MOTION DETECTION
Internship
Progress Report

Saim Abbas
sabbas486249@gmail.com
03188334442

Ayesha Amjad
ayeshamjad016@gmail.com
03125050172

Milestones Work History			
Date	**Day**	**Milestone**	**Page No.**
5 July	Wednesday	Research on Yolo and CNN	3
10 July	Monday	Annotations of Car Images	4-8
18 July	Tuesday	Car Image Detection System using CNN	9-14
25 July	Tuesday	Car Video Detection System using Yolo	

Github Repository:
https://github.com/saimdev/Car_Theft_Detection_System
Ayesha Amjad is participating in this repo as a collaborator

Mostly used platforms:

- Google Colab
- Vs Code
- Python 3.11
- OS Windows 10

Research Papers:
https://docs.google.com/document/d/13jfCHFJJwPPegxhlRIX8L6jgJdFhhGx3KoMDCIzd8Ek/edit?usp=sharing
https://drive.google.com/file/d/18Q7Z4SuAABjvCGg3Q4PVzfZ5PIm8Oj3M/view?usp=sharing
https://drive.google.com/file/d/1DziVD6HW1NltYKvB3N3WGu2Q5Lgx_5L-/view?usp=sharing
https://drive.google.com/file/d/1RIkyB6vSRKdceEpVA4tsfedoXnvYoAQ1/view?usp=sharing

Milestone 1: Research on CNN & Yolo

CNN Research Paper Link:
https://drive.google.com/file/d/1uGB7d2rOWrXzEEm3PuY8jFZrgFBIYwLE/view?usp=sharing

YOLO Research Paper Link:
https://drive.google.com/file/d/1sb1dbN1DXWKXyPKinCtZXA3lWzQPjHeS/view?usp=sharing

CNN Notes we made:
https://drive.google.com/file/d/1r8KHTfN3VAVB66apYJFKeJUDZ8pCYNKM/view?usp=sharing

YOLO Notes we made:
https://drive.google.com/file/d/14lsx4xNusSo_K2O19Nye2WyhuX4DLIfp/view?usp=sharing

Milestone 2: Annotating Car Images for Dataset

Github Repo:
https://github.com/saimdev/Car_Theft_Detection_System/blob/master/Annotating_Pictures.py
Kaggle Dataset Link:
https://www.kaggle.com/datasets/prondeau/the-car-connection-picture-dataset

Discussion:
We downloaded the dataset of images of cars from kaggle website and then start doing annotation manually for training purpose on custom CNN model but later on we get that labeling is not necessary for simple CNN.

Libraries used:
- opencv-python
- os

Python Code:
```
import cv2
import os

# Set the path to your car images dataset directory
dataset_path = './dataset'

# Set the path to save the annotated dataset
annotated_dataset_path = './annotated'

# Create the annotated dataset directory if it doesn't exist
os.makedirs(annotated_dataset_path, exist_ok=True)

# Load the car images from the dataset directory
car_images = []
for root, dirs, files in os.walk(dataset_path):
    for file in files:
        # Check if the file is an image
        if file.endswith('.jpg') or file.endswith('.png'):
            # Load the image
            image_path = os.path.join(root, file)
            image = cv2.imread(image_path)
            car_images.append(image)

# Global variables for mouse events
annotations = []
```

```
current_annotation = {'x': -1, 'y': -1, 'width': -1, 'height': -1}
annotation_in_progress = False

# Mouse event callback function
def annotate_cars(event, x, y, flags, param):
    global current_annotation, annotation_in_progress,
annotated_image

    if event == cv2.EVENT_LBUTTONDOWN:
        # Start drawing a new bounding box
        current_annotation = {'x': x, 'y': y, 'width': -1, 'height':
-1}
        annotation_in_progress = True
    elif event == cv2.EVENT_LBUTTONUP:
        # Finish drawing the bounding box
        current_annotation['width'] = x - current_annotation['x']
        current_annotation['height'] = y - current_annotation['y']
        annotations.append(current_annotation.copy())
        annotation_in_progress = False

        # Draw the annotated bounding box on the image
        cv2.rectangle(annotated_image, (current_annotation['x'],
current_annotation['y']),
                      (current_annotation['x'] +
current_annotation['width'],
                       current_annotation['y'] +
current_annotation['height']),
                      (0, 255, 0), 2)

# Display the images and prompt for annotation
for i, image in enumerate(car_images):
    annotated_image = image.copy()

    # Create a named window and set the mouse callback function
    cv2.namedWindow('Annotate Cars')
    cv2.setMouseCallback('Annotate Cars', annotate_cars)

    while True:
        # Display the image
        cv2.imshow('Annotate Cars', annotated_image)

        # Prompt for annotation
        key = cv2.waitKey(1) & 0xFF
```

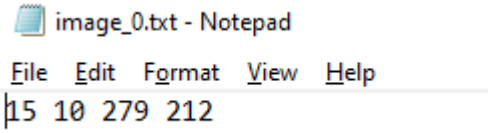
```
        # Annotation process
        if key == ord('q'):
            # Quit annotation
            break
        elif key == ord('c'):
            # Continue to the next image
            break
        elif key == ord('r'):
            # Reset the annotation for the current image
            annotated_image = image.copy()
            if annotations:
                annotations.pop()

    if key == ord('q'):
        # Quit annotation
        break

    # Save the annotated image with a unique name
    cv2.imwrite(os.path.join(annotated_dataset_path,
f'image_{i}.jpg'), annotated_image)

    # Save the bounding box coordinates in a text file for each
image
    with open(os.path.join(annotated_dataset_path,
f'image_{i}.txt'), 'w') as f:
        for annotation in annotations:
            bbox = annotation
            f.write(f'{bbox["x"]} {bbox["y"]} {bbox["width"]}
{bbox["height"]}\n')

    # Clear annotations for the next image
    annotations.clear()

cv2.destroyAllWindows()

# Print the total number of annotated images
print("Annotated dataset size:", len(car_images))
```

Output:

image_0.jpg	16/07/2023 11:33 PM	JPG File	28 KB
image_0.txt	16/07/2023 11:33 PM	Text Document	1 KB
image_1.jpg	16/07/2023 11:33 PM	JPG File	32 KB
image_1.txt	16/07/2023 11:33 PM	Text Document	1 KB
image_2.jpg	16/07/2023 11:33 PM	JPG File	17 KB
image_2.txt	16/07/2023 11:33 PM	Text Document	1 KB
image_3.jpg	16/07/2023 11:33 PM	JPG File	21 KB
image_3.txt	16/07/2023 11:33 PM	Text Document	1 KB
image_4.jpg	16/07/2023 11:33 PM	JPG File	16 KB
image_4.txt	16/07/2023 11:33 PM	Text Document	1 KB
image_5.jpg	16/07/2023 11:33 PM	JPG File	22 KB
image_5.txt	16/07/2023 11:33 PM	Text Document	1 KB
image_6.jpg	16/07/2023 11:33 PM	JPG File	21 KB
image_6.txt	16/07/2023 11:33 PM	Text Document	1 KB

Annotated Pictures Saved in folder name ‘annotated’

Annotated Picture



Annotated Picture Labeling (x y w h)

Failures:

After doing this process and some research we get that CNN doesn’t want labeling of pictures so we skip this step and move forward to get the dataset of only pictures of cars.

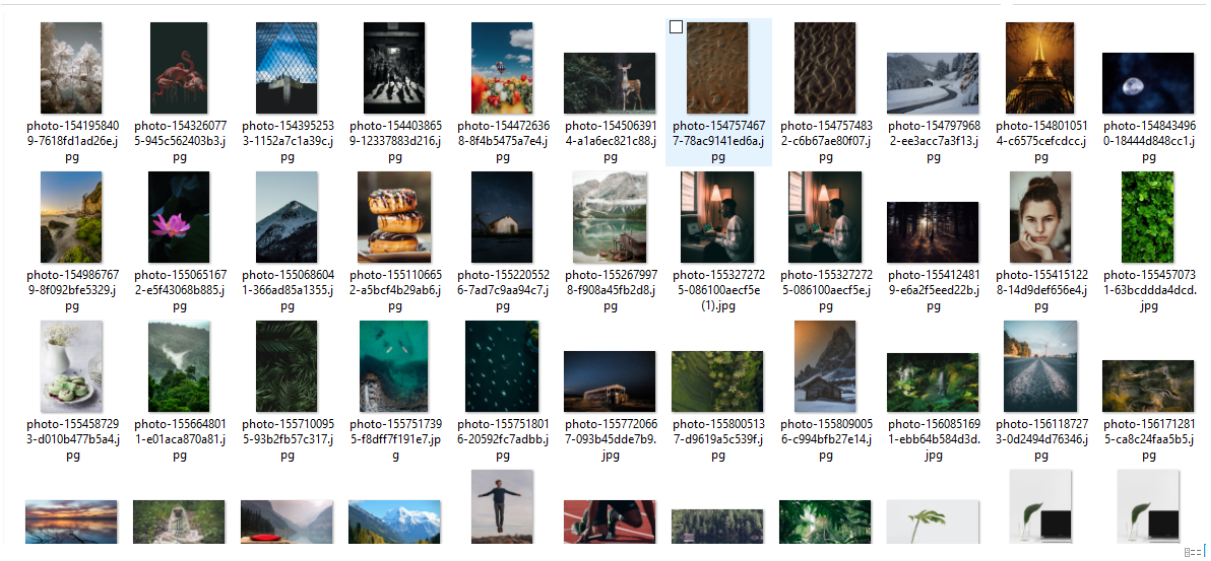
New Dataset:

Then we downloaded the new dataset of only car images and we have to download the images without car so our CNN model will train on both images types (cars, not cars) to predict the later input image whether input image consists of car or not.

And for random images collection we again download the dataset from kaggle https://www.kaggle.com/datasets/lprdosmil/unsplash-random-images-collection

Dataset Folder structure

Car Images



Not Car Images

<u>Dataset is balanced means there are same number of ‘car’ and ‘not car’ images</u>

Milestone 3: Car Image Detection using CNN

Github Repo:
https://github.com/saimdev/Car_Theft_Detection_System/blob/master/car_detection_CNN.py

Discussion:
Now using a new dataset, we create a custom CNN model. Some main features of code and model are written below:

1. Image size is 32
2. Random seed value is 42
3. Use scikit-learn for splitting dataset into train and testing data
4. Keras is used for creating custom model
5. Also used ImageDataGenerator for transforming images into different shapes
6. .h5 format is used for saving model
7. Did with 15 epochs
8. 'Adam' Optimizer is used
9. Categorical_crossentropy loss function is used
10. For input and hidden layers, relu activation function is used
11. For output dense layer, softmax activation function is used

Libraries Used:

- opencv-python
- numpy
- keras
- scikit-learn
- random

Python Code for Training and saving model:

```
import os
import numpy as np
import cv2
import random
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,
Dropout
from keras.preprocessing.image import ImageDataGenerator
from keras.utils import to_categorical

random.seed(42)

data_dir = './dataset'
```

```
categories = ['car', 'not_car']
img_size = 32

def create_data():
    data = []
    for category in categories:
        path = os.path.join(data_dir, category)
        class_num = categories.index(category)
        for img in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path, img))
                img_array = cv2.cvtColor(img_array,
cv2.COLOR_BGR2RGB)
                img_array = cv2.resize(img_array, (img_size,
img_size))
                data.append([img_array, class_num])
            except Exception as e:
                pass
    random.shuffle(data)
    X, y = [], []
    for features, label in data:
        X.append(features)
        y.append(label)
    X = np.array(X)
    y = to_categorical(y, num_classes=len(categories))
    return X, y

def create_model():
    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu',
input_shape=(img_size, img_size, 3)))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Flatten())
    model.add(Dense(64, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(len(categories), activation='softmax'))
    model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
    return model

def train_model():
```

```
    X, y = create_data()
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

    model = create_model()

    data_generator = ImageDataGenerator(rotation_range=10,
width_shift_range=0.1,
                                        height_shift_range=0.1,
zoom_range=0.1, horizontal_flip=True)

    data_generator.fit(X_train)

    batch_size = 64
    epochs = 15
    model.fit(data_generator.flow(X_train, y_train,
batch_size=batch_size), epochs=epochs,
              validation_data=(X_test, y_test),
steps_per_epoch=len(X_train) // batch_size)

    test_loss, test_acc = model.evaluate(X_test, y_test)
    print(f"Test accuracy: {test_acc}")

    model.save('car_detection_model.h5')

train_model()
```

Output:

```
20/20 [==============================] - 3s 124ms/step - loss: 0.1273 - accuracy: 0.9625 - val_loss: 0.3226 - val_accuracy: 0.91
59
Epoch 10/15
20/20 [==============================] - 2s 122ms/step - loss: 0.1134 - accuracy: 0.9705 - val_loss: 0.1206 - val_accuracy: 0.95
64
Epoch 11/15
20/20 [==============================] - 3s 131ms/step - loss: 0.3079 - accuracy: 0.9327 - val_loss: 0.6760 - val_accuracy: 0.88
16
Epoch 12/15
20/20 [==============================] - 3s 134ms/step - loss: 0.3927 - accuracy: 0.8450 - val_loss: 0.2679 - val_accuracy: 0.90
34
Epoch 13/15
20/20 [==============================] - 3s 124ms/step - loss: 0.2567 - accuracy: 0.8958 - val_loss: 0.2549 - val_accuracy: 0.87
85
Epoch 14/15
20/20 [==============================] - 3s 123ms/step - loss: 0.1851 - accuracy: 0.9278 - val_loss: 0.1301 - val_accuracy: 0.95
33
Epoch 15/15
20/20 [==============================] - 2s 122ms/step - loss: 0.1344 - accuracy: 0.9557 - val_loss: 0.1299 - val_accuracy: 0.94
08
11/11 [==============================] - 0s 14ms/step - loss: 0.1299 - accuracy: 0.9408
Test accuracy: 0.940809965133667
E:\Python\Lib\site-packages\keras\src\engine\training.py:3000: UserWarning: You are saving your model as an HDF5 file via `model
.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.
keras')`.
  saving_api.save_model(
```

After running this code

Loss	**12.99%**
Accuracy	**94.08%**

TESTING

Github repo:
https://github.com/saimdev/Car_Theft_Detection_System/blob/master/testing.py

Libraries used:

- opencv-python
- numpy
- keras

Python code for testing this model:

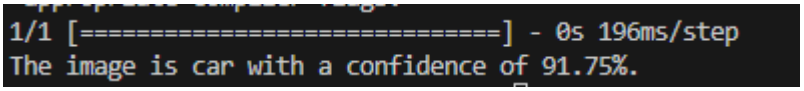
```
import cv2
import numpy as np
from keras.models import load_model

def preprocess_image(img_path, img_size):
    img = cv2.imread(img_path)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, (img_size, img_size))
    img = np.expand_dims(img, axis=0)
    return img

def predict_car(img_path, model_path):
    img_size = 32
    model = load_model(model_path)
    img = preprocess_image(img_path, img_size)
    prediction = model.predict(img)
    class_idx = np.argmax(prediction[0])
    class_label = 'car' if class_idx == 0 else 'not a car'
    confidence = prediction[0][class_idx] * 100
    return class_label, confidence

model_path = 'car_detection_model.h5'
image_path = 'car.jpg'

class_label, confidence = predict_car(image_path, model_path)
```

```
print(f"The image is {class_label} with a confidence of
{confidence:.2f}%.")
```

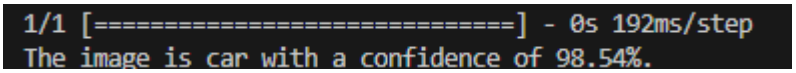
Images Used for Testing:

Output:

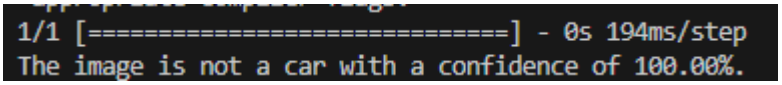
For car image



For rush image



For human image



For horsecar image