In [4]:

```
!pip3 install micromlgen
!pip3 install pandas numpy matplotlib
!pip3 install --upgrade tensorflow==2.3.0rc1
```

```
(local\programs\python\python310\lib\site-packages (from matplotlib) (1.
0.7)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\ronan\appdat
a\local\programs\python\python310\lib\site-packages (from matplotlib) (1.
3.2)
Requirement already satisfied: cycler>=0.10 in c:\users\ronan\appdata\loc
al\programs\python\python310\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: six>=1.5 in c:\users\ronan\appdata\local\p
rograms\python\python310\lib\site-packages (from python-dateutil>=2.8.1->
pandas) (1.16.0)
```

In [5]:

```
import numpy
numpy.version.version
import tensorflow as tf
```

In [6]:

```
import pandas as pd
df = pd.read_csv('htmallgesturesfinal.csv', low_memory=False)
df.head()
```

Out[6]:

	aX	aY	aZ	gX	gY	gZ	gesture
0	0.272	-1.297	0.457	69.519	-38.818	12.390	1
1	0.257	-1.252	0.480	65.063	-39.307	18.494	1
2	0.266	-1.249	0.483	62.073	-38.452	25.940	1
3	0.298	-1.223	0.468	51.880	-34.729	41.260	1
4	0.299	-1.164	0.462	47.791	-32.959	48.157	1

In [7]:

```
import pandas as pd
df1 = pd.read_csv('nonhtmallgesturesfinal.csv', low_memory=False)
df1.head()
```

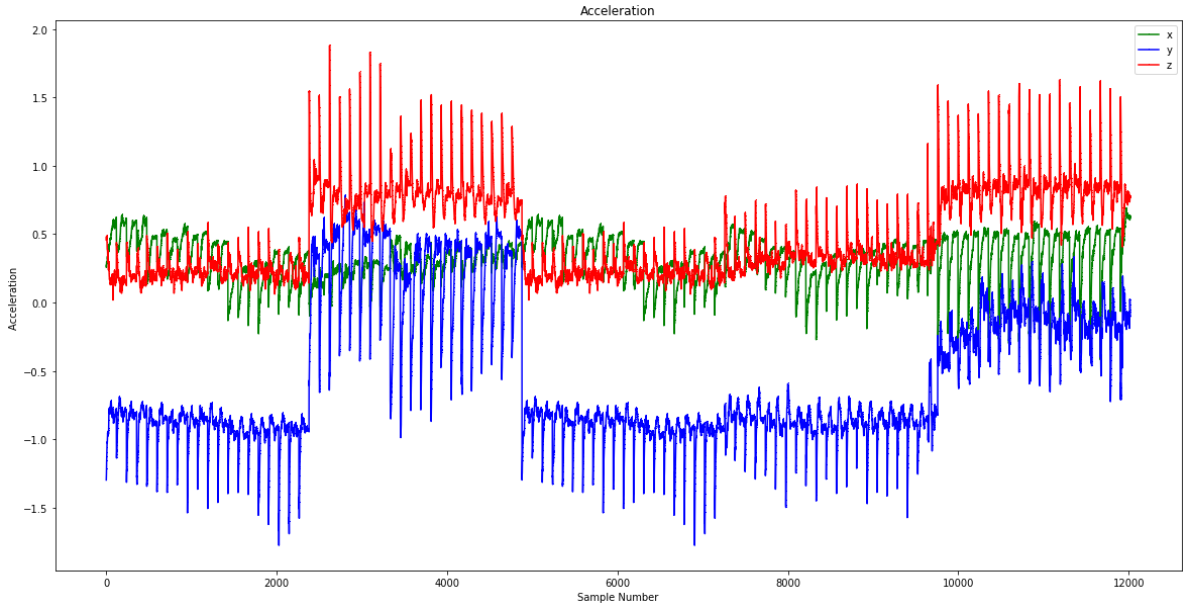
Out[7]:

	aX	aY	aZ	gX	gY	gZ	gesture
0	0.099	1.658	0.412	94.116	-36.255	-26.062	1
1	0.078	1.196	0.562	127.319	-26.978	-42.603	1
2	0.155	0.714	0.749	139.648	-13.000	-47.668	1
3	0.217	0.500	0.919	141.418	-4.578	-50.903	1
4	0.232	0.435	0.878	136.108	-1.587	-59.875	1

In [8]:

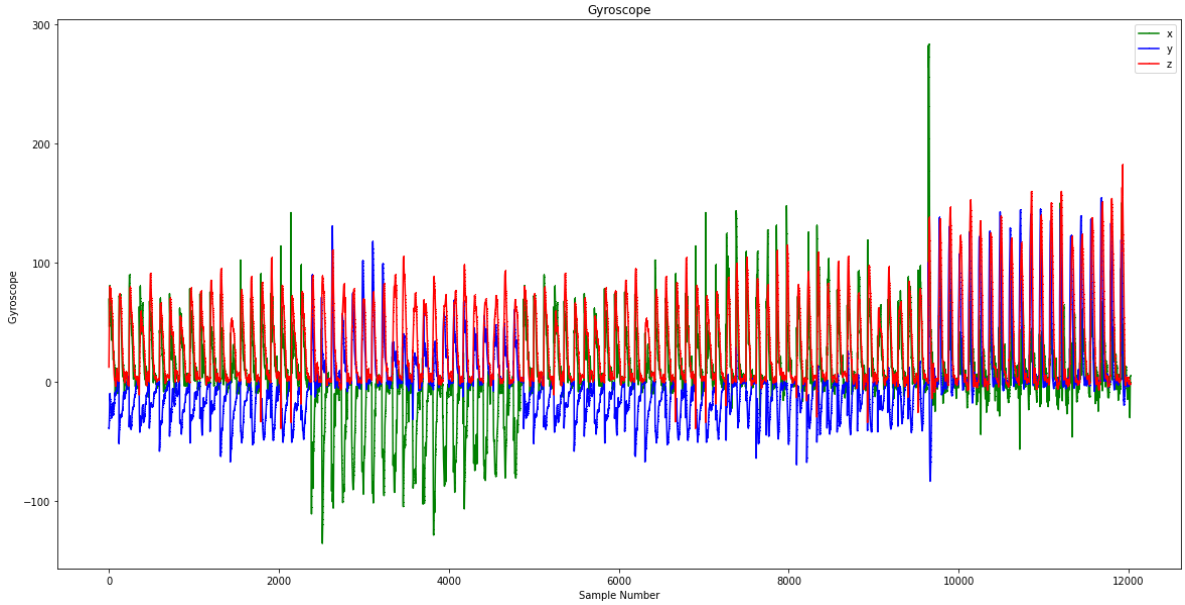
```
import matplotlib.pyplot as plt
index = range(1, len(df['aX']) + 1)
plt.rcParams["figure.figsize"] = (20,10)
plt.plot(index, df['aX'], 'g.', label='x', linestyle='solid', marker=',')
plt.plot(index, df['aY'], 'b.', label='y', linestyle='solid', marker=',')
plt.plot(index, df['aZ'], 'r.', label='z', linestyle='solid', marker=',')
plt.title("Acceleration")
plt.xlabel("Sample Number")
plt.ylabel("Acceleration ")
plt.legend()
plt.show()

plt.plot(index, df['gX'], 'g.', label='x', linestyle='solid', marker=',')
plt.plot(index, df['gY'], 'b.', label='y', linestyle='solid', marker=',')
plt.plot(index, df['gZ'], 'r.', label='z', linestyle='solid', marker=',')
plt.title("Gyroscope")
plt.xlabel("Sample Number")
plt.ylabel("Gyroscope ")
plt.legend()
plt.show()
```

```
C:\Users\ronan\AppData\Local\Temp\ipykernel_14160\3217031525.py:4: UserWarni
ng: marker is redundantly defined by the 'marker' keyword argument and the f
mt string "g." (-> marker='.'). The keyword argument will take precedence.
  plt.plot(index, df['aX'], 'g.', label='x', linestyle='solid', marker=',')
C:\Users\ronan\AppData\Local\Temp\ipykernel_14160\3217031525.py:5: UserWarni
ng: marker is redundantly defined by the 'marker' keyword argument and the f
mt string "b." (-> marker='.'). The keyword argument will take precedence.
  plt.plot(index, df['aY'], 'b.', label='y', linestyle='solid', marker=',')
C:\Users\ronan\AppData\Local\Temp\ipykernel_14160\3217031525.py:6: UserWarni
ng: marker is redundantly defined by the 'marker' keyword argument and the f
mt string "r." (-> marker='.'). The keyword argument will take precedence.
  plt.plot(index, df['aZ'], 'r.', label='z', linestyle='solid', marker=',')
```




```
C:\Users\ronan\AppData\Local\Temp\ipykernel_14160\3217031525.py:13: UserW
arning: marker is redundantly defined by the 'marker' keyword argument an
d the fmt string "g." (-> marker='.'). The keyword argument will take pre
cedence.
  plt.plot(index, df['gX'], 'g.', label='x', linestyle='solid', marker
=',')
```

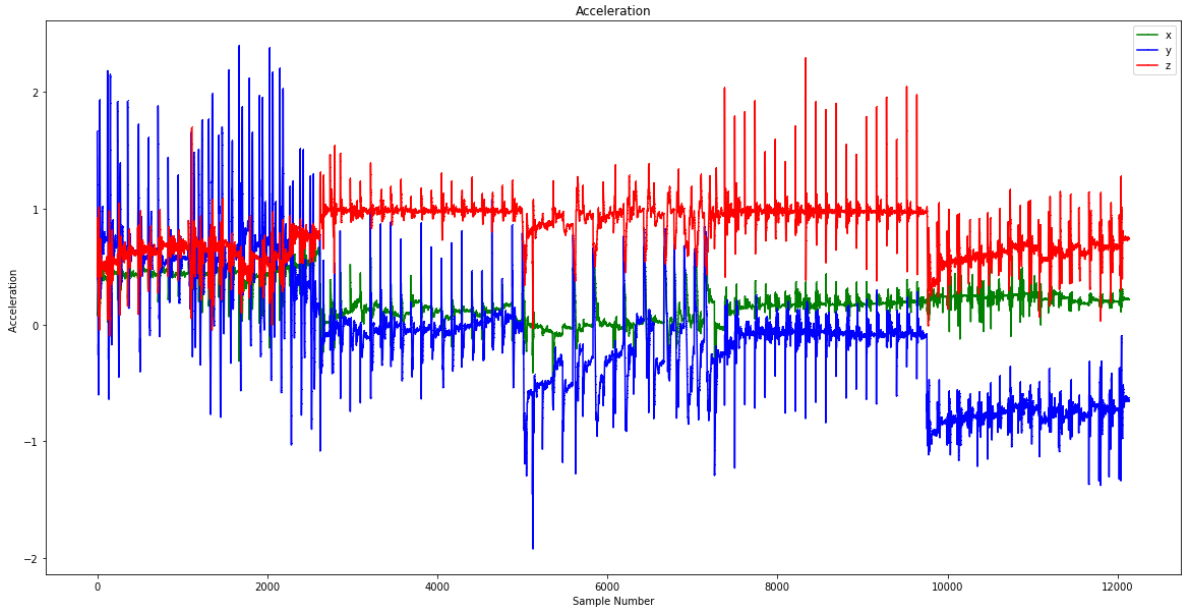
```
C:\Users\ronan\AppData\Local\Temp\ipykernel_14160\3217031525.py:14: UserW
arning: marker is redundantly defined by the 'marker' keyword argument an
d the fmt string "b." (-> marker='.'). The keyword argument will take pre
cedence.
  plt.plot(index, df['gY'], 'b.', label='y', linestyle='solid', marker
=',')
C:\Users\ronan\AppData\Local\Temp\ipykernel_14160\3217031525.py:15: UserW
arning: marker is redundantly defined by the 'marker' keyword argument an
d the fmt string "r." (-> marker='.'). The keyword argument will take pre
cedence.
  plt.plot(index, df['gZ'], 'r.', label='z', linestyle='solid', marker
=',')
```



In [9]:

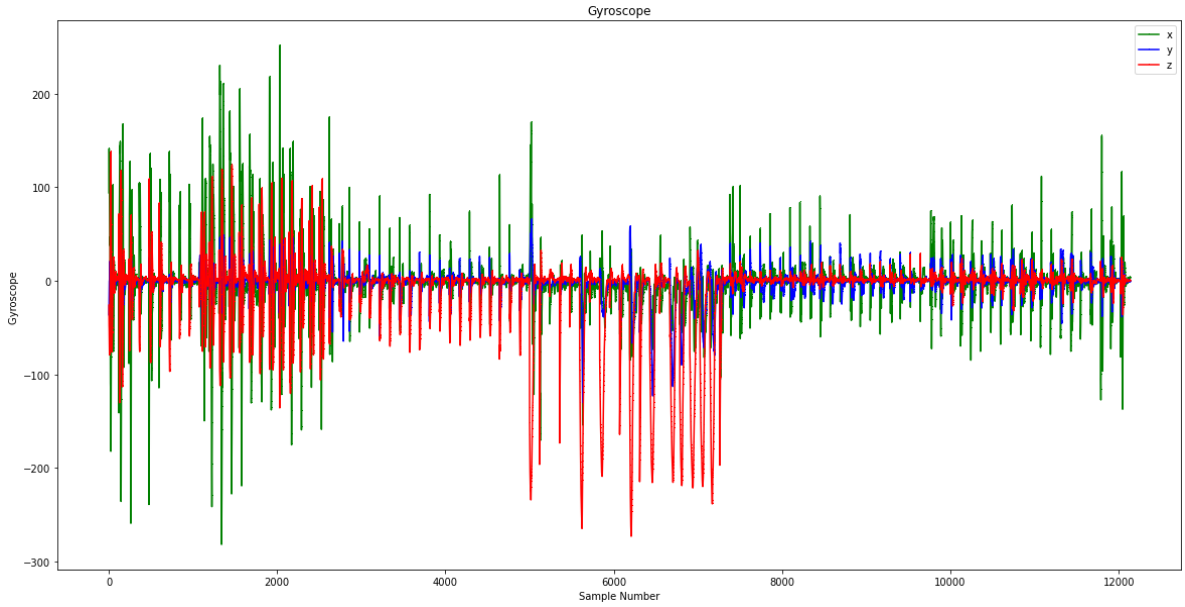
```
import matplotlib.pyplot as plt
index = range(1, len(df1['aX']) + 1)
plt.rcParams["figure.figsize"] = (20,10)
plt.plot(index, df1['aX'], 'g.', label='x', linestyle='solid', marker=',')
plt.plot(index, df1['aY'], 'b.', label='y', linestyle='solid', marker=',')
plt.plot(index, df1['aZ'], 'r.', label='z', linestyle='solid', marker=',')
plt.title("Acceleration")
plt.xlabel("Sample Number")
plt.ylabel("Acceleration ")
plt.legend()
plt.show()

plt.plot(index, df1['gX'], 'g.', label='x', linestyle='solid', marker=',')
plt.plot(index, df1['gY'], 'b.', label='y', linestyle='solid', marker=',')
plt.plot(index, df1['gZ'], 'r.', label='z', linestyle='solid', marker=',')
plt.title("Gyroscope")
plt.xlabel("Sample Number")
plt.ylabel("Gyroscope ")
plt.legend()
plt.show()
```

```
C:\Users\ronan\AppData\Local\Temp\ipykernel_14160\1393999084.py:4: UserWarni
ng: marker is redundantly defined by the 'marker' keyword argument and the f
mt string "g." (-> marker='.'). The keyword argument will take precedence.
  plt.plot(index, df1['aX'], 'g.', label='x', linestyle='solid', marker=',')
C:\Users\ronan\AppData\Local\Temp\ipykernel_14160\1393999084.py:5: UserWarni
ng: marker is redundantly defined by the 'marker' keyword argument and the f
mt string "b." (-> marker='.'). The keyword argument will take precedence.
  plt.plot(index, df1['aY'], 'b.', label='y', linestyle='solid', marker=',')
C:\Users\ronan\AppData\Local\Temp\ipykernel_14160\1393999084.py:6: UserWarni
ng: marker is redundantly defined by the 'marker' keyword argument and the f
mt string "r." (-> marker='.'). The keyword argument will take precedence.
  plt.plot(index, df1['aZ'], 'r.', label='z', linestyle='solid', marker=',')
```




```
C:\Users\ronan\AppData\Local\Temp\ipykernel_14160\1393999084.py:13: UserW
arning: marker is redundantly defined by the 'marker' keyword argument an
d the fmt string "g." (-> marker='.'). The keyword argument will take pre
cedence.
  plt.plot(index, df1['gX'], 'g.', label='x', linestyle='solid', marker
=',')
```

```
C:\Users\ronan\AppData\Local\Temp\ipykernel_14160\1393999084.py:14: UserW
arning: marker is redundantly defined by the 'marker' keyword argument an
d the fmt string "b." (-> marker='.'). The keyword argument will take pre
cedence.
  plt.plot(index, df1['gY'], 'b.', label='y', linestyle='solid', marker
=',')
C:\Users\ronan\AppData\Local\Temp\ipykernel_14160\1393999084.py:15: UserW
arning: marker is redundantly defined by the 'marker' keyword argument an
d the fmt string "r." (-> marker='.'). The keyword argument will take pre
cedence.
  plt.plot(index, df1['gZ'], 'r.', label='z', linestyle='solid', marker
=',')
```



In [10]:

```
import numpy as np
SEED = 1337
np.random.seed(SEED)
tf.random.set_seed(SEED)
GESTURES = [
    "htmallgestures",
    "nonhtmallgestures"

]

SAMPLES_PER_GESTURE = 119
NUM_GESTURES = len(GESTURES)

ONE_HOT_ENCODED_GESTURES = np.eye(NUM_GESTURES)
inputs = []
outputs = []

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(gesture + ".csv", low_memory=False)

  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

      tensor += [
          (df['aX'][index] + 4) / 8,
          (df['aY'][index] + 4) / 8,
          (df['aZ'][index] + 4) / 8,
          (df['gX'][index] + 2000) / 4000,
          (df['gY'][index] + 2000) / 4000,
          (df['gZ'][index] + 2000) / 4000
      ]

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

inputs = np.array(inputs)
outputs = np.array(outputs)
```

```
Processing index 0 for gesture 'htmallgestures'.
        There are 101 recordings of the htmallgestures gesture.
Processing index 1 for gesture 'nonhtmallgestures'.
        There are 102 recordings of the nonhtmallgestures gesture.
```

In [57]:

```
num_inputs = len(inputs)
randomize = np.arange(num_inputs)
np.random.shuffle(randomize)

inputs = inputs[randomize]
outputs = outputs[randomize]

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]
```

In [12]:

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

In [13]:

```
#building optimised model
model_optimised = tf.keras.Sequential()
model_optimised.add(tf.keras.layers.Dense(20, activation='relu'))
model_optimised.add(tf.keras.layers.Dense(15, activation='relu'))
model_optimised.add(tf.keras.layers.Dense(NUM_GESTURES, activation='softmax'))
model_optimised.compile(optimizer='rmsprop', loss='mse', metrics=['mae', 'accuracy'])
history_optimised = model_optimised.fit(inputs_train, outputs_train, epochs=600, batch_size
```

```
Epoch 596/600
121/121 [==============================] - 0s 1ms/step - loss: 0.0083 - m
ae: 0.0084 - accuracy: 0.9917 - val_loss: 3.1579e-06 - val_mae: 4.6236e-0
4 - val_accuracy: 1.0000
Epoch 597/600
121/121 [==============================] - 0s 1ms/step - loss: 9.4068e-08
- mae: 9.0047e-05 - accuracy: 1.0000 - val_loss: 5.6616e-08 - val_mae: 8.
8606e-05 - val_accuracy: 1.0000
Epoch 598/600
121/121 [==============================] - 0s 1ms/step - loss: 0.0083 - m
ae: 0.0083 - accuracy: 0.9917 - val_loss: 8.6673e-09 - val_mae: 6.0176e-0
5 - val_accuracy: 1.0000
Epoch 599/600
121/121 [==============================] - 0s 2ms/step - loss: 0.0121 - m
ae: 0.0167 - accuracy: 0.9835 - val_loss: 8.5549e-08 - val_mae: 9.5755e-0
5 - val_accuracy: 1.0000
Epoch 600/600
121/121 [==============================] - 0s 1ms/step - loss: 5.1065e-09
- mae: 4.8420e-05 - accuracy: 1.0000 - val_loss: 6.8115e-08 - val_mae: 9.
0002e-05 - val_accuracy: 1.0000
```

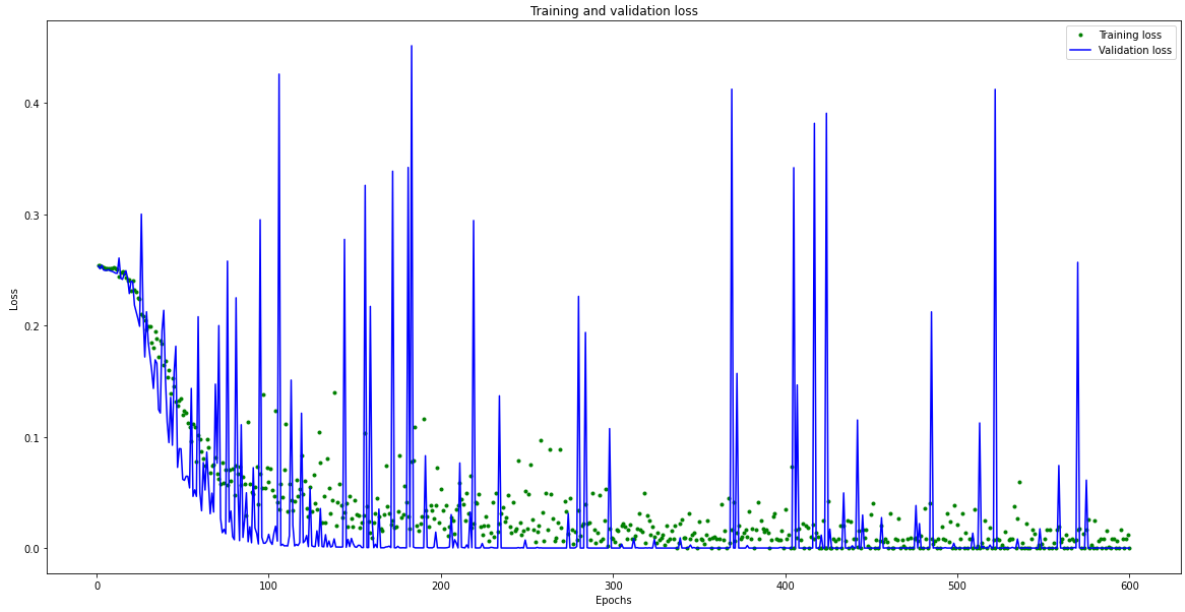
In [14]:

```
#building non-optimised model: 200 iterations
model_non200 = tf.keras.Sequential()
model_non200.add(tf.keras.layers.Dense(20, activation='relu'))
model_non200.add(tf.keras.layers.Dense(15, activation='relu'))
model_non200.add(tf.keras.layers.Dense(NUM_GESTURES, activation='softmax'))
model_non200.compile(optimizer='rmsprop', loss='mse', metrics=['mae', 'accuracy'])
history_nonoptimised = model_non200.fit(inputs_train, outputs_train, epochs=200, batch_size
```

```
121/121 [==============================] - 0s 1ms/step - loss: 0.2499 - m
ae: 0.4994 - accuracy: 0.5207 - val_loss: 0.2517 - val_mae: 0.5011 - val_
accuracy: 0.4762
Epoch 197/200
121/121 [==============================] - 0s 1ms/step - loss: 0.2499 - m
ae: 0.4992 - accuracy: 0.5207 - val_loss: 0.2516 - val_mae: 0.5011 - val_
accuracy: 0.4762
Epoch 198/200
121/121 [==============================] - 0s 1ms/step - loss: 0.2499 - m
ae: 0.4994 - accuracy: 0.5207 - val_loss: 0.2516 - val_mae: 0.5011 - val_
accuracy: 0.4762
Epoch 199/200
121/121 [==============================] - 0s 1ms/step - loss: 0.2499 - m
ae: 0.4992 - accuracy: 0.5207 - val_loss: 0.2516 - val_mae: 0.5011 - val_
accuracy: 0.4762
Epoch 200/200
121/121 [==============================] - 0s 1ms/step - loss: 0.2499 - m
ae: 0.4994 - accuracy: 0.5207 - val_loss: 0.2516 - val_mae: 0.5011 - val_
accuracy: 0.4762
```

In [15]:

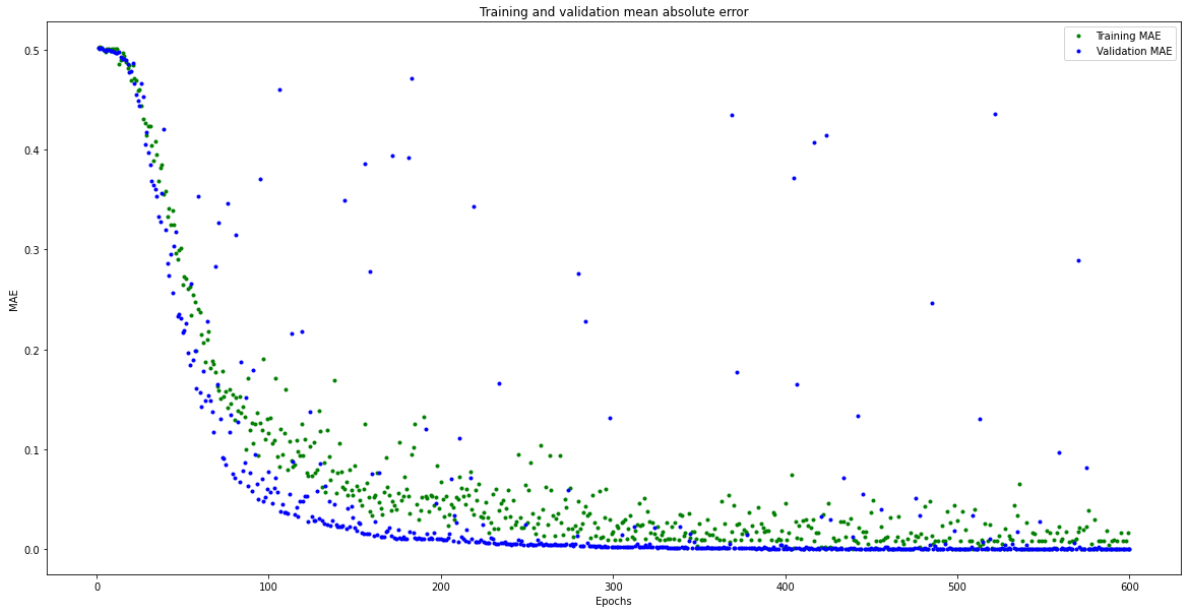
```
plt.rcParams["figure.figsize"] = (20,10)

# graph the loss, for the optimised model
loss = history_optimised.history['loss']
val_loss = history_optimised.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]
```

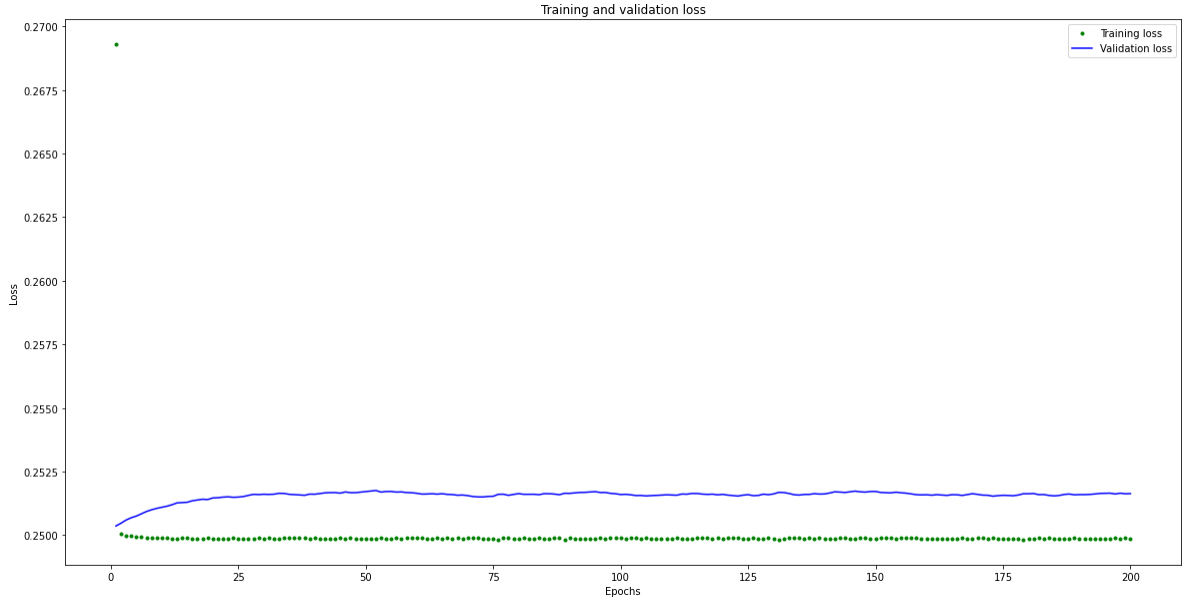
In [16]:

```
# graph of mean absolute error for optimised model
import matplotlib.pyplot as plt
training_mae = history_optimised.history['mae']
val_mae = history_optimised.history['val_mae']
plt.plot(epochs, training_mae, 'g.', label='Training MAE')
plt.plot(epochs, val_mae, 'b.', label='Validation MAE')
plt.title('Training and validation mean absolute error')
plt.xlabel('Epochs')
plt.ylabel('MAE')
plt.legend()
plt.show()
```



In [17]:

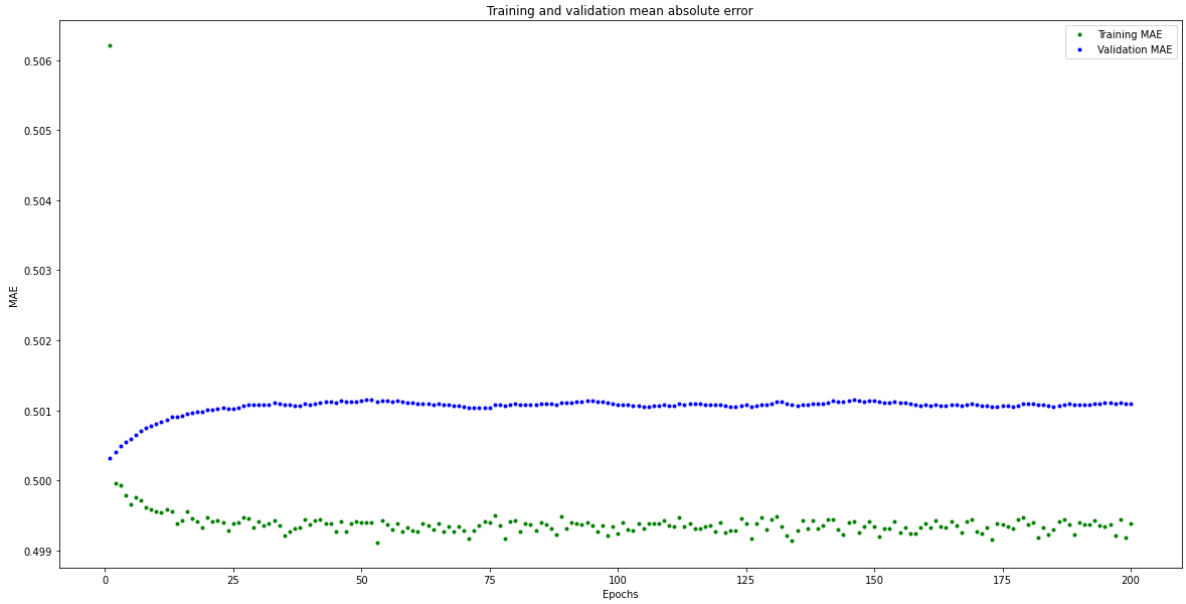
```
plt.rcParams["figure.figsize"] = (20,10)

# graph the loss, for the non-optimised model
loss = history_nonoptimised.history['loss']
val_loss = history_nonoptimised.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]
```

In [56]:

```
# graph of mean absolute error of the non-optimised model
training_mae = history_nonoptimised.history['mae']
val_mae = history_nonoptimised.history['val_mae']
plt.plot(epochs, training_mae, 'g.', label='Training MAE')
plt.plot(epochs, val_mae, 'b.', label='Validation MAE')
plt.title('Training and validation mean absolute error')
plt.xlabel('Epochs')
plt.ylabel('MAE')
plt.legend()
plt.show()
```

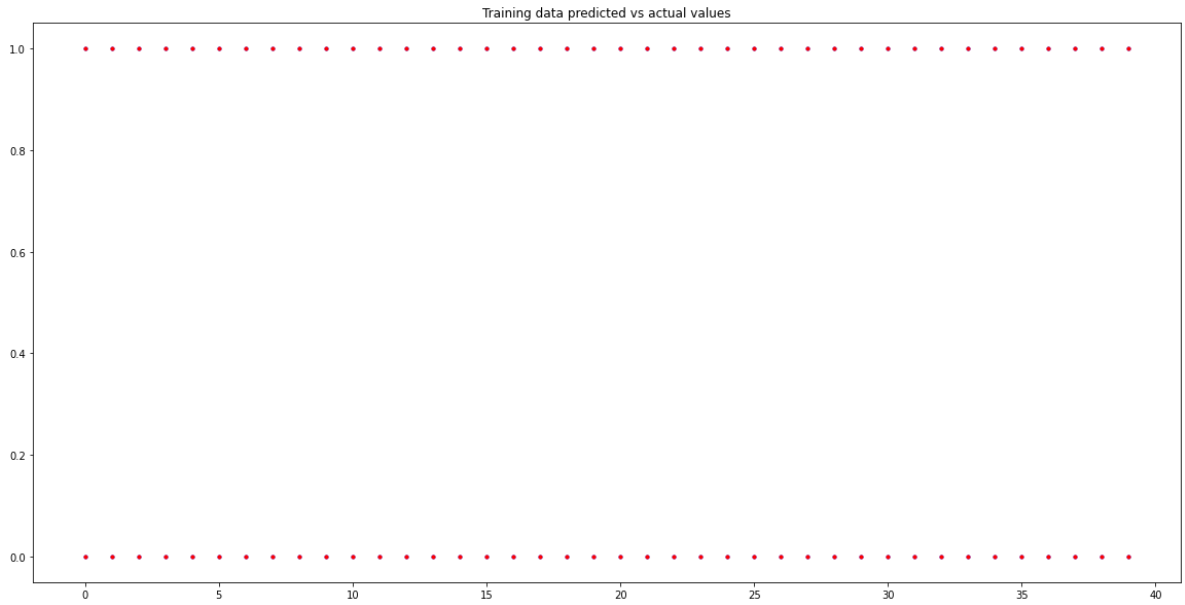


In [62]:

```
#printing predictions for optimised model
predictions = model_optimised.predict(inputs_test)
print("predictions =\n", np.round(predictions, decimals=1))
print("actual =\n", outputs_test)
plt.clf()
plt.title('Training data predicted vs actual values')
plt.plot( outputs_test, 'b.', label='Actual')
plt.plot( predictions, 'r.', label='Predicted')
plt.show()
```

```
predictions =
 [[0. 1.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [0. 1.]
 [1. 0.]
 [0. 1.]
 [0. 1.]
 [1. 0.]
 [0. 1.]
 [0. 1.]
 [1. 0.]
 [1. 0.]
 [0. 1.]
 [0. 1.]
 [0. 1.]
 [0. 1.]
 [0. 1.]
 [0. 1.]
 [0. 1.]
 [0. 1.]
 [1. 0.]
 [0. 1.]
 [0. 1.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [0. 1.]
 [1. 0.]
 [0. 1.]
 [1. 0.]
 [1. 0.]
 [0. 1.]
 [0. 1.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [0. 1.]]
actual =
 [[0. 1.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [0. 1.]
```

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

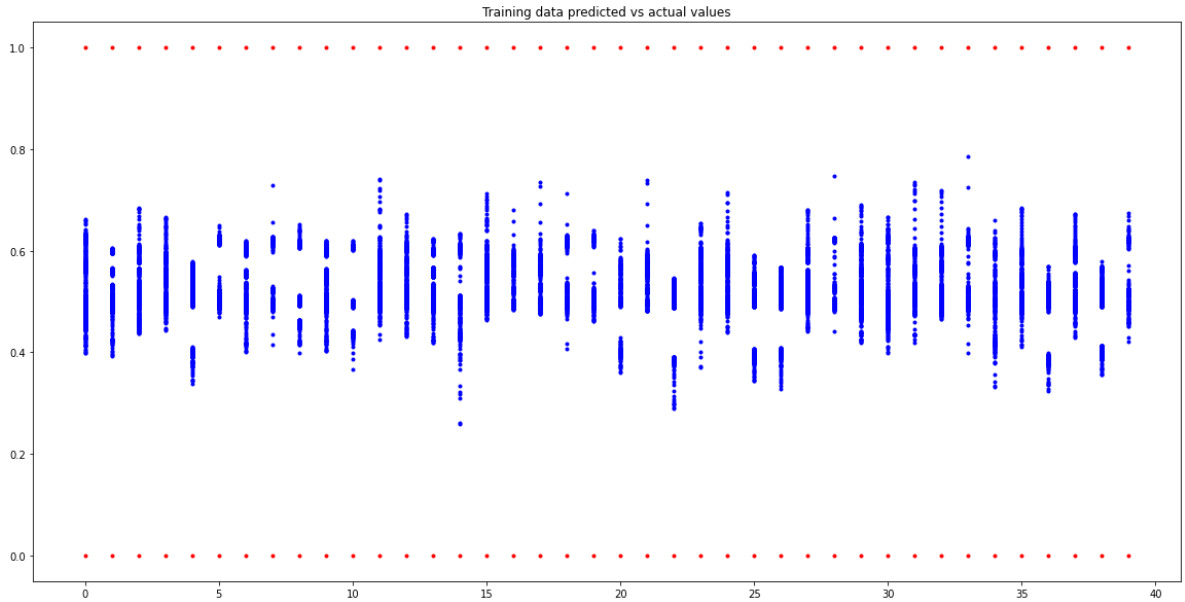


In [63]:

```
#printing presictions for non-optimised model
predictions = model_non200.predict(inputs_test)
print("predictions =\n", np.round(predictions, decimals=1))
print("actual =\n", outputs_test)
plt.clf()
plt.title('Training data predicted vs actual values')
plt.plot(inputs_test, 'b.', label='Actual')
plt.plot(outputs_test, 'r.', label='Predicted')
plt.show()
```

```
predictions =
 [[0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]]
actual =
 [[0. 1.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [0. 1.]
```

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



In [54]:

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

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

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

```
INFO:tensorflow:Assets written to: C:\Users\ronan\AppData\Local\Temp\tmpbiy3
xfbq\assets

WARNING:absl:Buffer deduplication procedure will be skipped when flatbuffer
library is not properly loaded

Model is 60412 bytes
```

In [55]:

```
# Convert the non-optimised model to the TensorFlow Lite format without quantization
converter = tf.lite.TFLiteConverter.from_keras_model(model_non200)
tflite_model = converter.convert()

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

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

```
INFO:tensorflow:Assets written to: C:\Users\ronan\AppData\Local\Temp\tmpi_hx
huly\assets

INFO:tensorflow:Assets written to: C:\Users\ronan\AppData\Local\Temp\tmpi_hx
huly\assets
WARNING:absl:Buffer deduplication procedure will be skipped when flatbuffer
library is not properly loaded

Model is 60444 bytes
```

In [ ]:

```
#create optimised model
!echo "const unsigned char model[] = {" > htmvsnonmodel.h
!cat htmvsnon.tflite  | gvim82   >> htmvsnonmodel.h
!echo "};"                              >> htmvsnonmodel.h



import os
htmvsnonmodel_h_size = os.path.getsize("htmvsnonmodel.h")
print(f"Header file, htmvsnonmodel.h, is {htmvsnonmodel_h_size:,} bytes.")
```

In [ ]:

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

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