Khanh Huynh

ITAI 1378

Prof. Patricia McManus

10/30/2024

L09

1. Conceptual Understanding

- What is the main difference between image classification and object detection? How is this difference evident in the output of this exercise?

Two important tasks in computer vision include object detection and image classification, though they are different in objective and technical difficulty. The basic idea behind image classification is to assign a single label to the whole image, locating the dominant object or scene only at an abstract level but without indicating where that object of interest is in the image. In essence, a categorization model is intended to answer the "what" question about the image, not the "where." For instance, given an image containing a cat and a dog, an image classification model would simply give out the label for the dominant object-however, if the dog is more centrally placed or larger, the label could be "dog." Indeed, this understanding is simpler than object detection: not only identifies object’s class but also localizes objects through bounding boxes surrounding each detected object. This complex method is reflected in the exercise's output, which contains bounding boxes and labels for every object in an image, which provides both the what and the where of a complicated scene.

-Explain why we chose the SSD MobileNet V2 model for this task. What are its advantages and limitations, especially in the context of limited computational resources?

The SSD MobileNet V2 model was intentionally selected for purposes of this assignment because it presents some balance between computing efficiency and good object detection. A single-shot detection (SSD) is used to accelerate the detection by combining object localization and classification in a single forward pass, which is usually faster than two-stage detectors, such as Faster R-CNN. Fast detection speed in this model is especially useful in real-time applications where the speed of the responses is key. The basic backbone of the model, MobileNet V2, employs depth-wise separable convolutions that reduce the number of trainable parameters and computational complexity without sacrificing speed. Besides, MobileNet V2 is a design output-based structure known for its efficiency and expected resource-constrained conditions such as mobile or edge computing devices. It is practically the case that they fail because the limited processing capability and memory fractionally stress and eradicate a competing work environment to conciliate tasks such as object recognition with a model like SSD MobileNet V2.

However, in pursuit of those objectives, some limitations arise. Because SSD MobileNet V2 is less accurate than the heavier models, such as Faster R-CNN, it suffers in some situations through the effective detection of small, nearby objects that require tight localization. To support real-time performance, a lightweight model like MobileNet v2; therein however may lie the compromise upon the quality of bounding boxes in all cases, particularly in cases like overcrowded or highly detailed scenarios. These limits serve as a paragon of the trade-offs informing SSD MobileNet V2, which pushes speed and efficiency over absolute accuracy. Such a choice thus becomes an illustration in the real world that demonstrates the emphasis on controlled resource use and utility performances; rendering it a right model for decent if somewhat speedy object recognition on limited hardware.

1. Code Interpretation

-Describe the role of the “find\_images\_with\_classes” function. Why is it useful when working with a large dataset like COCO?

The find\_images\_with\_classes is a very important method in thriving through humongous data sets because it involves finding the images containing the target classes. The method is suitable for bounding big datasets such as COCO (Common Objects In Context), which has millions of annotated images with many object categories, down to the images relevant for a certain particular project or study. Thus, the activities performed by this function relate to optimization, which, based on memory as well as processing time, works by not loading a full set that would be computationally too demanding and possibly filled with data not related to the current problem, but only a set of images with the desired classes given by the user.

-In the plot-detections function, how does the threshold value (0.5) impact the number of objects displayed?

The threshold argument in the plot\_detections function is fixed at 0.5, indicating the confidence score for object visualization through a detection. In other words, it indicates the minimum confidence score required for a specific detection object to be plotted. The confidence score produced by the model reflects the probability that a detected bounding box contains the designated object. With this threshold set at 0.5, if the confidence score is less than 50%, the function will not display the respective detection.

Raising the threshold (for example, to 0.7 or 0.8) would produce fewer detectable objects; only those with higher confidence would be kept for plotting. This might diminish false positives in the detection results but would mean overlooking some real detections with lower certainty. A lower threshold (e.g., 0.3) would let through the display of more objects, even if at low confidence scores. This would likely improve recall, but it could also increase false positives.

-Explain how the heatmap visualization helps you understand the model’s confidence in its detections

A heatmap representation can also be thought of as a visuo-spatial representation of the model's conviction about the identification of objects in an imager. This means that superimposing a heatmap onto those areas such that the model assigns high probability values or confidence scores makes it easier for the user to see which regions the model "thinks" most likely contain a certain object. Typically, the level of confidence is associated with a color within the heatmap; therefore, warmer colors (e.g., red and yellow) imply a greater level of confidence and vice versa for cooler colors (e.g., blue). However, I could not change much difference from my code to have the appearance of heatmap in the picture.

1. Observing Results and Limitations:

-Which types of objects does the model tend to detect more accurately? Which ones are more challenging? Can you explain why?

Objects appearing in the center of an image, those that are brightly lit, and those easily distinguished by shape or color-such as cars, humans, and animals in easy scenes-are usually more easily detected by the model. This is because such objects stand out strikingly against their backgrounds and have relatively clearer edges and textures; thus, they provide the model with unequivocal features to learn and recognize.

On the contrary, small, partially occluded, or camouflaged objects are hard for the model to recognize. Such objects include distant traffic lights, partially covered faces, or objects that fade into the background because they have a low level of contrast against it and thus provide fewer distinguishing cues that lower the detection level, whether it is the object detected or its classification. Similarly, items present in dark or dense environments present another hurdle as models find it difficult to extract discrete patterns necessary for detection.

The performance difference results from the fact that smaller or hidden objects occupy fewer pixels in an image; therefore, fewer details are available for models to use in extracting features. Furthermore, the properties of the colors or shapes that, in less cluttered backgrounds, are more easily interpreted become less distinguishable when the objects in question are slammed into messier backgrounds, which may confuse the model. These problems of detection highlight model limits, especially the models optimized for efficiency, which favor speed and could not have the required resolution or depth of feature needed for smaller and less structured objects.

-Observe the bounding boxes? Are there any instances where the boxes are inaccurate or miss the object entirely? What factors in the images might be contributing to these errors?

When processing and displaying images with detection, the bounding boxes detected objects within 3 images are accurate and did not miss the object entirely. Some partial objects that were not identified were also covered. However, some instances where it might be inaccurate were often deprived from various factors such as occlusion (objects being hidden or obscured), scale and size, complex background and clutter, and low contrast and poor lighting.

-How would you expect the accuracy of the model to change if we had used the entire Pascal VOX 2002 dataset? Why?

The increased size of the training domain may, thus, result in superior accuracy for such a model application on the entire Pascal VOC 2002 dataset. More extensive training data, indeed, furnish the model with variably expressed samples used for learning generalized, robust feature representations in different contexts, lighting conditions, scales, and orientations. If the model gets to see more examples of any one object class, it will be better at recognizing these when placed in new and varied scenes.

More specifically, a full Pascal VOC 2002 dataset would likely help models recognize small diseased leaves or those partially occluded, or those rarely seen in particular poses or orientations. This should introduce greater variability and minimize the effect of overfitting the model to specific instances from a smaller subset, thereby enhancing generalization for the unseen data. Also, a more thorough dataset would provide better representation for each class and hence would not lead the model to develop any prejudices because of an unbalanced or limited dataset and thus improve the overall performance and accuracy.

With the reduced datasets, however, while greater accuracy can be achieved, this would also increase the ask on computations to support training-a chronological aspect requiring greater amounts of learning memory and training time and thus tends to be limiting within environments having fairly low resources.

4. Critical Thinking

-How could you modify the code to detect a specific set of objects, like only animals or only vehicles?

Since the dataset included a specific set of objects such as animals, vehicles, etc. I could modify from target\_classes and apply filter detection from plot\_detection. This tweak allows me to restrict the output to display only the selected categories, facilitating a concentrated focus on certain objects of interest in the object detection task.

-If you wanted to train your own object detection model, what steps would you need to take? What are some challenges you might encounter?

Step-by-step process to train my own object detection model:

-Data collection and Annotation

-Data preprocessing

-Model Selection

-Training Setup

-Training model

-Evaluation and Fine-Tuning

-Deployment

Challenges: Object identification models are often hard to train due to computational requirements, data requirements, and the complex trade-offs between speed and accuracy. Training for object detection is memory-intensive and modulates requirements for powerful GPUs to handle large datasets of high-resolution images. Training time goes much higher when not using updated tech, turning much inefficiency and infeasibility for a range of users. Operations after training that include model inference and evaluation often require vast amounts of resources, more so with edge devices with reduced processing capability. Data quality and quantity are also major problems. Effective object detection models learn from large collections of well-annotated datasets. It would take time and much ado to manufacture high-grade object-related annotations with correct bounding boxes. Moreover, datasets must cover the object occupying the various conditions, lighting environment, and position. Finally, for real-time applications such as autonomous driving and mobile apps, balancing inference speed and accuracy becomes difficult. Other high-accuracy models, such as Faster R-CNN, are too slow for real-time use while SSD and YOLO sacrifice accuracy to gain speed.

-Given the limitation of this model, in what real-world scenarios might it still be useful for object detection

This model becomes very useful in applications that do not require the precision of the outputs but need real-time detection, such as in emergency response, where it helps drones to quickly find people in disaster zones or identifies landmarks to navigate rocky terrains. Parking management could be assisted in identifying the presence of free parking slots or counting the number of cars in parking spaces, providing a quick and cost-friendly solution in the monitor of occupancy. For construction site monitoring, it could be used to detect the presence of safety gear (such as helmets or vests) on workers, so as to ensure safety compliance without the requirement for high accuracy. In waste management, the model could help in aiding automated sorting systems by categorizing recyclable material into general categories (such as plastics, glass, or metals), thus speeding up the processing at recycling facilities. Overall, the efficacy of the model allows it to run on low-power devices set out in remote environments for the purpose of identifying specific animals, such as endangered species, for population monitoring.

5. Bonus Points

-Compare and Contrast with SSD MobileNet V2 in terms of accuracy, speed, and resources requirement

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Accuracy (mAP) | Speed (FPS) | Resource Requirement |
| SSD MobileNet V2 | 22-30 | ~60 | Low (ideal for mobile devices) |
| Faster R-CNN | 30-40 | 5-10 | High (requires powerful GPUs) |
| YOLO | 30-45 | >30 | Moderate (medium GPUs) |
| EfficientDet | 40-52 | Moderate | Varies (optimized for accuracy) |
| CenterNET | ~40 | 20-30 | Moderate (decent GPUs needed) |

-What difference do I notice from all models

SSD MobileNet V2 might prove incompetent in dealing with objects with small dimensions that are less frequent, thus jeopardizing their detection in cluttered scenes. Faster R-CNN are arguably more precise and accurate in finding objects, with them being extremely proficient in accurately finding small objects or detecting objects in complex environments due to their slow processing speeds, thus making them not particularly suitable for real-time usage. Both YOLO (v3/v4) and Faster R-CNN provide a better option between speediness versus detection quality but are prone, at times, to classification of smaller objects. EfficientDet- this probably combines great detection quality and efficient resource utilization with finding a more extensive range of classes while operating near real-time although I could not find the model to test it out. Lastly, CenterNet stands as mostly location accurate based on the finding on online resource, which would apply well to a class of applications in which precise positioning is required but may lose out on detecting smaller objects.