

Gym Equipment Image Detection and Classification App

Using Deep Convolutional Neural Networks

Chukwudi Onyema Ajoku



A Report of my Advanced Practice, Submitted in Partial
Fulfilment of the Requirement for the Degree of
Master of Science

Supervised by: Dr. Tariq Alwada'n



Applied Artificial Intelligence
Teesside University
Middlesbrough, England, United Kingdom
17th December 2021

Contents

1	Introduction	2
1.1	Motivation	2
1.2	Goals	2
2	Activities Undertaken	2
2.1	Machine Learning	2
2.1.1	Data Collection	2
2.1.2	Data Preparation	3
2.1.3	The Deep Learning model	4
2.1.4	Training, Validation and Testing	4
2.1.5	Augmenting the Model	5
	Things I did differently.	5
2.1.6	Results	5
2.2	Software Development	7
2.2.1	Application Flow:	7
3	Contribution to the Organisation	8
4	Reflections	8
4.1	What did I Learn?	8
4.2	What went well?	8
4.3	Challenges	8
4.3.1	Data Collection	9
4.3.2	Learning a new technology	9
4.3.3	GPU Availability	9
4.3.4	Reproducibility and Interoperability	9
4.3.5	Making the model smarter; The Open Set Recognition Problem	9
4.4	What I could have done better	10
4.5	Long-term implications	11
5	Personal Benefits	11
6	Professional Values and Behaviour	11
7	Appendices	12

1 Introduction

Having added some extra weight due to little or no activity during the lock-down, I became interested in bodybuilding and healthy living and, therefore, registered in the gym. Sadly, I could not afford a gym instructor and did not know how to use some of the equipment in the gym.

What if I researched all the gym equipment, the muscle group they trained and perhaps a link to a video showing how each piece of equipment is used? What if I can achieve this using Computer vision (Deep Learning) by developing an app where a user takes a picture of the equipment and makes a prediction through image recognition? This was where my story began.

1.1 Motivation

Since the beginning of this course, I have only concerned myself with machine learning techniques and procedures. I could interact with the codes, clean and prepare data, build a model, fit and make a prediction but have not been able to apply it in the form of software with which non-programmers can intuitively interact in the real world.

I have taken an interest in software development in the last few months. This project will sharpen my skills further, It still goes a long way to help me practically learn software development hands-on out of necessity.

Image recognition is one of the most valuable inventions in machine learning. Having been taught and having worked on several image datasets, I felt it would be nice to go out and fetch image data by myself and carry out the process from beginning till the end.

1.2 Goals

The final goal of this project is to have software that takes an image of gym equipment and predicts what type of equipment it is and how it is used. If my software development skills permit me, gym owners should be able to add more data by themselves if they buy new equipment that is not part of the trained data. This will require some transfer learning techniques. With this functionality, the app will become more versatile and can be used on any gym equipment of any colour and type anywhere in the world.

2 Activities Undertaken

The method used here consists of different procedures which include the following parts; **Machine Learning**, **Using the prediction to display something to the user** and **App Development**. The code used can be found on github: https://github.com/Chuukwudi/Gym_ER

2.1 Machine Learning

This part consists of Data Collection, Data Preparation, The Deep Learning model, Training, Validation and Testing/Making a prediction Augmenting the model and repeating until the best results are achieved.

2.1.1 Data Collection

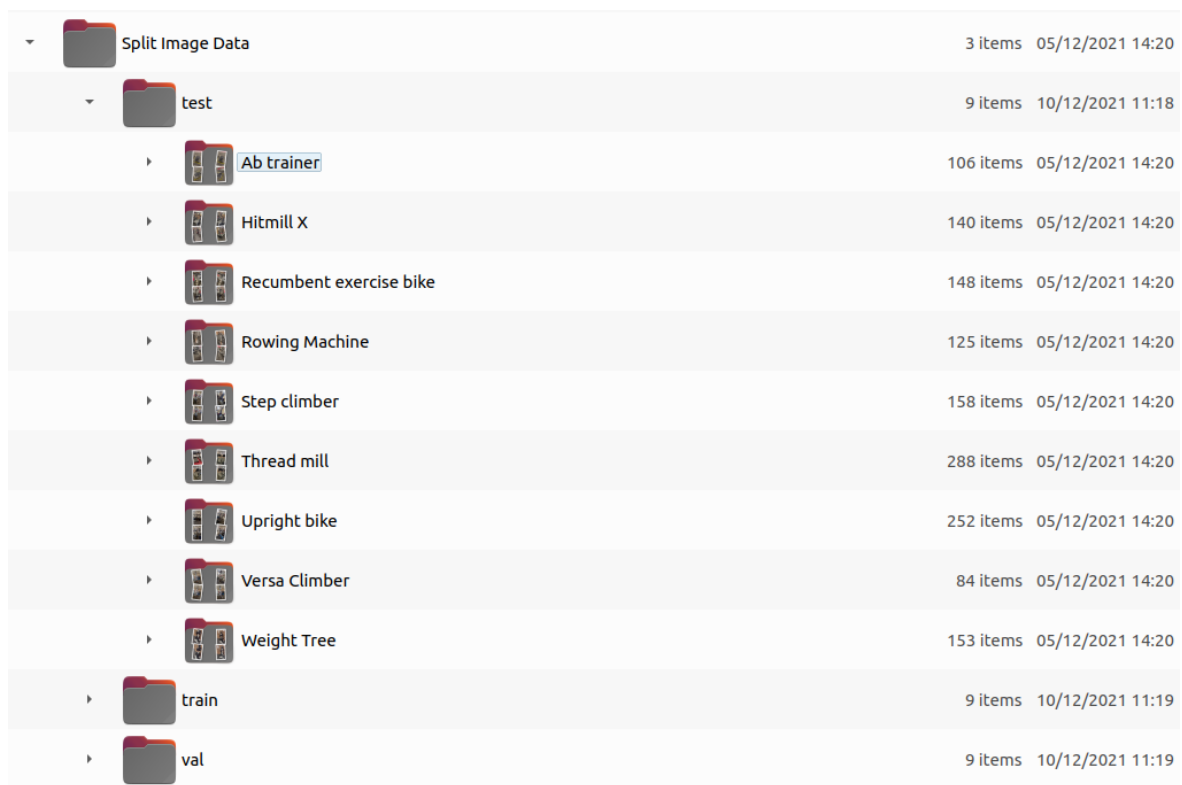
To achieve my purpose, I needed gym equipment images to train my model. I visited the gym, took several pictures of each piece of equipment, and put them in different folders. However, this method did not produce the best results. To achieve a better performance, I needed more images to train my model. Rather than capturing the image, I walked around each piece of equipment and took 360° videos of it. Each video had a resolution of 1280x720 and lasted between 30 and 60 seconds.

2.1.2 Data Preparation

I put the videos in hierarchical folders, Using the OpenCV-Python package, I was able to convert each video into picture frames. The major advantages of this method is that:

1. I have a 360° coverage of each equipment.
2. I can get more image data by increasing the amount of image frames to be captured per second.
3. Every image in my dataset is technically unique.

After the conversion has been done, I used another library called ‘split-folders’ to split the data into training, validation and testing datasets. This library helps maintain the folders in their correct hierarchy and organisation in a way that is suitable for machine learning.



▼	Split Image Data	3 items	05/12/2021 14:20
▼	test	9 items	10/12/2021 11:18
▶	Ab trainer	106 items	05/12/2021 14:20
▶	Hitmill X	140 items	05/12/2021 14:20
▶	Recumbent exercise bike	148 items	05/12/2021 14:20
▶	Rowing Machine	125 items	05/12/2021 14:20
▶	Step climber	158 items	05/12/2021 14:20
▶	Thread mill	288 items	05/12/2021 14:20
▶	Upright bike	252 items	05/12/2021 14:20
▶	Versa Climber	84 items	05/12/2021 14:20
▶	Weight Tree	153 items	05/12/2021 14:20
▶	train	9 items	10/12/2021 11:19
▶	val	9 items	10/12/2021 11:19

FIGURE 1

Image showing the desired hierarchy of the files in a way that is suitable for machine learning.

2.1.3 The Deep Learning model

To build the model, I used the Keras on top of TensorFlow to create my model. Image training for deep learning applications usually require Convolutional Neural Networks. My network has 5 convolutional layers with max pooling and 2 dense layers. The structure can be seen in the image on the right 🐼.

2.1.4 Training, Validation and Testing

The entire dataset consists of 7,244 images of 1.4GiB memory belonging to 9 different classes viz: Recumbent exercise bike, Versa Climber, Rowing Machine, Weight Tree, Ab trainer, Step climber, Upright bike, Hitmill X and Thread mill. This was finally split into 3 in the ratio of 60%, 20% and 20% belonging to the training, testing and validation sets respectively.

The distribution of this data can be seen in the table below.

Classes	Training	Validation	Test
Ab trainer	318	106	106
Hitmill X	416	138	140
Recumbent Exercise Bike	442	147	148
Rowing Machine	373	124	125
Step Climber	474	158	158
Tread Mill	861	287	288
Upright Bike	752	250	252
Versa Climber	252	84	84
Weight Tree	456	152	153
Total	4344	1446	1454

Although the classes are not balanced, the results from the final model have shown that the data provided for training is more than sufficient for the model to learn the problem.

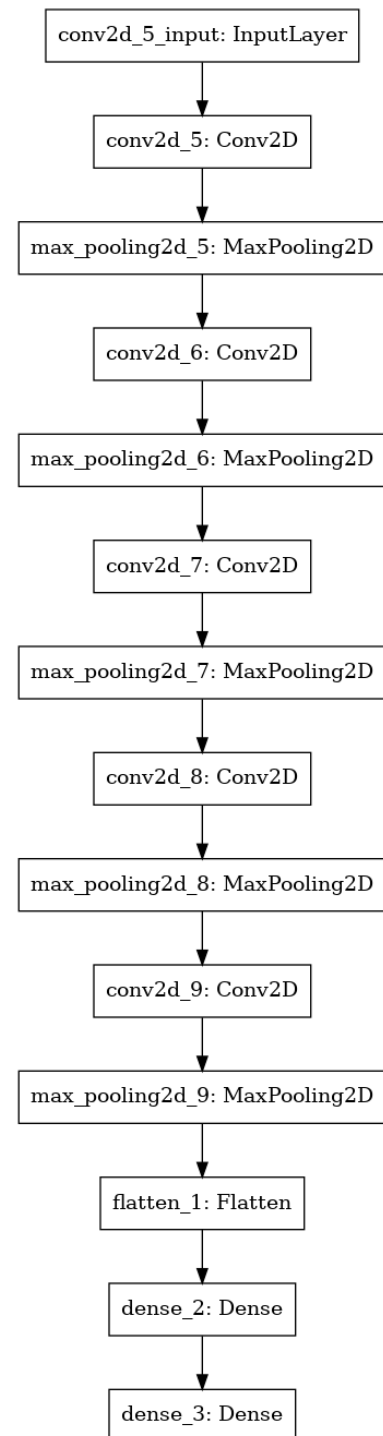


FIGURE 2
My Deep Learning model

2.1.5 Augmenting the Model

The model described in fig. 2 took efforts and several modifications. Initially, my model was not predicting very satisfactorily. The performance of the initial model can be seen below:

	precision	recall	f1-score	support
Recumbent exercise bike	0.54	0.70	0.61	148
Versa Climber	0.42	0.42	0.42	84
Rowing Machine	0.48	0.65	0.55	125
Weight Tree	0.76	0.67	0.71	153
Ab trainer	0.63	0.52	0.57	106
Step climber	0.73	0.68	0.71	158
Upright bike	0.86	0.80	0.83	252
Hitmill X	0.84	0.62	0.72	140
Thread mill	0.79	0.84	0.81	288
accuracy			0.70	1454
macro avg	0.67	0.65	0.66	1454
weighted avg	0.71	0.70	0.70	1454

FIGURE 3
Evaluating the initial model on test data

Things I did differently.

In order to get a better prediction, I had to reconstruct the model. I took the following steps:

1. **Data Collection:**

I took 360° videos of each equipment and cut it into images/frames rather than manually taking different pictures of the equipment myself from different angles.

2. **Changed the input resolution.**

The input resolution became 512×288 as opposed to the initial 200×200 . This increase in dimension in turn increased the number of trainable weights in my model. As a result of this, my model performed better.

3. **Changed the frame rate.**

During data processing, I changed the frame rate from 0.5 to 0.03 which increased the size of my dataset by 17 times.

2.1.6 Results

The performance of the model can be seen in the images below.

Applying the algorithm on the test set yielded 100% accuracy. This is as a result of having so much training data, very few classes and not too many variables. It is very unusual but was well expected. The performance of the data on test set can be seen below.

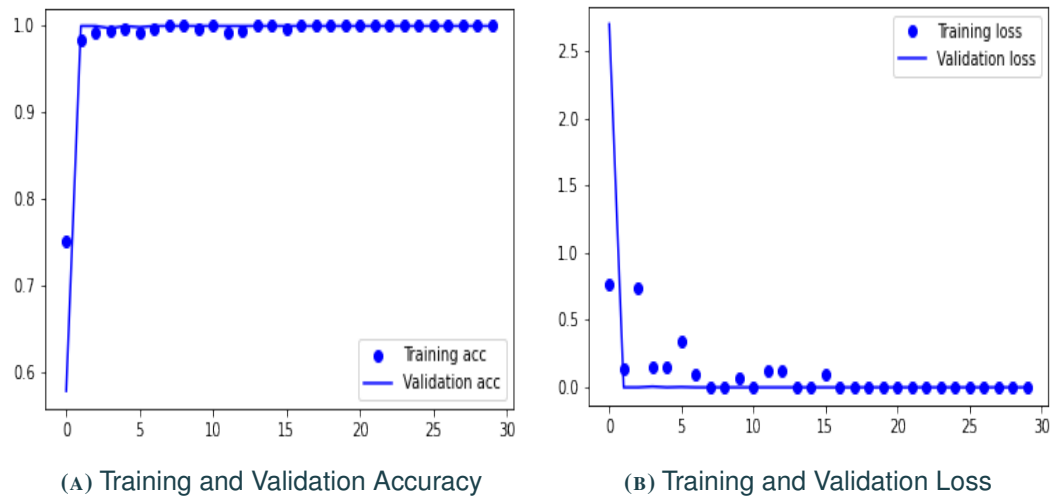


FIGURE 4
Results of the Final Model

	precision	recall	f1-score	support
Recumbent exercise bike	1.00	1.00	1.00	148
Versa Climber	1.00	1.00	1.00	84
Rowing Machine	1.00	1.00	1.00	125
Weight Tree	1.00	1.00	1.00	153
Ab trainer	1.00	1.00	1.00	106
Step climber	1.00	1.00	1.00	158
Upright bike	1.00	1.00	1.00	252
Hitmill X	1.00	1.00	1.00	140
Thread mill	1.00	1.00	1.00	288
accuracy			1.00	1454
macro avg	1.00	1.00	1.00	1454
weighted avg	1.00	1.00	1.00	1454

FIGURE 5
Evaluating the final model on test data

[[148	0	0	0	0	0	0	0	0]
[0	84	0	0	0	0	0	0	0]
[0	0	125	0	0	0	0	0	0]
[0	0	0	153	0	0	0	0	0]
[0	0	0	0	106	0	0	0	0]
[0	0	0	0	0	158	0	0	0]
[0	0	0	0	0	0	252	0	0]
[0	0	0	0	0	0	0	140	0]
[0	0	0	0	0	0	0	0	288]]

FIGURE 6
Confusion Matrix

2.2 Software Development

The application was designed and developed using the Python Flask web framework. I have chosen the FLASK framework because it is a micro-framework; it does not have many software dependencies and can be quickly developed and deployed. The web application is simple to install, maintain and is highly reliable.

The machine learning model, which was previously trained and tested with captured data, is deployed into the framework. The model is programmatically accessed to provide the user with the requested information about the gym equipment without the hassle of looking for a gym instructor or searching for manuals. The complete high-level architecture is presented in fig. 7

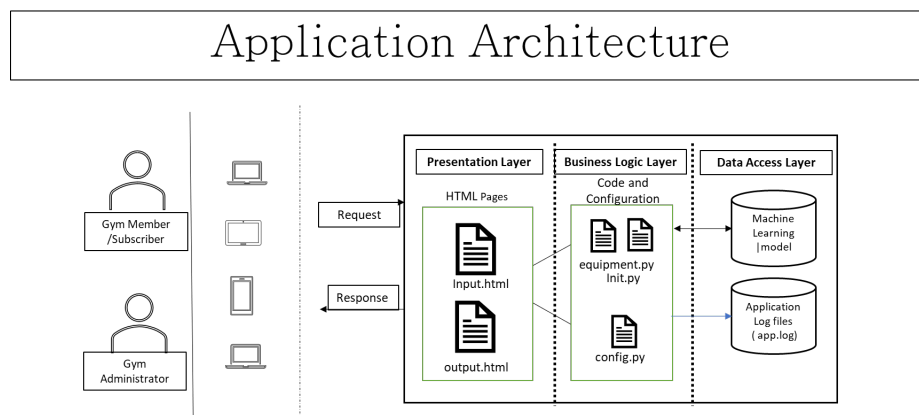


FIGURE 7
Application Architecture

The application is owned and deployed by the Gym administrator. An administrator deploys it, maintains and updates the information of gym equipment - The user who is subscribed member of the gym will be able to access the deployed web services.

2.2.1 Application Flow:

The Gym app follows a traditional flow of requests and responses like any typical web app. The user wants to know some information about a piece of equipment, so he takes a picture and uploads it using *input.html* page (which is the default index page); then this request is forwarded to *equipment.py* which processes the image and feeds the image to the model and gets the model prediction. The model prediction is then mapped with the equipment information in *equipment_info.txt* file and sends this mapped info to *output.html*. This *output.html* is populated with the predicted info and presents the response to the user. If an image is not from the gym, ideally, the response says "Equipment does not belong to our gym".

3 Contribution to the Organisation

Before gathering the data used for this project, permission was granted by the gym owner, who has also shown keen interest in this project from the beginning till this point. According to him, this software, if commercialised, can be marketed and utilised by other gym owners as well, and with time, more improvements can be added.

It is also possible to use the model for other applications where image recognition and classification is needed. One would then have to gather data for any input category of choice.

In general, the importance of image recognition models to businesses can not be overemphasised. Its impact can be felt in the following areas:

1. Process Management in Businesses
2. E-commerce and Social Media
3. Industrial Automation
4. Surveillance and Security, etc.

4 Reflections

4.1 What did I Learn?

Although this is not my first time building and training machine learning models, it requires some technical knowledge to utilise because everything is just in its code format. An average user would not know how to load my model, process the input data, and make a prediction with my model. For this reason, I created a web application GUI to enable users with no programming experience to interact with my Deep Learning model.

Recently, I started learning basic HTML and CSS. Venturing into this project allowed me to practice what I have learnt so far, including solidifying my knowledge about the concepts of Convolutional Neural Networks.

4.2 What went well?

In general, the Machine Learning problem had a minimal scope. For instance, most of the equipment of the same type in the gym were from the same manufacturer and looked the same. The only variation between two images of the same category is the angle at which the picture was taken, the camera settings and environmental factors such as ambience and exposure. For this reason, it was straightforward for my model to learn correctly because the images have a very limited scope of possibilities by which they can be known.

On the contrary, if I was distinguishing between classes with a wider range of possibilities and appearance, for instance, classifying between a dog and a cat, two dogs belonging to the same category could be of different colours, different sizes, different species, take different postures whereas, in the case of gym equipment, the equipment is stationary and nothing else changes.

4.3 Challenges

Some of the challenges I faced include:

4.3.1 Data Collection

While collecting data for this project, it was not easy getting videos of the equipment as not many gym goers would be comfortable with a stranger pointing his camera towards them. Secondly, I had to choose days and times when the gym was least busy and take the videos only when the equipment was unoccupied.

4.3.2 Learning a new technology

Using jinja2, a web template engine for python was completely new to me and only heard of it for the first time in the course of carrying out this project.

4.3.3 GPU Availability

Neural networks generally require a Graphical Processing Unit (GPU) to speed up computation time; otherwise, it would take ages to train. To represent a graphical image as a tensor, each pixel that makes up the image, including the colour of that pixel, is altogether represented as a tensor. An image with dimension 512×288 will have 147456 pixels. Each of these pixels also has its colour represented by another array of three values(RGB), showing the amount of Red, Green and Blue that form that colour. In the end, the dimension becomes $(512 \times 288 \times 3)$.

For a dataset comprising of thousands of images, handling this amount of data is computationally expensive.

4.3.4 Reproducibility and Interoperability

To my disappointment, I observed that while my model predicts perfectly well on my machine, this was not the case when the same model was loaded on a different machine. An issue has been previously raised on the official github repository of the Keras' python package on <https://github.com/keras-team/keras/issues/7676> and I have also asked a question about this on stackoverflow <https://stackoverflow.com/questions/70296860/saved-model-predicts-wrongly-on-a-different-machine-but-works-perfectly-on-mine> which uptill the moment of writing this is still unanswered.

Personal research on this matter suggests that GPU enabled machines are more precise than other machines running only on CPU. Even though the errors may be small, it adds up and explodes when bigger and more complex calculations are involved.

4.3.5 Making the model smarter; The Open Set Recognition Problem

Although the machine learning model was accurate, I kept getting predictions, most times with over 90% certainty, when random images from the outside space were passed into my model. It did not matter if the image was gym equipment. The model only predicted the images of any of the equipment from the gym correctly. It sometimes did not perform well when images of equipment from a different gym were passed in. This is because the model was not trained to identify other images except the ones it was trained with.

I needed a way to output information on the screen in the event that a wrong picture had been passed into the model. An error message in the lines of *"Sorry, please, use an image from the gym, not just a random image"* would be perfect.

The machine learning model only outputs a list of probabilities which sums up to 1. The class that has the highest probability is the predicted class.

Usually, one would expect that with a strange image from the outside scope, the output probability of the model should reflect some uncertainty, but it did not. For instance, if $[0.200, 0.700, 0.100]$ is the output of

a model trained to predict three classes, one would be confident that the predicted class is the one in the middle 0.700 but if the output were to be $[0.400, 0.399, 0.210]$, although the highest probability is the first one 0.400, it is not too different from the second 0.399. This means that in reality, the predicted image has some chances of belonging to the second class, so some uncertainty is reflected.

It is not possible to train a classifier on every class imaginable. It will always run into some other class that it is unfamiliar with and hasn't already seen before.

Some of the solutions I thought about were:

1. **Using a separate binary classifier**

This involves creating a binary classifier that can distinguish between images from our gym versus images from anywhere else. Furthermore, only when the image is from our gym will we pass it over to the main classification model.

One disadvantage of doing this is that the application becomes less responsive as more time is taken to pass one image through two different models. Secondly, more random images and all the ones I already have will be needed to train this binary classifier. This is computationally expensive, with plenty of effort for so little gain.

2. **Create an extra class: Unknown**

This is similar to the solution discussed above, except that instead of a separate binary classifier, an extra class is created, which would consist of lots of random images.

This solution is not optimal because of the same reason discussed above.

3. **Thresholding**

This is the solution I used. Although not optimum, it still works in most cases. When the image passed in is something else (ex. a car), the probabilities are expected to be about the same for all classes. In other words, the probability for neither of the classes stand out, but this is not the case. To resolve this, I set a threshold at 1×10^{-9} below 1. This way, if a model is too sure, it has to be close to 1 or almost 1. The further away a prediction is from 1, the higher the tendency that the prediction is not from an actual image from our gym.

I designed it this way because the model was predicting with high probability, even on an image from outer space.

4.4 What I could have done better

There are a number of things which I could not do due to time constraints. They are:

1. **Adding more classes**

There were more than 9 classes of equipment in the gym. I could have collected image data for more.

2. **Training with images from more than one gym**

This will make my model applicable to more than one gym. It will be able to generalise better and become more relevant to a larger group of subscribers.

3. **Taking images under different lighting conditions**

This guarantees that the model predicts well despite the quality and clarity of the images passed in.

4. **Making a mobile application interface or hosting the web app**

Presently, this web app only runs local. It would be better to deploy it as a mobile or web application. This way, it becomes more portable.

4.5 Long-term implications

As with all software technologies, over time, new requests come in, users will want more features and functionalities. There will be need for adequate maintenance from time to time.

5 Personal Benefits

There are lots of benefits attached to the advanced practice. For instance, it has enhanced my qualification by adding some practical experience to give my CV a competitive edge. It also has developed my analytical and thinking skills.

More particularly, this advanced practice has helped me in these areas:

1. **New Knowledge**

At first, when this idea was born, I had no idea how to actualise it until I started. Apart from the deep learning part of the project, the challenges were very tough, but I learned about the flask package, the jinja template, and more HTML and CSS. I now also know the usage of gym equipment because I had to research them.

2. **Networking**

I met many interesting people through this project, such as the CEO and many others.

3. **More Confidence**

More importantly, this advanced practice experience has consolidated my course knowledge.

6 Professional Values and Behaviour

I have learned and imbibed certain professional values and behaviours through advanced practice. for instance:

1. **Self Motivation:**

It is essential to be self-motivated in a professional setting. Without being self-motivated, I would not have been able to drive myself past the challenges, setbacks and mistakes I made. However, it was gratifying to finally fix the bugs after several hours of pondering and attempting different things. Most of the time, I had to work on my own and would not have made it through without being self-motivated.

2. **Willingness to learn more:**

This is a crucial factor for everyone in the software-related field. Without the willingness to learn more and grow, it is very easy to be outdated. In my advanced practice, through persistence and perseverance, I was able to learn new technologies and techniques.

3. **Time management:**

When undertaking a project and managing deadlines, planning and setting smaller goals with sooner deadlines is vital. I did this weekly as I had to record and report to my project supervisor every week.

4. **Communication:**

Coming from a very technical field, conveying what one is doing to a non-specialist is an essential skill that is often not emphasised enough. In my internship, I realised that I had this problem and started working on it.

5. **Confidence and Proper Conduct:** Having undergone this process, I am more confident in discussing my subject and carrying out my duties.

7 Appendices

The codes used in this project can be found in https://github.com/Chuukwudi/Gym_ER. I will be very happy to demonstrate what I have done in person when called upon.