Machine Learning Engineer Nanodegree

Capstone Proposal

Nathaniel Watkins
July 12th, 2018

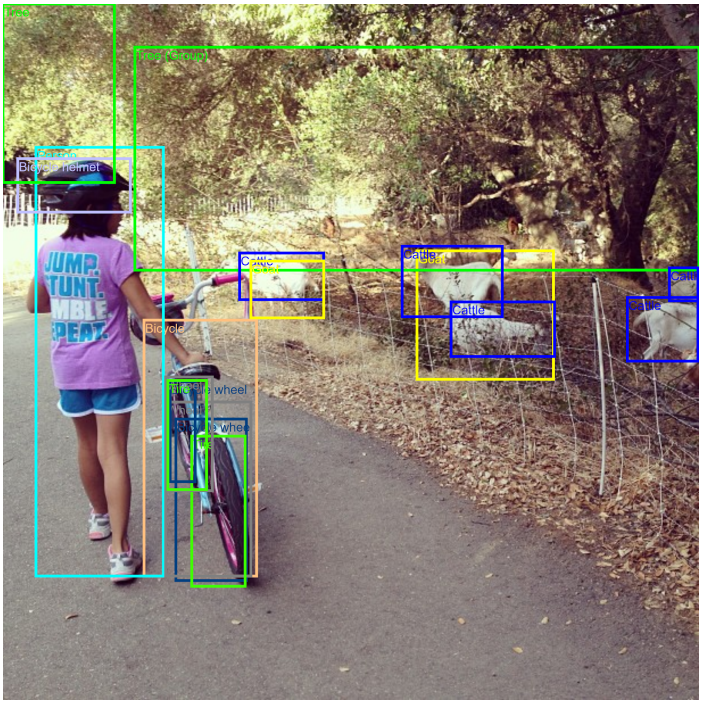
Proposal: Kaggle Competition - Google AI Open Images - Object Detection Track

Competition:
https://www.kaggle.com/c/google-ai-open-images-object-detection-track
Dataset:
https://storage.googleapis.com/openimages/web/index.html

Domain Background

Computer Vision (CV) is getting better and better every day, especially when focused on narrow tasks, such as detecting skin cancer; however, CV is still orders of magnitude weaker than humans when it comes to interpreting everything going on in a given picture.

For example, while CV can reliably label the Mountain Goat in the picture on the left[1], I'm sure most people would be able to identify several other things in it, such as a cloudless sky, boulders, bushes, and more. In order for Machine Learning to be able to benefit people on a whole new level, it'll need a much more comprehensive CV system that can extract almost all the important details out of a frame as seen in the picture below[2]:

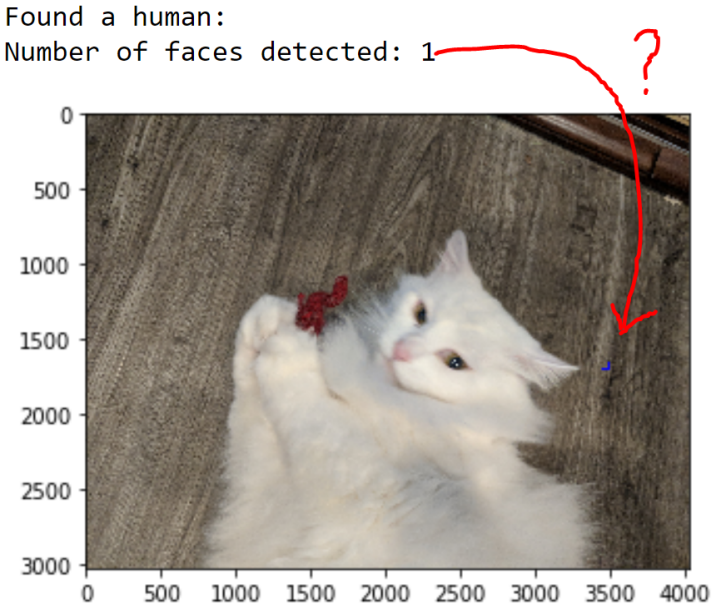


[1] *Mountain Sheep by Tony Hisgett (License)*
[2] *#goats #chivas #morning #bike #rider by Ray Bouknight (License)*

I would like to play a part in driving the state-of-the-art in CV, thus enabling new, life-improving applications, some that we can't currently predict. Just like no one really predicted that combining GPS and a ubiquitous data connection in a device that people always had with them would lead to transportation revolutions like Uber and Lyft, I see this challenge like adding GPS to phones, and I want to help make it happen!

Problem Statement

In most CV applications, the model is trained to output only 1 result (or sometimes a confidence list), thus only being able to handle pictures with just 1 subject. But we all know that real world applications are messy and many pictures (or video feeds) will have multiple items that could be considered a subject, or worse yet the CV model might completely miss the point and focus on an irrelevant part of the picture. For example, see the picture to the right[3] where a model trained to recognize dog breeds fixated on a spot of the floor thinking there was a human there, and completely ignored the cat that was the subject of the picture.



We can try to solve this problem by training a CV model on a large dataset that has been marked up with bounding boxes for multiple classes by professional human annotators, and success will be measured by calculating the mean Average Precision of the model compared to the ground-truth annotations on a withheld test data set.

Datasets and Inputs

The main dataset for this project is going to be the Open Images V4 dataset[4], provided by Google Inc. under the CC by 4.0 License. Plus, I plan on using a ConvNet that was pretrained on a dataset like ImageNet[5] or COCO[6], such as Inception-v3[7] or YOLO-v3[8], for transfer learning. All of these are publicly and freely available.

This dataset is perfect for this task since it contains about 9 million images with a special subset of 1.74M images featuring 14.6M bounding boxes for 600 object classes. Not only is this the largest collection of images with object location annotations, but they were mostly drawn by professional annotators for the highest quality and the machine drawn boxes were all human verified. Here is a heirarchy diagram of all 600 boxable classes:
https://storage.googleapis.com/openimages/2018_04/bbox_labels_600_hierarchy_visualizer/circle.html

---

[3] *Koko with an eye by Nathaniel Watkins (License)*

[4] *Krasin I., Duerig T., Alldrin N., Ferrari V., Abu-El-Haija S., Kuznetsova A., Rom H., Uijlings J., Popov S., Kamali S., Malloci M., Pont-Tuset J., Veit A., Belongie S., Gomes V., Gupta A., Sun C., Chechik G., Cai D., Feng Z., Narayanan D., Murphy K.* OpenImages: A public dataset for large-scale multi-label and multi-class image classification, 2017. Available from https://storage.googleapis.com/openimages/web/index.html.

[5] Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. (* = equal contribution) **ImageNet Large Scale Visual Recognition Challenge**. *IJCV,* 2015. paper | bibtex | paper content on arxiv | attribute annotations

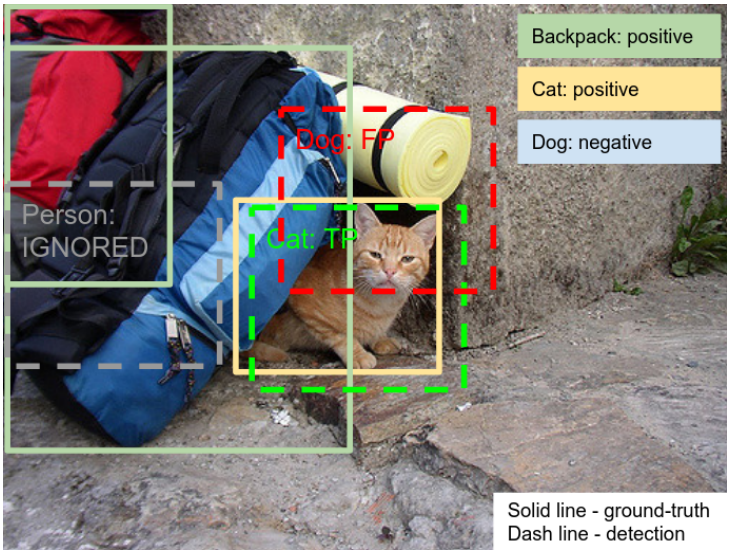
[6] arXiv:1405.0312 [cs.CV]

[7] arXiv:1512.00567 [cs.CV]

[8] arXiv:1804.02767 [cs.CV]

The Open Images V4 dataset will be obtained directly from Kaggle who is hosting the competition and used to train an ImageNet pre-trained ConvNet (obtained as a native Application from Keras). This competition was designed specifically to push the state-of-the-art in CV using this new dataset. I plan on using cloud computing to power through training on such a large dataset, either using Kaggle Kernels or in a Google Cloud TPU. So that the reviewer doesn't need to download 561GB of data, I'll start the notebook by exploring some samples and include a smaller subset of the dataset in my submission.

Solution Statement



This problem can be solved by generating predictions and bounding boxes that closely match what the human annotators have created. The labels are organized in a semantic hierarchy, so the model will need to output not just the object classes (such as 'Mango'), but also any relevant parent classes ('Fruit' and 'Food'). Since the label data in this set is non-exhaustive, False Positives that do not coincide with any labeled classes for that image will be ignored, except in the case of a detection that lines up with a negative label on an image, such as an image with a cat that was labeled specifically as not containing a dog. (see example to the left[9]). Furthermore, many images contain groups of a class (5+ instances that are touching and occluding each other), so a True Positive can be counted if there is at least 1 correct detection of that class with a bounding box that is at least halfway inside the 'Group of' box. I'll replicate this behavior with a separate validation dataset, and the competition will eventually test my model with a withheld test group.

Benchmark Model

To solve this problem, I plan on starting with a starter notebook and using it as the benchmark model, like the ResNet-50 model linked below, then building on it by tuning the parameters and model structure, before trying a more cutting edge algorithm to substantially improve on the mAP performance from that baseline model. This would provide a direct comparison to measure the amount of improvement that I was able to provide throughout the course of this project. The linked model is already able to achieve the solution, as laid out earlier, with a reasonable degree of accuracy, but seems to have a bug in outputting the coordinates to the Prediction String (of course I would ensure that the final benchmark model I settle on would be bug free). A random choice model would statistically result in a near 0 mAP score, (higher is better; see Evaluation Metrics below for more details). And since a 0 mAP would not be very helpful in determining relative progress, I'll compare my results to the the best results I can find when starting this journey.

https://www.kaggle.com/shivamb/objects-bounding-boxes-using-resnet50-imageai

[9] Image from Open Images Dataset V4 website:
https://storage.googleapis.com/openimages/web/object_detection_metric.html

Evaluation Metrics

The mean Average Precision (mAP) is simply the mean of the Average Precision (AP) for each of the 500 classes , which will be the final ranking metric for the competition and my means of quantifying progress. A simple AP is calculated as the current sum of every correct detection over the sum of all instances of that class encountered so far, but this competition is using the definition of AP set forward on page 10 of PASCAL VOC 2010[10], which is basically the area under the Precision/Recall curve. These values are only calculated based on classes tagged in images (either positive or negative), so if a class is not present in an image, False Positives get ignored. This method of evaluation seems to be the most practical way to measure the relevance of the results condensed down into one number, and thus allowing a fair comparison of the relative performance between models.

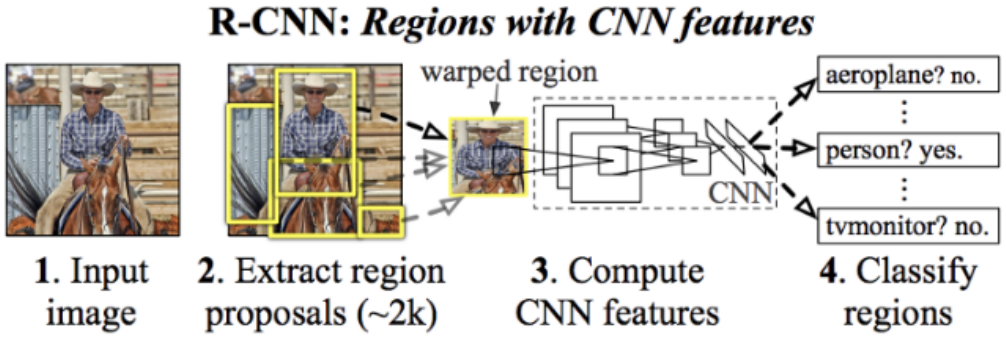
Project Design

My primary goal for this project will be to figure out how to implement at least one of the top algorithms representing the state-of-the-art in CV and Object Detection today, such as YOLO, SSD[11] or Faster R-CNN[12] and thus learn more about what makes it tick. Furthermore, if I have the spare time and computing power, I would like to tweak my implementation(s) for improved performance over a standard deployment of one of these algorithms.

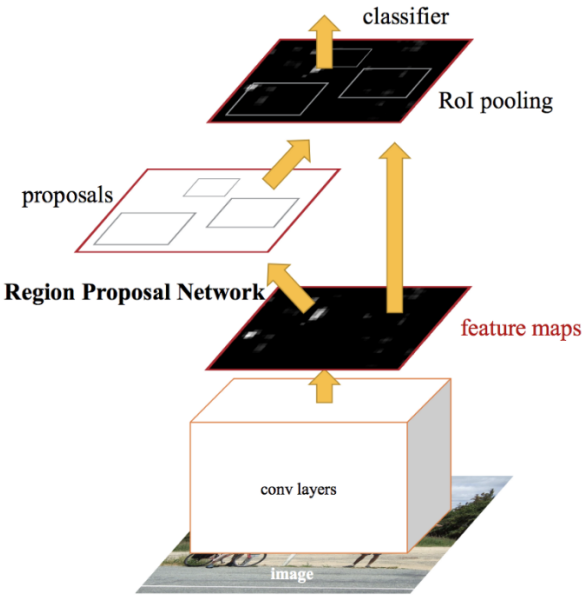
Based on my research so far, You Only Look Once (YOLO) version 3 and Faster R-CNN are looking the most promising for this task; both claim to achieve comparably on metrics such as mAP as other top algorithms, while training or processing images in a fraction of the time, sometimes orders of magnitude less time.

Standard Convolutional Neural Networks are not well suited to object detection tasks because the number of outputs can be variable from one image to another, thus requiring different output layer structures from one image to the next. One way to solve this would be to create many regions within an image that an object might be located within, then running the network on each region like a cropped picture. However, to be thorough, you'd have to try impractically many regions in each image, which would get out of hand very quickly.

R-CNN[13] (see illustration below to the right) came along to help with that by limiting the number to 2000 regions, then reducing it further by grouping similar regions together using a selective search algorithm. This allowed you to only run a cropped image through the network if it was predicted to have a specific object in it. This was a huge step forward making object detection feasible, but it still was extremely slow.



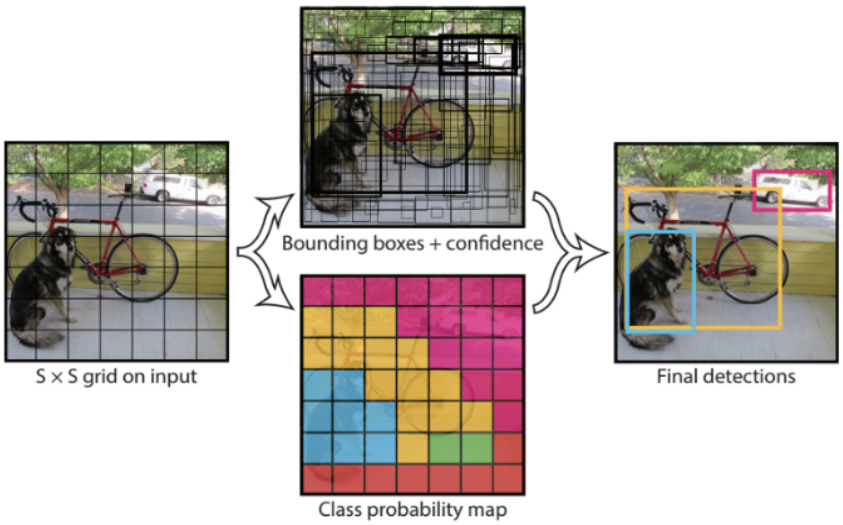
[10] http://host.robots.ox.ac.uk/pascal/VOC/voc2010/devkit_doc_08-May-2010.pdf
[11] arXiv:1512.02325 [cs.CV]
[12] arXiv:1506.01497 [cs.CV]
[13] arXiv:1311.2524 [cs.CV]



Next up, Fast R-CNN sped things up by first generating a feature map via the convolutional layers, then only running the region proposals generated from the feature map through the fully connected layers to predict the class and the specific bounding box coordinates, reducing the data run through the conv layers.

Finally, Faster R-CNN (see illustration to the left from their paper) took this a step further by replacing the slow selective search with a faster object detection algorithm. A feature map is still created from feeding the image through the convolutional layers, but then a different network is used to draw the region proposals. Then they are reshaped in a RoI Pooling layer before being inputted into the fully connected classifier layers for specific bounding box and class predictions.

Meanwhile, YOLO works by splitting the image into a grid of class probability while also drawing many random bounding boxes, then assigns probability to the boxes based on their alignment with the class map (see illustration to the right[14]). While this allows it to work extremely quickly, it struggles with detecting objects that are smaller than the class probability map cells and it might struggle with higher numbers of classes like the 500 in this challenge.



I'll begin by forking a Kernel that achieves modest results using an algorithm I'm already familiar with to use as my baseline, as mentioned in the earlier Benchmark Model section. Then, I will try tweaking the parameters and model from the baseline, seeing how much improvement I can pull out of it in a reasonable amount of time. Once I've reached an apparent practical limit, I'll move on and try a bleeding edge algorithm, like one of the ones previously mentioned. Depending on how training goes, I'd like to either tweak it further to eek out better results and/or try another state-of-the-art algorithm. Along the way, I'll be evaluating the results and watching out for likely pitfalls in my model's performance.

Additionally, I could try steps like Image Augmentation if my available computing power seems to be handling training in reasonable amounts of time, but perhaps it wouldn't be necessary or feasible with such a large dataset. I also reserve the right to modify my strategy as the competition progresses and other Kagglers share strategies that are working for them. While Kaggle 'competitions' are presented as such, they tend to be highly collaborative with people sharing what works and what doesn't work for them. While I have no illusions of placing near the top of the leaderboard in my first competition when up against teams of researchers that may be developing new, better algorithms specifically for this, I trust that my ability to research and collaborate with other Kagglers will lead me to a result I can be proud of.

[14] YOLO illustration and a better overview of the algorithms:
https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e