YOLO Net on iOS

Maneesh Apte, Simar Mangat, Priyanka Sekhar
(mapte05, smangat, psekhar)

Introduction

With the growing popularity of neural networks, object detection on images and subsequently videos has increasingly become a reality. However, fast object detection in real time with high accuracy still has large room for improvement. Our project aims to extend existing work to 1) have higher accuracy and 2) to be deployed to iOS via an iPhone app that uses the phone's video camera. Mobile detection has applications in education, autonomous vehicles, and consumer facial recognition.

Previous Approaches

Previous work has focused on optimum design on large desktop GPUs. However, the speed of the net becomes much more critical on mobile.

Approach and Evaluation

We are using GoogleNet [3] and SqueezeNet [1] to modify tiny-YOLO. We evaluate the speed (FPS) and accuracy (mAP) of our hybrid nets relative to the tiny-YOLO baseline. Our initial app targets language learning through object detection.

Architectures

We work primarily with the VOC 2012 dataset. This dataset has 20 classes and a tractable 11,000 images. The dataset allows for high precision and thus meaningful consumer use cases on mobile.

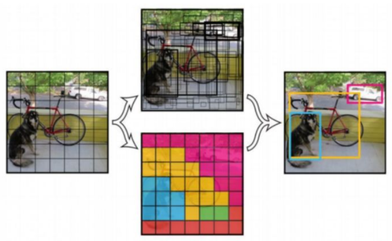


Figure 1: We rely heavily on the tiny YOLO model, a modification to YOLO that uses 9 conv/maxpool layers.

We modify the darkflow [2] YOLO framework to include either Fire layers - as implemented in SqueezeNet - or GoogleNet-inspired "inception" layers.

Figure 2: Fire Layer

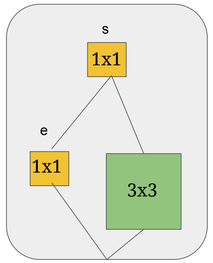
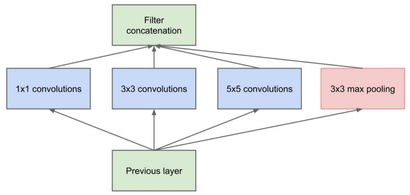
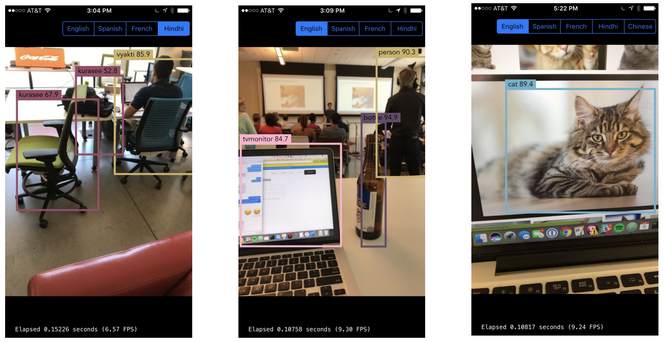


Figure 3: Inception Layer



Mobile

To implement the net on mobile, we made use of the Metal framework [4] and the Forge wrapper library [4] that abstracts some of the low level Metal code. This tensorflow-based framework allows real-time feedforward prediction on mobile devices. That said, one drawback to Metal is the lack of a built in batchnorm layer, which is important for both speed and accuracy in the darkflow implementation. To tackle this, we folded the trained batch norm weights into the preceding convolutional layers. The app in its current form does real time object detection of 20 classes in 5 different languages. Screenshots are shown below.



Discussion

Our current highest performing tiny-YOLO net runs at 8-10 FPS on the iPhone 7 (iOS 10). We calculated a mAP of 0.5 using equations (1) and (2). [5]

(1) $$AvgPrecision = \sum_{k=1...N} P(k)\Delta recall(k)$$

(2) $$MAP = \frac{\sum_{q=1}^{Q} AvgPrecision(q)}{Q}$$

While our hybrid nets did converge, they had low mAP scores, suggesting the need for more thorough hyperparameter tuning. The SqueezeNet inspired model trained at a comparable speed to tiny-YOLO.

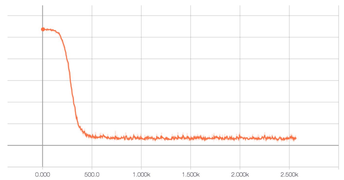


Figure 4: Fire Layer Net Loss

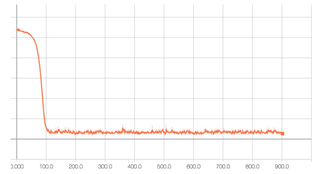


Figure 5: Google LayerNet Loss

Next Steps

Our preliminary results indicate that a SqueezeNet inspired model could have comparable, potentially faster, training and prediction times to the benchmark tiny-YOLO, while the GoogleNet inspiration appears to move more slowly but more accurately. The app we built indicates that real-time detection for mobile can be done in near real time. We plan to further tune hyperparameters and refine training on our two nets. We are also looking into applications for mobile facial recognition and its implications for mobile AR.

References

[1] SqueezeNet - https://arxiv.org/pdf/1602.07360.pdf
[2] Darkflow - https://github.com/thtrieu/darkflow
[3] GoogleNet - https://arxiv.org/pdf/1409.4842.pdf
[4] Metal - http://machinethink.net/blog/object-detection-with-yolo/
[5] mAP - http://cs229.stanford.edu/proj2016/report/BuhlerLambertVilim-CS229FinalProjectReport.pdf