THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Machine Learning
UML501

July 2022 – December 2022

Weapon Detection System

Project Report

Submitted by:

Ananya Malik	(102003124)
Rishab Jalota	(102003158)

3COE6

Submitted to:

Mr. Niyaz Ahmed Wani

Index

Problem Description

Security & surveillance is one of the fastest-growing industries in the world. It is rapidly becoming a necessity in both personal and professional settings. However, across the country and the world, there exists the problem of inefficiency due to overworked security personnel. Quite often a single person is looking into feeds from multiple cameras which is a hectic task and is prone to errors. Moreover, security feeds in a majority of settings are just to check recordings in case of a mishap. They don't have any preventative measures.

Overview

AI surveillance can help detect dangerous objects like guns or knives at entry points or inside the building and execute appropriate preventative actions, such as: setting off alarms, alerting security personnel, and/or automatically locking doors depending upon the situation.

Our program involves training a model that is able to detect knives, and applying it to a camera stream in order to print an alert when one is detected. The feed from the CCTV will be passed into the model frame by frame which will detect the presence of any weapon. If the weapon is detected at the gate, it will immediately print an alert to notify the relevant security personnel. This model can be used in CCTV camera streams to pre-emptively detect dangerous objects & hence prevent untoward incidents from happening.

Our model utilises the YOLO (You Only Look Once) algorithm and is trained on a labelled dataset of images to detect knives. The YOLO algorithm outperforms other object detection algorithms, such as R-CNN and its variants, in terms of speed.
The YOLO framework is unique in that it takes the entire image in a single instance and predicts the bounding box coordinates and class probabilities for these boxes. Whereas algorithms like Faster RCNN work by detecting possible regions of interest using the Region Proposal Network and then perform recognition on those regions separately. Methods that use Region Proposal Networks thus end up performing multiple iterations for the same image, while YOLO achieves the same result with a single iteration.

Dataset Description

The dataset used in this project consists of two separate folders, each containing images in which a knife may be present. The ***with_knife*** folder consists of 646 images extracted from various sources that depict knives held in human hands from various angles. On the other hand, the ***without_knife*** folder contains 1605 images of random objects. The images contained in both these folders have been collected from various sources on the Internet, such as Kaggle and Github, as well as ~250 images that have been manually taken by the project creators and added to the database. These labelled folders are used for training the model.

Python Libraries

1. `os`:
   Python OS module provides the facility to establish the interaction between the user and the operating system. It offers many useful OS functions that are used to perform OS-based tasks and get related information about operating system. The OS comes under Python's standard utility modules.

2. `keras`:
   It is an open-source high-level Neural Network library, which is written in Python is capable enough to run on Theano, TensorFlow, or CNTK. It was developed by one of the Google engineers, Francois Chollet. It is made user-friendly, extensible, and modular for facilitating faster experimentation with deep neural networks. It not only supports Convolutional Networks and Recurrent Networks individually but also their combination.

3. `sklearn`:
   Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

4. `numpy`:
   NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It is open-source software.

5. `matplotlib`:

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

6. `imutils`:

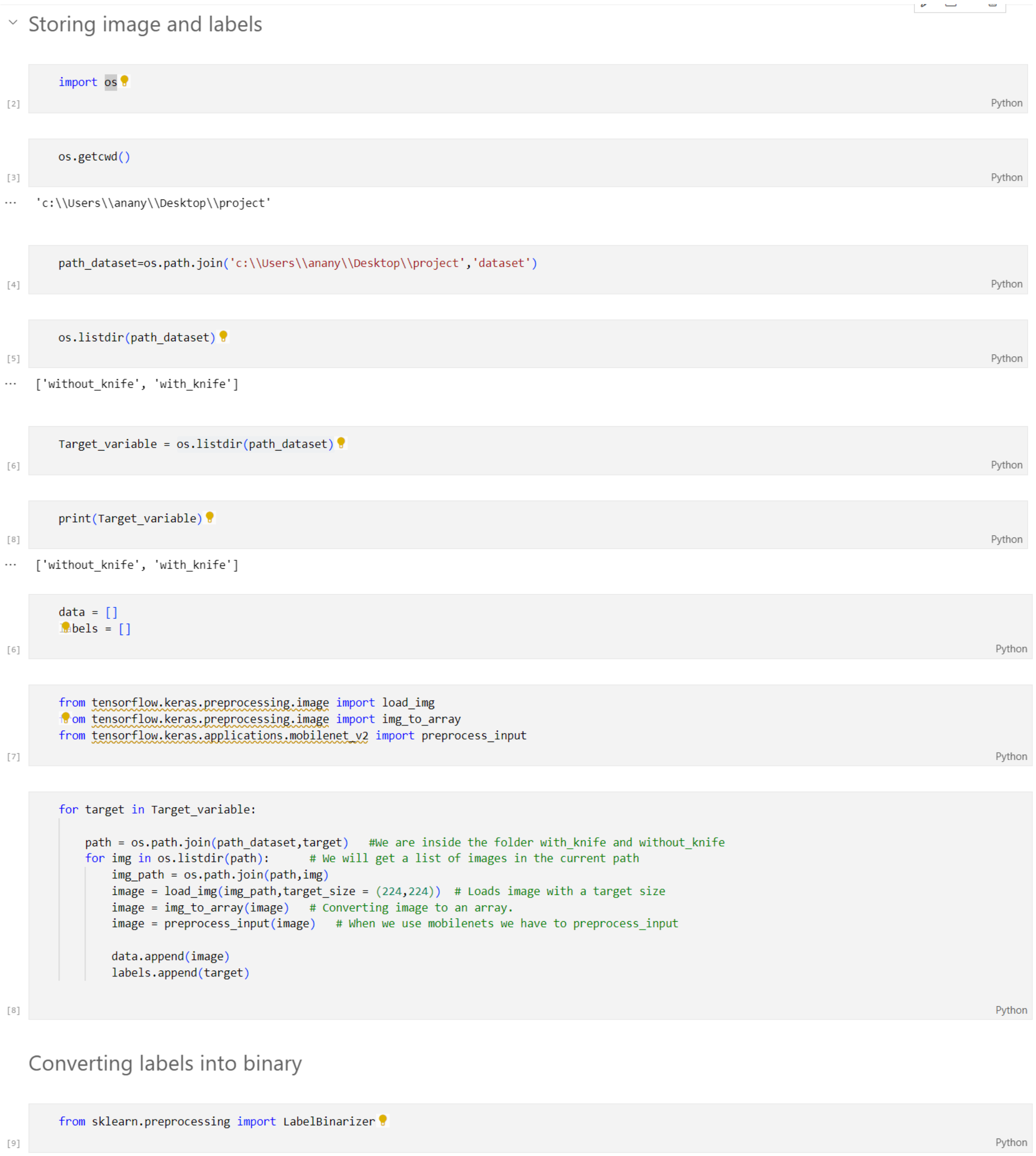
   A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, displaying Matplotlib images, sorting contours, detecting edges, and much more easier with OpenCV and both Python 2.7 and Python 3.

7. `cv2`:

   OpenCV-Python is a library of Python bindings designed to solve computer vision problems. cv2.imread() method loads an image from the specified file. If the image cannot be read (because of missing file, improper permissions, unsupported or invalid format) then this method returns an empty matrix.

Code

model-training.ipynb



Storing image and labels

[2]

```python
import os
```

[3]

```python
os.getcwd()
```

```
'c:\\Users\\anany\\Desktop\\project'
```

[4]

```python
path_dataset=os.path.join('c:\\Users\\anany\\Desktop\\project','dataset')
```

[5]

```python
os.listdir(path_dataset)
```

```
['without_knife', 'with_knife']
```

[6]

```python
Target_variable = os.listdir(path_dataset)
```

[8]

```python
print(Target_variable)
```

```
['without_knife', 'with_knife']
```

[6]

```python
data = []
labels = []
```

[7]

```python
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
```

[8]

```python
for target in Target_variable:

    path = os.path.join(path_dataset,target)   #We are inside the folder with_knife and without_knife
    for img in os.listdir(path):      # We will get a list of images in the current path
        img_path = os.path.join(path,img)
        image = load_img(img_path,target_size = (224,224))  # Loads image with a target size
        image = img_to_array(image)   # Converting image to an array.
        image = preprocess_input(image)   # When we use mobilenets we have to preprocess_input

        data.append(image)
        labels.append(target)
```

Converting labels into binary

[9]

```python
from sklearn.preprocessing import LabelBinarizer
```

```
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
```

[10] Python

```
from tensorflow.keras.utils import to_categorical
```

[11] Python

```
labels = to_categorical(labels)
```

[12] Python

Converting data and labels into array

```
import numpy as np
```

[13] Python

```
data = np.array(data,dtype="float32")
```

[14] Python

```
labels = np.array(labels)
```

[15] Python

Creating train test split

```
from sklearn.model_selection import train_test_split
```

[16] Python

```
X_train,X_test,y_train,y_test = train_test_split(data,labels,test_size=.20,stratify = labels,random_state=42)
```

[17] Python

Generating augmented data as dataset is small.

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

[18] Python

```
aug = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.15,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.15,
    horizontal_flip=True,
    fill_mode="nearest")
```

[19] Python

Training

```python
Learning_rate = 1e-3
```

[20] Python

```python
Epochs = 15
```

[21] Python

```python
BS = 25
```

[22] Python

```python
from tensorflow.keras.applications import MobileNetV2
```

[23] Python

```python
from tensorflow.keras.layers import Input
```

[24] Python

```python
base_model = MobileNetV2(weights = 'imagenet',include_top = False,input_tensor = Input(shape=(224,224,3)))
```

[25] Python

Top Layer(Pooling+Flatten+Dense+Dropout+Dense)

```python
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
```

[26] Python

```python
headModel = base_model.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)   #remove less relevant features
headModel = Flatten(name="flatten")(headModel)               #converts input to 1D->required form of dense input
headModel = Dense(128, activation="relu")(headModel)         #relu: for non-linear
headModel = Dropout(0.5)(headModel)                          #model learning from same set, hence 50% neurons dropped per pass to prevent overfitting
headModel = Dense(2, activation="softmax")(headModel)        #softmax for O/P layer: probability based activation function, between 0 & 1
```

[27] Python

```python
from tensorflow.keras.models import Model
```

[28] Python

```python
model = Model(inputs=base_model.input, outputs=headModel)
```

[29] Python

```python
for layer in base_model.layers:
    layer.trainable = False
```

[30] Python

Compiling

```python
from tensorflow.keras.optimizers import Adam
```

[31] Python

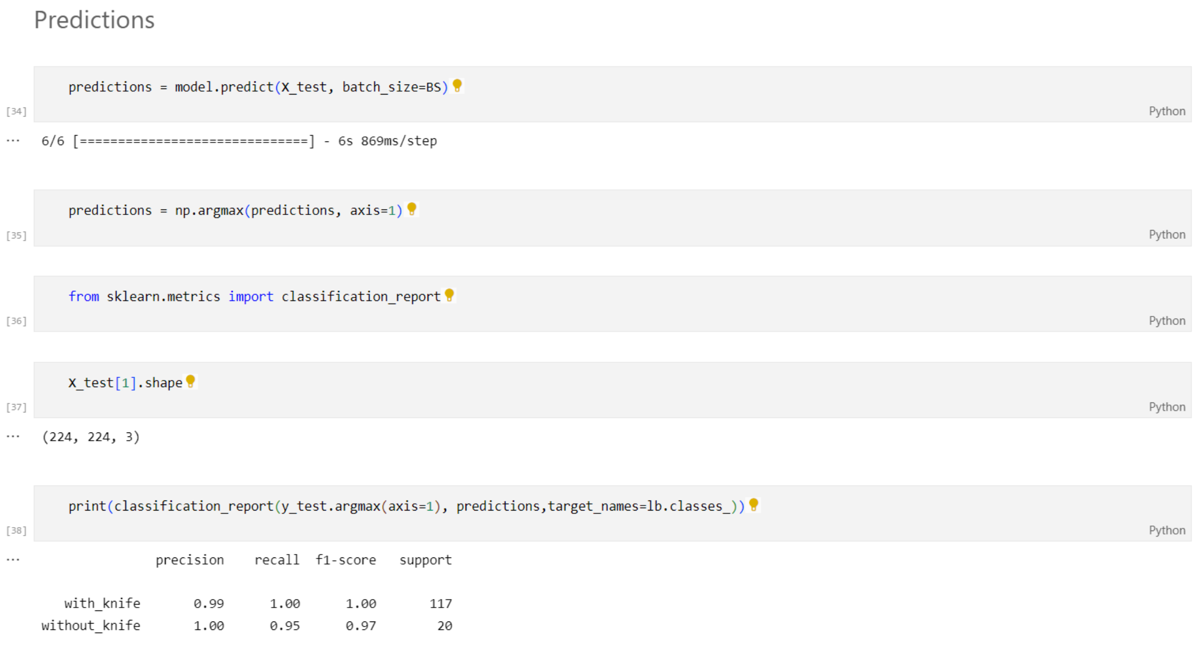
```python
optim = Adam(lr=Learning_rate, decay=Learning_rate/ Epochs)
model.compile(loss="binary_crossentropy", optimizer=optim,metrics=["accuracy"])
#optimiser: decides how weights will be updated in back propagation
```

[32] Python

```
c:\Python310\lib\site-packages\keras\optimizers\optimizer_v2\adam.py:114: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)
```

```python
H = model.fit(
    aug.flow(X_train, y_train, batch_size=BS),
    steps_per_epoch=len(X_train) // BS,
    validation_data=(X_test,y_test),
    validation_steps=len(X_test) // BS,
    epochs=Epochs)
```

[33] Python



Predictions

```
predictions = model.predict(X_test, batch_size=BS)
```

```
6/6 [==============================] - 6s 869ms/step
```

```
predictions = np.argmax(predictions, axis=1)
```

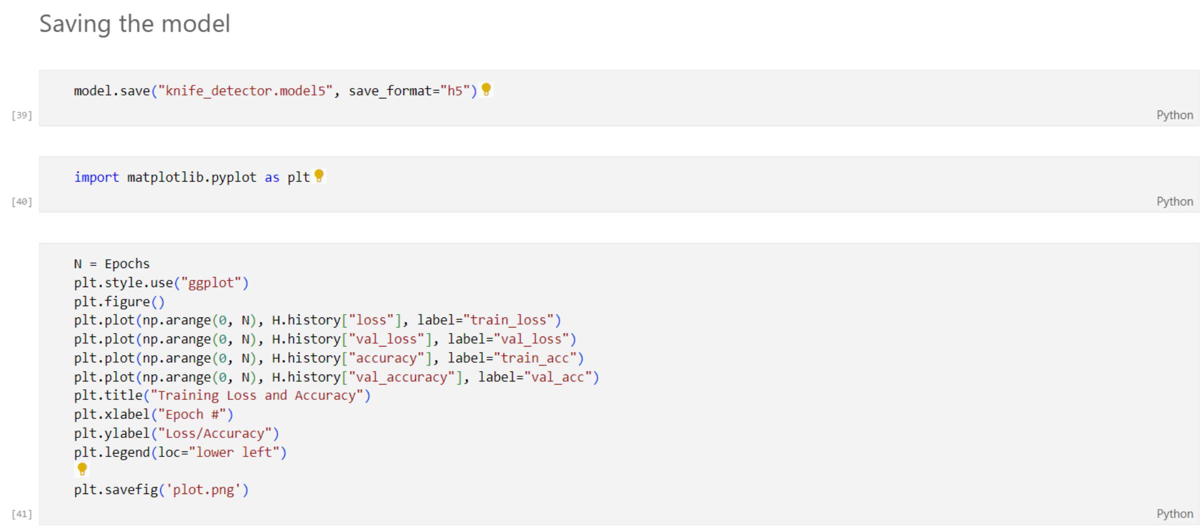
```
from sklearn.metrics import classification_report
```

```
X_test[1].shape
```

```
(224, 224, 3)
```

```
print(classification_report(y_test.argmax(axis=1), predictions,target_names=lb.classes_))
```

```
               precision    recall  f1-score   support

   with_knife       0.99      1.00      1.00       117
without_knife       1.00      0.95      0.97        20
```

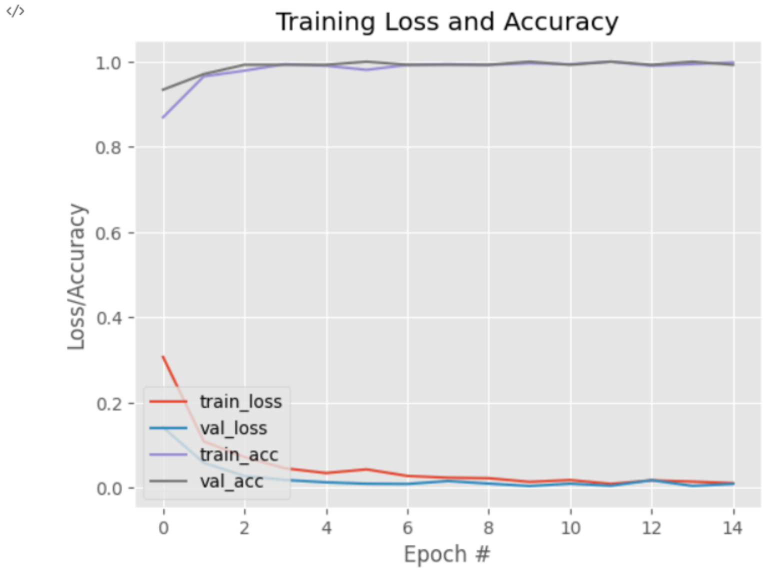


Saving the model

```
model.save("knife_detector.model5", save_format="h5")
```

```
import matplotlib.pyplot as plt
```

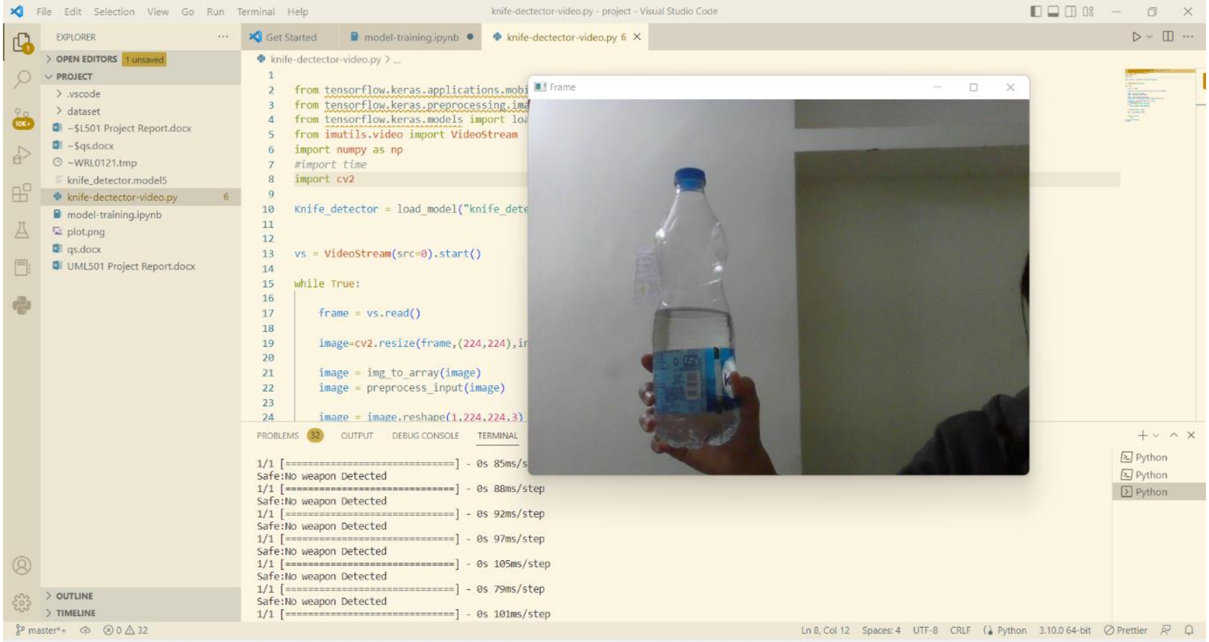
```
N = Epochs
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")

plt.savefig('plot.png')
```



knife-detector-video.py

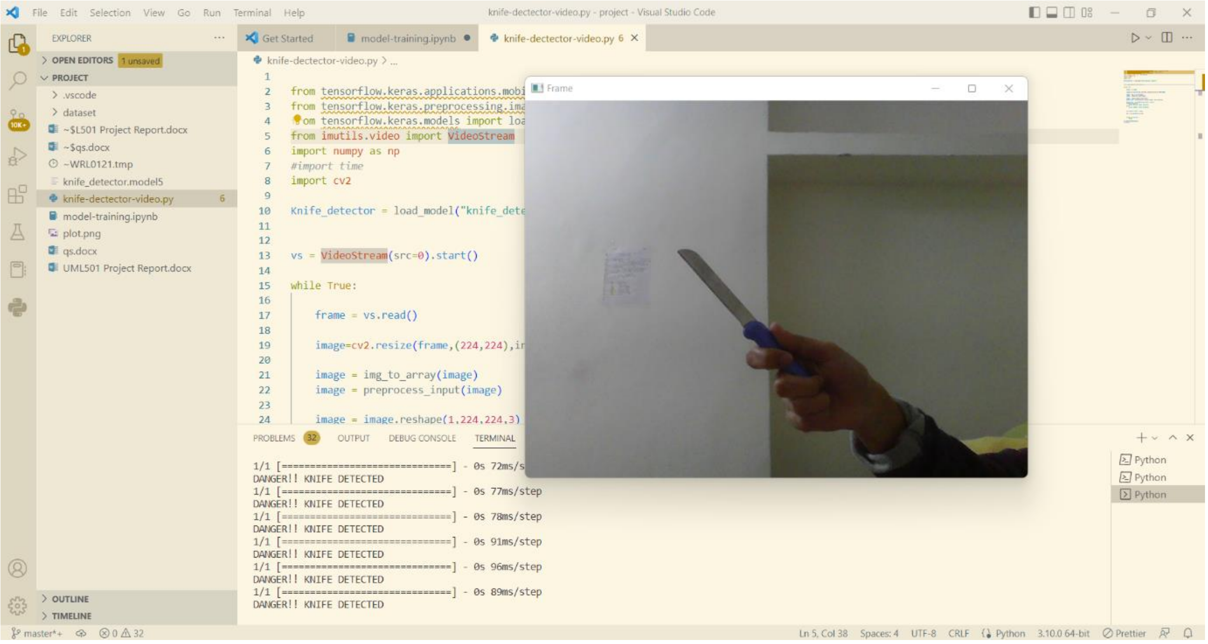
```
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import time
import cv2
import os

os.getcwd()

Knife_detector = load_model("knife_detector.model5")

#print(Knife_detector.summary())

vs = VideoStream(src=0).start()

while True:

    frame = vs.read()

    image=cv2.resize(frame,(224,224),interpolation=cv2.INTER_AREA)

    image = img_to_array(image)
    image = preprocess_input(image)

    image = image.reshape(1,224,224,3)
    predictions = Knife_detector.predict(image, batch_size=32)

    predictions = np.argmax(predictions, axis=1)
    if(predictions==1):
        print('Safe:No weapon Detected')
    else:
        print('DANGER!! KNIFE DETECTED')


    cv2.imshow("Frame", frame)

    key = cv2.waitKey(1) & 0xFF


    if key == ord("q"):
        break

cv2.destroyAllWindows()
vs.stop()
```

Output



```

from tensorflow.keras.applications.mobi
from tensorflow.keras.preprocessing.ima
from tensorflow.keras.models import loa
from imutils.video import VideoStream
import numpy as np
#import time
import cv2

Knife_detector = load_model("knife_dete


vs = VideoStream(src=0).start()

while True:

    frame = vs.read()

    image=cv2.resize(frame,(224,224),ir

    image = img_to_array(image)
    image = preprocess_input(image)

    image = image.reshape(1,224,224,3)
```

```
1/1 [==============================] - 0s 85ms/s
Safe:No weapon Detected
1/1 [==============================] - 0s 88ms/step
Safe:No weapon Detected
1/1 [==============================] - 0s 92ms/step
Safe:No weapon Detected
1/1 [==============================] - 0s 97ms/step
Safe:No weapon Detected
1/1 [==============================] - 0s 105ms/step
Safe:No weapon Detected
1/1 [==============================] - 0s 79ms/step
Safe:No weapon Detected
1/1 [==============================] - 0s 101ms/step
```



```python

from tensorflow.keras.applications.mobi
from tensorflow.keras.preprocessing.ima
from tensorflow.keras.models import loa
from imutils.video import VideoStream
import numpy as np
#import time
import cv2

Knife_detector = load_model("knife_dete


vs = VideoStream(src=0).start()

while True:

    frame = vs.read()

    image=cv2.resize(frame,(224,224),ir

    image = img_to_array(image)
    image = preprocess_input(image)

    image = image.reshape(1,224,224,3)
```

```
1/1 [==============================] - 0s 72ms/s
DANGER!! KNIFE DETECTED
1/1 [==============================] - 0s 77ms/step
DANGER!! KNIFE DETECTED
1/1 [==============================] - 0s 78ms/step
DANGER!! KNIFE DETECTED
1/1 [==============================] - 0s 91ms/step
DANGER!! KNIFE DETECTED
1/1 [==============================] - 0s 96ms/step
DANGER!! KNIFE DETECTED
1/1 [==============================] - 0s 89ms/step
DANGER!! KNIFE DETECTED
```