In [5]:
```
!pip install tensorflow==2.12
```

```
Collecting tensorflow==2.12
  Downloading tensorflow-2.12.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x8
6_64.whl (585.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 585.9/585.9 MB 1.1 MB/s eta 0:00:00
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-pa
ckages (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/di
st-packages (from tensorflow==2.12) (0.2.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/di
st-packages (from tensorflow==2.12) (1.62.1)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packa
ges (from tensorflow==2.12) (3.9.0)
Requirement already satisfied: jax>=0.3.15 in /usr/local/lib/python3.10/dist-packa
ges (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 68.3 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 36.8 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-package
s (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 ten
sorflow==2.12) (3.20.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packag
es (from tensorflow==2.12) (67.7.2)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packa
ges (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 49.4 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 37.0 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/di
st-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/dis
t-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-packag
es (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->tensorflo
w==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 pack
ages 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 incompatibl
e.
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 wh
ich 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
```

In [1]:

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

```
--2024-04-06 18:11:01--  https://s3.amazonaws.com/keras-datasets/jena_climate_2009
_2016.csv.zip
Resolving s3.amazonaws.com (s3.amazonaws.com)... 52.216.251.54, 54.231.224.128, 5
2.216.44.136, ...
Connecting to s3.amazonaws.com (s3.amazonaws.com)|52.216.251.54|: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  40.1MB/s    in 0.3s

2024-04-06 18:11:01 (40.1 MB/s) - ‘jena_climate_2009_2016.csv.zip’ saved [1356564
2/13565642]

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

Examining the contents of the Jena weather dataset reveals 420,451 rows and 15 attributes.

In [2]:

```
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
```

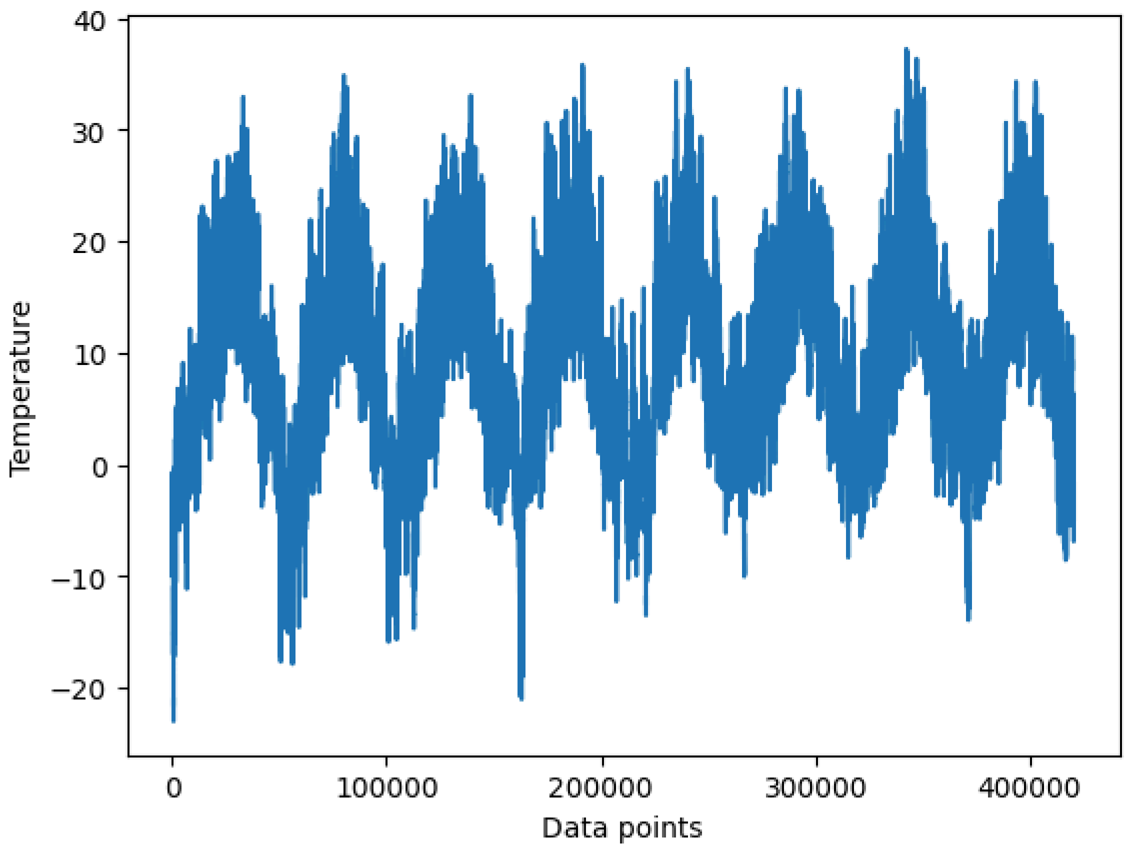
Parsing the data- The process of parsing the data involves transforming the values separated by commas into floating-point integers. Specific values are then stored for later processing or analysis in the raw_data and temperature arrays.

In [3]:
```
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[:]
```

creating a graphical representation of the temperature timeseries.

In [4]:
```
from matplotlib import pyplot as plt
plt.plot(range(len(Temperature)), Temperature)
plt.xlabel('Data points')
plt.ylabel('Temperature')
```

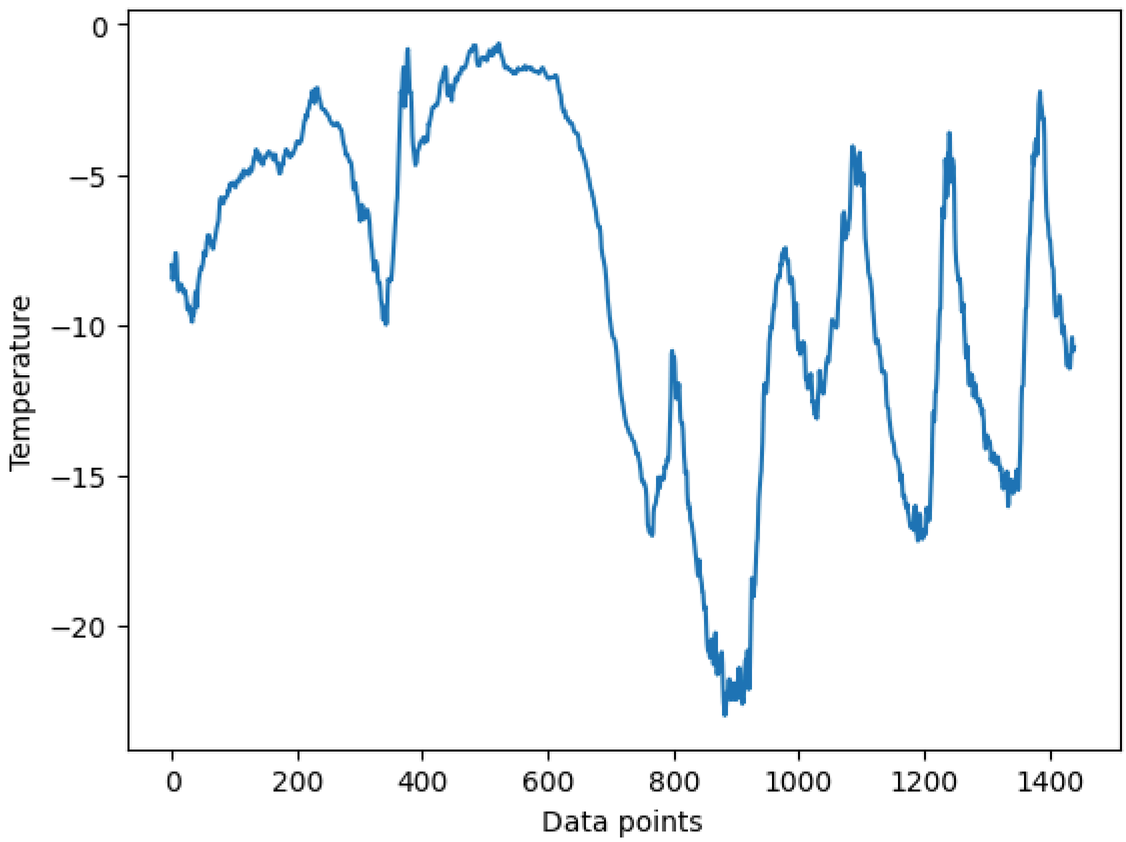
Out[4]:
```
Text(0, 0.5, 'Temperature')
```



Generating a plot for the temperature timeseries data for the initial 10 days, considering that each day consists of 144 data points, resulting in a total of 1440 data points for the specified period.

In [5]:
```
plt.plot(range(1440), Temperature[:1440])
plt.xlabel('Data points')
plt.ylabel('Temperature')
```

Out[5]:
```
Text(0, 0.5, 'Temperature')
```



Determining the quantity of samples allocated for each division of data: 50% designated for training, and 25% for validation.

In [6]:
```
number_train_samples = int(0.5 * len(raw_data))
number_val_samples = int(0.25 * len(raw_data))
number_test_samples = len(raw_data) - number_train_samples - number_val_samples
print("number_train_samples:", number_train_samples)
print("number_val_samples:", number_val_samples)
print("number_test_samples:", number_test_samples)
```

```
number_train_samples: 210225
number_val_samples: 105112
number_test_samples: 105114
```

Preparing the data

Normalizing the data- Vectorization is not required because the data is already in a numerical representation. Nevertheless, it is advised to normalize all variables because the data scales vary across them, with temperature ranging from -20 to +30 and pressure recorded in millibars.

In [7]:
```
mean = raw_data[:number_train_samples].mean(axis=0)
raw_data -= mean
std = raw_data[:number_train_samples].std(axis=0)
raw_data /= std
```

In [8]:
```
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 - This is necessary due to the dataset's highly redundant nature. Allocating memory for each individual sample explicitly would be inefficient. Therefore, we opt to generate the samples dynamically.

In [9]:
```
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=number_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=number_train_samples,
    end_index=number_train_samples + number_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=number_train_samples + number_val_samples)
```

Examining the output of one of our datasets

In [10]:
```
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,)
```

A common-sense, non-machine-learning baseline

Computing the common-sense baseline MAE - The function "evaluate_naive_method" establishes a baseline for assessing the effectiveness of a straightforward forecasting method. This method involves using the last value in the input sequence as the prediction for the subsequent value..

In [11]:
```
def evaluating_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: {evaluating_naive_method(val_dataset):.2f}")
print(f"Test MAE: {evaluating_naive_method(test_dataset):.2f}")
```

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

Predicting that the temperature in the next 24 hours will match the current temperature is a commonsense baseline approach. The Mean Absolute Error (MAE) for testing is 2.62 degrees Celsius, while for validation, it is 2.44 degrees Celsius using this straightforward baseline. Essentially, this means that anticipating the temperature to stay constant would lead to an average deviation of roughly two and a half degrees.

A basic machine-learning model - Dense Layer

Training and assessing a model with densely connected layers.

In [12]:
```
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)
```

In [13]:
```
callbacks = [
    keras.callbacks.ModelCheckpoint("jena_dense.keras",
                                    save_best_only=True)]
```

In [14]:
```
model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
```

In [15]:
```
history = model.fit(train_dataset, epochs=10,
                    validation_data = val_dataset, callbacks=callbacks)
```

```
Epoch 1/10
819/819 [==============================] - 56s 67ms/step - loss: 12.9800 - mae: 2.
7699 - val_loss: 10.5169 - val_mae: 2.5670
Epoch 2/10
819/819 [==============================] - 51s 62ms/step - loss: 9.0147 - mae: 2.3
626 - val_loss: 11.9154 - val_mae: 2.7414
Epoch 3/10
819/819 [==============================] - 51s 62ms/step - loss: 8.4020 - mae: 2.2
816 - val_loss: 10.9421 - val_mae: 2.6231
Epoch 4/10
819/819 [==============================] - 56s 68ms/step - loss: 7.9881 - mae: 2.2
247 - val_loss: 10.8272 - val_mae: 2.6249
Epoch 5/10
819/819 [==============================] - 56s 68ms/step - loss: 7.7040 - mae: 2.1
848 - val_loss: 10.1424 - val_mae: 2.5290
Epoch 6/10
819/819 [==============================] - 52s 64ms/step - loss: 7.4789 - mae: 2.1
523 - val_loss: 11.5365 - val_mae: 2.7059
Epoch 7/10
819/819 [==============================] - 51s 62ms/step - loss: 7.2702 - mae: 2.1
233 - val_loss: 10.5281 - val_mae: 2.5771
Epoch 8/10
819/819 [==============================] - 57s 69ms/step - loss: 7.1005 - mae: 2.0
964 - val_loss: 11.2295 - val_mae: 2.6710
Epoch 9/10
819/819 [==============================] - 52s 64ms/step - loss: 6.9426 - mae: 2.0
748 - val_loss: 11.0684 - val_mae: 2.6338
Epoch 10/10
819/819 [==============================] - 52s 63ms/step - loss: 6.8377 - mae: 2.0
558 - val_loss: 11.2249 - val_mae: 2.6504
```

In [16]:

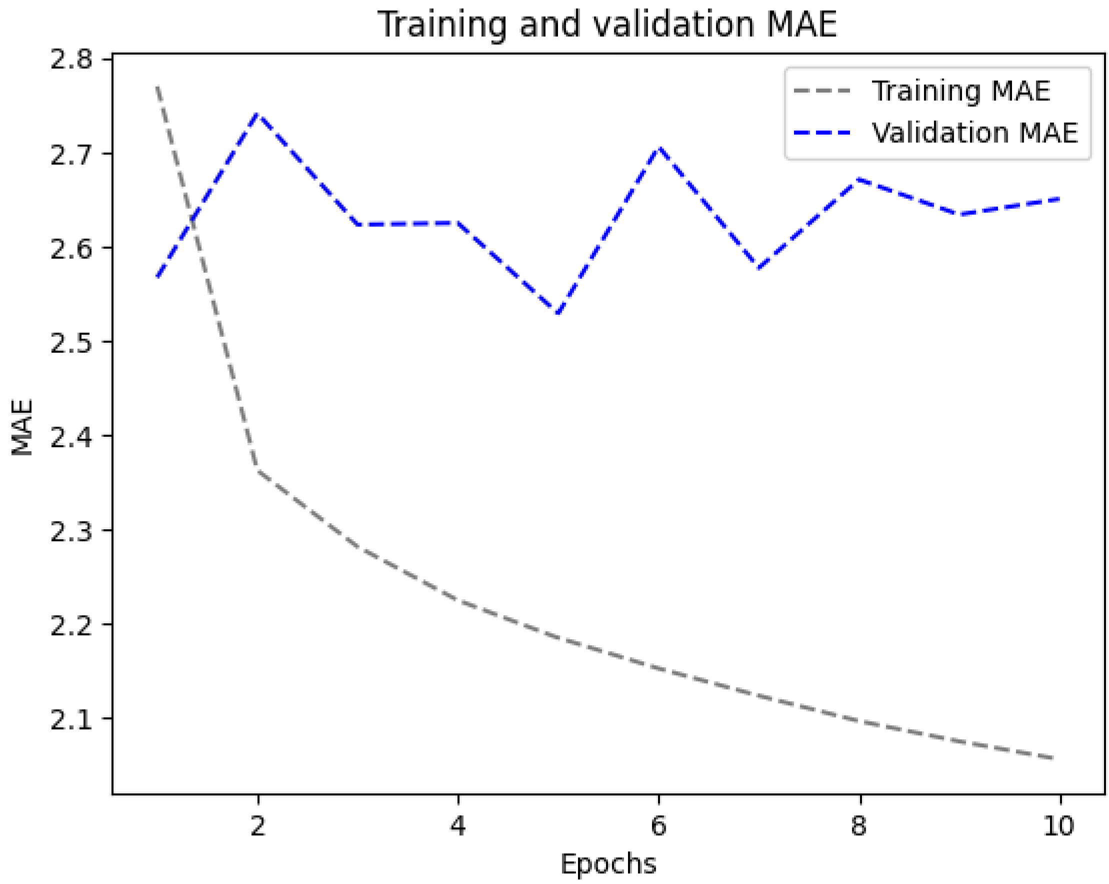
```
model = keras.models.load_model("jena_dense.keras")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
405/405 [==============================] - 21s 50ms/step - loss: 11.4907 - mae: 2.
6701
Test MAE: 2.67
```

Plotting the results

In [17]:

```
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()
```



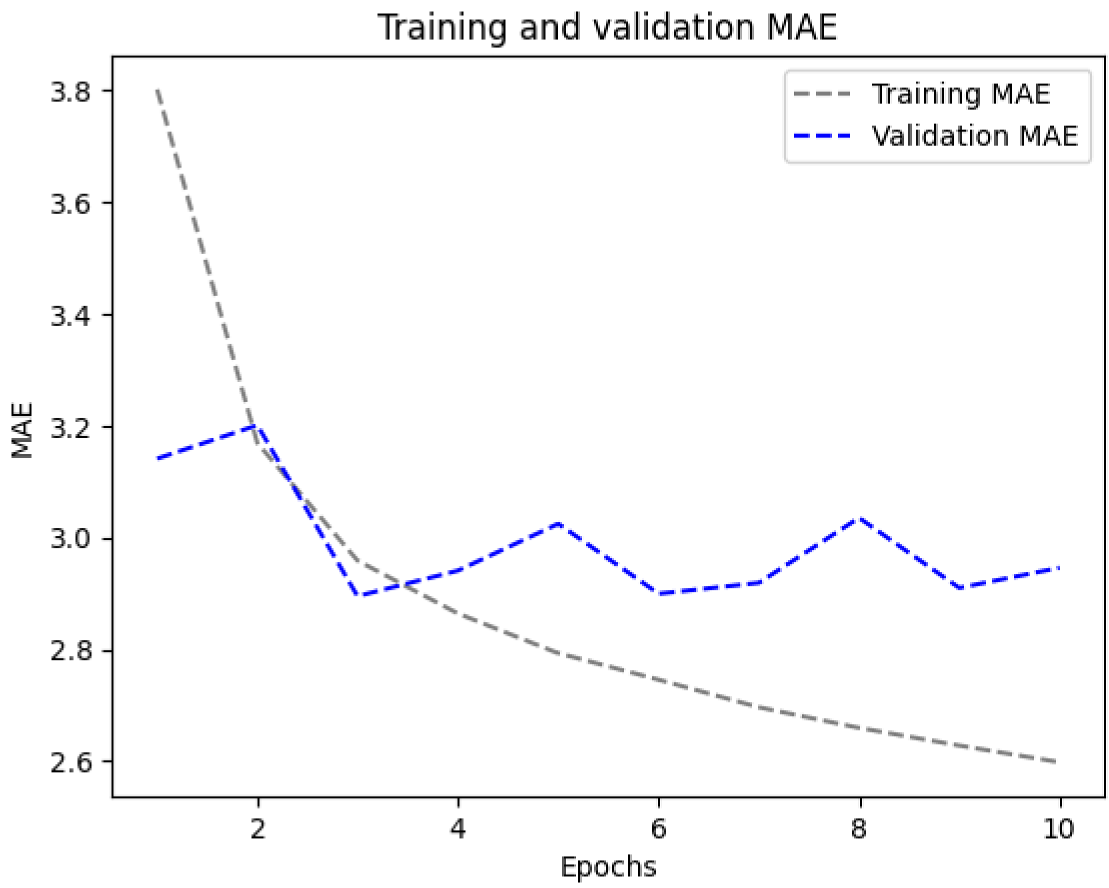
We'll experiment with a 1-dimensional convolutional model.

In [18]:
```
inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
a = layers.Conv1D(8, 24, activation="relu")(inputs)
a = layers.MaxPooling1D(2)(a)
a = layers.Conv1D(8, 12, activation="relu")(a)
a = layers.MaxPooling1D(2)(a)
a = layers.Conv1D(8, 6, activation="relu")(a)
a = layers.GlobalAveragePooling1D()(a)
outputs = layers.Dense(1)(a)
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)5645321

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

```
Epoch 1/10
819/819 [==============================] - 96s 116ms/step - loss: 23.3257 - mae:
3.8011 - val_loss: 16.0054 - val_mae: 3.1411
Epoch 2/10
819/819 [==============================] - 93s 114ms/step - loss: 15.9652 - mae:
3.1690 - val_loss: 16.6596 - val_mae: 3.2009
Epoch 3/10
819/819 [==============================] - 122s 148ms/step - loss: 13.9398 - mae:
2.9587 - val_loss: 13.5656 - val_mae: 2.8952
Epoch 4/10
819/819 [==============================] - 98s 119ms/step - loss: 13.0945 - mae:
2.8644 - val_loss: 14.0583 - val_mae: 2.9407
Epoch 5/10
819/819 [==============================] - 94s 115ms/step - loss: 12.4856 - mae:
2.7933 - val_loss: 14.9615 - val_mae: 3.0244
Epoch 6/10
819/819 [==============================] - 97s 118ms/step - loss: 12.0650 - mae:
2.7458 - val_loss: 13.6935 - val_mae: 2.8995
Epoch 7/10
819/819 [==============================] - 121s 148ms/step - loss: 11.6573 - mae:
2.6965 - val_loss: 13.7937 - val_mae: 2.9190
Epoch 8/10
819/819 [==============================] - 109s 133ms/step - loss: 11.3520 - mae:
2.6597 - val_loss: 15.0747 - val_mae: 3.0352
Epoch 9/10
819/819 [==============================] - 96s 116ms/step - loss: 11.0859 - mae:
2.6281 - val_loss: 13.7297 - val_mae: 2.9100
Epoch 10/10
819/819 [==============================] - 94s 115ms/step - loss: 10.8123 - mae:
2.5984 - val_loss: 14.1641 - val_mae: 2.9460
405/405 [==============================] - 23s 55ms/step - loss: 15.1228 - mae: 3.
0824
Test MAE: 3.08
```

In [19]:

```
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()
```



The performance of the convolutional model appears to be inferior compared to the straightforward or densely connected model. This could be attributed to two main factors:

The assumption of translation invariance doesn't align well with weather data.

The sequential order of the data is crucial. Recent past data holds much more predictive value for forecasting the temperature of the following day compared to older data. Regrettably, a 1D convolutional neural network lacks the ability to adequately capture this significant temporal order.

A Simple RNN

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

In [20]:
```
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 97ms/step - loss: 138.8651 - mae:
9.7006 - val_loss: 144.0883 - val_mae: 9.9072
Epoch 2/10
819/819 [==============================] - 81s 98ms/step - loss: 136.3884 - mae:
9.5630 - val_loss: 143.7894 - val_mae: 9.8792
Epoch 3/10
819/819 [==============================] - 78s 95ms/step - loss: 136.2441 - mae:
9.5492 - val_loss: 143.6132 - val_mae: 9.8625
Epoch 4/10
819/819 [==============================] - 79s 96ms/step - loss: 136.1973 - mae:
9.5452 - val_loss: 143.5972 - val_mae: 9.8574
Epoch 5/10
819/819 [==============================] - 86s 104ms/step - loss: 136.1420 - mae:
9.5367 - val_loss: 143.5690 - val_mae: 9.8543
Epoch 6/10
819/819 [==============================] - 82s 100ms/step - loss: 136.1186 - mae:
9.5341 - val_loss: 143.5271 - val_mae: 9.8496
Epoch 7/10
819/819 [==============================] - 80s 97ms/step - loss: 136.1018 - mae:
9.5323 - val_loss: 143.5374 - val_mae: 9.8501
Epoch 8/10
819/819 [==============================] - 78s 95ms/step - loss: 136.0968 - mae:
9.5313 - val_loss: 143.5265 - val_mae: 9.8494
Epoch 9/10
819/819 [==============================] - 84s 103ms/step - loss: 136.0802 - mae:
9.5294 - val_loss: 143.5291 - val_mae: 9.8499
Epoch 10/10
819/819 [==============================] - 80s 97ms/step - loss: 136.0748 - mae:
9.5289 - val_loss: 143.4970 - val_mae: 9.8465
405/405 [==============================] - 21s 50ms/step - loss: 151.2585 - mae:
9.9151
Test MAE: 9.92
```

2.Simple RNN - Stacking RNN layers

In [21]:

```
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 [==============================] - 161s 194ms/step - loss: 136.7681 - mae:
9.5635 - val_loss: 143.4606 - val_mae: 9.8445
Epoch 2/10
819/819 [==============================] - 159s 193ms/step - loss: 135.9636 - mae:
9.5133 - val_loss: 143.4017 - val_mae: 9.8343
Epoch 3/10
819/819 [==============================] - 157s 191ms/step - loss: 135.9102 - mae:
9.5067 - val_loss: 143.4708 - val_mae: 9.8417
Epoch 4/10
819/819 [==============================] - 159s 194ms/step - loss: 135.8834 - mae:
9.5038 - val_loss: 143.3965 - val_mae: 9.8346
Epoch 5/10
819/819 [==============================] - 159s 193ms/step - loss: 135.8572 - mae:
9.4994 - val_loss: 143.3912 - val_mae: 9.8341
Epoch 6/10
819/819 [==============================] - 159s 193ms/step - loss: 135.8478 - mae:
9.4980 - val_loss: 143.3914 - val_mae: 9.8333
Epoch 7/10
819/819 [==============================] - 145s 177ms/step - loss: 135.8311 - mae:
9.4950 - val_loss: 143.4005 - val_mae: 9.8357
Epoch 8/10
819/819 [==============================] - 143s 174ms/step - loss: 135.8168 - mae:
9.4928 - val_loss: 143.4048 - val_mae: 9.8354
Epoch 9/10
819/819 [==============================] - 144s 175ms/step - loss: 135.8049 - mae:
9.4908 - val_loss: 143.4118 - val_mae: 9.8359
Epoch 10/10
819/819 [==============================] - 144s 175ms/step - loss: 135.7988 - mae:
9.4898 - val_loss: 143.4327 - val_mae: 9.8398
405/405 [==============================] - 31s 75ms/step - loss: 151.1230 - mae:
9.9009
Test MAE: 9.90
```

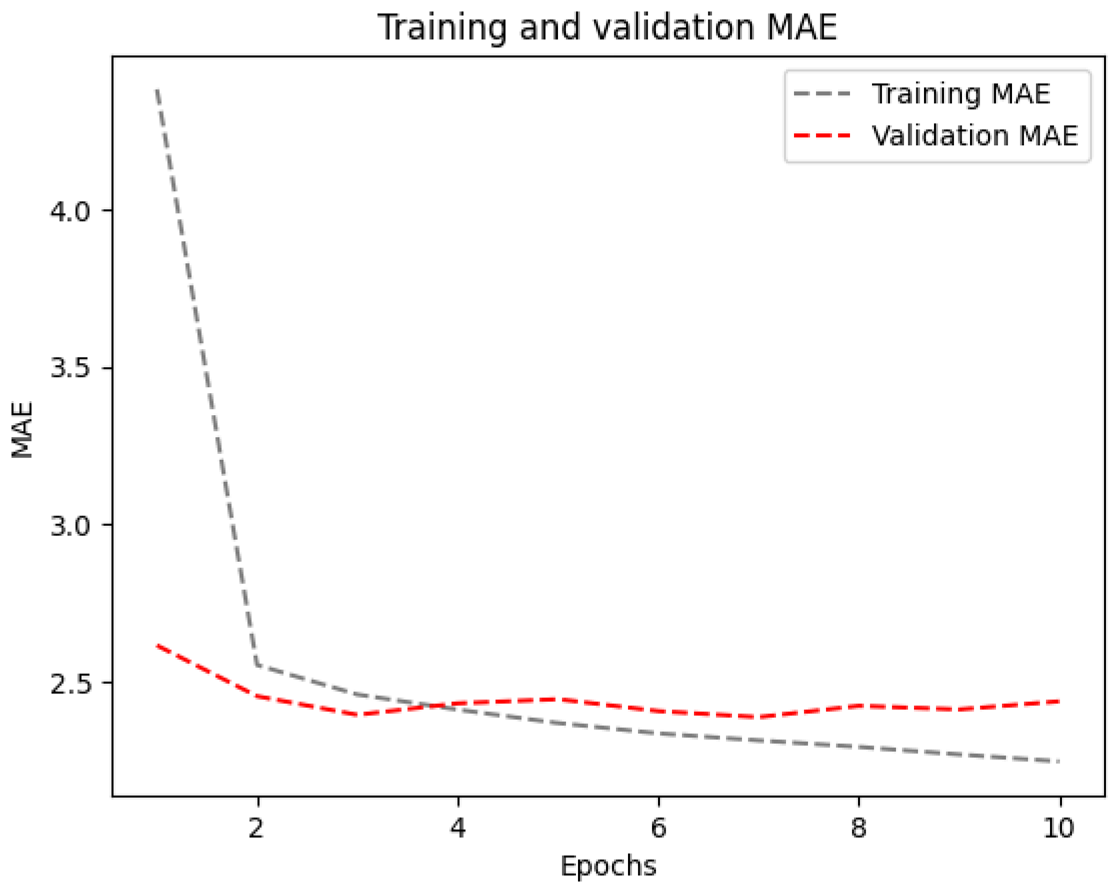
A Simple GRU (Gated Recurrent Unit)

In [22]:
```
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 [==============================] - 141s 168ms/step - loss: 36.3609 - mae:
4.3810 - val_loss: 11.8900 - val_mae: 2.6164
Epoch 2/10
819/819 [==============================] - 122s 148ms/step - loss: 10.7004 - mae:
2.5542 - val_loss: 10.1259 - val_mae: 2.4551
Epoch 3/10
819/819 [==============================] - 139s 169ms/step - loss: 9.8931 - mae:
2.4605 - val_loss: 9.5997 - val_mae: 2.3959
Epoch 4/10
819/819 [==============================] - 122s 148ms/step - loss: 9.5173 - mae:
2.4125 - val_loss: 10.0118 - val_mae: 2.4324
Epoch 5/10
819/819 [==============================] - 137s 167ms/step - loss: 9.1867 - mae:
2.3698 - val_loss: 10.1581 - val_mae: 2.4455
Epoch 6/10
819/819 [==============================] - 120s 146ms/step - loss: 8.9471 - mae:
2.3369 - val_loss: 9.8362 - val_mae: 2.4077
Epoch 7/10
819/819 [==============================] - 139s 169ms/step - loss: 8.7793 - mae:
2.3147 - val_loss: 9.5360 - val_mae: 2.3886
Epoch 8/10
819/819 [==============================] - 136s 166ms/step - loss: 8.6172 - mae:
2.2941 - val_loss: 9.8912 - val_mae: 2.4245
Epoch 9/10
819/819 [==============================] - 136s 166ms/step - loss: 8.4282 - mae:
2.2701 - val_loss: 9.7921 - val_mae: 2.4130
Epoch 10/10
819/819 [==============================] - 135s 165ms/step - loss: 8.2469 - mae:
2.2480 - val_loss: 9.8877 - val_mae: 2.4387
405/405 [==============================] - 27s 65ms/step - loss: 9.9800 - mae: 2.4
877
Test MAE: 2.49
```

In [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="red",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)

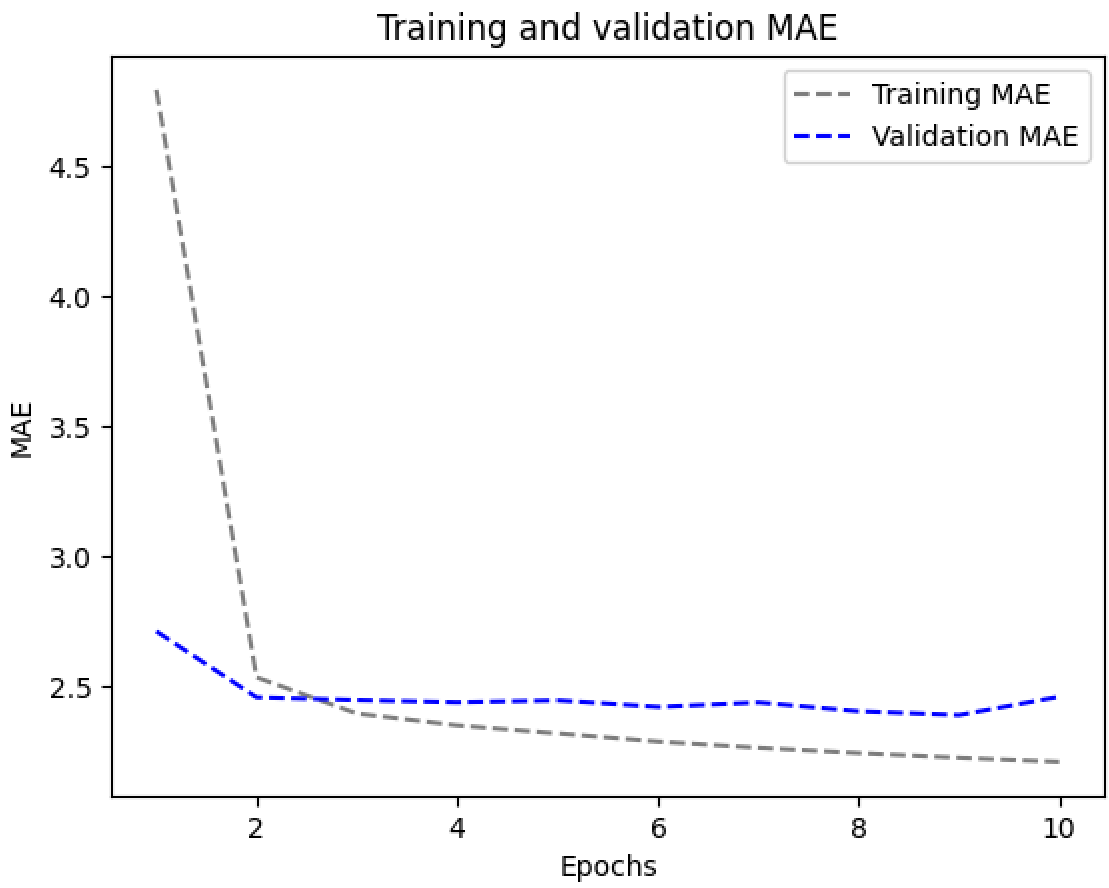
1.LSTM-Simple

In [24]:
```
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 [==============================] - 138s 165ms/step - loss: 43.7437 - mae:
4.7955 - val_loss: 12.8459 - val_mae: 2.7114
Epoch 2/10
819/819 [==============================] - 122s 149ms/step - loss: 10.6991 - mae:
2.5343 - val_loss: 9.9379 - val_mae: 2.4572
Epoch 3/10
819/819 [==============================] - 121s 147ms/step - loss: 9.4754 - mae:
2.3950 - val_loss: 9.8268 - val_mae: 2.4468
Epoch 4/10
819/819 [==============================] - 134s 163ms/step - loss: 9.1170 - mae:
2.3495 - val_loss: 9.7588 - val_mae: 2.4381
Epoch 5/10
819/819 [==============================] - 120s 147ms/step - loss: 8.8801 - mae:
2.3184 - val_loss: 9.8003 - val_mae: 2.4456
Epoch 6/10
819/819 [==============================] - 137s 167ms/step - loss: 8.6615 - mae:
2.2870 - val_loss: 9.6199 - val_mae: 2.4205
Epoch 7/10
819/819 [==============================] - 124s 150ms/step - loss: 8.4874 - mae:
2.2633 - val_loss: 9.7033 - val_mae: 2.4376
Epoch 8/10
819/819 [==============================] - 137s 167ms/step - loss: 8.3282 - mae:
2.2428 - val_loss: 9.5578 - val_mae: 2.4037
Epoch 9/10
819/819 [==============================] - 134s 164ms/step - loss: 8.1744 - mae:
2.2248 - val_loss: 9.5048 - val_mae: 2.3892
Epoch 10/10
819/819 [==============================] - 137s 167ms/step - loss: 8.0484 - mae:
2.2092 - val_loss: 10.0479 - val_mae: 2.4597
405/405 [==============================] - 29s 70ms/step - loss: 10.4045 - mae: 2.
5224
Test MAE: 2.52
```

In [25]:

```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()
```



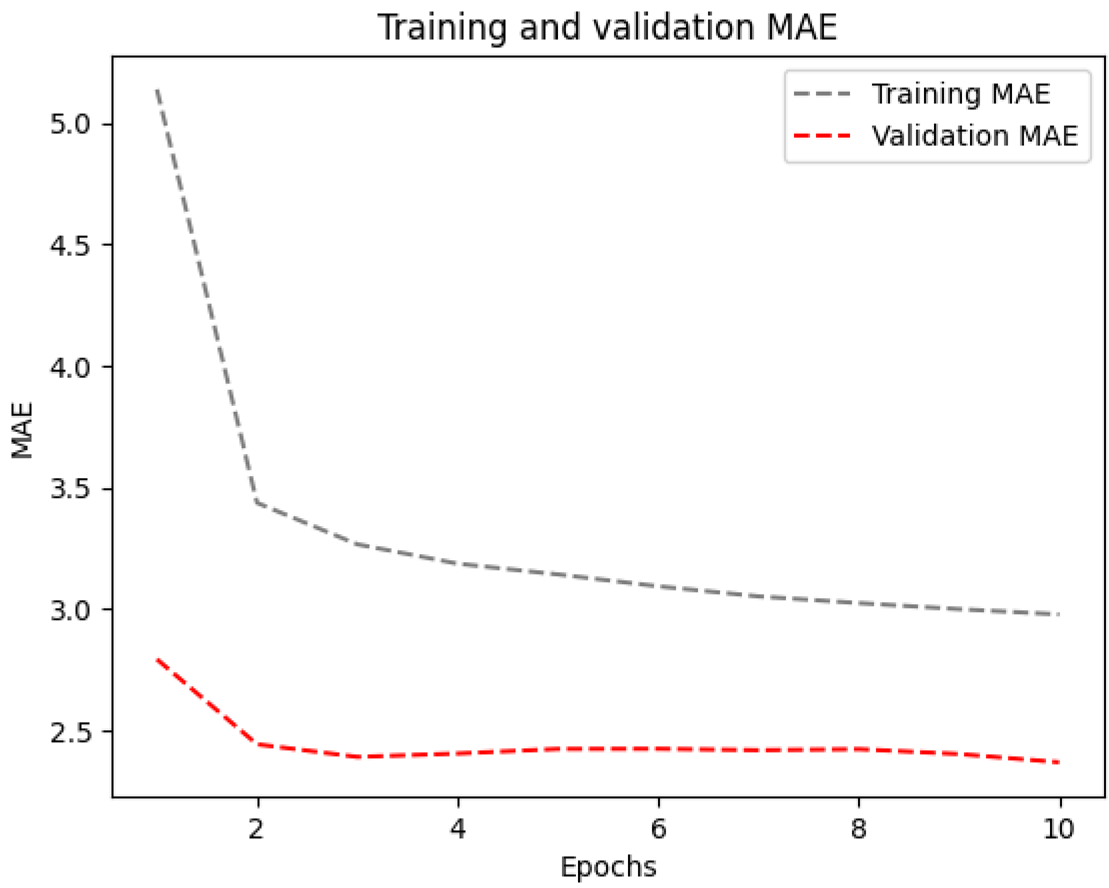
2.LSTM - dropout Regularization

In [26]:
```
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 [==============================] - 207s 248ms/step - loss: 47.5964 - mae:
5.1387 - val_loss: 13.7919 - val_mae: 2.7927
Epoch 2/10
819/819 [==============================] - 190s 231ms/step - loss: 20.0437 - mae:
3.4372 - val_loss: 9.8815 - val_mae: 2.4421
Epoch 3/10
819/819 [==============================] - 205s 250ms/step - loss: 18.0743 - mae:
3.2662 - val_loss: 9.3332 - val_mae: 2.3902
Epoch 4/10
819/819 [==============================] - 191s 232ms/step - loss: 17.1902 - mae:
3.1866 - val_loss: 9.4313 - val_mae: 2.4041
Epoch 5/10
819/819 [==============================] - 191s 233ms/step - loss: 16.6469 - mae:
3.1422 - val_loss: 9.6062 - val_mae: 2.4228
Epoch 6/10
819/819 [==============================] - 190s 231ms/step - loss: 16.1485 - mae:
3.0931 - val_loss: 9.5633 - val_mae: 2.4236
Epoch 7/10
819/819 [==============================] - 204s 248ms/step - loss: 15.6809 - mae:
3.0516 - val_loss: 9.5469 - val_mae: 2.4179
Epoch 8/10
819/819 [==============================] - 206s 251ms/step - loss: 15.3733 - mae:
3.0241 - val_loss: 9.6503 - val_mae: 2.4221
Epoch 9/10
819/819 [==============================] - 201s 245ms/step - loss: 15.1020 - mae:
2.9991 - val_loss: 9.4657 - val_mae: 2.4021
Epoch 10/10
819/819 [==============================] - 202s 246ms/step - loss: 14.8742 - mae:
2.9775 - val_loss: 9.2563 - val_mae: 2.3681
405/405 [==============================] - 30s 73ms/step - loss: 10.7263 - mae: 2.
5759
Test MAE: 2.58
```

In [27]:

```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="red",linestyle="dashed", label="Validation MAE")
plt.title("Training and validation MAE")
plt.xlabel("Epochs")
plt.ylabel("MAE")
plt.legend()
plt.show()
```



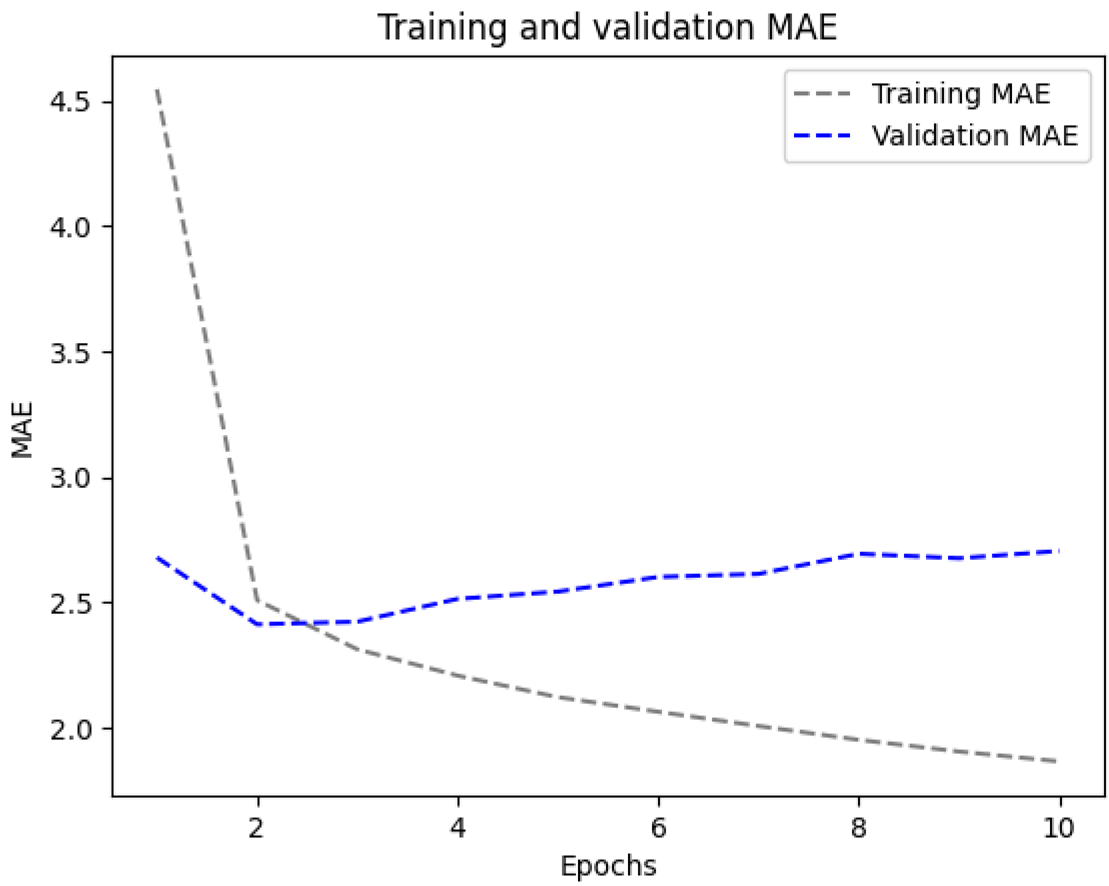
3.LSTM - Stacked setup with 16 units

In [28]:
```
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 [==============================] - 210s 252ms/step - loss: 39.1092 - mae: 
4.5460 - val_loss: 12.4770 - val_mae: 2.6780
Epoch 2/10
819/819 [==============================] - 200s 244ms/step - loss: 10.4569 - mae: 
2.5076 - val_loss: 9.6022 - val_mae: 2.4114
Epoch 3/10
819/819 [==============================] - 204s 249ms/step - loss: 8.8030 - mae: 
2.3115 - val_loss: 9.6140 - val_mae: 2.4216
Epoch 4/10
819/819 [==============================] - 204s 249ms/step - loss: 7.9985 - mae: 
2.2066 - val_loss: 10.3874 - val_mae: 2.5129
Epoch 5/10
819/819 [==============================] - 204s 249ms/step - loss: 7.4054 - mae: 
2.1203 - val_loss: 10.6091 - val_mae: 2.5421
Epoch 6/10
819/819 [==============================] - 204s 249ms/step - loss: 6.9957 - mae: 
2.0618 - val_loss: 11.0474 - val_mae: 2.6006
Epoch 7/10
819/819 [==============================] - 202s 247ms/step - loss: 6.6270 - mae: 
2.0051 - val_loss: 11.1384 - val_mae: 2.6126
Epoch 8/10
819/819 [==============================] - 206s 251ms/step - loss: 6.2702 - mae: 
1.9499 - val_loss: 12.0690 - val_mae: 2.6931
Epoch 9/10
819/819 [==============================] - 204s 248ms/step - loss: 5.9790 - mae: 
1.9032 - val_loss: 11.8353 - val_mae: 2.6752
Epoch 10/10
819/819 [==============================] - 201s 246ms/step - loss: 5.7425 - mae: 
1.8638 - val_loss: 12.1792 - val_mae: 2.7043
405/405 [==============================] - 42s 101ms/step - loss: 10.6859 - mae: 
2.5596
Test MAE: 2.56
```

In [29]:

```
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()
```



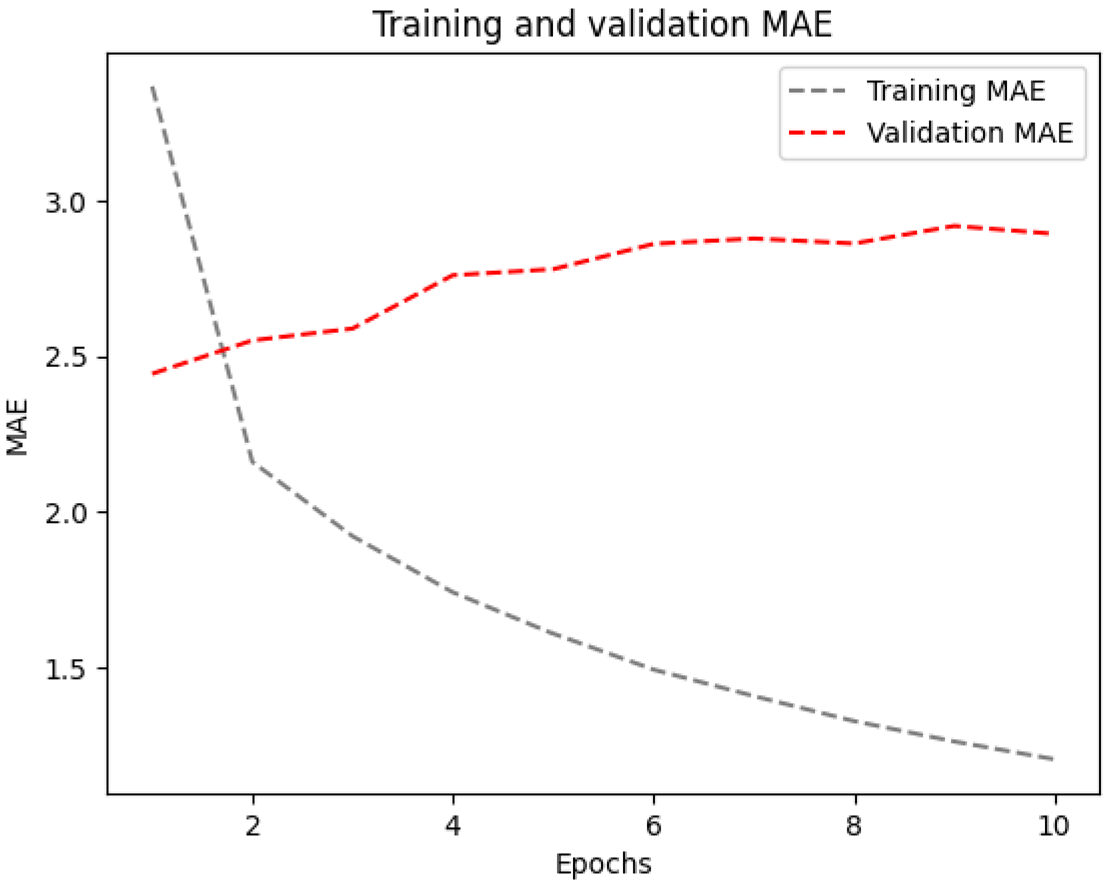
4.LSTM - Stacked setup with 32 units

In [30]:
```
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 [==============================] - 330s 396ms/step - loss: 22.2477 - mae: 3.3679 - val_loss: 9.7663 - val_mae: 2.4430
Epoch 2/10
819/819 [==============================] - 308s 376ms/step - loss: 7.7682 - mae: 2.1612 - val_loss: 10.6183 - val_mae: 2.5505
Epoch 3/10
819/819 [==============================] - 298s 363ms/step - loss: 6.2067 - mae: 1.9220 - val_loss: 10.8947 - val_mae: 2.5886
Epoch 4/10
819/819 [==============================] - 317s 386ms/step - loss: 5.1400 - mae: 1.7416 - val_loss: 12.3785 - val_mae: 2.7602
Epoch 5/10
819/819 [==============================] - 319s 389ms/step - loss: 4.3984 - mae: 1.6083 - val_loss: 12.6155 - val_mae: 2.7797
Epoch 6/10
819/819 [==============================] - 322s 393ms/step - loss: 3.8044 - mae: 1.4926 - val_loss: 13.2857 - val_mae: 2.8612
Epoch 7/10
819/819 [==============================] - 293s 357ms/step - loss: 3.3991 - mae: 1.4076 - val_loss: 13.3659 - val_mae: 2.8783
Epoch 8/10
819/819 [==============================] - 316s 385ms/step - loss: 3.0328 - mae: 1.3269 - val_loss: 13.2290 - val_mae: 2.8623
Epoch 9/10
819/819 [==============================] - 322s 393ms/step - loss: 2.7400 - mae: 1.2613 - val_loss: 13.8743 - val_mae: 2.9187
Epoch 10/10
819/819 [==============================] - 319s 390ms/step - loss: 2.5039 - mae: 1.2041 - val_loss: 13.6805 - val_mae: 2.8934
405/405 [==============================] - 57s 137ms/step - loss: 11.1510 - mae: 2.6264
Test MAE: 2.63
```

In [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="red",linestyle="dashed", label="Validation MAE")
plt.title("Training and validation MAE")
plt.xlabel("Epochs")
plt.ylabel("MAE")
plt.legend()
plt.show()
```



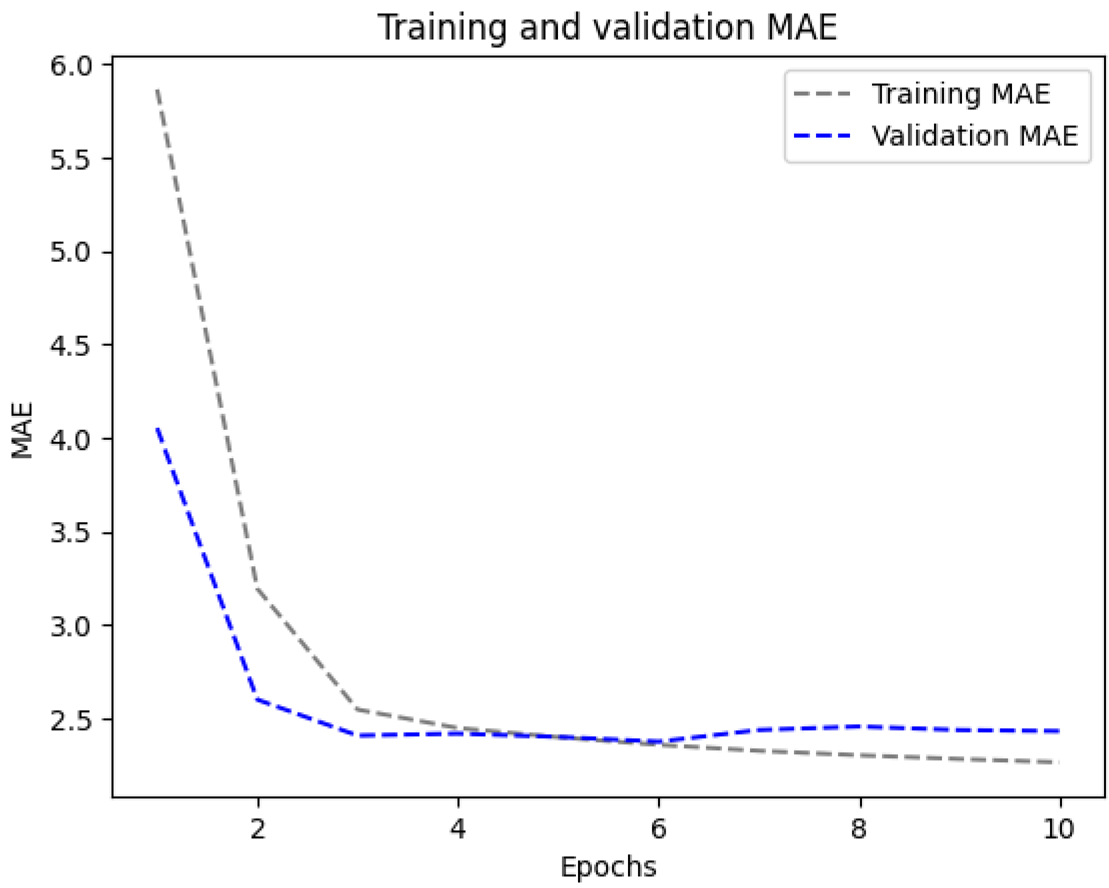
5.LSTM - Stacked setup with 8 units

In [32]:
```
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 [==============================] - 187s 223ms/step - loss: 59.9902 - mae:
5.8641 - val_loss: 29.8034 - val_mae: 4.0523
Epoch 2/10
819/819 [==============================] - 170s 208ms/step - loss: 18.3792 - mae:
3.1960 - val_loss: 11.7721 - val_mae: 2.6024
Epoch 3/10
819/819 [==============================] - 168s 205ms/step - loss: 10.7467 - mae:
2.5491 - val_loss: 9.6501 - val_mae: 2.4101
Epoch 4/10
819/819 [==============================] - 183s 223ms/step - loss: 9.8224 - mae:
2.4505 - val_loss: 9.7333 - val_mae: 2.4208
Epoch 5/10
819/819 [==============================] - 174s 211ms/step - loss: 9.4282 - mae:
2.4013 - val_loss: 9.6119 - val_mae: 2.4030
Epoch 6/10
819/819 [==============================] - 174s 211ms/step - loss: 9.1194 - mae:
2.3604 - val_loss: 9.4183 - val_mae: 2.3777
Epoch 7/10
819/819 [==============================] - 180s 220ms/step - loss: 8.8791 - mae:
2.3286 - val_loss: 9.9227 - val_mae: 2.4400
Epoch 8/10
819/819 [==============================] - 167s 203ms/step - loss: 8.6946 - mae:
2.3048 - val_loss: 10.0983 - val_mae: 2.4584
Epoch 9/10
819/819 [==============================] - 178s 218ms/step - loss: 8.5446 - mae:
2.2848 - val_loss: 9.9241 - val_mae: 2.4397
Epoch 10/10
819/819 [==============================] - 179s 218ms/step - loss: 8.4160 - mae:
2.2672 - val_loss: 9.9102 - val_mae: 2.4338
405/405 [==============================] - 36s 87ms/step - loss: 10.2210 - mae: 2.
4897
Test MAE: 2.49
```

In [33]:

```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()
```



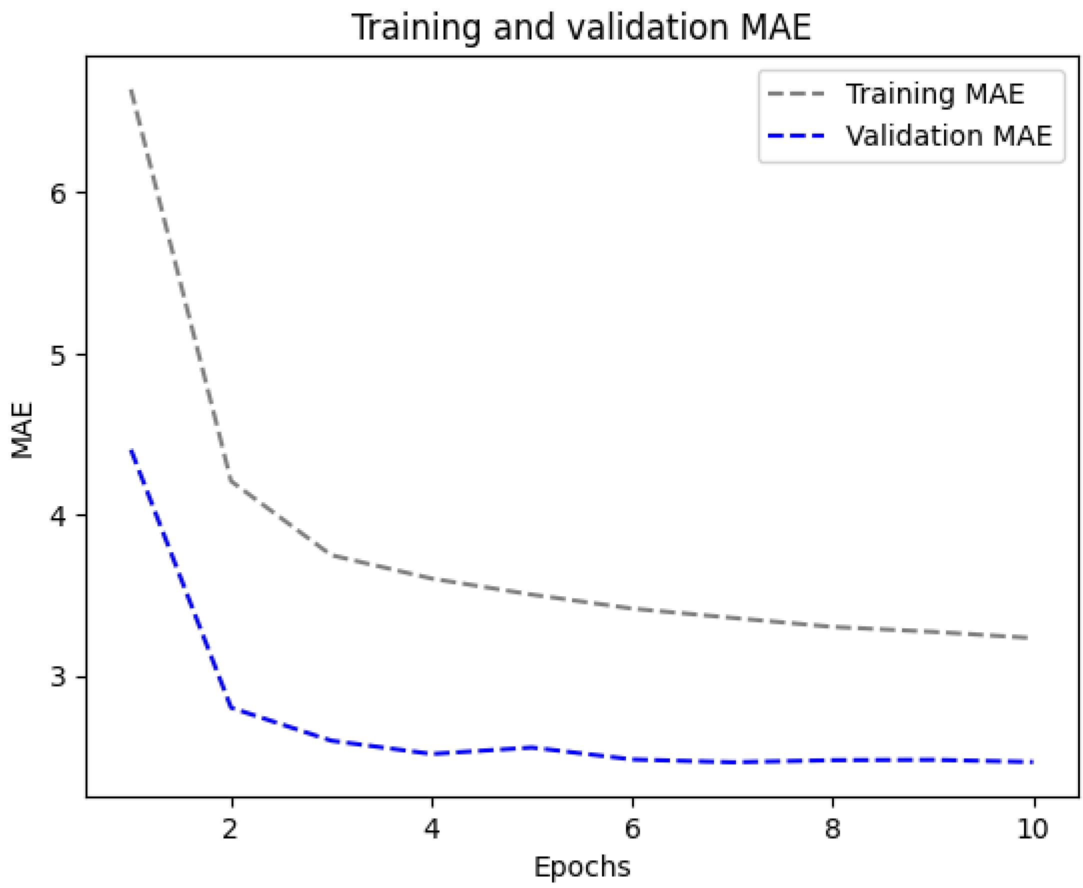
6.LSTM - dropout-regularized, stacked model

In [34]:
```
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 [==============================] - 312s 372ms/step - loss: 74.6289 - mae:
6.6415 - val_loss: 35.3969 - val_mae: 4.4048
Epoch 2/10
819/819 [==============================] - 304s 371ms/step - loss: 31.8055 - mae:
4.2093 - val_loss: 14.0733 - val_mae: 2.8008
Epoch 3/10
819/819 [==============================] - 304s 371ms/step - loss: 24.5831 - mae:
3.7492 - val_loss: 11.4326 - val_mae: 2.5958
Epoch 4/10
819/819 [==============================] - 302s 368ms/step - loss: 22.5916 - mae:
3.6038 - val_loss: 10.5954 - val_mae: 2.5135
Epoch 5/10
819/819 [==============================] - 302s 369ms/step - loss: 21.2571 - mae:
3.5041 - val_loss: 10.7594 - val_mae: 2.5533
Epoch 6/10
819/819 [==============================] - 305s 372ms/step - loss: 20.1485 - mae:
3.4169 - val_loss: 10.2157 - val_mae: 2.4791
Epoch 7/10
819/819 [==============================] - 303s 369ms/step - loss: 19.4377 - mae:
3.3593 - val_loss: 10.0390 - val_mae: 2.4617
Epoch 8/10
819/819 [==============================] - 302s 368ms/step - loss: 18.8518 - mae:
3.3033 - val_loss: 10.0837 - val_mae: 2.4747
Epoch 9/10
819/819 [==============================] - 303s 370ms/step - loss: 18.4077 - mae:
3.2728 - val_loss: 10.0891 - val_mae: 2.4770
Epoch 10/10
819/819 [==============================] - 306s 373ms/step - loss: 17.9948 - mae:
3.2337 - val_loss: 9.9424 - val_mae: 2.4635
405/405 [==============================] - 38s 91ms/step - loss: 10.9912 - mae: 2.
6173
Test MAE: 2.62
```

In [35]:

```
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()
```



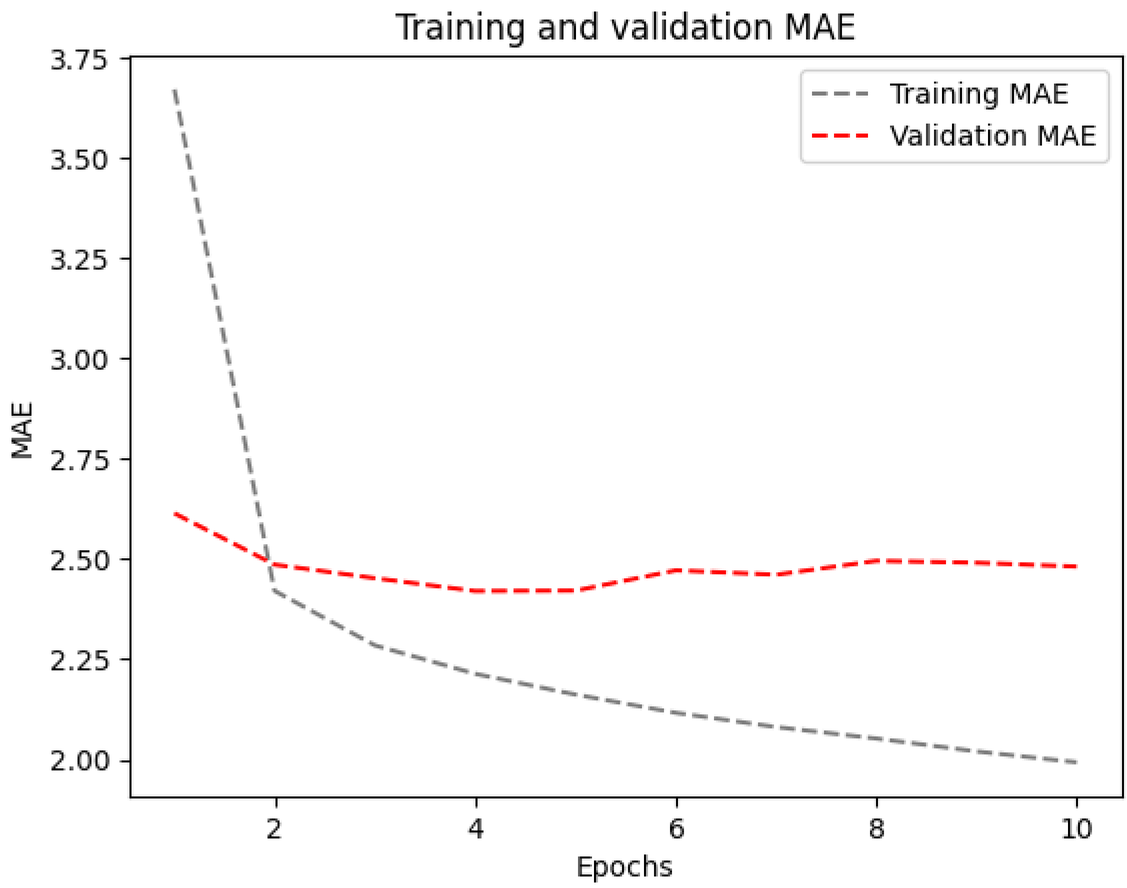
Bidirectional LSTM

In [36]:
```
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 [==============================] - 187s 223ms/step - loss: 26.2968 - mae:
3.6708 - val_loss: 11.4490 - val_mae: 2.6143
Epoch 2/10
819/819 [==============================] - 177s 216ms/step - loss: 9.6359 - mae:
2.4207 - val_loss: 10.1838 - val_mae: 2.4861
Epoch 3/10
819/819 [==============================] - 183s 223ms/step - loss: 8.6044 - mae:
2.2847 - val_loss: 10.0823 - val_mae: 2.4523
Epoch 4/10
819/819 [==============================] - 175s 213ms/step - loss: 8.0701 - mae:
2.2138 - val_loss: 9.7285 - val_mae: 2.4203
Epoch 5/10
819/819 [==============================] - 181s 220ms/step - loss: 7.6989 - mae:
2.1619 - val_loss: 9.7524 - val_mae: 2.4213
Epoch 6/10
819/819 [==============================] - 183s 223ms/step - loss: 7.3561 - mae:
2.1163 - val_loss: 10.0590 - val_mae: 2.4718
Epoch 7/10
819/819 [==============================] - 182s 222ms/step - loss: 7.1027 - mae:
2.0809 - val_loss: 10.0098 - val_mae: 2.4610
Epoch 8/10
819/819 [==============================] - 173s 211ms/step - loss: 6.9142 - mae:
2.0523 - val_loss: 10.4435 - val_mae: 2.4956
Epoch 9/10
819/819 [==============================] - 182s 222ms/step - loss: 6.6947 - mae:
2.0201 - val_loss: 10.3708 - val_mae: 2.4908
Epoch 10/10
819/819 [==============================] - 182s 222ms/step - loss: 6.5290 - mae:
1.9929 - val_loss: 10.3373 - val_mae: 2.4813
405/405 [==============================] - 38s 91ms/step - loss: 10.9553 - mae: 2.
6068
Test MAE: 2.61
```

In [37]:

```
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="red",linestyle="dashed", label="Validation MAE")
plt.title("Training and validation MAE")
plt.xlabel("Epochs")
plt.ylabel("MAE")
plt.legend()
plt.show()
```



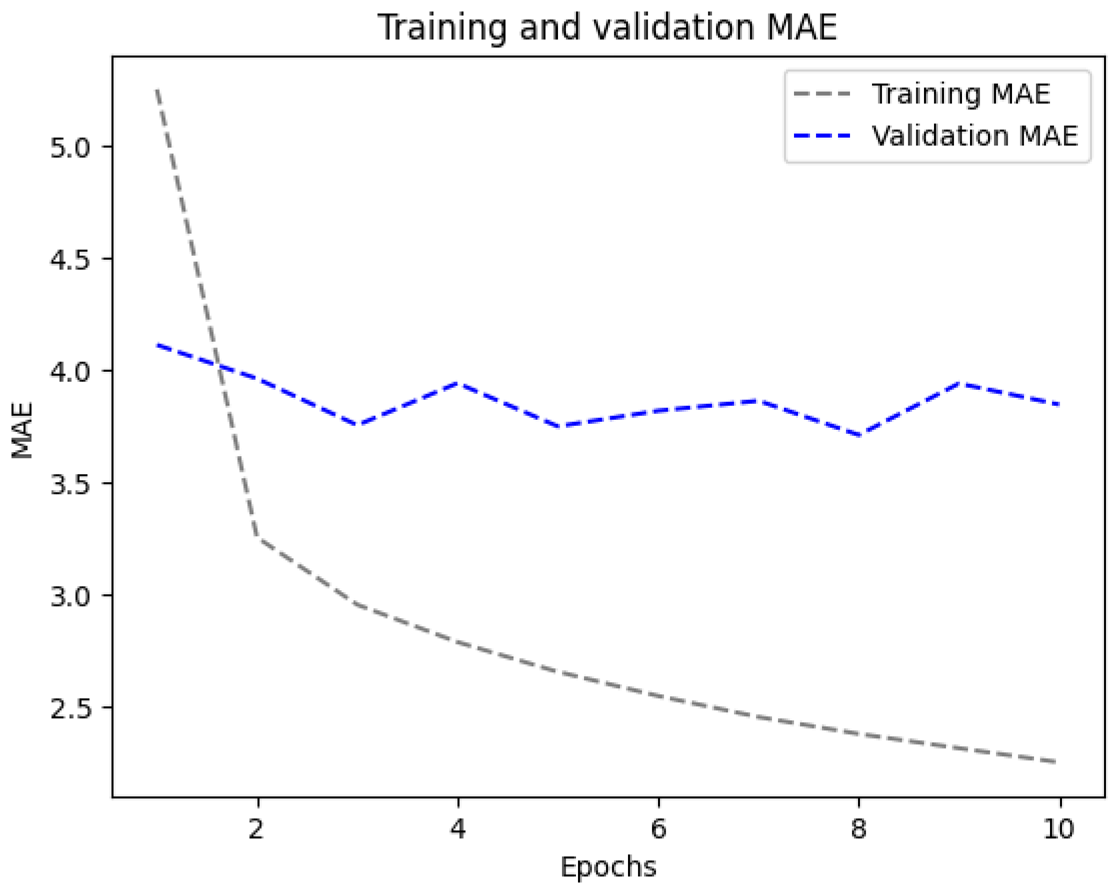
1D Convnets and LSTM togther

In [38]:
```
inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
y = layers.Conv1D(64, 3, activation='relu')(inputs)
y = layers.MaxPooling1D(3)(y)
y = layers.Conv1D(128, 3, activation='relu')(y)
y = layers.GlobalMaxPooling1D()(y)
y = layers.Reshape((-1, 128))(y)  # Reshape the data to be 3D
y = layers.LSTM(16)(y)
outputs = layers.Dense(1)(y)
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, callback

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

```
Epoch 1/10
819/819 [==============================] - 133s 159ms/step - loss: 49.2902 - mae:
5.2542 - val_loss: 28.4134 - val_mae: 4.1131
Epoch 2/10
819/819 [==============================] - 131s 160ms/step - loss: 17.8588 - mae:
3.2550 - val_loss: 25.8626 - val_mae: 3.9624
Epoch 3/10
819/819 [==============================] - 129s 157ms/step - loss: 14.5353 - mae:
2.9548 - val_loss: 22.7062 - val_mae: 3.7543
Epoch 4/10
819/819 [==============================] - 146s 178ms/step - loss: 12.9545 - mae:
2.7878 - val_loss: 25.2837 - val_mae: 3.9419
Epoch 5/10
819/819 [==============================] - 148s 181ms/step - loss: 11.8082 - mae:
2.6548 - val_loss: 21.4925 - val_mae: 3.7493
Epoch 6/10
819/819 [==============================] - 147s 179ms/step - loss: 10.9148 - mae:
2.5470 - val_loss: 22.8739 - val_mae: 3.8188
Epoch 7/10
819/819 [==============================] - 130s 158ms/step - loss: 10.1797 - mae:
2.4530 - val_loss: 23.1897 - val_mae: 3.8633
Epoch 8/10
819/819 [==============================] - 128s 156ms/step - loss: 9.6026 - mae:
2.3777 - val_loss: 21.9429 - val_mae: 3.7106
Epoch 9/10
819/819 [==============================] - 148s 180ms/step - loss: 9.1344 - mae:
2.3138 - val_loss: 24.4014 - val_mae: 3.9405
Epoch 10/10
819/819 [==============================] - 146s 177ms/step - loss: 8.6687 - mae:
2.2508 - val_loss: 23.8518 - val_mae: 3.8466
405/405 [==============================] - 27s 65ms/step - loss: 23.9505 - mae: 3.
9220
Test MAE: 3.92
```

In [39]:

```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()
```



We built 14 models: Following are the details;

Model 1: common-sense, non-machine-learning baseline

Model 2: A basic machine-learning model

Model 3: 1D convolutional model

Model 4: Simple RNN layer that can process sequences of any length

Model 5: Simple RNN - Stacking RNN layers

Model 6: A Simple GRU (Gated Recurrent Unit)

Model 7: LSTM-Simple

Model 8: LSTM - dropout Regularization

Model 9: Stacked setup with 16 units

Model 10: Stacked setup with 32 units

Model 11: Stacked setup with 8 units

Model 12: LSTM - dropout-regularized, stacked

Model 13: Bidirectional LSTM

Model 14: 1D Convnets and LSTM togther

In [40]:
```
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()
```