Final Report

Human Activity Recognition

Aaron Finlay, C00226131, 4th year Soft Eng

Supervisor: Greg Doyle

Institiúid Teicneolaíochta Cheatharlach
INSTITUTE of TECHNOLOGY CARLOW
At the Heart of South Leinster

Abstract

Human Activity Recognition (HAR), is aimed to be a web application with the capability of detecting falls, alerting that the fall took place. The primary objective of the project is to detect falls, as accurately as possible.

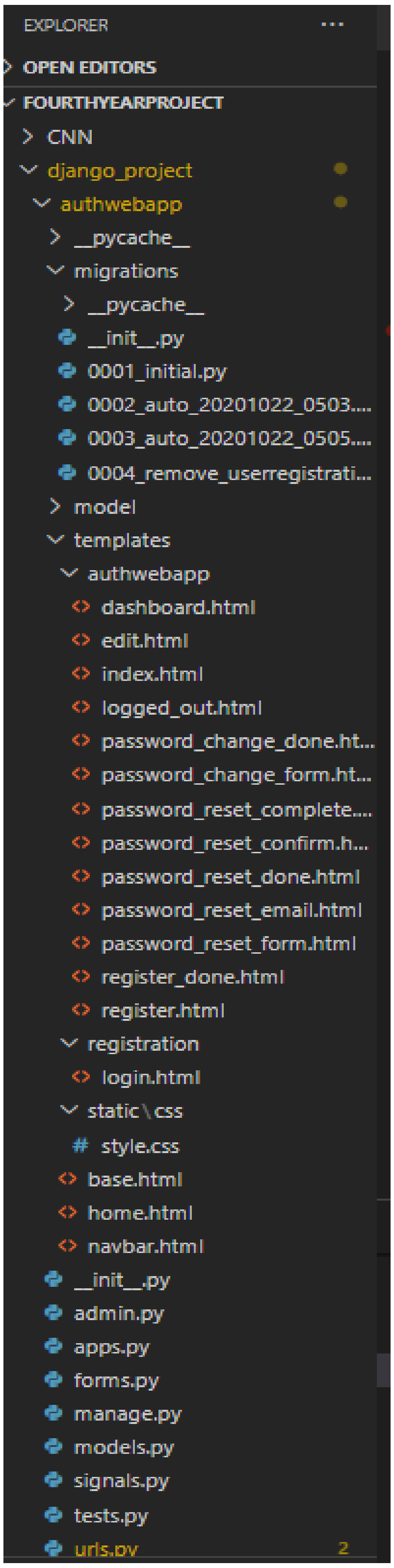
Table of Contents

Introduction

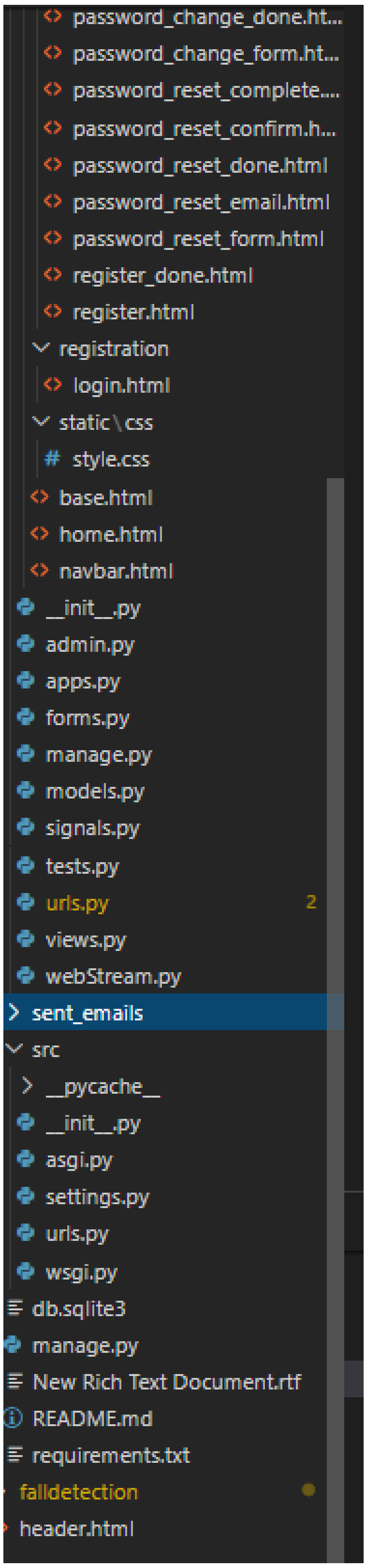
This document describes the processes involved during development of the Human Activity Recognition (HAR) system and web application. It is a web app, machine learning project which focuses on the betterment and safety of elderly people's lives through effective detection of falls and alerting of when the fall occurred. There is an unprecedented amount of value that can be added to the quality and safety of elderly people. Independence of wearable sensors, in a shift towards deep learning through vision based approaches such as a webcam or remote camera, allows for elderly people to act independently from wearing sensors. Furthermore, with a high accuracy, it means the software to a certain extent will be reliable and will pick up on falls which the system can later act on, e.g., triggering an alert. The software was developed using the webcam as a form of trial and error when testing the capability of the system in terms of detecting falls. Haar cascade classifier was, in the end, ultimately used to detect falls. Although Convolutional Neural Networks (CNN) was initially researched and adopted, it proved to be difficult to develop, as many issues occurred during the development of the algorithm. These issues will be later described in bigger detail.

Project Description

Project Layout



```
EXPLORER ...
> OPEN EDITORS
v FOURTHYEARPROJECT
  > CNN
  v django_project
    v authwebapp
      > __pycache__
      v migrations
        > __pycache__
        __init__.py
        0001_initial.py
        0002_auto_20201022_0503....
        0003_auto_20201022_0505....
        0004_remove_userregistrati...
      > model
      v templates
        v authwebapp
          dashboard.html
          edit.html
          index.html
          logged_out.html
          password_change_done.ht...
          password_change_form.ht...
          password_reset_complete....
          password_reset_confirm.h...
          password_reset_done.html
          password_reset_email.html
          password_reset_form.html
          register_done.html
          register.html
        v registration
          login.html
        v static\css
          style.css
        base.html
        home.html
        navbar.html
      __init__.py
      admin.py
      apps.py
      forms.py
      manage.py
      models.py
      signals.py
      tests.py
      urls.py                    2
```



```
password_change_done.ht...
password_change_form.ht...
password_reset_complete....
password_reset_confirm.h...
password_reset_done.html
password_reset_email.html
password_reset_form.html
register_done.html
register.html
registration
login.html
static \ css
style.css
base.html
home.html
navbar.html
__init__.py
admin.py
apps.py
forms.py
manage.py
models.py
signals.py
tests.py
urls.py                    2
views.py
webStream.py
sent_emails
src
__pycache__
__init__.py
asgi.py
settings.py
urls.py
wsgi.py
db.sqlite3
manage.py
New Rich Text Document.rtf
README.md
requirements.txt
falldetection
header.html
```

Fall Detection

Success and Drawbacks

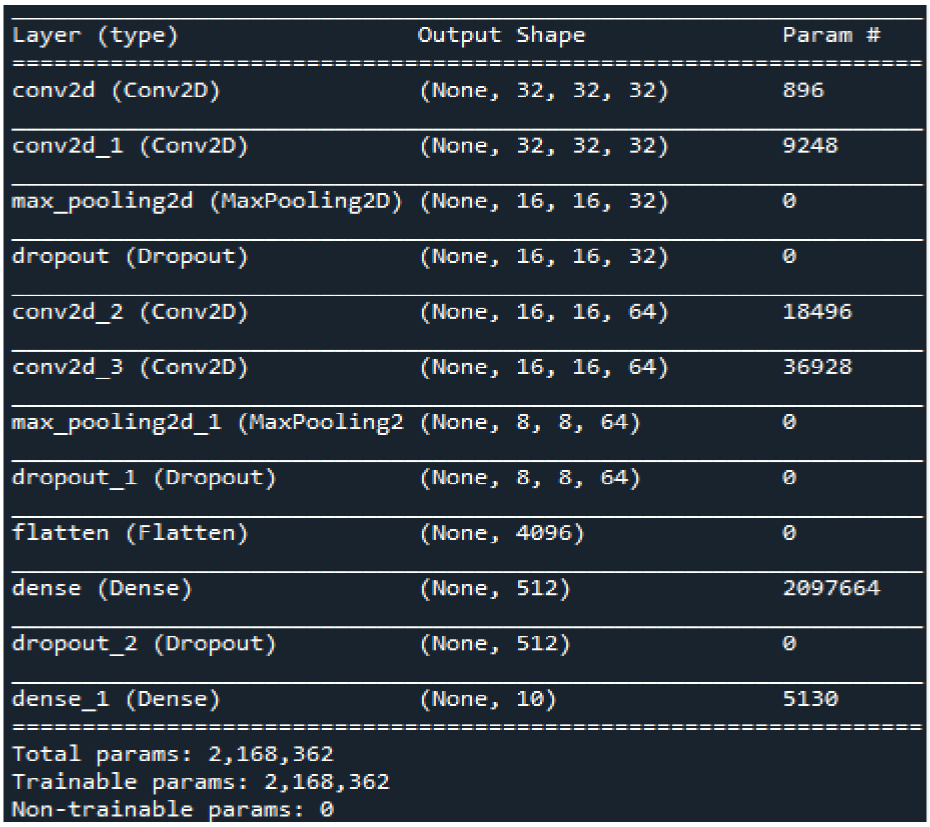
There have been some key things to note throughout the development of this project. There have been some setbacks and some successes. It was hoped that using convolutional neural networks, a fall may be detected, in real-time via webcam, within the web application proposed.

Unfortunately the CNN proved to be unsuccessful in detecting falls and although this can be considered a failure, ultimately, a fall was detected using the Haar Cascade Classifier approach. This was not what was intended, but upon realizing it was risky sticking to only CNN, one had to adapt in the moment and begin developing the python-opencv script which harnesses the haar cascade classifier.

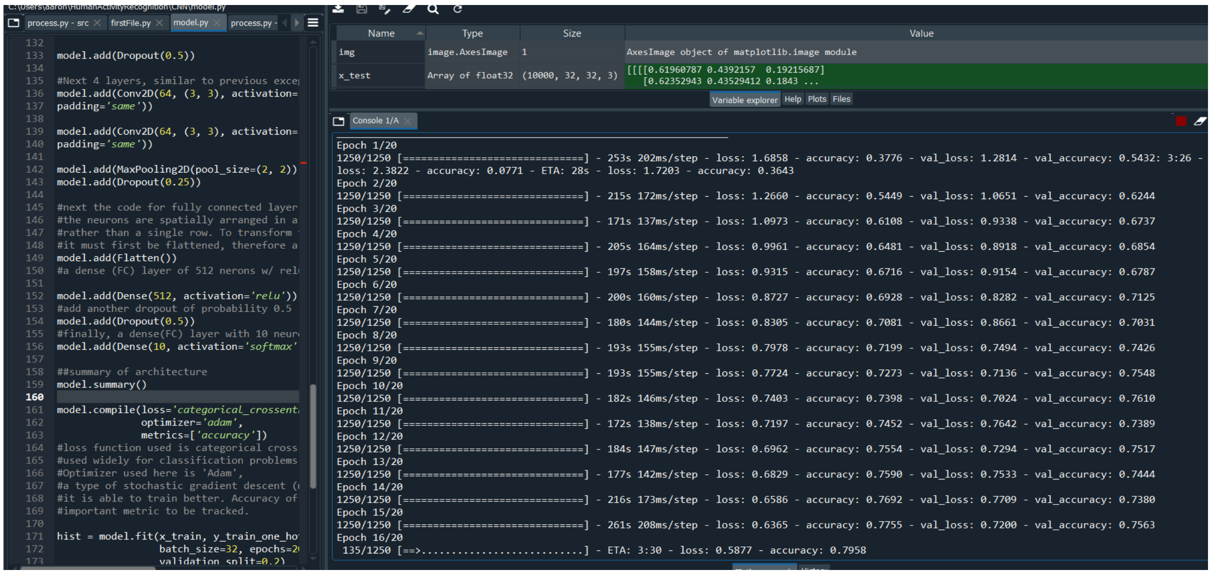
The implementation was capable of detecting falls, however, it is important to note that when aiming to detect a fall when the camera is aimed at the side of you, it is a lot less accurate in predictions. Furthermore, it did prove to generate some false positives, but to a reasonable degree, was also accurate in detecting falls. As for the web application, it was designed successfully utilising online documentation, variety of sources, however it proved difficult to render the machine learning fall detection within the web application.

Some key factors to note were, falling sick in the last few weeks, realising the CNN was not going to work too late, and endless software bugs and machine issues which led to a slow, unperformant machine to develop off of for some time. The fall detection was operating as was the web app, but they were not integrated, therefore making it both partially successful and failure.

It is important to note that for developing the cnn, while it proved unsuccessful in ultimately detecting falls, some considerable results were achieved while using cnn the first time ever with trial and error, alongside research. Some images below will illustrate the architecture and output of the neural network(s) implemented. See below.



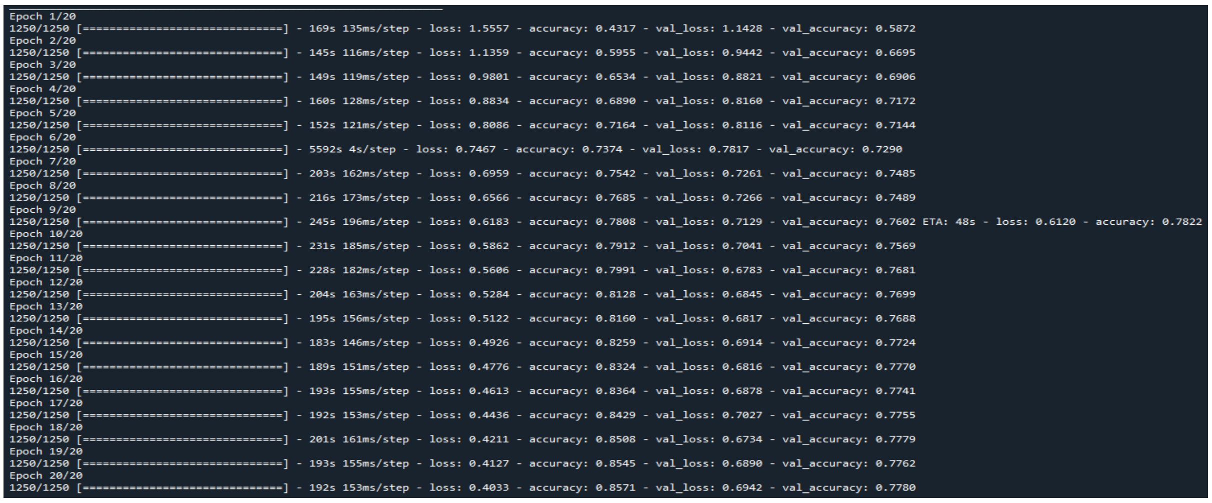
```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 32)        896

conv2d_1 (Conv2D)            (None, 32, 32, 32)        9248

max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0

dropout (Dropout)            (None, 16, 16, 32)        0

conv2d_2 (Conv2D)            (None, 16, 16, 64)        18496

conv2d_3 (Conv2D)            (None, 16, 16, 64)        36928

max_pooling2d_1 (MaxPooling2 (None, 8, 8, 64)          0

dropout_1 (Dropout)          (None, 8, 8, 64)          0

flatten (Flatten)            (None, 4096)              0

dense (Dense)                (None, 512)               2097664

dropout_2 (Dropout)          (None, 512)               0

dense_1 (Dense)              (None, 10)                5130
=================================================================
Total params: 2,168,362
Trainable params: 2,168,362
Non-trainable params: 0
```



```
model.add(Dropout(0.5))

#Next 4 layers, similar to previous excep
model.add(Conv2D(64, (3, 3), activation=
padding='same'))

model.add(Conv2D(64, (3, 3), activation=
padding='same'))

model.add(MaxPooling2D(pool_size=(2, 2))
model.add(Dropout(0.25))

#next the code for fully connected layer
#the neurons are spatially arranged in a
#rather than a single row. To transform
#it must first be flattened, therefore a
model.add(Flatten())
#a dense (FC) layer of 512 nerons w/ rel

model.add(Dense(512, activation='relu'))
#add another dropout of probability 0.5
model.add(Dropout(0.5))
#finally, a dense(FC) layer with 10 neur
model.add(Dense(10, activation='softmax'

##summary of architecture
model.summary()

model.compile(loss='categorical_crossent
              optimizer='adam',
              metrics=['accuracy'])
#loss function used is categorical cross
#used widely for classification problems
#Optimizer used here is 'Adam',
#a type of stochastic gradient descent (
#it is able to train better. Accuracy of
#important metric to be tracked.

hist = model.fit(x_train, y_train_one_ho
                 batch_size=32, epochs=2
                 validation_split=0.2)
```

Name	Type	Size	Value
img	image.AxesImage	1	AxesImage object of matplotlib.image module
x_test	Array of float32	(10000, 32, 32, 3)	[[[[0.61960787 0.4392157 0.19215687] [0.62352943 0.43529412 0.1843 ...

```
Epoch 1/20
1250/1250 [==============================] - 253s 202ms/step - loss: 1.6858 - accuracy: 0.3776 - val_loss: 1.2814 - val_accuracy: 0.5432: 3:26 -
loss: 2.3822 - accuracy: 0.0771 - ETA: 28s - loss: 1.7203 - accuracy: 0.3643
Epoch 2/20
1250/1250 [==============================] - 215s 172ms/step - loss: 1.2660 - accuracy: 0.5449 - val_loss: 1.0651 - val_accuracy: 0.6244
Epoch 3/20
1250/1250 [==============================] - 171s 137ms/step - loss: 1.0973 - accuracy: 0.6108 - val_loss: 0.9338 - val_accuracy: 0.6737
Epoch 4/20
1250/1250 [==============================] - 205s 164ms/step - loss: 0.9961 - accuracy: 0.6481 - val_loss: 0.8918 - val_accuracy: 0.6854
Epoch 5/20
1250/1250 [==============================] - 197s 158ms/step - loss: 0.9315 - accuracy: 0.6716 - val_loss: 0.9154 - val_accuracy: 0.6787
Epoch 6/20
1250/1250 [==============================] - 200s 160ms/step - loss: 0.8727 - accuracy: 0.6928 - val_loss: 0.8282 - val_accuracy: 0.7125
Epoch 7/20
1250/1250 [==============================] - 180s 144ms/step - loss: 0.8305 - accuracy: 0.7081 - val_loss: 0.8661 - val_accuracy: 0.7031
Epoch 8/20
1250/1250 [==============================] - 193s 155ms/step - loss: 0.7978 - accuracy: 0.7199 - val_loss: 0.7494 - val_accuracy: 0.7426
Epoch 9/20
1250/1250 [==============================] - 193s 155ms/step - loss: 0.7724 - accuracy: 0.7273 - val_loss: 0.7136 - val_accuracy: 0.7548
Epoch 10/20
1250/1250 [==============================] - 182s 146ms/step - loss: 0.7403 - accuracy: 0.7398 - val_loss: 0.7024 - val_accuracy: 0.7610
Epoch 11/20
1250/1250 [==============================] - 172s 138ms/step - loss: 0.7197 - accuracy: 0.7452 - val_loss: 0.7642 - val_accuracy: 0.7389
Epoch 12/20
1250/1250 [==============================] - 184s 147ms/step - loss: 0.6962 - accuracy: 0.7554 - val_loss: 0.7294 - val_accuracy: 0.7517
Epoch 13/20
1250/1250 [==============================] - 177s 142ms/step - loss: 0.6829 - accuracy: 0.7590 - val_loss: 0.7533 - val_accuracy: 0.7444
Epoch 14/20
1250/1250 [==============================] - 216s 173ms/step - loss: 0.6586 - accuracy: 0.7692 - val_loss: 0.7709 - val_accuracy: 0.7380
Epoch 15/20
1250/1250 [==============================] - 261s 208ms/step - loss: 0.6365 - accuracy: 0.7755 - val_loss: 0.7200 - val_accuracy: 0.7563
Epoch 16/20
 135/1250 [==>...........................] - ETA: 3:30 - loss: 0.5877 - accuracy: 0.7958
```



```
Epoch 1/20
1250/1250 [==============================] - 169s 135ms/step - loss: 1.5557 - accuracy: 0.4317 - val_loss: 1.1428 - val_accuracy: 0.5872
Epoch 2/20
1250/1250 [==============================] - 145s 116ms/step - loss: 1.1359 - accuracy: 0.5955 - val_loss: 0.9442 - val_accuracy: 0.6695
Epoch 3/20
1250/1250 [==============================] - 149s 119ms/step - loss: 0.9801 - accuracy: 0.6534 - val_loss: 0.8821 - val_accuracy: 0.6906
Epoch 4/20
1250/1250 [==============================] - 160s 128ms/step - loss: 0.8834 - accuracy: 0.6890 - val_loss: 0.8160 - val_accuracy: 0.7172
Epoch 5/20
1250/1250 [==============================] - 152s 121ms/step - loss: 0.8086 - accuracy: 0.7164 - val_loss: 0.8116 - val_accuracy: 0.7144
Epoch 6/20
1250/1250 [==============================] - 5592s 4s/step - loss: 0.7467 - accuracy: 0.7374 - val_loss: 0.7817 - val_accuracy: 0.7290
Epoch 7/20
1250/1250 [==============================] - 203s 162ms/step - loss: 0.6959 - accuracy: 0.7542 - val_loss: 0.7261 - val_accuracy: 0.7485
Epoch 8/20
1250/1250 [==============================] - 216s 173ms/step - loss: 0.6566 - accuracy: 0.7685 - val_loss: 0.7266 - val_accuracy: 0.7489
Epoch 9/20
1250/1250 [==============================] - 245s 196ms/step - loss: 0.6183 - accuracy: 0.7808 - val_loss: 0.7129 - val_accuracy: 0.7602 ETA: 48s - loss: 0.6120 - accuracy: 0.7822
Epoch 10/20
1250/1250 [==============================] - 231s 185ms/step - loss: 0.5862 - accuracy: 0.7912 - val_loss: 0.7041 - val_accuracy: 0.7569
Epoch 11/20
1250/1250 [==============================] - 228s 182ms/step - loss: 0.5606 - accuracy: 0.7991 - val_loss: 0.6783 - val_accuracy: 0.7681
Epoch 12/20
1250/1250 [==============================] - 204s 163ms/step - loss: 0.5284 - accuracy: 0.8128 - val_loss: 0.6845 - val_accuracy: 0.7699
Epoch 13/20
1250/1250 [==============================] - 195s 156ms/step - loss: 0.5122 - accuracy: 0.8160 - val_loss: 0.6817 - val_accuracy: 0.7688
Epoch 14/20
1250/1250 [==============================] - 183s 146ms/step - loss: 0.4926 - accuracy: 0.8259 - val_loss: 0.6914 - val_accuracy: 0.7724
Epoch 15/20
1250/1250 [==============================] - 189s 151ms/step - loss: 0.4776 - accuracy: 0.8324 - val_loss: 0.6816 - val_accuracy: 0.7770
Epoch 16/20
1250/1250 [==============================] - 193s 155ms/step - loss: 0.4613 - accuracy: 0.8364 - val_loss: 0.6878 - val_accuracy: 0.7741
Epoch 17/20
1250/1250 [==============================] - 192s 153ms/step - loss: 0.4436 - accuracy: 0.8429 - val_loss: 0.7027 - val_accuracy: 0.7755
Epoch 18/20
1250/1250 [==============================] - 201s 161ms/step - loss: 0.4211 - accuracy: 0.8508 - val_loss: 0.6734 - val_accuracy: 0.7779
Epoch 19/20
1250/1250 [==============================] - 193s 155ms/step - loss: 0.4127 - accuracy: 0.8545 - val_loss: 0.6890 - val_accuracy: 0.7762
Epoch 20/20
1250/1250 [==============================] - 192s 153ms/step - loss: 0.4033 - accuracy: 0.8571 - val_loss: 0.6942 - val_accuracy: 0.7780
```

Development Challenges

It proved difficult developing the project with the current machine, but due to covid and due to financial constraints, it had to be used. The machine operated on windows 10, using anaconda as a virtual environment to enable development utilising important libraries such as opencv and django inside the conda prompt. Windows is a very hard platform to develop machine learning in one's opinion as much material on the web is in linux based development leading to configuration, environment and package issues, e.g. conda, pip dependency errors.

Some packages that are integrated with others would be developed with one manager, while the others would be installed with the other leading to misconfigurations. It took a long time to be able to do a checklist of what is required to even begin to work at a machine learning source code project because of the amount of importing typically required, installations, and then the various steps involved in data preprocessing which, in itself, is only one step of several in the life-cycle of a machine learning algorithm, e.g. CNN.

Learning Outcomes

Most notably, the information gained around machine learning, fall detection, human activity recognition and all the technologies involved, is a huge learning outcome for one. The areas in which learning was mostly identified will be detailed below.

Convolutional Neural Network

One of the greatest outcomes of studying and developing convolutional neural networks (CNN) was the insight it brought into machine learning. Learning what a convolutional neural network is, led to a different view on how to handle and manipulate data since there was no prior experience. CNN can be considered to be a one or more layered neural network that is used mostly for image classification, segmentation, feature extraction and image processing. The convolution can be considered to be an input which is filtered, it can be considered to be viewed as operating in such a way that it views a function's environment and surroundings, in order to make a reasonably accurate prediction of the task outcome.

It breaks down images by focusing on sub components of images such as vertical edges and lines, which develop a pattern that is analysed based on the data the algorithm was predisposed to before, therefore, CNN can accurately predict the outcome of the classification task. How this detective-based approach of analysing images in one layer affect the other is that the function known as the "activation" function is responsible for neurons firing up throughout the network and each activation of a neuron impacts the next layer's neurons activations based of what was previously discussed, i.e. predicting similar patterns, e.g. vertical edges in an image to enhance prediction quality.

From the moment the image meets the input (1st) layer of the neural network, the neurons in each layer will impact the outcome of the network, based on which activate according to likelihood of image prediction. In order to better handle the data through training the neural network, the dropout should be set 0.5 so that in every second pass through the network data will be lost, allowing for better admission of data transfer into the network. This prevents overfitting the network during training, on all the training data. This particular subject occurred more than once. It is where the accuracy is high based on the training, but it performs well on the test/unseen data. Another occurrence was where the accuracy was high but so was the loss, which means the algorithm can accurately predict an image but lacks confidence in it's prediction, therefore this is a sign of overfitting and the model will likely perform even worse when validating both loss and accuracy.

Each convolutional layer stores a variety of filters called convolutional kernels. Moreover, the filter, which is a matrix of integers, which are also a subset of the input pixel values, is the same size as the kernel. All of the pixels are multiplied by each entry value within the kernel, which the result then becomes summed up for the single value which is responsible for representing something simple, perhaps a grid cell, or maybe a pixel, which is then the designated output feature map.

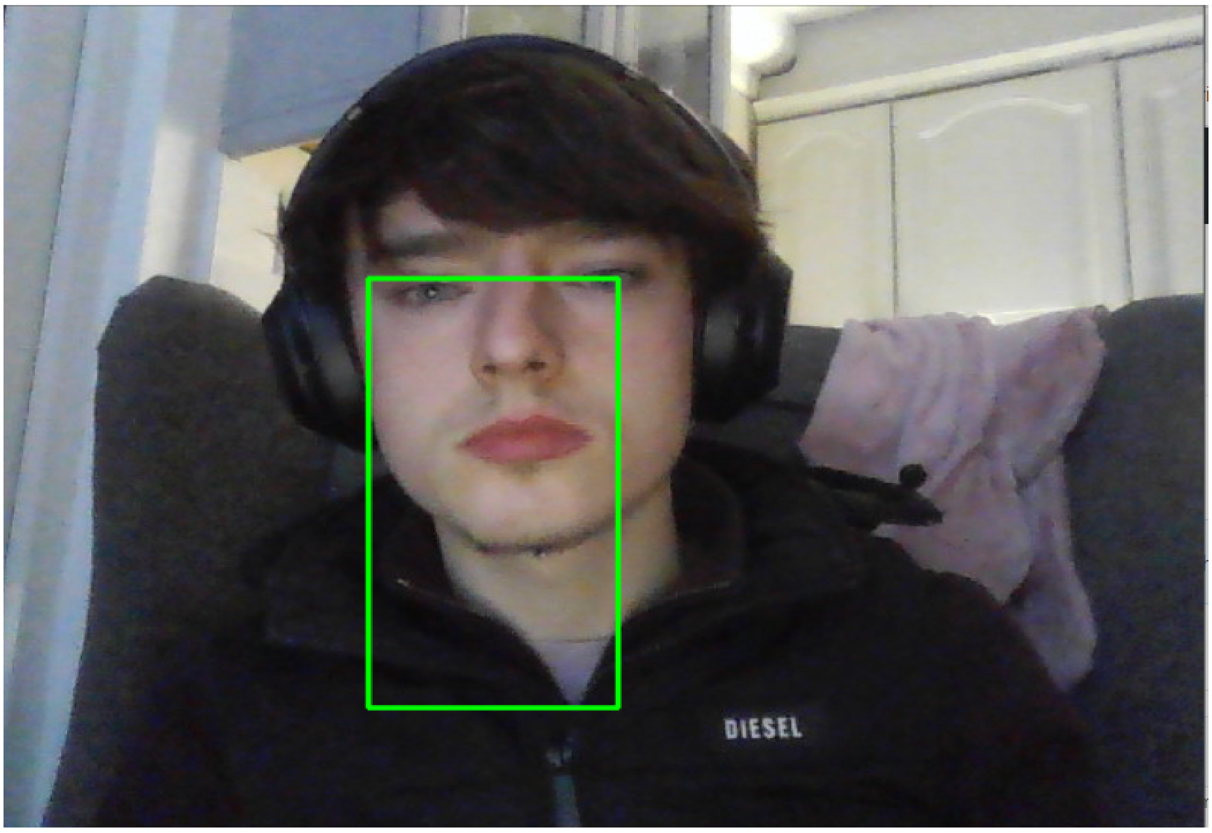
Haar Cascade Classifier

Haar Cascade Classifier turned out to be a paramount technology used within the end of the project life-cycle. Before two months ago, one would not have known much about this area, however upon reading some research papers one noticed a common citation which is Paul Viola and Michael Jones in their paper, “Rapid Object Detection using a Boosted Cascade of Simple Features” in 2001. This seemed to be an effective approach to object detection, so one figured why not harness the raw power of this modality and use it to detect falls. This is a machine learning based approach where a cascade function is being trained from a multitude of images all of which are positive and negative. A positive image would be where a fall has occurred, i.e. it’s a positive if we detect a fall. A negative can be considered as a no fall, which is not predicting a fall, therefore implying it will be negative.

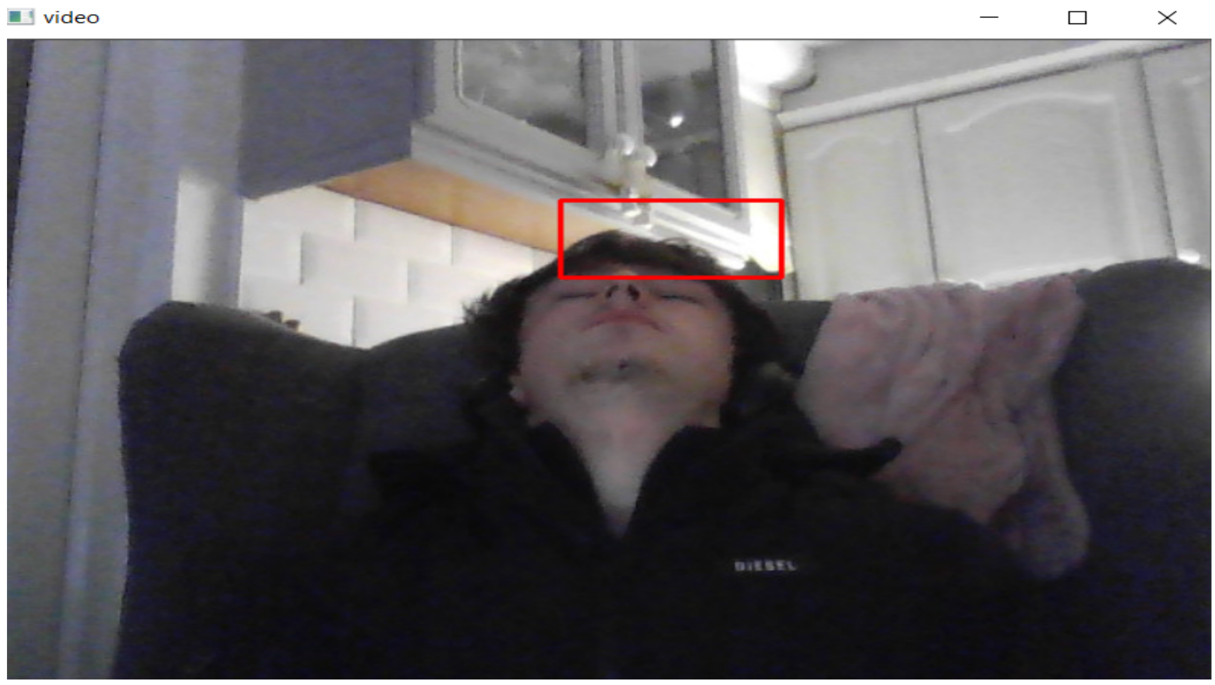
Theory behind HAR

Learning the theory behind HAR by starting at the basic level helped understand and identify the key components behind HAR systems. Such basic steps include learning that an action may be considered like a sequence of primitive actions which serve a purpose, for example, running or falling. Moreover, an activity can be viewed as containing sequences of actions over the duration of their spatio-temporal relationship. Another interesting thing is that every activity generally has a relation between at least one or more people including objects in the immediate vicinity.

Fall Detection Results

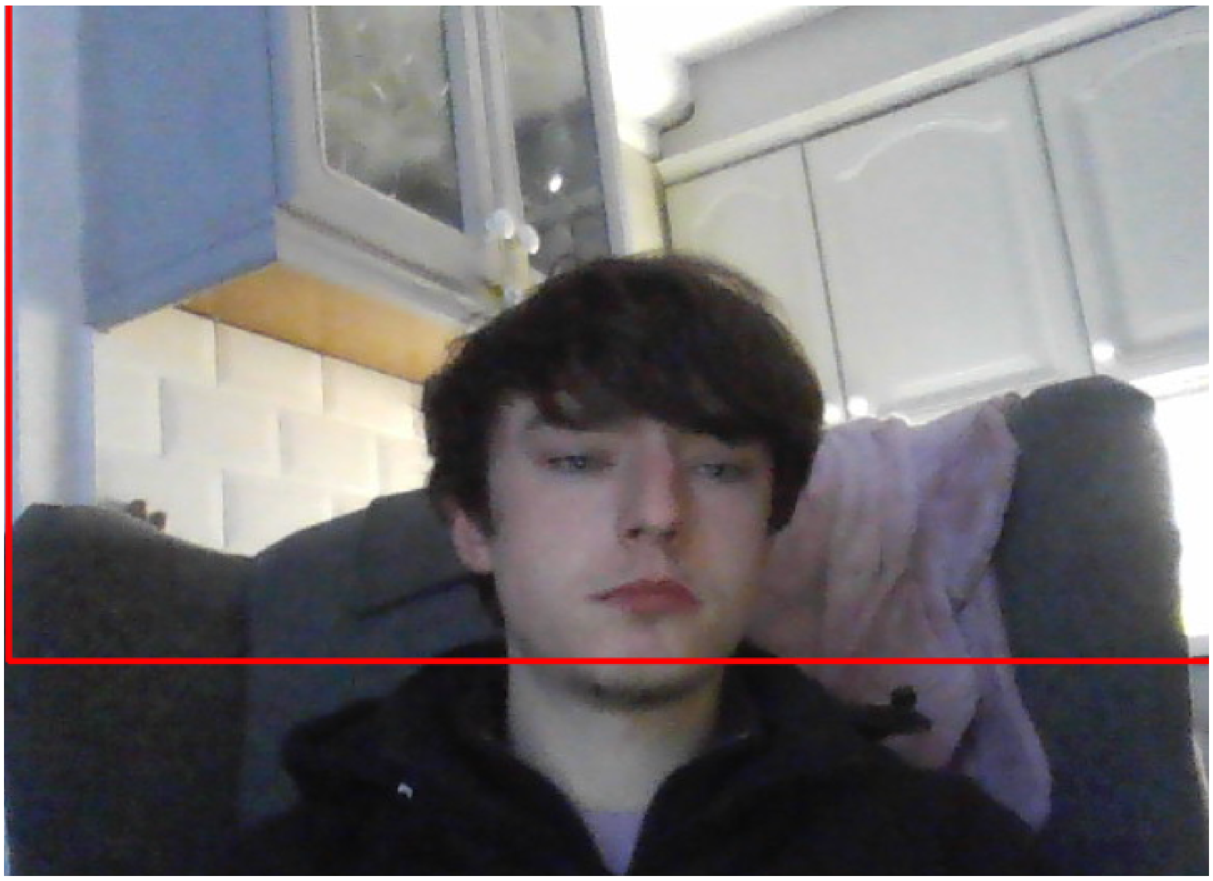


The above photo illustrates the frame detecting the upper body and storing the frame onto the spatial part.



The above shows the frame red, which signals a fall has occurred, which in this instance did as one fell his head backwards upon having it register to the frames on the webcam. This was successful.

Finally, it’s important to showcase the false negative and to explain why this happened.



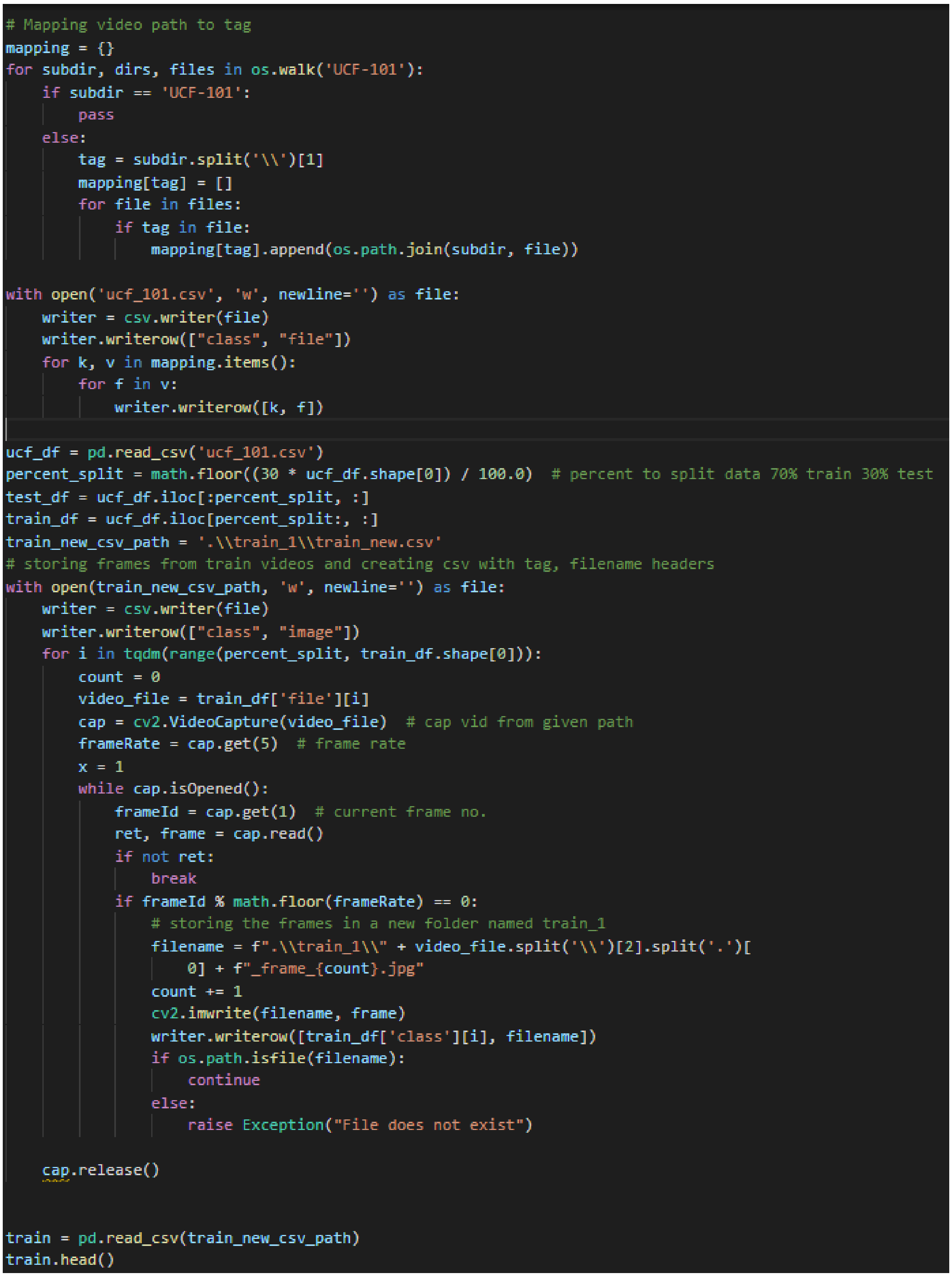
The main issue identified with the haar cascade classifier is that the frames latch onto the body part they see and focus on it and at times, if bad environment issues, e.g. lighting, occlusion, etc. the classifier will detect the slightest move of that body part and will trick the classifier into thinking it’s a fall. This then produces the false positive, and although this occurred a couple of times, it was more so successful in generating true positives.

Programming Languages

Using Python in both the web application development and the numerous implementations used in the machine learning project, led to a better level of experience and quality with python, making me feel more competent and comfortable using python. Python truly is one's favourite language and it is fun to code in python, along with identifying with the various interesting arsenal of tools and technologies that are both built into python and easily extendable to python packages.

For example, Django has plenty of cool built in features such as CBV (class based views) and FBVs (functional based views), not to mention the in-built library for django to quickly develop web applications. Flask on the other hand, is a microframework with little built-in functionalities, but its key difference is that it is easily modifiable and extensible, so that other technologies outside of flasks' toolbox are accessible and retrievable.

Data Pre-Processing



```python
# Mapping video path to tag
mapping = {}
for subdir, dirs, files in os.walk('UCF-101'):
    if subdir == 'UCF-101':
        pass
    else:
        tag = subdir.split('\\')[1]
        mapping[tag] = []
        for file in files:
            if tag in file:
                mapping[tag].append(os.path.join(subdir, file))

with open('ucf_101.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(["class", "file"])
    for k, v in mapping.items():
        for f in v:
            writer.writerow([k, f])

ucf_df = pd.read_csv('ucf_101.csv')
percent_split = math.floor((30 * ucf_df.shape[0]) / 100.0)  # percent to split data 70% train 30% test
test_df = ucf_df.iloc[:percent_split, :]
train_df = ucf_df.iloc[percent_split:, :]
train_new_csv_path = '.\\train_1\\train_new.csv'
# storing frames from train videos and creating csv with tag, filename headers
with open(train_new_csv_path, 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(["class", "image"])
    for i in tqdm(range(percent_split, train_df.shape[0])):
        count = 0
        video_file = train_df['file'][i]
        cap = cv2.VideoCapture(video_file)  # cap vid from given path
        frameRate = cap.get(5)  # frame rate
        x = 1
        while cap.isOpened():
            frameId = cap.get(1)  # current frame no.
            ret, frame = cap.read()
            if not ret:
                break
            if frameId % math.floor(frameRate) == 0:
                # storing the frames in a new folder named train_1
                filename = f".\\train_1\\" + video_file.split('\\')[2].split('.')[
                    0] + f"_frame_{count}.jpg"
                count += 1
                cv2.imwrite(filename, frame)
                writer.writerow([train_df['class'][i], filename])
                if os.path.isfile(filename):
                    continue
                else:
                    raise Exception("File does not exist")

    cap.release()


train = pd.read_csv(train_new_csv_path)
train.head()
```

The above code snippets illustrate mapping the path to the video tags which are stored in text files that were transferred into a data frame. A key step in data preprocessing which is distinct in video classification from image classification, is that the frames need to be extracted before undergoing the procedural steps of data preprocessing. The frames along with the labels and classes will be stored in a csv file. This gives shape and then using numpy array, flatten the data and then print the shape to confirm the array is what size it should be based on the data being fed. Most of the data preprocessing hard work is done outside of that code snippet, but in this instance, with the data involved, the extraction of frames proved one of the lengthiest steps.

<u>Personal Learning</u>

On a personal note, the project led one to have a big interest in machine learning, however it can be thought that what one thought of the area prior to embarking on the project was naive compared to experiencing a project full of machine learning and web development. It thought one at times that with projects as lengthy as this, it is important to have a plan set out always, so that rather than being mind-bogged with the complexity of the entirety of a project, which in this case felt like two (machine learning and web app), one can analyze the sub-components individually so that one task can be effectively conquered at a time leading to higher results due to less context switching. Unfortunately this skill wasn't learned immediately but it helped using this approach as it was needed to adapt and overcome the various deadlines and difficulties with not just the project but all work.

This was the first project at a magnitude of this scale undergone by one, so it was definitely a challenge and being disciplined was something one learned, at times the hard way, other times consistently. Time management is a key variable to learn when planning and developing projects. At first one would read a lot to learn but forget to document learnings, which meant time being wasted due to forgetting that you covered such a topic but never documented and you go back in a loop with information.

One always studied or did work by time spent, but this year, there were times were hours on end would be spent in front of the screen stuck on bugs, and it felt as though no work had been accomplished and time was wasted. This happened a lot with some tedious bugs. This is when one realised that planning work to be done based on time was not gonna work, but to measure work by work done, which meant some days what could be achieved in one hour, may have only been achieved in four another.

Project Evaluation and Conclusion

There were a number of objectives outlined and implementations undergone. It was initially proposed the fall detection would be harnessing the power of a convolutional neural network, however, come the end of the product life-cycle, it was actually an OpenCV - HAAR Cascade classification modality that was utilized. It was capable of detecting falls, however it is important to note, there were limitations with this technique. Whilte it was performant in detecting falls at a good angle point, with good lighting and contrast, it proved harder to recognize a fall when the candidate fell sideways. It is also important to note there were no false negatives and false positives within the application of the fall detector. Some images displaying results will be attached below.

What went wrong?

Upon initially taking on the project, there was a large amount of research that had to be done, leaving implementation of code later than one expected. After the first demonstration of the first iteration of the product, it was feared, following the feedback, that the project may not be enough, in terms of functionality. This was immediately number one then on the priority along with several other assignments. The web application was a lengthy process building as one has only ever had such experience with django and python once, last year.

Upon initially gathering the dataset, it was proven difficult to find a suitable data source containing videos, and since the deadline for the iteration one began to proceed, a proof of concept of the CNN working on separate data, with separate labels and classes was performed. This succeeded achieving desirable results, however, it was with unrelated data to the specification of the project. This then led to increased pressure of both working on the web app after christmas alongside the machine learning project.

The web app was eventually created in time for the second iteration/demo but this was not what one should have prioritized, as there was not a functionality improvement ready for the detection of falls. After this, the fall detection took precedence, however along with other assignments and continued pressure, it proved too difficult to make the convolutional neural network work with the data found for falls. This led to a quick implementation following a multitude of research and tutorials, of a haar cascade approach which utilizes rapid object detection, and it made it possible, following a opencv python script developed, linked with the webcam, to detect the fall.

Unfortunately, the last two weeks before the deadline, one got sick and this had a tremendous impact on the ability to do work which led to increased time constraints and pressure. Bugs proved to be a major issue in this project, since the machine worked on used windows 10, and a lot of material available is done on a Linux OS, this meant anaconda had to be adopted which took a while getting used to. Environmental issues such as which conda environment, or path issues proved to be another major issue and also package dependencies. Some packages had been installed with pip, while others with conda, this was before one knew about the consequences of using varied package managers. This seriously halted the development of machine learning and web application development because most of the time was spent dealing with uninstalling, installing, identifying root cause errors and not to mention the notorious ModuleNotFoundError.

A major halt in the progress was not understanding the requirements clearly enough earlier on, this led to changes throughout the project, which contributed to self doubt and a sense of misdirection in terms of where the project is heading, from what was originally anticipated.

Working on both the web application and machine learning project felt like working on two projects and meant having to shift my resources and time on to each front. This at times, led to periods where one took precedent and the other was not looked at, which contributed to a poorer process of development for the CNN.

Future Research/Work

Machine learning is something that interests one very much and despite the setbacks, and not achieving the fall detection within the web app, it was a fun, insightful experience and proved to be very difficult. If one were to embark on a project again, or even into the work environment or especially research, this would be very interesting and something one would be happy to do.

Convolutional neural networks were a very interesting algorithm to study, it proved to be a powerful algorithm, but also one that requires lots of baby steps before achieving desirable results. Although the majority of this project's duration was spent researching and developing convolutional neural networks, one has still only scratched the surface and this is a topic one would be interested in becoming an expert in.

Some good reputable sources, mentioned by a lecturer, which helped in learning about convolutional neural networks was Andrew Ng. His youtube videos proved to be very good at illustrating both with code examples and in a conceptual way, what a CNN is, how it works, structure, hyper parameterization and evaluation of the model. Other sources proved to be good quality at sites such as towardsdatascience, machinelearningmastery and some good datasets at kaggle. These are all sources one will continue to pursue following completion of fourth year as this is an area one finds immersive.

Plagiarism Declaration

INSTITUTE of TECHNOLOGY CARLOW

Institiúid Teicneolaíochta Cheatharlach

*I declare that all material in this submission e.g. thesis/essay/project/assignment is entirely my/our own work except where duly acknowledged.

*I have cited the sources of all quotations, paraphrases, summaries of information, tables, diagrams or other material; including software and other electronic media in which intellectual property rights may reside.

*I have provided a complete bibliography of all works and sources used in the preparation of this submission.

*I understand that failure to comply with the Institute's regulations governing plagiarism constitutes a serious offence.

Student Name: (Printed) AARON FINLAY

Student Number(s): C00226131

Signature(s): Aaron Finlay

Date: 30th of April, 2021