Quadcopter Vehicle Recognition for Law Enforcement

Aarav Prakash

10/31/2023

1. Abstract

Quadcopters are modern and fast, and with quadcopter technology improving drastically. Flight controllers and micro cameras are constantly getting cheaper and more reliable. With this change in technology, many first responders have begun to use drones to aid their work. However, the police force has not found many ways to implement drones to full capacity. This leads to my research question: How can drones be used to modernize and aid the police force? The current world record for the fastest quadcopter speed was set at 235 miles per hour by the XLR V3, which was made from off the shelf parts. The XLR V3 also has a camera on it, allowing for computer vision models to be run [5]. With this technology readily available, it could help the police force improve by large amounts for a cheap cost. We found that the color, make, and model of a car in an automated car detection system would be useful to the police force. In our approach, we created two Convolutional Neural Networks. One detected the make and model of a car, the other detected the color.

2. Introduction

Quadcopter technology is surprisingly mainstream. It is very powerful, and it can be seen in Russia's war in Ukraine. The DJI Mavic two and three are being used to drop grenades or help scout the opponent [6]. Also, Russia has purchased large quantities of IFlight FPV quadcopters to use against Ukraine [7]. Quadcopter technology is incredibly relevant, however, it isn't being used to its complete potential when it comes to policing. We sought out to create an AI that is able to detect the color, make, and model of a car on the road from a quadcopter. This AI could be used in a variety of ways. The main way that we purposed it for was to detect cars that may be involved in an amber or silver alert. Our approach was to create two Convolutional Neural Networks, one of which detected the color of a car, and the other that detected the make and model. Our data will be verified by the accuracy of the model. The more accurate a specific model is, the more useful it is in detecting certain cars.

3. Background

I. Quadcopters

A quadcopter is a Unmanned Aerial Vehicle (UAV) which has 4 propellers and motors of which it can use to hover, or move in different directions. There are two different types of quadcopters, FPV quadcopters, and stabilized quadcopters. The difference between the two is that an FPV quadcopter relays analog signals from the quadcopter to a pilot, who has to manually maneuver the quadcopter without help from gyro or GPS. Stabilized quadcopters, on the other hand, use GPS and gyro to keep steady or stable. For this research, we will be testing with a stabilized quadcopter. A stabilized quadcopter consists of a frame, a flight controller (FC), and electronic speed controller (ESC), 4 motors, a camera, a video transmitter (VTX), and a receiver. The FC controls all of the gyro equipment and houses the CPU of the quadcopter, it also connects to everything on the quadcopter. The ESC takes power from the battery and distributes it to motors at the instruction of the FC. The VTX and camera send video signal back to the video receiver. The receiver is receives radio signals from the pilot of the drone.[4]

II. Sighthound

Sighthound is a car detection system based off of the compCar dataset[reference here]. It detects the make and model of cars, however it does not detect color. With a verification accuracy of 93 percent, Sighthound utilizes traffic cameras and photos taken by phones to train and verify their dataset. Sighthound also can detect the license plate of a car, however, with quadcopters, the license plate would not be easily visible, due to blurry camera vision and the angle at which a quadcopters are seeing the car.[1]

4. Methodology and Models

I. Datasets

We used two datasets for the creation of this system. The first was the VCoR (Vehicle Color Recognition dataset [2]. The other was the Stanford Cars Dataset[3].

II. VCoR Dataset and Pre-Processing

The VCoR dataset was used to create a Convolutional Neural Network (CNN) that detected the color of a car. The VCoR dataset has 15 classes and around 10.5 thousand images: 7.5 thousand training images, 1.5 thousand testing images, and 1.5 thousand validation images. Due to RAM limitations, we only used the training and testing images. Therefore, we used around 17% of images to test and 83% to train the CNN. In order to process the data, we took the folders of data and sorted it into a large array of images, and another with labels. The images could not be grayscale because we were trying to detect color, therefore, the images had to be converted from jpegs to three dimensional arrays. The images were also not formatted in RGB, but BGR. We utilized OpenCV to convert the images to RGB. We also resized the images to 100x100 using interpolation.

Figure 1. Training image 58 pre pre-processing

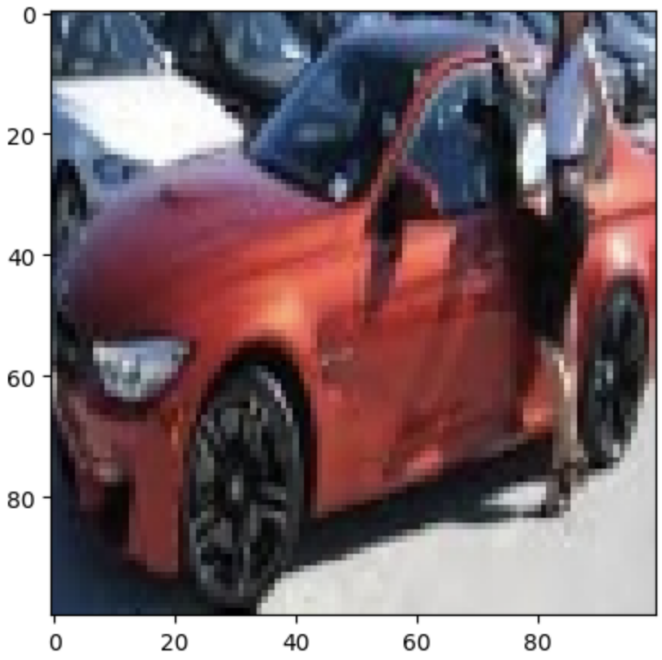

Figure 2. Training image 58 post pre-processing.

III. Stanford Cars Dataset and Pre-Processing

The Stanford Cars Dataset[3] was used to create a CNN for detecting the make and model of the classes. It has 16,185 images, with 8,144 training images and 8,041 testing images. With 196 classes, the classes give you the make, model, and year of the vehicle. We were only able to use the 8,144 training images for this CNN due to the testing images not being labeled correctly, and also RAM limitations. To preprocess the Stanford Cars Dataset, we created a large array with all of the images. The labels and annotations came in a mat file, which we created a pandas dataframe out of. The annotations came with the bounding boxes and the class. We only needed to use the labels, so we created a large array with all of the labels with the relative image path. We then reshaped all of the images to 128x128 using interpolation again.

Figure 3. Stanford Cars Dataset image pre pre-processing.

Figure 4. Stanford Cars Dataset image post pre-processing.

IV. Color Model

For the color model, we created a Convolutional Neural Network. We first had to preprocess the data(described above). After this, we utilized Tensorflow to create a CNN with 1 stack of Conv2d and MaxPooling layer, and then a flatten and dense layer. Throughout the process we realized that the model

was overfitting, so we ended up increasing dropout and kernel size. Our final batch size was 16, and epochs were 10(Interchange these).

```
model = Sequential()
model.add(Reshape((100, 100, 3)))

model.add(Conv2D(32, (5, 5)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(4, 4)))
model.add(Dropout(.5))


model.add(Flatten())
model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dense(15))
model.add(Dropout(.5))
model.add(Activation('softmax'))

# initiate RMSprop optimizer
opt = keras.optimizers.RMSprop(lr=0.0001, decay=1e-6)

model.compile(loss='sparse_categorical_crossentropy',
              optimizer=opt,
              metrics=['accuracy'])
```

Figure 5. Structure of the VCoR Dataset model.

V. Make and Model Model

For the make and model CNN, we ended up using a similar structure, however, due to the sheer amount of classes, the model was bound to overfit. We ended up cutting some of the classes to help the accuracy of the model. We used two stacks of Conv2D and MaxPooling layers. After defining the architecture of the model, we compiled it with the RMSProp optimizer and the categorical cross entropy loss function.

```
model = Sequential()
model.add(Reshape((128, 128, 3)))

model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(.5))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(.5))

model.add(Flatten())
model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dropout(.2))
model.add(Dense(196))
model.add(Activation('softmax'))

# initiate RMSprop optimizer
opt = keras.optimizers.RMSprop(lr=0.0001, decay=1e-6)

# Let's train the model using RMSprop
model.compile(loss='sparse_categorical_crossentropy',
              optimizer=opt,
              metrics=['accuracy'])
```

Figure 6. The structure of the Stanford Cars Dataset model.

VI. Quadcopter

For these models, we constructed a quadcopter using the TBS Source One V5 Carbon Fibre Frame [5]. We used a SpeedyBee F405 ESC and FC stack with 1800KV motors. [10]

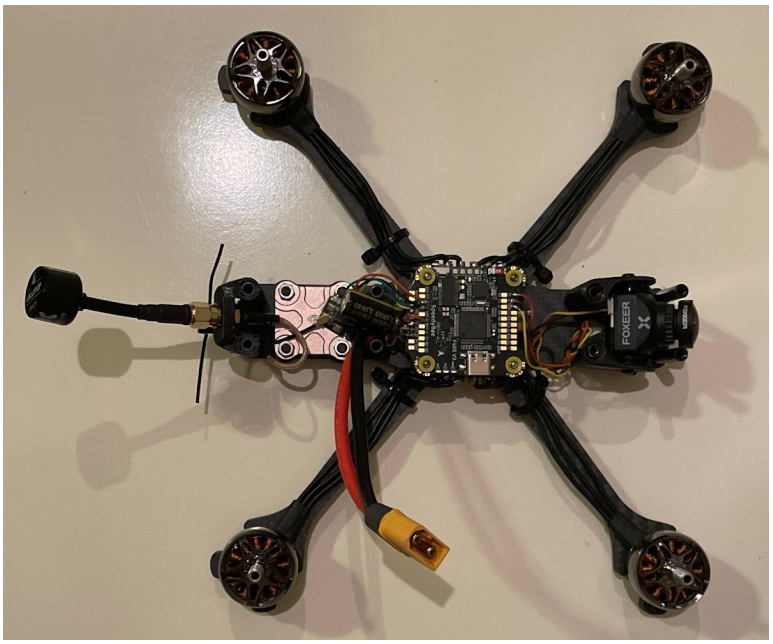

Figure 7. The Quadcopter created for these models(without cover)

5. Results

I. Results of Models

Overall, the color model seemed to be fairly accurate, with a validation accuracy of 75 percent. However, the make and model CNN was not working as well as intended, with a validation accuracy of 10 percent.

II. Results of Quadcopter

The Quadcopter was very fast, with a top speed of 100 miles per hour. There were some issues with battery life. On average, the battery goes from its maximum cell capacity of 4.2V to 3.3V in around 3 minutes of flight time. However, its capacity is lower than most with 1500mAH, and its cell count is also lower than most with 4 cells. [8]

Figure 9. OSD (On Screen Display) of the quadcopter in action.

III. Discussion

Our results were not as good as Sighthound due to the make and model not working as well as anticipated. The primary reason was the amount of classes (196) in the make and model CNN was too high. The color model was more accurate because it only had 14 classes. Also, we were using less data than Sighthound. A model will never be 100 percent accurate, and this brings some ethical questions into play. Such as, who is to blame if a quadcopter locks onto the wrong target, or finds the wrong target? Will ethical reasons prevent technology like this to exist? How will lawmakers around the world deal with this technology? While there are not many answers to these questions, the technology exists to create these models and quadcopters.

6. Conclusion

We set out to create an AI that was able to detect the make, model, and color of a car. Our results prove that this theory could work in a variety of first responding issues. As stated before, quadcopters are incredibly fast and have cameras on them. While our models were not entirely accurate, in the future, we

intend to reduce the amount of classes in our data and use new, more modern datasets. We also plan on updating the quadcopter with a longer battery life. The quadcopter constructed had clear camera vision and a large top speed, making it possible to aid law enforcement. However, our CNNs were not accurate and most likely would not aid law enforcement. In the future, improvements such as use of different datasets may help improve the CNNs to aid law enforcement.

7. Acknowledgements

I would like to formally acknowledge Inspirit AI for making this project possible. I would also like to acknowledge my mentor, Ronil Synghal, for helping me write the code and paper for this project.

8. References

1: https://arxiv.org/pdf/1702.01721.pdf
2: https://www.kaggle.com/datasets/landrykezebou/vcor-vehicle-color-recognition-dataset
3:https://www.kaggle.com/datasets/jessicali9530/stanford-cars-dataset
4:https://www.space.com/what-are-fpv-drones
5.https://betaflight.com/blog/New%20World%20Record%20for%20Fastest%20Drone#:~:text=The%20Guinness%20World%20Records%20has,love%20to%20reach%20those%20speeds.
6.https://www.c4isrnet.com/global/europe/2023/10/23/ukraine-continues-to-snap-up-chinese-dji-drones-for-its-defense/#:~:text=The%20Oct.%208%20statement%20made,with%20military%20utility%20can%20permeate
7.https://www.youtube.com/watch?v=qtHOg063tXs
8. https://oscarliang.com/lipo-battery-guide/
9.https://www.team-blacksheep.com/products/prod:sourceone_v5
10. https://oscarliang.com/speedybee-f405-v3/