assignment-3-time-series-data

April 5, 2024

Assignment 3: Time-Series Data

Group -24

Shiva Chaitanya Goud Gadila (811252042)

Praneeth Simha Mallenahalli Vijaya (811252718)

[6]:
```
!wget https://s3.amazonaws.com/keras-datasets/jena_climate_2009_2016.csv.zip
!unzip jena_climate_2009_2016.csv.zip
```

```
--2024-04-04 00:27:05--  https://s3.amazonaws.com/keras-
datasets/jena_climate_2009_2016.csv.zip
Resolving s3.amazonaws.com (s3.amazonaws.com)… 52.217.114.176, 54.231.200.128,
54.231.139.200, …
Connecting to s3.amazonaws.com (s3.amazonaws.com)|52.217.114.176|:443…
connected.
HTTP request sent, awaiting response… 200 OK
Length: 13565642 (13M) [application/zip]
Saving to: ‘jena_climate_2009_2016.csv.zip.2’

jena_climate_2009_2 100%[===================>]  12.94M  45.4MB/s    in 0.3s

2024-04-04 00:27:05 (45.4 MB/s) - ‘jena_climate_2009_2016.csv.zip.2’ saved
[13565642/13565642]

Archive:  jena_climate_2009_2016.csv.zip
replace jena_climate_2009_2016.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: jena_climate_2009_2016.csv
replace __MACOSX/._jena_climate_2009_2016.csv? [y]es, [n]o, [A]ll, [N]one,
[r]ename: y
  inflating: __MACOSX/._jena_climate_2009_2016.csv
```

[7]:
```
import os
fname = os.path.join("jena_climate_2009_2016.csv")
```

[8]:
```
with open(fname) as f:
    data = f.read()
```

[9]:
```
lines = data.split("\n")
header = lines[0].split(",")
lines = lines[1:]
print(header)
print(len(lines))
import os
fname = os.path.join("jena_climate_2009_2016.csv")
```

```
['"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
```

[10]:
```
with open(fname) as f:
    data = f.read()
```

[11]:
```
lines = data.split("\n")
header = lines[0].split(",")
lines = lines[1:]
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 (mmol/mol)"', '"rho (g/m**3)"', '"wv (m/s)"', '"max. wv (m/s)"', '"wd
(deg)"']
420451
```

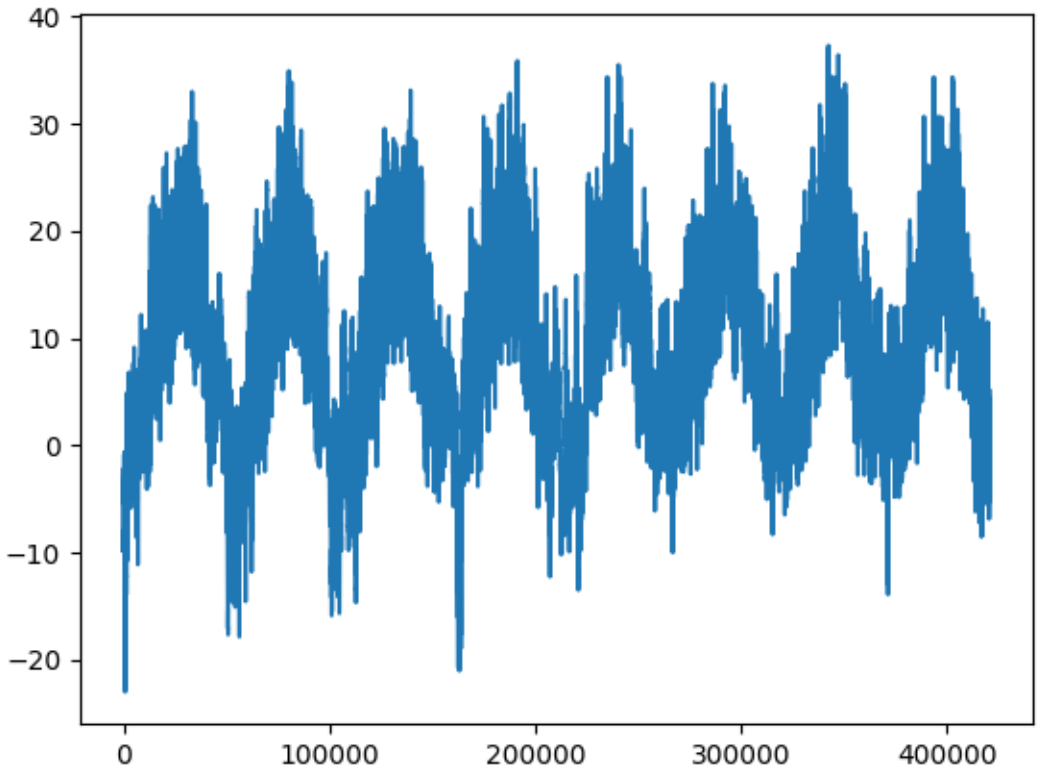
Parsing CSV data to extract temperature values and store them in a numpy array alongside other raw data.

[12]:
```
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[:]
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 temperature data using matplotlib's pyplot module.

[13]:
```
from matplotlib import pyplot as plt
plt.plot(range(len(temperature)), temperature)
```

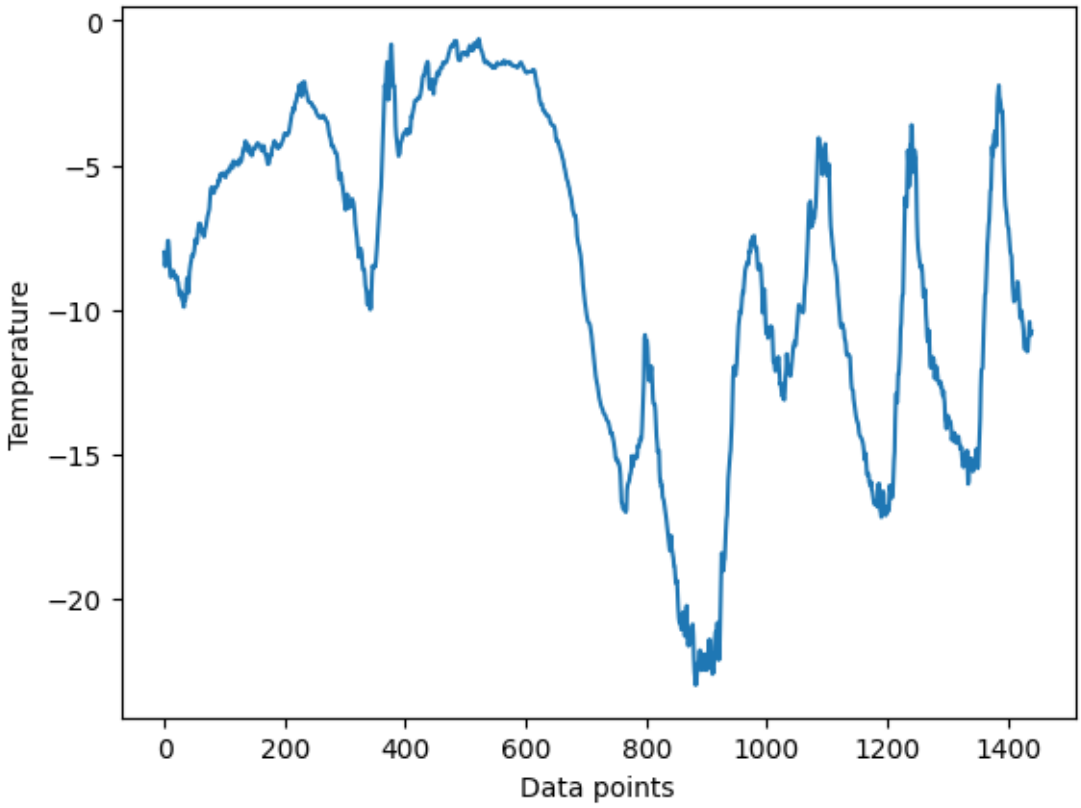
[13]:
```
[<matplotlib.lines.Line2D at 0x7ce8264b99c0>]
```



Plotting the first 1440 data points of temperature against data points with labeled axes.

[41]:
```
plt.plot(range(1440), temperature[:1440])
plt.xlabel('Data points')
plt.ylabel('Temperature')
```

[41]:
```
Text(0, 0.5, 'Temperature')
```



Calculating the number of training, validation, and test samples, each representing 50%, 25%, and 25% of the total data respectively. Printing out the counts for each category.

[15]:
```
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
```

Normalizing the raw data by subtracting the mean and dividing by the standard deviation calculated from the training set. Additionally, creating a dummy time series dataset using TensorFlow's Keras API for further analysis.

[16]:
```
mean = raw_data[:num_train_samples].mean(axis=0)
raw_data -= mean
std = raw_data[:num_train_samples].std(axis=0)
```

```
raw_data /= std
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,
)
```

Printing the input sequences and their corresponding targets from the dummy dataset. Each input sequence has a length of 3, and the target is the next element in the sequence.

[17]:
```
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
```

Setting up parameters for time series data processing:

sampling_rate: Sampling rate of the data, assumed to be 6 in this case. sequence_length: Length of each sequence in the time series data, set to 120. delay: Delay for the target labels, calculated based on the sampling rate and sequence length. batch_size: Batch size for training the model, chosen as 256.

[18]:
```
sampling_rate = 6
sequence_length = 120
delay = sampling_rate * (sequence_length + 24 - 1)
batch_size = 256
```

Creating a TensorFlow dataset for training:

Using timeseries_dataset_from_array function from Keras utilities. Utilizing the raw data and temperature targets. Setting the sampling rate, sequence length, and batch size. Shuffling the dataset and specifying the start and end indices for training.

[19]:
```
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)
```

[20]:
```
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)
```

[21]:
```
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)
```

[22]:
```
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,)
```

Defining a function evaluate_naive_method(dataset) to evaluate a naive forecasting method:

It calculates the total absolute error between predicted and actual values. Iterates through the dataset, extracting samples and targets. Makes predictions using the last temperature value from each sample. Scales the predictions back to the original scale using mean and standard deviation. Returns the mean absolute error across all samples.

[23]:
```
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
```

Printing the mean absolute error (MAE) for the validation and test datasets using the naive forecasting method:

[42]:
```
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
```

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

```
Requirement already satisfied: tensorflow==2.12 in
/usr/local/lib/python3.10/dist-packages (2.12.0)
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)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (0.4.0)
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)
Requirement already satisfied: keras<2.13,>=2.12.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (2.12.0)
Requirement already satisfied: libclang>=13.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (18.1.1)
Requirement already satisfied: numpy<1.24,>=1.22 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (1.23.5)
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)
Requirement already satisfied: tensorboard<2.13,>=2.12 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (2.12.3)
Requirement already satisfied: tensorflow-estimator<2.13,>=2.12.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (2.12.0)
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)
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.13,>=2.12->tensorflow==2.12) (1.0.0)
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)
```

Here's the code for defining, training, and evaluating a simple dense neural network model using TensorFlow and Keras:

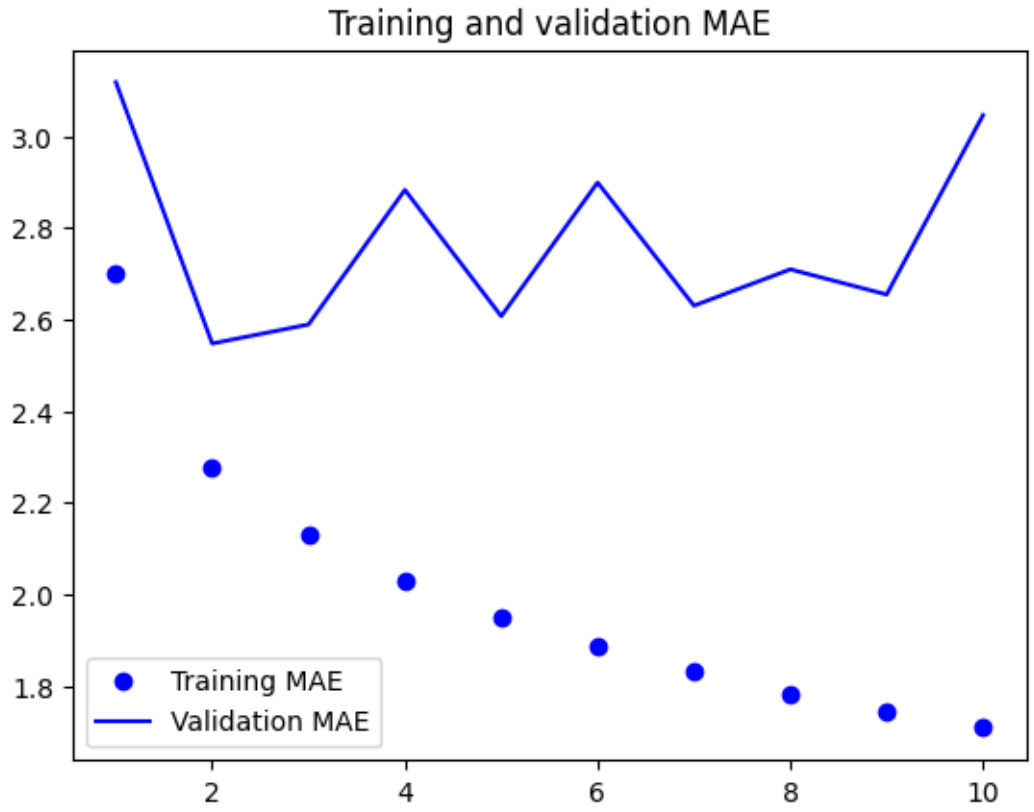
[25]:
```python
from tensorflow import keras
from tensorflow.keras import layers

inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.Flatten()(inputs)
x = layers.Dense(64, activation="relu")(x)
outputs = layers.Dense(1)(x)
model = keras.Model(inputs, outputs)

callbacks = [
    keras.callbacks.ModelCheckpoint("jena_dense.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_dense.keras")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
#Plotting results
```

```
Epoch 1/10
819/819 [==============================] - 48s 58ms/step - loss: 12.1419 - mae:
2.7007 - val_loss: 15.0232 - val_mae: 3.1178
```

```
Epoch 2/10
819/819 [==============================] - 38s 46ms/step - loss: 8.4430 - mae:
2.2751 - val_loss: 10.4297 - val_mae: 2.5473
Epoch 3/10
819/819 [==============================] - 38s 46ms/step - loss: 7.3702 - mae:
2.1308 - val_loss: 10.7738 - val_mae: 2.5889
Epoch 4/10
819/819 [==============================] - 38s 46ms/step - loss: 6.6873 - mae:
2.0297 - val_loss: 13.2782 - val_mae: 2.8826
Epoch 5/10
819/819 [==============================] - 40s 49ms/step - loss: 6.1396 - mae:
1.9475 - val_loss: 11.0336 - val_mae: 2.6069
Epoch 6/10
819/819 [==============================] - 38s 47ms/step - loss: 5.7575 - mae:
1.8884 - val_loss: 13.2894 - val_mae: 2.8987
Epoch 7/10
819/819 [==============================] - 38s 46ms/step - loss: 5.4175 - mae:
1.8315 - val_loss: 11.1364 - val_mae: 2.6299
Epoch 8/10
819/819 [==============================] - 38s 46ms/step - loss: 5.1374 - mae:
1.7835 - val_loss: 11.7978 - val_mae: 2.7092
Epoch 9/10
819/819 [==============================] - 38s 46ms/step - loss: 4.8890 - mae:
1.7418 - val_loss: 11.4758 - val_mae: 2.6542
Epoch 10/10
819/819 [==============================] - 47s 57ms/step - loss: 4.6960 - mae:
1.7083 - val_loss: 14.8288 - val_mae: 3.0458
405/405 [==============================] - 13s 30ms/step - loss: 11.2859 - mae:
2.6697
Test MAE: 2.67
```

[26]:
```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, "bo", label="Training MAE")
plt.plot(epochs, val_loss, "b", label="Validation MAE")
plt.title("Training and validation MAE")
plt.legend()
plt.show()
#Let's try a 1D convolutional model
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)
```



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

[43]:
```
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 [==============================] - 80s 96ms/step - loss: 138.6828 - mae:
9.6837 - val_loss: 143.9035 - val_mae: 9.8909
Epoch 2/10
819/819 [==============================] - 62s 75ms/step - loss: 136.4341 - mae:
9.5647 - val_loss: 143.7349 - val_mae: 9.8754
Epoch 3/10
819/819 [==============================] - 57s 69ms/step - loss: 136.2783 - mae:
9.5516 - val_loss: 143.6675 - val_mae: 9.8685
Epoch 4/10
819/819 [==============================] - 58s 70ms/step - loss: 136.2044 - mae:
9.5442 - val_loss: 143.5766 - val_mae: 9.8542
Epoch 5/10
819/819 [==============================] - 57s 70ms/step - loss: 136.1465 - mae:
9.5356 - val_loss: 143.5159 - val_mae: 9.8469
Epoch 6/10
819/819 [==============================] - 60s 74ms/step - loss: 136.1315 - mae:
9.5348 - val_loss: 143.5391 - val_mae: 9.8501
Epoch 7/10
819/819 [==============================] - 61s 74ms/step - loss: 136.1181 - mae:
9.5329 - val_loss: 143.5097 - val_mae: 9.8463
Epoch 8/10
819/819 [==============================] - 59s 72ms/step - loss: 136.0952 - mae:
9.5299 - val_loss: 143.5160 - val_mae: 9.8491
Epoch 9/10
819/819 [==============================] - 60s 73ms/step - loss: 136.0791 - mae:
9.5279 - val_loss: 143.4976 - val_mae: 9.8462
405/405 [==============================] - 15s 37ms/step - loss: 151.2771 - mae:
9.9174
Test MAE: 9.92
```

2.Simple RNN - Stacking RNN layers

[44]:
```
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 [==============================] - 109s 130ms/step - loss: 136.9142 -
mae: 9.5699 - val_loss: 143.4303 - val_mae: 9.8384
Epoch 2/10
819/819 [==============================] - 108s 132ms/step - loss: 135.9478 -
mae: 9.5125 - val_loss: 143.4102 - val_mae: 9.8380
Epoch 3/10
819/819 [==============================] - 115s 140ms/step - loss: 135.9009 -
mae: 9.5065 - val_loss: 143.3955 - val_mae: 9.8326
Epoch 4/10
819/819 [==============================] - 108s 132ms/step - loss: 135.8799 -
mae: 9.5027 - val_loss: 143.3930 - val_mae: 9.8333
Epoch 5/10
819/819 [==============================] - 109s 132ms/step - loss: 135.8532 -
mae: 9.4994 - val_loss: 143.3798 - val_mae: 9.8313
Epoch 6/10
819/819 [==============================] - 107s 130ms/step - loss: 135.8342 -
mae: 9.4965 - val_loss: 143.3792 - val_mae: 9.8320
Epoch 7/10
819/819 [==============================] - 109s 133ms/step - loss: 135.8004 -
mae: 9.4913 - val_loss: 143.3634 - val_mae: 9.8272
Epoch 8/10
819/819 [==============================] - 126s 154ms/step - loss: 135.7863 -
mae: 9.4887 - val_loss: 143.3931 - val_mae: 9.8341
Epoch 9/10
819/819 [==============================] - 108s 131ms/step - loss: 135.7831 -
mae: 9.4881 - val_loss: 143.4082 - val_mae: 9.8354
Epoch 10/10
819/819 [==============================] - 113s 138ms/step - loss: 135.7638 -
mae: 9.4854 - val_loss: 143.4698 - val_mae: 9.8449
405/405 [==============================] - 23s 54ms/step - loss: 151.0813 - mae:
9.8992
Test MAE: 9.90
```

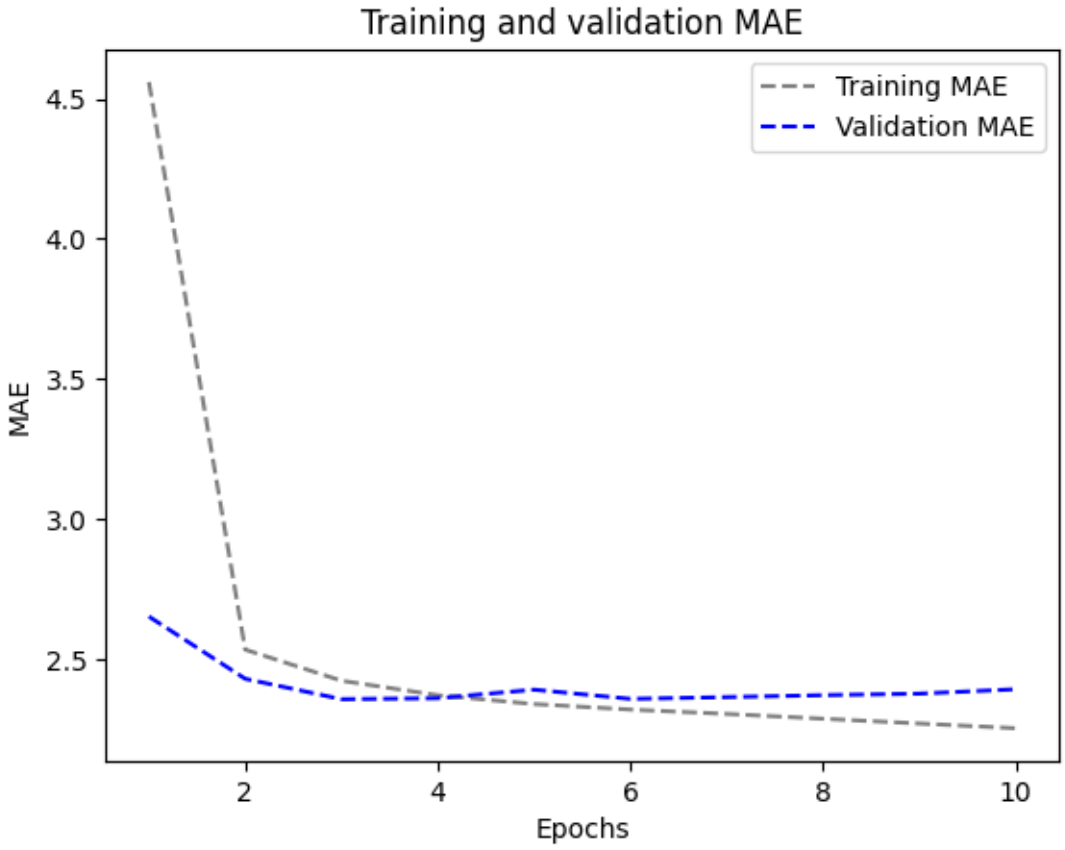
A Simple GRU (Gated Recurrent Unit)

[45]:
```
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 [==============================] - 93s 111ms/step - loss: 39.5131 - mae:
4.5597 - val_loss: 12.3568 - val_mae: 2.6523
Epoch 2/10
819/819 [==============================] - 92s 112ms/step - loss: 10.6319 - mae:
2.5343 - val_loss: 9.9693 - val_mae: 2.4302
Epoch 3/10
819/819 [==============================] - 93s 113ms/step - loss: 9.6343 - mae:
2.4232 - val_loss: 9.2709 - val_mae: 2.3573
Epoch 4/10
819/819 [==============================] - 90s 110ms/step - loss: 9.2377 - mae:
2.3715 - val_loss: 9.3694 - val_mae: 2.3609
Epoch 5/10
819/819 [==============================] - 91s 110ms/step - loss: 9.0003 - mae:
2.3400 - val_loss: 9.6584 - val_mae: 2.3914
Epoch 6/10
819/819 [==============================] - 93s 113ms/step - loss: 8.8374 - mae:
2.3203 - val_loss: 9.2765 - val_mae: 2.3584
Epoch 7/10
819/819 [==============================] - 92s 111ms/step - loss: 8.6971 - mae:
2.3048 - val_loss: 9.3369 - val_mae: 2.3647
Epoch 8/10
819/819 [==============================] - 92s 112ms/step - loss: 8.5596 - mae:
2.2876 - val_loss: 9.3728 - val_mae: 2.3716
Epoch 9/10
819/819 [==============================] - 91s 111ms/step - loss: 8.4232 - mae:
2.2706 - val_loss: 9.3985 - val_mae: 2.3771
Epoch 10/10
819/819 [==============================] - 92s 112ms/step - loss: 8.2932 - mae:
```

```
2.2537 - val_loss: 9.5277 - val_mae: 2.3929
405/405 [==============================] - 20s 47ms/step - loss: 10.0385 - mae:
2.4859
Test MAE: 2.49
```

[46]:
```
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()
```



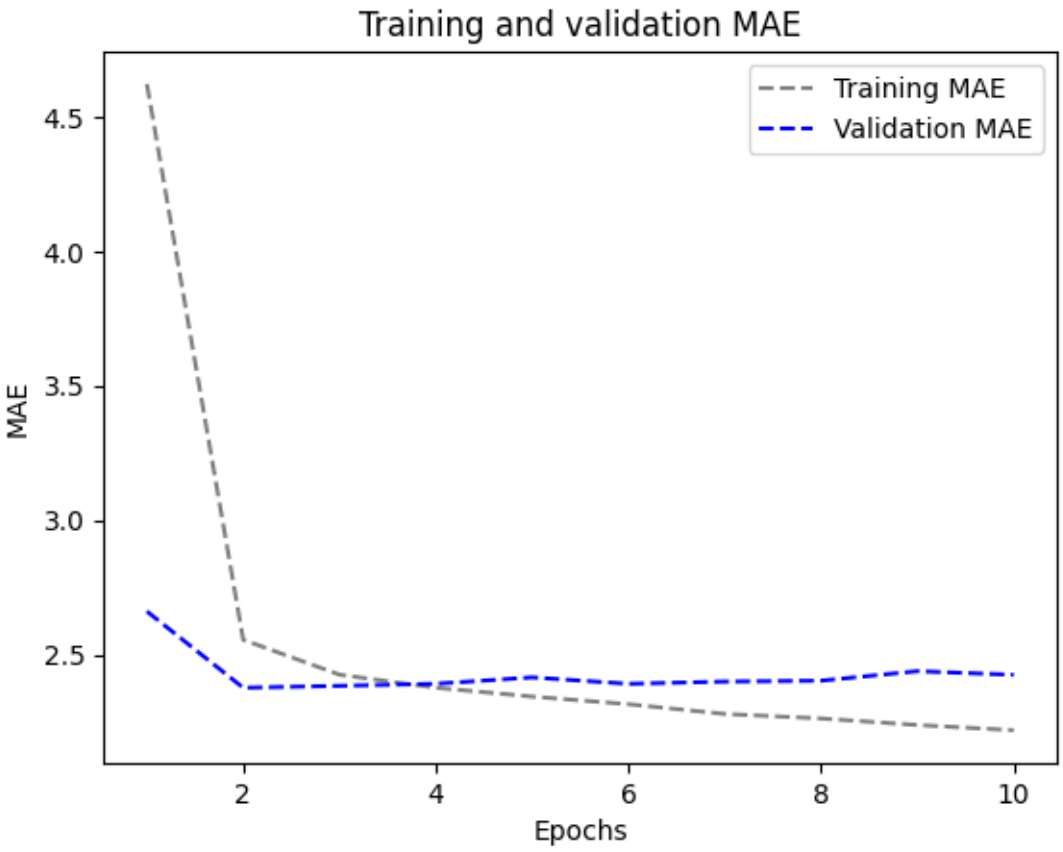
LSTM(Long Short-Term Memory)

[47]:
```
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 [==============================] - 92s 110ms/step - loss: 40.2607 - mae:
4.6215 - val_loss: 12.2565 - val_mae: 2.6609
Epoch 2/10
819/819 [==============================] - 91s 111ms/step - loss: 10.8136 - mae:
2.5554 - val_loss: 9.4701 - val_mae: 2.3770
Epoch 3/10
819/819 [==============================] - 110s 135ms/step - loss: 9.6414 - mae:
2.4254 - val_loss: 9.4785 - val_mae: 2.3841
Epoch 4/10
819/819 [==============================] - 92s 112ms/step - loss: 9.2739 - mae:
2.3770 - val_loss: 9.6148 - val_mae: 2.3918
Epoch 5/10
819/819 [==============================] - 93s 113ms/step - loss: 9.0310 - mae:
2.3440 - val_loss: 9.6704 - val_mae: 2.4154
Epoch 6/10
819/819 [==============================] - 93s 113ms/step - loss: 8.8291 - mae:
2.3159 - val_loss: 9.5246 - val_mae: 2.3916
Epoch 7/10
819/819 [==============================] - 90s 110ms/step - loss: 8.5707 - mae:
2.2795 - val_loss: 9.5929 - val_mae: 2.3999
Epoch 8/10
819/819 [==============================] - 91s 111ms/step - loss: 8.4377 - mae:
2.2627 - val_loss: 9.4788 - val_mae: 2.4032
Epoch 9/10
819/819 [==============================] - 95s 116ms/step - loss: 8.2700 - mae:
2.2384 - val_loss: 9.8037 - val_mae: 2.4385
Epoch 10/10
```

```
819/819 [==============================] - 96s 117ms/step - loss: 8.1560 - mae:
2.2190 - val_loss: 9.6946 - val_mae: 2.4251
405/405 [==============================] - 25s 60ms/step - loss: 10.4366 - mae:
2.5319
Test MAE: 2.53
```

[48]:
```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()
```



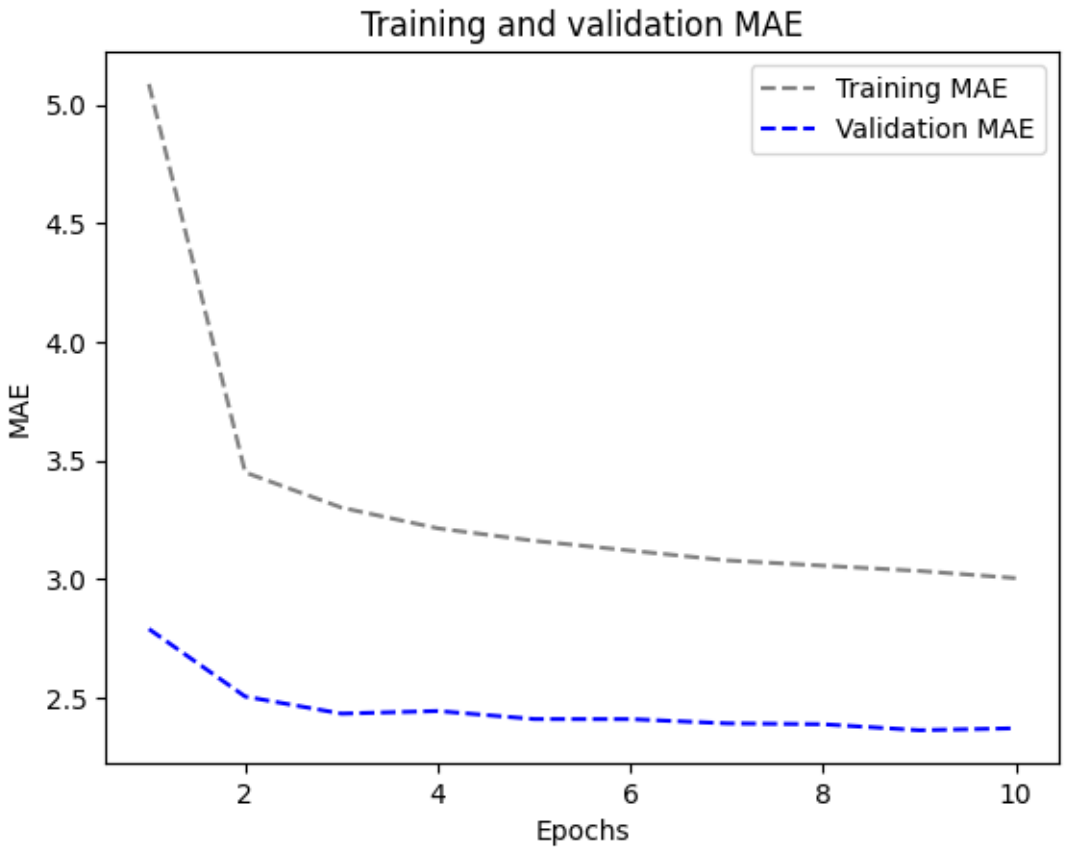
2.LSTM - dropout Regularization

[49]:
```
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 [==============================] - 186s 223ms/step - loss: 46.3510 -
mae: 5.0878 - val_loss: 13.5440 - val_mae: 2.7896
Epoch 2/10
819/819 [==============================] - 156s 190ms/step - loss: 20.0653 -
mae: 3.4500 - val_loss: 10.3534 - val_mae: 2.5047
Epoch 3/10
819/819 [==============================] - 156s 190ms/step - loss: 18.3547 -
mae: 3.3015 - val_loss: 9.6818 - val_mae: 2.4327
Epoch 4/10
819/819 [==============================] - 155s 189ms/step - loss: 17.4118 -
mae: 3.2141 - val_loss: 9.7123 - val_mae: 2.4438
Epoch 5/10
819/819 [==============================] - 153s 187ms/step - loss: 16.8742 -
mae: 3.1616 - val_loss: 9.4938 - val_mae: 2.4098
Epoch 6/10
819/819 [==============================] - 201s 245ms/step - loss: 16.4310 -
mae: 3.1202 - val_loss: 9.5132 - val_mae: 2.4095
Epoch 7/10
819/819 [==============================] - 175s 213ms/step - loss: 15.9409 -
mae: 3.0790 - val_loss: 9.3897 - val_mae: 2.3916
Epoch 8/10
819/819 [==============================] - 154s 188ms/step - loss: 15.7403 -
mae: 3.0567 - val_loss: 9.3528 - val_mae: 2.3879
Epoch 9/10
819/819 [==============================] - 157s 191ms/step - loss: 15.4791 -
mae: 3.0345 - val_loss: 9.1842 - val_mae: 2.3627
```

```
Epoch 10/10
819/819 [==============================] - 156s 191ms/step - loss: 15.1394 -
mae: 3.0043 - val_loss: 9.2603 - val_mae: 2.3710
405/405 [==============================] - 24s 57ms/step - loss: 10.5299 - mae:
2.5439
Test MAE: 2.54
```

[50]:
```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()
```



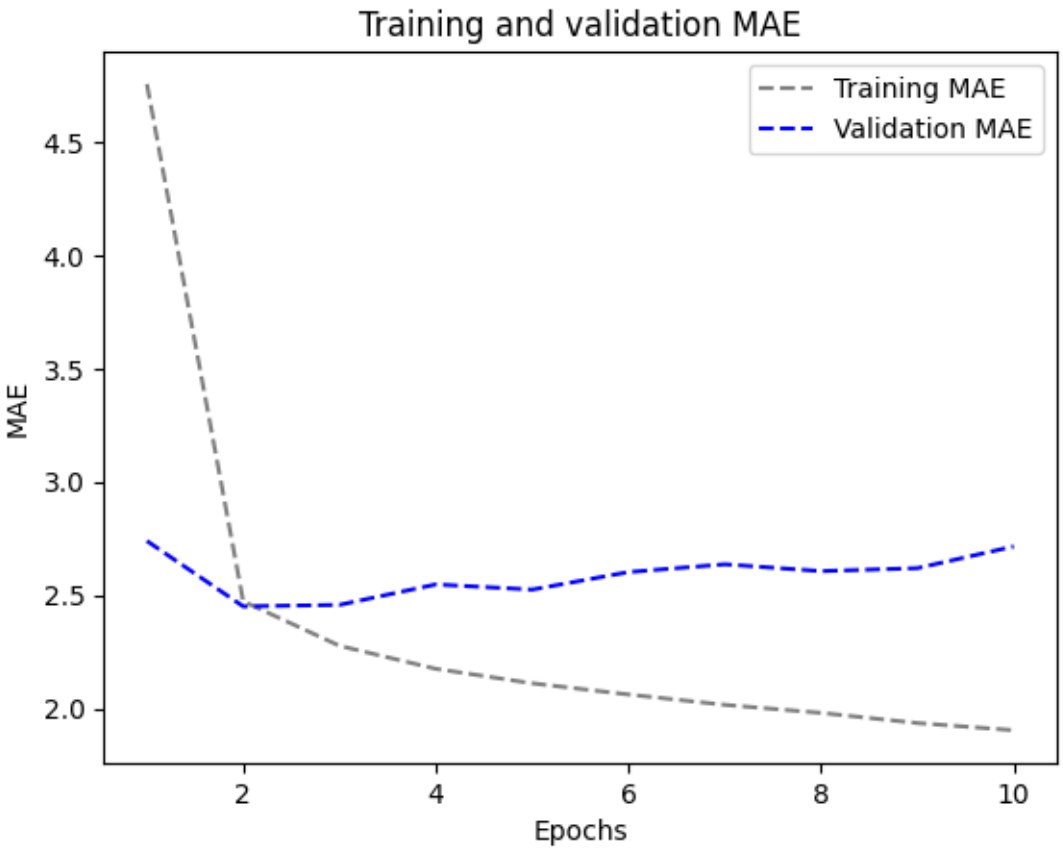
3.LSTM - Stacked setup with 16 units

[51]:
```
inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.LSTM(16, return_sequences=True)(inputs)
x = layers.LSTM(16)(x)
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 [==============================] - 162s 193ms/step - loss: 43.0180 -
mae: 4.7559 - val_loss: 13.1899 - val_mae: 2.7389
Epoch 2/10
819/819 [==============================] - 156s 190ms/step - loss: 10.2170 -
mae: 2.4725 - val_loss: 9.7800 - val_mae: 2.4502
Epoch 3/10
819/819 [==============================] - 159s 194ms/step - loss: 8.4920 - mae:
2.2760 - val_loss: 9.7862 - val_mae: 2.4567
Epoch 4/10
819/819 [==============================] - 156s 190ms/step - loss: 7.7367 - mae:
2.1745 - val_loss: 10.4851 - val_mae: 2.5470
Epoch 5/10
819/819 [==============================] - 156s 190ms/step - loss: 7.3078 - mae:
2.1106 - val_loss: 10.3408 - val_mae: 2.5241
Epoch 6/10
819/819 [==============================] - 157s 191ms/step - loss: 6.9703 - mae:
2.0611 - val_loss: 10.9750 - val_mae: 2.6015
Epoch 7/10
819/819 [==============================] - 161s 196ms/step - loss: 6.6779 - mae:
2.0155 - val_loss: 11.3270 - val_mae: 2.6359
Epoch 8/10
819/819 [==============================] - 160s 195ms/step - loss: 6.4582 - mae:
1.9800 - val_loss: 10.9980 - val_mae: 2.6058
Epoch 9/10
819/819 [==============================] - 160s 195ms/step - loss: 6.1876 - mae:
```

```
1.9350 - val_loss: 11.0461 - val_mae: 2.6188
Epoch 10/10
819/819 [==============================] - 162s 198ms/step - loss: 6.0024 - mae:
1.9034 - val_loss: 11.8752 - val_mae: 2.7148
405/405 [==============================] - 33s 78ms/step - loss: 11.0372 - mae:
2.6081
Test MAE: 2.61
```

[52]:
```
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()
```



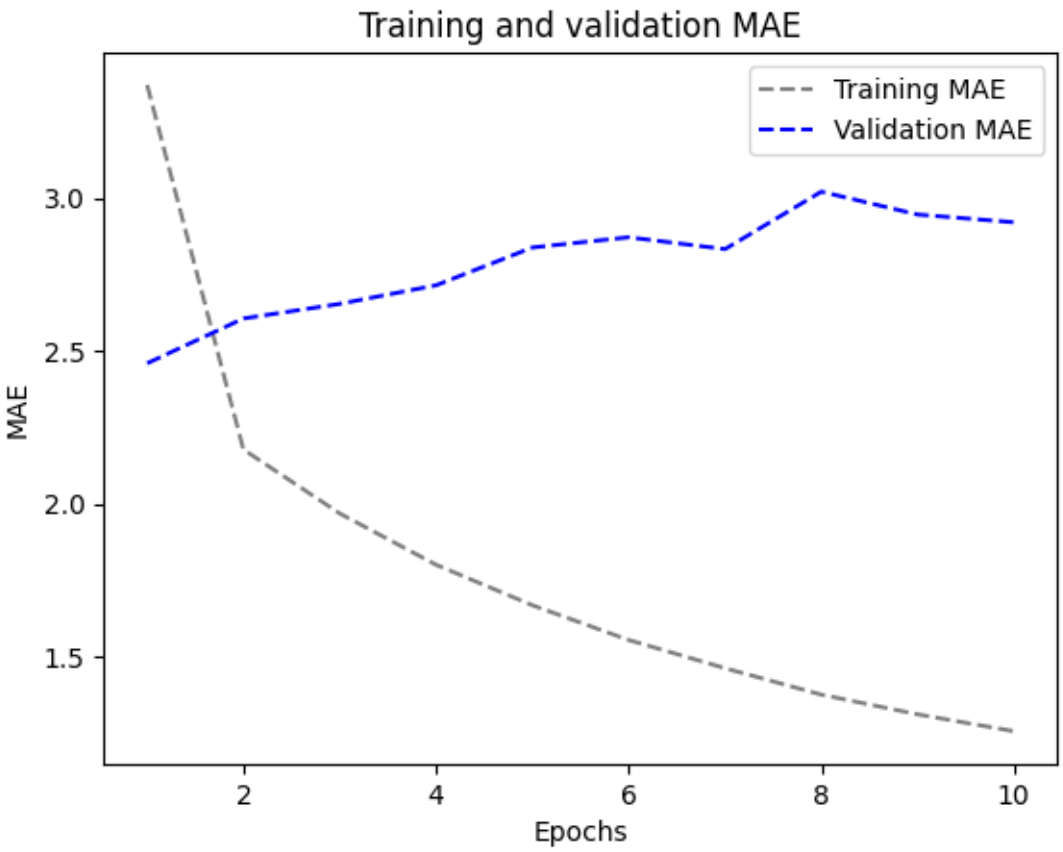
4.LSTM - Stacked setup with 32 units

[53]:
```
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 [==============================] - 246s 294ms/step - loss: 22.3836 -
mae: 3.3688 - val_loss: 10.0397 - val_mae: 2.4590
Epoch 2/10
819/819 [==============================] - 247s 302ms/step - loss: 7.8395 - mae:
2.1771 - val_loss: 10.9510 - val_mae: 2.6056
Epoch 3/10
819/819 [==============================] - 243s 297ms/step - loss: 6.4739 - mae:
1.9681 - val_loss: 11.2825 - val_mae: 2.6534
Epoch 4/10
819/819 [==============================] - 244s 298ms/step - loss: 5.4664 - mae:
1.8004 - val_loss: 11.9034 - val_mae: 2.7142
Epoch 5/10
819/819 [==============================] - 246s 300ms/step - loss: 4.7370 - mae:
1.6680 - val_loss: 12.9674 - val_mae: 2.8377
Epoch 6/10
819/819 [==============================] - 243s 297ms/step - loss: 4.1355 - mae:
1.5543 - val_loss: 13.1681 - val_mae: 2.8711
Epoch 7/10
819/819 [==============================] - 278s 339ms/step - loss: 3.6759 - mae:
1.4621 - val_loss: 12.8662 - val_mae: 2.8325
Epoch 8/10
819/819 [==============================] - 240s 293ms/step - loss: 3.2677 - mae:
1.3757 - val_loss: 14.5585 - val_mae: 3.0203
Epoch 9/10
```

```
819/819 [==============================] - 240s 293ms/step - loss: 2.9740 - mae:
1.3113 - val_loss: 13.9293 - val_mae: 2.9454
Epoch 10/10
819/819 [==============================] - 240s 292ms/step - loss: 2.7346 - mae:
1.2561 - val_loss: 13.6646 - val_mae: 2.9203
405/405 [==============================] - 47s 113ms/step - loss: 12.1486 - mae:
2.7199
Test MAE: 2.72
```

[54]:
```
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()
```



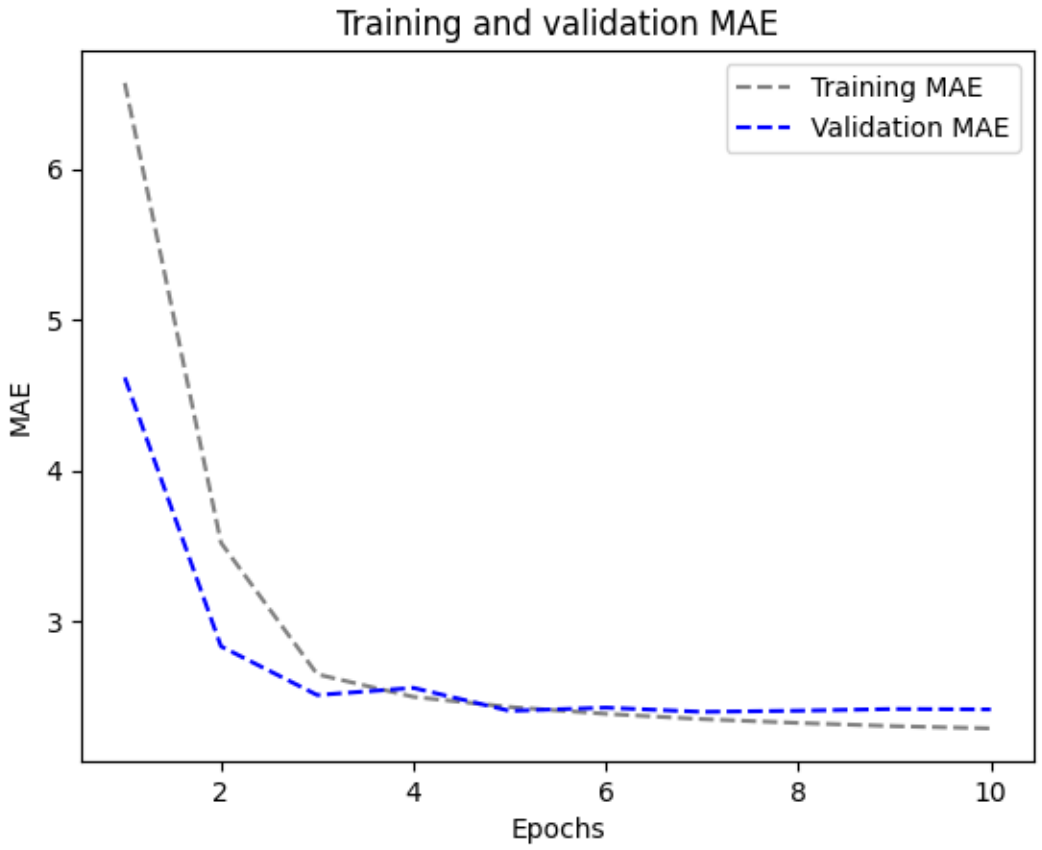
4.LSTM - Stacked setup with 8 units

[55]:
```
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 [==============================] - 139s 165ms/step - loss: 73.6114 -
mae: 6.5709 - val_loss: 38.3792 - val_mae: 4.6154
Epoch 2/10
819/819 [==============================] - 132s 161ms/step - loss: 22.7957 -
mae: 3.5191 - val_loss: 14.6470 - val_mae: 2.8314
Epoch 3/10
819/819 [==============================] - 132s 161ms/step - loss: 11.8090 -
mae: 2.6461 - val_loss: 11.2976 - val_mae: 2.5082
Epoch 4/10
819/819 [==============================] - 131s 160ms/step - loss: 10.3273 -
mae: 2.4961 - val_loss: 11.3836 - val_mae: 2.5559
Epoch 5/10
819/819 [==============================] - 149s 182ms/step - loss: 9.7267 - mae:
2.4291 - val_loss: 9.5559 - val_mae: 2.4032
Epoch 6/10
819/819 [==============================] - 131s 159ms/step - loss: 9.3704 - mae:
2.3836 - val_loss: 9.7530 - val_mae: 2.4246
Epoch 7/10
819/819 [==============================] - 130s 159ms/step - loss: 9.1002 - mae:
2.3485 - val_loss: 9.5921 - val_mae: 2.3968
Epoch 8/10
819/819 [==============================] - 149s 182ms/step - loss: 8.9007 - mae:
2.3232 - val_loss: 9.6653 - val_mae: 2.4044
Epoch 9/10
819/819 [==============================] - 131s 160ms/step - loss: 8.7325 - mae:
2.3017 - val_loss: 9.8270 - val_mae: 2.4158
Epoch 10/10
819/819 [==============================] - 129s 158ms/step - loss: 8.6133 - mae:
2.2870 - val_loss: 9.7644 - val_mae: 2.4128
405/405 [==============================] - 28s 66ms/step - loss: 10.9293 - mae:
2.5841
Test MAE: 2.58
```

[56]:
```
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

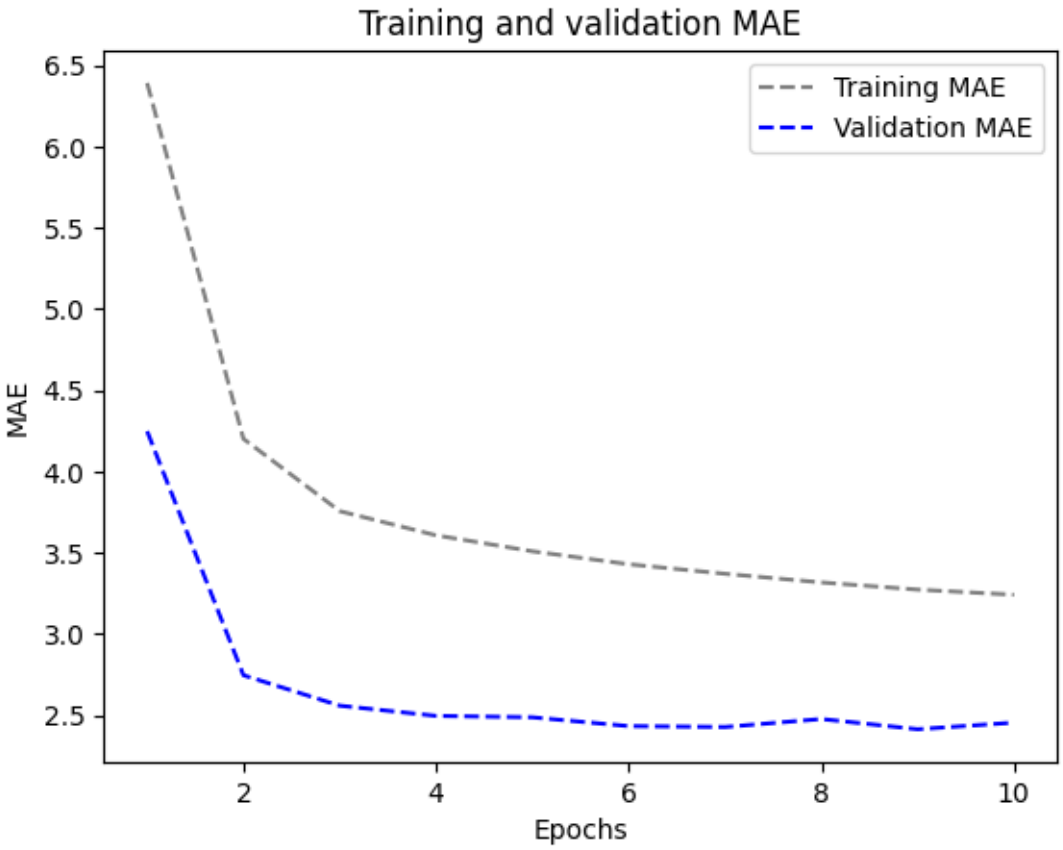
[57]:
```
inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.LSTM(8, recurrent_dropout=0.5, return_sequences=True)(inputs)
x = layers.LSTM(8, recurrent_dropout=0.5)(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1)(x)
model = keras.Model(inputs, outputs)

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 [==============================] - 240s 287ms/step - loss: 69.3434 -
mae: 6.3897 - val_loss: 32.2885 - val_mae: 4.2474
Epoch 2/10
819/819 [==============================] - 238s 290ms/step - loss: 31.2629 -
mae: 4.2008 - val_loss: 13.4545 - val_mae: 2.7463
Epoch 3/10
819/819 [==============================] - 241s 294ms/step - loss: 24.5491 -
mae: 3.7559 - val_loss: 11.1259 - val_mae: 2.5590
Epoch 4/10
819/819 [==============================] - 241s 294ms/step - loss: 22.4875 -
mae: 3.6073 - val_loss: 10.4511 - val_mae: 2.4963
Epoch 5/10
819/819 [==============================] - 251s 306ms/step - loss: 21.2447 -
mae: 3.5089 - val_loss: 10.3157 - val_mae: 2.4875
Epoch 6/10
819/819 [==============================] - 240s 293ms/step - loss: 20.2196 -
mae: 3.4293 - val_loss: 9.9142 - val_mae: 2.4334
Epoch 7/10
819/819 [==============================] - 235s 286ms/step - loss: 19.5627 -
mae: 3.3705 - val_loss: 9.8420 - val_mae: 2.4278
Epoch 8/10
819/819 [==============================] - 238s 290ms/step - loss: 18.9018 -
mae: 3.3175 - val_loss: 10.1269 - val_mae: 2.4770
Epoch 9/10
819/819 [==============================] - 236s 288ms/step - loss: 18.3804 -
mae: 3.2732 - val_loss: 9.6429 - val_mae: 2.4142
Epoch 10/10
819/819 [==============================] - 234s 285ms/step - loss: 18.0501 -
mae: 3.2416 - val_loss: 9.9505 - val_mae: 2.4547
405/405 [==============================] - 28s 68ms/step - loss: 11.0511 - mae:
2.5783
Test MAE: 2.58
```

[58]:
```
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()
```



Bidirectional LSTM

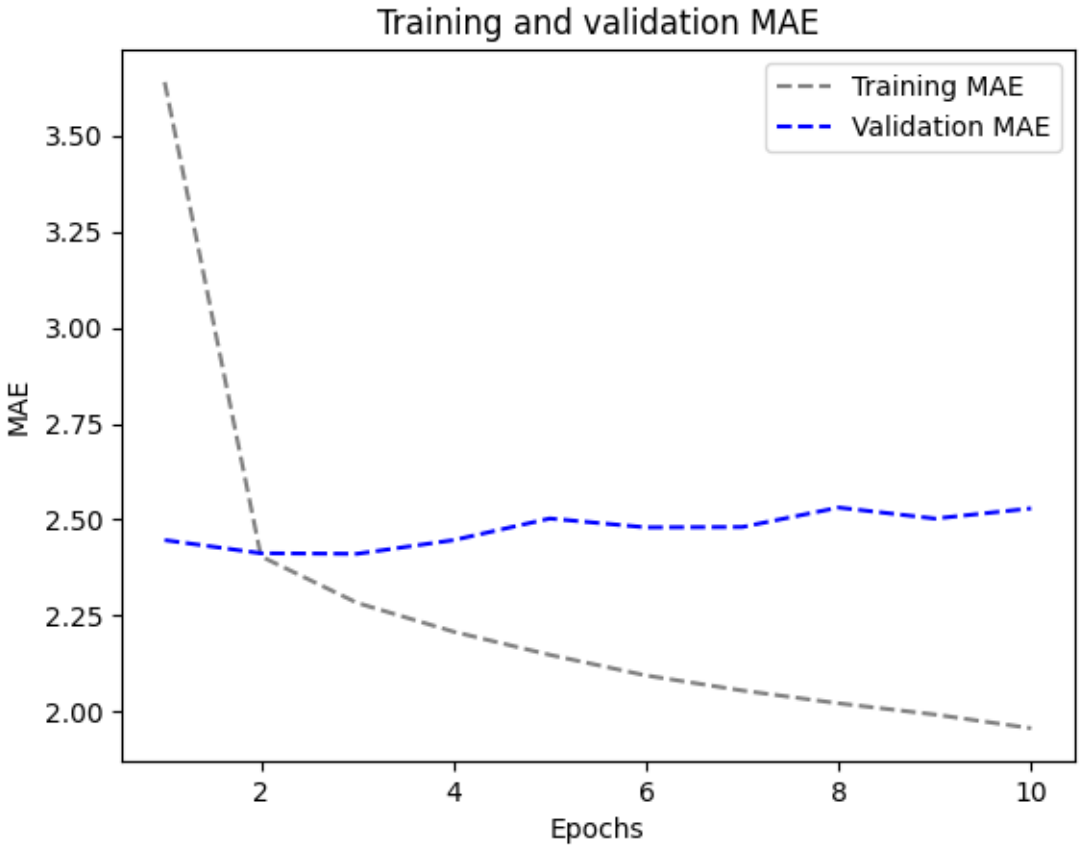
[59]:
```
inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.Bidirectional(layers.LSTM(16))(inputs)
outputs = layers.Dense(1)(x)
model = keras.Model(inputs, outputs)

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 [==============================] - 139s 165ms/step - loss: 25.4954 -
mae: 3.6385 - val_loss: 9.9810 - val_mae: 2.4468
Epoch 2/10
819/819 [==============================] - 135s 164ms/step - loss: 9.5457 - mae:
2.4046 - val_loss: 9.5946 - val_mae: 2.4126
Epoch 3/10
819/819 [==============================] - 139s 170ms/step - loss: 8.6401 - mae:
2.2834 - val_loss: 9.6844 - val_mae: 2.4112
Epoch 4/10
819/819 [==============================] - 135s 165ms/step - loss: 8.0613 - mae:
2.2086 - val_loss: 9.9892 - val_mae: 2.4467
Epoch 5/10
819/819 [==============================] - 134s 164ms/step - loss: 7.6328 - mae:
2.1483 - val_loss: 10.5241 - val_mae: 2.5030
Epoch 6/10
819/819 [==============================] - 135s 165ms/step - loss: 7.2579 - mae:
2.0947 - val_loss: 10.3817 - val_mae: 2.4799
Epoch 7/10
819/819 [==============================] - 137s 167ms/step - loss: 6.9821 - mae:
2.0555 - val_loss: 10.1949 - val_mae: 2.4809
Epoch 8/10
819/819 [==============================] - 133s 163ms/step - loss: 6.7603 - mae:
2.0222 - val_loss: 10.7194 - val_mae: 2.5315
Epoch 9/10
819/819 [==============================] - 133s 162ms/step - loss: 6.5590 - mae:
1.9930 - val_loss: 10.5383 - val_mae: 2.5028
Epoch 10/10
819/819 [==============================] - 135s 164ms/step - loss: 6.3415 - mae:
1.9574 - val_loss: 10.6848 - val_mae: 2.5293
405/405 [==============================] - 29s 70ms/step - loss: 10.9508 - mae:
2.5886
Test MAE: 2.59
```

[60]:
```
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()
```



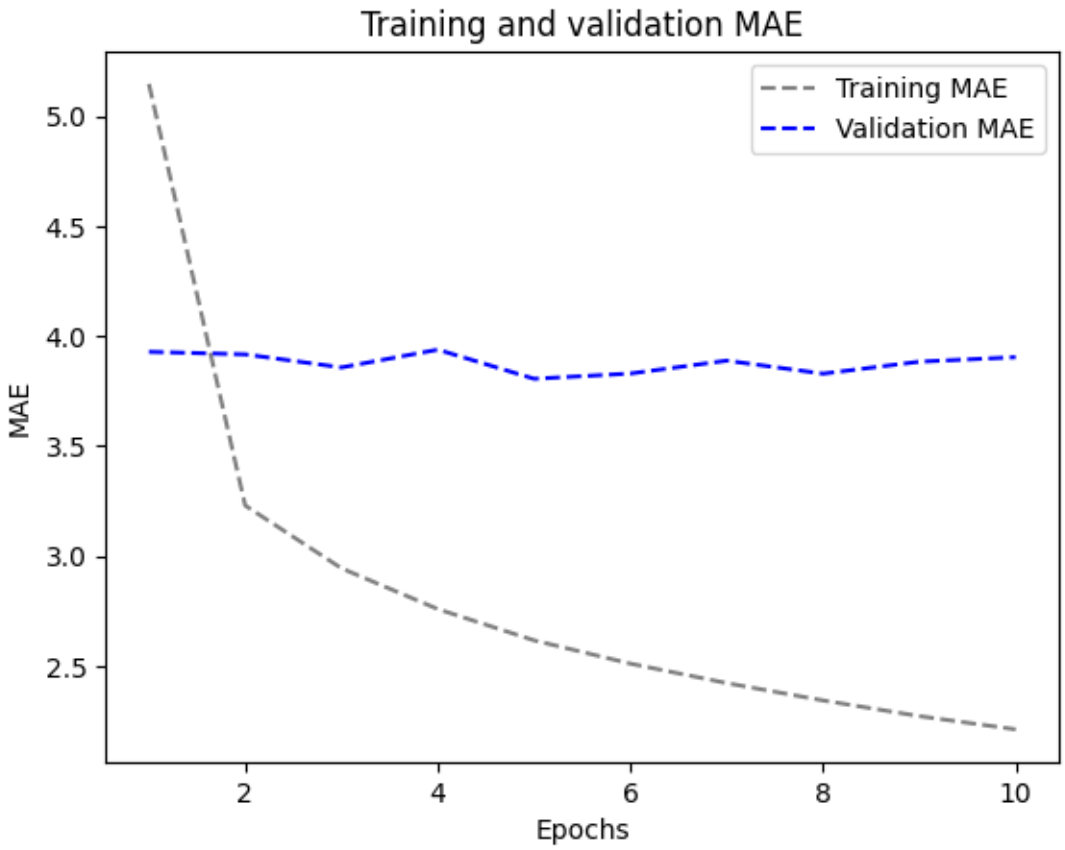
1D Convnets and LSTM togther

[61]:
```
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 [==============================] - 107s 126ms/step - loss: 47.4034 -
mae: 5.1465 - val_loss: 25.5359 - val_mae: 3.9287
Epoch 2/10
819/819 [==============================] - 102s 125ms/step - loss: 17.4520 -
mae: 3.2298 - val_loss: 25.3732 - val_mae: 3.9170
Epoch 3/10
819/819 [==============================] - 104s 126ms/step - loss: 14.4247 -
mae: 2.9443 - val_loss: 24.4947 - val_mae: 3.8577
Epoch 4/10
819/819 [==============================] - 104s 127ms/step - loss: 12.7033 -
mae: 2.7590 - val_loss: 25.3368 - val_mae: 3.9385
Epoch 5/10
819/819 [==============================] - 105s 128ms/step - loss: 11.4676 -
mae: 2.6167 - val_loss: 23.6083 - val_mae: 3.8060
Epoch 6/10
819/819 [==============================] - 104s 126ms/step - loss: 10.6234 -
mae: 2.5109 - val_loss: 22.9505 - val_mae: 3.8300
Epoch 7/10
819/819 [==============================] - 103s 125ms/step - loss: 9.8894 - mae:
2.4225 - val_loss: 24.6421 - val_mae: 3.8885
Epoch 8/10
819/819 [==============================] - 106s 129ms/step - loss: 9.3317 - mae:
2.3436 - val_loss: 23.3066 - val_mae: 3.8294
Epoch 9/10
819/819 [==============================] - 105s 128ms/step - loss: 8.7825 - mae:
2.2722 - val_loss: 24.3439 - val_mae: 3.8838
Epoch 10/10
819/819 [==============================] - 103s 126ms/step - loss: 8.3556 - mae:
2.2121 - val_loss: 24.2627 - val_mae: 3.9042
405/405 [==============================] - 21s 49ms/step - loss: 23.8137 - mae:
3.9080
Test MAE: 3.91
```

[62]:
```
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()
```



[63]:
```
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()
```