Assignment3_yli130

November 5, 2023

1 Assignment_3

Yanxi Li, yli130@kent.edu

In this assignment, I run 7 different models to test/learn the time-series deep learning model. **The best performance is 2 stacked GRU(32) and 2 stacked GRU(16), their test set MAE are both 2.46**.

The 7 models test set MAE are:
1. 2 stacked GRU with 32 units: 2.46
2. 2 stacked GRU with 16 units: 2.45
3. 2 stacked LSTM with 32 units: 2.56
4. 2 stacked LSTM with 16 units: 2.55
5. 1 conv1D combine 1 GRU 16 units: 2.54
6. 1 conv1D combine 1 LSTM 16 units: 2.59
7. 1 conv1D combine 1 SimpleRNN 16 units: 2.53

Because of the computational resource limit, for stacked LSTM 32 units I only run for 12 epochs, and other models is 20 epochs. I think the result above is not reliable because some of the model I cannot tell the overfit region because of the less epochs. But this is the best I can do to finish this assignment.

1.0.1 Import the data

[1]:
```
#download data
#!wget https://s3.amazonaws.com/keras-datasets/jena_climate_2009_2016.csv.zip
#!unzip jena_climate_2009_2016.csv.zip
```

```
--2023-11-03 20:25:18--  https://s3.amazonaws.com/keras-
datasets/jena_climate_2009_2016.csv.zip
Resolving s3.amazonaws.com (s3.amazonaws.com)… 16.182.70.128, 52.217.142.208,
52.217.165.128, …
Connecting to s3.amazonaws.com (s3.amazonaws.com)|16.182.70.128|:443…
connected.
HTTP request sent, awaiting response… 200 OK
Length: 13565642 (13M) [application/zip]
Saving to: ‘jena_climate_2009_2016.csv.zip’

100%[======================================>] 13,565,642  6.73MB/s   in 1.9s
```

```
2023-11-03 20:25:21 (6.73 MB/s) - ‘jena_climate_2009_2016.csv.zip’ saved
[13565642/13565642]

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

1.0.2 Import the libraries & data

[1]:

```python
from tensorflow import keras
from tensorflow.keras import layers
import os
import numpy as np
from matplotlib import pyplot as plt
```

[2]:

```python
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))
```

```
['"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
```

1.0.3 Parsing the data

[3]:

```python
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[:]
```

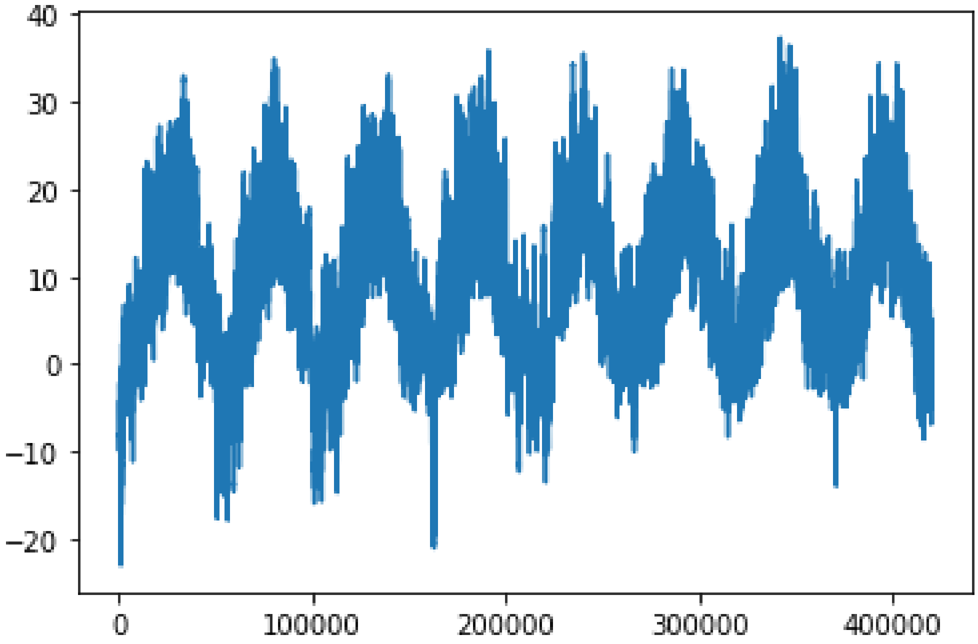
1.0.4 Plot the temperature figure

[5]:

```python
plt.plot(range(len(temperature)), temperature)
```

[5]:

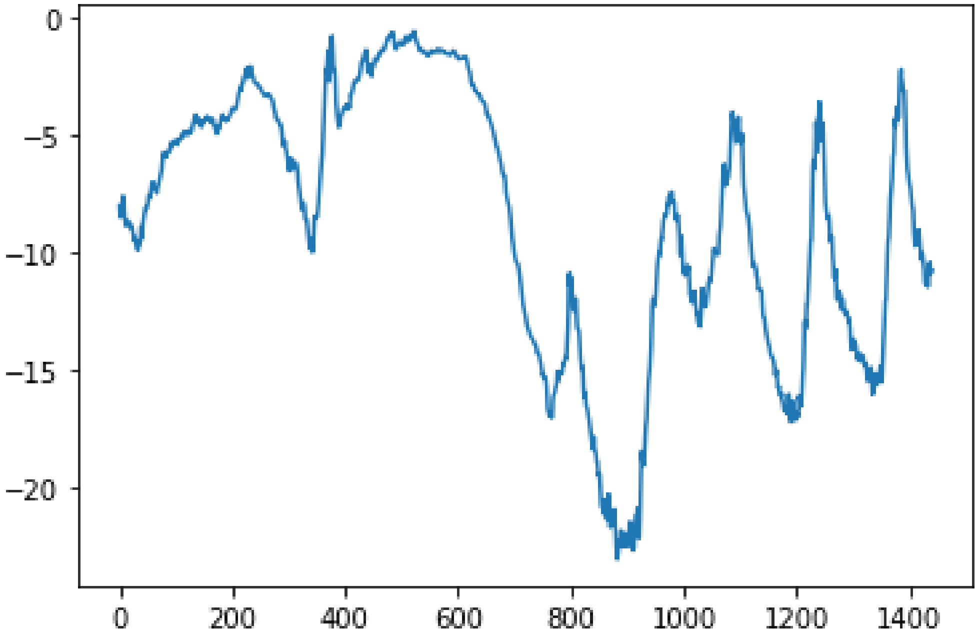
```
[<matplotlib.lines.Line2D at 0x7f1c372adda0>]
```



1.0.5 Plot the first 10 days temperature

```
[5]: plt.plot(range(1440), temperature[:1440])
```

```
[5]: [<matplotlib.lines.Line2D at 0x7f3341cfbb00>]
```



1.0.6 Computing the number of samples

[4]:

```
num_train_samples = int(0.5 * len(raw_data))
num_val_samples = int(0.25 * len(raw_data))
num_test_samples = len(raw_data) - num_train_samples - num_val_samples
print("num_train_samples:", num_train_samples)
print("num_val_samples:", num_val_samples)
print("num_test_samples:", num_test_samples)
```

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

1.0.7 Normalize the data

[5]:

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

1.0.8 Instantiating datasets for training, validation, and testing

[6]:

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

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

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

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

[9]:

```
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,)
```

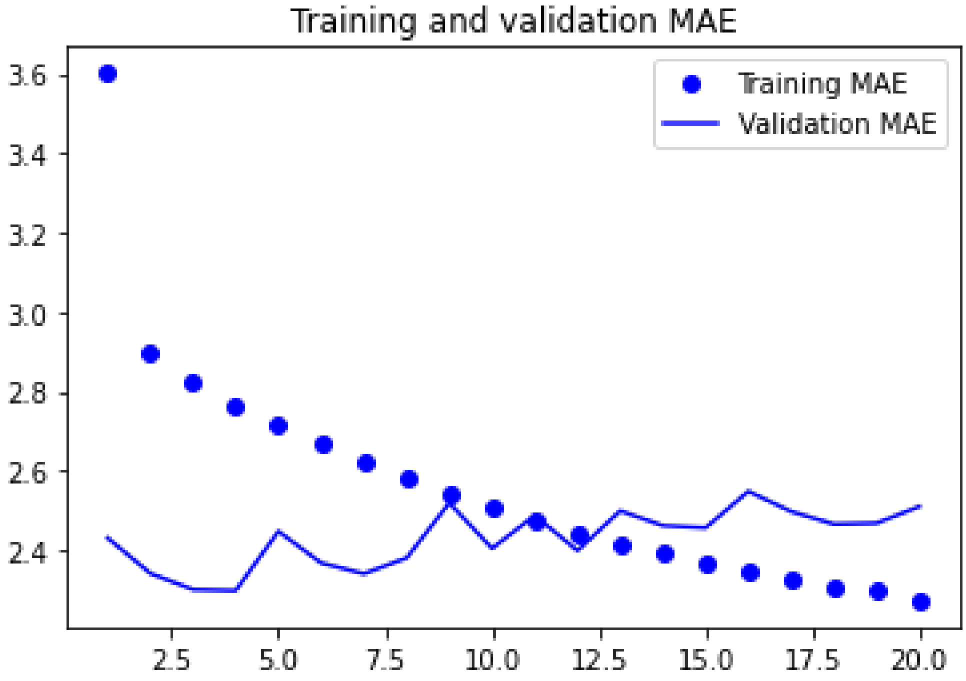
1.0.9 Build and plot the figure

[7]:

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

callbacks = [
    keras.callbacks.ModelCheckpoint("jena_stacked_gru_dropout.keras",
                                    save_best_only=True)
]
model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
history = model.fit(train_dataset,
                    epochs=20,
                    validation_data=val_dataset,
                    callbacks=callbacks)

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

```
Epoch 1/20
819/819 [==============================] - 205s 246ms/step - loss: 23.5106 -
mae: 3.6022 - val_loss: 9.7931 - val_mae: 2.4317
Epoch 2/20
819/819 [==============================] - 198s 242ms/step - loss: 13.9951 -
mae: 2.8991 - val_loss: 9.1079 - val_mae: 2.3426
Epoch 3/20
819/819 [==============================] - 196s 239ms/step - loss: 13.2993 -
mae: 2.8229 - val_loss: 8.8294 - val_mae: 2.3021
Epoch 4/20
819/819 [==============================] - 199s 242ms/step - loss: 12.7014 -
mae: 2.7632 - val_loss: 8.8616 - val_mae: 2.2997
Epoch 5/20
819/819 [==============================] - 197s 241ms/step - loss: 12.2640 -
mae: 2.7170 - val_loss: 10.0299 - val_mae: 2.4483
Epoch 6/20
819/819 [==============================] - 200s 244ms/step - loss: 11.8404 -
mae: 2.6684 - val_loss: 9.4310 - val_mae: 2.3685
Epoch 7/20
819/819 [==============================] - 199s 242ms/step - loss: 11.3940 -
mae: 2.6232 - val_loss: 9.2537 - val_mae: 2.3417
Epoch 8/20
819/819 [==============================] - 199s 243ms/step - loss: 11.0363 -
mae: 2.5805 - val_loss: 9.4793 - val_mae: 2.3812
Epoch 9/20
819/819 [==============================] - 199s 243ms/step - loss: 10.6818 -
mae: 2.5420 - val_loss: 10.6818 - val_mae: 2.5199
Epoch 10/20
819/819 [==============================] - 199s 243ms/step - loss: 10.3910 -
mae: 2.5058 - val_loss: 9.8492 - val_mae: 2.4053
Epoch 11/20
819/819 [==============================] - 199s 243ms/step - loss: 10.1021 -
mae: 2.4757 - val_loss: 10.3068 - val_mae: 2.4884
Epoch 12/20
819/819 [==============================] - 199s 243ms/step - loss: 9.8558 - mae:
2.4434 - val_loss: 9.6701 - val_mae: 2.3996
Epoch 13/20
819/819 [==============================] - 199s 243ms/step - loss: 9.6033 - mae:
2.4126 - val_loss: 10.4441 - val_mae: 2.5000
Epoch 14/20
819/819 [==============================] - 199s 243ms/step - loss: 9.4312 - mae:
2.3919 - val_loss: 10.1139 - val_mae: 2.4627
Epoch 15/20
819/819 [==============================] - 197s 241ms/step - loss: 9.2370 - mae:
2.3710 - val_loss: 10.0406 - val_mae: 2.4573
```

```
Epoch 16/20
819/819 [==============================] - 200s 244ms/step - loss: 9.0774 - mae:
2.3477 - val_loss: 11.0705 - val_mae: 2.5491
Epoch 17/20
819/819 [==============================] - 198s 242ms/step - loss: 8.9337 - mae:
2.3298 - val_loss: 10.6044 - val_mae: 2.4983
Epoch 18/20
819/819 [==============================] - 199s 243ms/step - loss: 8.7845 - mae:
2.3104 - val_loss: 10.2998 - val_mae: 2.4665
Epoch 19/20
819/819 [==============================] - 197s 241ms/step - loss: 8.6902 - mae:
2.2982 - val_loss: 10.3484 - val_mae: 2.4688
Epoch 20/20
819/819 [==============================] - 199s 243ms/step - loss: 8.5049 - mae:
2.2742 - val_loss: 10.5708 - val_mae: 2.5108
```



[8]:
```
model = keras.models.load_model("jena_stacked_gru_dropout.keras")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
405/405 [==============================] - 20s 48ms/step - loss: 9.8399 - mae:
2.4558
Test MAE: 2.46
```

Question_1

November 5, 2023

0.1 Question_1

0.1.1 Import libraries and dataset

[ ]:

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

[1]:

```
from tensorflow import keras
from tensorflow.keras import layers
import os
import numpy as np
from matplotlib import pyplot as plt
```

[2]:

```
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))
```

```
['"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
```

0.1.2 Parsing the data

[3]:

```
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[:]
```

0.1.3 Number of samples in each set

[4]:

```
num_train_samples = int(0.5 * len(raw_data))
num_val_samples = int(0.25 * len(raw_data))
num_test_samples = len(raw_data) - num_train_samples - num_val_samples
print("num_train_samples:", num_train_samples)
print("num_val_samples:", num_val_samples)
print("num_test_samples:", num_test_samples)
```

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

0.1.4 Normalize the data

[5]:

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

0.1.5 Datasets for training, validation, testing

[6]:

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

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

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

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

0.1.6 Build, train, evaluate the model

[7]:

```
inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.GRU(16, recurrent_dropout=0.5, return_sequences=True)(inputs)
x = layers.GRU(16, 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_gru_16_dropout.keras",
                                    save_best_only=True)
]
model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
history = model.fit(train_dataset,
                    epochs=20,
                    validation_data=val_dataset,
                    callbacks=callbacks)
```

```
Epoch 1/20
819/819 [==============================] - 187s 224ms/step - loss: 44.3168 -
mae: 4.9178 - val_loss: 13.0437 - val_mae: 2.6947
Epoch 2/20
819/819 [==============================] - 181s 221ms/step - loss: 18.7848 -
mae: 3.3210 - val_loss: 9.8564 - val_mae: 2.4285
Epoch 3/20
819/819 [==============================] - 181s 221ms/step - loss: 17.1242 -
mae: 3.1760 - val_loss: 9.5982 - val_mae: 2.4058
Epoch 4/20
819/819 [==============================] - 182s 222ms/step - loss: 16.1844 -
mae: 3.0943 - val_loss: 9.9358 - val_mae: 2.4528
Epoch 5/20
819/819 [==============================] - 181s 221ms/step - loss: 15.5898 -
mae: 3.0376 - val_loss: 8.9548 - val_mae: 2.3261
Epoch 6/20
```

```
819/819 [==============================] - 180s 220ms/step - loss: 14.9848 -
mae: 2.9817 - val_loss: 9.2634 - val_mae: 2.3683
Epoch 7/20
819/819 [==============================] - 182s 222ms/step - loss: 14.5051 -
mae: 2.9329 - val_loss: 8.6872 - val_mae: 2.2881
Epoch 8/20
819/819 [==============================] - 179s 219ms/step - loss: 14.0979 -
mae: 2.8871 - val_loss: 8.7833 - val_mae: 2.2927
Epoch 9/20
819/819 [==============================] - 183s 223ms/step - loss: 13.7109 -
mae: 2.8482 - val_loss: 8.6431 - val_mae: 2.2723
Epoch 10/20
819/819 [==============================] - 181s 221ms/step - loss: 13.4319 -
mae: 2.8226 - val_loss: 8.7329 - val_mae: 2.2920
Epoch 11/20
819/819 [==============================] - 181s 221ms/step - loss: 13.1444 -
mae: 2.7955 - val_loss: 8.7241 - val_mae: 2.2872
Epoch 12/20
819/819 [==============================] - 180s 219ms/step - loss: 12.9758 -
mae: 2.7797 - val_loss: 9.2705 - val_mae: 2.3578
Epoch 13/20
819/819 [==============================] - 179s 219ms/step - loss: 12.8320 -
mae: 2.7617 - val_loss: 8.9951 - val_mae: 2.3171
Epoch 14/20
819/819 [==============================] - 180s 220ms/step - loss: 12.7110 -
mae: 2.7502 - val_loss: 8.5788 - val_mae: 2.2639
Epoch 15/20
819/819 [==============================] - 179s 218ms/step - loss: 12.5661 -
mae: 2.7339 - val_loss: 8.6823 - val_mae: 2.2793
Epoch 16/20
819/819 [==============================] - 180s 220ms/step - loss: 12.5253 -
mae: 2.7314 - val_loss: 8.7688 - val_mae: 2.2847
Epoch 17/20
819/819 [==============================] - 179s 218ms/step - loss: 12.4073 -
mae: 2.7185 - val_loss: 8.6872 - val_mae: 2.2759
Epoch 18/20
819/819 [==============================] - 180s 219ms/step - loss: 12.3845 -
mae: 2.7134 - val_loss: 8.8227 - val_mae: 2.2914
Epoch 19/20
819/819 [==============================] - 175s 213ms/step - loss: 12.3413 -
mae: 2.7118 - val_loss: 8.7773 - val_mae: 2.2884
Epoch 20/20
819/819 [==============================] - 175s 213ms/step - loss: 12.2505 -
mae: 2.7025 - val_loss: 8.9179 - val_mae: 2.3145
```

[7]:

```
model = keras.models.load_model("jena_stacked_gru_16_dropout.keras")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
405/405 [==============================] - 20s 47ms/step - loss: 9.8117 - mae:
2.4558
Test MAE: 2.46
```

Question_2

November 5, 2023

0.1 Question_2

0.1.1 Import libraries and dataset

[1]:

```
from tensorflow import keras
from tensorflow.keras import layers
import os
import numpy as np
from matplotlib import pyplot as plt
```

[2]:

```
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))
```

```
['"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
```

0.1.2 Parsing the data

[3]:

```
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[:]
```

0.1.3 Number of samples in each set

[4]:

```
num_train_samples = int(0.5 * len(raw_data))
num_val_samples = int(0.25 * len(raw_data))
num_test_samples = len(raw_data) - num_train_samples - num_val_samples
print("num_train_samples:", num_train_samples)
print("num_val_samples:", num_val_samples)
print("num_test_samples:", num_test_samples)
```

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

0.1.4 Normalize the data

[5]:

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

0.1.5 Datasets for training, validation, testing

[6]:

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

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

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

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

0.1.6 Build, train, evaluate the model

[8]:

```
inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.LSTM(32, recurrent_dropout=0.5, return_sequences=True)(inputs)
x = layers.LSTM(32, 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("stacked_lstm_dropout.keras",
                                    save_best_only=True)
]
model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
history = model.fit(train_dataset,
                    epochs=12,
                    validation_data=val_dataset,
                    callbacks=callbacks)
```

```
Epoch 1/12
819/819 [==============================] - 259s 312ms/step - loss: 26.1377 -
mae: 3.7636 - val_loss: 9.5032 - val_mae: 2.3740
Epoch 2/12
819/819 [==============================] - 254s 310ms/step - loss: 13.5765 -
mae: 2.8523 - val_loss: 10.1472 - val_mae: 2.4908
Epoch 3/12
819/819 [==============================] - 253s 309ms/step - loss: 12.4085 -
mae: 2.7236 - val_loss: 10.5108 - val_mae: 2.5405
Epoch 4/12
819/819 [==============================] - 254s 310ms/step - loss: 11.5711 -
mae: 2.6272 - val_loss: 10.3090 - val_mae: 2.5199
Epoch 5/12
819/819 [==============================] - 254s 311ms/step - loss: 10.9207 -
mae: 2.5513 - val_loss: 10.4561 - val_mae: 2.5414
Epoch 6/12
819/819 [==============================] - 254s 310ms/step - loss: 10.3078 -
mae: 2.4752 - val_loss: 11.3725 - val_mae: 2.6460
Epoch 7/12
```

```
819/819 [==============================] - 254s 310ms/step - loss: 9.7701 - mae:
2.4128 - val_loss: 10.5508 - val_mae: 2.5454
Epoch 8/12
819/819 [==============================] - 254s 310ms/step - loss: 9.3689 - mae:
2.3594 - val_loss: 11.1275 - val_mae: 2.6315
Epoch 9/12
819/819 [==============================] - 252s 307ms/step - loss: 8.9764 - mae:
2.3080 - val_loss: 10.6356 - val_mae: 2.5697
Epoch 10/12
819/819 [==============================] - 255s 311ms/step - loss: 8.6662 - mae:
2.2646 - val_loss: 10.8489 - val_mae: 2.5829
Epoch 11/12
819/819 [==============================] - 251s 306ms/step - loss: 8.3455 - mae:
2.2277 - val_loss: 10.9313 - val_mae: 2.5965
Epoch 12/12
819/819 [==============================] - 250s 306ms/step - loss: 8.1114 - mae:
2.1918 - val_loss: 11.1819 - val_mae: 2.6236
```

[9]:

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

```
405/405 [==============================] - 22s 54ms/step - loss: 10.9082 - mae:
2.5564
Test MAE: 2.56
```

[ ]:

```python

```

Question_2_2

November 5, 2023

0.1 Question_2_2

0.1.1 Import libraries and dataset

[1]:
```
from tensorflow import keras
from tensorflow.keras import layers
import os
import numpy as np
from matplotlib import pyplot as plt
```

[2]:
```
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))
```

```
['"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
```

0.1.2 Parsing the data

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

0.1.3 Number of samples in each set

[4]:

```
num_train_samples = int(0.5 * len(raw_data))
num_val_samples = int(0.25 * len(raw_data))
num_test_samples = len(raw_data) - num_train_samples - num_val_samples
print("num_train_samples:", num_train_samples)
print("num_val_samples:", num_val_samples)
print("num_test_samples:", num_test_samples)
```

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

0.1.4 Normalize the data

[5]:

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

0.1.5 Datasets for training, validation, testing

[6]:

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

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

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

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

0.1.6 Build, train, evaluate the model

[7]:

```
inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.LSTM(16, recurrent_dropout=0.5, return_sequences=True)(inputs)
x = layers.LSTM(16, 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("stacked_lstm_16_dropout.keras",
                                    save_best_only=True)
]
model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
history = model.fit(train_dataset,
                    epochs=20,
                    validation_data=val_dataset,
                    callbacks=callbacks)
```

```
Epoch 1/20
819/819 [==============================] - 238s 286ms/step - loss: 43.2598 -
mae: 4.8651 - val_loss: 12.7110 - val_mae: 2.6897
Epoch 2/20
819/819 [==============================] - 231s 282ms/step - loss: 18.7576 -
mae: 3.3249 - val_loss: 10.0697 - val_mae: 2.4613
Epoch 3/20
819/819 [==============================] - 230s 281ms/step - loss: 17.1360 -
mae: 3.1803 - val_loss: 9.8739 - val_mae: 2.4473
Epoch 4/20
819/819 [==============================] - 231s 282ms/step - loss: 16.2845 -
mae: 3.0986 - val_loss: 9.9046 - val_mae: 2.4385
Epoch 5/20
819/819 [==============================] - 232s 283ms/step - loss: 15.3963 -
mae: 3.0160 - val_loss: 9.4869 - val_mae: 2.3876
Epoch 6/20
819/819 [==============================] - 232s 283ms/step - loss: 14.7372 -
mae: 2.9509 - val_loss: 9.2155 - val_mae: 2.3547
Epoch 7/20
```

```
819/819 [==============================] - 231s 282ms/step - loss: 14.3308 -
mae: 2.9110 - val_loss: 9.6137 - val_mae: 2.4096
Epoch 8/20
819/819 [==============================] - 230s 281ms/step - loss: 13.8246 -
mae: 2.8606 - val_loss: 9.2456 - val_mae: 2.3602
Epoch 9/20
819/819 [==============================] - 230s 281ms/step - loss: 13.4897 -
mae: 2.8326 - val_loss: 9.2555 - val_mae: 2.3685
Epoch 10/20
819/819 [==============================] - 230s 281ms/step - loss: 13.1145 -
mae: 2.7943 - val_loss: 9.5056 - val_mae: 2.3939
Epoch 11/20
819/819 [==============================] - 231s 282ms/step - loss: 12.8334 -
mae: 2.7633 - val_loss: 9.2116 - val_mae: 2.3648
Epoch 12/20
819/819 [==============================] - 231s 282ms/step - loss: 12.6194 -
mae: 2.7443 - val_loss: 10.0260 - val_mae: 2.4624
Epoch 13/20
819/819 [==============================] - 229s 280ms/step - loss: 12.4551 -
mae: 2.7220 - val_loss: 9.5028 - val_mae: 2.4011
Epoch 14/20
819/819 [==============================] - 231s 282ms/step - loss: 12.2959 -
mae: 2.7023 - val_loss: 9.9008 - val_mae: 2.4460
Epoch 15/20
819/819 [==============================] - 231s 281ms/step - loss: 12.0898 -
mae: 2.6824 - val_loss: 9.7053 - val_mae: 2.4239
Epoch 16/20
819/819 [==============================] - 230s 281ms/step - loss: 11.9877 -
mae: 2.6704 - val_loss: 9.6224 - val_mae: 2.4177
Epoch 17/20
819/819 [==============================] - 232s 283ms/step - loss: 11.8660 -
mae: 2.6573 - val_loss: 9.7481 - val_mae: 2.4415
Epoch 18/20
819/819 [==============================] - 231s 282ms/step - loss: 11.7395 -
mae: 2.6445 - val_loss: 9.9164 - val_mae: 2.4641
Epoch 19/20
819/819 [==============================] - 232s 283ms/step - loss: 11.6769 -
mae: 2.6351 - val_loss: 9.7391 - val_mae: 2.4404
Epoch 20/20
819/819 [==============================] - 233s 284ms/step - loss: 11.6319 -
mae: 2.6339 - val_loss: 9.9229 - val_mae: 2.4590
```

[8]:

```python
model = keras.models.load_model("stacked_lstm_16_dropout.keras")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
405/405 [==============================] - 19s 43ms/step - loss: 10.4438 - mae:
2.5474
Test MAE: 2.55
```

Question_3

November 5, 2023

0.1 Question_3

0.1.1 Import libraries and dataset

[1]:

```
from tensorflow import keras
from tensorflow.keras import layers
import os
import numpy as np
from matplotlib import pyplot as plt
```

[2]:

```
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))
```

```
['"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
```

0.1.2 Parsing the data

[3]:

```
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[:]
```

0.1.3 Number of samples in each set

[4]:

```python
num_train_samples = int(0.5 * len(raw_data))
num_val_samples = int(0.25 * len(raw_data))
num_test_samples = len(raw_data) - num_train_samples - num_val_samples
print("num_train_samples:", num_train_samples)
print("num_val_samples:", num_val_samples)
print("num_test_samples:", num_test_samples)
```

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

0.1.4 Normalize the data

[5]:

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

0.1.5 Datasets for training, validation, testing

[6]:

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

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

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

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

0.1.6 Build, train, evaluate the model

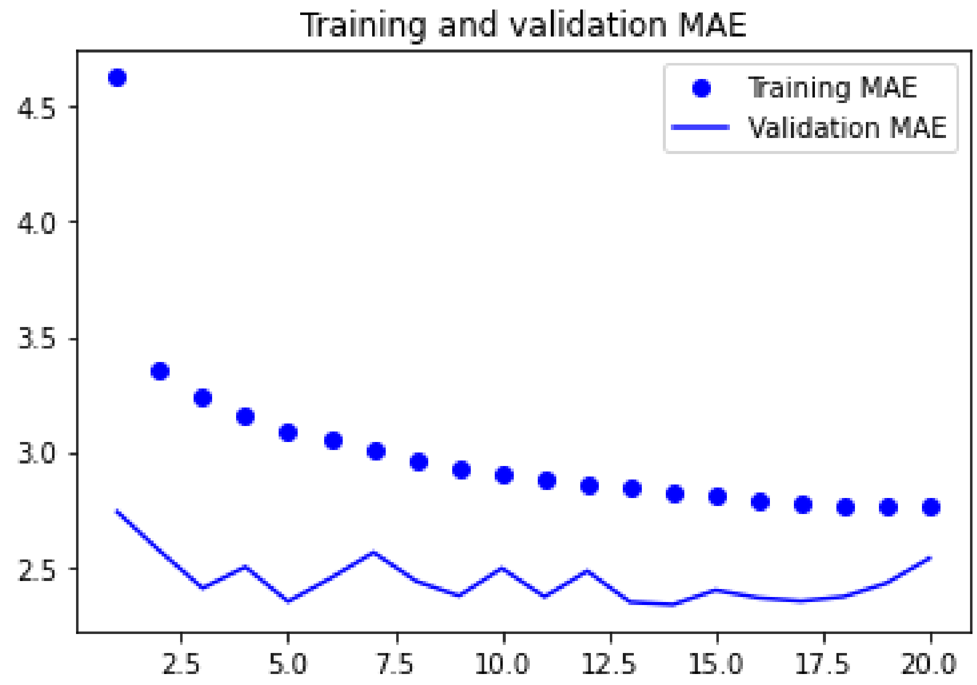
[7]:

```
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.GRU(16, 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("conv_gru_dropout.keras",
                                    save_best_only=True)
]
model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
history = model.fit(train_dataset,
                    epochs=20,
                    validation_data=val_dataset,
                    callbacks=callbacks)

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

```
Epoch 1/20
819/819 [==============================] - 84s 100ms/step - loss: 39.0054 - mae:
4.6298 - val_loss: 13.0200 - val_mae: 2.7386
Epoch 2/20
819/819 [==============================] - 78s 96ms/step - loss: 19.1489 - mae:
3.3585 - val_loss: 10.9797 - val_mae: 2.5671
Epoch 3/20
```

```
819/819 [==============================] - 78s 95ms/step - loss: 17.6833 - mae:
3.2369 - val_loss: 9.5651 - val_mae: 2.4064
Epoch 4/20
819/819 [==============================] - 78s 95ms/step - loss: 16.8181 - mae:
3.1615 - val_loss: 10.1004 - val_mae: 2.4985
Epoch 5/20
819/819 [==============================] - 78s 95ms/step - loss: 16.1238 - mae:
3.0919 - val_loss: 9.0914 - val_mae: 2.3486
Epoch 6/20
819/819 [==============================] - 77s 94ms/step - loss: 15.7046 - mae:
3.0534 - val_loss: 9.6751 - val_mae: 2.4506
Epoch 7/20
819/819 [==============================] - 78s 95ms/step - loss: 15.2249 - mae:
3.0060 - val_loss: 10.5263 - val_mae: 2.5613
Epoch 8/20
819/819 [==============================] - 78s 95ms/step - loss: 14.7750 - mae:
2.9646 - val_loss: 9.5405 - val_mae: 2.4364
Epoch 9/20
819/819 [==============================] - 78s 95ms/step - loss: 14.4110 - mae:
2.9289 - val_loss: 9.2051 - val_mae: 2.3732
Epoch 10/20
819/819 [==============================] - 78s 96ms/step - loss: 14.1473 - mae:
2.9009 - val_loss: 10.0338 - val_mae: 2.4927
Epoch 11/20
819/819 [==============================] - 76s 93ms/step - loss: 13.9149 - mae:
2.8747 - val_loss: 9.1119 - val_mae: 2.3692
Epoch 12/20
819/819 [==============================] - 77s 93ms/step - loss: 13.7225 - mae:
2.8589 - val_loss: 10.0053 - val_mae: 2.4806
Epoch 13/20
819/819 [==============================] - 79s 96ms/step - loss: 13.6401 - mae:
2.8464 - val_loss: 9.0678 - val_mae: 2.3463
Epoch 14/20
819/819 [==============================] - 78s 96ms/step - loss: 13.3996 - mae:
2.8225 - val_loss: 8.9307 - val_mae: 2.3349
Epoch 15/20
819/819 [==============================] - 78s 95ms/step - loss: 13.2730 - mae:
2.8079 - val_loss: 9.3562 - val_mae: 2.3981
Epoch 16/20
819/819 [==============================] - 79s 96ms/step - loss: 13.1057 - mae:
2.7893 - val_loss: 9.1533 - val_mae: 2.3645
Epoch 17/20
819/819 [==============================] - 78s 96ms/step - loss: 13.0180 - mae:
2.7804 - val_loss: 9.0478 - val_mae: 2.3503
Epoch 18/20
819/819 [==============================] - 77s 94ms/step - loss: 12.9366 - mae:
2.7691 - val_loss: 9.1689 - val_mae: 2.3695
Epoch 19/20
```

```
819/819 [==============================] - 79s 96ms/step - loss: 12.8808 - mae:
2.7615 - val_loss: 9.6645 - val_mae: 2.4283
Epoch 20/20
819/819 [==============================] - 79s 96ms/step - loss: 12.8380 - mae:
2.7596 - val_loss: 10.3409 - val_mae: 2.5354
```



[8]:

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

```
405/405 [==============================] - 12s 28ms/step - loss: 10.3664 - mae:
2.5418
Test MAE: 2.54
```

[ ]:

Question_3_2

November 5, 2023

0.1 Question_3_2

0.1.1 Import libraries and dataset

[1]:

```
from tensorflow import keras
from tensorflow.keras import layers
import os
import numpy as np
from matplotlib import pyplot as plt
```

[2]:

```
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))
```

```
['"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
```

0.1.2 Parsing the data

[3]:

```
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[:]
```

0.1.3 Number of samples in each set

[4]:

```
num_train_samples = int(0.5 * len(raw_data))
num_val_samples = int(0.25 * len(raw_data))
num_test_samples = len(raw_data) - num_train_samples - num_val_samples
print("num_train_samples:", num_train_samples)
print("num_val_samples:", num_val_samples)
print("num_test_samples:", num_test_samples)
```

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

0.1.4 Normalize the data

[5]:

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

0.1.5 Datasets for training, validation, testing

[6]:

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

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

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

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

0.1.6 Build, train, evaluate the model

[ ]:

```python
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.LSTM(16, 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("conv_lstm_dropout.keras",
                                    save_best_only=True)
]
model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
history = model.fit(train_dataset,
                    epochs=20,
                    validation_data=val_dataset,
                    callbacks=callbacks)
```

```
Epoch 1/20
819/819 [==============================] - 90s 107ms/step - loss: 43.7603 - mae:
4.9117 - val_loss: 13.2882 - val_mae: 2.7718
Epoch 2/20
819/819 [==============================] - 85s 104ms/step - loss: 19.2459 - mae:
3.3663 - val_loss: 10.5141 - val_mae: 2.5216
Epoch 3/20
819/819 [==============================] - 83s 101ms/step - loss: 17.8453 - mae:
3.2471 - val_loss: 9.5341 - val_mae: 2.4055
Epoch 4/20
819/819 [==============================] - 86s 105ms/step - loss: 16.9112 - mae:
3.1629 - val_loss: 9.8786 - val_mae: 2.4547
Epoch 5/20
819/819 [==============================] - 83s 101ms/step - loss: 16.3067 - mae:
3.1029 - val_loss: 9.8445 - val_mae: 2.4591
Epoch 6/20
819/819 [==============================] - 85s 103ms/step - loss: 15.7723 - mae:
3.0545 - val_loss: 9.2341 - val_mae: 2.3695
```

```
Epoch 7/20
819/819 [==============================] - 84s 103ms/step - loss: 15.2910 - mae:
3.0099 - val_loss: 9.5489 - val_mae: 2.4187
Epoch 8/20
819/819 [==============================] - 84s 102ms/step - loss: 15.0274 - mae:
2.9788 - val_loss: 9.4913 - val_mae: 2.4108
Epoch 9/20
819/819 [==============================] - 86s 105ms/step - loss: 14.5708 - mae:
2.9374 - val_loss: 9.7544 - val_mae: 2.4355
Epoch 10/20
819/819 [==============================] - 86s 104ms/step - loss: 14.2676 - mae:
2.9087 - val_loss: 9.8126 - val_mae: 2.4409
Epoch 11/20
819/819 [==============================] - 85s 104ms/step - loss: 14.0370 - mae:
2.8824 - val_loss: 9.7626 - val_mae: 2.4313
Epoch 12/20
819/819 [==============================] - 85s 104ms/step - loss: 13.8012 - mae:
2.8554 - val_loss: 10.8043 - val_mae: 2.5677
Epoch 13/20
819/819 [==============================] - 85s 104ms/step - loss: 13.7048 - mae:
2.8448 - val_loss: 9.9166 - val_mae: 2.4567
Epoch 14/20
819/819 [==============================] - 86s 105ms/step - loss: 13.5504 - mae:
2.8239 - val_loss: 9.7539 - val_mae: 2.4447
Epoch 15/20
819/819 [==============================] - 85s 103ms/step - loss: 13.4505 - mae:
2.8161 - val_loss: 10.3682 - val_mae: 2.5120
Epoch 16/20
309/819 [==========>…] - ETA: 45s - loss: 13.1636 - mae:
2.7888
```

[8]:

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

```
405/405 [==============================] - 13s 29ms/step - loss: 10.9326 - mae:
2.5945
Test MAE: 2.59
```

Question_3_3

November 5, 2023

0.1 Question_3_3

0.1.1 Import libraries and dataset

[1]:

```python
from tensorflow import keras
from tensorflow.keras import layers
import os
import numpy as np
from matplotlib import pyplot as plt
```

[2]:

```python
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))
```

```
['"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
```

0.1.2 Parsing the data

[3]:

```python
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[:]
```

0.1.3 Number of samples in each set

[4]:

```
num_train_samples = int(0.5 * len(raw_data))
num_val_samples = int(0.25 * len(raw_data))
num_test_samples = len(raw_data) - num_train_samples - num_val_samples
print("num_train_samples:", num_train_samples)
print("num_val_samples:", num_val_samples)
print("num_test_samples:", num_test_samples)
```

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

0.1.4 Normalize the data

[5]:

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

0.1.5 Datasets for training, validation, testing

[6]:

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

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

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

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

0.1.6 Build, train, evaluate the model

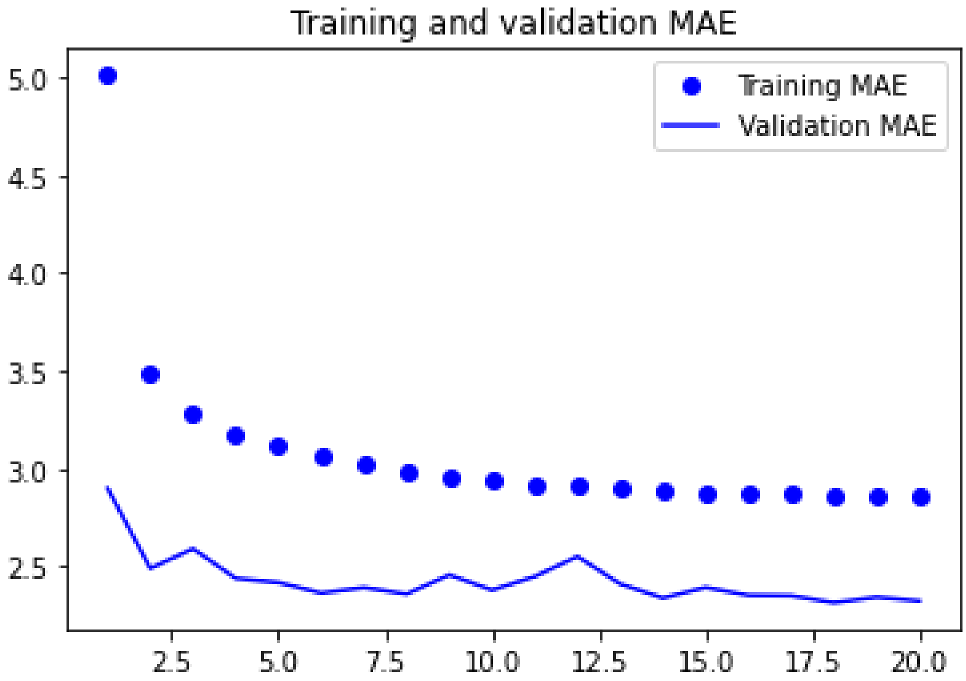
[7]:

```
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.SimpleRNN(16, 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("conv_rnn_dropout.keras",
                                    save_best_only=True)
]
model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
history = model.fit(train_dataset,
                    epochs=20,
                    validation_data=val_dataset,
                    callbacks=callbacks)

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

```
Epoch 1/20
819/819 [==============================] - 57s 67ms/step - loss: 44.4517 - mae:
5.0105 - val_loss: 14.6364 - val_mae: 2.8994
Epoch 2/20
819/819 [==============================] - 58s 70ms/step - loss: 20.5249 - mae:
3.4866 - val_loss: 10.4539 - val_mae: 2.4899
Epoch 3/20
```

```
819/819 [==============================] - 57s 69ms/step - loss: 18.1242 - mae:
3.2760 - val_loss: 10.9020 - val_mae: 2.5902
Epoch 4/20
819/819 [==============================] - 56s 68ms/step - loss: 17.0687 - mae:
3.1793 - val_loss: 9.8307 - val_mae: 2.4393
Epoch 5/20
819/819 [==============================] - 57s 69ms/step - loss: 16.3656 - mae:
3.1193 - val_loss: 9.6241 - val_mae: 2.4184
Epoch 6/20
819/819 [==============================] - 55s 67ms/step - loss: 15.8317 - mae:
3.0636 - val_loss: 9.2071 - val_mae: 2.3649
Epoch 7/20
819/819 [==============================] - 57s 69ms/step - loss: 15.3506 - mae:
3.0214 - val_loss: 9.3696 - val_mae: 2.3925
Epoch 8/20
819/819 [==============================] - 55s 67ms/step - loss: 14.9938 - mae:
2.9863 - val_loss: 9.1892 - val_mae: 2.3598
Epoch 9/20
819/819 [==============================] - 53s 65ms/step - loss: 14.7353 - mae:
2.9615 - val_loss: 9.8199 - val_mae: 2.4573
Epoch 10/20
819/819 [==============================] - 54s 66ms/step - loss: 14.5635 - mae:
2.9478 - val_loss: 9.3496 - val_mae: 2.3799
Epoch 11/20
819/819 [==============================] - 56s 69ms/step - loss: 14.2773 - mae:
2.9194 - val_loss: 9.9131 - val_mae: 2.4489
Epoch 12/20
819/819 [==============================] - 55s 67ms/step - loss: 14.2115 - mae:
2.9086 - val_loss: 10.5592 - val_mae: 2.5506
Epoch 13/20
819/819 [==============================] - 57s 69ms/step - loss: 14.0573 - mae:
2.8951 - val_loss: 9.5445 - val_mae: 2.4110
Epoch 14/20
819/819 [==============================] - 55s 68ms/step - loss: 13.9281 - mae:
2.8856 - val_loss: 9.0515 - val_mae: 2.3393
Epoch 15/20
819/819 [==============================] - 56s 69ms/step - loss: 13.8534 - mae:
2.8799 - val_loss: 9.4057 - val_mae: 2.3933
Epoch 16/20
819/819 [==============================] - 56s 68ms/step - loss: 13.8114 - mae:
2.8717 - val_loss: 9.1666 - val_mae: 2.3530
Epoch 17/20
819/819 [==============================] - 56s 68ms/step - loss: 13.8288 - mae:
2.8733 - val_loss: 9.1058 - val_mae: 2.3511
Epoch 18/20
819/819 [==============================] - 53s 65ms/step - loss: 13.7229 - mae:
2.8624 - val_loss: 8.8840 - val_mae: 2.3156
Epoch 19/20
```

```
819/819 [==============================] - 54s 66ms/step - loss: 13.7029 - mae:
2.8657 - val_loss: 9.0502 - val_mae: 2.3435
Epoch 20/20
819/819 [==============================] - 58s 71ms/step - loss: 13.6789 - mae:
2.8608 - val_loss: 8.9159 - val_mae: 2.3248
```



[8]:

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

```
405/405 [==============================] - 12s 28ms/step - loss: 10.5032 - mae:
2.5280
Test MAE: 2.53
```