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

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

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

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

```
from matplotlib import pyplot as plt
plt.plot(range(len(temperature)), temperature)
plt.show()
plt.plot(range(1440), temperature[:1440])
```

Nhấp đúp (hoặc nhấn Enter) để chỉnh sửa

```
# define a number train, val, test dataset
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_train_samples

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

```
# Instantiating datasets for training, validation, and testing
from tensorflow import keras
```

```

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)

```

```
# Compute 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}")
```

```
  File "<ipython-input-8-a0adcb7b1cd5>", line 7
    preds = samples[:, -1, 1] * std[1] + mean[1]
        ^
IndentationError: expected an indented block
```

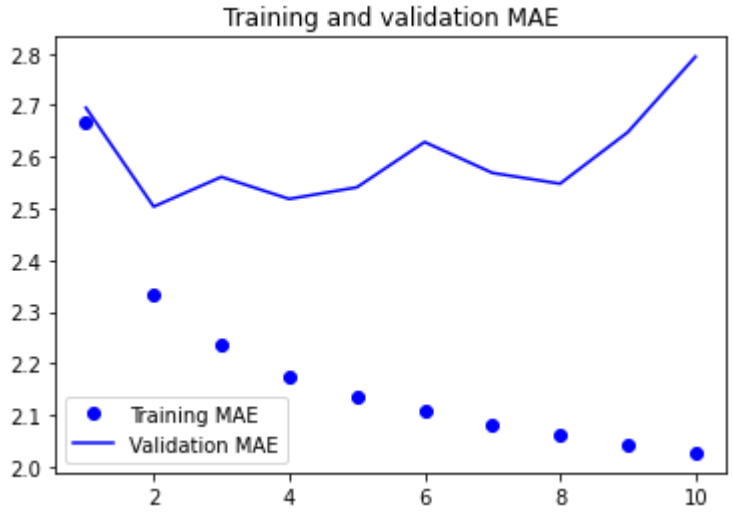
SEARCH STACK OVERFLOW

```
from tensorflow.python.module.module import valid_identifier
# 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.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}")
```

```
Epoch 1/10
819/819 [==============================] - 42s 48ms/step - loss: 11.8469 - mae: 2.6683 -
Epoch 2/10
819/819 [==============================] - 38s 46ms/step - loss: 8.7941 - mae: 2.3314 -
Epoch 3/10
819/819 [==============================] - 38s 46ms/step - loss: 8.1036 - mae: 2.2360 -
Epoch 4/10
819/819 [==============================] - 39s 47ms/step - loss: 7.6788 - mae: 2.1760 -
Epoch 5/10
819/819 [==============================] - 38s 47ms/step - loss: 7.3880 - mae: 2.1348 -
Epoch 6/10
819/819 [==============================] - 38s 46ms/step - loss: 7.2065 - mae: 2.1076 -
Epoch 7/10
819/819 [==============================] - 38s 46ms/step - loss: 7.0212 - mae: 2.0814 -
Epoch 8/10
819/819 [==============================] - 39s 48ms/step - loss: 6.8902 - mae: 2.0611 -
Epoch 9/10
819/819 [==============================] - 38s 46ms/step - loss: 6.7490 - mae: 2.0410 -
Epoch 10/10
819/819 [==============================] - 47s 57ms/step - loss: 6.6517 - mae: 2.0272 -
```

```
405/405 [==============================] - 13s 31ms/step - loss: 11.3980 - mae: 2.6664
Test MAE: 2.67
```

```
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()
```




```
# use Conv1D and maxpooling for layers
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)
```

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

```
Epoch 1/10
819/819 [==============================] - 47s 49ms/step - loss: 22.7086 - mae: 3.7513 -
Epoch 2/10
819/819 [==============================] - 40s 49ms/step - loss: 15.9681 - mae: 3.1748 -
Epoch 3/10
819/819 [==============================] - 40s 48ms/step - loss: 14.4462 - mae: 3.0107 -
Epoch 4/10
819/819 [==============================] - 40s 48ms/step - loss: 13.5934 - mae: 2.9186 -
Epoch 5/10
819/819 [==============================] - 40s 49ms/step - loss: 12.9501 - mae: 2.8427 -
Epoch 6/10
819/819 [==============================] - 41s 49ms/step - loss: 12.4787 - mae: 2.7922 -
Epoch 7/10
819/819 [==============================] - 40s 48ms/step - loss: 12.0760 - mae: 2.7473 -
Epoch 8/10
819/819 [==============================] - 40s 48ms/step - loss: 11.7926 - mae: 2.7141 -
Epoch 9/10
819/819 [==============================] - 42s 51ms/step - loss: 11.4932 - mae: 2.6799 -
Epoch 10/10
819/819 [==============================] - 42s 51ms/step - loss: 11.2759 - mae: 2.6559 -
405/405 [==============================] - 13s 32ms/step - loss: 16.2588 - mae: 3.1944
Test MAE: 3.19
```

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

```
# Use Long Sort Term Memory (LSTM)
inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.LSTM(16)(inputs)
outputs = layers.Dense(1)(x)
model = keras.Model(inputs, outputs)
callbacks = [
 keras.callbacks.ModelCheckpoint("jena_lstm.keras",
 save_best_only=True)
]
model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
history = model.fit(train_dataset,
 epochs=10,
 validation_data=val_dataset,
```

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

```

```
Epoch 1/10
819/819 [==============================] - 46s 53ms/step - loss: 40.0879 - mae: 4.6010 -
Epoch 2/10
819/819 [==============================] - 43s 52ms/step - loss: 11.0932 - mae: 2.5831 -
Epoch 3/10
819/819 [==============================] - 43s 52ms/step - loss: 9.9374 - mae: 2.4544 -
Epoch 4/10
819/819 [==============================] - 43s 52ms/step - loss: 9.5004 - mae: 2.3982 -
Epoch 5/10
819/819 [==============================] - 43s 52ms/step - loss: 9.2485 - mae: 2.3632 -
Epoch 6/10
819/819 [==============================] - 44s 53ms/step - loss: 9.0162 - mae: 2.3293 -
Epoch 7/10
819/819 [==============================] - 42s 52ms/step - loss: 8.8471 - mae: 2.3028 -
Epoch 8/10
819/819 [==============================] - 42s 51ms/step - loss: 8.6536 - mae: 2.2762 -
Epoch 9/10
819/819 [==============================] - 44s 53ms/step - loss: 8.5227 - mae: 2.2611 -
Epoch 10/10
819/819 [==============================] - 42s 51ms/step - loss: 8.4235 - mae: 2.2483 -
405/405 [==============================] - 14s 32ms/step - loss: 10.9081 - mae: 2.5726
Test MAE: 2.57
```