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

```
--2024-11-03 18:45:23--  https://s3.amazonaws.com/keras-datasets/jena_climate_2009_2016.csv.zip
Resolving s3.amazonaws.com (s3.amazonaws.com)... 52.216.34.208, 16.182.43.56, 52.216.62.200, ...
Connecting to s3.amazonaws.com (s3.amazonaws.com)|52.216.34.208|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 13565642 (13M) [application/zip]
Saving to: 'jena_climate_2009_2016.csv.zip.1'

jena_climate_2009_2 100%[===================>]  12.94M  20.8MB/s    in 0.6s

2024-11-03 18:45:23 (20.8 MB/s) - 'jena_climate_2009_2016.csv.zip.1' saved [13565642/13565642]

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

```
import os
file_path = os.path.join("jena_climate_2009_2016.csv")


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


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

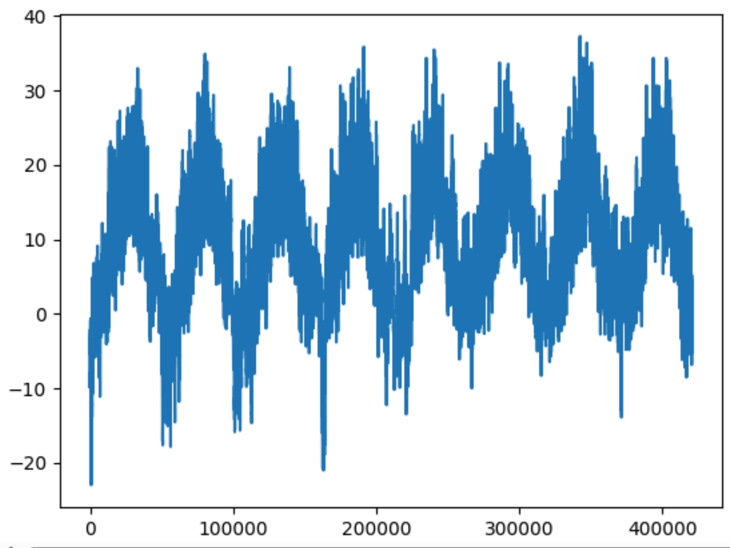
```
['"Date Time"', '"p (mbar)"', '"T (degC)"', '"Tpot (K)"', '"Tdew (degC)"', '"rh (%)"', '"VPmax (mbar)"', '"VPact (mbar)"', '"VPdef (mbar
420451
```

```
with open(file_path) as f:
    data = f.read()


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

```
['"Date Time"', '"p (mbar)"', '"T (degC)"', '"Tpot (K)"', '"Tdew (degC)"', '"rh (%)"', '"VPmax (mbar)"', '"VPact (mbar)"', '"VPdef (mbar
420451
```

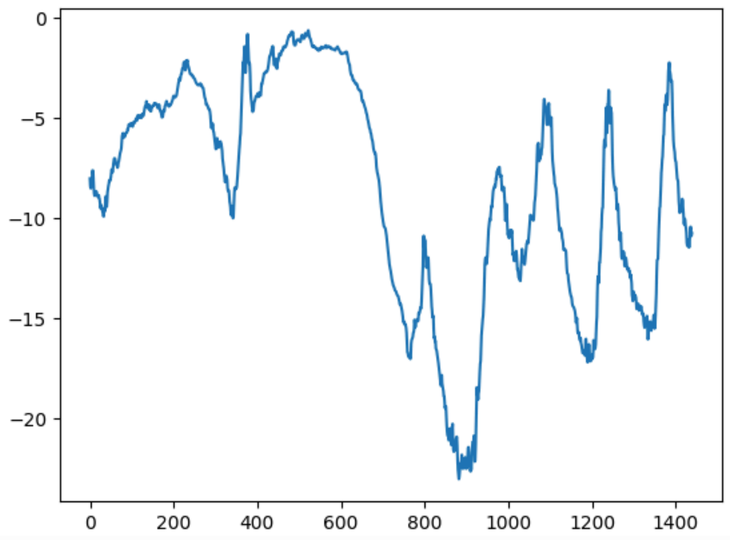
```
import numpy as np
temp_data = np.zeros((len(lines),))
raw_data = np.zeros((len(lines), len(header) - 1))
for i, line in enumerate(lines):
    values = [float(x) for x in line.split(",")[1:]]
    temp_data[i] = values[1]
    raw_data[i, :] = values[:]
import numpy as np
temp_data = np.zeros((len(lines),))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


from matplotlib import pyplot as plt
plt.plot(range(len(temp_data)), temp_data)
#Plotting the first 10 days of the temperature timeseries
```

```
[<matplotlib.lines.Line2D at 0x7ff84abf3160>]
```



```
plt.plot(range(1440), temp_data[:1440])
#Computing the number of samples we'll use for each data split
```

```
[<matplotlib.lines.Line2D at 0x7ff8482e6380>]
```



```
train_sample_count = int(0.5 * len(raw_data))
val_sample_count = int(0.25 * len(raw_data))
test_sample_count = len(raw_data) - train_sample_count - val_sample_count
print("train_sample_count:", train_sample_count)
print("val_sample_count:", val_sample_count)
print("test_sample_count:", test_sample_count)
#Preparing the data
#Normalizing the data
```

```
train_sample_count: 210225
val_sample_count: 105112
test_sample_count: 105114
```

```
mean = raw_data[:train_sample_count].mean(axis=0)
raw_data -= mean
std = raw_data[:train_sample_count].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]))
#Instantiating datasets for training, validation, and testing
```

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

```
data_sampling_rate = 6
sequence_length = 120
prediction_delay = data_sampling_rate * (sequence_length + 24 - 1)
training_batch_size = 256


train_dataset = keras.utils.timeseries_dataset_from_array(
    raw_data[:-prediction_delay],
    targets=temp_data[prediction_delay:],
    sampling_rate=data_sampling_rate,
    sequence_length=sequence_length,
    shuffle=True,
    batch_size=training_batch_size,
    start_index=0,
    end_index=train_sample_count)


val_dataset = keras.utils.timeseries_dataset_from_array(
    raw_data[:-prediction_delay],
    targets=temp_data[prediction_delay:],
    sampling_rate=data_sampling_rate,
    sequence_length=sequence_length,
    shuffle=True,
    batch_size=training_batch_size,
    start_index=train_sample_count,
    end_index=train_sample_count + val_sample_count)


test_dataset = keras.utils.timeseries_dataset_from_array(
    raw_data[:-prediction_delay],
    targets=temp_data[prediction_delay:],
    sampling_rate=data_sampling_rate,
    sequence_length=sequence_length,
    shuffle=True,
    batch_size=training_batch_size,
    start_index=train_sample_count + val_sample_count)
#Inspecting the output of one of our datasets


for samples, targets in train_dataset:
    print("samples shape:", samples.shape)
    print("targets shape:", targets.shape)
    break
#A common-sense, non-machine-learning baseline
#Computing the common-sense baseline MAE
```

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

```
def evaluate_naive_method(dataset):
    cumulative_abs_error = 0.
    total_samples = 0
    for samples, targets in dataset:
        preds = samples[:, -1, 1] * std[1] + mean[1]
        cumulative_abs_error += np.sum(np.abs(preds - targets))
        total_samples += samples.shape[0]
    return cumulative_abs_error / total_samples


print(f"Validation MAE: {evaluate_naive_method(val_dataset):.2f}")
print(f"Test MAE: {evaluate_naive_method(test_dataset):.2f}")
#Let's try a basic machine-learning model
#Training and evaluating a densely connected model
```

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

```
!pip install tensorflow==2.12
```

```
Collecting tensorflow==2.12
  Downloading tensorflow-2.12.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (3.4 kB)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (1.6.3)
Requirement already satisfied: flatbuffers>=2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (24.3.25)
Collecting gast<=0.4.0,>=0.2.1 (from tensorflow==2.12)
  Downloading gast-0.4.0-py3-none-any.whl.metadata (1.1 kB)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (0.2.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (1.64.1)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (3.12.1)
Requirement already satisfied: jax>=0.3.15 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (0.4.33)
Collecting keras<2.13,>=2.12.0 (from tensorflow==2.12)
  Downloading keras-2.12.0-py2.py3-none-any.whl.metadata (1.4 kB)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (18.1.1)
Collecting numpy<1.24,>=1.22 (from tensorflow==2.12)
  Downloading numpy-1.23.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (2.3 kB)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (24.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/python
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (75.1.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (1.16.0)
Collecting tensorboard<2.13,>=2.12 (from tensorflow==2.12)
  Downloading tensorboard-2.12.3-py3-none-any.whl.metadata (1.8 kB)
Collecting tensorflow-estimator<2.13,>=2.12.0 (from tensorflow==2.12)
  Downloading tensorflow_estimator-2.12.0-py2.py3-none-any.whl.metadata (1.3 kB)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (4.12.2)
Collecting wrapt<1.15,>=1.11.0 (from tensorflow==2.12)
  Downloading wrapt-1.14.1-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (6
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.12) (
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow==2.12)
Requirement already satisfied: jaxlib<=0.4.33,>=0.4.33 in /usr/local/lib/python3.10/dist-packages (from jax>=0.3.15->tensorflow==2.12) (
Requirement already satisfied: ml-dtypes>=0.2.0 in /usr/local/lib/python3.10/dist-packages (from jax>=0.3.15->tensorflow==2.12) (0.4.1)
INFO: pip is looking at multiple versions of jax to determine which version is compatible with other requirements. This could take a whi
Collecting jax>=0.3.15 (from tensorflow==2.12)
  Downloading jax-0.4.35-py3-none-any.whl.metadata (22 kB)
Collecting jaxlib<=0.4.35,>=0.4.34 (from jax>=0.3.15->tensorflow==2.12)
  Downloading jaxlib-0.4.35-cp310-cp310-manylinux2014_x86_64.whl.metadata (983 bytes)
Collecting jax>=0.3.15 (from tensorflow==2.12)
  Downloading jax-0.4.34-py3-none-any.whl.metadata (22 kB)
Collecting jaxlib<=0.4.34,>=0.4.34 (from jax>=0.3.15->tensorflow==2.12)
  Downloading jaxlib-0.4.34-cp310-cp310-manylinux2014_x86_64.whl.metadata (983 bytes)
Collecting jax>=0.3.15 (from tensorflow==2.12)
  Downloading jax-0.4.31-py3-none-any.whl.metadata (22 kB)
Collecting jaxlib<=0.4.31,>=0.4.30 (from jax>=0.3.15->tensorflow==2.12)
  Downloading jaxlib-0.4.31-cp310-cp310-manylinux2014_x86_64.whl.metadata (983 bytes)
Collecting jax>=0.3.15 (from tensorflow==2.12)
  Downloading jax-0.4.30-py3-none-any.whl.metadata (22 kB)
Collecting jaxlib<=0.4.30,>=0.4.27 (from jax>=0.3.15->tensorflow==2.12)
  Downloading jaxlib-0.4.30-cp310-cp310-manylinux2014_x86_64.whl.metadata (1.0 kB)
Requirement already satisfied: scipy>=1.9 in /usr/local/lib/python3.10/dist-packages (from jax>=0.3.15->tensorflow==2.12) (1.13.1)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>=2.12->tensorflo
Collecting google-auth-oauthlib<1.1,>=0.5 (from tensorboard<2.13,>=2.12->tensorflow==2.12)
  Downloading google_auth_oauthlib-1.0.0-py2.py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>=2.12->tensorflow==2.1
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>=2.12->tensorflow=
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>=2.12->tensorflow==2.1
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboar
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.13,>=
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthlib<1.1,>=0.5-
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboar
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.13,>=2.1
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.13
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.13
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard<2.13,>=2.
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-auth<
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oa
Downloading tensorflow-2.12.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (585.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 585.9/585.9 MB 3.2 MB/s eta 0:00:00
Downloading gast-0.4.0-py3-none-any.whl (9.8 kB)
Downloading jax-0.4.30-py3-none-any.whl (2.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.0/2.0 MB 82.6 MB/s eta 0:00:00
Downloading keras-2.12.0-py2.py3-none-any.whl (1.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 76.6 MB/s eta 0:00:00
Downloading numpy-1.23.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.1/17.1 MB 91.7 MB/s eta 0:00:00
Downloading tensorboard-2.12.3-py3-none-any.whl (5.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.6/5.6 MB 67.7 MB/s eta 0:00:00
Downloading tensorflow_estimator-2.12.0-py2.py3-none-any.whl (440 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 440.7/440.7 kB 30.7 MB/s eta 0:00:00
Downloading wrapt-1.14.1-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (77 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 77.9/77.9 kB 6.9 MB/s eta 0:00:00
```

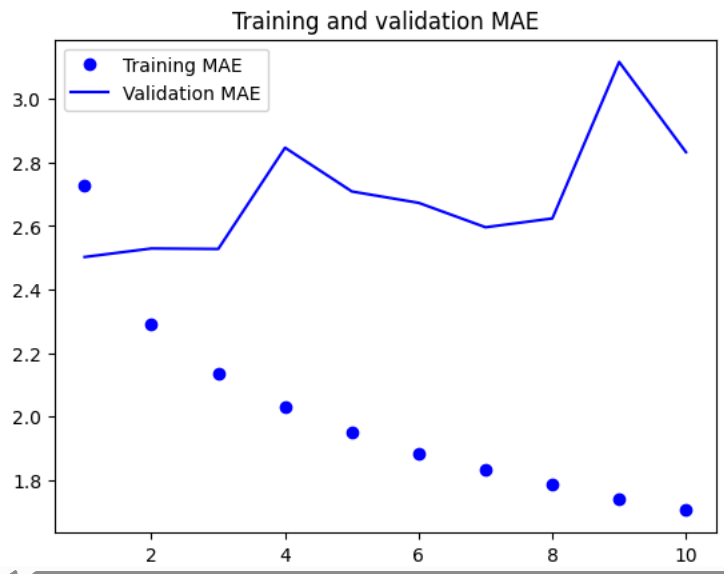
```
Downloading google_auth_oauthlib-1.0.0-py2.py3-none-any.whl (18 kB)
Downloading jaxlib-0.4.30-cp310-cp310-manylinux2014_x86_64.whl (79.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.6/79.6 MB 9.1 MB/s eta 0:00:00
Installing collected packages: wrapt, tensorflow-estimator, numpy, keras, gast, jaxlib, google-auth-oauthlib, tensorboard, jax, tensorfl
  Attempting uninstall: wrapt
    Found existing installation: wrapt 1.16.0
    Uninstalling wrapt-1.16.0:
      Successfully uninstalled wrapt-1.16.0
  Attempting uninstall: numpy
    Found existing installation: numpy 1.26.4
    Uninstalling numpy-1.26.4:
      Successfully uninstalled numpy-1.26.4
  Attempting uninstall: keras
    Found existing installation: keras 3.4.1
    Uninstalling keras-3.4.1:
      Successfully uninstalled keras-3.4.1
  Attempting uninstall: gast
    Found existing installation: gast 0.6.0
    Uninstalling gast-0.6.0:
      Successfully uninstalled gast-0.6.0
  Attempting uninstall: jaxlib
    Found existing installation: jaxlib 0.4.33
    Uninstalling jaxlib-0.4.33:
      Successfully uninstalled jaxlib-0.4.33
  Attempting uninstall: google-auth-oauthlib
    Found existing installation: google-auth-oauthlib 1.2.1
    Uninstalling google-auth-oauthlib-1.2.1:
      Successfully uninstalled google-auth-oauthlib-1.2.1
  Attempting uninstall: tensorboard
    Found existing installation: tensorboard 2.17.0
    Uninstalling tensorboard-2.17.0:
      Successfully uninstalled tensorboard-2.17.0
  Attempting uninstall: jax
    Found existing installation: jax 0.4.33
    Uninstalling jax-0.4.33:
      Successfully uninstalled jax-0.4.33
  Attempting uninstall: tensorflow
    Found existing installation: tensorflow 2.17.0
    Uninstalling tensorflow-2.17.0:
      Successfully uninstalled tensorflow-2.17.0
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source
albucore 0.0.19 requires numpy>=1.24.4, but you have numpy 1.23.5 which is incompatible.
albumentations 1.4.20 requires numpy>=1.24.4, but you have numpy 1.23.5 which is incompatible.
bigframes 1.25.0 requires numpy>=1.24.0, but you have numpy 1.23.5 which is incompatible.
chex 0.1.87 requires numpy>=1.24.1, but you have numpy 1.23.5 which is incompatible.
tf-keras 2.17.0 requires tensorflow<2.18,>=2.17, but you have tensorflow 2.12.0 which is incompatible.
xarray 2024.10.0 requires numpy>=1.24, but you have numpy 1.23.5 which is incompatible.
Successfully installed gast-0.4.0 google-auth-oauthlib-1.0.0 jax-0.4.30 jaxlib-0.4.30 keras-2.12.0 numpy-1.23.5 tensorboard-2.12.3 tensc
WARNING: The following packages were previously imported in this runtime:
  [gast,jax,jaxlib,keras,numpy,tensorflow,wrapt]
You must restart the runtime in order to use newly installed versions.
```

RESTART SESSION

```
from tensorflow import keras
from tensorflow.keras import layers

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

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

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

```
Epoch 1/10
819/819 [==============================] - 43s 51ms/step - loss: 12.4232 - mae: 2.7283 - val_loss: 10.0694 - val_mae: 2.5025
Epoch 2/10
819/819 [==============================] - 42s 51ms/step - loss: 8.4752 - mae: 2.2887 - val_loss: 10.2791 - val_mae: 2.5293
Epoch 3/10
819/819 [==============================] - 41s 50ms/step - loss: 7.3807 - mae: 2.1333 - val_loss: 10.2993 - val_mae: 2.5278
Epoch 4/10
819/819 [==============================] - 41s 50ms/step - loss: 6.6952 - mae: 2.0315 - val_loss: 12.7769 - val_mae: 2.8469
Epoch 5/10
819/819 [==============================] - 41s 49ms/step - loss: 6.1590 - mae: 1.9502 - val_loss: 11.6689 - val_mae: 2.7089
Epoch 6/10
819/819 [==============================] - 48s 59ms/step - loss: 5.7382 - mae: 1.8846 - val_loss: 11.3996 - val_mae: 2.6731
Epoch 7/10
819/819 [==============================] - 48s 59ms/step - loss: 5.4135 - mae: 1.8309 - val_loss: 10.9152 - val_mae: 2.5962
Epoch 8/10
819/819 [==============================] - 40s 49ms/step - loss: 5.1346 - mae: 1.7859 - val_loss: 11.1484 - val_mae: 2.6239
Epoch 9/10
819/819 [==============================] - 40s 49ms/step - loss: 4.8804 - mae: 1.7417 - val_loss: 15.4525 - val_mae: 3.1171
Epoch 10/10
819/819 [==============================] - 40s 48ms/step - loss: 4.6580 - mae: 1.7042 - val_loss: 12.9074 - val_mae: 2.8329
405/405 [==============================] - 13s 31ms/step - loss: 11.0748 - mae: 2.6276
Test MAE: 2.63
```

```
import matplotlib.pyplot as plt
loss = history.history["mae"]
val_loss = history.history["val_mae"]
epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, "bo", label="Training MAE")
plt.plot(epochs, val_loss, "b", label="Validation MAE")
plt.title("Training and validation MAE")
plt.legend()
plt.show()
#Let's try a 1D convolutional model
inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.Conv1D(8, 24, activation="relu")(inputs)
x = layers.MaxPooling1D(2)(x)
x = layers.Conv1D(8, 12, activation="relu")(x)
x = layers.MaxPooling1D(2)(x)
x = layers.Conv1D(8, 6, activation="relu")(x)
x = layers.GlobalAveragePooling1D()(x)
outputs = layers.Dense(1)(x)
model = keras.Model(inputs, outputs)
```



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

```
Epoch 1/10
819/819 [==============================] - 69s 83ms/step - loss: 23.6207 - mae: 3.7749 - val_loss: 15.7902 - val_mae: 3.1011
Epoch 2/10
819/819 [==============================] - 73s 89ms/step - loss: 15.6191 - mae: 3.1368 - val_loss: 15.5976 - val_mae: 3.1359
Epoch 3/10
819/819 [==============================] - 75s 91ms/step - loss: 14.4001 - mae: 3.0077 - val_loss: 13.8882 - val_mae: 2.9398
Epoch 4/10
819/819 [==============================] - 68s 83ms/step - loss: 13.5780 - mae: 2.9199 - val_loss: 15.1572 - val_mae: 3.1114
Epoch 5/10
819/819 [==============================] - 79s 96ms/step - loss: 12.9283 - mae: 2.8469 - val_loss: 16.0815 - val_mae: 3.1702
Epoch 6/10
819/819 [==============================] - 75s 92ms/step - loss: 12.4281 - mae: 2.7889 - val_loss: 13.8277 - val_mae: 2.9120
Epoch 7/10
819/819 [==============================] - 75s 91ms/step - loss: 12.0593 - mae: 2.7452 - val_loss: 13.9177 - val_mae: 2.9276
Epoch 8/10
819/819 [==============================] - 76s 92ms/step - loss: 11.6699 - mae: 2.6985 - val_loss: 14.3330 - val_mae: 2.9682
Epoch 9/10
819/819 [==============================] - 74s 90ms/step - loss: 11.3696 - mae: 2.6618 - val_loss: 15.7933 - val_mae: 3.1141
Epoch 10/10
819/819 [==============================] - 79s 97ms/step - loss: 11.0809 - mae: 2.6272 - val_loss: 14.5763 - val_mae: 3.0114
```

```
model = keras.models.load_model("jena_conv.keras")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
#A first recurrent baseline
#A simple LSTM-based model
```

```
405/405 [==============================] - 16s 38ms/step - loss: 15.2564 - mae: 3.0941
Test MAE: 3.09
```

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

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

```
Epoch 1/10
819/819 [==============================] - 86s 102ms/step - loss: 38.7420 - mae: 4.5221 - val_loss: 12.1325 - val_mae: 2.6660
Epoch 2/10
819/819 [==============================] - 85s 103ms/step - loss: 10.8865 - mae: 2.5631 - val_loss: 10.3570 - val_mae: 2.4852
Epoch 3/10
819/819 [==============================] - 88s 107ms/step - loss: 9.6429 - mae: 2.4160 - val_loss: 9.0816 - val_mae: 2.3542
Epoch 4/10
819/819 [==============================] - 88s 107ms/step - loss: 9.1828 - mae: 2.3661 - val_loss: 9.2211 - val_mae: 2.3787
Epoch 5/10
819/819 [==============================] - 86s 104ms/step - loss: 8.8131 - mae: 2.3215 - val_loss: 9.6965 - val_mae: 2.4339
Epoch 6/10
819/819 [==============================] - 87s 105ms/step - loss: 8.5679 - mae: 2.2868 - val_loss: 9.6538 - val_mae: 2.4184
Epoch 7/10
819/819 [==============================] - 86s 105ms/step - loss: 8.4303 - mae: 2.2659 - val_loss: 9.7052 - val_mae: 2.4289
Epoch 8/10
819/819 [==============================] - 86s 104ms/step - loss: 8.2466 - mae: 2.2418 - val_loss: 9.6441 - val_mae: 2.4240
Epoch 9/10
819/819 [==============================] - 84s 103ms/step - loss: 8.1469 - mae: 2.2283 - val_loss: 9.6664 - val_mae: 2.4300
Epoch 10/10
819/819 [==============================] - 85s 103ms/step - loss: 8.0260 - mae: 2.2129 - val_loss: 9.6730 - val_mae: 2.4446
```

```
model = keras.models.load_model("jena_lstm.keras")
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
#Understanding recurrent neural networks
#NumPy implementation of a simple RNN
```

```
405/405 [==============================] - 21s 49ms/step - loss: 10.6253 - mae: 2.5553
Test MAE: 2.56
```

```
import numpy as np
timesteps = 100
input_features = 32
output_features = 64
inputs = np.random.random((timesteps, input_features))
state_t = np.zeros((output_features,))
W = np.random.random((output_features, input_features))
U = np.random.random((output_features, output_features))
b = np.random.random((output_features,))
successive_outputs = []
for input_t in inputs:
    output_t = np.tanh(np.dot(W, input_t) + np.dot(U, state_t) + b)
    successive_outputs.append(output_t)
    state_t = output_t
final_output_sequence = np.stack(successive_outputs, axis=0)
#A recurrent layer in Keras
#An RNN layer that can process sequences of any length
```

```
num_features = 14
inputs = keras.Input(shape=(None, num_features))
outputs = layers.SimpleRNN(16)(inputs)
#An RNN layer that returns only its last output step
```

```
num_features = 14
steps = 120
```