Deep learning for timeseries

In [1]:
```
# Download the jena_climate_2009_2016.csv.zip file from AWS S3
!wget https://s3.amazonaws.com/keras-datasets/jena_climate_2009_2016.csv.zip

# Unzip the jena_climate_2009_2016.csv.zip file
!unzip jena_climate_2009_2016.csv.zip
```

```
--2024-04-07 22:30:28--  https://s3.amazonaws.com/keras-datasets/jena_climate_2009
_2016.csv.zip
Resolving s3.amazonaws.com (s3.amazonaws.com)... 16.182.37.64, 52.217.87.110, 16.1
82.70.72, ...
Connecting to s3.amazonaws.com (s3.amazonaws.com)|16.182.37.64|: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  6.43MB/s    in 2.0s

2024-04-07 22:30:31 (6.43 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
```

Importing the data file

In [2]:
```
#Importing csv
import os
fname = os.path.join("jena_climate_2009_2016.csv")
```

In [3]:
```
#Reading Data from file
with open(fname) as f:
    data = f.read()
```

In [4]:
```
import pandas as pd
import os
for dirname, _, filenames in os.walk('/kaggle/'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

In [5]:
```
# Split the data into lines
lines = data.split("\n")
# Get the header row
header = lines[0].split(",")
# Remove the header row from the list of lines
lines = lines[1:]
# Print the header row
print(header)
# Print the number of lines in the data
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 the Data

In [6]:
```
import numpy as np

# Create a NumPy array to store the temperature data
temperature = np.zeros((len(lines),))

# Create a NumPy array to store the raw data
raw_data = np.zeros((len(lines), len(header) - 1))

# Iterate over the lines in the data
for i, line in enumerate(lines):

    # Split the line into a list of values
    values = [float(x) for x in line.split(",")[1:]]

    # Store the temperature in the temperature array
    temperature[i] = values[1]

    # Store the raw data in the raw_data array
    raw_data[i, :] = values[:]
```

In [7]:
```
temperature[:5]
```

Out[7]:
```
array([-8.02, -8.41, -8.51, -8.31, -8.27])
```

In [8]:
```
view = pd.DataFrame(temperature)
view.describe()
```

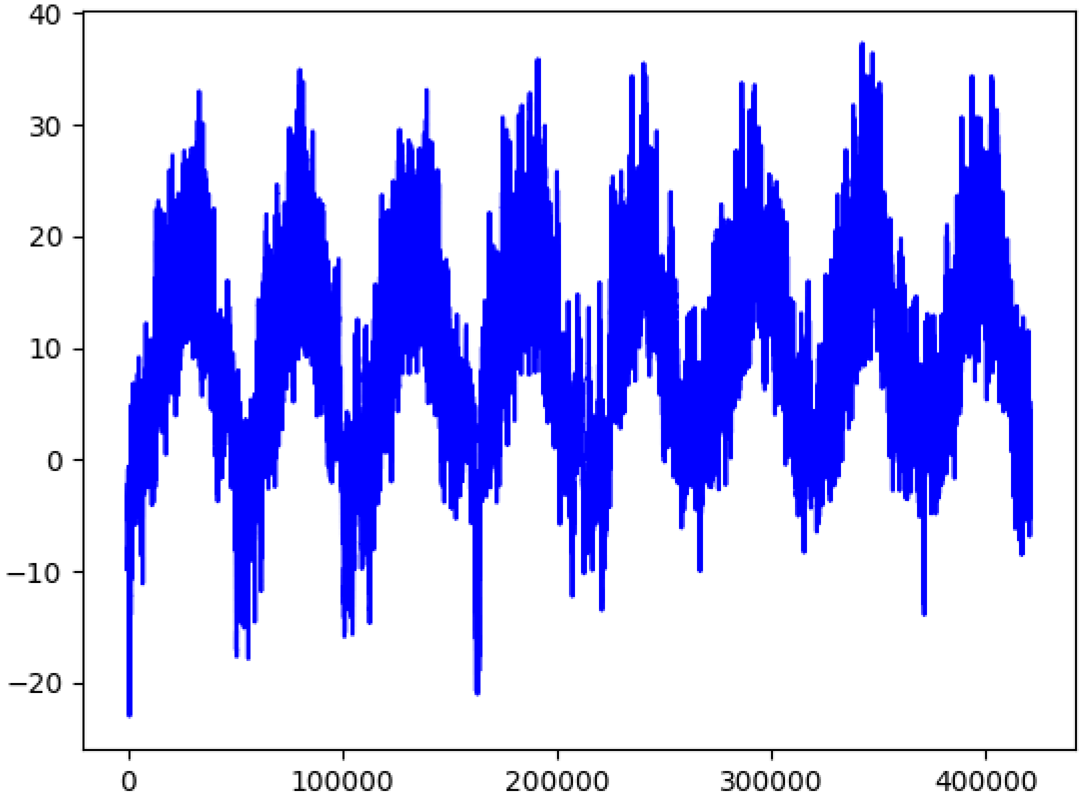
Out[8]:

	0
count	420451.000000
mean	9.448567
std	8.423685
min	-23.010000
25%	3.360000
50%	9.410000
75%	15.470000
max	37.280000

Plotting the temparature

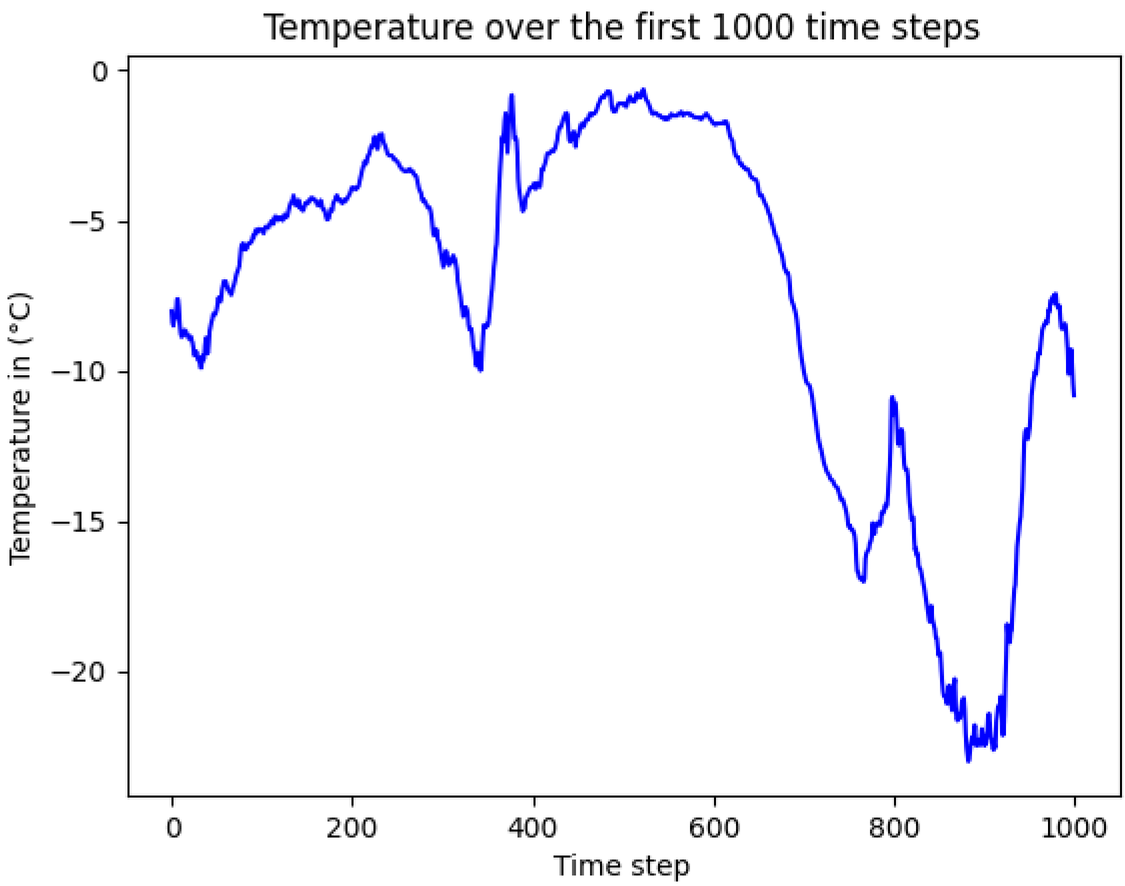
In [9]:
```
import matplotlib.pyplot as plt

plt.plot(range(len(temperature)), temperature, color='blue')
```

Out[9]:
```
[<matplotlib.lines.Line2D at 0x785516972aa0>]
```



Plotting the first 15 days of the temperature timeseries

In [10]:
```
## Plot the first 1000 temperature values
plt.plot(range(1000), temperature[:1000], color='blue')
# Set the x-axis label
plt.xlabel("Time step")

# Set the y-axis label
plt.ylabel("Temperature in (°C)")

# Set the title of the plot
plt.title("Temperature over the first 1000 time steps")

# Display the plot
plt.show()
```



Figuring out how many samples each data split will require

In [11]:
```
# Split the data into train, validation, and test sets
num_train_samples = int(0.5 * len(raw_data))  # 50% of the data for training
num_val_samples = int(0.25 * len(raw_data))  # 25% of the data for validation
num_test_samples = len(raw_data) - num_train_samples - num_val_samples  # The remai
```

Getting the data ready Data normalization

In [12]:
```
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,
)
for inputs, targets in dummy_dataset:
    for i in range(inputs.shape[0]):
        print([int(x) for x in inputs[i]], int(targets[i]))
```

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

In [13]:
```
#Instantiating datasets for training, validation, and testing
sampling_rate = 6
```

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

In [14]:

```
train_dataset = keras.utils.timeseries_dataset_from_array(
    raw_data[:-delay],  # The input data
    targets=temperature[delay:],  # The target data
    sampling_rate=sampling_rate,  # The sampling rate
    sequence_length=sequence_length,  # The length of the sequences
    shuffle=True,  # Whether to shuffle the data
    batch_size=batch_size,  # The batch size
    start_index=0,  # The start index
    end_index=num_train_samples)  # The end index
```

In [15]:

```
val_dataset = keras.utils.timeseries_dataset_from_array(
    raw_data[:-delay],  # The input data
    targets=temperature[delay:],  # The target data
    sampling_rate=sampling_rate,  # The sampling rate
    sequence_length=sequence_length,  # The length of the sequences
    shuffle=True,  # Whether to shuffle the data
    batch_size=batch_size,  # The batch size
    start_index=num_train_samples,  # The start index
    end_index=num_train_samples + num_val_samples)  # The end index
```

In [16]:

```
test_dataset = keras.utils.timeseries_dataset_from_array(
    raw_data[:-delay],  # The input data
    targets=temperature[delay:],  # The target data
    sampling_rate=sampling_rate,  # The sampling rate
    sequence_length=sequence_length,  # The length of the sequences
    shuffle=True,  # Whether to shuffle the data
    batch_size=batch_size,  # The batch size
    start_index=num_train_samples + num_val_samples,  # The start index
    # Set the end index to None so that the dataset will contain all of the remaini
    end_index=None)
```

In [17]:

```
# Iterate over the first sample in the train dataset
for samples, targets in train_dataset:
    # Print the shape of the samples and targets
    print("samples shape:", samples.shape)
    print("targets shape:", targets.shape)

    # Break out of the loop after the first iteration
    break
```

```
samples shape: (256, 120, 14)
targets shape: (256,)
```

common-sense, non-machine-learning baseline

In [18]:

```
#Computing the common-sense baseline MAE
def evaluate_naive_method(dataset):
    total_abs_err = 0.
    samples_seen = 0
    for samples, targets in dataset:
        preds = samples[:, -1, 1] * std[1] + mean[1]
        total_abs_err += np.sum(np.abs(preds - targets))
        samples_seen += samples.shape[0]
    return total_abs_err / samples_seen
print(f"Validation MAE: {evaluate_naive_method(val_dataset):.2f}")
print(f"Test MAE: {evaluate_naive_method(test_dataset):.2f}")
```

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

In [19]:
```python
from tensorflow import keras
from keras import layers
```

Constructing and assessing a densely linked model

In [20]:
```python
#Training and evaluating a densely connected model
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)

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

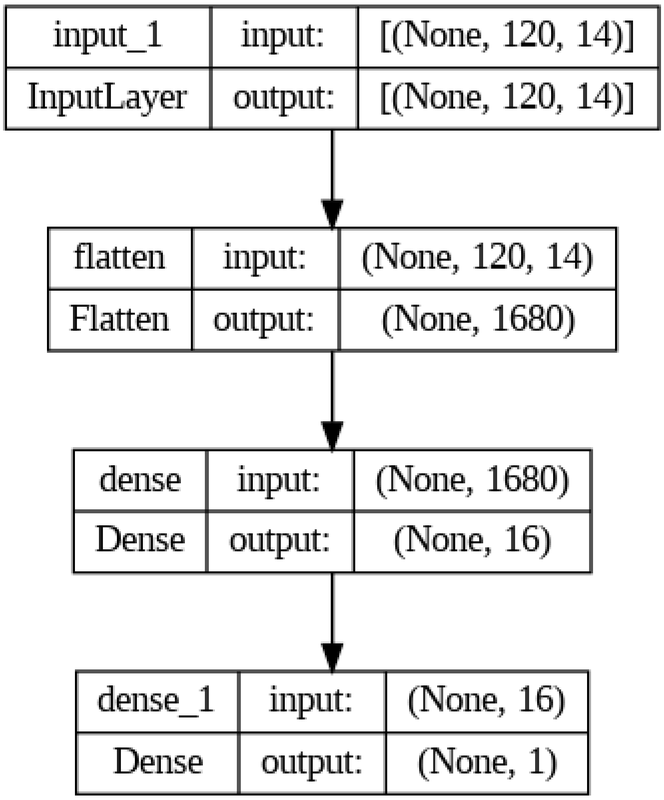
```
Epoch 1/10
819/819 [==============================] - 13s 15ms/step - loss: 13.2791 - mae: 2.
8277 - val_loss: 12.3551 - val_mae: 2.7730
Epoch 2/10
819/819 [==============================] - 12s 14ms/step - loss: 9.5497 - mae: 2.4
330 - val_loss: 11.0970 - val_mae: 2.6248
Epoch 3/10
819/819 [==============================] - 12s 14ms/step - loss: 8.7439 - mae: 2.3
274 - val_loss: 11.3616 - val_mae: 2.6611
Epoch 4/10
819/819 [==============================] - 12s 14ms/step - loss: 8.2253 - mae: 2.2
587 - val_loss: 13.4715 - val_mae: 2.9222
Epoch 5/10
819/819 [==============================] - 12s 14ms/step - loss: 7.9030 - mae: 2.2
168 - val_loss: 10.5612 - val_mae: 2.5718
Epoch 6/10
819/819 [==============================] - 11s 14ms/step - loss: 7.6325 - mae: 2.1
803 - val_loss: 10.9429 - val_mae: 2.6184
Epoch 7/10
819/819 [==============================] - 11s 13ms/step - loss: 7.4166 - mae: 2.1
513 - val_loss: 11.4363 - val_mae: 2.6894
Epoch 8/10
819/819 [==============================] - 11s 13ms/step - loss: 7.2431 - mae: 2.1
260 - val_loss: 11.1422 - val_mae: 2.6537
Epoch 9/10
819/819 [==============================] - 11s 13ms/step - loss: 7.0750 - mae: 2.1
006 - val_loss: 11.2063 - val_mae: 2.6545
Epoch 10/10
819/819 [==============================] - 11s 13ms/step - loss: 6.9414 - mae: 2.0
811 - val_loss: 11.1233 - val_mae: 2.6475
405/405 [==============================] - 4s 9ms/step - loss: 11.2387 - mae: 2.64
92
Test MAE: 2.65
```

In [21]:
```python
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_1 (InputLayer)        [(None, 120, 14)]         0         
                                                                 
 flatten (Flatten)           (None, 1680)              0         
                                                                 
 dense (Dense)               (None, 16)                26896     
                                                                 
 dense_1 (Dense)             (None, 1)                 17        
                                                                 
=================================================================
Total params: 26913 (105.13 KB)
Trainable params: 26913 (105.13 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

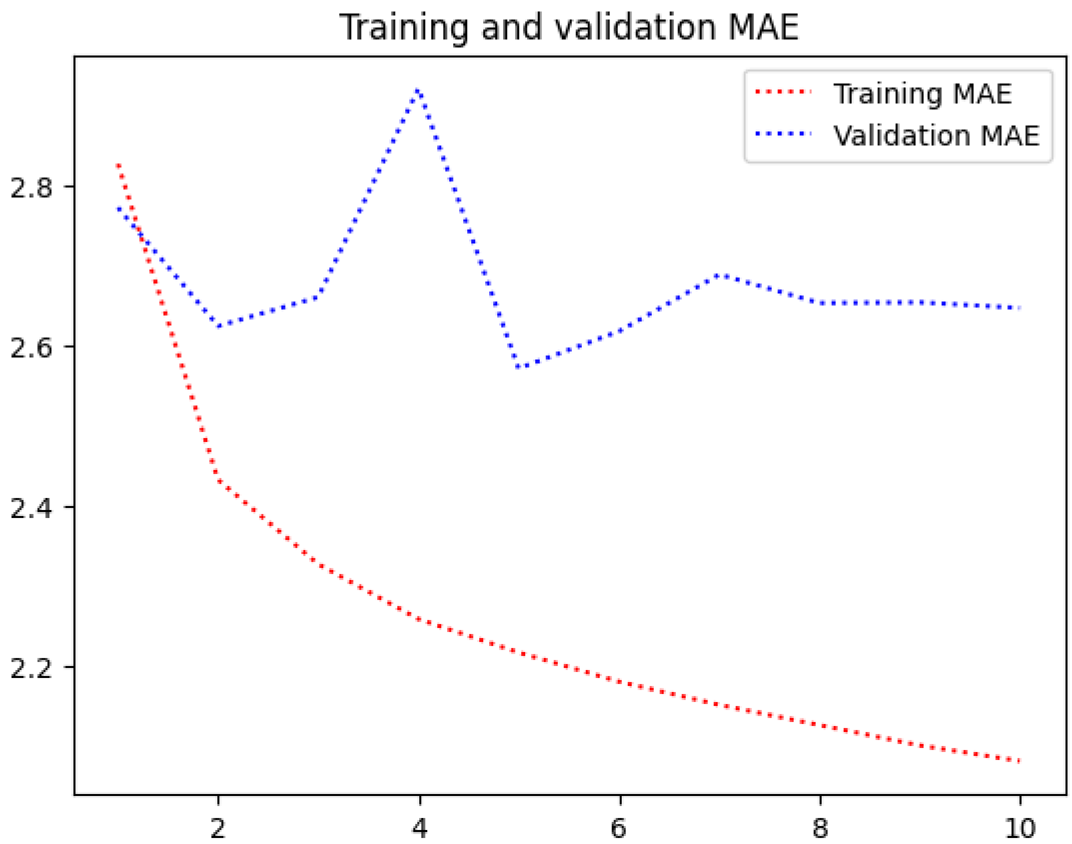
In [22]:
```python
keras.utils.plot_model(model, show_shapes=True)
```

Out[22]:



In [23]:

```python
import matplotlib.pyplot as plt

loss = history.history["mae"]
val_loss = history.history["val_mae"]
epochs = range(1, len(loss) + 1)

plt.figure()

# Update the color to red and line style to a dotted line for training MAE
plt.plot(epochs, loss, "r:", label="Training MAE")

# Update the color to blue and line style to a dotted line for validation MAE
plt.plot(epochs, val_loss, "b:", label="Validation MAE")

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



1D Convolution model with 10 epoch

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

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

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

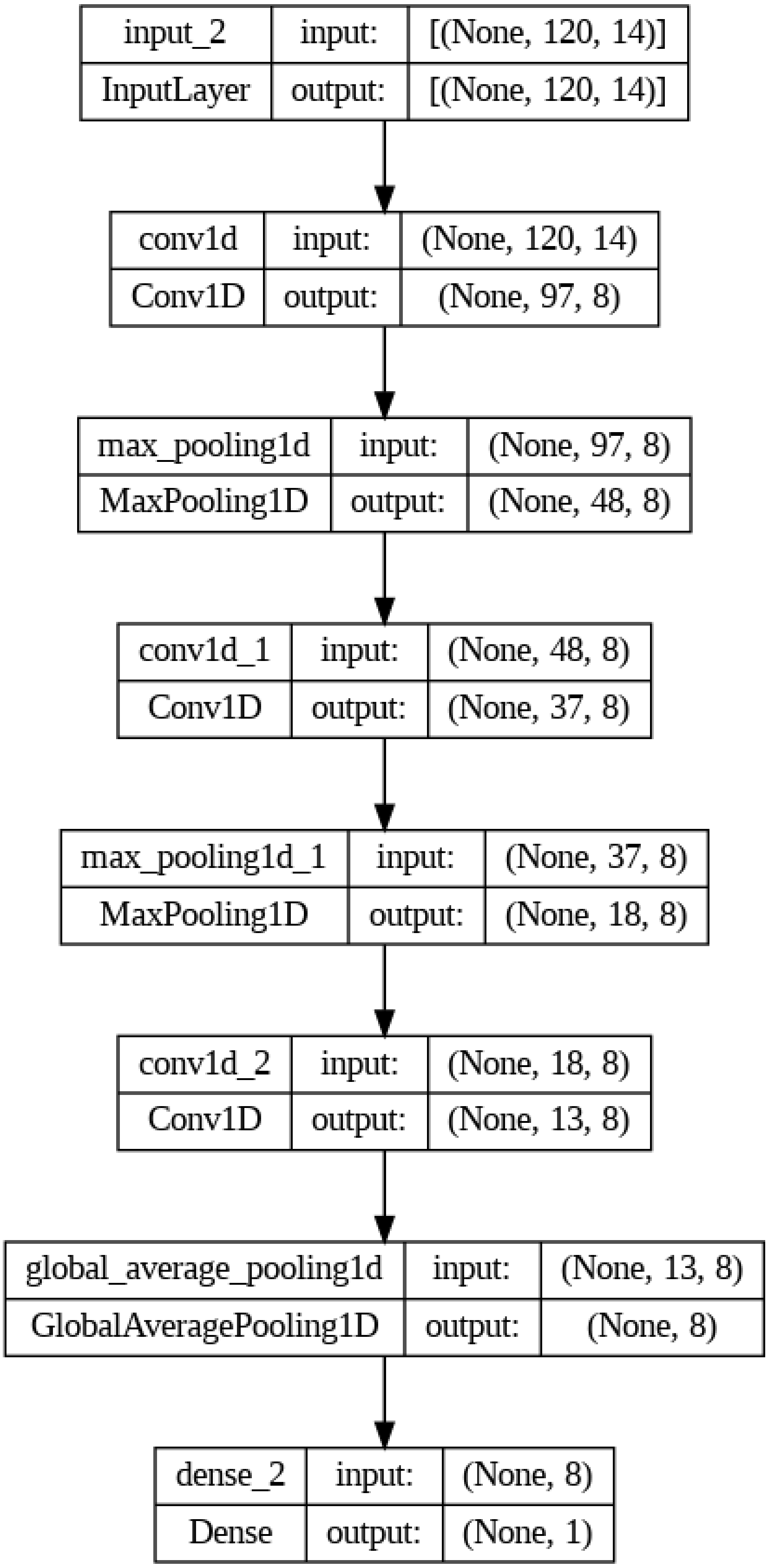
```
Epoch 1/10
819/819 [==============================] - 15s 15ms/step - loss: 21.7765 - mae: 3.
6716 - val_loss: 15.1529 - val_mae: 3.0795
Epoch 2/10
819/819 [==============================] - 12s 14ms/step - loss: 15.7164 - mae: 3.
1405 - val_loss: 14.4813 - val_mae: 3.0253
Epoch 3/10
819/819 [==============================] - 12s 14ms/step - loss: 14.4108 - mae: 3.
0067 - val_loss: 14.3182 - val_mae: 2.9788
Epoch 4/10
819/819 [==============================] - 11s 14ms/step - loss: 13.4329 - mae: 2.
9014 - val_loss: 14.8376 - val_mae: 3.0292
Epoch 5/10
819/819 [==============================] - 11s 14ms/step - loss: 12.7203 - mae: 2.
8232 - val_loss: 15.4623 - val_mae: 3.0842
Epoch 6/10
819/819 [==============================] - 11s 14ms/step - loss: 12.1124 - mae: 2.
7552 - val_loss: 14.8451 - val_mae: 3.0582
Epoch 7/10
819/819 [==============================] - 11s 14ms/step - loss: 11.6685 - mae: 2.
6992 - val_loss: 15.5517 - val_mae: 3.0988
Epoch 8/10
819/819 [==============================] - 11s 14ms/step - loss: 11.2676 - mae: 2.
6538 - val_loss: 14.6210 - val_mae: 3.0035
Epoch 9/10
819/819 [==============================] - 11s 14ms/step - loss: 10.9469 - mae: 2.
6150 - val_loss: 17.0661 - val_mae: 3.2570
Epoch 10/10
819/819 [==============================] - 11s 14ms/step - loss: 10.6152 - mae: 2.
5757 - val_loss: 14.9081 - val_mae: 3.0316
405/405 [==============================] - 4s 9ms/step - loss: 16.0753 - mae: 3.17
01
Test MAE: 3.17
```

In [25]:
```
model.summary()
```

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

 conv1d (Conv1D)             (None, 97, 8)             2696

 max_pooling1d (MaxPooling1  (None, 48, 8)             0
 D)

 conv1d_1 (Conv1D)           (None, 37, 8)             776

 max_pooling1d_1 (MaxPoolin  (None, 18, 8)             0
 g1D)

 conv1d_2 (Conv1D)           (None, 13, 8)             392

 global_average_pooling1d (  (None, 8)                 0
 GlobalAveragePooling1D)

 dense_2 (Dense)             (None, 1)                 9

=================================================================
Total params: 3873 (15.13 KB)
Trainable params: 3873 (15.13 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

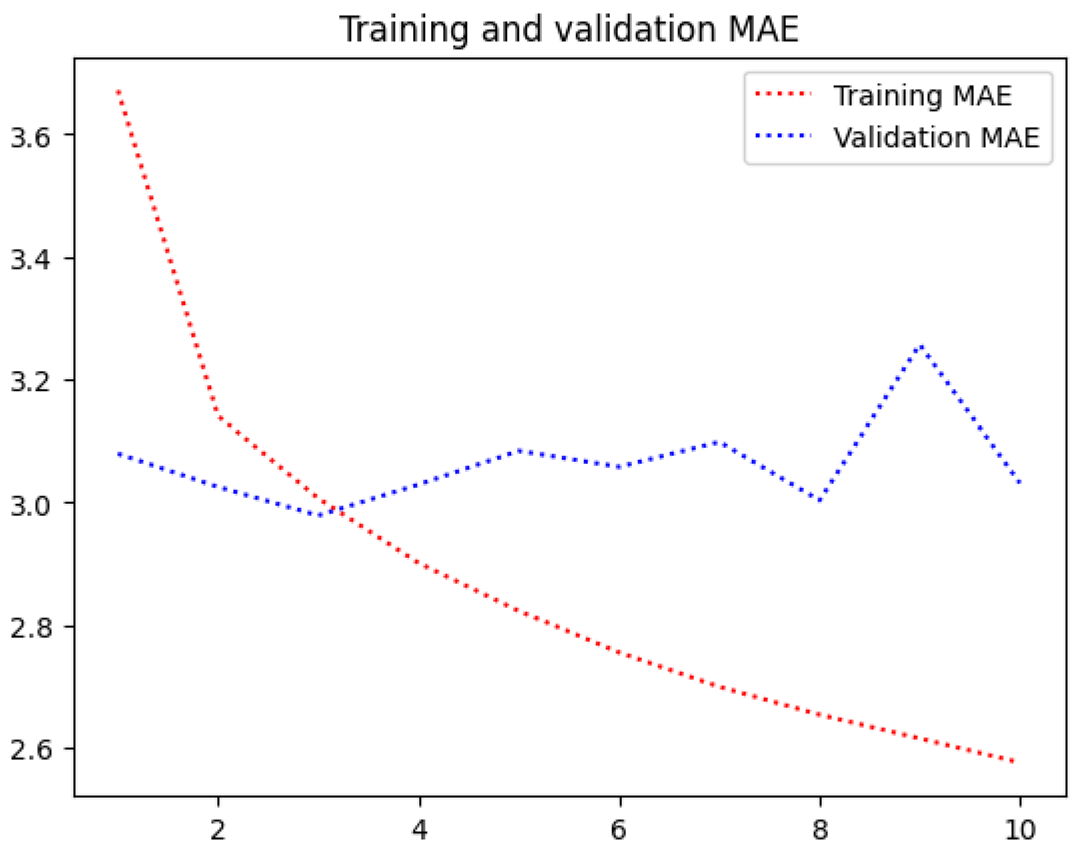
In [26]:
```
keras.utils.plot_model(model, show_shapes=True)
```

Out[26]:



In [27]:
```
import matplotlib.pyplot as plt

loss = history.history["mae"]
val_loss = history.history["val_mae"]
epochs = range(1, len(loss) + 1)

plt.figure()
```

```
# Update the color to red and line style to a dotted line for training MAE
plt.plot(epochs, loss, "r:", label="Training MAE")

# Update the color to blue and line style to a dotted line for validation MAE
plt.plot(epochs, val_loss, "b:", label="Validation MAE")

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



As time goes on, both the training and validation losses decrease, indicating that the model is learning. It is common that the validation loss is marginally greater than the training loss. To ensure that the model is not overfitting the training set, it is crucial to keep an eye on the validation loss. The model may not be properly generalizing to new data, as indicated by the test MAE being considerably higher than the validation MAE. There are several possible reasons for this, including noise in the data or an undertrained model.

A Simple RNN

In [28]:
```
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.st",
                                    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.st")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

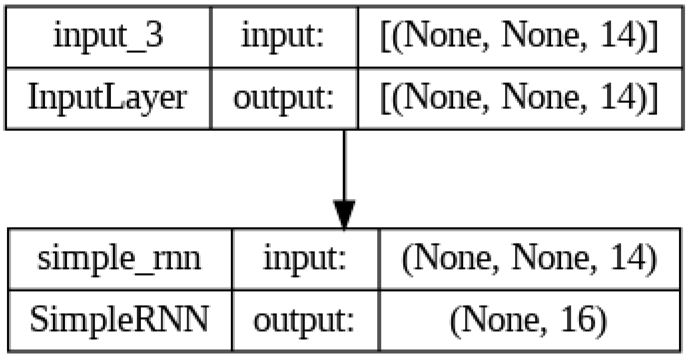
```
Epoch 1/10
819/819 [==============================] - 47s 56ms/step - loss: 139.1357 - mae:
9.7067 - val_loss: 143.8847 - val_mae: 9.8867
Epoch 2/10
819/819 [==============================] - 46s 56ms/step - loss: 136.3859 - mae:
9.5603 - val_loss: 143.7051 - val_mae: 9.8688
Epoch 3/10
819/819 [==============================] - 45s 55ms/step - loss: 136.2942 - mae:
9.5505 - val_loss: 143.6327 - val_mae: 9.8576
Epoch 4/10
819/819 [==============================] - 46s 55ms/step - loss: 136.2349 - mae:
9.5446 - val_loss: 143.5929 - val_mae: 9.8534
Epoch 5/10
819/819 [==============================] - 46s 56ms/step - loss: 136.1976 - mae:
9.5403 - val_loss: 143.5897 - val_mae: 9.8548
Epoch 6/10
819/819 [==============================] - 45s 55ms/step - loss: 136.1642 - mae:
9.5376 - val_loss: 143.5294 - val_mae: 9.8493
Epoch 7/10
819/819 [==============================] - 45s 55ms/step - loss: 136.1381 - mae:
9.5345 - val_loss: 143.5399 - val_mae: 9.8514
Epoch 8/10
819/819 [==============================] - 44s 54ms/step - loss: 136.1289 - mae:
9.5337 - val_loss: 143.5414 - val_mae: 9.8525
Epoch 9/10
819/819 [==============================] - 44s 54ms/step - loss: 136.1107 - mae:
9.5317 - val_loss: 143.5461 - val_mae: 9.8528
Epoch 10/10
819/819 [==============================] - 45s 54ms/step - loss: 136.1015 - mae:
9.5306 - val_loss: 143.5316 - val_mae: 9.8517
405/405 [==============================] - 6s 14ms/step - loss: 151.2813 - mae: 9.
9175
Test MAE: 9.92
```

In [29]:
```
model.summary()
```

```
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, None, 14)]        0

 simple_rnn (SimpleRNN)      (None, 16)                496

=================================================================
Total params: 496 (1.94 KB)
Trainable params: 496 (1.94 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

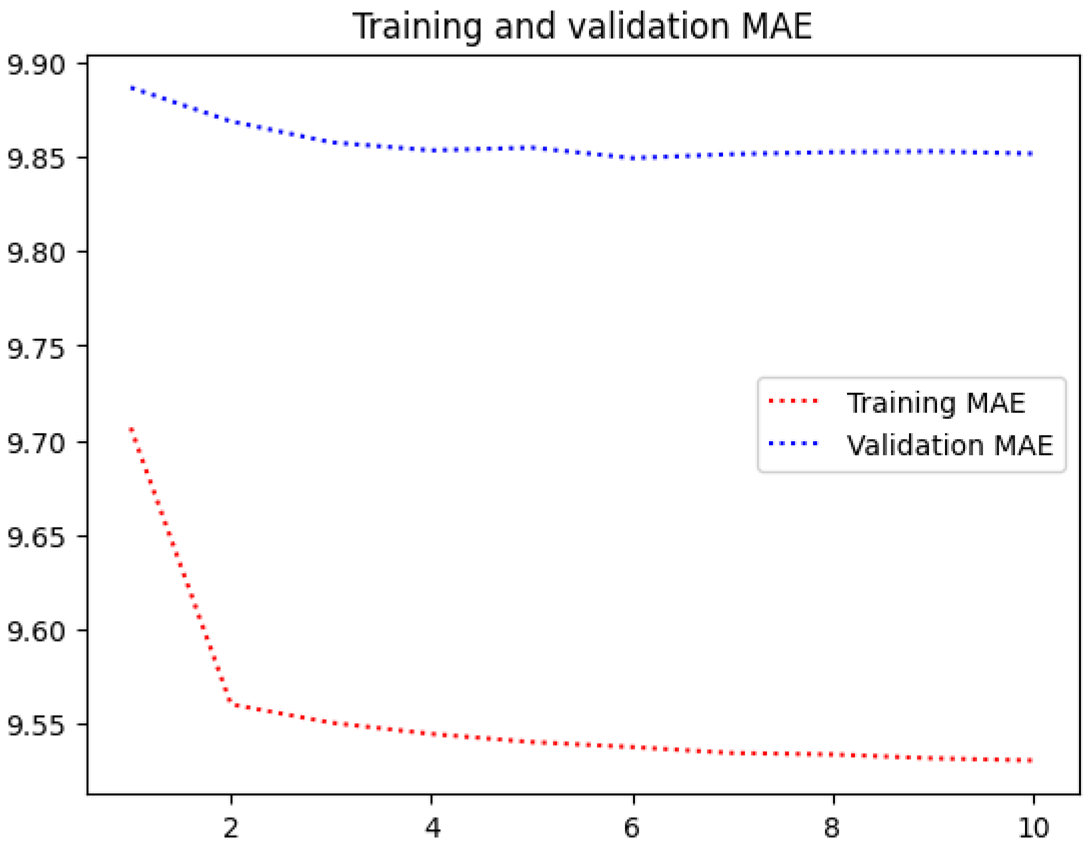
In [30]:
```
keras.utils.plot_model(model, show_shapes=True)
```

Out[30]:



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()

# Update the color to red and line style to a dotted line for training MAE
plt.plot(epochs, loss, "r:", label="Training MAE")

# Update the color to blue and line style to a dotted line for validation MAE
plt.plot(epochs, val_loss, "b:", label="Validation MAE")

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



The solution to the above is to experiment with increasing or decreasing the model complexity. After two epochs, the model has finished learning because the MAE is a constant line and has converged to a specific level of performance. Further training does not result in a meaningful improvement. We can explore other structures, add

more levels, or increase the number of units in the layers that already exist.attempting to stack the RNNs in an attempt to add extra layers to the data collection

Simple RNN - Stacking RNN layers

In [32]:
```
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.st",
                                    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.st")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

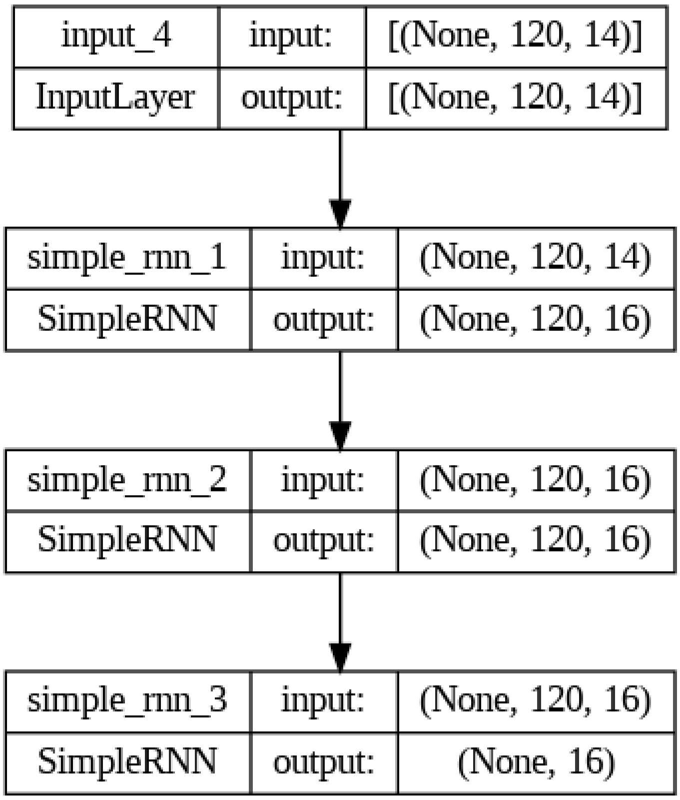
```
Epoch 1/10
819/819 [==============================] - 188s 227ms/step - loss: 136.9309 - mae:
9.5741 - val_loss: 143.4676 - val_mae: 9.8414
Epoch 2/10
819/819 [==============================] - 187s 228ms/step - loss: 136.0621 - mae:
9.5274 - val_loss: 143.4446 - val_mae: 9.8393
Epoch 3/10
819/819 [==============================] - 184s 225ms/step - loss: 136.0032 - mae:
9.5181 - val_loss: 143.3821 - val_mae: 9.8310
Epoch 4/10
819/819 [==============================] - 184s 225ms/step - loss: 135.9274 - mae:
9.5083 - val_loss: 143.3766 - val_mae: 9.8341
Epoch 5/10
819/819 [==============================] - 180s 220ms/step - loss: 135.8805 - mae:
9.5015 - val_loss: 143.3880 - val_mae: 9.8321
Epoch 6/10
819/819 [==============================] - 180s 220ms/step - loss: 135.8664 - mae:
9.4994 - val_loss: 143.3816 - val_mae: 9.8294
Epoch 7/10
819/819 [==============================] - 180s 219ms/step - loss: 135.8464 - mae:
9.4968 - val_loss: 143.3848 - val_mae: 9.8325
Epoch 8/10
819/819 [==============================] - 179s 218ms/step - loss: 135.8250 - mae:
9.4927 - val_loss: 143.4053 - val_mae: 9.8384
Epoch 9/10
819/819 [==============================] - 178s 217ms/step - loss: 135.8163 - mae:
9.4912 - val_loss: 143.3974 - val_mae: 9.8346
Epoch 10/10
819/819 [==============================] - 182s 222ms/step - loss: 135.8105 - mae:
9.4906 - val_loss: 143.3593 - val_mae: 9.8296
405/405 [==============================] - 13s 30ms/step - loss: 151.1117 - mae:
9.9015
Test MAE: 9.90
```

In [33]:
```
model.summary()
```

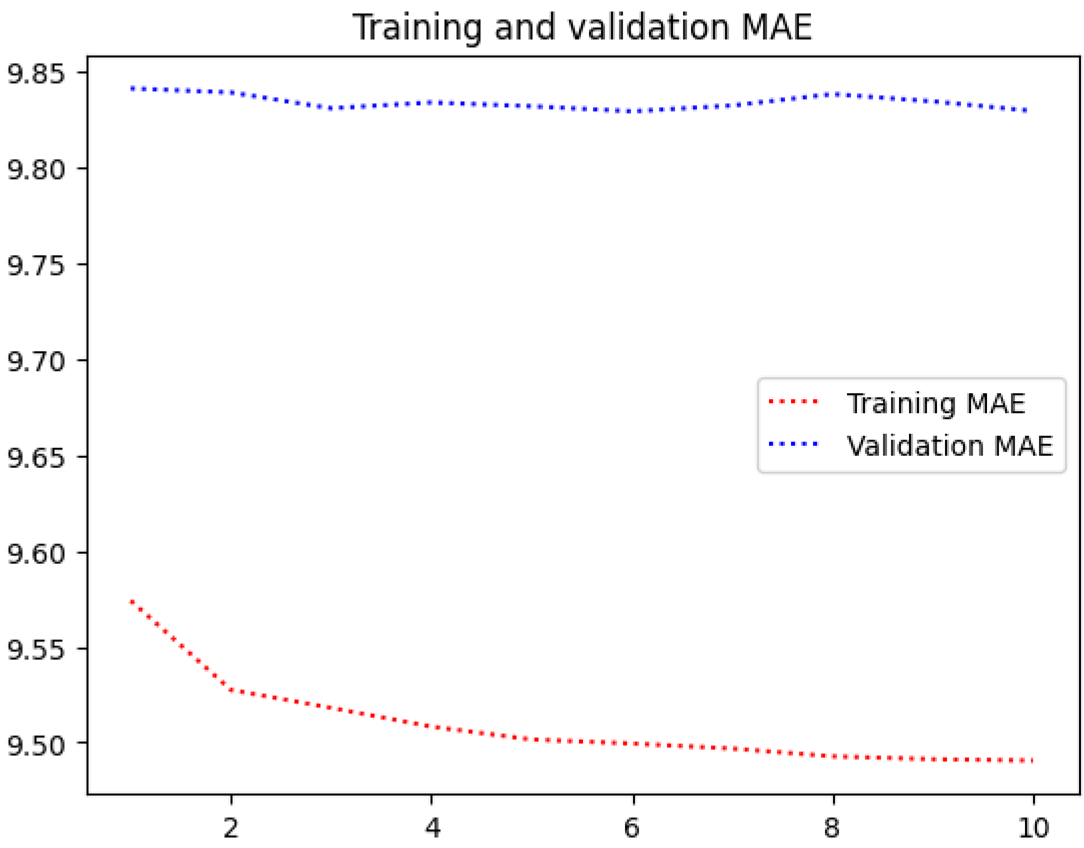
```
Model: "model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_4 (InputLayer)        [(None, 120, 14)]         0

 simple_rnn_1 (SimpleRNN)    (None, 120, 16)           496

 simple_rnn_2 (SimpleRNN)    (None, 120, 16)           528

 simple_rnn_3 (SimpleRNN)    (None, 16)                528

=================================================================
Total params: 1552 (6.06 KB)
Trainable params: 1552 (6.06 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [34]:
```python
keras.utils.plot_model(model, show_shapes=True)
```

Out[34]:



In [35]:
```python
import matplotlib.pyplot as plt

loss = history.history["mae"]
val_loss = history.history["val_mae"]
epochs = range(1, len(loss) + 1)

plt.figure()

# Update the color to red and line style to a dotted line for training MAE
plt.plot(epochs, loss, "r:", label="Training MAE")

# Update the color to blue and line style to a dotted line for validation MAE
plt.plot(epochs, val_loss, "b:", label="Validation MAE")

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



The model's inability to identify the underlying patterns in the data is indicated by the Training and Validation MAE, which seems to be a constant line.

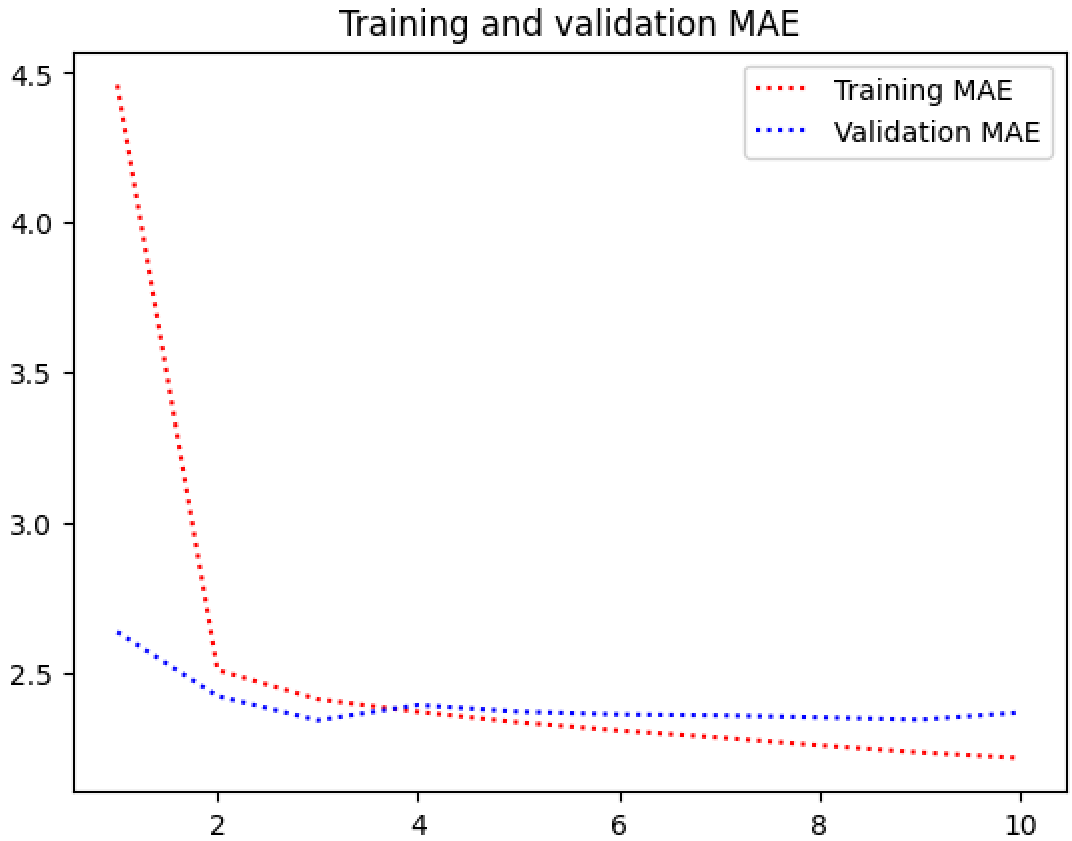
Simple GRU (Gated Recurrent Unit)

In [36]:
```
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.st",
                                    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.st")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
Epoch 1/10
819/819 [==============================] - 16s 17ms/step - loss: 37.9946 - mae: 4.
4591 - val_loss: 12.1957 - val_mae: 2.6376
Epoch 2/10
819/819 [==============================] - 14s 17ms/step - loss: 10.3635 - mae: 2.
5105 - val_loss: 9.9849 - val_mae: 2.4236
Epoch 3/10
819/819 [==============================] - 14s 17ms/step - loss: 9.5215 - mae: 2.4
133 - val_loss: 9.1291 - val_mae: 2.3428
Epoch 4/10
819/819 [==============================] - 12s 15ms/step - loss: 9.1857 - mae: 2.3
710 - val_loss: 9.6521 - val_mae: 2.3937
Epoch 5/10
819/819 [==============================] - 12s 14ms/step - loss: 8.8995 - mae: 2.3
360 - val_loss: 9.4650 - val_mae: 2.3717
Epoch 6/10
819/819 [==============================] - 12s 14ms/step - loss: 8.6781 - mae: 2.3
084 - val_loss: 9.3988 - val_mae: 2.3622
Epoch 7/10
819/819 [==============================] - 12s 15ms/step - loss: 8.4873 - mae: 2.2
853 - val_loss: 9.3577 - val_mae: 2.3593
Epoch 8/10
819/819 [==============================] - 12s 15ms/step - loss: 8.2642 - mae: 2.2
592 - val_loss: 9.3445 - val_mae: 2.3525
Epoch 9/10
819/819 [==============================] - 12s 14ms/step - loss: 8.0774 - mae: 2.2
355 - val_loss: 9.3027 - val_mae: 2.3450
Epoch 10/10
819/819 [==============================] - 12s 14ms/step - loss: 7.9325 - mae: 2.2
172 - val_loss: 9.5235 - val_mae: 2.3687
405/405 [==============================] - 4s 9ms/step - loss: 10.0783 - mae: 2.48
80
Test MAE: 2.49
```

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()

# Update the color to red and line style to a dotted line for training MAE
plt.plot(epochs, loss, "r:", label="Training MAE")

# Update the color to blue and line style to a dotted line for validation MAE
plt.plot(epochs, val_loss, "b:", label="Validation MAE")

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



An abbreviated form of the Long Short-Term Memory (LSTM) network architecture is called LSTM-Simple.

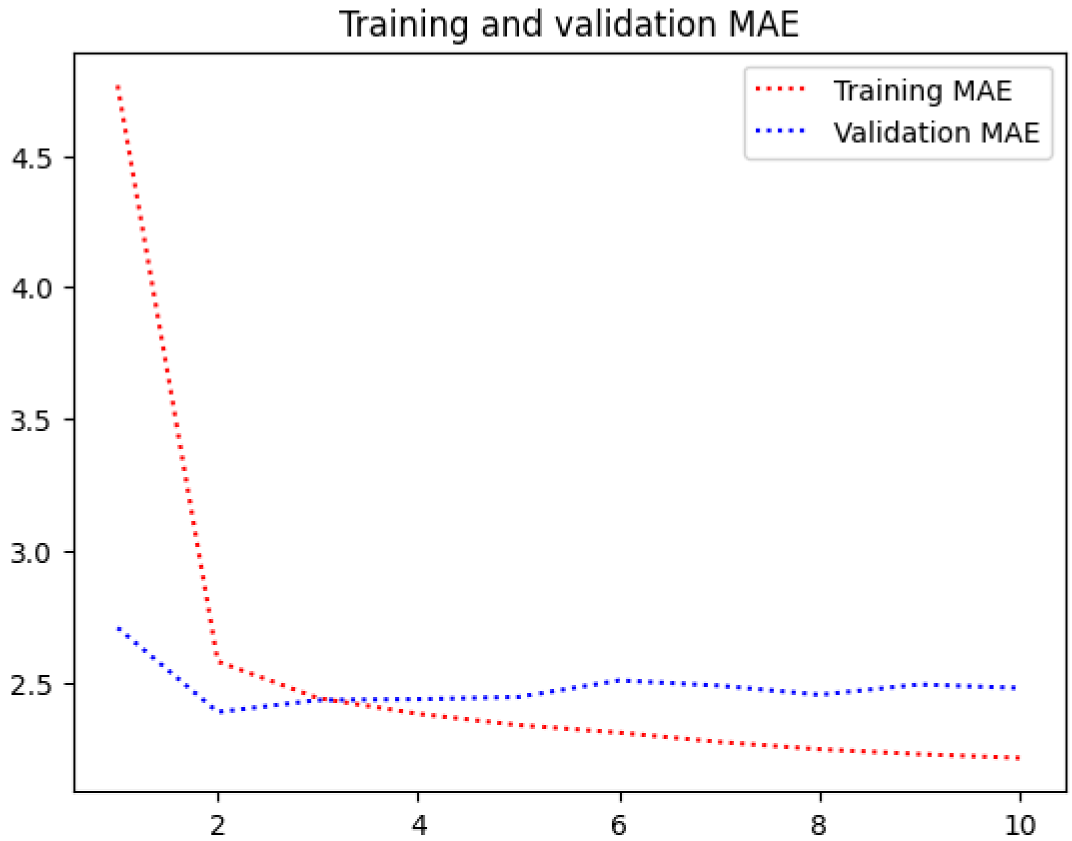
In [38]:

```
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.st",
                                    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.st")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
Epoch 1/10
819/819 [==============================] - 17s 18ms/step - loss: 42.4580 - mae: 4.
7673 - val_loss: 12.7456 - val_mae: 2.7065
Epoch 2/10
819/819 [==============================] - 14s 17ms/step - loss: 11.0033 - mae: 2.
5781 - val_loss: 9.4115 - val_mae: 2.3859
Epoch 3/10
819/819 [==============================] - 12s 15ms/step - loss: 9.7311 - mae: 2.4
391 - val_loss: 10.0108 - val_mae: 2.4313
Epoch 4/10
819/819 [==============================] - 13s 15ms/step - loss: 9.3278 - mae: 2.3
792 - val_loss: 10.1624 - val_mae: 2.4347
Epoch 5/10
819/819 [==============================] - 12s 15ms/step - loss: 9.0326 - mae: 2.3
363 - val_loss: 10.2920 - val_mae: 2.4430
Epoch 6/10
819/819 [==============================] - 12s 15ms/step - loss: 8.8320 - mae: 2.3
074 - val_loss: 11.2977 - val_mae: 2.5063
Epoch 7/10
819/819 [==============================] - 12s 15ms/step - loss: 8.5410 - mae: 2.2
720 - val_loss: 10.7163 - val_mae: 2.4862
Epoch 8/10
819/819 [==============================] - 12s 15ms/step - loss: 8.3147 - mae: 2.2
442 - val_loss: 10.1885 - val_mae: 2.4516
Epoch 9/10
819/819 [==============================] - 12s 15ms/step - loss: 8.1669 - mae: 2.2
257 - val_loss: 10.5671 - val_mae: 2.4908
Epoch 10/10
819/819 [==============================] - 12s 15ms/step - loss: 8.1200 - mae: 2.2
112 - val_loss: 10.3527 - val_mae: 2.4772
405/405 [==============================] - 4s 9ms/step - loss: 10.7846 - mae: 2.57
41
Test MAE: 2.57
```

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()

# Update the color to red and line style to a dotted line for training MAE
plt.plot(epochs, loss, "r:", label="Training MAE")

# Update the color to blue and line style to a dotted line for validation MAE
plt.plot(epochs, val_loss, "b:", label="Validation MAE")

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



LSTM - dropout (RNN) regularization that is frequently employed in sequential data-intensive activities like time series analysis and natural language processing. The purpose of LSTM networks is to identify long-term dependencies in data.

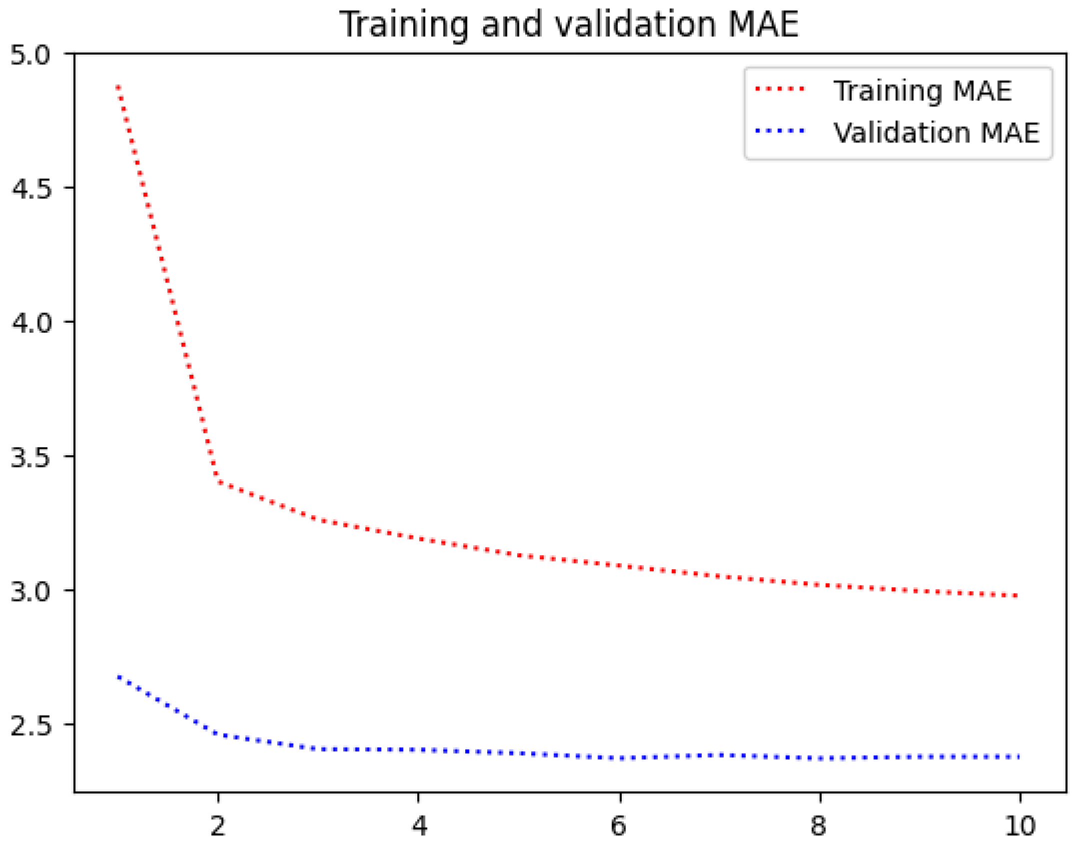
In [40]:
```
inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.LSTM(16, recurrent_dropout=0.25)(inputs)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1)(x)
model = keras.Model(inputs, outputs)

callbacks = [
    keras.callbacks.ModelCheckpoint("jena_lstm_dropout.st",
                                    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.st")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
WARNING:tensorflow:Layer lstm_1 will not use cuDNN kernels since it doesn't meet t
he criteria. It will use a generic GPU kernel as fallback when running on GPU.
```

```
Epoch 1/10
819/819 [==============================] - 197s 237ms/step - loss: 42.9776 - mae:
4.8761 - val_loss: 12.4762 - val_mae: 2.6726
Epoch 2/10
819/819 [==============================] - 194s 237ms/step - loss: 19.5298 - mae:
3.4003 - val_loss: 10.0883 - val_mae: 2.4555
Epoch 3/10
819/819 [==============================] - 194s 237ms/step - loss: 17.9057 - mae:
3.2574 - val_loss: 9.4769 - val_mae: 2.4014
Epoch 4/10
819/819 [==============================] - 194s 237ms/step - loss: 17.1586 - mae:
3.1876 - val_loss: 9.4717 - val_mae: 2.3990
Epoch 5/10
819/819 [==============================] - 191s 233ms/step - loss: 16.5084 - mae:
3.1250 - val_loss: 9.3487 - val_mae: 2.3862
Epoch 6/10
819/819 [==============================] - 192s 235ms/step - loss: 16.0133 - mae:
3.0863 - val_loss: 9.2829 - val_mae: 2.3673
Epoch 7/10
819/819 [==============================] - 192s 234ms/step - loss: 15.5740 - mae:
3.0463 - val_loss: 9.3961 - val_mae: 2.3797
Epoch 8/10
819/819 [==============================] - 191s 233ms/step - loss: 15.2843 - mae:
3.0143 - val_loss: 9.2679 - val_mae: 2.3668
Epoch 9/10
819/819 [==============================] - 189s 230ms/step - loss: 15.0070 - mae:
2.9914 - val_loss: 9.3266 - val_mae: 2.3732
Epoch 10/10
819/819 [==============================] - 189s 230ms/step - loss: 14.8669 - mae:
2.9738 - val_loss: 9.3490 - val_mae: 2.3731
WARNING:tensorflow:Layer lstm_1 will not use cuDNN kernels since it doesn't meet t
he criteria. It will use a generic GPU kernel as fallback when running on GPU.
405/405 [==============================] - 15s 36ms/step - loss: 10.7630 - mae: 2.
5822
Test MAE: 2.58
```

In [41]:

```python
import matplotlib.pyplot as plt

loss = history.history["mae"]
val_loss = history.history["val_mae"]
epochs = range(1, len(loss) + 1)

plt.figure()

# Update the color to red and line style to a dotted line for training MAE
plt.plot(epochs, loss, "r:", label="Training MAE")

# Update the color to blue and line style to a dotted line for validation MAE
plt.plot(epochs, val_loss, "b:", label="Validation MAE")

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



Using numerous LSTM layers placed on top of one another in a neural network is known as a stacked LSTM arrangement. With the ability to capture various levels of abstraction in the input data, each LSTM layer enhances the modeling capabilities of sequential information.

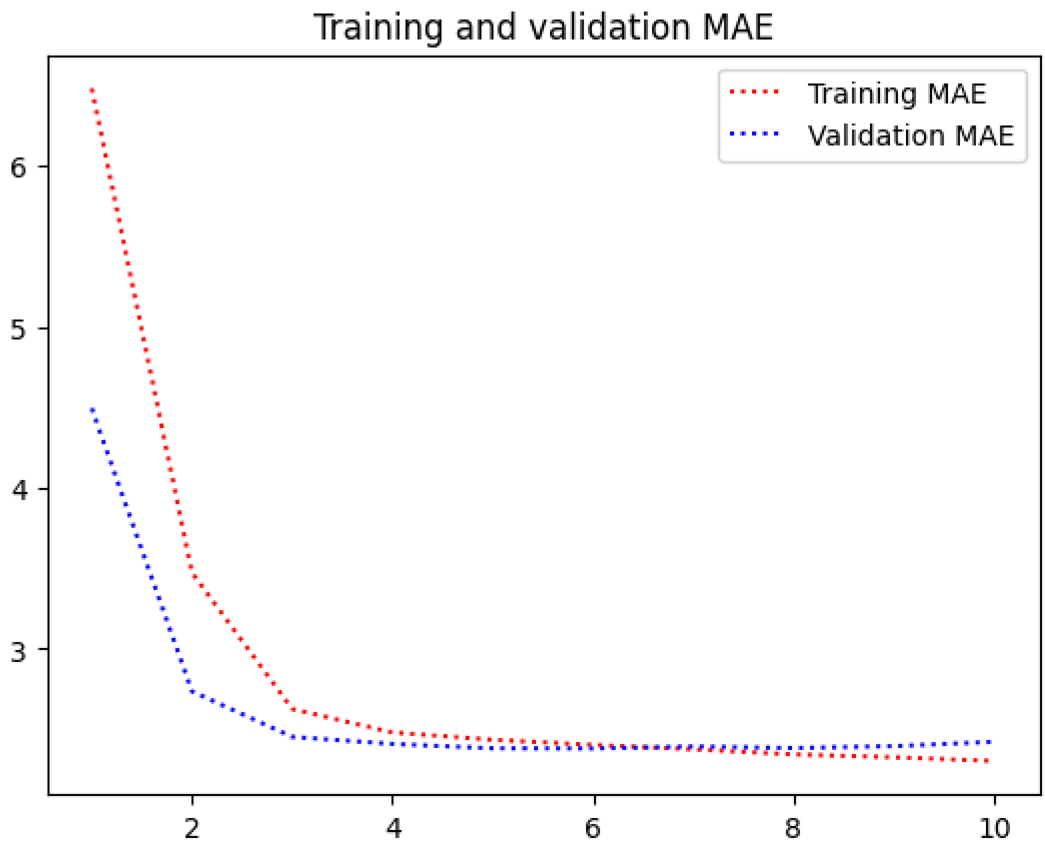
8 UNITS

In [42]:

```
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_stacked1.st",
                                    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.st")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
Epoch 1/10
819/819 [==============================] - 23s 25ms/step - loss: 71.6531 - mae: 6.
4814 - val_loss: 36.6526 - val_mae: 4.4927
Epoch 2/10
819/819 [==============================] - 19s 24ms/step - loss: 22.0938 - mae: 3.
4740 - val_loss: 13.2498 - val_mae: 2.7364
Epoch 3/10
819/819 [==============================] - 20s 25ms/step - loss: 11.5408 - mae: 2.
6272 - val_loss: 10.0911 - val_mae: 2.4532
Epoch 4/10
819/819 [==============================] - 19s 23ms/step - loss: 10.1388 - mae: 2.
4810 - val_loss: 9.6101 - val_mae: 2.4096
Epoch 5/10
819/819 [==============================] - 20s 24ms/step - loss: 9.7814 - mae: 2.4
355 - val_loss: 9.3762 - val_mae: 2.3821
Epoch 6/10
819/819 [==============================] - 14s 17ms/step - loss: 9.5184 - mae: 2.4
035 - val_loss: 9.4453 - val_mae: 2.3812
Epoch 7/10
819/819 [==============================] - 14s 18ms/step - loss: 9.3098 - mae: 2.3
761 - val_loss: 9.5335 - val_mae: 2.3936
Epoch 8/10
819/819 [==============================] - 14s 17ms/step - loss: 9.0812 - mae: 2.3
444 - val_loss: 9.5297 - val_mae: 2.3841
Epoch 9/10
819/819 [==============================] - 15s 18ms/step - loss: 8.9650 - mae: 2.3
266 - val_loss: 9.5346 - val_mae: 2.3958
Epoch 10/10
819/819 [==============================] - 14s 18ms/step - loss: 8.7747 - mae: 2.3
037 - val_loss: 9.8103 - val_mae: 2.4227
405/405 [==============================] - 5s 9ms/step - loss: 10.5657 - mae: 2.53
56
Test MAE: 2.54
```

In [43]:
```python
import matplotlib.pyplot as plt

loss = history.history["mae"]
val_loss = history.history["val_mae"]
epochs = range(1, len(loss) + 1)

plt.figure()

# Update the color to red and line style to a dotted line for training MAE
plt.plot(epochs, loss, "r:", label="Training MAE")

# Update the color to blue and line style to a dotted line for validation MAE
plt.plot(epochs, val_loss, "b:", label="Validation MAE")

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



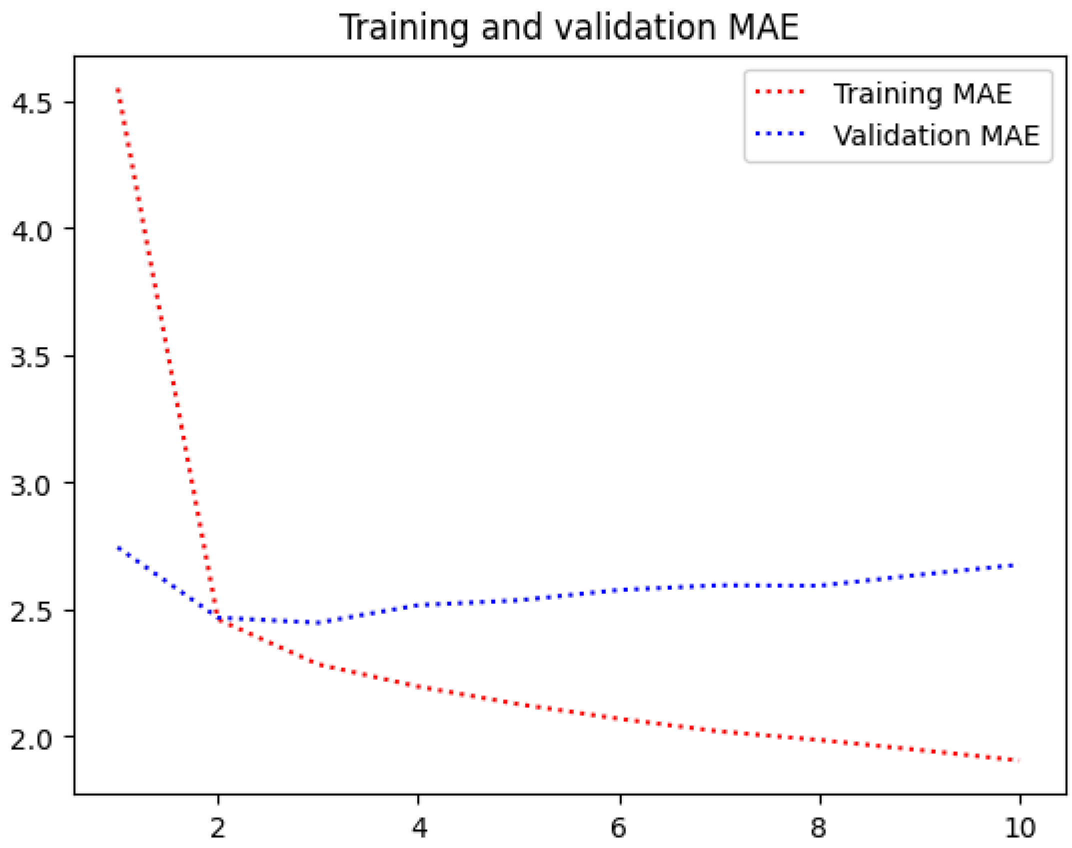
16 UNITS

In [44]:
```
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_stacked2.st",
                                    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.st")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
Epoch 1/10
819/819 [==============================] - 23s 24ms/step - loss: 38.8531 - mae: 4.
5511 - val_loss: 12.9220 - val_mae: 2.7445
Epoch 2/10
819/819 [==============================] - 20s 25ms/step - loss: 10.0461 - mae: 2.
4608 - val_loss: 10.1463 - val_mae: 2.4678
Epoch 3/10
819/819 [==============================] - 19s 23ms/step - loss: 8.6005 - mae: 2.2
838 - val_loss: 9.9778 - val_mae: 2.4478
Epoch 4/10
819/819 [==============================] - 15s 18ms/step - loss: 7.9527 - mae: 2.1
961 - val_loss: 10.4293 - val_mae: 2.5170
Epoch 5/10
819/819 [==============================] - 14s 17ms/step - loss: 7.4893 - mae: 2.1
269 - val_loss: 10.5627 - val_mae: 2.5360
Epoch 6/10
819/819 [==============================] - 14s 18ms/step - loss: 7.0908 - mae: 2.0
692 - val_loss: 10.8618 - val_mae: 2.5767
Epoch 7/10
819/819 [==============================] - 15s 18ms/step - loss: 6.7579 - mae: 2.0
196 - val_loss: 11.0478 - val_mae: 2.5953
Epoch 8/10
819/819 [==============================] - 15s 18ms/step - loss: 6.5339 - mae: 1.9
857 - val_loss: 10.9654 - val_mae: 2.5935
Epoch 9/10
819/819 [==============================] - 14s 18ms/step - loss: 6.2905 - mae: 1.9
462 - val_loss: 11.3620 - val_mae: 2.6366
Epoch 10/10
819/819 [==============================] - 14s 17ms/step - loss: 6.0317 - mae: 1.9
048 - val_loss: 11.6901 - val_mae: 2.6765
405/405 [==============================] - 5s 10ms/step - loss: 11.4495 - mae: 2.6
554
Test MAE: 2.66
```

In [45]:

```
import matplotlib.pyplot as plt

loss = history.history["mae"]
val_loss = history.history["val_mae"]
epochs = range(1, len(loss) + 1)

plt.figure()

# Update the color to red and line style to a dotted line for training MAE
plt.plot(epochs, loss, "r:", label="Training MAE")

# Update the color to blue and line style to a dotted line for validation MAE
plt.plot(epochs, val_loss, "b:", label="Validation MAE")

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



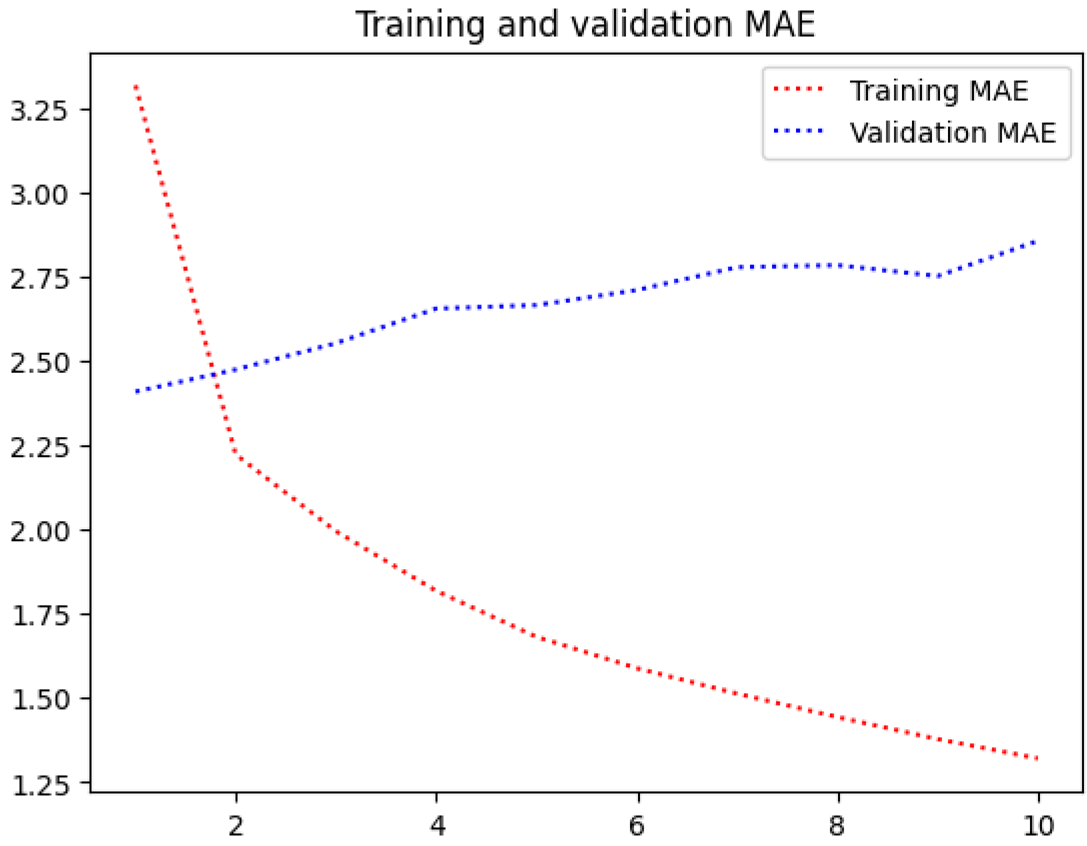
32 UNITS

In [46]:
```
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_stacked3.st",
                                    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.st")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
Epoch 1/10
819/819 [==============================] - 22s 24ms/step - loss: 21.6655 - mae: 3.
3170 - val_loss: 9.5987 - val_mae: 2.4080
Epoch 2/10
819/819 [==============================] - 14s 17ms/step - loss: 8.1561 - mae: 2.2
231 - val_loss: 10.0138 - val_mae: 2.4737
Epoch 3/10
819/819 [==============================] - 14s 18ms/step - loss: 6.5997 - mae: 1.9
952 - val_loss: 10.6741 - val_mae: 2.5519
Epoch 4/10
819/819 [==============================] - 15s 18ms/step - loss: 5.5523 - mae: 1.8
182 - val_loss: 11.5233 - val_mae: 2.6547
Epoch 5/10
819/819 [==============================] - 15s 18ms/step - loss: 4.8300 - mae: 1.6
816 - val_loss: 11.6769 - val_mae: 2.6649
Epoch 6/10
819/819 [==============================] - 14s 17ms/step - loss: 4.3394 - mae: 1.5
883 - val_loss: 12.1840 - val_mae: 2.7098
Epoch 7/10
819/819 [==============================] - 14s 17ms/step - loss: 3.9565 - mae: 1.5
128 - val_loss: 12.6120 - val_mae: 2.7776
Epoch 8/10
819/819 [==============================] - 14s 18ms/step - loss: 3.6129 - mae: 1.4
433 - val_loss: 12.8054 - val_mae: 2.7836
Epoch 9/10
819/819 [==============================] - 15s 18ms/step - loss: 3.3073 - mae: 1.3
779 - val_loss: 12.6290 - val_mae: 2.7513
Epoch 10/10
819/819 [==============================] - 15s 18ms/step - loss: 3.0445 - mae: 1.3
206 - val_loss: 13.3755 - val_mae: 2.8557
405/405 [==============================] - 4s 9ms/step - loss: 11.0538 - mae: 2.58
74
Test MAE: 2.59
```

In [47]:

```python
import matplotlib.pyplot as plt

loss = history.history["mae"]
val_loss = history.history["val_mae"]
epochs = range(1, len(loss) + 1)

plt.figure()

# Update the color to red and line style to a dotted line for training MAE
plt.plot(epochs, loss, "r:", label="Training MAE")

# Update the color to blue and line style to a dotted line for validation MAE
plt.plot(epochs, val_loss, "b:", label="Validation MAE")

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

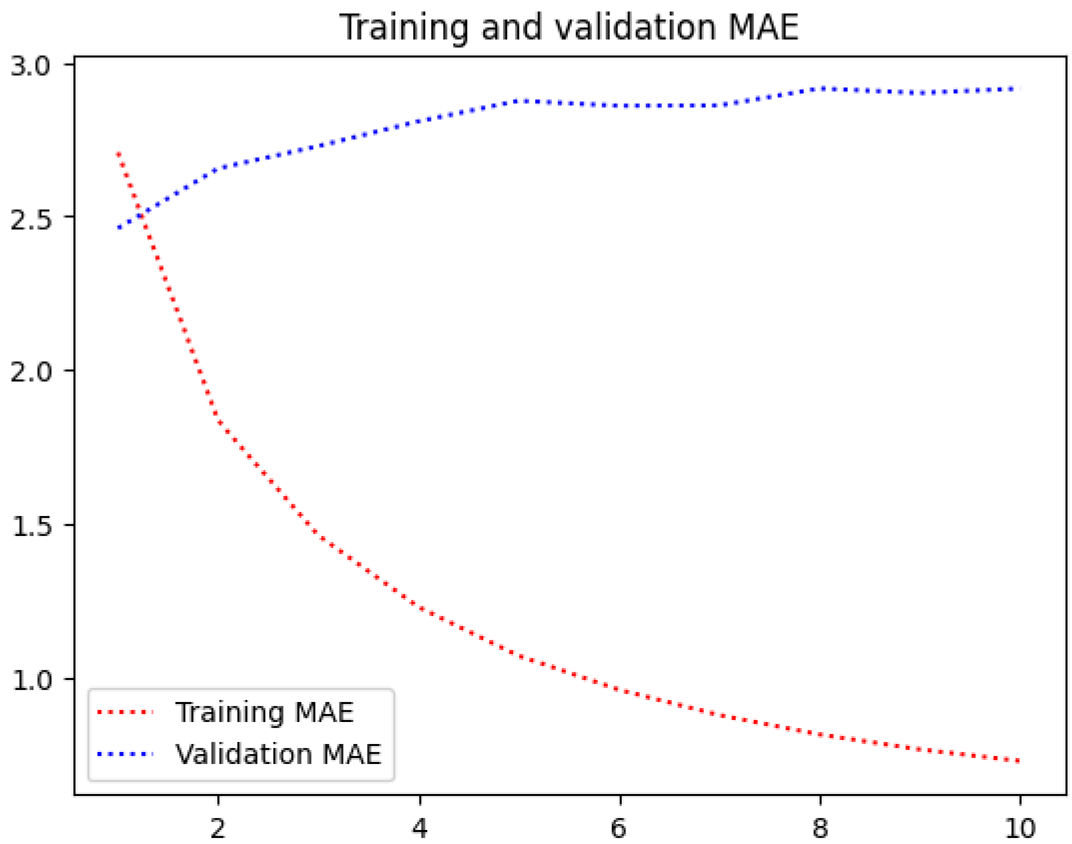


In [48]:
```
inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.LSTM(64, return_sequences=True)(inputs)
x = layers.LSTM(64)(x)
outputs = layers.Dense(1)(x)
model = keras.Model(inputs, outputs)

callbacks = [
    keras.callbacks.ModelCheckpoint("jena_LSTM_stacked4.st",
                                    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_stacked4.st")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
Epoch 1/10
819/819 [==============================] - 23s 26ms/step - loss: 13.4848 - mae: 2.
7081 - val_loss: 10.0222 - val_mae: 2.4617
Epoch 2/10
819/819 [==============================] - 15s 18ms/step - loss: 5.6859 - mae: 1.8
364 - val_loss: 11.5892 - val_mae: 2.6566
Epoch 3/10
819/819 [==============================] - 15s 18ms/step - loss: 3.6801 - mae: 1.4
630 - val_loss: 12.1547 - val_mae: 2.7290
Epoch 4/10
819/819 [==============================] - 15s 18ms/step - loss: 2.6351 - mae: 1.2
298 - val_loss: 12.9282 - val_mae: 2.8097
Epoch 5/10
819/819 [==============================] - 15s 18ms/step - loss: 2.0220 - mae: 1.0
707 - val_loss: 13.4367 - val_mae: 2.8769
Epoch 6/10
819/819 [==============================] - 15s 18ms/step - loss: 1.6255 - mae: 0.9
599 - val_loss: 13.4157 - val_mae: 2.8603
Epoch 7/10
819/819 [==============================] - 15s 18ms/step - loss: 1.3630 - mae: 0.8
776 - val_loss: 13.2941 - val_mae: 2.8618
Epoch 8/10
819/819 [==============================] - 15s 18ms/step - loss: 1.1796 - mae: 0.8
145 - val_loss: 13.7425 - val_mae: 2.9168
Epoch 9/10
819/819 [==============================] - 15s 18ms/step - loss: 1.0519 - mae: 0.7
659 - val_loss: 13.5821 - val_mae: 2.9022
Epoch 10/10
819/819 [==============================] - 15s 18ms/step - loss: 0.9551 - mae: 0.7
283 - val_loss: 13.7624 - val_mae: 2.9166
405/405 [==============================] - 4s 9ms/step - loss: 11.3783 - mae: 2.65
81
Test MAE: 2.66
```

In [49]:

```python
import matplotlib.pyplot as plt

loss = history.history["mae"]
val_loss = history.history["val_mae"]
epochs = range(1, len(loss) + 1)

plt.figure()

# Update the color to red and line style to a dotted line for training MAE
plt.plot(epochs, loss, "r:", label="Training MAE")

# Update the color to blue and line style to a dotted line for validation MAE
plt.plot(epochs, val_loss, "b:", label="Validation MAE")

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



64 UNITS

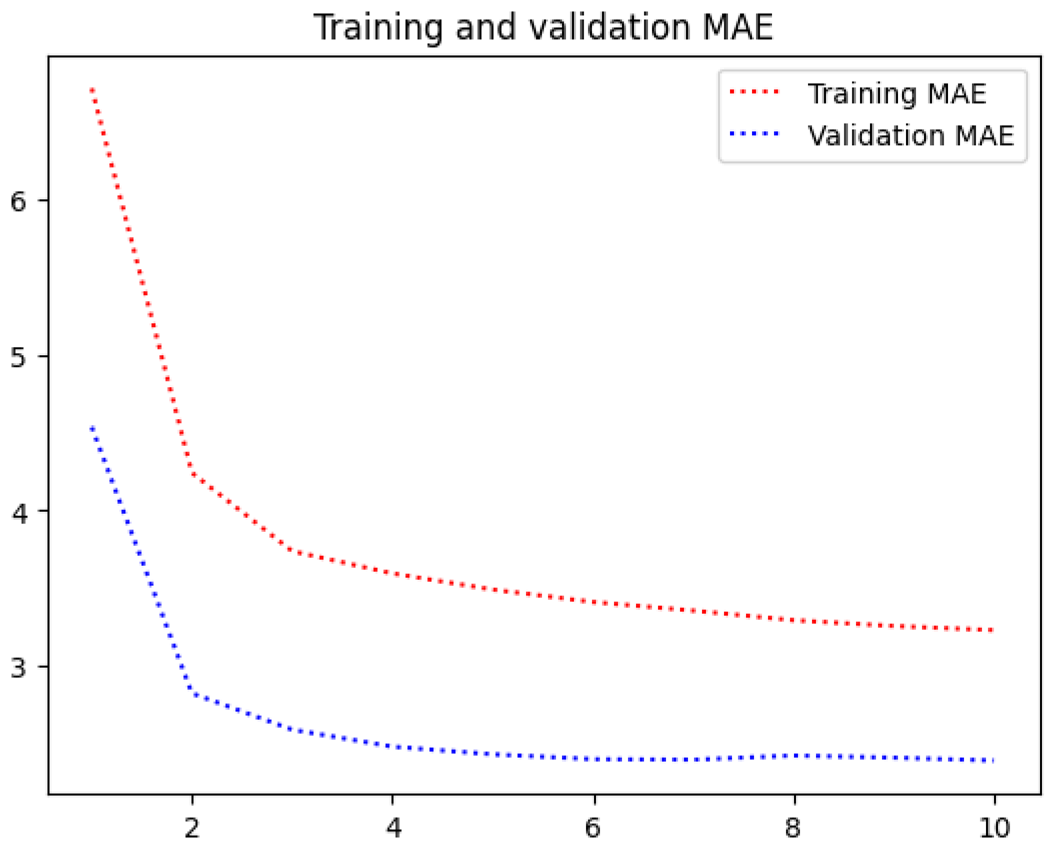
In [50]:
```
inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.LSTM(8, recurrent_dropout=0.5, return_sequences=True)(inputs)
x = layers.LSTM(8, recurrent_dropout=0.5)(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1)(x)
model = keras.Model(inputs, outputs)

callbacks = [
    keras.callbacks.ModelCheckpoint("jena_stacked_LSTM_dropout.st",
                                    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.st")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
WARNING:tensorflow:Layer lstm_10 will not use cuDNN kernels since it doesn't meet
the criteria. It will use a generic GPU kernel as fallback when running on GPU.
WARNING:tensorflow:Layer lstm_11 will not use cuDNN kernels since it doesn't meet
the criteria. It will use a generic GPU kernel as fallback when running on GPU.
```

```
Epoch 1/10
819/819 [==============================] - 409s 495ms/step - loss: 75.9007 - mae:
6.7153 - val_loss: 37.0579 - val_mae: 4.5377
Epoch 2/10
819/819 [==============================] - 404s 494ms/step - loss: 32.1463 - mae:
4.2431 - val_loss: 14.3928 - val_mae: 2.8240
Epoch 3/10
819/819 [==============================] - 406s 496ms/step - loss: 24.4282 - mae:
3.7393 - val_loss: 11.4730 - val_mae: 2.5901
Epoch 4/10
819/819 [==============================] - 410s 501ms/step - loss: 22.4957 - mae:
3.5958 - val_loss: 10.4573 - val_mae: 2.4805
Epoch 5/10
819/819 [==============================] - 409s 500ms/step - loss: 21.1179 - mae:
3.4919 - val_loss: 9.9832 - val_mae: 2.4313
Epoch 6/10
819/819 [==============================] - 409s 499ms/step - loss: 20.0924 - mae:
3.4115 - val_loss: 9.6812 - val_mae: 2.3999
Epoch 7/10
819/819 [==============================] - 409s 500ms/step - loss: 19.4278 - mae:
3.3561 - val_loss: 9.6462 - val_mae: 2.3966
Epoch 8/10
819/819 [==============================] - 405s 494ms/step - loss: 18.6520 - mae:
3.2944 - val_loss: 9.8613 - val_mae: 2.4231
Epoch 9/10
819/819 [==============================] - 400s 488ms/step - loss: 18.2118 - mae:
3.2571 - val_loss: 9.7556 - val_mae: 2.4095
Epoch 10/10
819/819 [==============================] - 408s 498ms/step - loss: 17.8725 - mae:
3.2307 - val_loss: 9.5700 - val_mae: 2.3912
WARNING:tensorflow:Layer lstm_10 will not use cuDNN kernels since it doesn't meet
the criteria. It will use a generic GPU kernel as fallback when running on GPU.
WARNING:tensorflow:Layer lstm_11 will not use cuDNN kernels since it doesn't meet
the criteria. It will use a generic GPU kernel as fallback when running on GPU.
405/405 [==============================] - 27s 66ms/step - loss: 10.9484 - mae: 2.
5809
Test MAE: 2.58
```

In [51]:

```python
import matplotlib.pyplot as plt

loss = history.history["mae"]
val_loss = history.history["val_mae"]
epochs = range(1, len(loss) + 1)

plt.figure()

# Update the color to red and line style to a dotted line for training MAE
plt.plot(epochs, loss, "r:", label="Training MAE")

# Update the color to blue and line style to a dotted line for validation MAE
plt.plot(epochs, val_loss, "b:", label="Validation MAE")

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



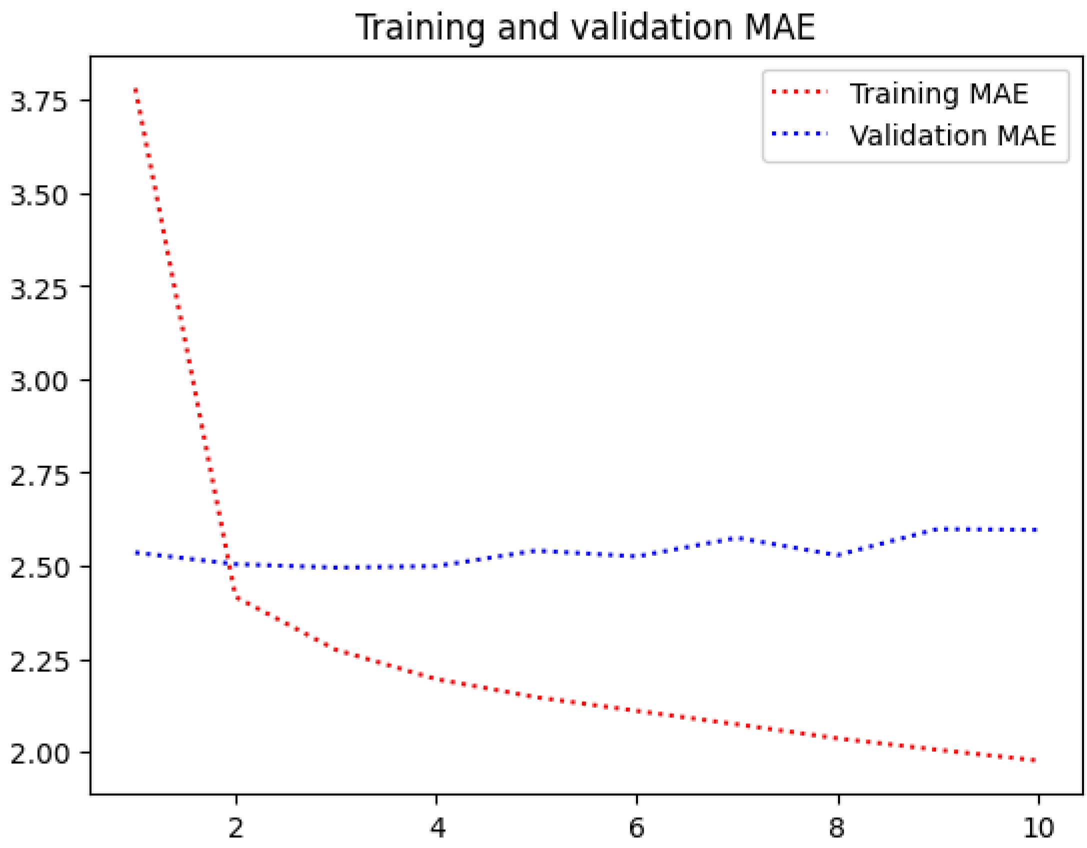
LSTM - dropout-regularized, stacked model

In [52]:
```
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.st",
                                    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.st")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
Epoch 1/10
819/819 [==============================] - 26s 29ms/step - loss: 27.5738 - mae: 3.
7810 - val_loss: 10.7398 - val_mae: 2.5353
Epoch 2/10
819/819 [==============================] - 23s 28ms/step - loss: 9.5760 - mae: 2.4
162 - val_loss: 10.4937 - val_mae: 2.5041
Epoch 3/10
819/819 [==============================] - 23s 28ms/step - loss: 8.5109 - mae: 2.2
743 - val_loss: 10.3009 - val_mae: 2.4950
Epoch 4/10
819/819 [==============================] - 14s 17ms/step - loss: 7.9285 - mae: 2.1
953 - val_loss: 10.4739 - val_mae: 2.4986
Epoch 5/10
819/819 [==============================] - 15s 18ms/step - loss: 7.5696 - mae: 2.1
465 - val_loss: 10.8696 - val_mae: 2.5405
Epoch 6/10
819/819 [==============================] - 14s 17ms/step - loss: 7.2944 - mae: 2.1
099 - val_loss: 10.7107 - val_mae: 2.5248
Epoch 7/10
819/819 [==============================] - 15s 18ms/step - loss: 7.0447 - mae: 2.0
738 - val_loss: 11.1661 - val_mae: 2.5750
Epoch 8/10
819/819 [==============================] - 14s 17ms/step - loss: 6.7834 - mae: 2.0
360 - val_loss: 10.8024 - val_mae: 2.5282
Epoch 9/10
819/819 [==============================] - 14s 18ms/step - loss: 6.5815 - mae: 2.0
054 - val_loss: 11.4922 - val_mae: 2.5985
Epoch 10/10
819/819 [==============================] - 14s 17ms/step - loss: 6.4082 - mae: 1.9
766 - val_loss: 11.3771 - val_mae: 2.5963
405/405 [==============================] - 5s 9ms/step - loss: 10.4551 - mae: 2.54
43
Test MAE: 2.54
```

In [53]:

```python
import matplotlib.pyplot as plt

loss = history.history["mae"]
val_loss = history.history["val_mae"]
epochs = range(1, len(loss) + 1)

plt.figure()

# Update the color to red and line style to a dotted line for training MAE
plt.plot(epochs, loss, "r:", label="Training MAE")

# Update the color to blue and line style to a dotted line for validation MAE
plt.plot(epochs, val_loss, "b:", label="Validation MAE")

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



Bi Directional LSTM

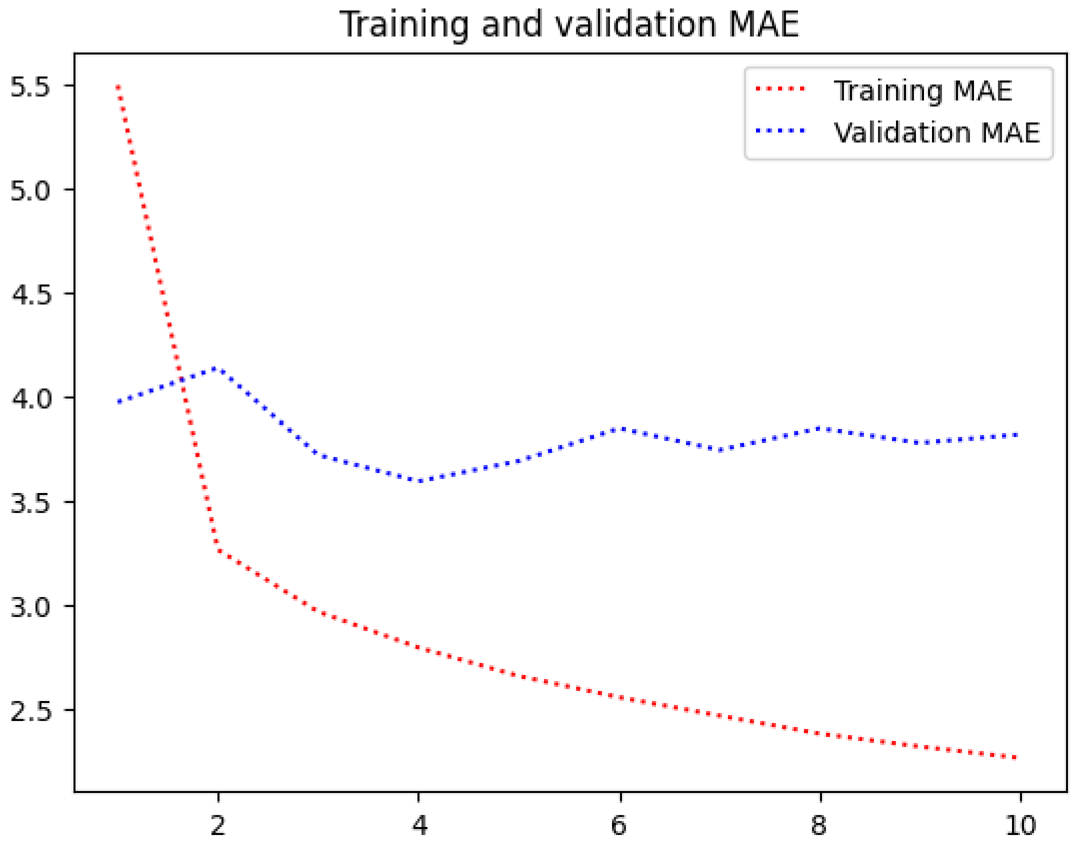
In [54]:

```
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.st", save_best_only=True)
]

history = model.fit(train_dataset, epochs=10, validation_data=val_dataset, callback

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

```
Epoch 1/10
819/819 [==============================] - 17s 18ms/step - loss: 53.9860 - mae: 5.
4955 - val_loss: 26.9692 - val_mae: 3.9739
Epoch 2/10
819/819 [==============================] - 12s 14ms/step - loss: 18.0204 - mae: 3.
2664 - val_loss: 28.0418 - val_mae: 4.1391
Epoch 3/10
819/819 [==============================] - 14s 17ms/step - loss: 14.7304 - mae: 2.
9691 - val_loss: 22.8467 - val_mae: 3.7222
Epoch 4/10
819/819 [==============================] - 14s 17ms/step - loss: 13.1107 - mae: 2.
7968 - val_loss: 21.0134 - val_mae: 3.5956
Epoch 5/10
819/819 [==============================] - 12s 14ms/step - loss: 11.9008 - mae: 2.
6598 - val_loss: 21.6781 - val_mae: 3.6927
Epoch 6/10
819/819 [==============================] - 12s 14ms/step - loss: 11.0358 - mae: 2.
5573 - val_loss: 24.4002 - val_mae: 3.8503
Epoch 7/10
819/819 [==============================] - 11s 14ms/step - loss: 10.3269 - mae: 2.
4694 - val_loss: 21.8322 - val_mae: 3.7457
Epoch 8/10
819/819 [==============================] - 12s 14ms/step - loss: 9.6489 - mae: 2.3
826 - val_loss: 23.0966 - val_mae: 3.8502
Epoch 9/10
819/819 [==============================] - 12s 14ms/step - loss: 9.1947 - mae: 2.3
206 - val_loss: 23.2724 - val_mae: 3.7798
Epoch 10/10
819/819 [==============================] - 12s 14ms/step - loss: 8.7746 - mae: 2.2
658 - val_loss: 23.4737 - val_mae: 3.8206
405/405 [==============================] - 4s 9ms/step - loss: 23.0256 - mae: 3.77
41
Test MAE: 3.77
```

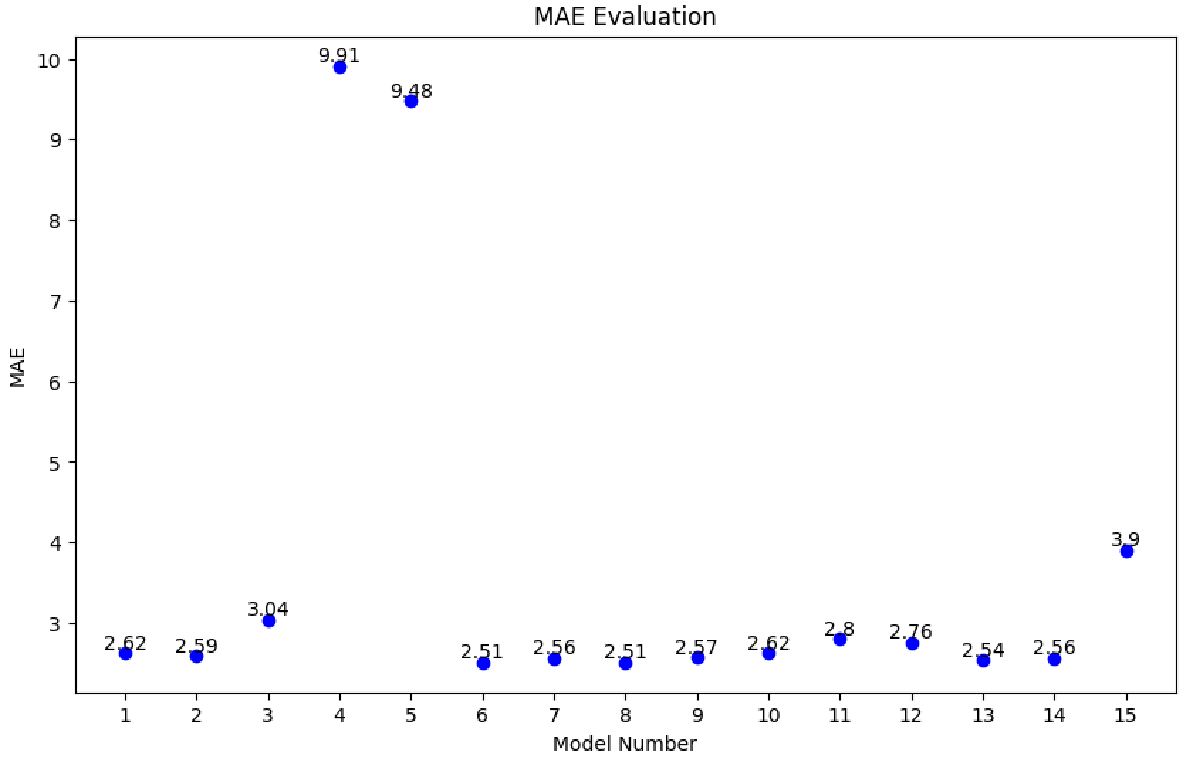
In [55]:

```python
import matplotlib.pyplot as plt

loss = history.history["mae"]
val_loss = history.history["val_mae"]
epochs = range(1, len(loss) + 1)

plt.figure()

# Update the color to red and line style to a dotted line for training MAE
plt.plot(epochs, loss, "r:", label="Training MAE")

# Update the color to blue and line style to a dotted line for validation MAE
plt.plot(epochs, val_loss, "b:", label="Validation MAE")

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



1D Convnets and LSTM togther

In [56]:
```
import matplotlib.pyplot as plt
import numpy as np

Models = ("1","2","3","4","5","6","7","8","9","10","11","12","13","14","15")
Mae = (2.62,2.59,3.04,9.91,9.48,2.51,2.56,2.51,2.57,2.62,2.80,2.76,2.54,2.56,3.90)

# MAE Evaluation
plt.figure(figsize=(10, 6))
plt.scatter(Models, Mae, color="blue")
plt.title("MAE Evaluation")
plt.xlabel("Model Number")
plt.ylabel("MAE")

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

plt.show()
```


MAE Evaluation
MAE
Model Number
10
9
8
7
6
5
4
3
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
2.62
2.59
3.04
9.91
9.48
2.51
2.56
2.51
2.57
2.62
2.8
2.76
2.54
2.56
3.9