Date of publication April 27, 2022, date of current version April 21, 2022.

Digital Object Identifier

Image Processing for Object Detection

ALEXA MANSSA, HARLEY VALDEZ, RYAN HANRAHAN, LEILA ALAMOS, AND BRETT BARTOSHEVICH

ABSTRACT Image recognition is a widely used observation technique for software to identify objects or people within an image. Using coding methods for sifting through databases, we will be creating a Python code to recognize objects within a camera's frame of reference and accurately identify the object to the user. Our goal is to obtain multiple databases of common objects from online sources to use as our library of reference objects. The code will identify the individual pixels of the image and then match the pixels to the object library. All the objects within the frame will be able to be identified and displayed in the output. The results of this project will yield a Python algorithm for precise object detection in an image.

I. INTRODUCTION

Object recognition is a task of computer vision in the field of machine learning. It is one of the more difficult and stimulating tasks of computer vision.

Object detection or recognition aims to detect all instances of objects of a known class, such as people, cars or faces in an image. Usually, only a small number of instances of the object are present in the image, but there are a very large number of possible locations and scales at which they can occur that need to be explored in a way. or another.

Each detection of the image is signaled with some form of pose information. It's as simple as the location of the object, a location and a scale, or the extent of the object defined in terms of a bounding box. In some cases, the pose information is more detailed and contains the parameters of a linear or nonlinear transformation. For example, for face detection in a face detector, the locations of the eyes, nose and mouth can be calculated, in addition to the boundary area of the face. Similar projects have been done using this detection technique where researchers utilize object recognition and detection in remote sensing images.

For a machine to perform effective object identification through a neural network, it will have to be programmed with a machine learning algorithm using deep learning methods. This means the machine will be able to function and improve its processes autonomously, or without the manual input of a user. One of the best methods for object identification is Deep Learning. Deep Learning is a process where a machine performs iterative calculations to recognize various patterns by its own process. [2] This is a powerful tool; however, this method often runs into a roadblock of low performance due to over-fitting objects. A solution to this problem is employing the use of a Convolutional Neural Network (CNN). A Neural Network designed in python will require a CNN, which can be created from resources like Tensorflow and Keras. These two resources are open-source dataflow and neural network software that enable rapid experimentation to efficiently train a neural network in python.

One real world application of machine learning can be seen is Tesla's Autopilot feature which makes their vehicles fully autonomous when utilized. Some of the subsystems within the software features safety measures including automatic emergency braking, forward and side collision warnings, obstacle aware acceleration, blind spot monitoring, lane departure avoidance, and emergency lane departure avoidance [3]. The underlying algorithm for Autopilot creates a very distinct mapping of your surroundings using cameras and sensors, and a series of neutral networks. The neutral networking mirrors brain functions for a computer that initiates the deep learning processes. These techniques, when combined, allow the system to make decisions in "complicated real-world situations under uncertainty" [4].

Tesla uses bootloaders which is a program that can load an operating system (Linux in this case) when the computer is turned on. This means that when the autopilot feature is activated by the driver, customized Linux kernels are pulled up which directly interface with the computer's hardware and the embedded code. The advantage of this style of coding is that "over-the-air updates" can be made in real time. Overall, this provides the driver with the most up-to-date, self-sufficient system that is reliable. The presence of neutral networks in the code aids the accuracy of the system. Training the neutral networks in both perception and control involves creating a 3D space of objects to learn complicated and diverse scenarios. The Autopilot software overall involves 48 networks that take about 70,000 GPU

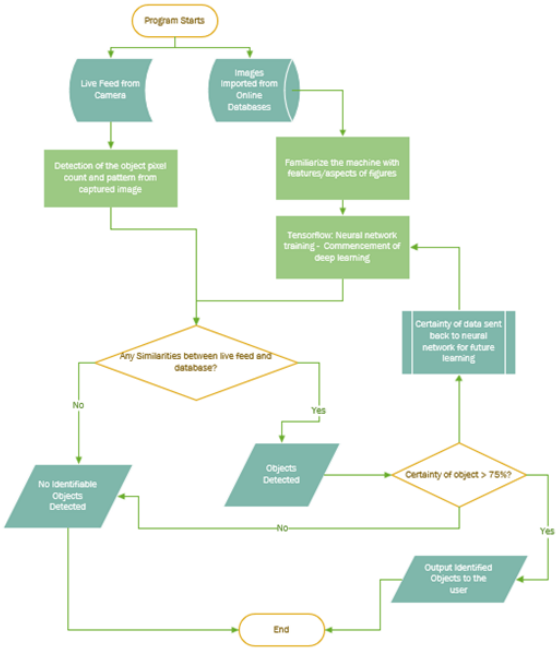

FIGURE 1. Algorithm flowchart for object detection software in Python

hours to train [4].

II. SYSTEM DESIGN

The system architecture begins with importing image databases from an online source to be used in the comparison of live camera feed images to pre-existing library images. Upon startup, the algorithm follows the processes specified in Figure 1. Each process goes through decision-making to accurately identify the contents of the image to the user. To start the object recognition process with the program, the camera must be functioning and providing live frames so that the data can be fed to the program. With this, the input is the live video of the object that is in front of the camera for the program to attempt to recognize it through the database that it used to learn from.

The input images are then segmented into a series of smaller bounding boxes. This is a key step in pre-processing the image so it may be properly dissected by the Convolutional Neural Network. During this segmentation process the input images are compared to previously stored models of similarly detected images. Next, the image is run through a feature extraction algorithm for rectangular areas to determine if any valid objects exist. All valid overlapping bounding boxes are then merged into one large bounding box and labeled. Lastly, a probability calculation is created for the output based on accuracy to previously stored models.

Object recognition is the key output of deep learning and machine learning algorithms. While training for machine learning, you pass an algorithm with training data. The learning algorithm finds patterns in the training data such that the input parameters correspond to the target. The output of the training process is creating and teaching a program

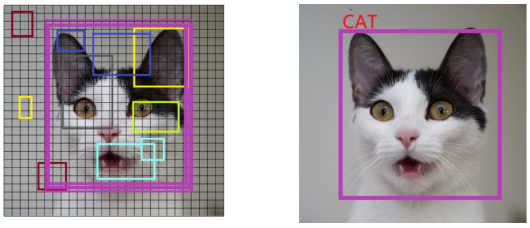

FIGURE 2. Image segmentation and processing of an image of a cat by real time object recognition

that can then be used to accurately recognize objects. Once the model has been trained, deep learning algorithms offer a high level of accuracy but require a large amount of data to make accurate predictions. Ultimately, when the model has received a significant portion of data where it has "learned", it will produce an output that will identify the object presented to the model when provided with a live camera feed input.

III. METHODS

The first portion of the code is made up of machine learning. Object recognition is a complex process so the TensorFlow Object Recognition Application Programming Interface (API) will be used to help along with this process. A custom python environment was created with its own dependencies and packages to support this API. The code initializes itself by importing 16 packages, configuring a model, runs a segmentation for a single image, compares it to the loaded model, and outputs a bounding box of specific portions of the image for further comparison. Then, utilizing OpenCV, the program opens the webcam, takes the input feed from the webcam, converts it into an array, and continuously does this process while comparing the incoming data to the "machine learned" data so that an output may be produced. Once a recognizable outcome is made, a bounding box is produced around this outcome with the name of what the machine thinks it is based on the database it had learned from.

IV. RESULTS AND DISCUSSION

The primary result of our project is currently in the process of being constructed due to us only being to produce results from an inputted image, instead of a live camera feed. We ultimately plan to grow our neural network for our code to be able to identify objects as the output. Once this is done, we will assess the needs of improvement and critique our progress to ensure our understanding of the assignment at hand. Currently, the basis of the neural network has been established along with access to the TensorFlow API. The next step in the process is accessing the object recognition toolbox in the Model Garden for TensorFlow. Creating reference models, training methods, and modeling solutions provides a function for the code that contains learnable parameters for the machine. These learnable parameters map out the inputs and outputs and train the code to provide desired outcomes. This will yield a more accurate depiction of the images being assessed.

The next step of this code involved establishing the TensorFlow project directory structure within the code to aid in the object detection API installation. The TensorFlow project directory was established in the python code after downloading and extracting the TensorFlow model garden from the repository. Due to the nature of the hardware used, the code could not be executed initially. However, using a different hardware setup, the code was able to be executed and outputs were achieved.

Despite the many issues that arose while creating the object detection algorithm, we sought an alternative which yielded a successful object detection output. This was achieved by using an online version of the TensorFlow object detection code. The drawback to this choice was the omission of the models we custom created for segmentation of images and a limit to the images we could use for final object detection thus far. After utilizing several helper functions given by the online version of code, we were able to pull in two images and successfully process them in a static format. The first is a before/after shown below:

FIGURE 3. Image pulled into python program and successfully processed through object detection

From figure 3, we can see the pixel graph of the image and then its image processing with bounding boxes to detect the multiple objects within the image. For each individual object shown there is a bounding box with a confidence percentage calculation attached. To show just how versatile this algorithm can be, we can see two additional models with numerous objects of high similarity represented by figure 4.

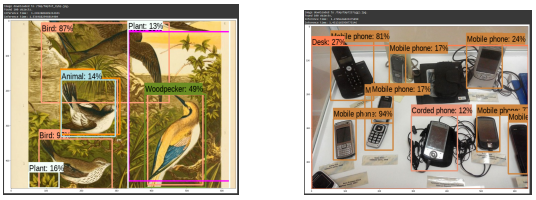

FIGURE 4. Image processing of Bird and Cell phone images with highly similar objects

Moving forward, Protobuf will then be downloaded and installed to get all the model training parameters configured in the TensorFlow directory. This is essential in making sure that we have high accuracy in the object detection output. In addition, the COCO API will be installed which allows the user to read and extract annotations in a manner that is quick and convenient. This API can be run in the Linux terminal to be added into the TensorFlow directory structure. Upon installation of all the object detection API, we will be able to output highly accurate object analysis to the user.

V. CONCLUSION

Despite currently only having results utilizing a stationary picture instead of live camera feed due to the code still being under construction, our goal remains true and near complete. This goal being that our program being able to use an integrated camera, scan an object in front of it, and classify that object. Limitations that we may run into once the program is complete are the camera's scanning capability for certain objects, an object in the database not recognizing its correspondence being presented to the camera, or an incorrect classification of an object. Overall, as we continue to develop the program and its functionality in its ability to recognize objects, this project has contributed greatly to our knowledge and applications of how object/facial recognition works and its importance to the ever-advancing world of technology and programming.

REFERENCES

[1] S. A. Fatima, A. Kumar, A. Pratap and S. S. Raoof, "Object Recognition and Detection in Remote Sensing Images: A Comparative Study," 2020 International Conference on Artificial Intelligence and Signal Processing (AISP), 2020, pp. 1-5, doi: 10.1109/AISP48273.2020.9073614.

[2] Prasad, K. R., Sravani, P. C., Mounika, P. S. N., Navya, N., Shyamala, M. (2019). MACHINE LEARNING BASED OBJECT IDENTIFICATION SYSTEM USING PYTHON.

[3] Tesla. (2021, June 16). Autopilot and full self-driving capability. Tesla. Retrieved February 10, 2022, from https://www.tesla.com/support/autopilot: :text=How

[4] Artificial Intelligence Autopilot. Tesla. (n.d.). Retrieved February 10, 2022, from https://www.tesla.com/AI