Tiny ML on Arduino

Gesture recognition tutorial

CSCE 5612

Setup Python Environment

The next cell sets up the dependencies in required for the notebook, run it.

```
# Setup environment
!apt-get -qq install xxd
!pip install pandas numpy matplotlib
!pip install tensorflow==2.0.0-rc1
```

```
Requirement already satisfied: pandas in
/usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: numpy in
/usr/local/lib/python3.11/dist-packages (1.26.4)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.11/dist-packages (from pandas) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in
/usr/local/lib/python3.11/dist-packages (from pandas) (2025.1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (4.56.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.1)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2-
>pandas) (1.17.0)
ERROR: Could not find a version that satisfies the requirement
```

```
tensorflow==2.0.0-rc1 (from versions: 2.12.0rc0, 2.12.0rc1, 2.12.0,
2.12.1, 2.13.0rc0, 2.13.0rc1, 2.13.0rc2, 2.13.0, 2.13.1, 2.14.0rc0,
2.14.0rc1, 2.14.0, 2.14.1, 2.15.0rc0, 2.15.0rc1, 2.15.0, 2.15.0.post1,
2.15.1, 2.16.0rc0, 2.16.1, 2.16.2, 2.17.0rc0, 2.17.0rc1, 2.17.0,
2.17.1, 2.18.0rc0, 2.18.0rc1, 2.18.0rc2, 2.18.0, 2.19.0rc0)
ERROR: No matching distribution found for tensorflow==2.0.0-rc1

from google.colab import drive
drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

Upload Data

1. Open the panel on the left side of Colab by clicking on the **>**
2. Select the files tab
3. Drag `punch.csv` and `flex.csv` files from your computer to the tab to upload them into colab.

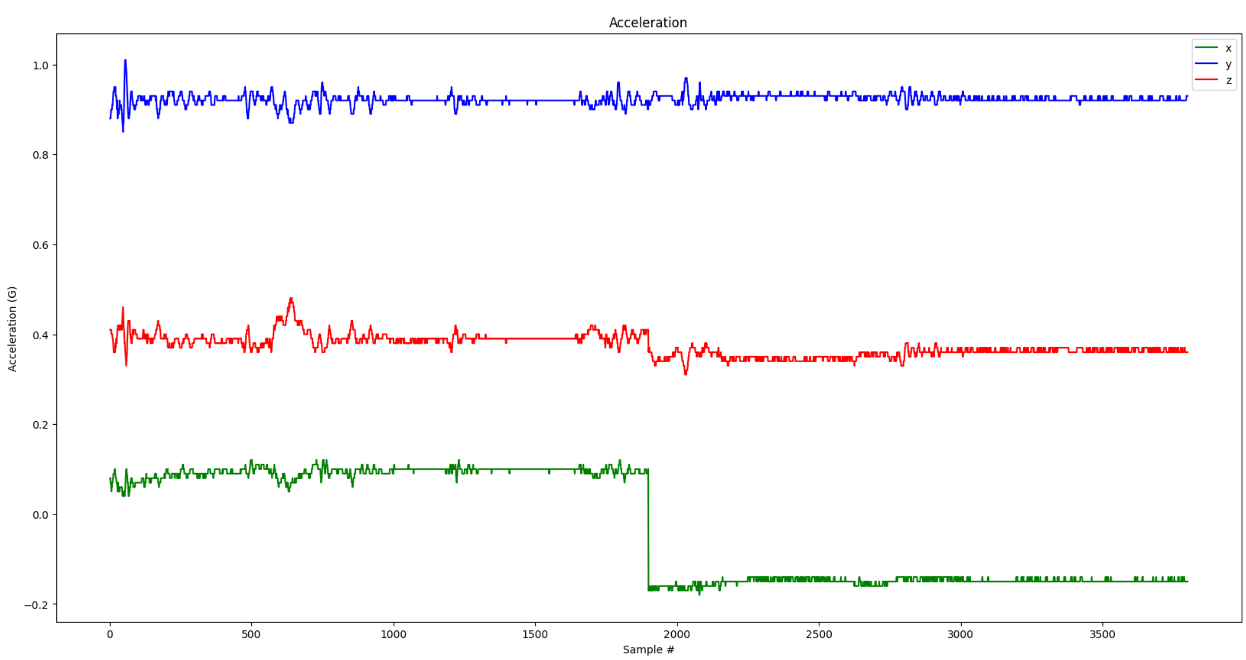
Graph Data (optional)

We'll graph the input files on two separate graphs, acceleration and gyroscope, as each data set has different units and scale.

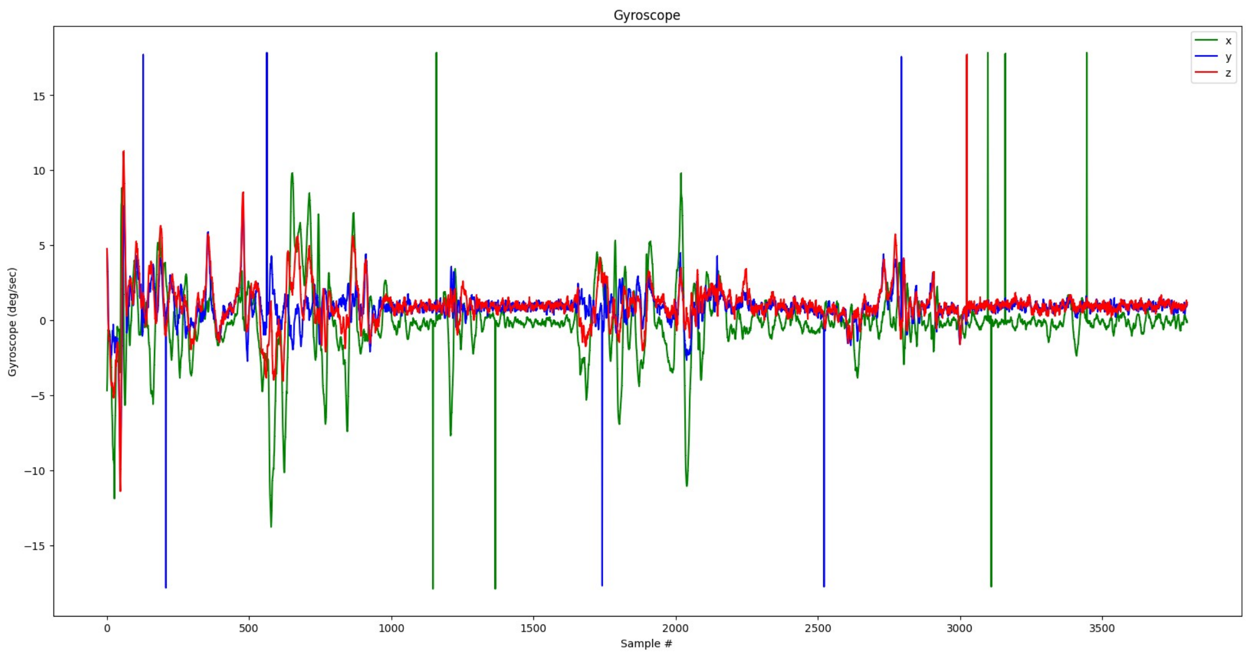
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

filename = "sitting.csv"

df = pd.read_csv("/content/" + filename)

index = range(1, len(df['Accel_x']) + 1)

plt.rcParams["figure.figsize"] = (20,10)

plt.plot(index, df['Accel_x'], 'g.', label='x', linestyle='solid',
marker=',')
plt.plot(index, df['Accel_y'], 'b.', label='y', linestyle='solid',
marker=',')
plt.plot(index, df['Accel_z'], 'r.', label='z', linestyle='solid',
marker=',')
plt.title("Acceleration")
plt.xlabel("Sample #")
plt.ylabel("Acceleration (G)")
```

```
plt.legend()
plt.show()

plt.plot(index, df['Gyr_x'], 'g.', label='x', linestyle='solid',
marker=',')
plt.plot(index, df['Gyr_y'], 'b.', label='y', linestyle='solid',
marker=',')
plt.plot(index, df['Gyr_z'], 'r.', label='z', linestyle='solid',
marker=',')
plt.title("Gyroscope")
plt.xlabel("Sample #")
plt.ylabel("Gyroscope (deg/sec)")
plt.legend()
plt.show()
```

```
<ipython-input-4-57698e345532>:13: UserWarning: marker is redundantly
defined by the 'marker' keyword argument and the fmt string "g." (->
marker='.'). The keyword argument will take precedence.
  plt.plot(index, df['Accel_x'], 'g.', label='x', linestyle='solid',
marker=',')
<ipython-input-4-57698e345532>:14: UserWarning: marker is redundantly
defined by the 'marker' keyword argument and the fmt string "b." (->
marker='.'). The keyword argument will take precedence.
  plt.plot(index, df['Accel_y'], 'b.', label='y', linestyle='solid',
marker=',')
<ipython-input-4-57698e345532>:15: UserWarning: marker is redundantly
defined by the 'marker' keyword argument and the fmt string "r." (->
marker='.'). The keyword argument will take precedence.
  plt.plot(index, df['Accel_z'], 'r.', label='z', linestyle='solid',
marker=',')
```



```
<ipython-input-4-57698e345532>:22: UserWarning: marker is redundantly
defined by the 'marker' keyword argument and the fmt string "g." (->
marker='.'). The keyword argument will take precedence.
  plt.plot(index, df['Gyr_x'], 'g.', label='x', linestyle='solid',
marker=',')
<ipython-input-4-57698e345532>:23: UserWarning: marker is redundantly
defined by the 'marker' keyword argument and the fmt string "b." (->
marker='.'). The keyword argument will take precedence.
  plt.plot(index, df['Gyr_y'], 'b.', label='y', linestyle='solid',
marker=',')
<ipython-input-4-57698e345532>:24: UserWarning: marker is redundantly
defined by the 'marker' keyword argument and the fmt string "r." (->
marker='.'). The keyword argument will take precedence.
  plt.plot(index, df['Gyr_z'], 'r.', label='z', linestyle='solid',
marker=',')
```



Train Neural Network

Parse and prepare the data

The next cell parses the csv files and transforms them to a format that will be used to train the fully connected neural network.

Update the `GESTURES` list with the gesture data you've collected in `.csv` format.

```
import matplotlib.pyplot as plt
import numpy as np
```

```python
import pandas as pd
import tensorflow as tf

print(f"TensorFlow version = {tf.__version__}\n")

# Set a fixed random seed value, for reproducibility, this will allow
us to get
# the same random numbers each time the notebook is run
SEED = 1337
np.random.seed(SEED)
tf.random.set_seed(SEED)

# the list of gestures that data is available for
GESTURES = [
    "sitting",
    "walking"
]

SAMPLES_PER_GESTURE = 500

NUM_GESTURES = len(GESTURES)

# create a one-hot encoded matrix that is used in the output
ONE_HOT_ENCODED_GESTURES = np.eye(NUM_GESTURES)

inputs = []
outputs = []

# read each csv file and push an input and output
for gesture_index in range(NUM_GESTURES):
  gesture = GESTURES[gesture_index]
  print(f"Processing index {gesture_index} for gesture '{gesture}'.")

  output = ONE_HOT_ENCODED_GESTURES[gesture_index]

  df = pd.read_csv("/content/" + gesture + ".csv")

  # calculate the number of gesture recordings in the file
  num_recordings = int(df.shape[0] / SAMPLES_PER_GESTURE)

  print(f"\tThere are {num_recordings} recordings of the {gesture}
gesture.")

  for i in range(num_recordings):
    tensor = []
    for j in range(SAMPLES_PER_GESTURE):
      index = i * SAMPLES_PER_GESTURE + j
      # normalize the input data, between 0 to 1:
      # - acceleration is between: -4 to +4
      # - gyroscope is between: -2000 to +2000
```

```
      tensor += [
          (df['Accel_x'][index] + 4) / 8,
          (df['Accel_y'][index] + 4) / 8,
          (df['Accel_z'][index] + 4) / 8,
          (df['Gyr_x'][index] + 2000) / 4000,
          (df['Gyr_y'][index] + 2000) / 4000,
          (df['Gyr_z'][index] + 2000) / 4000
      ]

    inputs.append(tensor)
    outputs.append(output)

# convert the list to numpy array
inputs = np.array(inputs)
outputs = np.array(outputs)

print("Data set parsing and preparation complete.")
```

```
TensorFlow version = 2.18.0

Processing index 0 for gesture 'sitting'.
     There are 7 recordings of the sitting gesture.
Processing index 1 for gesture 'walking'.
     There are 10 recordings of the walking gesture.
Data set parsing and preparation complete.
```

Randomize and split the input and output pairs for training

Randomly split input and output pairs into sets of data: 60% for training, 20% for validation, and 20% for testing.

- the training set is used to train the model
- the validation set is used to measure how well the model is performing during training
- the testing set is used to test the model after training

```
# Randomize the order of the inputs, so they can be evenly distributed
for training, testing, and validation
# https://stackoverflow.com/a/37710486/2020087
num_inputs = len(inputs)
randomize = np.arange(num_inputs)
np.random.shuffle(randomize)

# Swap the consecutive indexes (0, 1, 2, etc) with the randomized
indexes
inputs = inputs[randomize]
outputs = outputs[randomize]

# Split the recordings (group of samples) into three sets: training,
testing and validation
TRAIN_SPLIT = int(0.6 * num_inputs)
```

```
TEST_SPLIT = int(0.2 * num_inputs + TRAIN_SPLIT)

inputs_train, inputs_test, inputs_validate = np.split(inputs,
[TRAIN_SPLIT, TEST_SPLIT])
outputs_train, outputs_test, outputs_validate = np.split(outputs,
[TRAIN_SPLIT, TEST_SPLIT])

print("Data set randomization and splitting complete.")
```

```
Data set randomization and splitting complete.
```

Build & Train the Model

Build and train a TensorFlow model using the high-level Keras API.

```
# build the model and train it
model = tf.keras.Sequential()
model.add(tf.keras.layers.Dense(50, activation='relu')) # relu is used
for performance
model.add(tf.keras.layers.Dense(20, activation='relu'))
model.add(tf.keras.layers.Dense(NUM_GESTURES, activation='softmax')) #
softmax is used, because we only expect one gesture to occur per input
model.compile(optimizer='rmsprop', loss='mse', metrics=['mae'])
history = model.fit(inputs_train, outputs_train, epochs=200,
batch_size=1, validation_data=(inputs_validate, outputs_validate))
```

```
Epoch 1/200

Exception ignored in: <function _xla_gc_callback at 0x7fb866ccb380>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.11/dist-packages/jax/_src/lib/__init__.py",
line 96, in _xla_gc_callback
    def _xla_gc_callback(*args):

KeyboardInterrupt:

10/10 ━━━━━━━━━━━━━━━━━━━━ 68s 32ms/step - loss: 0.5109 - mae: 0.5786
- val_loss: 0.5000 - val_mae: 0.5000
Epoch 2/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.4592 - mae: 0.4592 -
val_loss: 0.5000 - val_mae: 0.5000
Epoch 3/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.4592 - mae: 0.4592 -
val_loss: 0.5000 - val_mae: 0.5000
Epoch 4/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 0.4592 - mae: 0.4592 -
val_loss: 0.5000 - val_mae: 0.5000
Epoch 5/200
```

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.4592 - mae: 0.4592 -
val_loss: 0.5000 - val_mae: 0.5000
Epoch 6/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.4592 - mae: 0.4592 -
val_loss: 0.5000 - val_mae: 0.5000
Epoch 7/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 0.4592 - mae: 0.4592 -
val_loss: 0.5000 - val_mae: 0.5000
Epoch 8/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - loss: 0.4592 - mae: 0.4592 -
val_loss: 0.5000 - val_mae: 0.5000
Epoch 9/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step - loss: 0.4592 - mae: 0.4592 -
val_loss: 0.5000 - val_mae: 0.5000
Epoch 10/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.4592 - mae: 0.4592 -
val_loss: 0.5000 - val_mae: 0.5000
Epoch 11/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - loss: 0.4592 - mae: 0.4592 -
val_loss: 0.5000 - val_mae: 0.5000
Epoch 12/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - loss: 0.4592 - mae: 0.4592 -
val_loss: 0.5000 - val_mae: 0.5000
Epoch 13/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step - loss: 0.4592 - mae: 0.4592 -
val_loss: 0.5000 - val_mae: 0.5000
Epoch 14/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.4592 - mae: 0.4592 -
val_loss: 0.5000 - val_mae: 0.5000
Epoch 15/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.4592 - mae: 0.4592 -
val_loss: 0.5000 - val_mae: 0.5000
Epoch 16/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 0.4756 - mae: 0.4776 -
val_loss: 0.5000 - val_mae: 0.5000
Epoch 17/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.5408 - mae: 0.5408 -
val_loss: 0.5000 - val_mae: 0.5000
Epoch 18/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.5408 - mae: 0.5408 -
val_loss: 0.5000 - val_mae: 0.5000
Epoch 19/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 0.5408 - mae: 0.5408 -
val_loss: 0.5000 - val_mae: 0.5000
Epoch 20/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.5408 - mae: 0.5408 -
val_loss: 0.5000 - val_mae: 0.5000
Epoch 21/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step - loss: 0.5408 - mae: 0.5408 -
```

```
val_loss: 0.5000 - val_mae: 0.5000
Epoch 22/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - loss: 0.5408 - mae: 0.5408 -
val_loss: 0.5000 - val_mae: 0.5000
Epoch 23/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - loss: 0.5408 - mae: 0.5408 -
val_loss: 0.4999 - val_mae: 0.5000
Epoch 24/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.5408 - mae: 0.5408 -
val_loss: 0.4998 - val_mae: 0.4999
Epoch 25/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 0.5543 - mae: 0.5653 -
val_loss: 0.4989 - val_mae: 0.4997
Epoch 26/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 0.5396 - mae: 0.5405 -
val_loss: 0.4982 - val_mae: 0.4995
Epoch 27/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.5385 - mae: 0.5404 -
val_loss: 0.4874 - val_mae: 0.4973
Epoch 28/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.4978 - mae: 0.5457 -
val_loss: 0.2673 - val_mae: 0.4924
Epoch 29/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.4659 - mae: 0.6109 -
val_loss: 0.2335 - val_mae: 0.4791
Epoch 30/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 0.4165 - mae: 0.6236 -
val_loss: 0.2269 - val_mae: 0.4763
Epoch 31/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.3776 - mae: 0.5952 -
val_loss: 0.2231 - val_mae: 0.4722
Epoch 32/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.3567 - mae: 0.5785 -
val_loss: 0.2192 - val_mae: 0.4680
Epoch 33/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.3448 - mae: 0.5689 -
val_loss: 0.2155 - val_mae: 0.4640
Epoch 34/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.3356 - mae: 0.5613 -
val_loss: 0.2121 - val_mae: 0.4602
Epoch 35/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 0.3277 - mae: 0.5547 -
val_loss: 0.2088 - val_mae: 0.4566
Epoch 36/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.3205 - mae: 0.5486 -
val_loss: 0.2057 - val_mae: 0.4531
Epoch 37/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.3138 - mae: 0.5430 -
val_loss: 0.2027 - val_mae: 0.4498
```

```
Epoch 38/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - loss: 0.3076 - mae: 0.5378 -
val_loss: 0.1999 - val_mae: 0.4466
Epoch 39/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 0.3017 - mae: 0.5329 -
val_loss: 0.1972 - val_mae: 0.4436
Epoch 40/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.2962 - mae: 0.5282 -
val_loss: 0.1946 - val_mae: 0.4407
Epoch 41/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.2910 - mae: 0.5238 -
val_loss: 0.1921 - val_mae: 0.4379
Epoch 42/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.2860 - mae: 0.5196 -
val_loss: 0.1897 - val_mae: 0.4352
Epoch 43/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.2812 - mae: 0.5156 -
val_loss: 0.1875 - val_mae: 0.4326
Epoch 44/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 0.2767 - mae: 0.5118 -
val_loss: 0.1853 - val_mae: 0.4301
Epoch 45/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.2724 - mae: 0.5082 -
val_loss: 0.1832 - val_mae: 0.4277
Epoch 46/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.2683 - mae: 0.5047 -
val_loss: 0.1812 - val_mae: 0.4254
Epoch 47/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.2672 - mae: 0.5033 -
val_loss: 0.2579 - val_mae: 0.4212
Epoch 48/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.2663 - mae: 0.4264 -
val_loss: 0.1774 - val_mae: 0.4196
Epoch 49/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.2518 - mae: 0.4717 -
val_loss: 0.1646 - val_mae: 0.3933
Epoch 50/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 0.2459 - mae: 0.4655 -
val_loss: 0.1921 - val_mae: 0.4025
Epoch 51/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.2538 - mae: 0.4535 -
val_loss: 0.1839 - val_mae: 0.3990
Epoch 52/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.2426 - mae: 0.4503 -
val_loss: 0.1619 - val_mae: 0.3832
Epoch 53/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.2333 - mae: 0.4443 -
val_loss: 0.1566 - val_mae: 0.3781
Epoch 54/200
```

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.2291 - mae: 0.4405 -
val_loss: 0.1429 - val_mae: 0.3684
Epoch 55/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 0.2257 - mae: 0.4396 -
val_loss: 0.1592 - val_mae: 0.3595
Epoch 56/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.2213 - mae: 0.3993 -
val_loss: 0.1399 - val_mae: 0.3490
Epoch 57/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.2151 - mae: 0.4068 -
val_loss: 0.1350 - val_mae: 0.3438
Epoch 58/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.2092 - mae: 0.4029 -
val_loss: 0.1315 - val_mae: 0.3395
Epoch 59/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.2058 - mae: 0.4000 -
val_loss: 0.1277 - val_mae: 0.3350
Epoch 60/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 0.2015 - mae: 0.3968 -
val_loss: 0.1237 - val_mae: 0.3310
Epoch 61/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 0.1932 - mae: 0.3884 -
val_loss: 0.1316 - val_mae: 0.3414
Epoch 62/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - loss: 0.1971 - mae: 0.3932 -
val_loss: 0.1208 - val_mae: 0.3245
Epoch 63/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.1877 - mae: 0.3801 -
val_loss: 0.1183 - val_mae: 0.3199
Epoch 64/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.1898 - mae: 0.3816 -
val_loss: 0.1153 - val_mae: 0.3158
Epoch 65/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.1870 - mae: 0.3787 -
val_loss: 0.1122 - val_mae: 0.3117
Epoch 66/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 0.1843 - mae: 0.3758 -
val_loss: 0.1092 - val_mae: 0.3076
Epoch 67/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.1815 - mae: 0.3731 -
val_loss: 0.1084 - val_mae: 0.3090
Epoch 68/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.1766 - mae: 0.3680 -
val_loss: 0.1046 - val_mae: 0.3002
Epoch 69/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 0.1768 - mae: 0.3666 -
val_loss: 0.1057 - val_mae: 0.3075
Epoch 70/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.1734 - mae: 0.3649 -
```

```
val_loss: 0.1002 - val_mae: 0.2926
Epoch 71/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.1727 - mae: 0.3599 -
val_loss: 0.0975 - val_mae: 0.2895
Epoch 72/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - loss: 0.1805 - mae: 0.3713 -
val_loss: 0.1163 - val_mae: 0.3316
Epoch 73/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.1907 - mae: 0.4085 -
val_loss: 0.1133 - val_mae: 0.3197
Epoch 74/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.1862 - mae: 0.3878 -
val_loss: 0.0904 - val_mae: 0.2966
Epoch 75/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.1688 - mae: 0.3720 -
val_loss: 0.0838 - val_mae: 0.2887
Epoch 76/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - loss: 0.1691 - mae: 0.3734 -
val_loss: 0.0790 - val_mae: 0.2802
Epoch 77/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.1644 - mae: 0.3658 -
val_loss: 0.0774 - val_mae: 0.2771
Epoch 78/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.1635 - mae: 0.3633 -
val_loss: 0.0736 - val_mae: 0.2705
Epoch 79/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.1609 - mae: 0.3591 -
val_loss: 0.0708 - val_mae: 0.2652
Epoch 80/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.1586 - mae: 0.3550 -
val_loss: 0.0681 - val_mae: 0.2601
Epoch 81/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.1567 - mae: 0.3514 -
val_loss: 0.0657 - val_mae: 0.2554
Epoch 82/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step - loss: 0.1547 - mae: 0.3491 -
val_loss: 0.0632 - val_mae: 0.2505
Epoch 83/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - loss: 0.1529 - mae: 0.3455 -
val_loss: 0.0611 - val_mae: 0.2462
Epoch 84/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step - loss: 0.1522 - mae: 0.3434 -
val_loss: 0.0571 - val_mae: 0.2384
Epoch 85/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.1481 - mae: 0.3364 -
val_loss: 0.0559 - val_mae: 0.2358
Epoch 86/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.1482 - mae: 0.3356 -
val_loss: 0.0527 - val_mae: 0.2291
```

```
Epoch 87/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.1445 - mae: 0.3295 -
val_loss: 0.0521 - val_mae: 0.2277
Epoch 88/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.1454 - mae: 0.3297 -
val_loss: 0.0495 - val_mae: 0.2220
Epoch 89/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 0.1427 - mae: 0.3246 -
val_loss: 0.0466 - val_mae: 0.2153
Epoch 90/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.1399 - mae: 0.3190 -
val_loss: 0.0467 - val_mae: 0.2156
Epoch 91/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.1347 - mae: 0.3186 -
val_loss: 0.0475 - val_mae: 0.2154
Epoch 92/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.1397 - mae: 0.3178 -
val_loss: 0.0423 - val_mae: 0.2040
Epoch 93/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.1416 - mae: 0.3135 -
val_loss: 0.0391 - val_mae: 0.1975
Epoch 94/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.1250 - mae: 0.2987 -
val_loss: 0.0401 - val_mae: 0.1988
Epoch 95/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.1286 - mae: 0.3001 -
val_loss: 0.0403 - val_mae: 0.1986
Epoch 96/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.1308 - mae: 0.3027 -
val_loss: 0.0411 - val_mae: 0.1998
Epoch 97/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.1334 - mae: 0.3037 -
val_loss: 0.0364 - val_mae: 0.1892
Epoch 98/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.1275 - mae: 0.2939 -
val_loss: 0.0352 - val_mae: 0.1856
Epoch 99/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 0.1286 - mae: 0.2918 -
val_loss: 0.0335 - val_mae: 0.1810
Epoch 100/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.1282 - mae: 0.2890 -
val_loss: 0.0319 - val_mae: 0.1767
Epoch 101/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.1275 - mae: 0.2862 -
val_loss: 0.0305 - val_mae: 0.1730
Epoch 102/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.1266 - mae: 0.2838 -
val_loss: 0.0294 - val_mae: 0.1700
Epoch 103/200
```

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.1258 - mae: 0.2816 -
val_loss: 0.0285 - val_mae: 0.1675
Epoch 104/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.1250 - mae: 0.2798 -
val_loss: 0.0278 - val_mae: 0.1655
Epoch 105/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.1241 - mae: 0.2782 -
val_loss: 0.0272 - val_mae: 0.1640
Epoch 106/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.1233 - mae: 0.2769 -
val_loss: 0.0268 - val_mae: 0.1628
Epoch 107/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.1225 - mae: 0.2758 -
val_loss: 0.0265 - val_mae: 0.1619
Epoch 108/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.1215 - mae: 0.2745 -
val_loss: 0.0256 - val_mae: 0.1591
Epoch 109/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.1203 - mae: 0.2721 -
val_loss: 0.0243 - val_mae: 0.1554
Epoch 110/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 0.1193 - mae: 0.2700 -
val_loss: 0.0248 - val_mae: 0.1566
Epoch 111/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.1188 - mae: 0.2701 -
val_loss: 0.0227 - val_mae: 0.1502
Epoch 112/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.1174 - mae: 0.2663 -
val_loss: 0.0222 - val_mae: 0.1487
Epoch 113/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.1166 - mae: 0.2645 -
val_loss: 0.0223 - val_mae: 0.1489
Epoch 114/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.1160 - mae: 0.2641 -
val_loss: 0.0213 - val_mae: 0.1455
Epoch 115/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.1152 - mae: 0.2623 -
val_loss: 0.0214 - val_mae: 0.1458
Epoch 116/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.1144 - mae: 0.2614 -
val_loss: 0.0206 - val_mae: 0.1433
Epoch 117/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.1137 - mae: 0.2602 -
val_loss: 0.0202 - val_mae: 0.1417
Epoch 118/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.1128 - mae: 0.2587 -
val_loss: 0.0204 - val_mae: 0.1424
Epoch 119/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 0.1121 - mae: 0.2588 -
```

```
val_loss: 0.0195 - val_mae: 0.1391
Epoch 120/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 0.1111 - mae: 0.2562 -
val_loss: 0.0193 - val_mae: 0.1386
Epoch 121/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 0.1105 - mae: 0.2561 -
val_loss: 0.0190 - val_mae: 0.1373
Epoch 122/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.1096 - mae: 0.2547 -
val_loss: 0.0186 - val_mae: 0.1358
Epoch 123/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 0.1088 - mae: 0.2534 -
val_loss: 0.0184 - val_mae: 0.1350
Epoch 124/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.1079 - mae: 0.2521 -
val_loss: 0.0181 - val_mae: 0.1338
Epoch 125/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.1073 - mae: 0.2515 -
val_loss: 0.0179 - val_mae: 0.1333
Epoch 126/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.1065 - mae: 0.2507 -
val_loss: 0.0175 - val_mae: 0.1318
Epoch 127/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 0.1057 - mae: 0.2491 -
val_loss: 0.0173 - val_mae: 0.1309
Epoch 128/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.1049 - mae: 0.2482 -
val_loss: 0.0171 - val_mae: 0.1300
Epoch 129/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.1042 - mae: 0.2472 -
val_loss: 0.0169 - val_mae: 0.1291
Epoch 130/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.1034 - mae: 0.2462 -
val_loss: 0.0167 - val_mae: 0.1283
Epoch 131/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.1027 - mae: 0.2452 -
val_loss: 0.0162 - val_mae: 0.1263
Epoch 132/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 0.1020 - mae: 0.2435 -
val_loss: 0.0161 - val_mae: 0.1257
Epoch 133/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.1014 - mae: 0.2428 -
val_loss: 0.0159 - val_mae: 0.1251
Epoch 134/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.1007 - mae: 0.2419 -
val_loss: 0.0158 - val_mae: 0.1245
Epoch 135/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.1000 - mae: 0.2412 -
val_loss: 0.0159 - val_mae: 0.1251
```

```
Epoch 136/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0996 - mae: 0.2432 -
val_loss: 0.0157 - val_mae: 0.1241
Epoch 137/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0988 - mae: 0.2421 -
val_loss: 0.0156 - val_mae: 0.1234
Epoch 138/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 0.0980 - mae: 0.2410 -
val_loss: 0.0154 - val_mae: 0.1229
Epoch 139/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0971 - mae: 0.2400 -
val_loss: 0.0153 - val_mae: 0.1224
Epoch 140/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 0.0964 - mae: 0.2391 -
val_loss: 0.0152 - val_mae: 0.1218
Epoch 141/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0956 - mae: 0.2382 -
val_loss: 0.0155 - val_mae: 0.1227
Epoch 142/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0948 - mae: 0.2379 -
val_loss: 0.0153 - val_mae: 0.1219
Epoch 143/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - loss: 0.0878 - mae: 0.2320 -
val_loss: 0.0122 - val_mae: 0.1057
Epoch 144/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.1025 - mae: 0.2324 -
val_loss: 0.0111 - val_mae: 0.1045
Epoch 145/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0974 - mae: 0.2250 -
val_loss: 0.0131 - val_mae: 0.1126
Epoch 146/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0955 - mae: 0.2298 -
val_loss: 0.0132 - val_mae: 0.1130
Epoch 147/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0920 - mae: 0.2277 -
val_loss: 0.0136 - val_mae: 0.1143
Epoch 148/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0869 - mae: 0.2258 -
val_loss: 0.0111 - val_mae: 0.1015
Epoch 149/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0969 - mae: 0.2251 -
val_loss: 0.0112 - val_mae: 0.1039
Epoch 150/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - loss: 0.0914 - mae: 0.2196 -
val_loss: 0.0120 - val_mae: 0.1069
Epoch 151/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0892 - mae: 0.2213 -
val_loss: 0.0127 - val_mae: 0.1103
Epoch 152/200
```

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0834 - mae: 0.2194 -
val_loss: 0.0102 - val_mae: 0.0964
Epoch 153/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0945 - mae: 0.2213 -
val_loss: 0.0112 - val_mae: 0.1030
Epoch 154/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step - loss: 0.0889 - mae: 0.2181 -
val_loss: 0.0115 - val_mae: 0.1041
Epoch 155/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0812 - mae: 0.2136 -
val_loss: 0.0106 - val_mae: 0.0961
Epoch 156/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - loss: 0.0932 - mae: 0.2167 -
val_loss: 0.0095 - val_mae: 0.0955
Epoch 157/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 0.0874 - mae: 0.2101 -
val_loss: 0.0104 - val_mae: 0.0989
Epoch 158/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0849 - mae: 0.2119 -
val_loss: 0.0110 - val_mae: 0.1014
Epoch 159/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0791 - mae: 0.2106 -
val_loss: 0.0095 - val_mae: 0.0913
Epoch 160/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 0.0886 - mae: 0.2136 -
val_loss: 0.0092 - val_mae: 0.0954
Epoch 161/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0822 - mae: 0.2049 -
val_loss: 0.0090 - val_mae: 0.0921
Epoch 162/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 0.0829 - mae: 0.2051 -
val_loss: 0.0081 - val_mae: 0.0872
Epoch 163/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0944 - mae: 0.2093 -
val_loss: 0.0081 - val_mae: 0.0886
Epoch 164/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0767 - mae: 0.1954 -
val_loss: 0.0079 - val_mae: 0.0857
Epoch 165/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0921 - mae: 0.2064 -
val_loss: 0.0079 - val_mae: 0.0872
Epoch 166/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0748 - mae: 0.1929 -
val_loss: 0.0077 - val_mae: 0.0845
Epoch 167/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0913 - mae: 0.2054 -
val_loss: 0.0079 - val_mae: 0.0871
Epoch 168/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0801 - mae: 0.1989 -
val_loss: 0.0091 - val_mae: 0.0842
```

```
Epoch 169/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0902 - mae: 0.2055 -
val_loss: 0.0062 - val_mae: 0.0780
Epoch 170/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0828 - mae: 0.1919 -
val_loss: 0.0075 - val_mae: 0.0836
Epoch 171/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0798 - mae: 0.1950 -
val_loss: 0.0082 - val_mae: 0.0871
Epoch 172/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 0.0700 - mae: 0.1903 -
val_loss: 0.0075 - val_mae: 0.0833
Epoch 173/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - loss: 0.0870 - mae: 0.2009 -
val_loss: 0.0075 - val_mae: 0.0849
Epoch 174/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0708 - mae: 0.1884 -
val_loss: 0.0075 - val_mae: 0.0831
Epoch 175/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0878 - mae: 0.2019 -
val_loss: 0.0073 - val_mae: 0.0835
Epoch 176/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 0.0774 - mae: 0.1940 -
val_loss: 0.0066 - val_mae: 0.0780
Epoch 177/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 0.0853 - mae: 0.1964 -
val_loss: 0.0067 - val_mae: 0.0801
Epoch 178/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0679 - mae: 0.1831 -
val_loss: 0.0071 - val_mae: 0.0796
Epoch 179/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0869 - mae: 0.1992 -
val_loss: 0.0068 - val_mae: 0.0810
Epoch 180/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 0.0677 - mae: 0.1832 -
val_loss: 0.0069 - val_mae: 0.0792
Epoch 181/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0858 - mae: 0.1979 -
val_loss: 0.0067 - val_mae: 0.0799
Epoch 182/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0735 - mae: 0.1869 -
val_loss: 0.0061 - val_mae: 0.0731
Epoch 183/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0837 - mae: 0.1920 -
val_loss: 0.0057 - val_mae: 0.0747
Epoch 184/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 0.0651 - mae: 0.1761 -
val_loss: 0.0065 - val_mae: 0.0764
Epoch 185/200
```

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0838 - mae: 0.1938 -
val_loss: 0.0065 - val_mae: 0.0786
Epoch 186/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0654 - mae: 0.1799 -
val_loss: 0.0066 - val_mae: 0.0770
Epoch 187/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0700 - mae: 0.1841 -
val_loss: 0.0063 - val_mae: 0.0725
Epoch 188/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 0.0882 - mae: 0.1960 -
val_loss: 0.0052 - val_mae: 0.0719
Epoch 189/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0638 - mae: 0.1714 -
val_loss: 0.0063 - val_mae: 0.0721
Epoch 190/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0840 - mae: 0.1917 -
val_loss: 0.0052 - val_mae: 0.0718
Epoch 191/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0629 - mae: 0.1709 -
val_loss: 0.0063 - val_mae: 0.0720
Epoch 192/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0826 - mae: 0.1903 -
val_loss: 0.0052 - val_mae: 0.0717
Epoch 193/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0623 - mae: 0.1704 -
val_loss: 0.0063 - val_mae: 0.0718
Epoch 194/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0812 - mae: 0.1889 -
val_loss: 0.0052 - val_mae: 0.0714
Epoch 195/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0611 - mae: 0.1693 -
val_loss: 0.0062 - val_mae: 0.0711
Epoch 196/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 0.0799 - mae: 0.1872 -
val_loss: 0.0051 - val_mae: 0.0706
Epoch 197/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 0.0603 - mae: 0.1682 -
val_loss: 0.0062 - val_mae: 0.0705
Epoch 198/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0656 - mae: 0.1761 -
val_loss: 0.0051 - val_mae: 0.0655
Epoch 199/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 0.0810 - mae: 0.1835 -
val_loss: 0.0045 - val_mae: 0.0668
Epoch 200/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0607 - mae: 0.1650 -
val_loss: 0.0057 - val_mae: 0.0689
```

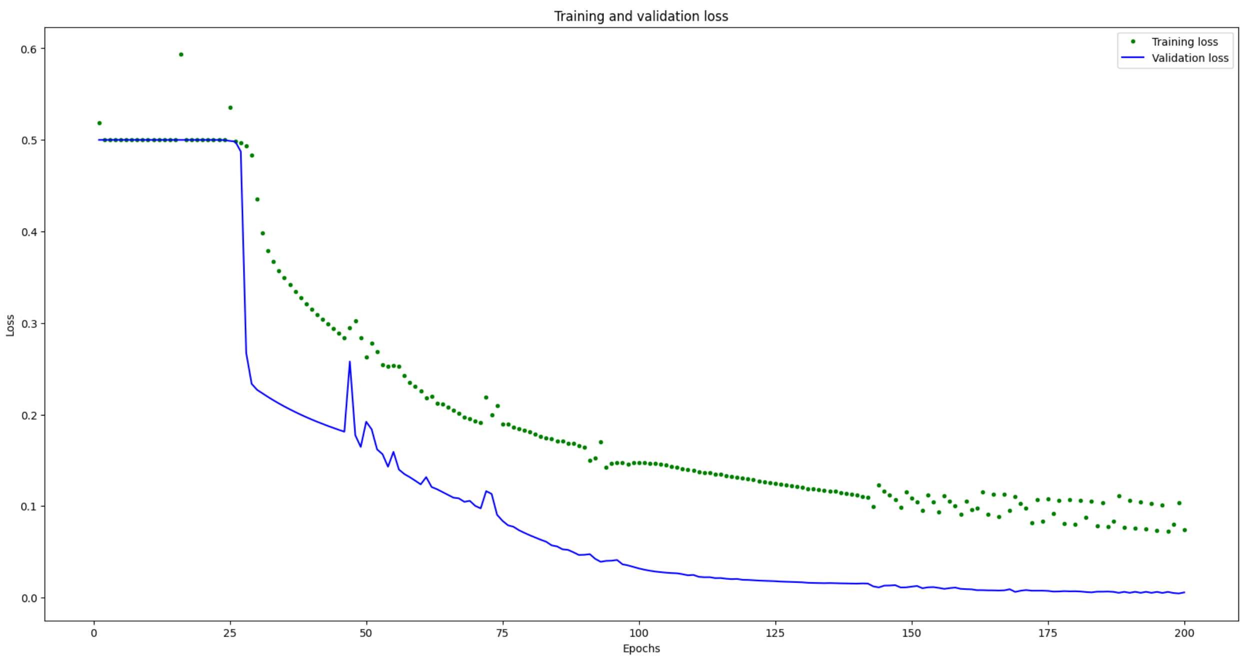
Verify

Graph the models performance vs validation.

Graph the loss

Graph the loss to see when the model stops improving.

```
# increase the size of the graphs. The default size is (6,4).
plt.rcParams["figure.figsize"] = (20,10)

# graph the loss, the model above is configure to use "mean squared
error" as the loss function
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'g.', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

print(plt.rcParams["figure.figsize"])
```

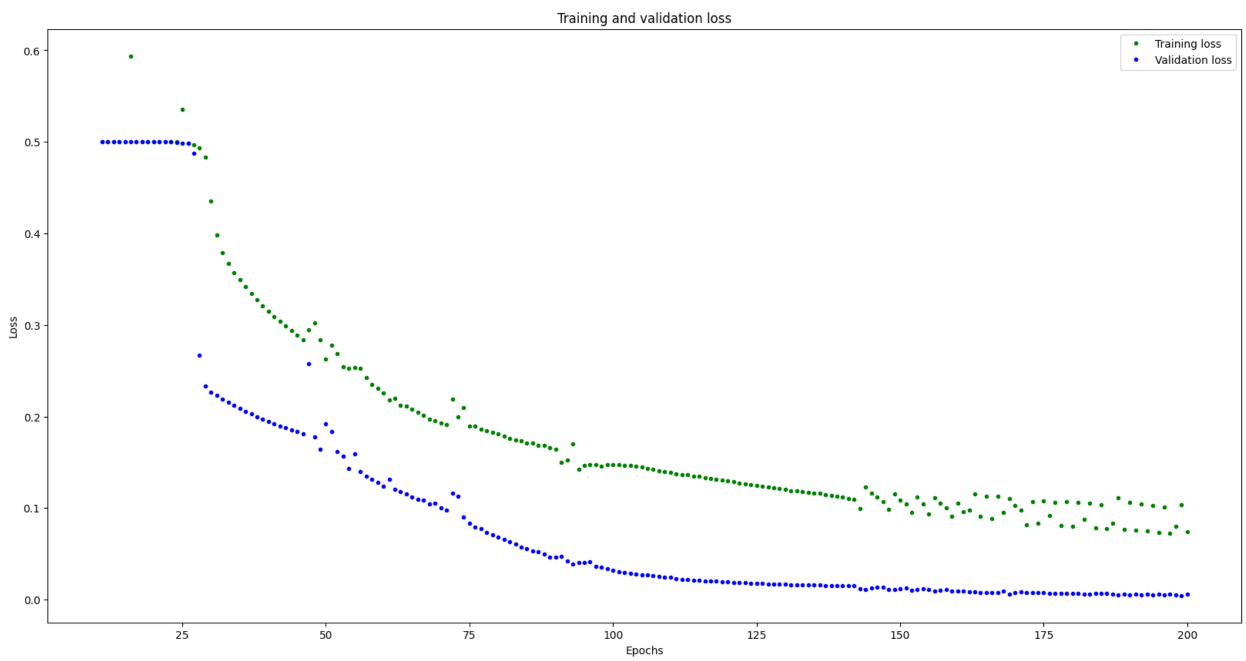



```
[20.0, 10.0]
```

Graph the loss again, skipping a bit of the start

We'll graph the same data as the previous code cell, but start at index 100 so we can further zoom in once the model starts to converge.

```
# graph the loss again skipping a bit of the start
SKIP = 10
plt.plot(epochs[SKIP:], loss[SKIP:], 'g.', label='Training loss')
plt.plot(epochs[SKIP:], val_loss[SKIP:], 'b.', label='Validation
loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

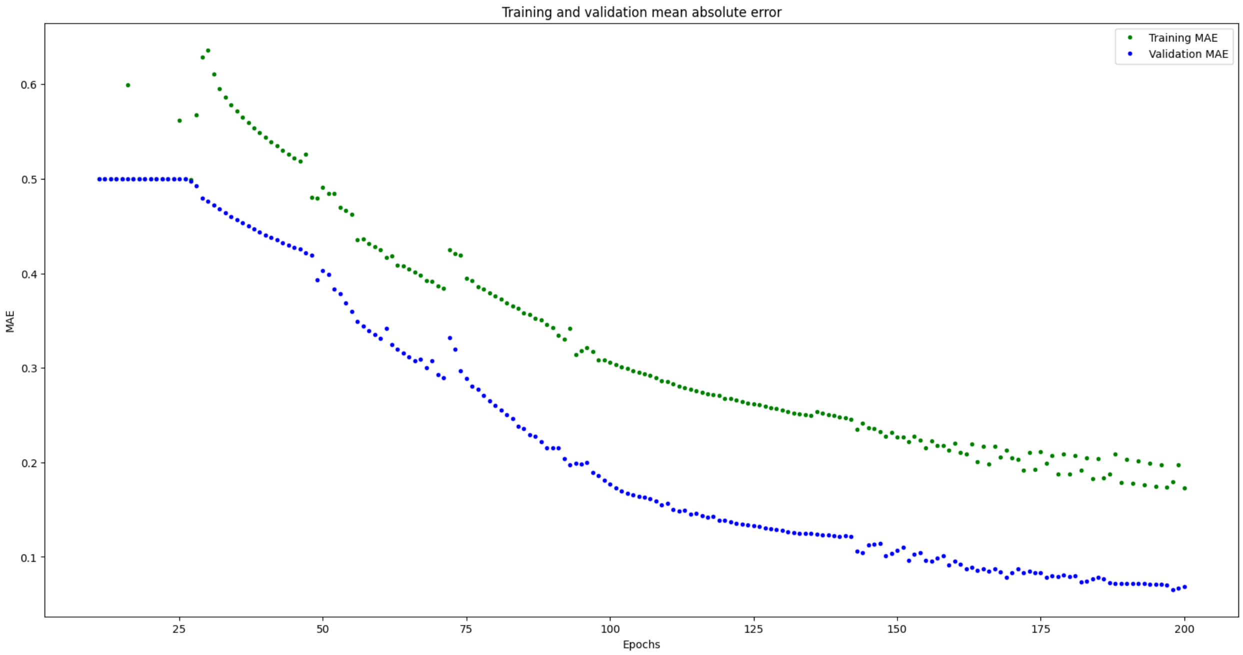


Graph the mean absolute error

Mean absolute error is another metric to judge the performance of the model.

```
# graph of mean absolute error
mae = history.history['mae']
val_mae = history.history['val_mae']
plt.plot(epochs[SKIP:], mae[SKIP:], 'g.', label='Training MAE')
plt.plot(epochs[SKIP:], val_mae[SKIP:], 'b.', label='Validation MAE')
plt.title('Training and validation mean absolute error')
plt.xlabel('Epochs')
plt.ylabel('MAE')
```

```
plt.legend()
plt.show()
```



Run with Test Data

Put our test data into the model and plot the predictions

```
# use the model to predict the test inputs
predictions = model.predict(inputs_test)

# print the predictions and the expected ouputs
print("predictions =\n", np.round(predictions, decimals=3))
print("actual =\n", outputs_test)

# Plot the predictions along with to the test data
##plt.clf()
##plt.title('Training data predicted vs actual values')
##plt.plot(inputs_test, outputs_test, 'b.', label='Actual')
##plt.plot(inputs_test, predictions, 'r.', label='Predicted')
##plt.show()

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 114ms/step
predictions =
 [[0.261 0.739]
 [0.669 0.331]
 [0.056 0.944]]
actual =
 [[0. 1.]
```

```
 [0. 1.]
 [0. 1.]]
```

Convert the Trained Model to Tensor Flow Lite

The next cell converts the model to TFlite format. The size in bytes of the model is also printed out.

```
# Convert the model to the TensorFlow Lite format without quantization
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Save the model to disk
open("gesture_model.tflite", "wb").write(tflite_model)

import os
basic_model_size = os.path.getsize("gesture_model.tflite")
print("Model is %d bytes" % basic_model_size)
```

```
Saved artifact at '/tmp/tmp6tk6fw8o'. The following endpoints are
available:

* Endpoint 'serve'
  args_0 (POSITIONAL_ONLY): TensorSpec(shape=(1, 3000),
dtype=tf.float32, name='keras_tensor_52')
Output Type:
  TensorSpec(shape=(1, 2), dtype=tf.float32, name=None)
Captures:
  140428993371920: TensorSpec(shape=(), dtype=tf.resource, name=None)
  140428993373072: TensorSpec(shape=(), dtype=tf.resource, name=None)
  140428993371728: TensorSpec(shape=(), dtype=tf.resource, name=None)
  140428993373648: TensorSpec(shape=(), dtype=tf.resource, name=None)
  140428993373456: TensorSpec(shape=(), dtype=tf.resource, name=None)
  140428993374416: TensorSpec(shape=(), dtype=tf.resource, name=None)
Model is 606476 bytes
```

Encode the Model in an Arduino Header File

The next cell creates a constant byte array that contains the TFlite model. Import it as a tab with the sketch below.

```
!echo "const unsigned char model[] = {" > /content/model.h
!cat gesture_model.tflite | xxd -i      >> /content/model.h
!echo "};"                              >> /content/model.h

import os
```

```
model_h_size = os.path.getsize("model.h")
print(f"Header file, model.h, is {model_h_size:,} bytes.")
print("\nOpen the side panel (refresh if needed). Double click model.h
to download the file.")
```

```
Header file, model.h, is 3,739,970 bytes.

Open the side panel (refresh if needed). Double click model.h to
download the file.
```

Classifying IMU Data

Now it's time to switch back to the tutorial instructions and run our new model on the Arduino Nano 33 BLE or Seeed Xiao nrf52 Sense to classify the accelerometer and gyroscope data.