Technical Manual

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

Created with EDIT.org

Abstract

Human Activity Recognition (HAR) is a machine learning algorithm capable of detecting falls in real-time via OpenCV and webcam, which the cam is hosted on a user authenticated django web application. The main objective of the project is to detect a fall, if possible within the web application. This document is the technical manual

Table of Contents

What is HAR?

Human Activity Recognition (HAR) is a fourth year student project that is designed to detect when people fall. It is sought to be wrapped into a web application upon completion of detecting falls. It will be user authenticated, allowing only logged in users access to the webcam interface on a webpage when wishing to try detect a fall or monitor.

Code Structure

For the code to be presented, some of it was created entirely, meanwhile other parts have been adapted from a multitude of sources researched while developing the project. The folder - fallDetection is the main folder for attempted convolutional neural network progress made to date. Any sources that were helped in developing the system were cited in the code files and also in the readme file attached to the code directory.

HAR Requirements

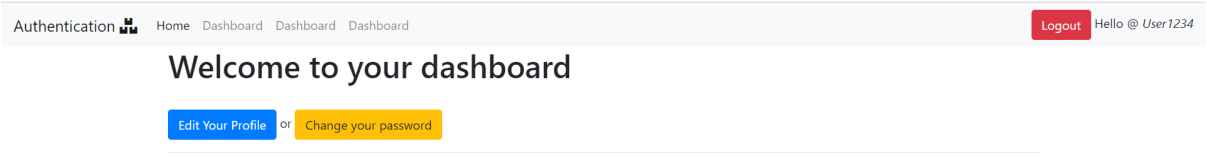
In order to compile and execute the web application alongside any underlying code, one must first undergo a series of steps described below.

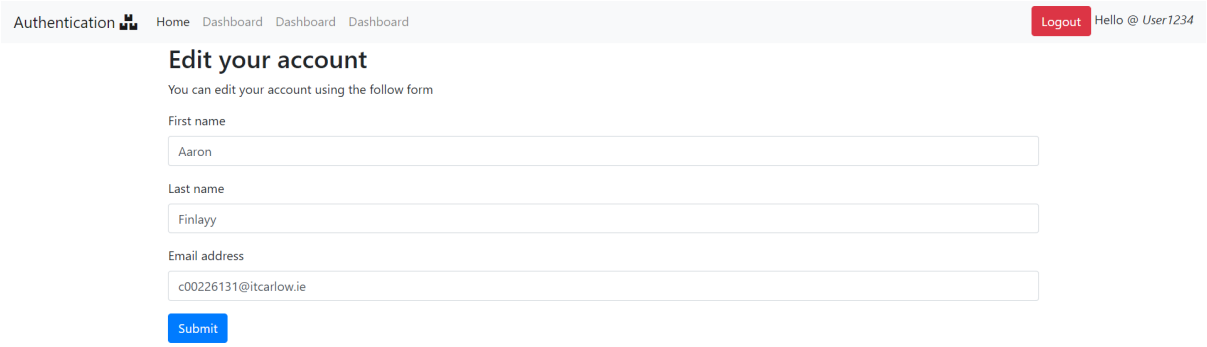
1. Create a virtual conda environment -> create conda --name name python=version
    a. In this case it was conda create --name UserAuthWebApp python=3.8
2. The conda create command builds the virtual environment. The --name keyword flag allocated the new virtual environment the name assigned, which in this instance is UserAuthWebApp.
3. To activate this environment proceed to type the following: conda activate name of conda environment, e,g, conda activate UserAuthWebApp.
4. Install package manager and use it to install dependencies, e.g. conda or pip.
5. Import necessary libraries and technologies, including but not limited to:
    a. Pandas
    b. Sklearn
    c. IDE (choose one)
        i. Spyder
        ii. VS Code
        iii. PyCharm
    d. Django
    e. Flask
    f. Opencv
    g. Matplotlib
    h. Scipy
    i. Keras
    j. Sklearn
    k. Tqdm
    l. Modules found in source code, but cannot be found in program, may typically be found and installed through the same practice but changing the name of the module in question.
6. It is not necessary to install python since it is pre-installed within the anaconda package. In order to run the GPU via tensorflow, one first had to go to the CUDA website, choose the correct installation configuration for the machine and then install. Upon completion of this, the GPU powered by CUDA should fire up upon every execution within the machine learning source code.

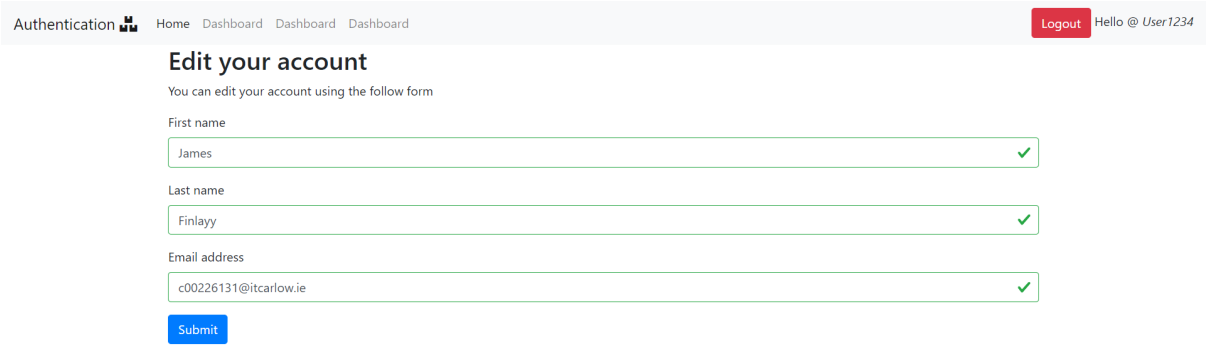
7. To start the django web application, in your new conda environment, create a new directory via mkdir succeeded by name of directory.
8. Switch into this directory - cd name of directory.
9. In the anaconda command prompt, key the following: django-admin startproject followed by name of project. This will set up the layout of your django project.
10. Traverse to this directory - key the following: cd src.
11. Upon entering the new project directory, enter into the conda prompt - django-admin startapp led by the name of the app.
12. django-admin runserver or python manage.py runserver (within manage.py dir) will start the django WSGI. python manage.py followed by migrate and then a separate

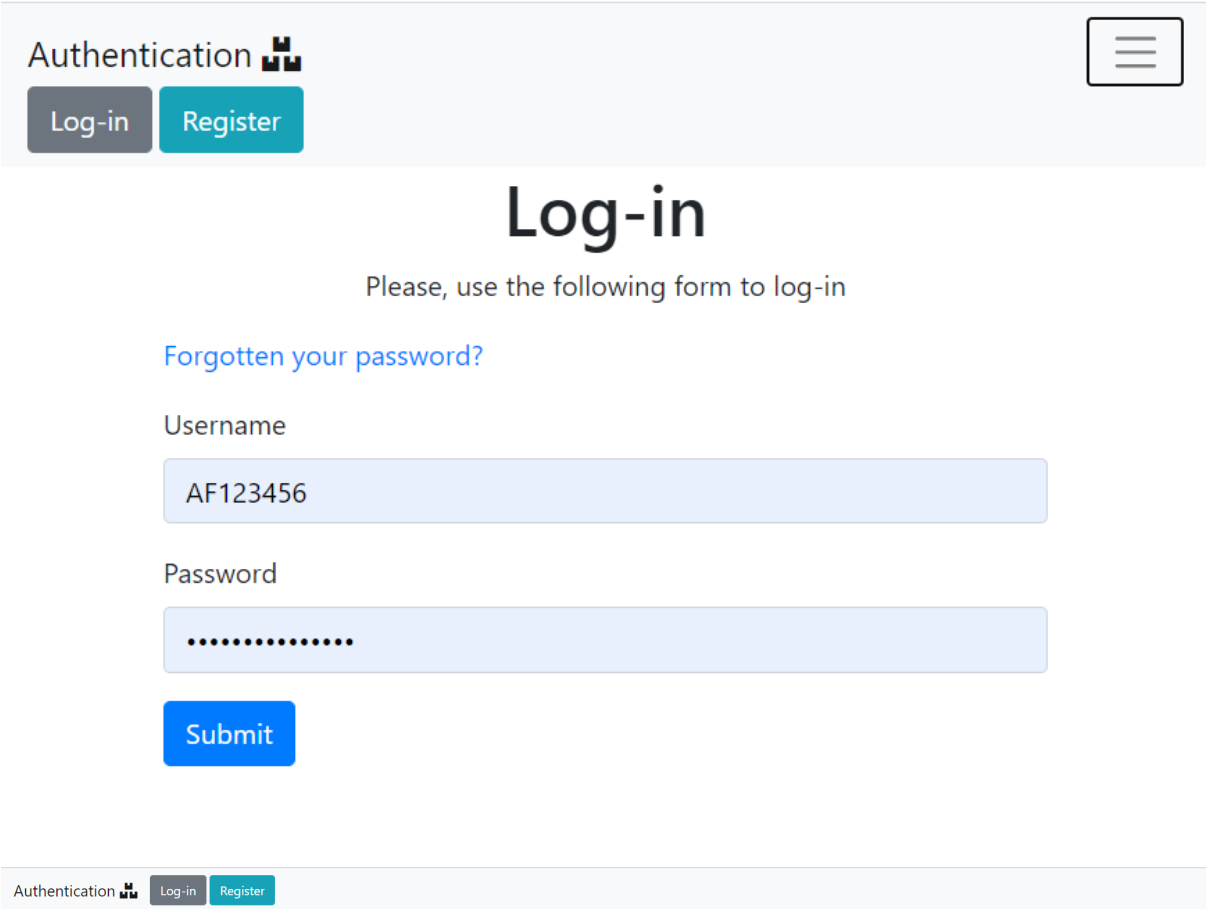
command with the same layout except instead of migrate, use makemigrations. This is to confirm you have migrated all data on your applications side.

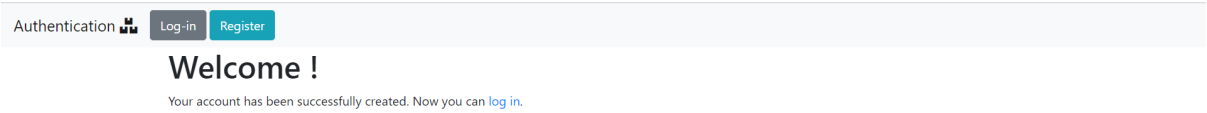
HAR - Front-End Screens

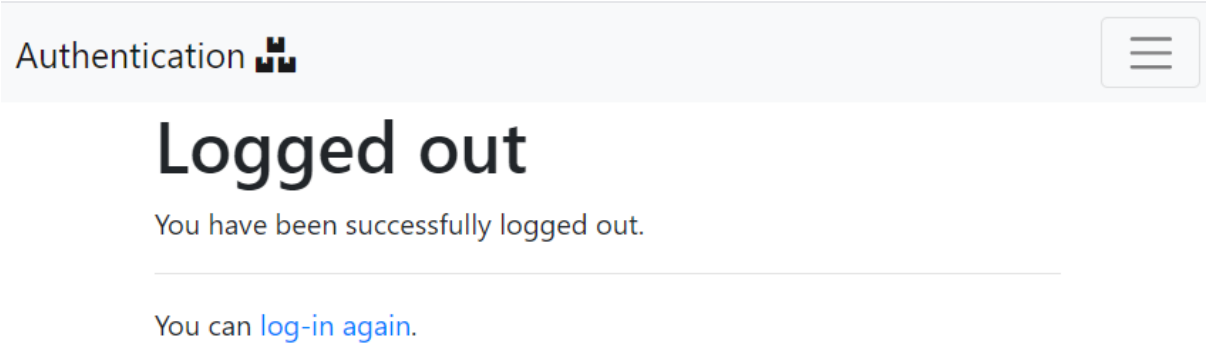







Authentication
Log-in
Register
Log-in
Please, use the following form to log-in
Forgotten your password?
Username
AF123456
Password
Submit


Authentication
Log-in
Register
Welcome !
Your account has been successfully created. Now you can log in.



Logged out

You have been successfully logged out.

You can log-in again.

HAR - Code & Screens Functionality

This section will contain all developed code throughout the project, including implementations that are and are not working. Furthermore, throughout the course of the project, much of the code was written while some was adapted from a multitude of sources. There are three categories to which the code in this document relate to, and that is: A) Written by Developer, B) Adapted from a source(s), C) Used exactly from a source. A and B are what is mostly categorised in the coding of the project. Any code referenced will be labelled within the program and therefore code itself.

It is important to note, the haar cascade classifier .xml configuration file is very large and has been adapted by a tutorial on how to create xml files and then an implementation of an xml file published in a research report, all of which is acknowledged within the fallClassifier.xml file.

HAR - Database Structure

Utilising django built in functionalities, it was decided that it would be efficient to utilize sqlite, as in the settings.py file, there is a configuration for this under the DATABASE setting. SQLite is a fully featured, small, fast, highly reliable SQL database engine. It was decided for ease of use and simplicity, this would be incorporated into the project.

Project Source Code - Front-End

Registerdone.html

```
{% extends "base.html" %}
{% block title %}Home{% endblock title %}
{% block content %}

{% endblock content %}
```

Navbar.html

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand" href="">Authentication <i class="fas fa-boxes"></i></a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent"
        aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>

    <div class="collapse navbar-collapse" id="navbarSupportedContent">

        {% if request.user.is_authenticated %}
        <ul class="navbar-nav mr-auto">
            <li class="nav-item active">
                <a class="nav-link" href="{% url 'authwebapp:dashboard' %}">Home <span class="sr-only">(current)</span></a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#">Dashboard</a>
            </li>        Find related code in django_project
            <li class="nav-item">
                <a class="nav-link" href="#">Dashboard</a>
            </li>
            <li class="nav-item">
                <a class="nav-link " href="#">Dashboard</a>
            </li>
        </ul>
        {% endif %}
        <span>
            {% if request.user.is_authenticated %}
            <a href="{% url 'authwebapp:logout' %}" class="btn btn-danger">Logout</a>
            Hello
            @ <i>{{user.username}}</i>
            {% else %}
            <div class="login__left">
                <a href="{% url 'authwebapp:login' %}" class="btn btn-secondary">Log-in</a>
                <a href="{% url 'authwebapp:register' %}" class="btn btn-info ">Register</a>
            </div>

            {% endif %}

        </span>
    </div>
</nav>
```

Edit.html

```
{% extends "base.html" %}
{% block title %}Dashboard{% endblock title %}
{% block content %}
<h1>{{welcome}}</h1>
<br>
<a href="{% url 'authwebapp:edit' %}" class="btn btn-primary">Edit Your Profile</a> or
<a href="{% url 'authwebapp:password_change' %}" class="btn btn-warning">Change your password</a>
<hr>
{% endblock content %}        Find related code in django_project
```

Logout.html

```
{% extends "base.html" %}
{% block title %}Logged out{% endblock %}
{% block content %}
<h1>Logged out</h1>
<p>
    You have been successfully logged out.
    <hr>
    You can <a href="{% url 'authwebapp:login' %}">log-in again</a>.
</p>
{% endblock %}        Find related code in django_project
```

password_change_form.html

```
{% extends "base.html" %}
{% load bootstrap4 %}
{% block content %}
<h2>Change Your Password</h2>
<p>Use the form below to change your Password</p>
<form action="" method="post" class="form">
    {% csrf_token %}
    {% bootstrap_form form %}
    {% buttons %}
    <button type="submit" class="btn btn-primary">Submit</button>
    {% endbuttons %}
</form>
{% endblock content %}
```

password_reset_complete.html

```
{% extends "base.html" %}
{% block title %}Reset your password{% endblock %}
{% block content %}
<h1>Reset your password</h1>
<p>We've emailed you instructions for setting your password.</p>
<p>If you don't receive an email, please make sure you've entered
    the address you registered with.</p>
{% endblock %}
```

password_reset_confirm.html

```
Someone asked for password reset for email {{email}}.Follow the Link below
{{ protocol }}://{{ domain }}{% url "authwebapp:password_reset_confirm"
uidb64=uid token=token %}
```

#

password_reset_form.html

```
{% extends 'base.html' %}
{% load bootstrap4 %}
{% block title %}Reset your password

{% endblock title %}
{% block content %}
<div class="container">
    <h2>Forgotten your Password</h2>
    <p>Enter your email address to obtain a new password</p>
    <form action="" method="post" class="form ">
        {% csrf_token %}
        {% bootstrap_form form %}
        {% buttons %}
        <button type="submit" class="btn btn-primary" value="send e-mail">Submit</button>
        {% endbuttons %}
    </form>
</div>
{% endblock content %}
```

register.html

```
{% extends "base.html" %}
{% load bootstrap4 %}
{% block title %}Registration

{% endblock title %}
{% block content %}

{% if form.errors %}

{% endif %}        Find related code in django_project

<form action="" method="post" class="form container">
    {% csrf_token %}
    {% bootstrap_form form %}
    {% buttons %}
    <button type="submit" class="btn btn-primary">Submit</button>
    {% endbuttons %}
</form>
{% endblock content %}
```

password_register_done.html

```
{% extends "base.html" %}
{% block content %}
<h1>Welcome {{ new_user.first_name }}!</h1>
<p>Your account has been successfully created. Now you can <a href="{% url 'authwebapp:login' %}">log in</a>.</p>
{% endblock content %}        Find related code in django_project
```

login.html

```
{% extends "base.html" %}
{% load bootstrap4 %}
{% block title %}Login{% endblock title %}

{% block content %}
<div class="">
    <div style="text-align: center;">
        <h1>Log-in</h1>
        {% if form.errors %}
        <p>
            Your username and password didn't match.Please
            Please try again.
        </p>
        {% else %}
        <p>Please, use the following form to log-in</p>
        {% endif %}
    </div>

    <form action="" method="post" class="form ">
        <p><a href="{% url 'authwebapp:password_reset' %}">Forgotten your password?</a></p>
        {% csrf_token %}
        {% bootstrap_form form %}
        {% buttons %}
        <input type="hidden" name="next" value="{{ next }}" />
        <button type="submit" class="btn btn-primary">Submit</button>
        {% endbuttons %}
    </form>
</div>
{% endblock content %}
```

Base.html

```
{% extends "base.html" %}
{% block title %}Home{% endblock title %}
{% block content %}

{% endblock content %}
```

home.tml

```
{% extends "base.html" %}
{% block title %}Home{% endblock title %}
{% block content %}

{% endblock content %}
```

Project Source Code - Back-End

Authwebapp__init__.py

```
# loads the appconfig subclass
default_app_config = 'authwebapp.apps.AuthWebAppConfig'
```

Authwebapp_admin.py

```
from django.contrib import admin

# Register your models here.
```

Authwebapp_apps.py

```
from django.apps import AppConfig


class AuthWebAppConfig(AppConfig):
    name = 'authwebapp'

    def ready(self):
        from . import signals
```

Authwebapp_models.py

```
from django.db import models
from django.conf import settings
# Create your models here.
##Setting up the User Profile, using django signals
##he following class contains a OneToOneField() with data relating to a specific user
class UserRegistrationModel(models.Model):
        user = models.OneToOneField(
            settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
```

Authwebapp_signals.py

```
from django.db.models.signals import post_save
from django.contrib.auth.models import User
from django.dispatch import receiver
from .models import UserRegistrationModel


@receiver(post_save, sender=User)
def creater_profile(sender, instance, created, **kwargs):
    if created:
        profile = UserRegistrationModel.objects.create(user=instance)
        profile.save()
        Find related code in django_project
```

Authwebapp_urls.py

```
from django.urls import path
from authwebapp.views import edit, dashboard, register #, VideoCamera, video_feed, route
from .model import cam
from django.urls import reverse_lazy
from django.contrib.auth.views import (LoginView, LogoutView, PasswordResetDoneView, PasswordResetView,        Find relat
                                       PasswordResetCompleteView, PasswordResetConfirmView,
                                       PasswordChangeView, PasswordChangeDoneView,
                                       PasswordResetDoneView)

from authwebapp import views

app_name = 'authwebapp'

urlpatterns = [
    path('register/', register, name='register'),
    path('edit/', edit, name='edit'),
    path('dashboard/', dashboard, name='dashboard'),
    path('', LoginView.as_view(template_name='registration/login.html'), name='login'),
    path('logout/', LogoutView.as_view(template_name='authwebapp/logged_out.html'), name='logout'),
    path('password_change/', PasswordChangeView.as_view(
        template_name='authwebapp/password_change_form.html'), name='password_change'),
    path('password_change/dond/', PasswordChangeDoneView.as_view(template_name='authwebapp/password_change_done.html'),
         name='password_change_done'),
    path('password_reset/', PasswordResetView.as_view(
        template_name='authwebapp/password_reset_form.html',
        email_template_name='authwebapp/password_reset_email.html',
        success_url=reverse_lazy('authwebapp:password_reset_done')), name='password_reset'),
    path('password_reset/done/', PasswordResetDoneView.as_view(
        template_name='authwebapp/password_reset_done.html'), name='password_reset_done'),
    path('reset/<uidb64>/<token>/', PasswordResetConfirmView.as_view(
        template_name='authwebapp/password_reset_confirm.html',
        success_url=reverse_lazy('authwebapp:login')), name='password_reset_confirm'),
    path('reset/done/', PasswordResetCompleteView.as_view(
        template_name='authwebapp/password_reset_complete.html'), name='password_reset_complete'),

]

 # path('index', views.VideoCamera, template_name='authwebapp/index.html',
    #        success_url=reverse_lazy('authwebapp:index'),  name='VideoCamera'),
    #path('index', views.video_feed(), template_name='authwebapp/index.html',
     #       success_url=reverse_lazy('authwebapp:index'),  name='video_feed'),
    #path('index', views.index, name='index'),
```

Authwebapp_views.py

```
from authwebapp import webStream
from django.core.files import File
#from authwebapp.webStream import generate
from django.http import request
from django.shortcuts import render
from django.contrib.auth.decorators import login_required        Find related code in d
from flask import app
from flask.templating import render_template
from .forms import UserRegistration, UserEditForm
from django.template import RequestContext
import cv2
import time


# Create your views here.

@login_required
def dashboard(request):
    context = {
        "welcome": "Welcome to your dashboard"
    }
    return render(request, 'authwebapp/dashboard.html', context=context)


def register(request):
    if request.method == 'POST':
        form = UserRegistration(request.POST or None)
        if form.is_valid():
            new_user = form.save(commit=False)
            new_user.set_password(
                form.cleaned_data.get('password')
            )
            new_user.save()
            return render(request, 'authwebapp/register_done.html')
    else:
        form = UserRegistration()

    context = {
        "form": form
    }

    return render(request, 'authwebapp/register.html', context=context)


@login_required
def edit(request):
    if request.method == 'POST':
        user_form = UserEditForm(instance=request.user,
                                 data=request.POST)
        if user_form.is_valid():
            user_form.save()
    else:
        user_form = UserEditForm(instance=request.user)
    context = {
        'form': user_form,
    }
    return render(request, 'authwebapp/edit.html', context=context)
```

Authwebapp_views.py

```
from django.views.decorators import gzip
from django.http import StreamingHttpResponse
import cv2
import threading

class Videoself:
    cam = request.File['webStream']
    video = cv2.VideoCapture(1)

        Find related code in django_project
    while(video.isOpened()):
        ret, frame = cam.read()

        if not ret:
            break

        do_something()

        k = cv2.waitKey(1)
        if k == 27:
            break

    def __init__(self):
        self.video = cv2.VideoCapture(0)
        (self.grabbed, self.frame) = self.video.read()
        threading.Thread(target=self.update, args=()).start()

    def __del__(self):
        self.video.release()

    def get_frame(self):
        image = self.frame
        _, jpeg = cv2.imencode('.jpg', image)
        return jpeg.tobytes()

    def update(self):
        while True:
            (self.grabbed, self.frame) = self.video.read()


def gen(cam):
    while True:
        frame = cam.get_frame()
        yield(b'--frame\r\n'
              b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')
```

(OpenCV Script) Camera.py

```python
# -*- coding: utf-8 -*-
"""
Created on Tue Apr 20 22:41:20 2021

@author: aaron
"""'''
from authwebapp.model.singlemotiondetector import SingleMotionDetector
import imutils
from imutils.video import VideoStream
from flask import Response
from flask import Flask
from flask import render_template
import thcondreading
import argparse
import datetime
import time
import cv2
import authwebapp
from importlib import reload
import cv2
import time
from flask import Flask, render_template, Response
'''
#Initialize Flask app
#sapp = Flask(__name__)
import cv2
import time
import numpy as np
import django
fallClassifier = cv2.CascadeClassifier('fallClassifier.xml')
fitToEllipse = False
cap = cv2.VideoCapture(0)
time.sleep(3)
countFALL = 0
k = 0
foregroundBackground = cv2.createBackgroundSubtractorMOG2()
while(1):
    ret, frame = cap.read()

    try:     #Converts every frame to the colour of gray scale and then the background is subtracted

        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        foregroundMask = foregroundBackground.apply(gray)

        #Find contours
        contours, _ = cv2.findContours(foregroundMask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

        if contours:

            # list which stores all areas
            areas = []

            for contour in contours:

                ar = cv2.contourArea(contour)
                areas.append(ar)
```

Login Configure Live Share

```python
            # list which stores all areas
            areas = []

            for contour in contours:

                ar = cv2.contourArea(contour)
                areas.append(ar)

            maximum_Area = max(areas, default = 0)

            maximum_Area_index = areas.index(maximum_Area)

            cntr = contours[maximum_Area_index]

            M = cv2.moments(cntr)

            x, y, w, h = cv2.boundingRect(cntr)

            cv2.drawContours(foregroundMask, [cntr], 0, (255,255,255), 3, maxLevel = 0)

            if h < w:

                k += 1

            if k > 10:

                print("FALL" + countFALL + 1)

                cv2.putText(foregroundMask, 'FALL', (x, y), cv2.FONT_HERSHEY_TRIPLEX, 0.5, (255,255,255), 2)
                cv2.rectangle(frame,(x,y),(x+w,y+h),(0,0,255),2)

            if w < h:

                k = 0
                cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)

            cv2.imshow('video', frame)

            if cv2.waitKey(33) == 27:
             break
    except Exception as e:
        break
'''
'''
'''
def gen_frames():

    while True:

        success, frame = cap.read()  # read the cap frame

        if not success:
            break
        else:
            ret, buffer = cv2.imencode('.kpg', frame)
            frame = buffer.tobytes()
```

```
'''
def gen_frames():

    while True:

        success, frame = cap.read()  # read the cap frame

        if not success:
            break
        else:
            ret, buffer = cv2.imencode('.kpg', frame)
            frame = buffer.tobytes()
            yield (b'--frame\r\n'
                   b'Content-Type: image/kpeg\r\n\r\n' + frame + b'\r\n')  # concat frame one by one and show result

cv2.destroyAllWindows()
'''
##Def: app route for default page of webapp
#route refers to url pattern of an app
'''
@app.route('/')
def index():
    return render_template('index.html')
#Def app route for video feed
@app.route('/video_feed')
def video_feed():
    return Response(gen_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')
    return countFALL

#The '/video_feed' route returns the streaming response.
#  Because this stream returns the images that are to be displayed
# in the web page, the URL to this route is in the "src"
#  attribute of the image tag
if __name__ == "__main__":
    app.run(debug=True)

#The '/video_feed' route returns the streaming response.
#  Because this stream returns the images that are to be displayed
# in the web page, the URL to this route is in the "src"
#  attribute of the image tag

if __name__ == "__main__":
    app.run(debug=True)

    app.run(debug=True)
'''
```

Manage.py

```
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'src.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)        Find related code in django_proje


if __name__ == '__main__':
    main()
```

src_settings.py

```
"""
Django settings for src project.

Generated by 'django-admin startproject' using Django 3.1.2.

For more information on this file, see
https://docs.djangoproject.com/en/3.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.1/ref/settings/
"""
        Find related code in django_project
from pathlib import Path
import os

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent
TEMPLATE_DIR = os.path.join('templates', BASE_DIR)
STATIC_DIR = os.path.join('static', BASE_DIR)

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'pxz!5e7vki$4%fx2+xb-z6v_85b)0j)nwontigspp#!+fpxu*2'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    'authwebapp',
    'bootstrap4'
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'src.urls'
```

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [TEMPLATE_DIR, ],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'src.wsgi.application'


# Database
# https://docs.djangoproject.com/en/3.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': str(BASE_DIR / 'db.sqlite3'),
    }
}


# Password validation
# https://docs.djangoproject.com/en/3.1/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/3.1/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'
        Find related code in django_project
USE_I18N = True
```

```
# Internationalization
# https://docs.djangoproject.com/en/3.1/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.1/howto/static-files/

STATIC_URL = '/static/'
STATICFILES_DIRS = [STATIC_DIR, ]


LOGIN_REDIRECT_URL = 'authwebapp:dashboard'
LOGIN_URL = 'login'
LOGOUT_URL = 'logout'


EMAIL_BACKEND = "django.core.mail.backends.filebased.EmailBackend"
EMAIL_FILE_PATH = str(BASE_DIR.joinpath('sent_emails'))
```

Src_wsgi.py

```
"""
WSGI config for src project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/3.1/howto/deployment/wsgi/
"""

import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'src.settings')

application = get_wsgi_application()
```

Src_asgi.py

```
"""
ASGI config for src project.

It exposes the ASGI callable as a module-level variable named ``application``

For more information on this file, see
https://docs.djangoproject.com/en/3.1/howto/deployment/asgi/
"""

import os

from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'src.settings')

application = get_asgi_application()
```

Src_urls.py

```
"""src URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/3.1/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('', include('authwebapp.urls', namespace='authwebapp')),
    path('admin/', admin.site.urls),
]
```

(CNN iterOne)model.py

```
#Model.py is the class responsible for the architecture of the CNN model at use
#Plan for structuring the CNN
#CONV LAYER -> CONV LAYER -> MAX POOL LAYER -> DROPOUT LAYER
#-> CONV LAYER -> CONV LAYER -> MAX POOL LAYER -> DROPOUT LAYER
#-> FULLY CONNECTED LAYER -> DROPOUT LAYER -> FULLY
#CONNECTED LAYER -> SOFTMAX LAYER

##

##PROCESS.PY CODE INCLUDED
##  Aaron Finlay
## HAR
## 4th Year Project
## 01/01/21
## **** Exploring AND Processing Data **** ##
## Primary Objective - download the datatset & visualize the images
## Change the label to one-hot encoding
## Scale the image pixel values to take between 0 and 1
## CIFAR-10 Dataset used for base implementation
##Img to be recognized - 32*32 pixels
##Labels - 10 possible (airplane, automobile, bird, cat, deer
## frog, horse, ship, and truck)
## Dataset size - 60,000 images, 50k training, 10k testing

#Step 1 - get dataset
from tensorflow.keras.datasets import cifar10
(x_train, y_train), (x_test, y_test) = pd.f = open('C:\Users\aaron\authwebapp\src\authwebapp\fall_classifier\SA01\C:\Users\aaron\authwebapp\src\authwebapp\sSA01\D01_SA01_R01.txt', 'r')
print(f.readline())

##The data needed is now stored in their own respective arrays
## e.g. (x_train, y_train, x_test and y_test)
##Next Step Explor the data, figure out shape of input feature array
print ('x_train shape:', x_train.shape)
## result of print statement:
    ## 50k img, 32 pixels in h x w and 3 pixels in depth (e.g. RBG)
##Shape of Label array
print('y_train shape:', y_train.shape)
##output = one number (corresponding to label) for each out of
##the 50,000 images
##try see example of image and label to solidify things.
## Example 1
print(x_train[0])
## above will display matrices of 3x3 which aren't too useful however
##using matplotlib it should help better the view of the image
```

```
##using matplotlib it should help better the view of the image
import matplotlib.pyplot as plt
%matplotlib inline

img = plt.imshow(x_train[0])

print('The label is:', y_train[0])

img = plt.imshow(x_train[1])
print('The label is:', y_train[1])

#Exploration of data complete now processing to take part
#will attempt to divide the class labels appropiately rather than no.
#goal is to get probability of 10 different classes
# therefore we will need 10 output neurons in our network
#since we have 10 output neurons, labels must match
#WE DO THIS BY:::
    #convert label into set of 10 nums where each num
    #represents if the img belongs to that class or not
    #thus, if an image belongs to first class, the first num
    #of this set will be a 1 and all other numbers in the set 0
#This is known as one-hot encoding

import tensorflow.keras
y_train_one_hot = tensorflow.keras.utils.to_categorical(y_train,10)
y_test_one_hot = tensorflow.keras.utils.to_categorical(y_test, 10)
## first line means the initial array is taken with just number,
## y_train, and convert it to the one_hot_encodings, y_train_one_hot.
##The number 10 is required as a parameter as you need to tell the
##function how many classes there are (10).

##If we want to see how the label for our second image (the truck, label:9)
## looks like in the one-hotsetting:
print('The one hot label is:', y_train_one_hot[1])

#Now that the labels are processed (y), we want to process our image (x).
#Common solution is to let the values be between 0-1, which will aid
#in the training of the neural network. As the pixel values already take
#the values between 0 and 255, we simply need to divide by 255.
x_train = x_train.astype('float32')
x_test = x_test.astype('Float32')
x_train = x_train / 255
x_test = x_test / 255
## ^^ converts the type to float32, which is in turn a datatype
## that can store values with decimal points. Then each cell is divided by 255.
## can view the array values of the first train image by:
x_train[0]

##End of program
## Summary of objectives:
    ##Download dataset and visualize images
    ##Changed label to one-hot encodings
    ##Scale the image pixel values to take between 0-1

##SECOND PROGRAM FOR MODEL

from tensorflow.keras.models import Sequential
```

```
model = Sequential()
##above calls an empty sequential model

##one layer at a time will be added, the f1st is a conv layer
#with filter size 3x3, stride size 1 (in both dimensions),
#and depth 32. The padding is the same and the activation is 'relu'
#(these two settings shall apply to all layers in the CNN)
model.add(Conv2D(32, (3, 3), activation=
'relu', padding='same', input_shape=(32,32,3)))

##What the above code does is add this layer to the empty sequential model,
##first no. 32 refers to depth, Next is the specification of activation in
#terms of which is 'relu' and padding which is 'same'.
# Stride remains 1 as a default setting, hence the non-specification of it,
#unless this setting is desired to be changed, there is no need to specify it.

#we must specify an inpuy size for the first (input) layer
#subsequent layers need not be specifie as they can infer the input
#size from the output size of the previous layer

model.add(Conv2D(32, (3, 3), activation='relu',
padding='same'))

#Next layer is the max pooling layer with pool size 2x2 & stride 2 (in both Dimensions).
#The default for max pooling layer stride is pool size, therefore, we do not
#need to specify the stride

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

#Finally, a dropout layer is needed with probability of 0.25 of
#dropout to prevent overfitting the model

model.add(Dropout(0.5))

#Next 4 layers, similar to previous except depth of conv layer = 64 rather than 32
model.add(Conv2D(64, (3, 3), activation='relu',
padding='same'))

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

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

#next the code for fully connected layer must be completed,
#the neurons are spatially arranged in a cube like structure,
#rather than a single row. To transform this structure into one row,
#it must first be flattened, therefore a Flatten layer shall be implemented
model.add(Flatten())
#a dense (FC) layer of 512 nerons w/ relu activation

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 neurons and softmax activation:
model.add(Dense(10, activation='softmax'))
```

```
#next the code for fully connected layer must be completed,
#the neurons are spatially arranged in a cube like structure,
#rather than a single row. To transform this structure into one row,
#it must first be flattened, therefore a Flatten layer shall be implemented
model.add(Flatten())
#a dense (FC) layer of 512 nerons w/ relu activation

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 neurons and softmax activation:
model.add(Dense(10, activation='softmax'))

##summary of architecture
model.summary()

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

hist = model.fit(x_train, y_train_one_hot,
                 batch_size=32, epochs=20,
                 validation_split=0.2)
#model is trained with batch size 32, 20 epochs.
#Using the setting validation_split=0.2 enables a quick and easy
#partition of the dataset, removing the need to manually split the train
#and validation sets at the beginning, 20% used to validate the model

#can visualize the model training & validation loss over
#the number of epochs using this code
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()

model.evaluate(x_test, y_test_one_hot)[1]
model.save('model.h5')
##above code saves trained model in HDF5 format

##to load the saved model
from tensorflow.keras.models import load_model
model = load_model('model.h5')

my_image = plt.imread("C:/Users/aaron/HumanActivityRecognition/cat.jpg")
```

(CNN iterTwo)fall.py

```
# -*- coding: utf-8 -*-

# Video Classification tutorial
# overview, steps involved in building the model, exploring the dataset
# training the model and evaluating the model
# traditional method of image classification is as follows:
# take images, use extract the features
"""
import os.path
os.path.exists('mydirectory/myfile.txt')
True
os.path.exists('does-not-exist.txt')
False
os.path.exists('mydirectory')
True

"""
import csv

import cv2  # for capturing video
import math  # for mathematical  operations
import os.path
import matplotlib
import matplotlib.pyplot as plt  # for plotting images
import pandas as pd
from keras.preprocessing import image  # for preprocessing img
import numpy as np  # math operations
from keras.utils import np_utils
from skimage.transform import resize  # for image resizing
from sklearn.model_selection import train_test_split
from glob import glob

from tensorflow.python.keras import Sequential
from tqdm import tqdm

import keras
from keras.models import Sequential
from keras.applications.vgg16 import VGG16
from keras.layers import Dense, InputLayer, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D, GlobalMaxPooling2D
from keras.preprocessing import image

from keras.applications.vgg16 import VGG16
import os
from scipy import stats as s

from keras.layers import Dense, Dropout, Flatten, BatchNormalization, Activation
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.constraints import maxnorm

# open the .txt file which have names of training videos
f = open("trainlist01.txt", "r")
temp = f.read()
videos = temp.split('\n')

# creating a dataframe having video names
train = pd.DataFrame()
```

```python
# creating a dataframe having video names
train = pd.DataFrame()
train['video_name'] = videos
train = train[:-1]
train.head()

# open the .txt file which have names of test videos
f = open("testlist01.txt", "r")
temp = f.read()
videos = temp.split('\n')

# creating a dataframe having video names
test = pd.DataFrame()
test['video_name'] = videos
test = test[:-1]
test.head()

train_video_tag = []
for i in range(train.shape[0]):
    train_video_tag.append(train['video_name'][i].split('/')[0])

train['tag'] = train_video_tag

# creating tags for test videos
test_video_tag = []
for i in range(test.shape[0]):
    test_video_tag.append(test['video_name'][i].split('/')[0])

test['tag'] = test_video_tag

# 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:
```

```
# 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()



# getting the names of all the images
images = glob("train_1/*.jpg")
# creating empty list
train_image = []
train_class = []

# for loop to read and store frames
for i in tqdm(range(train.shape[0])):
    # loading img and keeping target size as (224,244,3)
    img = image.load_img(train['image'][i], target_size=(224,224,3))
    # converting to array
    img = image.img_to_array(img)
    # Normalizing pixel value
    img = img / 255
    # appending the img to train_image lisy
    train_image.append(img)





# converting the list to numpy array
```

```
model = Sequential()
model.add(Dense(1024, activation='relu', input_shape=(25088,)))
model.add(Dropout(0.5))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(101, activation='softmax'))

# loading the trained weights
model.load_weights("bestWeight.hdf5")

# compiling the model
model.compile(loss='categorical_crossentropy',optimizer='Adam',metrics=['accuracy'])

# getting the test list
f = open("testlist01.txt", "r")
temp = f.read()
videos = temp.split('\n')

# creating the dataframe
test = pd.DataFrame()
test['video_name'] = videos
test = test[:-1]
test_videos = test['video_name']
test.head()


# creating the tags
train = pd.read_csv('UCF-101/train_new.csv')
y = train['class']
y = pd.get_dummies(y)
'''
# creating two lists to store predicted and actual tags
predict = []
actual = []
    prediction_images = []
    for i in range(len(images)):
        img = image.load_img(images[i], target_size=(224,224,3))
        img = image.img_to_array(img)
        img = img/255
        prediction_images.append(img)
    '''
        '''
    # converting all the frames for a test video into numpy array
    prediction_images = np.array(prediction_images)
    # extracting features using pre-trained model
    prediction_images = base_model.predict(prediction_images)
    # converting features in one dimensional array
    prediction_images = prediction_images.reshape(prediction_images.shape[0], 7*7*512)
    # predicting tags for each array
    prediction = model.predict_classes(prediction_images)
    # appending the mode of predictions in predict list to assign the tag to the video
    predict.append(y.columns.values[s.mode(prediction)[0][0]])
    # appending the actual tag of the video
```

Plagiarism Declaration

*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