Machine Learning Engineer Nanodegree

Capstone Proposal

FC Su
April 17th, 2018

Domain Background

Amblyopia, also called lazy eye, is a disorder of sight due to the eye and brain not working well together It results in decreased vision in an eye that otherwise typically appears normal.Amblyopia begins by the age of five In adults, the disorder is estimated to affect 1–5% of the population While treatment improves vision, it does not typically restore it to normal in the affected eye.[1]

I am the one who suffer this disease with two eye, my vision is really bad. I have no glass, the only thing I can do to clearly see something is using magnifier for near thing and telescope for far thing. I can't easily see something in front of me. In class, I can't easily see the text on the blackboard. In work, as a firmware engineer, I can't easily see other colleague's code on their monitor. I really need something that can help me see something! I think this is the common requirement for low vision people or elder. A software magnifier can detect things, auto zoom in to things or manually zoom in or out.

Problem Statement

So, in this project, I would like to develop a magnifier application that can detect object with my iphone, trace object and zoom in to what I am interesting in. There are many magnifier app in App Store, such as SuperVision+ Magnifier [2]. Most of them emphasis on zoom in or out feature, But none of them can detect motion and predict what it is, This is what I really want. With this feature, if colleague would like to share their code with me and shake their mouse or hand on their monitor I can easily catch what they want me to focus on. In the presentation hall, I can easily focus on the point what speaker want to emphasis on through detecting the motion during the presentation.

Datasets and Inputs

Because I would like to detect object in life, such as people, cat, laptop, sessor, …. So I would like to use MS-COCO dataset. [3] Besides, since MS-COCO have no information about hand, which is very important in my application. I need to detect hand in image to get focus on what people want me to focus on. So I would like to include a hand dataset, EgoHands [4][5]. This dataset works well for several reasons. It contains high quality, pixel level annotations (>15000 ground truth labels) where hands are located across 4800 images. All images are captured from an egocentric view (Google glass) across 48 different environments (indoor, outdoor) and activities (playing cards, chess, jenga, solving puzzles etc). [6]

Solution Statement

Because my magnifier will have some features, below are my solutions.

1. Real time object detection
    a. Dataset
       MS-COCO and EgoHand
    b. Target
       life object and hand
    c. Algorithm:
       I would like to use YOLO (You Only Look Once) [7]. Because YOLO is good on real time object detection, so I would like to implement it on my magnifier.
2. Motion Detection
    a. Target
       cusor, light spot, or not detected object in feature 1
    b. Algorithm
       Background Subtraction (OpenCV) [8]
3. Zoom in/out
    a. Algorithm
       var videoZoomFactor: CGFloat { get set } (swift build-in function)

Benchmark Model

Because feature 2 and 3 are not the point of this application, I only use library to achieve. So I will not compare feature 2 and 3 with other benchmark.

For feature 1, since I will do real time object detection with YOLO, and Apple already provide a real time object detection model, which is MobileNet [9]. So I will compare with the evaluation metrics as below section descirbed.

Below is the Neural Network structure of YOLOv2

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Below is the Neural Network structure of MobileNet

Type / Stride	Filter Shape	Input Size
Conv / s2	3 × 3 × 3 × 32	224 × 224 × 3
Conv dw / s1	3 × 3 × 32 dw	112 × 112 × 32
Conv / s1	1 × 1 × 32 × 64	112 × 112 × 32
Conv dw / s2	3 × 3 × 64 dw	112 × 112 × 64
Conv / s1	1 × 1 × 64 × 128	56 × 56 × 64
Conv dw / s1	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 128	56 × 56 × 128
Conv dw / s2	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 256	28 × 28 × 128
Conv dw / s1	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 256	28 × 28 × 256
Conv dw / s2	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 512	14 × 14 × 256
5× Conv dw / s1	3 × 3 × 512 dw	14 × 14 × 512
5× Conv / s1	1 × 1 × 512 × 512	14 × 14 × 512
Conv dw / s2	3 × 3 × 512 dw	14 × 14 × 512
Conv / s1	1 × 1 × 512 × 1024	7 × 7 × 512
Conv dw / s2	3 × 3 × 1024 dw	7 × 7 × 1024
Conv / s1	1 × 1 × 1024 × 1024	7 × 7 × 1024
Avg Pool / s1	Pool 7 × 7	7 × 7 × 1024
FC / s1	1024 × 1000	1 × 1 × 1024
Softmax / s1	Classifier	1 × 1 × 1000

Evaluation Metrics

Real time object detection always use mAP and FPS to compare their performace. So, I will also use them to compare in the this project. Below are the mAP and FPS definition.

Precision : (Number of detected bounding box) / (Number of predicted bounding box)
Mean Average precision (mAP): average the APs over all classes

Frame Per Second (FPS)

Project Design

Below are my workflow for this project.

1. combine MS-COCO and EgoHands to my dataset
2. Train my dataset with YOLO and collect mAP
   -> Get sample implementation from github.
3. Train my dataset with MobileNet and collect mAP
   -> Use Keras build-in function.
4. Convert step2 and step3 model to Apple CoreML format
5. Create an iOS magnifier app and import the CoreML model from step4.
6. Make the app do real time object detection and collect FPS.

Reference

1. Amblyopia (https://en.wikipedia.org/wiki/Amblyopia)

2. SuperVision+ Magnifier
(https://itunes.apple.com/us/app/supervision-magnifier/id691435681?mt=8)
3. Microsoft Common Objects in Context (COCO) (http://cocodataset.org/#home)
4. EgoHands: A Dataset for Hands in Complex Egocentric Interactions
(http://vision.soic.indiana.edu/projects/egohands/)
5. Bambach, Sven, et al. "Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions." Proceedings of the IEEE International Conference on Computer Vision. 2015.
6. How to Build a Real-time Hand-Detector using Neural Networks (SSD) on Tensorflow
(https://towardsdatascience.com/how-to-build-a-real-time-hand-detector-using-neural-networks-ssd-on-tensorflow-d6bac0e4b2ce)
7. YOLO: Real-Time Object Detection (https://pjreddie.com/darknet/yolo/)
8. How to Use Background Subtraction Methods
(https://docs.opencv.org/3.1.0/d1/dc5/tutorial_background_subtraction.html)
9. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications
(https://arxiv.org/abs/1704.04861)