs = keras.Input(shape=(steps, num_features))
x = layers.SimpleRNN(32, return_sequences=True)(inputs)
x = layers.SimpleRNN(32, return_sequences=True)(x)
outputs = layers.SimpleRNN(32)(x)
model = keras.Model(inputs, outputs)

callbacks = [
    keras.callbacks.ModelCheckpoint("jena_SRNN2.keras",
                                    save_best_only=True)
]
model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
history = model.fit(train_dataset,
                    epochs=10,
                    validation_data=val_dataset,
                    callbacks=callbacks)
```

```
model = keras.models.load_model("jena_SRNN2.keras")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
Epoch 1/10
819/819 [==============================] - 174s 209ms/step - loss: 136.3739 -
mae: 9.5378 - val_loss: 143.3804 - val_mae: 9.8312
Epoch 2/10
819/819 [==============================] - 157s 191ms/step - loss: 135.8933 -
mae: 9.5033 - val_loss: 143.3900 - val_mae: 9.8314
Epoch 3/10
819/819 [==============================] - 164s 200ms/step - loss: 135.8582 -
mae: 9.4976 - val_loss: 143.4469 - val_mae: 9.8382
Epoch 4/10
819/819 [==============================] - 173s 211ms/step - loss: 135.8336 -
mae: 9.4933 - val_loss: 143.4016 - val_mae: 9.8326
Epoch 5/10
819/819 [==============================] - 168s 204ms/step - loss: 135.8266 -
mae: 9.4919 - val_loss: 143.4083 - val_mae: 9.8347
Epoch 6/10
819/819 [==============================] - 163s 199ms/step - loss: 135.8010 -
mae: 9.4885 - val_loss: 143.4120 - val_mae: 9.8335
Epoch 7/10
819/819 [==============================] - 169s 205ms/step - loss: 135.7984 -
mae: 9.4884 - val_loss: 143.4360 - val_mae: 9.8397
Epoch 8/10
819/819 [==============================] - 157s 192ms/step - loss: 135.7697 -
mae: 9.4832 - val_loss: 143.4848 - val_mae: 9.8467
Epoch 9/10
819/819 [==============================] - 156s 190ms/step - loss: 135.7840 -
mae: 9.4854 - val_loss: 143.4220 - val_mae: 9.8360
Epoch 10/10
819/819 [==============================] - 156s 190ms/step - loss: 135.7507 -
mae: 9.4800 - val_loss: 143.4832 - val_mae: 9.8444
405/405 [==============================] - 34s 81ms/step - loss: 151.0980 - mae:
9.8999
Test MAE: 9.90
```

[31]:
```
import matplotlib.pyplot as plt
loss = history.history["mae"]
val_loss = history.history["val_mae"]

epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, color="grey", linestyle="dashed", label="Training MAE")
plt.plot(epochs, val_loss, color="blue",linestyle="dashed", label="Validation MAE")
plt.title("Training and validation MAE")
```

```
plt.xlabel("Epochs")
plt.ylabel("MAE")
plt.legend()
plt.show()
```



7 A Simple GRU (Gated Recurrent Unit)

[32]:
```
inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.GRU(16)(inputs)
outputs = layers.Dense(1)(x)
model = keras.Model(inputs, outputs)

callbacks = [
    keras.callbacks.ModelCheckpoint("jena_gru.keras",
                                    save_best_only=True)
]
model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
history = model.fit(train_dataset,
                    epochs=10,
```

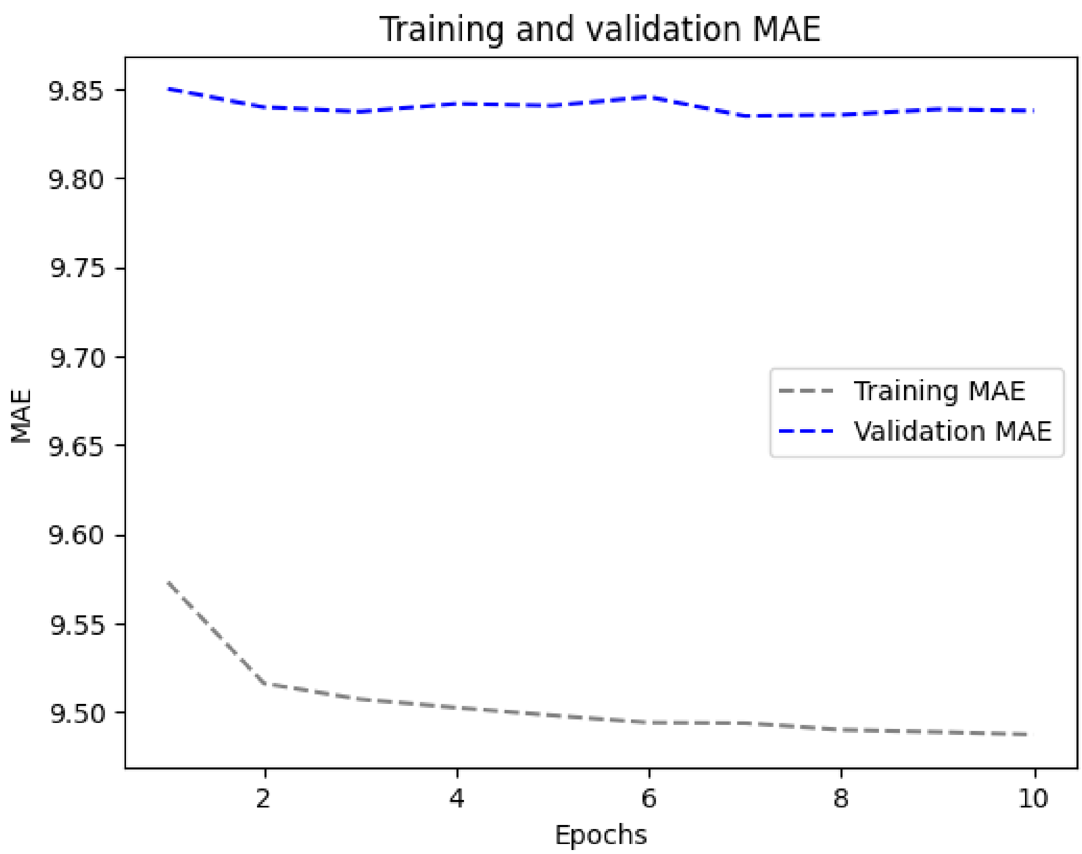
```
                        validation_data=val_dataset,
                        callbacks=callbacks)

model = keras.models.load_model("jena_gru.keras")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
Epoch 1/10
819/819 [==============================] - 126s 151ms/step - loss: 39.2510 -
mae: 4.5325 - val_loss: 12.4765 - val_mae: 2.6730
Epoch 2/10
819/819 [==============================] - 104s 127ms/step - loss: 10.6581 -
mae: 2.5413 - val_loss: 10.5723 - val_mae: 2.4745
Epoch 3/10
819/819 [==============================] - 107s 130ms/step - loss: 9.6812 - mae:
2.4259 - val_loss: 9.8932 - val_mae: 2.4024
Epoch 4/10
819/819 [==============================] - 125s 151ms/step - loss: 9.2699 - mae:
2.3766 - val_loss: 10.0421 - val_mae: 2.4086
Epoch 5/10
819/819 [==============================] - 126s 153ms/step - loss: 8.9985 - mae:
2.3427 - val_loss: 9.7026 - val_mae: 2.3785
Epoch 6/10
819/819 [==============================] - 127s 154ms/step - loss: 8.7886 - mae:
2.3143 - val_loss: 9.8412 - val_mae: 2.4059
Epoch 7/10
819/819 [==============================] - 106s 129ms/step - loss: 8.6114 - mae:
2.2907 - val_loss: 10.6256 - val_mae: 2.4738
Epoch 8/10
819/819 [==============================] - 105s 128ms/step - loss: 8.4532 - mae:
2.2691 - val_loss: 9.5922 - val_mae: 2.3902
Epoch 9/10
819/819 [==============================] - 106s 129ms/step - loss: 8.3134 - mae:
2.2501 - val_loss: 10.5957 - val_mae: 2.4764
Epoch 10/10
819/819 [==============================] - 108s 132ms/step - loss: 8.1802 - mae:
2.2343 - val_loss: 9.8451 - val_mae: 2.4226
405/405 [==============================] - 27s 63ms/step - loss: 9.9697 - mae:
2.4911
Test MAE: 2.49
```

[26]:

```
import matplotlib.pyplot as plt
loss = history.history["mae"]
val_loss = history.history["val_mae"]

epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, color="grey", linestyle="dashed", label="Training MAE")
```

```
plt.plot(epochs,    val_loss,  color="blue",linestyle="dashed",  label="Validation␣
↪MAE")
plt.title("Training and validation MAE")
plt.xlabel("Epochs")
plt.ylabel("MAE")
plt.legend()
plt.show()
```



8 LSTM(Long Short-Term Memory)

1. LSTM-Simple

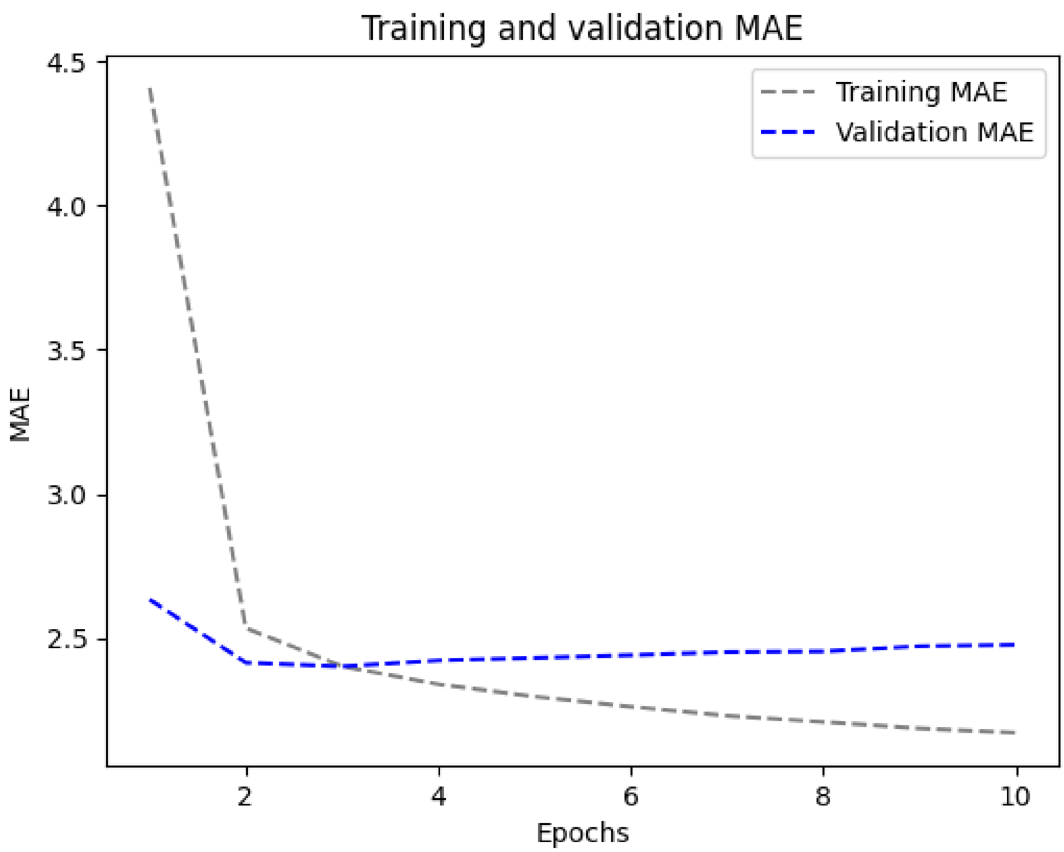
[33]:
```
inputs  =  keras.Input(shape=(sequence_length,  raw_data.shape[-1]))
x  =  layers.LSTM(16)(inputs)
outputs  =  layers.Dense(1)(x)
model  =  keras.Model(inputs,  outputs)

callbacks  =  [
    keras.callbacks.ModelCheckpoint("jena_lstm.keras",
```

```
                                                save_best_only=True)
]
model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
history = model.fit(train_dataset,
                    epochs=10,
                    validation_data=val_dataset,
                    callbacks=callbacks)

model = keras.models.load_model("jena_lstm.keras")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
Epoch 1/10
819/819 [==============================] - 126s 150ms/step - loss: 36.9219 -
mae: 4.4073 - val_loss: 11.7273 - val_mae: 2.6338
Epoch 2/10
819/819 [==============================] - 123s 150ms/step - loss: 10.6663 -
mae: 2.5346 - val_loss: 9.5305 - val_mae: 2.4149
Epoch 3/10
819/819 [==============================] - 123s 150ms/step - loss: 9.5357 - mae:
2.4025 - val_loss: 9.5538 - val_mae: 2.4018
Epoch 4/10
819/819 [==============================] - 105s 128ms/step - loss: 9.0070 - mae:
2.3401 - val_loss: 9.6338 - val_mae: 2.4227
Epoch 5/10
819/819 [==============================] - 105s 128ms/step - loss: 8.6483 - mae:
2.2976 - val_loss: 9.6704 - val_mae: 2.4317
Epoch 6/10
819/819 [==============================] - 125s 152ms/step - loss: 8.3853 - mae:
2.2625 - val_loss: 9.7288 - val_mae: 2.4412
Epoch 7/10
819/819 [==============================] - 105s 128ms/step - loss: 8.1479 - mae:
2.2311 - val_loss: 9.7713 - val_mae: 2.4514
Epoch 8/10
819/819 [==============================] - 104s 126ms/step - loss: 7.9839 - mae:
2.2095 - val_loss: 9.8108 - val_mae: 2.4536
Epoch 9/10
819/819 [==============================] - 122s 149ms/step - loss: 7.8202 - mae:
2.1874 - val_loss: 9.9233 - val_mae: 2.4719
Epoch 10/10
819/819 [==============================] - 131s 160ms/step - loss: 7.7210 - mae:
2.1724 - val_loss: 9.9470 - val_mae: 2.4769
405/405 [==============================] - 28s 65ms/step - loss: 10.9701 - mae:
2.5842
Test MAE: 2.58
```

[34]:
```
import matplotlib.pyplot as plt
loss = history.history["mae"]
```

```
val_loss  =  history.history["val_mae"]

epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, color="grey", linestyle="dashed", label="Training MAE")
plt.plot(epochs,   val_loss, color="blue",linestyle="dashed", label="Validation␣
↪MAE")
plt.title("Training and validation MAE")
plt.xlabel("Epochs")
plt.ylabel("MAE")
plt.legend()
plt.show()
```



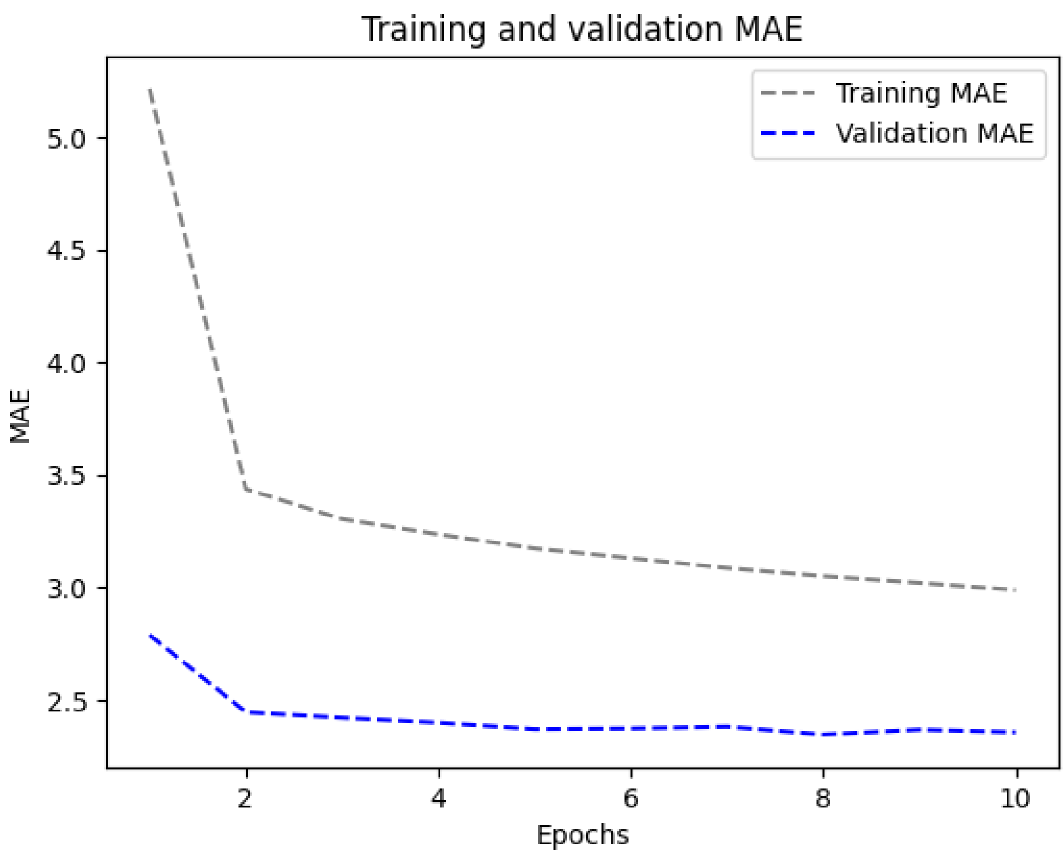
2. LSTM-Dropout Regularization

[35]:
```
inputs  =  keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x  =  layers.LSTM(16, recurrent_dropout=0.25)(inputs)
x  =  layers.Dropout(0.5)(x)
outputs  =  layers.Dense(1)(x)
model  =  keras.Model(inputs, outputs)
```

```
callbacks = [
    keras.callbacks.ModelCheckpoint("jena_lstm_dropout.keras",
                                    save_best_only=True)
]
model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
history = model.fit(train_dataset,
                    epochs=10,
                    validation_data=val_dataset,
                    callbacks=callbacks)

model = keras.models.load_model("jena_lstm_dropout.keras")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
Epoch 1/10
819/819 [==============================] - 192s 229ms/step - loss: 49.0892 -
mae: 5.2156 - val_loss: 13.5326 - val_mae: 2.7879
Epoch 2/10
819/819 [==============================] - 188s 230ms/step - loss: 20.0311 -
mae: 3.4372 - val_loss: 9.8801 - val_mae: 2.4453
Epoch 3/10
819/819 [==============================] - 171s 209ms/step - loss: 18.4131 -
mae: 3.3032 - val_loss: 9.6463 - val_mae: 2.4205
Epoch 4/10
819/819 [==============================] - 170s 206ms/step - loss: 17.6769 -
mae: 3.2357 - val_loss: 9.4178 - val_mae: 2.3984
Epoch 5/10
819/819 [==============================] - 171s 208ms/step - loss: 16.9887 -
mae: 3.1730 - val_loss: 9.1855 - val_mae: 2.3698
Epoch 6/10
819/819 [==============================] - 167s 204ms/step - loss: 16.5439 -
mae: 3.1298 - val_loss: 9.2100 - val_mae: 2.3728
Epoch 7/10
819/819 [==============================] - 185s 226ms/step - loss: 16.0833 -
mae: 3.0856 - val_loss: 9.3072 - val_mae: 2.3809
Epoch 8/10
819/819 [==============================] - 173s 210ms/step - loss: 15.6756 -
mae: 3.0498 - val_loss: 9.0421 - val_mae: 2.3458
Epoch 9/10
819/819 [==============================] - 186s 226ms/step - loss: 15.3861 -
mae: 3.0201 - val_loss: 9.2331 - val_mae: 2.3676
Epoch 10/10
819/819 [==============================] - 184s 224ms/step - loss: 15.0624 -
mae: 2.9883 - val_loss: 9.1305 - val_mae: 2.3564
405/405 [==============================] - 30s 70ms/step - loss: 10.5570 - mae:
2.5531
Test MAE: 2.55
```

[36]:
```python
import matplotlib.pyplot as plt
loss = history.history["mae"]
val_loss = history.history["val_mae"]

epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, color="grey", linestyle="dashed", label="Training MAE")
plt.plot(epochs, val_loss, color="blue",linestyle="dashed", label="Validation MAE")
plt.title("Training and validation MAE")
plt.xlabel("Epochs")
plt.ylabel("MAE")
plt.legend()
plt.show()
```



3. LSTM - Stacked setup with 16 units

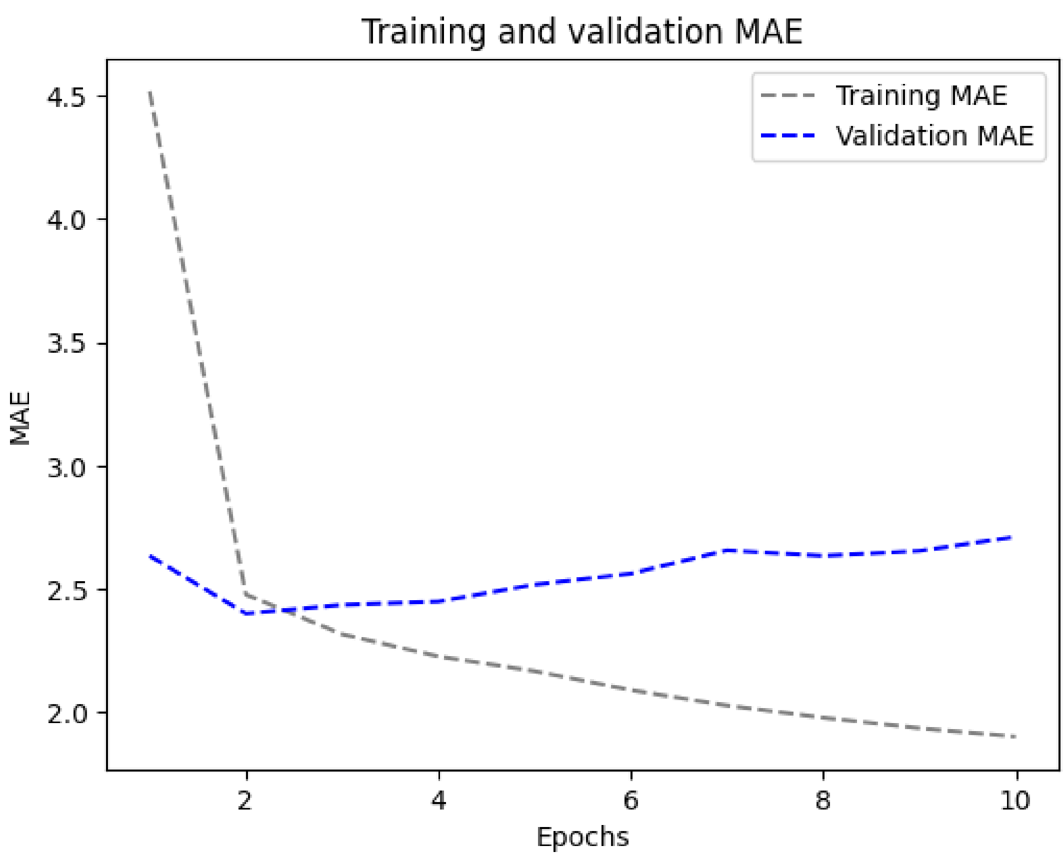
[37]:
```python
inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.LSTM(16, return_sequences=True)(inputs)
x = layers.LSTM(16)(x)
```

```python
outputs = layers.Dense(1)(x)
model = keras.Model(inputs, outputs)

callbacks = [
    keras.callbacks.ModelCheckpoint("jena_LSTM_stacked1.keras",
                                    save_best_only=True)
]
model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
history = model.fit(train_dataset,
                    epochs=10,
                    validation_data=val_dataset,
                    callbacks=callbacks)
model = keras.models.load_model("jena_LSTM_stacked1.keras")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
Epoch 1/10
819/819 [==============================] - 189s 225ms/step - loss: 38.6083 -
mae: 4.5170 - val_loss: 12.0450 - val_mae: 2.6342
Epoch 2/10
819/819 [==============================] - 190s 232ms/step - loss: 10.2072 -
mae: 2.4774 - val_loss: 9.5883 - val_mae: 2.4002
Epoch 3/10
819/819 [==============================] - 183s 223ms/step - loss: 8.8359 - mae:
2.3167 - val_loss: 9.6811 - val_mae: 2.4353
Epoch 4/10
819/819 [==============================] - 182s 222ms/step - loss: 8.1514 - mae:
2.2266 - val_loss: 9.7583 - val_mae: 2.4488
Epoch 5/10
819/819 [==============================] - 185s 225ms/step - loss: 7.7511 - mae:
2.1672 - val_loss: 10.3997 - val_mae: 2.5176
Epoch 6/10
819/819 [==============================] - 196s 239ms/step - loss: 7.2186 - mae:
2.0904 - val_loss: 10.7279 - val_mae: 2.5622
Epoch 7/10
819/819 [==============================] - 192s 234ms/step - loss: 6.7921 - mae:
2.0277 - val_loss: 11.4845 - val_mae: 2.6566
Epoch 8/10
819/819 [==============================] - 176s 214ms/step - loss: 6.4591 - mae:
1.9784 - val_loss: 11.2765 - val_mae: 2.6341
Epoch 9/10
819/819 [==============================] - 180s 220ms/step - loss: 6.1614 - mae:
1.9356 - val_loss: 11.6302 - val_mae: 2.6544
Epoch 10/10
819/819 [==============================] - 184s 225ms/step - loss: 5.9391 - mae:
1.9017 - val_loss: 12.0840 - val_mae: 2.7111
405/405 [==============================] - 39s 91ms/step - loss: 10.4460 - mae:
2.5091
```

Test MAE: 2.51

[38]:
```python
import matplotlib.pyplot as plt
loss = history.history["mae"]
val_loss = history.history["val_mae"]

epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, color="grey", linestyle="dashed", label="Training MAE")
plt.plot(epochs, val_loss, color="blue",linestyle="dashed", label="Validation MAE")
plt.title("Training and validation MAE")
plt.xlabel("Epochs")
plt.ylabel("MAE")
plt.legend()
plt.show()
```



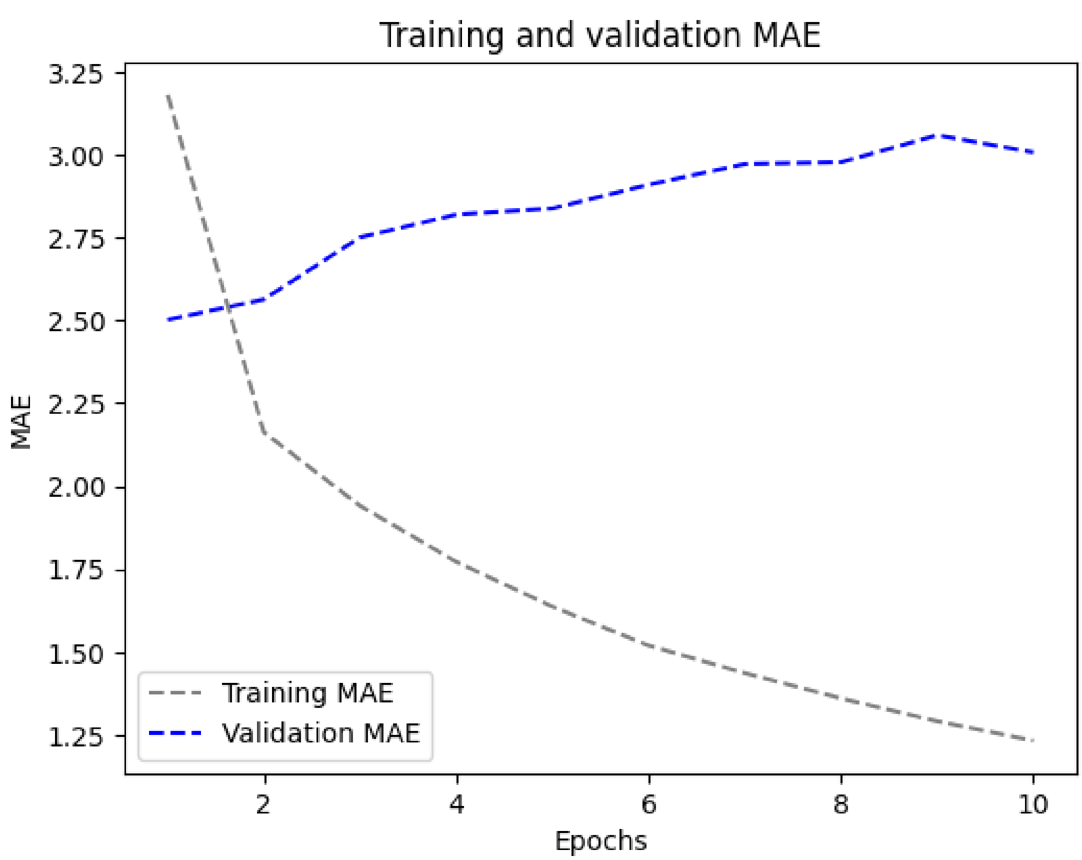
4. LSTM - Stacked setup with 32 units

[39]:
```python
inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.LSTM(32, return_sequences=True)(inputs)
x = layers.LSTM(32)(x)
outputs = layers.Dense(1)(x)
model = keras.Model(inputs, outputs)

callbacks = [
    keras.callbacks.ModelCheckpoint("jena_LSTM_stacked2.keras",
                                    save_best_only=True)
]
model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
history = model.fit(train_dataset,
                    epochs=10,
                    validation_data=val_dataset,
                    callbacks=callbacks)
model = keras.models.load_model("jena_LSTM_stacked2.keras")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
Epoch 1/10
819/819 [==============================] - 288s 346ms/step - loss: 19.6578 -
mae: 3.1808 - val_loss: 10.2804 - val_mae: 2.5021
Epoch 2/10
819/819 [==============================] - 247s 301ms/step - loss: 7.7294 - mae:
2.1630 - val_loss: 10.8321 - val_mae: 2.5637
Epoch 3/10
819/819 [==============================] - 284s 346ms/step - loss: 6.2670 - mae:
1.9413 - val_loss: 12.5510 - val_mae: 2.7508
Epoch 4/10
819/819 [==============================] - 284s 346ms/step - loss: 5.2460 - mae:
1.7722 - val_loss: 12.9916 - val_mae: 2.8196
Epoch 5/10
819/819 [==============================] - 250s 305ms/step - loss: 4.5007 - mae:
1.6370 - val_loss: 13.4033 - val_mae: 2.8380
Epoch 6/10
819/819 [==============================] - 284s 347ms/step - loss: 3.9024 - mae:
1.5196 - val_loss: 13.9598 - val_mae: 2.9098
Epoch 7/10
819/819 [==============================] - 255s 311ms/step - loss: 3.5043 - mae:
1.4364 - val_loss: 14.4846 - val_mae: 2.9718
Epoch 8/10
819/819 [==============================] - 289s 352ms/step - loss: 3.1523 - mae:
1.3604 - val_loss: 14.4538 - val_mae: 2.9774
Epoch 9/10
819/819 [==============================] - 282s 344ms/step - loss: 2.8405 - mae:
1.2913 - val_loss: 15.3092 - val_mae: 3.0587
Epoch 10/10
819/819 [==============================] - 255s 311ms/step - loss: 2.5970 - mae:
```

```
1.2331 - val_loss: 14.8208 - val_mae: 3.0074
405/405 [==============================] - 56s 133ms/step - loss: 11.5477 - mae:
2.6564
Test MAE: 2.66
```

[40]:
```python
import matplotlib.pyplot as plt
loss = history.history["mae"]
val_loss = history.history["val_mae"]

epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, color="grey", linestyle="dashed", label="Training MAE")
plt.plot(epochs, val_loss, color="blue",linestyle="dashed", label="Validation␣
↪MAE")
plt.title("Training and validation MAE")
plt.xlabel("Epochs")
plt.ylabel("MAE")
plt.legend()
plt.show()
```



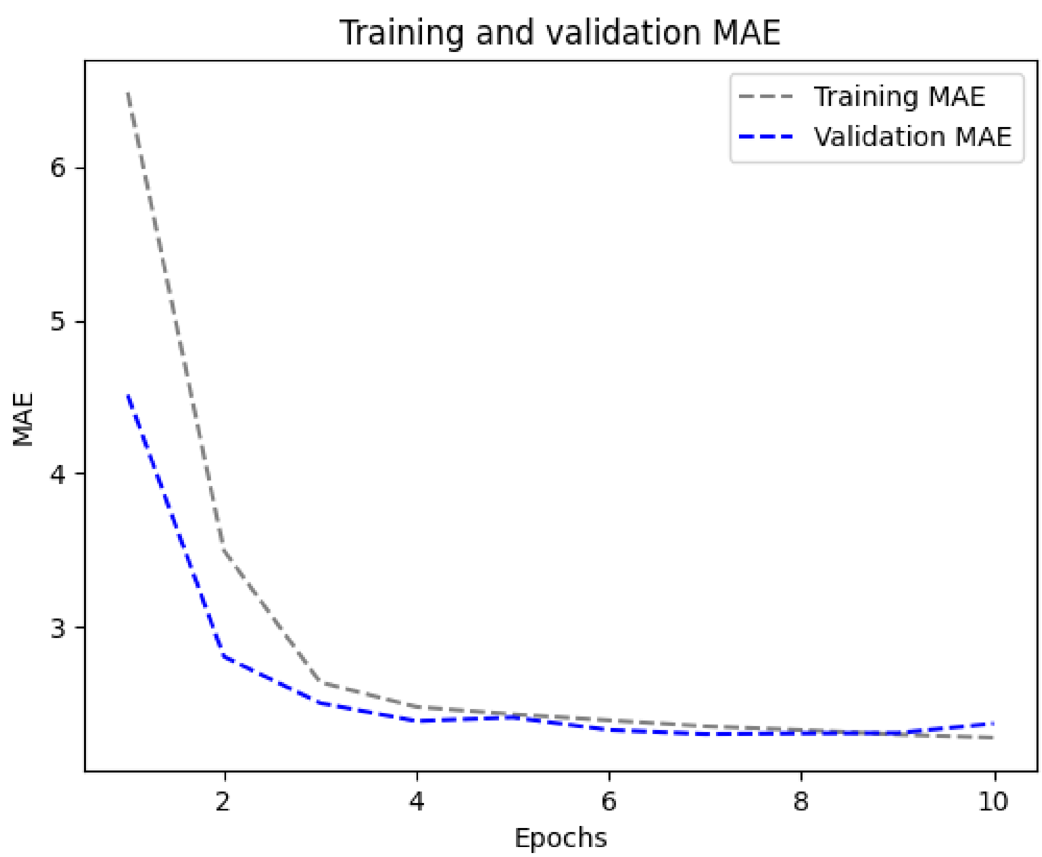
4. LSTM - Stacked setup with 8 units

```
[41]: inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.LSTM(8, return_sequences=True)(inputs)
x = layers.LSTM(8)(x)
outputs = layers.Dense(1)(x)
model = keras.Model(inputs, outputs)

callbacks = [
    keras.callbacks.ModelCheckpoint("jena_LSTM_stacked3.keras",
                                    save_best_only=True)
]
model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
history = model.fit(train_dataset,
                    epochs=10,
                    validation_data=val_dataset,
                    callbacks=callbacks)
model = keras.models.load_model("jena_LSTM_stacked3.keras")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
Epoch 1/10
819/819 [==============================] - 167s 199ms/step - loss: 71.4584 -
mae: 6.4851 - val_loss: 37.0355 - val_mae: 4.5137
Epoch 2/10
819/819 [==============================] - 151s 184ms/step - loss: 22.3344 -
mae: 3.5004 - val_loss: 13.8627 - val_mae: 2.8080
Epoch 3/10
819/819 [==============================] - 161s 196ms/step - loss: 11.6358 -
mae: 2.6407 - val_loss: 10.3696 - val_mae: 2.5055
Epoch 4/10
819/819 [==============================] - 157s 191ms/step - loss: 10.1030 -
mae: 2.4804 - val_loss: 9.4182 - val_mae: 2.3879
Epoch 5/10
819/819 [==============================] - 169s 206ms/step - loss: 9.6770 - mae:
2.4324 - val_loss: 9.5526 - val_mae: 2.4115
Epoch 6/10
819/819 [==============================] - 163s 199ms/step - loss: 9.3672 - mae:
2.3929 - val_loss: 8.9796 - val_mae: 2.3300
Epoch 7/10
819/819 [==============================] - 164s 199ms/step - loss: 9.0602 - mae:
2.3542 - val_loss: 8.8835 - val_mae: 2.3034
Epoch 8/10
819/819 [==============================] - 164s 199ms/step - loss: 8.8728 - mae:
2.3290 - val_loss: 8.8609 - val_mae: 2.3080
Epoch 9/10
819/819 [==============================] - 163s 199ms/step - loss: 8.6555 - mae:
2.3007 - val_loss: 8.8644 - val_mae: 2.3100
Epoch 10/10
```

```
819/819 [==============================] - 161s 197ms/step - loss: 8.5061 - mae:
2.2801 - val_loss: 9.1904 - val_mae: 2.3716
405/405 [==============================] - 35s 81ms/step - loss: 10.4114 - mae:
2.5200
Test MAE: 2.52
```

[42]:
```python
import matplotlib.pyplot as plt
loss = history.history["mae"]
val_loss = history.history["val_mae"]

epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, color="grey", linestyle="dashed", label="Training MAE")
plt.plot(epochs, val_loss, color="blue",linestyle="dashed", label="Validation␣
↪MAE")
plt.title("Training and validation MAE")
plt.xlabel("Epochs")
plt.ylabel("MAE")
plt.legend()
plt.show()
```



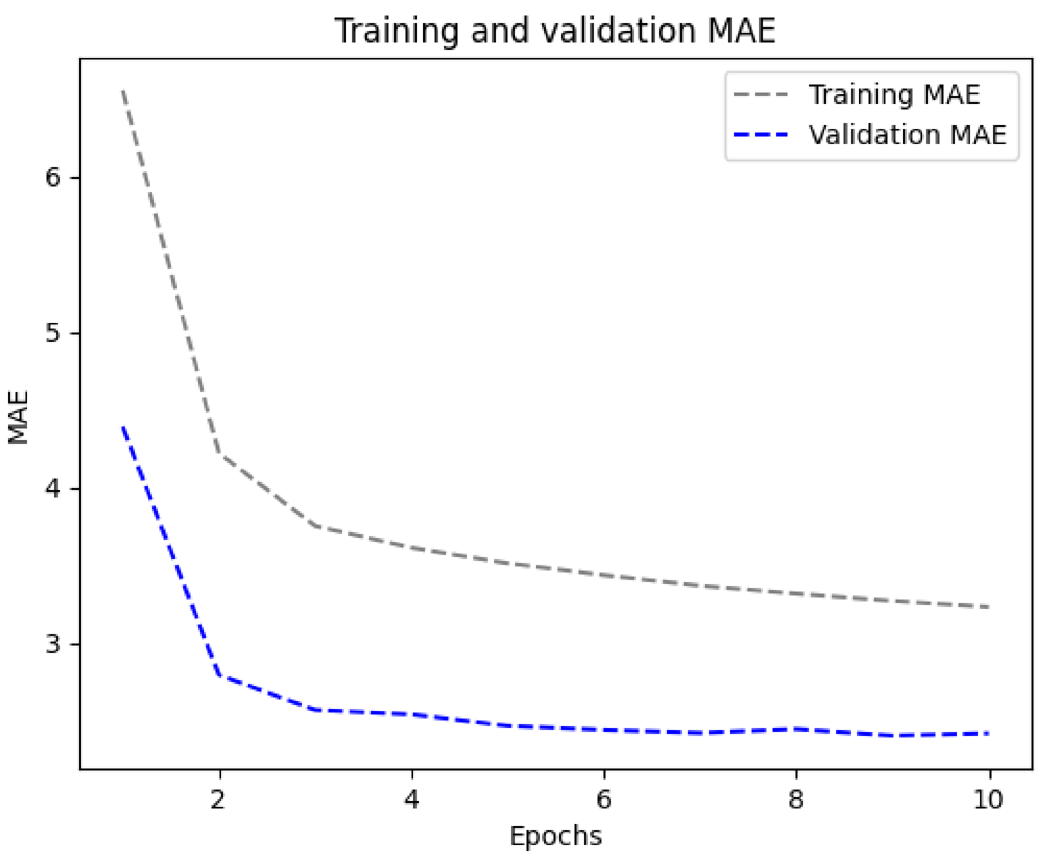
5. LSTM - dropout-regularized, stacked model

[43]:
```
inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.LSTM(8, recurrent_dropout=0.5, return_sequences=True)(inputs)
x = layers.LSTM(8, recurrent_dropout=0.5)(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1)(x)
model = keras.Model(inputs, outputs)

callbacks = [
    keras.callbacks.ModelCheckpoint("jena_stacked_LSTM_dropout.keras",
                                    save_best_only=True)
]
model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
history = model.fit(train_dataset,
                    epochs=10,
                    validation_data=val_dataset,
                    callbacks=callbacks)
model = keras.models.load_model("jena_stacked_LSTM_dropout.keras")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
Epoch 1/10
819/819 [==============================] - 271s 325ms/step - loss: 72.6522 -
mae: 6.5508 - val_loss: 34.6757 - val_mae: 4.3884
Epoch 2/10
819/819 [==============================] - 272s 331ms/step - loss: 31.6224 -
mae: 4.2181 - val_loss: 13.9650 - val_mae: 2.7925
Epoch 3/10
819/819 [==============================] - 272s 332ms/step - loss: 24.5615 -
mae: 3.7505 - val_loss: 11.1867 - val_mae: 2.5673
Epoch 4/10
819/819 [==============================] - 268s 327ms/step - loss: 22.6435 -
mae: 3.6104 - val_loss: 10.8404 - val_mae: 2.5390
Epoch 5/10
819/819 [==============================] - 258s 314ms/step - loss: 21.3638 -
mae: 3.5102 - val_loss: 10.1880 - val_mae: 2.4653
Epoch 6/10
819/819 [==============================] - 261s 319ms/step - loss: 20.3786 -
mae: 3.4344 - val_loss: 9.8552 - val_mae: 2.4392
Epoch 7/10
819/819 [==============================] - 275s 336ms/step - loss: 19.5037 -
mae: 3.3663 - val_loss: 9.7018 - val_mae: 2.4204
Epoch 8/10
819/819 [==============================] - 269s 328ms/step - loss: 18.9316 -
mae: 3.3150 - val_loss: 9.9003 - val_mae: 2.4444
Epoch 9/10
819/819 [==============================] - 272s 332ms/step - loss: 18.3716 -
mae: 3.2684 - val_loss: 9.5451 - val_mae: 2.4026
```

```
Epoch 10/10
819/819 [==============================] - 258s 315ms/step - loss: 17.8782 -
mae: 3.2292 - val_loss: 9.6397 - val_mae: 2.4179
405/405 [==============================] - 34s 82ms/step - loss: 11.1375 - mae:
2.5898
Test MAE: 2.59
```

[44]:
```python
import matplotlib.pyplot as plt
loss = history.history["mae"]
val_loss = history.history["val_mae"]

epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, color="grey", linestyle="dashed", label="Training MAE")
plt.plot(epochs, val_loss, color="blue",linestyle="dashed", label="Validation MAE")
plt.title("Training and validation MAE")
plt.xlabel("Epochs")
plt.ylabel("MAE")
plt.legend()
plt.show()
```



6. **Bidirectional LSTM**

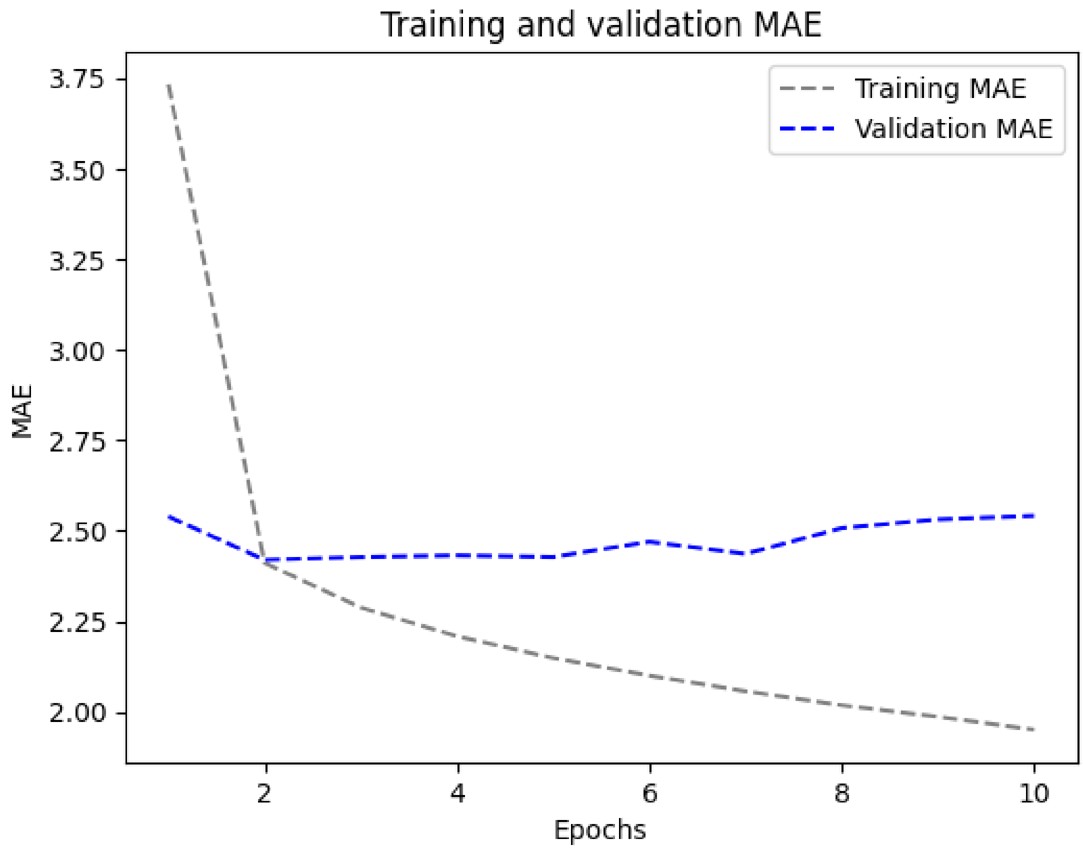
[45]:
```
inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.Bidirectional(layers.LSTM(16))(inputs)
outputs = layers.Dense(1)(x)
model = keras.Model(inputs, outputs)

callbacks = [
    keras.callbacks.ModelCheckpoint("jena_bidirec_LSTM.keras",
                                    save_best_only=True)
]

model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
history = model.fit(train_dataset,
                    epochs=10,
                    validation_data=val_dataset,
                    callbacks=callbacks)

model = keras.models.load_model("jena_bidirec_LSTM.keras")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
Epoch 1/10
819/819 [==============================] - 166s 196ms/step - loss: 27.0466 -
mae: 3.7328 - val_loss: 10.6734 - val_mae: 2.5394
Epoch 2/10
819/819 [==============================] - 156s 190ms/step - loss: 9.5604 - mae:
2.4115 - val_loss: 9.6774 - val_mae: 2.4199
Epoch 3/10
819/819 [==============================] - 153s 186ms/step - loss: 8.6406 - mae:
2.2883 - val_loss: 9.7280 - val_mae: 2.4276
Epoch 4/10
819/819 [==============================] - 154s 188ms/step - loss: 8.0362 - mae:
2.2090 - val_loss: 9.8251 - val_mae: 2.4324
Epoch 5/10
819/819 [==============================] - 161s 196ms/step - loss: 7.6009 - mae:
2.1490 - val_loss: 9.8496 - val_mae: 2.4279
Epoch 6/10
819/819 [==============================] - 163s 199ms/step - loss: 7.2491 - mae:
2.1003 - val_loss: 10.1469 - val_mae: 2.4699
Epoch 7/10
819/819 [==============================] - 156s 190ms/step - loss: 6.9529 - mae:
2.0574 - val_loss: 9.9412 - val_mae: 2.4365
Epoch 8/10
819/819 [==============================] - 162s 197ms/step - loss: 6.7032 - mae:
2.0185 - val_loss: 10.5457 - val_mae: 2.5075
Epoch 9/10
```

```
819/819 [==============================] - 163s 199ms/step - loss: 6.4875 - mae:
1.9868 - val_loss: 10.7442 - val_mae: 2.5316
Epoch 10/10
819/819 [==============================] - 163s 199ms/step - loss: 6.2524 - mae:
1.9513 - val_loss: 10.8836 - val_mae: 2.5412
405/405 [==============================] - 36s 83ms/step - loss: 10.5141 - mae:
2.5559
Test MAE: 2.56
```

[46]:

```python
import matplotlib.pyplot as plt
loss = history.history["mae"]
val_loss = history.history["val_mae"]

epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, color="grey", linestyle="dashed", label="Training MAE")
plt.plot(epochs, val_loss, color="blue",linestyle="dashed", label="Validation␣
↪MAE")
plt.title("Training and validation MAE")
plt.xlabel("Epochs")
plt.ylabel("MAE")
plt.legend()
plt.show()
```



1D Convnets and LSTM together

[47]:
```
inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.Conv1D(64, 3, activation='relu')(inputs)
x = layers.MaxPooling1D(3)(x)
x = layers.Conv1D(128, 3, activation='relu')(x)
x = layers.GlobalMaxPooling1D()(x)
x = layers.Reshape((-1, 128))(x)  # Reshape the data to be 3D
x = layers.LSTM(16)(x)
outputs = layers.Dense(1)(x)
model = keras.Model(inputs, outputs)

model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])

callbacks = [
    keras.callbacks.ModelCheckpoint("jena_Conv_LSTM.keras", save_best_only=True)
]

history = model.fit(train_dataset, epochs=10, validation_data=val_dataset,
 callbacks=callbacks)
```

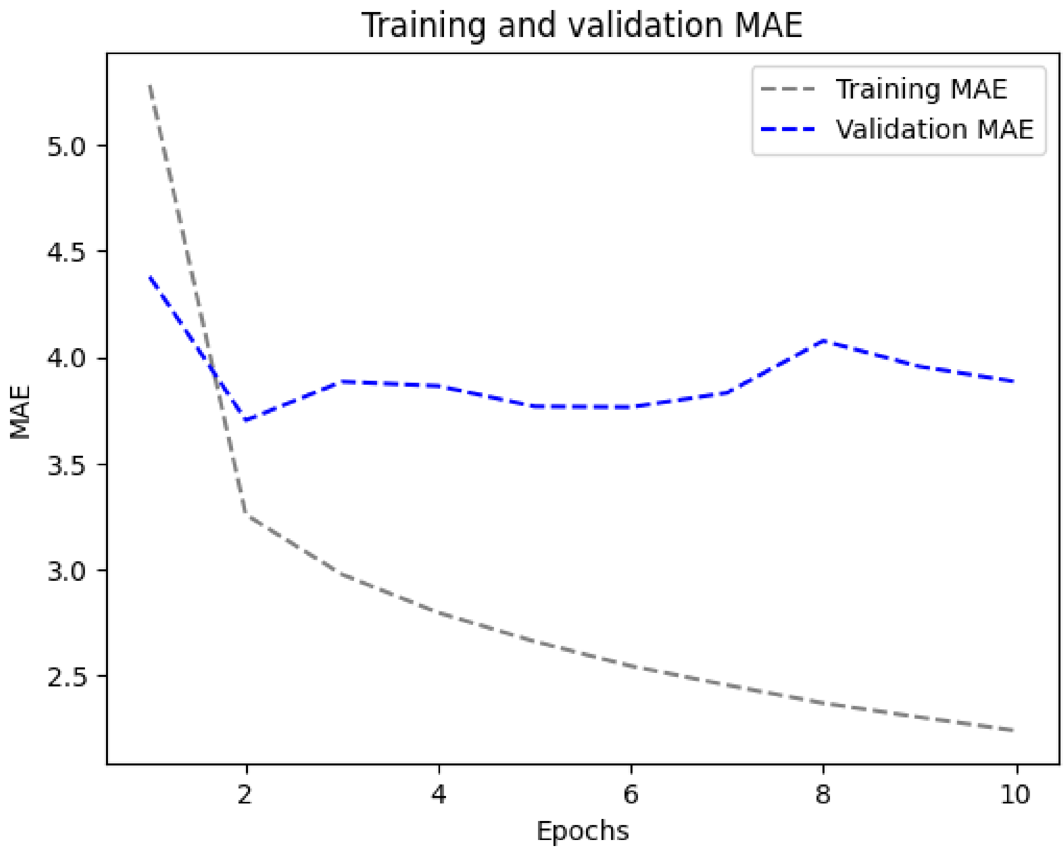
```
model = keras.models.load_model("jena_Conv_LSTM.keras")
print(f"Test  MAE:  {model.evaluate(test_dataset)[1]:.2f}")
```

```
Epoch 1/10
819/819 [==============================] - 117s 140ms/step - loss: 49.8492 -
mae: 5.2829 - val_loss: 31.7686 - val_mae: 4.3797
Epoch 2/10
819/819 [==============================] - 132s 161ms/step - loss: 17.9138 -
mae: 3.2583 - val_loss: 22.3645 - val_mae: 3.7028
Epoch 3/10
819/819 [==============================] - 134s 163ms/step - loss: 14.7419 -
mae: 2.9767 - val_loss: 23.3027 - val_mae: 3.8847
Epoch 4/10
819/819 [==============================] - 138s 169ms/step - loss: 13.0552 -
mae: 2.7967 - val_loss: 24.1605 - val_mae: 3.8643
Epoch 5/10
819/819 [==============================] - 116s 141ms/step - loss: 11.8949 -
mae: 2.6610 - val_loss: 22.0105 - val_mae: 3.7677
Epoch 6/10
819/819 [==============================] - 116s 141ms/step - loss: 10.8960 -
mae: 2.5457 - val_loss: 22.9830 - val_mae: 3.7645
Epoch 7/10
819/819 [==============================] - 114s 139ms/step - loss: 10.1881 -
mae: 2.4556 - val_loss: 22.7887 - val_mae: 3.8321
Epoch 8/10
819/819 [==============================] - 114s 139ms/step - loss: 9.5312 - mae:
2.3708 - val_loss: 27.5114 - val_mae: 4.0771
Epoch 9/10
819/819 [==============================] - 114s 139ms/step - loss: 9.0145 - mae:
2.3043 - val_loss: 24.8584 - val_mae: 3.9561
Epoch 10/10
819/819 [==============================] - 112s 137ms/step - loss: 8.5816 - mae:
2.2416 - val_loss: 24.1354 - val_mae: 3.8849
405/405 [==============================] - 26s 62ms/step - loss: 23.1759 - mae:
3.8745
Test MAE: 3.87
```

[48]:

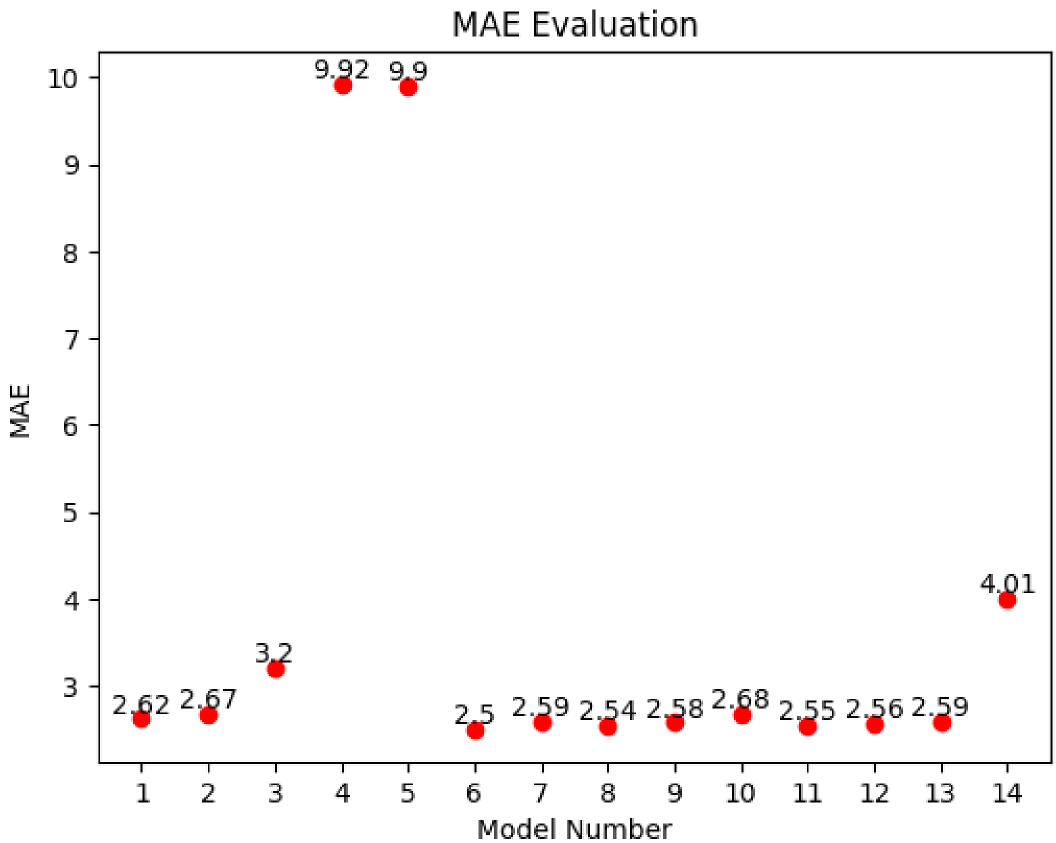
```
import matplotlib.pyplot as plt
loss  =  history.history["mae"]
val_loss  =  history.history["val_mae"]

epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, color="grey", linestyle="dashed", label="Training  MAE")
plt.plot(epochs,  val_loss, color="blue",linestyle="dashed", label="Validation␣
↪MAE")
```

```
plt.title("Training and validation MAE")
plt.xlabel("Epochs")
plt.ylabel("MAE")
plt.legend()
plt.show()
```



[49]:

```
Models = ("1","2","3","4","5","6","7","8","9","10","11","12","13","14")
Mae = (2.62,2.67,3.2,9.92,9.9,2.5,2.59,2.54,2.58,2.68,2.55,2.56,2.59,4.01)

# MAE Evaluation
plt.scatter(Models, Mae, color="red")
plt.title("MAE Evaluation")
plt.xlabel("Model Number")
plt.ylabel("MAE")

for (xi, yi) in zip(Models,Mae):
    plt.text(xi, yi, yi, va='bottom', ha='center')

plt.show()
```


MAE Evaluation
10
9
8
7
6
5
4
3
MAE
9.92
9.9
4.01
3.2
2.62
2.67
2.5
2.59
2.54
2.58
2.68
2.55
2.56
2.59
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Model Number