ELEC 490/498 Final Report

Weight Training Calorie Counter

CALORIE COUNTER

Group #24

Group Members: Doug Byers, Justin Carr, Niko Hoogeveen, Ryan Steiner

Submitted April 11th, 2023

To: Dr. Alireza Bakhshai, P. Eng., Dr. Mike Korenberg, P. Eng., Dr. Alex Tait, Dr. Sean Whitehall

Dr. Il-Min Kim, P. Eng.

Executive Summary

The purpose of this final report is to provide a summarizing overview of all the work put into the capstone project over the past 6 months by the group, from project selection to implementation. The intended audience for this report is all the Course Instructors, our Faculty Supervisor Dr. Il-Min Kim, and our TA Laura Connelly.

The Calorie Counter is a web-based application that is built upon a convolutional neural network (CNN) model for the purpose of classifying the exercise being performed in videos and estimating the number of calories burned. The project's motivation stems from interest within the group and the increasing interest in the personal fitness market. This project aims to address the growing need for accurate and efficient tracking of workout progress by combining advanced neural network technology and user profiling.

The design process first involved collecting and labeling a dataset of thousands of images that demonstrate a specified set of exercises, bench press, squat, and deadlift. Through an iterative process involving the steps of collecting additional data, tuning hyperparameters, and retraining the CNN model, the prediction accuracy continually improved until it surpassed the threshold determined in the blueprint report. During this training process, the web-based application was being developed according to the specifications from the blueprint report, taking in the necessary user inputs and displaying the necessary outputs for the user.

The project underwent extensive testing and evaluation to ensure its accuracy and reliability of predicting both the type of exercise and number of calories burned. The model's predictions were compared with that of an Apple Watch, which was used as the gold standard of performance.

Overall, The Calorie Counter offers a novel solution to the problem of accurately monitoring exercise routines and calorie expenditure. The group recommends further development and refinement of the dataset along with other factors discussed in the report in order to improve the model's accuracy and expand its functionality beyond its current capabilities.

Table of Contents

Table of Tables

Table of Figures

1. Project Background

The goal of the project was to develop a neural network model that can estimate the number of calories burned by an individual while performing an exercise, based on a video recording of the person performing the exercise. Currently, no technology exists on the market to perform calorie estimation without access to a person's heart rate while they are performing an exercise.

To achieve this goal, Group 24 explored existing deep learning models and decided to take a video-classification approach. The decision was made to implement a video classification network to determine the duration of exercise being performed in input videos, and pair that with user input of their own body metrics, height, weight, age, and sex, to calculate an accurate calorie estimation. To create this network, a CNN was trained to perform image classification. At runtime, the model splits up the input videos into individual frames, runs a prediction on each frame, and takes the rolling average of the past frames to generate a classification of what exercise the user is performing in the video.

The approach taken was based from another video-classification network coded in Python, which uses a base ResNet50 CNN [1]. Their implementation was for classifying videos of people participating in various sports and exercises including weightlifting. As a result, Group 24 decided to use the ResNet50 CNN as the base network and referenced the head model used in the Keras model for implementation. For more information on why a CNN was specifically chosen for this video classification task, refer to the section [3.3 Choosing a Network]. Additionally, an entirely unique dataset was generated for this task. The goal was to train the network to recognize, and differentiate between people performing bench-press, squats, and deadlifts. There are no public datasets that had the right images to train the network on, so a custom dataset was generated by taking thousands of images from online sources.

Once the model was trained, a web interface was created for users to access the model and allow them to run calorie predictions on videos they choose to input.

In summary, the project aimed to fill a gap in the market by developing a neural network model that can accurately estimate the number of calories burned during exercise, based on video recordings of the individual performing the exercise.

2. Motivation

Albeit simple, the motivation to choose this project stemmed from personal interest in the field of fitness within the group. Two members of the group actively workout and track their progress through metrics such as calories burnt and for this reason, interest grew in wanting to develop a system that would take this to the next level.

This project was additionally compelling because of its potential to make a significant impact in the field of fitness and health. With the increasing interest in personal fitness, there is a growing need for accurate and efficient means of tracking workout progress and calorie expenditure [3]. This project aims to provide a solution that can address this need by combining advanced neural network technology with user profiling to produce accurate and personalized predictions of calorie burn during weight training workouts.

From a technological perspective, this project is highly interesting due to the complexity of the problem it aims to solve. Developing a system that can accurately identify different types of workouts and predict calorie expenditure requires sophisticated machine learning algorithms and high-performance computing capabilities. Group 24 was excited to explore and implement the latest techniques in neural network design and optimization to create a system that could deliver reliable and personalized results. Additionally, the project's potential to utilize existing video data sources and integrate with other fitness tracking applications makes it a highly relevant and practical application of advanced technology. With recent studies showing the incredible heights the smartwatch chips global market can grow to over the next 5-10 years, technologically the developed product would share a similar valuation [4].

3. Design Process

The following sections outline the specific details of how the project was designed. Functional requirements are described, as well as the development strategy and process for the project.

3.1 Functional requirements

Defining functional requirements for the project was essential in determining the features that needed to be implemented to meet project objectives. Three main functional requirements were created and adhered to over the course of the project.

Firstly, the project needed to include a network that could perform image classification. The network must be capable of identifying and categorizing four different classes of images, bench press, squat, deadlift, and none classification. The final use of the network is to classify individual frames of videos in rapid succession and as a result, the network used needed to be efficient.

Secondly, a dataset that is large and diverse was needed to enable the network to learn effectively. Comprehensive image sets of each workout with many different variations, as well as body types and fitness levels needed to be collected to ensure the network would perform adequately. The dataset also needed to include many images of people not performing one of the three chosen exercises, bench-press, squat, or deadlift, to enable the network to accurately predict the duration of exercise being performed in videos.

Finally, the project needed a comprehensive user interface where the model could be accessed for on demand inferencing and calorie prediction. User-friendliness and ease of use were paramount in the design thinking process of the GUI. The goal of the GUI was to provide accurate calorie predictions based on the user's video input as well as input body metrics, giving them valuable insights into their workout performance.

In summary, the three main functional requirements of the project were to create an image-classification network, a dataset that is adequately large and diverse, and a GUI where the model can be accessed by the user for inferencing and calorie prediction on inputted videos.

3.2 Development Strategy

During the extent of development, Group 24 used Agile development processes to ensure all functional requirements were met and the project followed the timeline made in the blueprint. To begin development, a cross-functional team needed to be created to allow for parallel development of the main project tasks. These tasks were determined to be the neural network developer, the GUI developer, database developer, and integration coordinator. After developing roles, a product backlog was created to identify the specific features needed to achieve the project's goals and order them with respect to project importance, as seen in [Table 1].

Table 1: Backlog of tasks to be done, created at the beginning of the development process

Need	Tasks to Achieve Need
A neural network that classifies the workout in the video.	Task 1.0: Research and develop an appropriate CNN architecture for the workout classification. Task 1.1: Implement the CNN architecture. Task 1.2: Train the CNN model using a labeled dataset of workout images. Task 1.3: Evaluate the trained CNN model on the test set and optimize the performance.
A dataset that covers all classifications, edge cases, and environments. Ensure the dataset is vast enough to not be biased but also not underfit.	Task 1.0: Collect the dataset, ensuring it covers the three workout labels and the none label. Task 1.1: Preprocess the collected videos, including resizing, normalization, and frame extraction. Task 1.2: Label the images to their true label.
A GUI that can take video uploads and run inference using the backend.	Task 1.0: Design and implement video upload user interface. Task 1.1: Implement file selection. Task 1.2: Test the features of this page.
To process uploaded videos and extract data.	Task 1.0: Develop module to preprocess the video by resizing, normalizing, and conducting frame extraction. Task 1.1: Test the preprocessing module.

Integrate the CNN with the GUI	Task 1.0: Integrate the CNN into the backend and test the integration. Task 1.1: Test the integration.
To estimate the calories burnt during the workout	Task 1.0: Research and develop an algorithm to estimate the calories burnt during the workout. Task 1.1: Integrate the calculation into the backend system. Task 1.2: Test the calorie prediction.
To show the user the classification and calorie estimations	Task 1.0: Design a page to show the user the results taken from the backend. Task 1.1: Develop the html page. Task 1.2: Develop database to save previous classifications under a user id. Task 1.2: Test the performance of the page.
The GUI to be user friendly	Task 1.0: Develop the full front end. Task 1.1: Implement UI and UX techniques Task 1.2: Test the appeal and user experience of the developed front end.
To handle errors efficiently	Task 1.0: Implement an error handling system that covers video uploads, preprocessing, GUI display, and database management.

After creating a backlog, members picked tasks and would work individually and collaboratively on their tasks, updating the team during weekly meetings. This was done instead of developing individual sprints considering the team went through several stages of revision and large tasks would take much longer than 2 weeks, especially considering the external workload of each member. During the timeline Group 24 tested and developed incrementally to ensure adjustments/updates/changes could be made to improve the project.

3.3 Choosing a Network

For this project, Group 24 chose to use a Convolutional Neural Network (CNN) for classifying the workout videos because a CNN is very well-suited for image-based classification tasks considering they are based on the structure of the human visual cortex and for the following reasons.

1. Hierarchical feature learning: CNNs can automatically learn and extract hierarchical features from images without manual feature engineering. The lower layers learn basic features, such as

edges and textures, while the deeper layers learn more complex features, such as object parts or entire objects. This hierarchical feature learning allows CNNs to capture patterns and structures at different levels of abstraction, making them highly effective for image classification tasks [4].

2. Translation invariance: CNNs utilize convolutional layers that apply filters to local regions of the input image, enabling them to learn translation-invariant features. This means that the network can recognize a feature or pattern regardless of its location in the image, which is a crucial property for image classification tasks where the position of the objects may vary across different images, such as workout classification [5].
3. Parameter sharing: In convolutional layers filters or kernels are applied to regions of input frame. This reduces the parameters in the network compared to fully connected layers. This parameter sharing helps improve the efficiency of the network, reduces the risk of overfitting, and lowers the computational cost [6].
4. Robustness to variations: CNNs are generally more robust to variations in the input data, such as changes in lighting, viewpoint, scale, and occlusion, making them better suited for real-world image classification. This is essential for the project considering the dataset was personally created [4].
5. State-of-the-art performance: CNNs have consistently demonstrated state-of-the-art performance on a wide range of image classification tasks, including object recognition, scene classification, and facial recognition, making them a popular choice for image-based projects [1].

After deciding to use a CNN, Group 24 decided to use the base model ResNet50 which has 50 layers and has been trained on the ImageNet dataset. ResNet50 was chosen because of its performance, residual connections, transfer learning, computational efficiency, and public support. Other considered networks include VGG, Inception, and EfficientNet. The considerations included in choosing between the models can be seen in [Table 2].

Table 2: Other networks considered

Architecture	Pros Compared to ResNet50	Cons Compared to ResNet50
VGG	- Simpler architecture	- More computationally intensive due to fully connected layers. - Higher memory usage
Inception	- Efficient handling of different scale features with parallel paths.	- Complex architecture.
EfficientNet	- Computationally efficient and high performance. - Scales depth, width, and resolution.	- Requires choice of scaling parameters - New and less community support

3.4 Network Development

After choosing a network and dividing the tasks into a backlog as outlined in the sections [3.3 Choosing a Network] and [3.2 Development Strategy], Group 24 started building the neural network, developing a dataset, developing a GUI, and researching calorie calculation.

During the development of the neural network the main tasks involved developing a custom model to build on top of the base model ResNet50. This is a very essential part of the project for several reasons including tailoring the classes to the project, taking advantage of transfer learning, fine tuning the model layers, and transforming the feature maps into class probabilities. The code for the custom head can be found in the section [Solution Implementation] under the subsection [4.1 Description of Major Code Blocks]. The reasoning process for each part of the custom head can be seen in [Table 3]. Additionally, as seen in [Figure 1], another main part of the development process for the neural network involved fine tuning the hyperparameters to improve performance. One of the main breakthroughs when fine tuning the network was reducing overfitting, see the section [Testing/Evaluation/Verification/Validation of Project] for more information. This involved using dropout techniques along with L2 regularization. Overall, the development process for the neural network mainly involved developing the custom head on top of the ResNet50 base model and fine tuning the hyperparameters to improve the model's performance.

Table 3: Outline of why the custom head was made the way it is.

Technique	Reasoning
Average Pooling the base model	- Perform dimensionality reduction and make the model more computationally efficient and less prone to overfitting. - Make the model more robust to small variations in the input data. - Perform special summarization within the feature maps to capture features from the input image for use in the fully connected layers. - Reduce the number of parameters in subsequent layers.
Flattening average pooling layer into a one-dimensional vector.	- The fully connected layers require a one-dimensional input.
Dense layer one	- Learn the relationships between the features extracted by ResNet50 and the output classes.
PARAM_INPUTS for dense layer 1	- This is the number of neurons in the first dense layer. - The capacity of the layer to learn complex relationships. - Depends on the problem, therefore this parameter was highly tested to find the optimal value for the network.
L2 Regularization for dense layer 1	- Regularization was added to prevent overfitting and help the model rely less on individual features. - The strength of the regularization was tested to find the optimal value.
RELU activation function for dense layer 1	- Applied to the output of dense layer one. - Creates non-linearity which makes the model able to learn more complex relationships between input and output. - Used RELU because it is efficient compared to other activation functions.
Dropout layer	- Prevent overfitting by adding randomness and reducing reliance on individual neurons. - Improve performance on new data.
Second dense layer	- Create output layer with class probabilities.

3.5 GUI Development

While the neural network was being developed, the GUI was also being developed in parallel. During the development of the user interface there was little struggle. The main design decision that was made through the development of the GUI was the addition of a database to record previous classifications and save them under each user's profile. This addition came at the realization that the front-end lacked user experience and was simply a black box for input and output. To create the database, Group 24 used Flask and SQLite as further explained in the section [4.2 GUI Description]. Furthermore, after the addition of the database, it seemed appropriate to add a page that allowed users to see all their profiles information on past attempts. Additionally, it was decided that the front-end would have proper registration, login, and logout systems to follow the traditional user experience of applications. Lastly, a webcam feature was added to the GUI that performed inference directly from the webcam images to allow direct testing of the back-end CNN.

3.6 Dataset Development

In the beginning of the project, Group 24 planned to find a dataset for the project, so that one did not have to be created manually. Unfortunately, after searching for a suitable dataset online, it was deemed that a personal dataset would have to be created considering most datasets did not only have data for three workout classifications but all workouts. This proved a daunting task as creating a dataset that covers all classifications without overfitting or underfitting, while still having enough data to properly train the model, proved to take an unforeseen amount of time and effort. After scraping the internet for relevant pictures, then scraping YouTube videos, and then taking personal images. Group 24 still found the dataset created to be suboptimal and deemed it would be too much of a commitment to make the dataset perfect. Additionally, it was more important to ensure the network and GUI were functioning perfectly. More information on the trial and errors in relation to the dataset can be seen in the section [3.8 Constraints].

The only design process that changed in relation to the dataset is adding a none-label to the classifications. This was added so that the seconds a user was working out could be counted and used for calorie prediction.

3.7 Calorie Prediction Development

To create the calorie prediction, with enough accuracy to meet the goal with respect to an Apple Watch, Group 24 had to find a calculation that did not rely on the heart rate of the user. It was determined that since the Apple Watch also uses a basal metabolic rate (BMR) for their calculations, it would be ideal to follow the gold standard and do the same. The biggest design decision made in relation to the calorie prediction involved changing the original parameter ‘load used’, which is the weight used by the user in the video, to calculate the intensity, it was decided that an intensity measure would be used that had three options, easy, medium, and hard. This allowed users to determine how hard the exercise was instead of guessing based on their profiles weight and the time they performed the exercise. After the calculations were perfected using human testing data and comparing it with Apple Watch data, as seen in [Table 6], it was implemented in the back end and outputted to the frontend. Calculations can be seen in [Figure 5] using the equations from [Figure 14].

3.8 Constraints

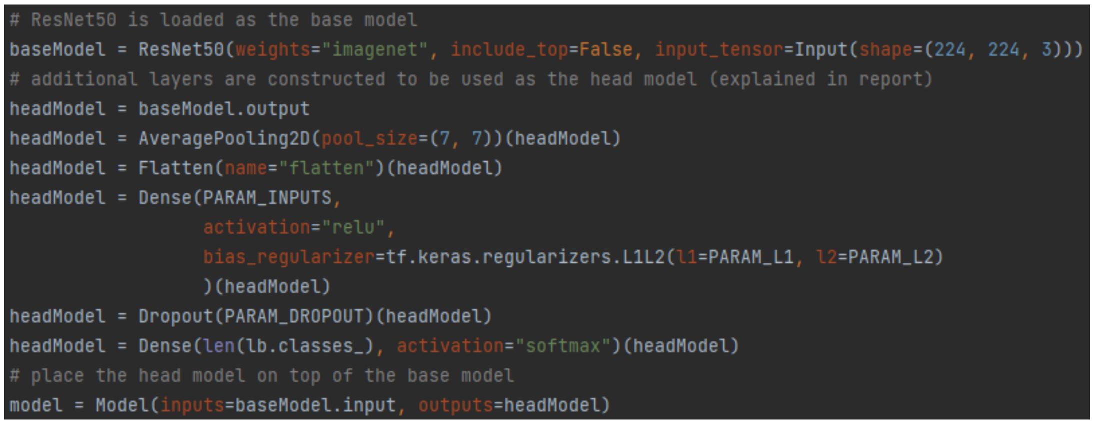
The primary constraint for this project was data collection. Creating a high-quality, diverse dataset is essential, as it directly influences the performance of the trained model. A dataset represents the problem space, and without diversity, the model may not cover edge cases and will perform poorly on untrained 5data. For example, because the initial dataset focused on gym environments, the model's accuracy decreased when classifying workouts in other environments, such as living rooms. Recognizing this constraint, the dataset was expanded to include various environments, reducing the risk of overfitting in the convolutional neural network. A larger dataset allows the model to learn a broader range of patterns and features, making the training data more representative of the true population and leading to better generalization on unseen examples.

However, this expansion initially resulted in less reliable classification of gym images, as the model needed a well-prepared and balanced dataset to recognize features directly associated with exercises rather than environments. To address this issue, the dataset was adjusted to ensure a balanced representation of both gym and home/normal room environments. This final adjustment balanced the initial overfitting with the new issue of high variance and seemed to be the best solution for creating the dataset. Despite these adjustments, the dataset still fell short of the ideal size for training a neural network with similar edge cases (e.g., distinguishing between squats and deadlifts or between working out and not working out) considering there are only so many images available that are directly associated with this project. Furthermore, only so many of the group's curated images can be used to ensure that the network is not biased to the group member's body types, clothing, workout equipment, camera quality, or camera angles.

4. Solution Implementation

4.1 Description of Major Code Blocks

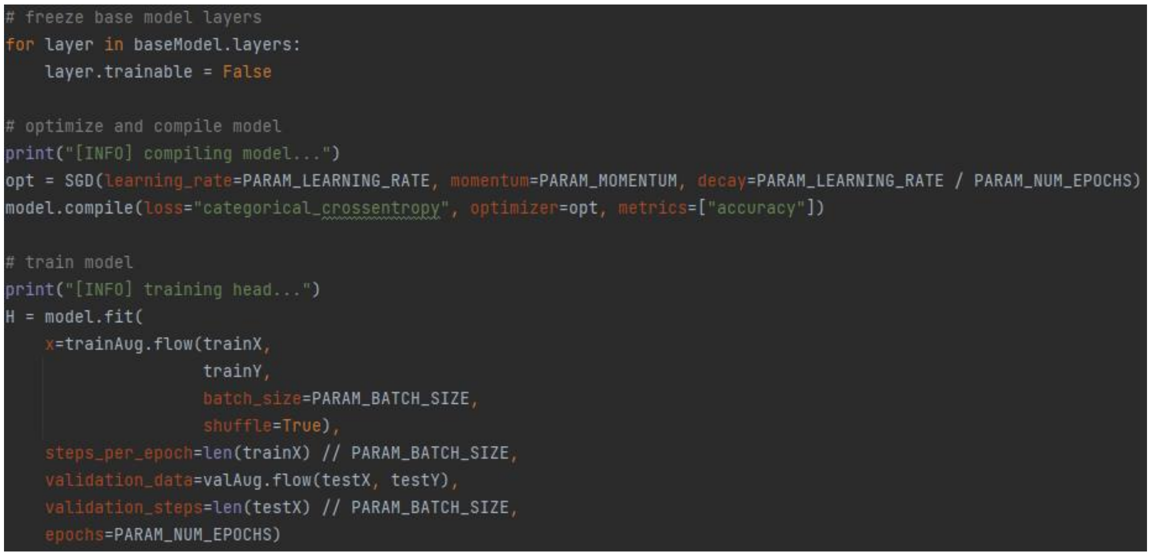
This subsection contains a few code blocks that are integral to the functionality of the model. They will be briefly described here, however more detail and explanations of specific design decisions will be elaborated on later in the report, particularly in Testing/Evaluation/Verification/Validation of Project. This is a result of the fact that the best way to build a machine learning model is not always apparent until testing has begun, especially when it comes to values of hyperparameters.



```
# ResNet50 is loaded as the base model
baseModel = ResNet50(weights="imagenet", include_top=False, input_tensor=Input(shape=(224, 224, 3)))
# additional layers are constructed to be used as the head model (explained in report)
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(PARAM_INPUTS,
                  activation="relu",
                  bias_regularizer=tf.keras.regularizers.L1L2(l1=PARAM_L1, l2=PARAM_L2)
                  )(headModel)
headModel = Dropout(PARAM_DROPOUT)(headModel)
headModel = Dense(len(lb.classes_), activation="softmax")(headModel)
# place the head model on top of the base model
model = Model(inputs=baseModel.input, outputs=headModel)
```

Figure 1: Base model and Head model code block for neural network design

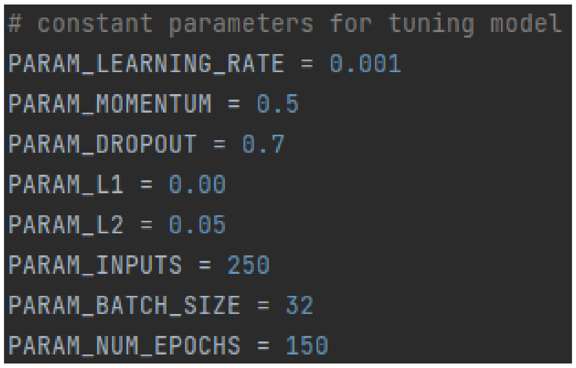
The code for building the model itself can be seen in [Figure 1]. As shown, the ResNet50 model trained on ImageNet is loaded as the base model. A variety of layers are then created and added to the head model which will be used for transfer-learning. Descriptions of the purpose of these layers can be found in [Table 3].



```
# freeze base model layers
for layer in baseModel.layers:
    layer.trainable = False

# optimize and compile model
print("[INFO] compiling model...")
opt = SGD(learning_rate=PARAM_LEARNING_RATE, momentum=PARAM_MOMENTUM, decay=PARAM_LEARNING_RATE / PARAM_NUM_EPOCHS)
model.compile(loss="categorical_crossentropy", optimizer=opt, metrics=["accuracy"])

# train model
print("[INFO] training head...")
H = model.fit(
    x=trainAug.flow(trainX,
                    trainY,
                    batch_size=PARAM_BATCH_SIZE,
                    shuffle=True),
    steps_per_epoch=len(trainX) // PARAM_BATCH_SIZE,
    validation_data=valAug.flow(testX, testY),
    validation_steps=len(testX) // PARAM_BATCH_SIZE,
    epochs=PARAM_NUM_EPOCHS)
```

Figure 2: Code for optimizing and training the model

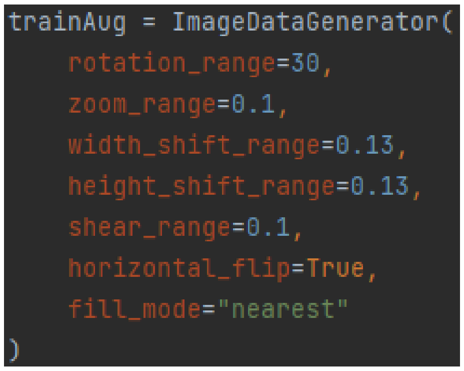
The model was then trained as shown in [Figure 2]. First the base model layers were flagged as untrainable (“frozen”) so that they are unaffected by the training process. The model is then optimized using SGD (gradient descent) and compiled with the categorical cross-entropy loss function. Finally `model.fit()` is called to train the model.



```
# constant parameters for tuning model
PARAM_LEARNING_RATE = 0.001
PARAM_MOMENTUM = 0.5
PARAM_DROPOUT = 0.7
PARAM_L1 = 0.00
PARAM_L2 = 0.05
PARAM_INPUTS = 250
PARAM_BATCH_SIZE = 32
PARAM_NUM_EPOCHS = 150
```

Figure 3: Network parameters

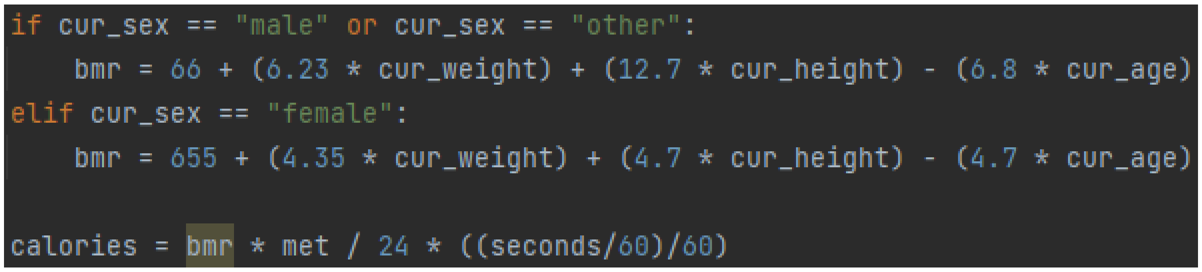
The exact values for each hyperparameter can be seen in [Figure 3]. The PARAM_LEARNING_RATE parameter was also used as the decay rate. The values were determined based on the parameter tuning process described in section 6.



```
trainAug = ImageDataGenerator(
    rotation_range=30,
    zoom_range=0.1,
    width_shift_range=0.13,
    height_shift_range=0.13,
    shear_range=0.1,
    horizontal_flip=True,
    fill_mode="nearest"
)
```

Figure 4: Data Augmentation code

Another important step in creating an effective and general model is data augmentation. Once images were resized and processed by OpenCV, data augmentation was performed as can be seen in [Figure 4]. This allows us to introduce variations into the dataset which can help reduce overfitting and effectively "extend" the dataset. This was particularly important since this project did not use an existing dataset and creating it was a difficult task.



```
if cur_sex == "male" or cur_sex == "other":
    bmr = 66 + (6.23 * cur_weight) + (12.7 * cur_height) - (6.8 * cur_age)
elif cur_sex == "female":
    bmr = 655 + (4.35 * cur_weight) + (4.7 * cur_height) - (4.7 * cur_age)

calories = bmr * met / 24 * ((seconds/60)/60)
```

Figure 5: BMR and calorie calculation code

Finally, once the model has determined the number of frames that are classified as exercise, all data needed to calculate calories has been collected. Frames are converted to seconds using the framerate of the source video and the profile data is queried from the database. The final calorie number is then calculated using the formulas in [Figure 5].

4.2 GUI Description

During the initial development of the neural network (blueprint phase), Group 24 decided it was desirable to implement a front end for the project to allow use of the product without opening the source code. This is an essential aspect of making a product usable by society. Initially, the GUI design was envisioned to be one HTML page that simply accepted three inputs including the weight, load used, and the video file, and would then output the training prediction, confidence level, and calories burnt, as seen in [Figure 6]. This design was then implemented using basic HTML, CSS, and JavaScript but Group 24 decided to go beyond the initial GUI goal and built a front end that felt complete and was much more user friendly. The design additions deemed essential by Group 24 included developing a home page to inform users of the project and how to use the UI, a proper login system that worked with a database to save user information, a My Profile page that allows users to see all their previous inputs and outputs, and simply perfecting the user experience (UX) using interface design techniques.

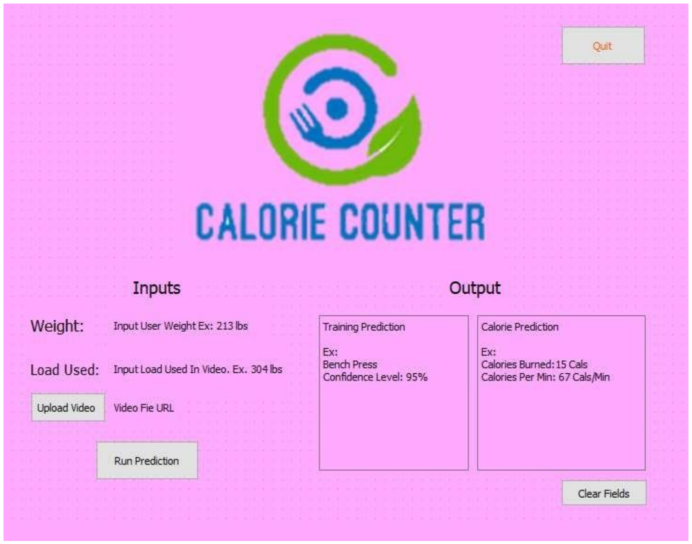


Figure 6: Blueprint design of GUI

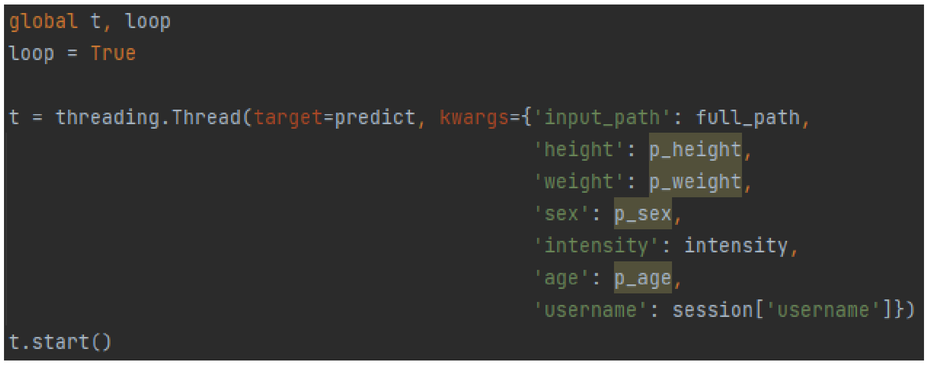
For these additions to be implemented, Group 24 decided to host a web server and database using Flask, a Python framework for creating web applications. Flask allows users to interact with database engines such as SQLite, which was used, and provides users with an interface that allowed Group 24 to effectively handle user data and application generated data. After developing the SQLite database, Group 24 developed the necessary HTML pages for implementing profiles, this included a login page, registration page, and a logout page. Along with these pages, Group 24 implemented a My Profile page that shows the users their total calories burnt over all workouts and shows each individual workout classification and calorie prediction in individual cards. All design changes related to the GUI can be seen in Table 4.

Table 4: GUI design changes from blueprint to final product

GUI Feature	Original Design	Final Design	Reason for Change
Inputs	User Weight, Load used, and video file all inputted to same page.	User inputs their weight, sex, height when creating a profile and doesn't have to again unless they are updating them.	BMR calculation relies on the intensity which is not directly correlated with load used. More weight with less reps is just as intense as less weight with faster/more reps.
Pages	One page as seen in [Figure 6].	My Profile, Home, Login, Registration, Calorie Prediction Page.	Wanted to create a UI with proper UX principles.
Output	Calorie prediction, workout classification, confidence level, calories burnt per minute.	Calorie prediction, workout classification.	Removed the confidence level and calories burnt/min outputs to follow user centered design in respect to the needs of the user. The user only wants the calories burnt and classification of the workout.
Database	None	Database that holds user information such as weight, sex, and height. Additionally, holds the output of each use for every user.	Allows for user to track their calorie expenditure from workouts over a long period of time by giving them individual access to their previous records.

4.3 Integration

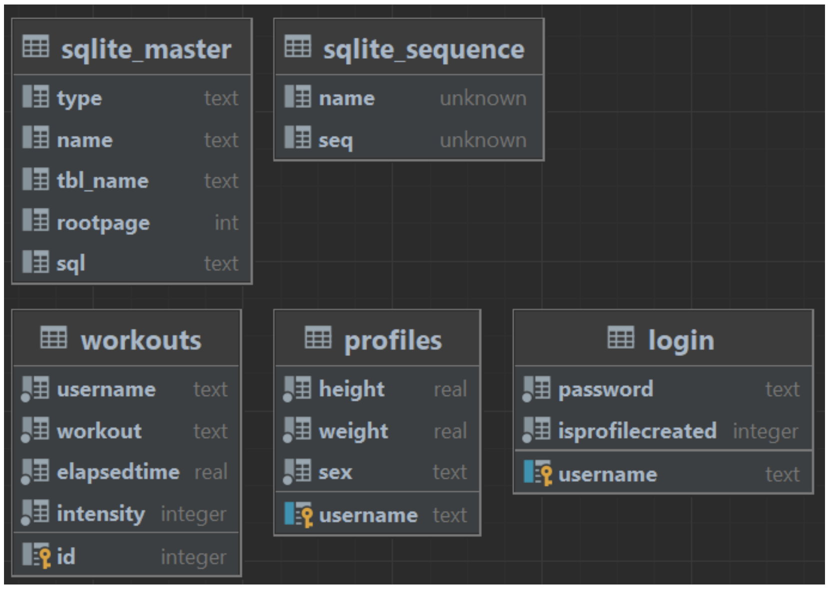
The main advantage of using Flask for the webserver and frontend framework is that it was easy to integrate with the backend. Since the prediction logic was written in Python, using a Python web framework allowed the frontend and backend functions to easily interact with one another. When a Flask route is accessed by a user loading a page, the python Threading library was used to call a function elsewhere in the same file to run prediction. This has the benefit of being fully parallelized, allowing a user to upload successive videos to be processed before the first one is completed. This was deemed an essential feature as machine learning can be a slow process, even when just inferencing. An example of a thread being created to run the `predict()` function can be seen in [Figure 7] below.



```
global t, loop
loop = True

t = threading.Thread(target=predict, kwargs={'input_path': full_path,
                                             'height': p_height,
                                             'weight': p_weight,
                                             'sex': p_sex,
                                             'intensity': intensity,
                                             'age': p_age,
                                             'username': session['username']})
t.start()
```

Figure 7: Example code of a thread being created to handle prediction.

When a user uploads a video through a simple POST method form, they are saved to a separate directory for each user, determined by the session token set at login. The file path of the video, along with user profile data (queried using the aforementioned username session token), are then passed into the previous n function as kwargs when the thread is created. With this information passed in, and now running in its own thread, the python predict function can operate exactly as it did before the frontend was integrated. Once the prediction is complete, the calories burned are calculated using the formulas shown in [Figure 5]. The resulting data is then inserted into the database. An example of a database operation being performed in Flask can be seen in Appendix [Figure 20], which is the code for the simple user account registration functionality. The schema for the database can be seen below in [Figure 8].



sqlite_master	
type	text
name	text
tbl_name	text
rootpage	int
sql	text

sqlite_sequence	
name	unknown
seq	unknown

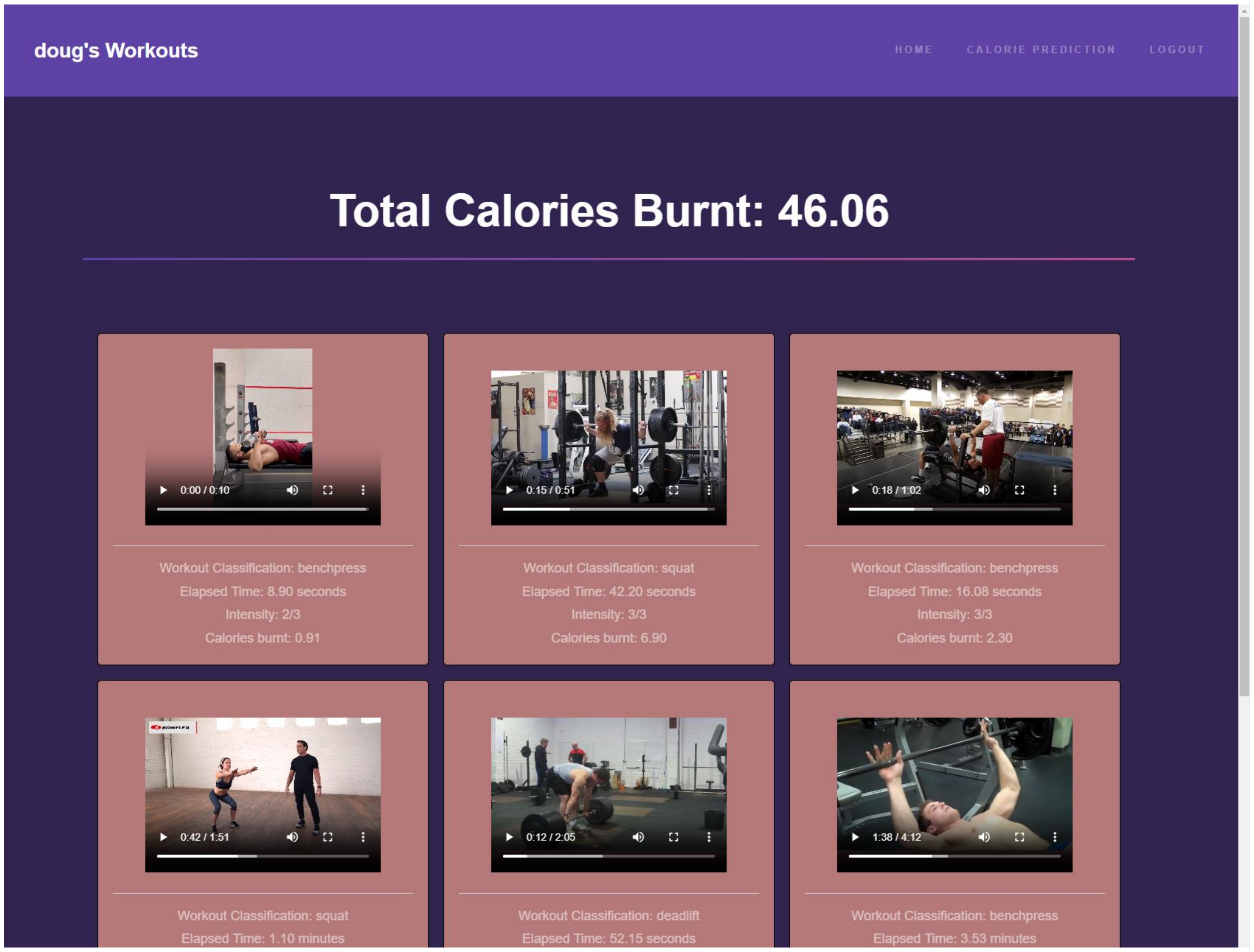
workouts	
username	text
workout	text
elapsedtime	real
intensity	integer
id	integer

profiles	
height	real
weight	real
sex	text
username	text

login	
password	text
isprofilecreated	integer
username	text

Figure 8: Diagram of the database schema

When the user navigates to the My Profile page they are able to view all of the workouts they have uploaded. Each of their workouts can be viewed and played back, and information such as calories burnt, duration, and intensity are displayed alongside them. This page can be seen below in [Figure 9].



Total Calories Burnt: 46.06

Workout Classification: benchpress
Elapsed Time: 8.90 seconds
Intensity: 2/3
Calories burnt: 0.91

Workout Classification: squat
Elapsed Time: 42.20 seconds
Intensity: 3/3
Calories burnt: 6.90

Workout Classification: benchpress
Elapsed Time: 16.08 seconds
Intensity: 3/3
Calories burnt: 2.30

Workout Classification: squat
Elapsed Time: 1.10 minutes

Workout Classification: deadlift
Elapsed Time: 52.15 seconds

Workout Classification: benchpress
Elapsed Time: 3.53 minutes

Figure 9: Frontend for My Profile page with 6 processed workouts displayed

The posts were displayed on the page by querying the database for all workouts for the logged in user upon page load. Similarly, to how PHP would be used on a more traditional website, Jinja2 was used in-line with HTML in order to loop through and display all of the posts and necessary information. The code for this process can be seen in Appendix [Figure 21]. Additional screenshots of the final frontend design can also be found in the Appendix from [Figure 22] to [Figure 26]. Finally, a diagram outlining the dataflow for the entire backend can be seen in [Figure 10] below.

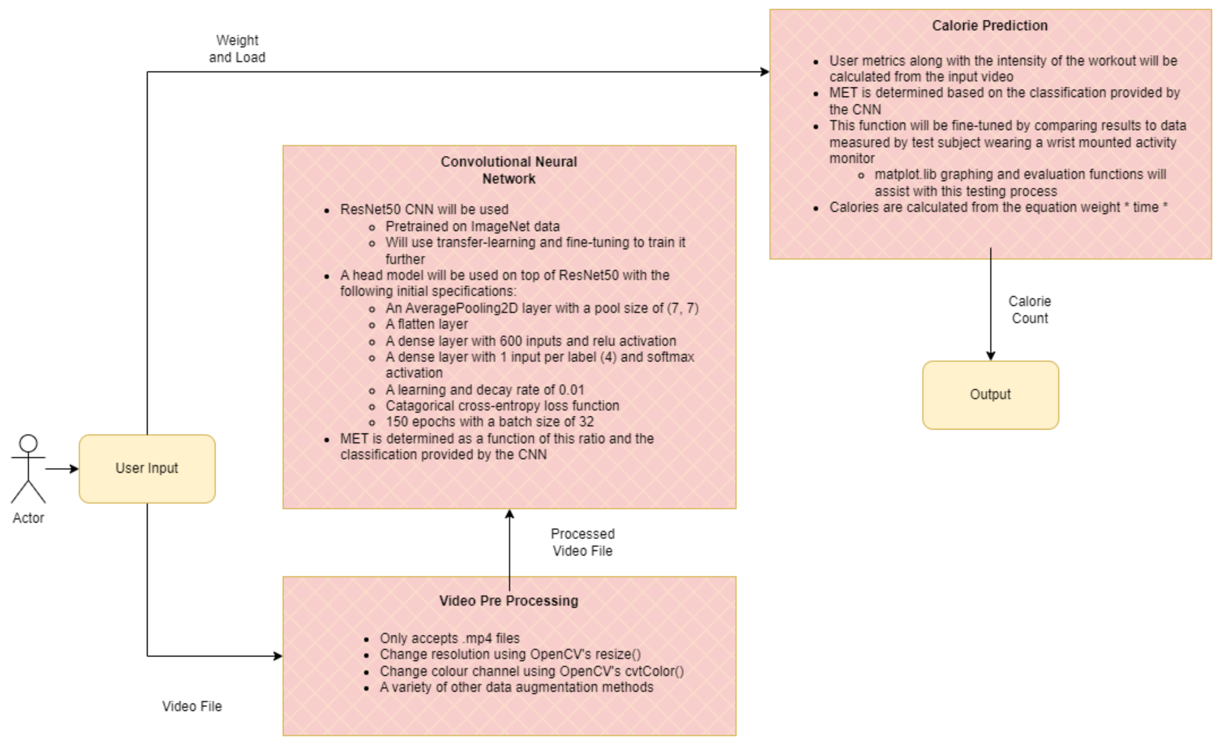

Weight and Load
Calorie Prediction
User metrics along with the intensity of the workout will be calculated from the input video
MET is determined based on the classification provided by the CNN
This function will be fine-tuned by comparing results to data measured by test subject wearing a wrist mounted activity monitor
matplot.lib graphing and evaluation functions will assist with this testing process
Calories are calculated from the equation weight * time *
Convolutional Neural Network
ResNet50 CNN will be used
Pretrained on ImageNet data
Will use transfer-learning and fine-tuning to train it further
A head model will be used on top of ResNet50 with the following initial specifications:
An AveragePooling2D layer with a pool size of (7, 7)
A flatten layer
A dense layer with 600 inputs and relu activation
A dense layer with 1 input per label (4) and softmax activation
A learning and decay rate of 0.01
Catagorical cross-entropy loss function
150 epochs with a batch size of 32
MET is determined as a function of this ratio and the classification provided by the CNN
Calorie Count
Output
Actor
User Input
Processed Video File
Video Pre Processing
Only accepts .mp4 files
Change resolution using OpenCV's resize()
Change colour channel using OpenCV's cvtColor()
A variety of other data augmentation methods
Video File

Figure 10: Flowchart describing the high-level inputs and outputs for the backend system

5. Bill of Materials/Project Budget

The project incurred a cost of $0. Originally it was thought that access to online computing resources would be needed for the training and testing of the network, through Google Colab Pro, but the model was trained and ran locally on the machines of the group members throughout the year.

6. Testing/Evaluation/Verification/Validation of Project

This section provides details of the steps taken in testing, evaluation, verification, and validation of the project. The two main components of the project, the model and calorie output, were tested thoroughly to ensure that the project met the functional requirements.

6.1 Model Evaluation

Two primary methods were used to measure the performance of the classification model. First, a confusion matrix was used to get raw numerical data from the model after it has been trained. However, it was also necessary to use plots that show the training/validation loss and accuracy at each epoch. This made it easier to determine whether the model was overfitting or underfitting the dataset. These plots also helped in selecting the required number of epochs. The specifications of the first head model tested can be seen below:

- An AveragePooling2D layer with a pool size of (7, 7)
- A flatten layer
- A dense layer with 500 inputs and ReLU activation
- A dense layer with 1 input per label (4) and Softmax activation
- A learning and decay rate of 0.01
- Categorical cross-entropy loss function
- 150 epochs with a batch size of 32

After the initial train, the confusion matrix showed seemingly promising results, as seen in Table 5. The team was satisfied with the overall accuracy of 0.87, especially since the model will be effectively averaging its predictions over many frames. However, the “none” label showed much lower precision, recall, and f1-scores. This was expected since it proved difficult to collect data for this class. Many images over a wide array of domains were included, along with pictures of gym equipment with no people around. Having a class with such a wide variety of possibilities necessitated a significant increase in dataset size for all labels. When the training plot was examined, signs of significant overfitting were exhibited. This can be seen below in [Figure 11].

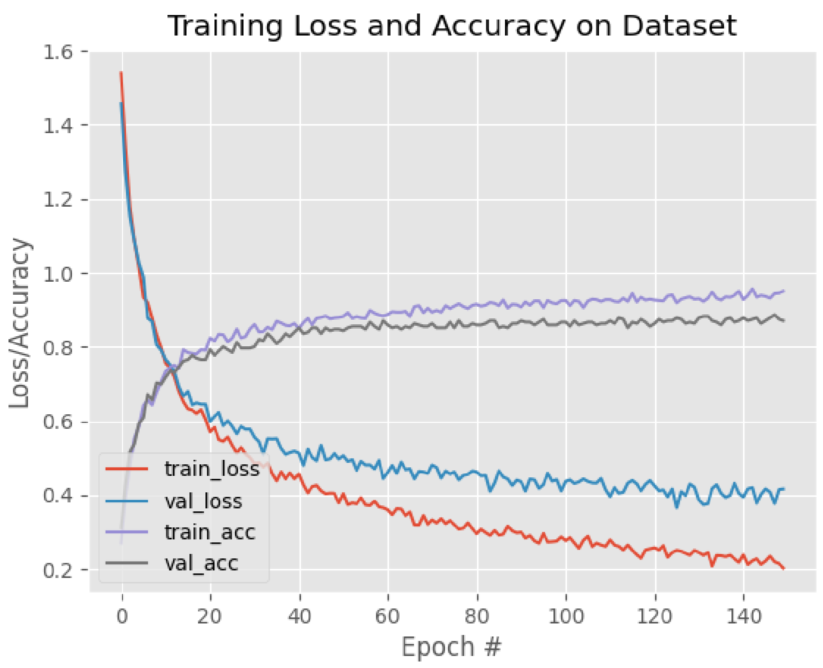


Figure 11: Training plot of initial model

As shown in the figure, the training loss of the model continued to decrease well beyond the point at which the validation loss plateaued. Similarly, the training accuracy increased beyond the validation accuracy. This means that the model was "memorizing" the dataset and it wouldn't be as effective at generalization. If such a model were deployed, an example of a possible error could be a user wearing similar clothing to a subject in the dataset. The model could misclassify the user as doing the same exercise as the subject in the dataset if it is not sufficiently generalized.

A number of techniques were introduced to reduce the overfitting of the model, and hopefully improve the model's ability to correctly classify the "none" label. Namely, dropout and L2 regularization were used to reduce the model's reliance on any particular inputs. Momentum was also added which reduced the number of epochs before convergence. More detail on each of these elements of the head model can be seen in [Table 3]. The model was also tested with 250, 500, and 1000 inputs in the first dense layer. After adding each of these elements, the model was retrained, and evaluation metrics were recorded. The plot of the final model can be seen in [Figure 12] below.

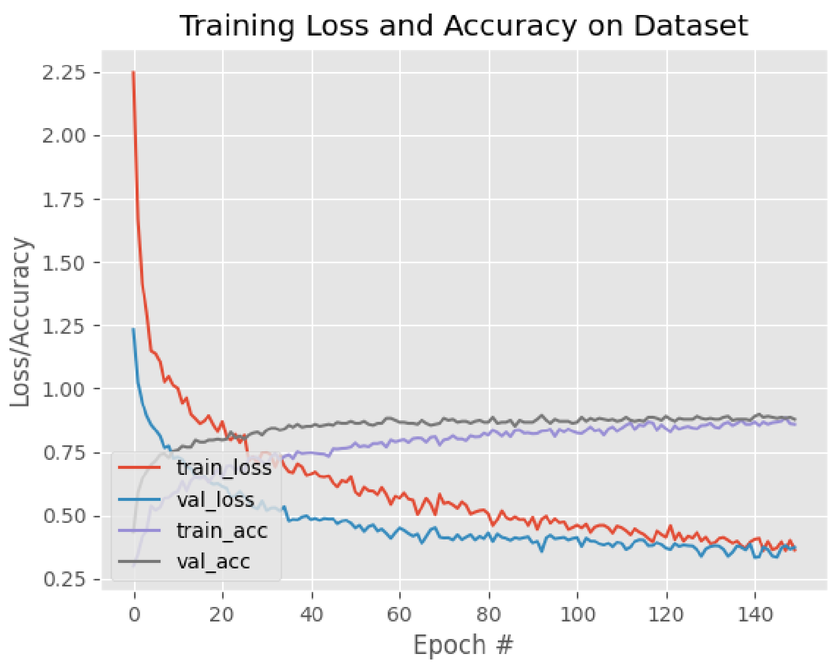


Figure 12: Training plot of final model (with dropout, regularization, momentum, and 250 inputs)

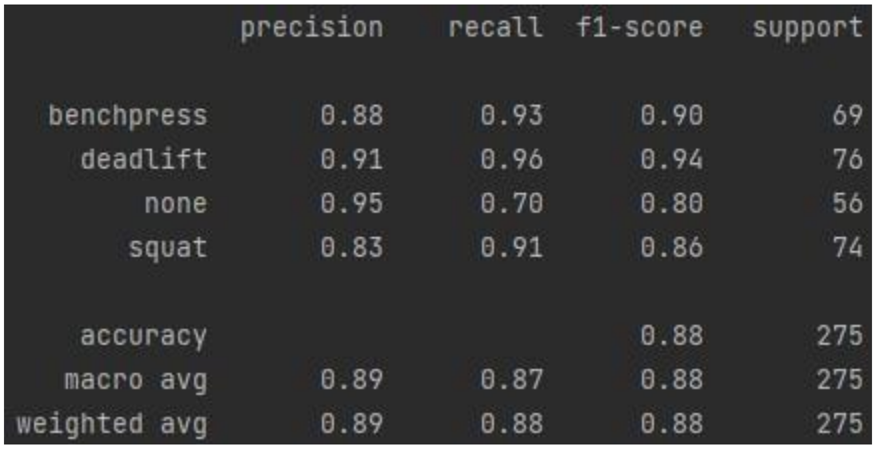
When compared with the plot in [Figure 11], the accuracies and losses converge at 150 epochs instead of a gradual divergence. This is evidence of a much more balanced model. It is also worth noting that the reason the validation loss and accuracy appear to start out better than the training loss and accuracy, (flipped compared to previous figure) is due to regularization. This is a result of dropout enforcing artificial limitations on the model (randomly removing inputs) during training, which helps it perform better in validation and general scenarios. The numerical data for each tuning step can be seen below in [Table 5].

Table 5: Confusion matrix data for several iterations of the fine-tuning process

Model	Accuracy	Label	Precision	Recall	F1-score	Training Plot
Initial model	0.87	benchpress	0.91	0.97	0.94	Figure 11
		deadlift	0.92	0.95	0.94	
		none	0.86	0.66	0.75	
		squat	0.80	0.86	0.83	
	0.86	benchpress	0.88	0.97	0.92	

Dropout added after first dense layer (rate = 0.7)		deadlift	0.91	0.92	0.92	Figure 15
		none	0.87	0.61	0.72	
		squat	0.8	0.89	0.84	
Regularization added (for biases only, L2 penalty = 0.05)	0.88	benchpress	0.91	0.99	0.94	Figure 16
		deadlift	0.92	0.92	0.92	
		none	0.86	0.64	0.73	
		squat	0.82	0.91	0.86	
Momentum added (0.5)	0.88	benchpress	0.87	0.99	0.93	Figure 17
		deadlift	0.9	0.96	0.93	
		none	0.9	0.62	0.74	
		squat	0.87	0.91	0.89	
With 1000 inputs (first dense layer)	0.88	benchpress	0.92	0.96	0.94	Figure 18
		deadlift	0.91	0.93	0.92	
		none	0.91	0.7	0.79	
		squat	0.82	0.91	0.86	
With 250 inputs (first dense layer)	0.88	benchpress	0.88	0.93	0.9	Figure 12
		deadlift	0.91	0.96	0.94	
		none	0.95	0.7	0.8	
		squat	0.83	0.91	0.86	
With additional 100 input layer (same parameters)	0.87	benchpress	0.89	0.97	0.93	Figure 19
		deadlift	0.91	0.92	0.92	
		none	0.87	0.61	0.72	
		squat	0.8	0.91	0.85	

As shown in green, the model with dropout, L2 regularization, momentum, and 250 inputs was most effective in classifying the none label. As all other statistics were within acceptable margins of error from their maximum value, this model was chosen to be the final model. A greater emphasis was put upon the "none" label, not only because it performed the weakest, but because when making a reasonable calorie prediction it is essential to know exactly when a workout stops and finishes in order to get an accurate time estimate. For the "none" label, this model showed improvements of 0.09, 0.04, and 0.05 in precision, recall, and f1-score respectively when comparing against the initial model. This improvement, along with the fact that the model was no longer overfitting, led the team to deem the parameter tuning process a success. The full confusion matrix for the final model can be found in [Figure 13] below.



```
              precision    recall  f1-score   support

  benchpress       0.88      0.93      0.90        69
    deadlift       0.91      0.96      0.94        76
        none       0.95      0.70      0.80        56
       squat       0.83      0.91      0.86        74

    accuracy                           0.88       275
   macro avg       0.89      0.87      0.88       275
weighted avg       0.89      0.88      0.88       275
```

Figure 13: Full Confusion Matrix of final model

6.2 Calorie Expenditure Evaluation

To evaluate the performance of the model in respect to its ability to accurately calculate calories burnt, Group 24 used an apple watch as its gold standard for predicting calories. An Apple Watch was chosen as the gold standard because of a studied conducted by Stanford University in 2017 that found Apple Watches were the best in its class at predicting calorie expenditure out of 60 competitors [7]. To determine how accurate the group's model prediction is in comparison to the Apple Watch the group conducted an experiment using two subjects, [Table 6]. Both subjects conducted two sets of each classification exercise, bench press, squat, and deadlift, while wearing an Apple Watch so calories burnt could be tracked. These exercises were filmed and then ran through the neural network to ensure the calorie predictions were accurate. It was found that the model was 96.9% accurate in comparison to an Apple Watch for these workouts specifically, which is well above the goal outlined in [Table 7]. It is essential to outline that tested data only had two test subjects and therefore the testing does not cover all use cases, which would be people of all ages, heights, weights, and fitness levels.

Table 6: Data representing the difference in calorie predictions between an apple watch (gold standard) and our model, using the same workouts recorded by anonymous test subjects.

Test Data	Sex	Height (in)	Weight (lbs)	Exercise	Time (sec)	Intensity	Apple Watch Calorie Prediction	Model Calorie Prediction
Subject X	M	74	216	Bench Press	34.12	Light	2.5	2.61
				Bench Press	28.76	Moderate	3.5	3.66
				Squat	64.49	Vigorous	13	13.15
				Squat	46.22	Moderate	7	7.07
				Deadlift	48.96	Vigorous	11	11.23
				Deadlift	34.66	Moderate	6	6.18
Subject Y	F	62	110	Bench Press	44.74	Light	2	2.06
				Bench Press	41.26	Moderate	3	3.17
				Squat	38.92	Vigorous	4.5	4.78
				Squat	40.48	Moderate	3.5	3.73
				Deadlift	39.71	Moderate	4	4.27
				Deadlift	42.39	Light	2.5	2.6

7. Stakeholder Needs

The project was geared towards one main stakeholder, the user. The users' needs were considered in every aspect of the design. The GUI was the main aspect of the project that the user would interact with. As result, the GUI was designed with user-experience as the main priority. A released version of the website and model were not officially created for public access, and because of this, there were some aspects of the design that undermined user safety and privacy.

Designing a website that prioritizes safety and privacy requires careful consideration of various design components, and the website created would have to be slightly modified in some respects to increase overall site security. The database connected to the website was run on a local machine, where the data would be accessed for calorie prediction after a user had input a video. There are some details stored about the user, specifically their height, weight, age, and sex that would have to be encrypted to ensure user privacy is upheld.

A few aspects that would increase website security, and boost user privacy and safety are the following design components:

1. Secure connection: ensuring the website has an HTTPS connection for safeguarding users' data.
2. Authentication & Authorization: providing users with strong authentication and authorization mechanisms can help prevent unauthorized access to accounts and user data.
3. Privacy Policy: create policy to help users understand exactly how their data is being used and protected. The policy would outline what data is collected, how it is used and who has access to it.
4. User Controls: providing the user with tools to control their data. This can help build trust and improve user engagement.
5. Regular Updates and Maintenance: keeping the website up to date with the latest security patches can prevent any vulnerabilities from being exploited. Regular maintenance can also help identify any security issues that arise, so they can be fixed.

Currently, the website contains a secure connection, but due to the nature of the project and the fact that the website is not available for public use, these other secure design aspects were not implemented. Should the project be taken any further, and the website developed for public use, these design components must be utilized to ensure maximum user safety and privacy.

8. Project Compliance with Blueprint Target Specifications

The system specifications are listed in the table below, [Table 7]. As the project was entirely software based, there were no hardware specifications to meet and therefore they are not addressed below.

Table 7: Software system specifications

	Specification	**Specification met? (Yes/No)**
1	**Functional Requirements**	
1.1	Predicting the weightlifting movement captured in a video, using a convolutional neural network trained with a dataset of images.	**Yes**
1.2	Create/obtain a dataset that is adequately large and diverse with the correct sets of workout related images (bench-press, squat, deadlifts) to train network	**Yes**
1.3	Fully integrate front end to back end.	**Yes**
2	**Interface Requirements**	
2.1	Create a functional GUI to use as the front end of the product. The GUI must be able to take users' weight, the load being used, and the video file as input and send it to the back end for completing the predictions and then receive the output and display it to the user.	**Yes**
3	**Performance Requirements**	
3.1	The response time for any one video, as seen by the user, should be under five seconds.	**No**
3.2	The accuracy of our classification over all tests performed must be greater than 80 percent.	**Yes**
3.3	The accuracy of calories burnt for any one video should be within 30 calories per minute of the recorded calories burnt per minute when testing using an Apple Watch.	**Yes**

Overall, the group was able to successfully address and implement almost all the software specifications outlined in the blueprint document. Unfortunately, there was one software specification that the group was not able to meet. Despite Group 24's best efforts, a response time for any one video that was under five seconds as seen by the user was unable to be achieved. Through testing the CNN, it was found that multiple factors such as longer videos with higher frame rates required more computational power to process and classify. As a result, the group realized this performance requirement was an unattainable goal without access to more powerful GPUs with multi-threading capabilities. Additionally, even with the ideal computational power provided to the CNN, videos beyond a certain length of time would not be able to be classified within a time of five seconds or less. Looking back, this performance requirement should be amended to have different response time goals for videos of lengths and frame rates within a specific range.

Regardless of this shortcoming in response time, the group was successful in implementing all the other software specifications that were laid out at the start of the project. The neural network was able to accurately predict the weightlifting movement captured in a video, and calories burned were calculated using inputs from the user and the predicted movement. As previously discussed, a functional GUI was created and fully integrated with the back end. The GUI was able to take users' personal information, however slightly different from the ones specified in [Table 7], and the video file as input, send it to the back end for completing the predictions, and then receive the output and display it to the user. Additionally, the accuracy for classification was far greater than 80 percent, [Figure 13], and the predicted calorie count was well within 30 calories per minute of the recorded calories burnt per minute when testing using an Apple Watch.

9. Conclusions and Recommendations

In conclusion, the groups research and project have shown that the use of object tracking (computer vision) alongside heart rate monitoring is a more effective approach for predicting calorie expenditure during a workout compared to image classification. Using a method like computer vision would allow tracking and counting of repetitions performed, track the amount of weight used for the exercise, and monitor the heart rate in real time. This information would not only enhance the accuracy of the calorie prediction, but also make the system simpler to use, and require less information input by the user compared to image classification. By using image classification, group 24 was still able to successfully estimate the number of calories burned based on videos of people performing exercises but more input was required from the user such as, height, weight, age, and sex, to accurately estimate the number of calories burned.

It was also discovered that having a high-quality dataset is critical for training a network. Much of the success of this project can be attributed to the dataset used in training the model. For future, having a more comprehensive dataset where a wider variety of athletes and workout environments are represented would lead to a more accurate model, and ultimately a more accurate calorie prediction.

Overall, the project met many of the goals for success, but there are still areas which could be improved for future work. Firstly, implementing a server to maintain a central database for user logins and workout sessions, along with implementing proper safety controls to protect user data. The current implementation requires a machine that can connect to a local database, which is not ideal for dissemination of the application. Hosting a database server would allow remote users to be able to access the full functionality of application and perform calorie predictions from their personal machine. Building upon this, implementing a cloud-based GPU processing environment for faster inferencing would be another quality-of-life improvement. For future adaptations of the project, the group proposes development of an IOS product where calorie prediction would be accessible near-instantly from any location. Users would be able to record videos from their cellphone and upload it to the app to generate on demand calorie predictions. Additionally, mobile applications could implement many more user features, such as a social aspect between users. Week Moving the project in this direction would incur many additional costs, as well as continued investment for future development and maintenance of the product.

10. Effort Expended

Table 8: Overall effort expended by all members on the project

Name	**Overall effort expended (%)**
Doug Byers	100
Justin Carr	100
Niko Hoogeveen	100
Ryan Steiner	100

References

[1] A. Rosebrock, ‘Video classification with Keras and Deep Learning’, *PyImageSearch*, Jul. 15, 2019. https://pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/ (accessed Oct. 25, 2022).

[2] ‘Sports Marketing’, *Routledge & CRC Press*. https://www.routledge.com/Sports-Marketing/Fetchko-Roy-Clow/p/book/9781138039841 (accessed Apr. 10, 2023).

[3] ‘Smartwatch Chips Global Market to Reach $3.22 Billion by 2030: Increasing Demand for Wearable Devices Drives Growth’, *Yahoo Finance*, Apr. 07, 2023. https://finance.yahoo.com/news/smartwatch-chips-global-market-reach-123800730.html (accessed Apr. 10, 2023).

[4] ‘Convolutional Neural Network: Benefits, Types, and Applications’, *Datagen*. https://datagen.tech/guides/computer-vision/cnn-convolutional-neural-network/ (accessed Apr. 10, 2023).

[5] O. Semih Kayhan and J. C. van Gemert, ‘On Translation Invariance in CNNs: Convolutional Layers Can Exploit Absolute Spatial Location’, in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA: IEEE, Jun. 2020, pp. 14262–14273. doi: 10.1109/CVPR42600.2020.01428.

[6] R. Alake, ‘Understanding Parameter Sharing (or weights replication) Within Convolutional Neural Networks’, *Medium*, Jun. 16, 2020. https://towardsdatascience.com/understanding-parameter-sharing-or-weights-replication-within-convolutional-neural-networks-cc26db7b645a (accessed Apr. 10, 2023).

[7] ‘How Accurate Are Apple Watch Calories? (2023)’, *www.iphonelife.com*, Dec. 09, 2020. https://www.iphonelife.com/content/how-accurate-are-apple-watch-calories-how-to-ensure-theyre-accurate (accessed Apr. 08, 2023).

Appendix

Table 9: Metabolic Equivalents (MET) for exercises

Exercise	Light Intensity	Moderate Intensity	Vigorous Intensity
Deadlift	4	7	9
Squat	3.5	6	8
Bench Press	3	5	7

$$BMR_{Male} = 66 + (6.23 \times weight\ in\ lbs) + (12.7 \times height\ in\ inches) - (6.8 \times age\ in\ years)$$

$$BMR_{Female} = 655 + (4.35 \times weight\ in\ lbs) + (4.7 \times height\ in\ inches) - (4.7 \times age\ in\ years)$$

$$CALs = BMR \times METs \div 24 \times duration\ of\ activity\ in\ hours$$

Figure 14: Basal Metabolism Rate and Calories Burnt equations

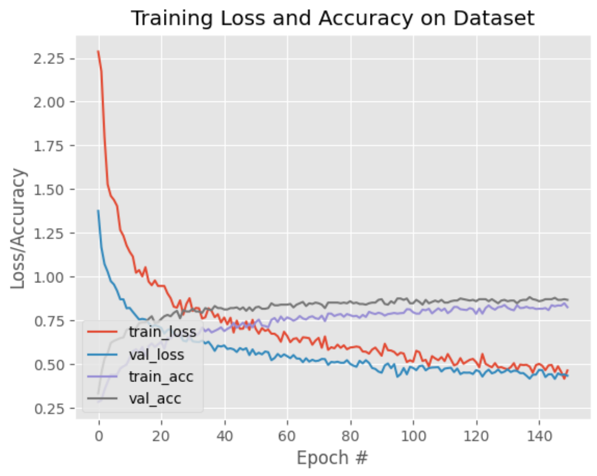


Figure 15: Plot of model with dropout

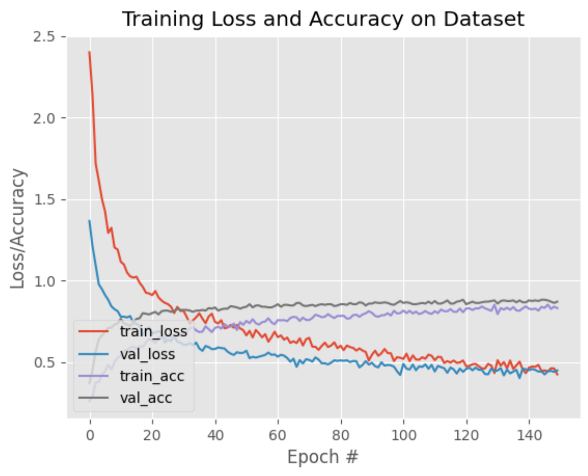


Figure 16: Plot of model with dropout and regularization

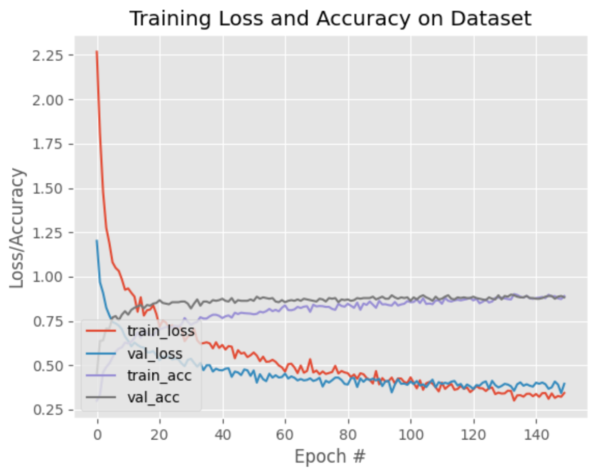


Figure 17: Plot of model with dropout, regularization, and momentum

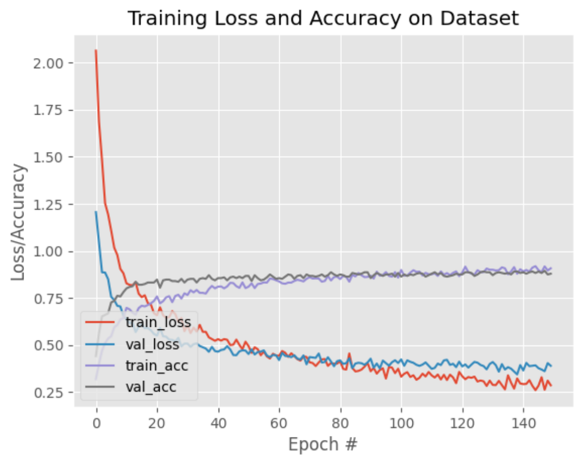


Figure 18: Plot of model with dropout, regularization, momentum, and 1000 inputs

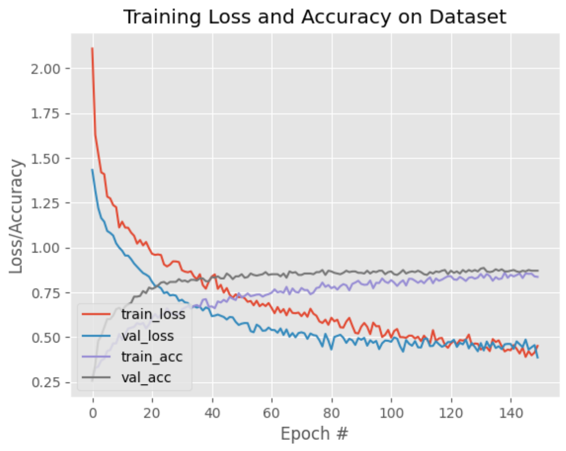
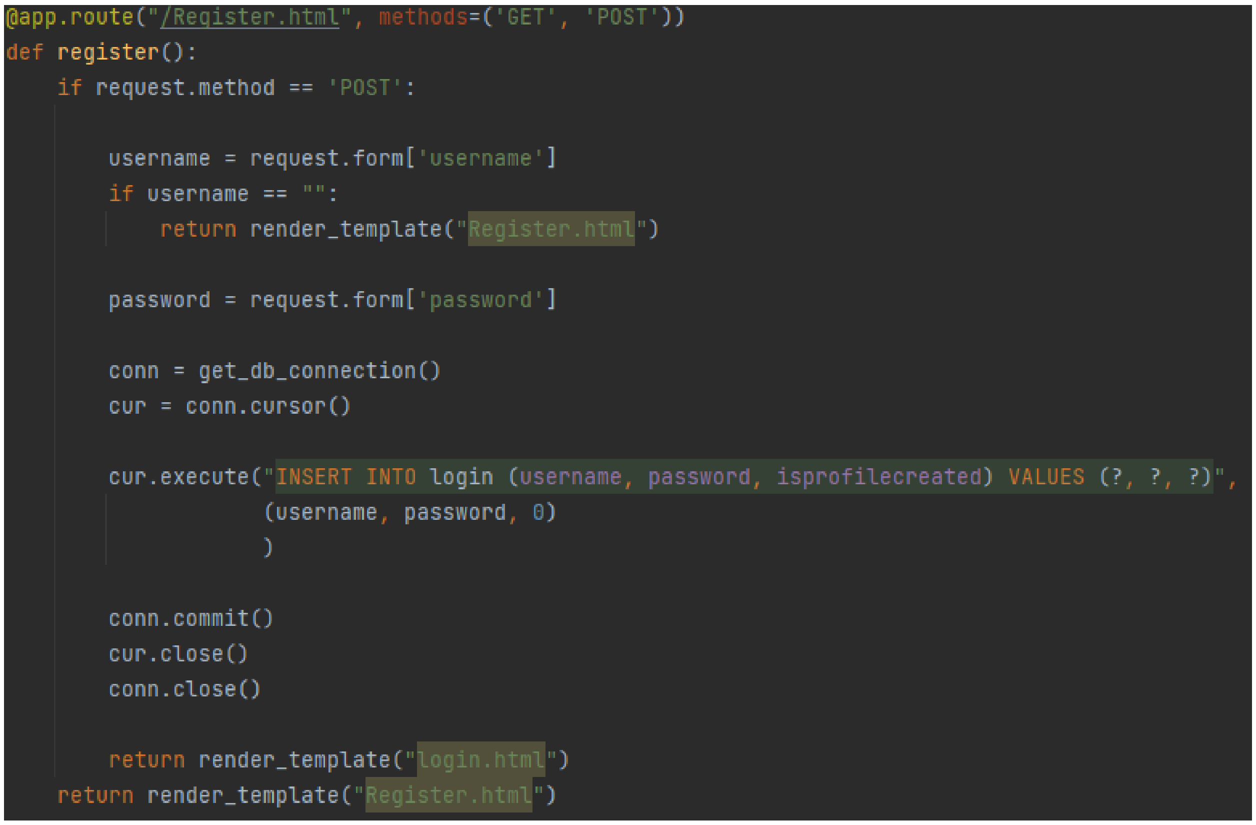
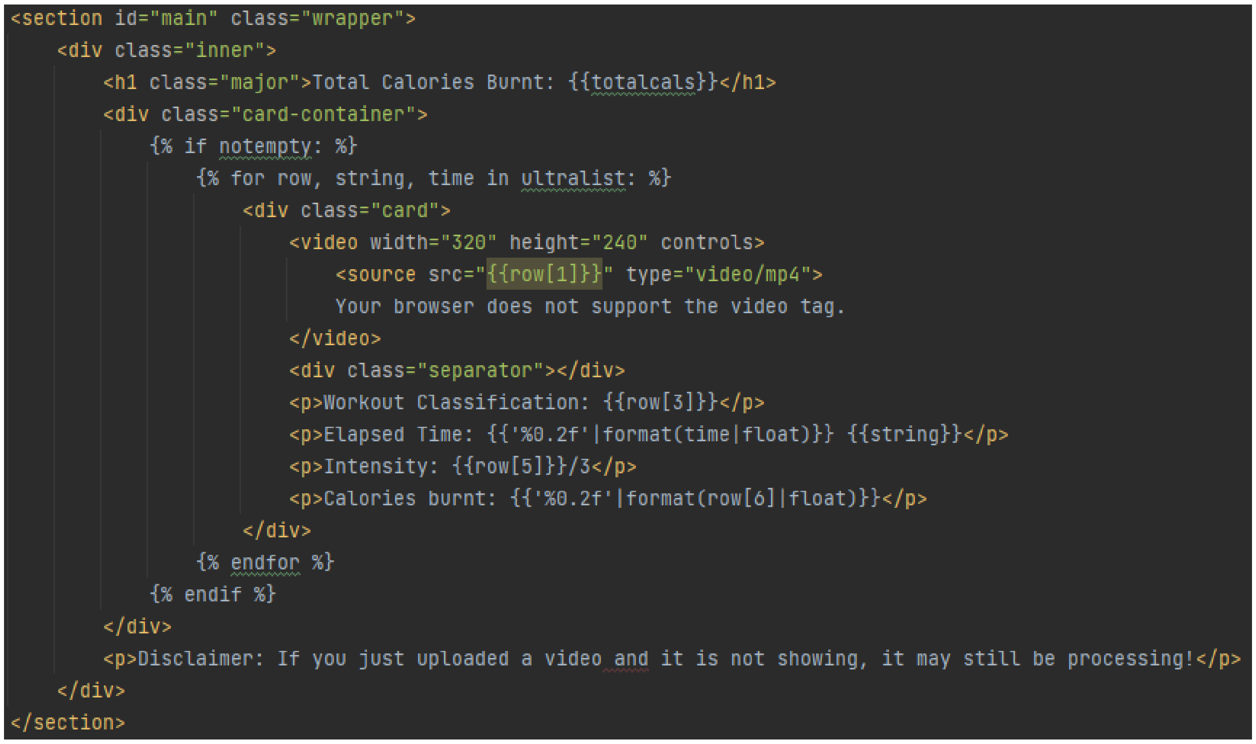


Figure 19: Plot of model with dropout, regularization, momentum, 250 inputs, and an extra 100 input layer



```
@app.route("/Register.html", methods=('GET', 'POST'))
def register():
    if request.method == 'POST':

        username = request.form['username']
        if username == "":
            return render_template("Register.html")

        password = request.form['password']

        conn = get_db_connection()
        cur = conn.cursor()

        cur.execute("INSERT INTO login (username, password, isprofilecreated) VALUES (?, ?, ?)",
                    (username, password, 0)
                    )

        conn.commit()
        cur.close()
        conn.close()

        return render_template("login.html")
    return render_template("Register.html")
```

Figure 20: Code for the simple account registration. This is not cybersecure and should not be used in a production environment



```
<section id="main" class="wrapper">
    <div class="inner">
        <h1 class="major">Total Calories Burnt: {{totalcals}}</h1>
        <div class="card-container">
            {% if notempty: %}
                {% for row, string, time in ultralist: %}
                    <div class="card">
                        <video width="320" height="240" controls>
                            <source src="{{row[1]}}" type="video/mp4">
                            Your browser does not support the video tag.
                        </video>
                        <div class="separator"></div>
                        <p>Workout Classification: {{row[3]}}</p>
                        <p>Elapsed Time: {{'%0.2f'|format(time|float)}} {{string}}</p>
                        <p>Intensity: {{row[5]}}/3</p>
                        <p>Calories burnt: {{'%0.2f'|format(row[6]|float)}}</p>
                    </div>
                {% endfor %}
            {% endif %}
        </div>
        <p>Disclaimer: If you just uploaded a video and it is not showing, it may still be processing!</p>
    </div>
</section>
```

Figure 21: Code for displaying workouts to the user

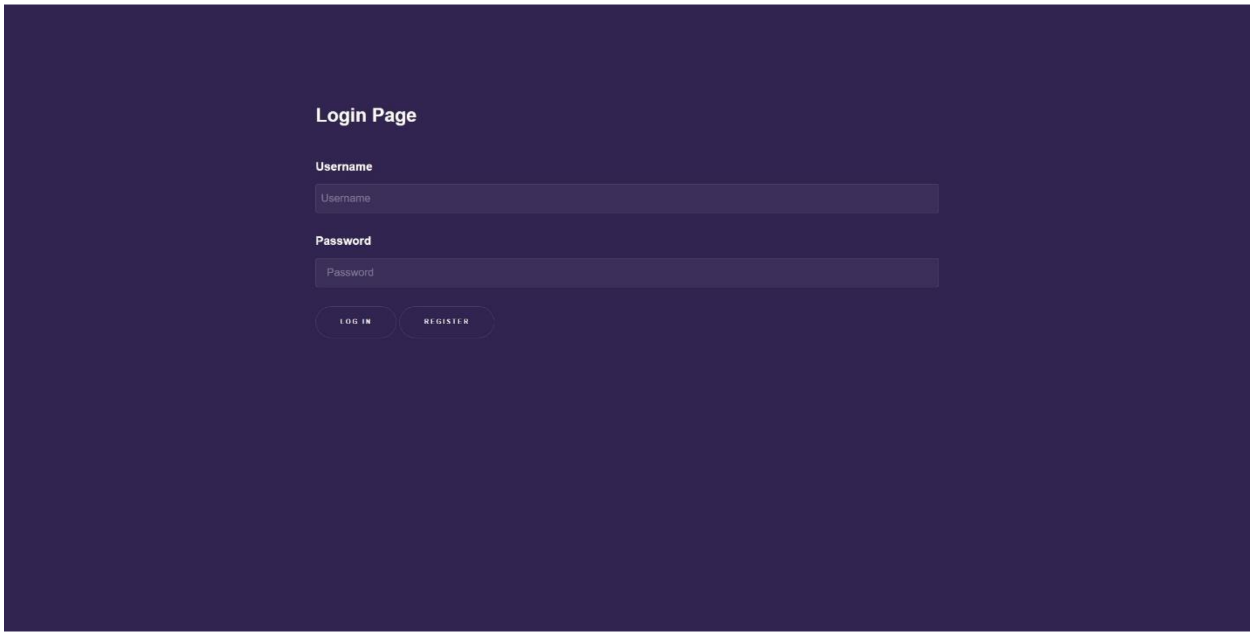


Figure 22: GUI Login page

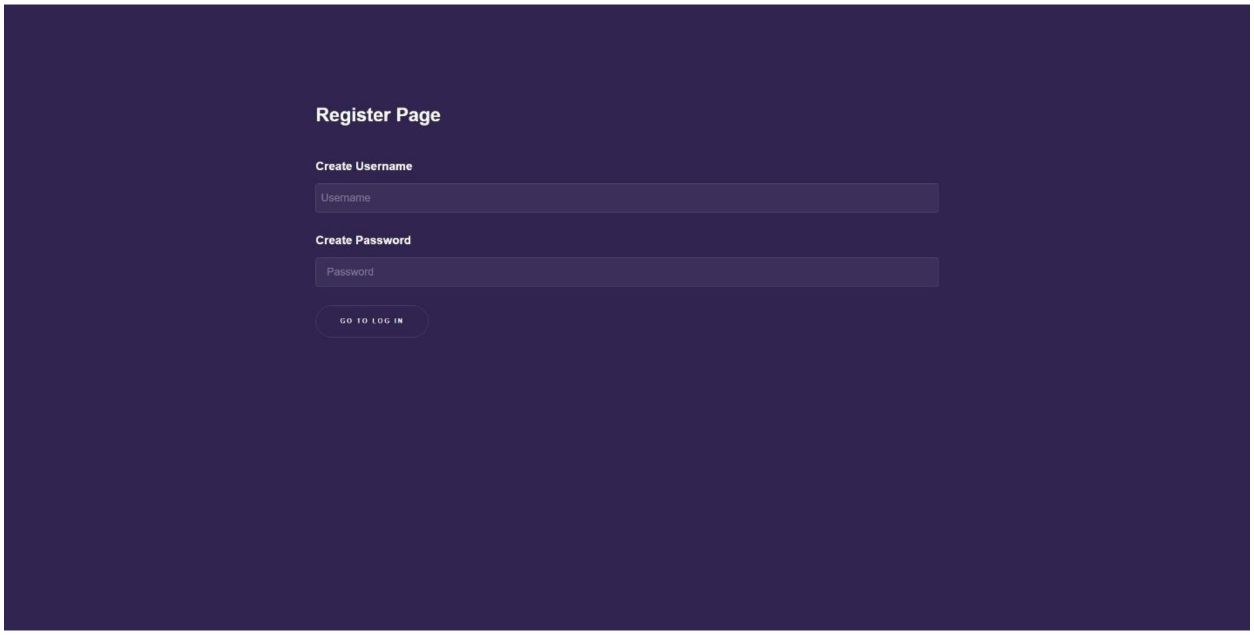


Figure 23: GUI Register Profile page

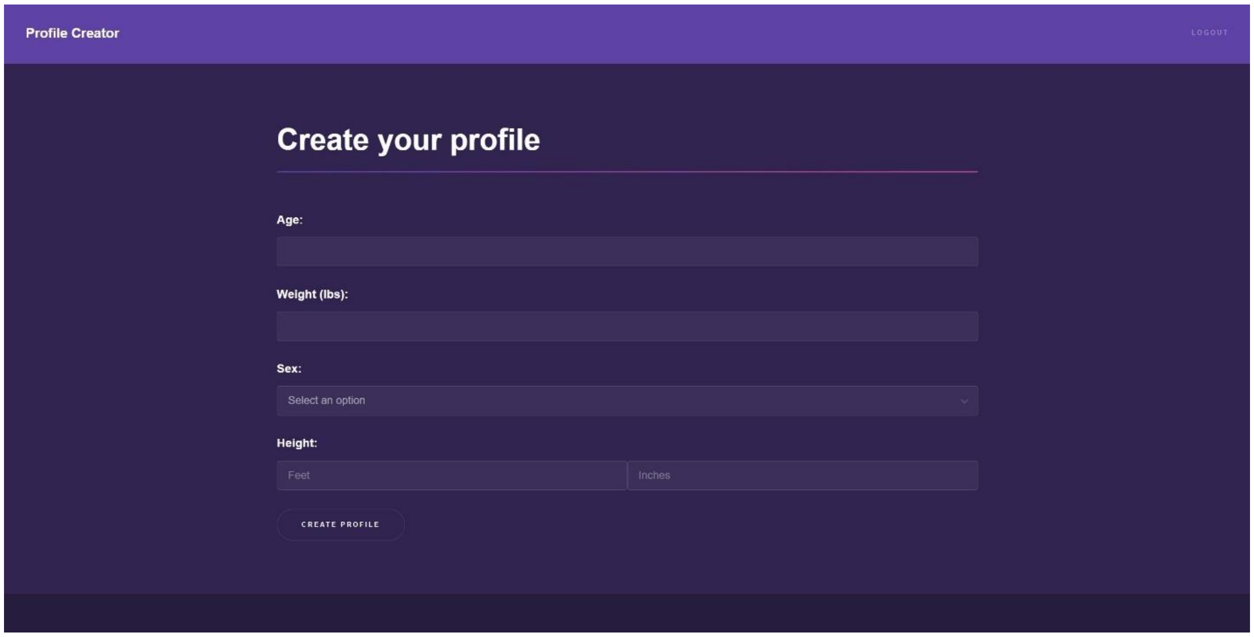
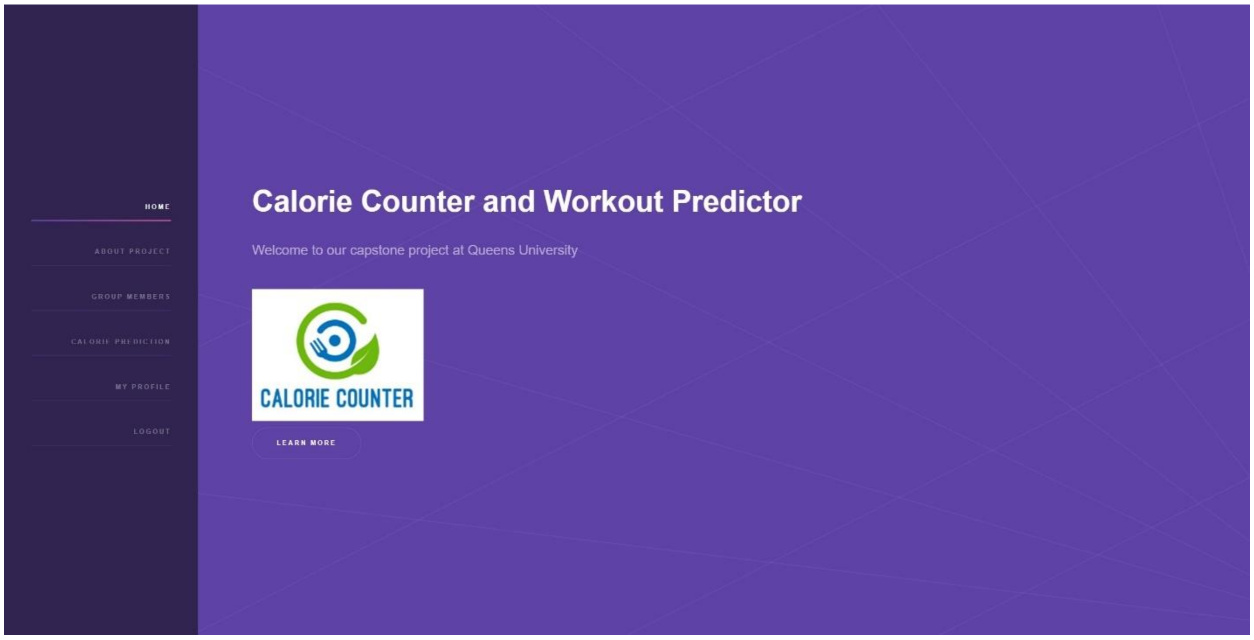


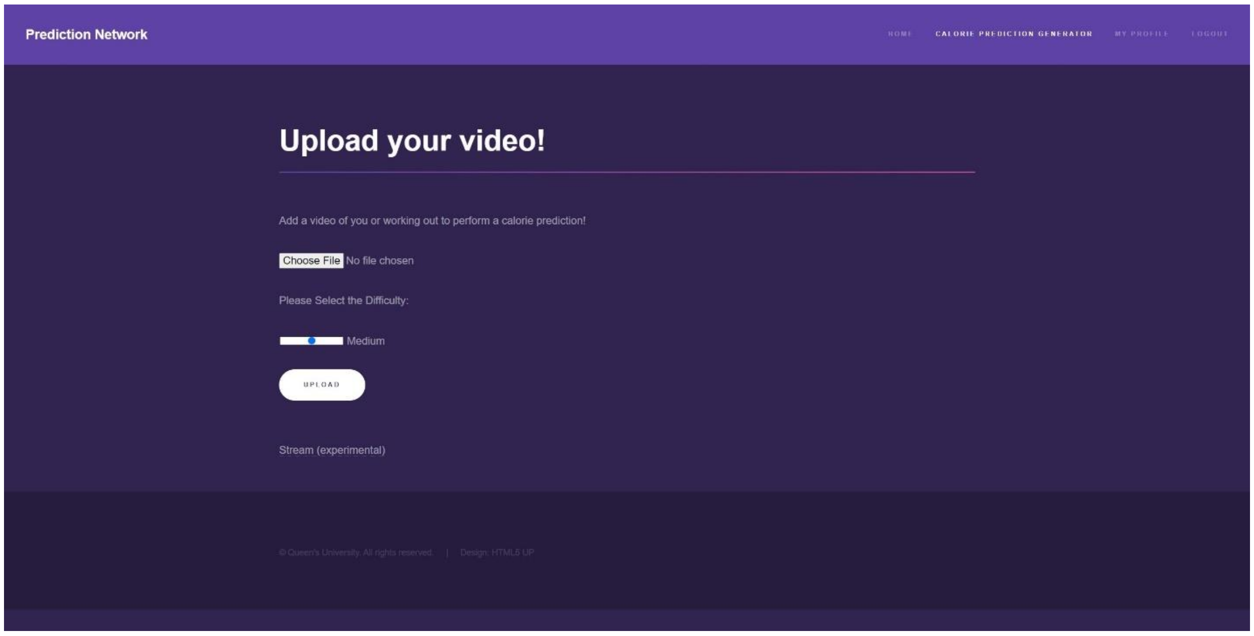
Figure 24: GUI Profile Creation page



Calorie Counter and Workout Predictor

Welcome to our capstone project at Queens University

Figure 25: GUI Home page



Upload your video!

Add a video of you or working out to perform a calorie prediction!

Choose File No file chosen

Please Select the Difficulty:

Medium

Stream (experimental)

© Queen's University. All rights reserved. | Design: HTML5 UP

Figure 26: GUI Video Upload page