Visionary: Voice Guide for Visually Handicapped People — CS3244 Group 26 Project

Jin Shuyuan*, Mou Ziyang*, Tian Xin*
Tian Xueyan*, Wang Tengda*, Zhao Tianze*
National University of Singapore
shuyuanjin@u.nus.edu(A0162475B), ziyang.mou@u.nus.edu(A0147984L), tian.xin@u.nus.edu(A0148036H),
tian.xueyan@u.nus.edu(A0177357R), tengda@u.nus.edu(A0177660X), zhao.tianze@u.nus.edu(A0147989B)

Abstract

This paper presents our idea of the app: *Visionary*, a voice guide app for visually handicapped people to assist them in avoiding approaching people on the street. With the help of camera, *Visionary* can capture real-time images, classify objects and make voice warning to inform the user that some people are approaching. Our app adopts a state-of-the-art object detection Machine Learning model called YOLOv3 whose architecture and algorithm are explained in YOLOv3 section. To determine other pedestrians moving direction and output audio, bounding boxes sizes are compared and CMU Flite Text-to-Speech Synthesis Engine is utilized. Additionally, some other plausible Machine Learning algorithms and their comparisons to YOLOv3 are shown in other Machine Learning Methods part. We believe *Visionary* will have an extensive impact on the lives of visually handicapped people and their family in Singapore.

Background

As of March 2017, the population of low-vision and blind residents registered to the Singapore Association of the Visually Handicapped is 3814 (sav 2017). The Singapore government has built tactile paving in all the subway stations and most of the walking paths for visually handicapped people. For those who would like to walk on the tactile paving, in-door and out-door walking guide system based on GPS can be easily built for them and the correct route can be feedbacked to the visually handicapped user to ensure they are heading in the correct direction. However, in this case, other pedestrians become one of the most critical uncertainty - they are not predictable. If a pedestrian does not notice the visually handicapped user, they may walk into each other which may cause severe consequences and is pretty dangerous.

We have implemented a system to detect approaching pedestrians with Machine Learning and translate the information to voice (Fig. 1). Our product will act as an add-on system on the guide system for visually handicapped users to offer real-time walking guide. With the help of our system, the blind users will be able to get more real-time conditions about the path ahead so that they can avoid approaching pedestrians. *Visionary* will help them walk faster and walk safer, improving their life quality.

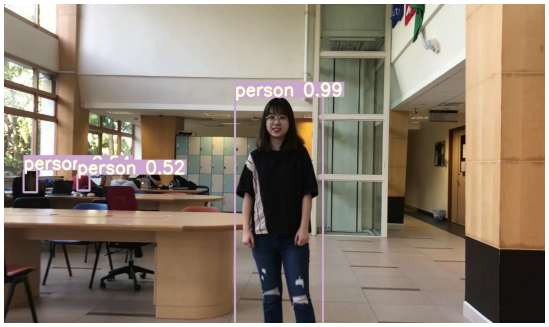

Figure 1: Our demonstration.

App Implementations

Visionary takes sequential RGB images or video as input and outputs voice reminders of any approaching pedestrians.

- The input accepts three formats: image series, videos and real-time webcam images.
- *Visionary* outputs "Approaching" voice when someone is approaching and remains silent otherwise.

Structure of *Visionary* is displayed in (Fig. 2) and the code can be found in our Github repository[1]. We choose YOLOv3 as our Machine Learning method. This algorithm can detect human objects within a short time period and achieve high accuracy, which is important for a real-time guide system.

Apart from YOLOv3, we design and implement an algorithm to detect objects moving direction. It compares the bounding box size in the current frame with the average size of the previous five frames. For each of the frame, we find the largest bounding box because we consider it as the nearest person to the user. If either width or height of the bounding box is larger than the average of previous 5

*All of us contributed equally

[1]https://github.com/CoderStellaJ/CS3244-ML-Project

frames, we assume the object is approaching. We design this algorithm as the camera may not always capture a full-size human. During our experiments, when the user is approaching another pedestrian, the area of the bounding box may decrease because the legs of the pedestrian are not within the camera's angle of view, whereas the width of the bounding box still increases. Thus, comparing box width or height is more reasonable.

After comparing different ways of transforming text to sound, we adopt CMU Flite Text-to-Speech Synthesis Engine to generate our sound files. Those sound files are played to notify the user when *Visionary* predicts someone is approaching.

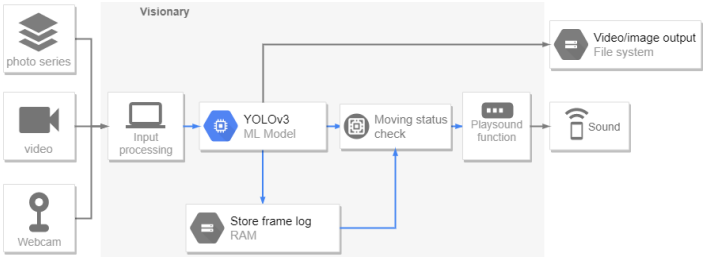

Figure 2: System architecture

App Features

- Camera based image capturing
- Object detection and classification
- Moving direction determination
- Voice Guide

App Algorithm Requirements

In order to achieve our goal to help blind people navigate around, our app algorithm needs to model human visual system. Human eyes can capture real-time scenes constantly for a long period of time and the visual system is fast and accurate in detecting objects in scenes, classifying them and estimating their moving directions. Equipped with this ability, humans are able to make proper decisions such as stopping or changing direction to avoid obstacles. However, for detection systems, there is always a trade-off between accuracy and speed. We want neither a highly accurate but over-complex slow model nor a fast but error-prone method. Thus, the chosen machine learning algorithm should satisfy the following requirements:

- Continuously take sequential images or videos as input and generate real-time outputs consisting of object detection and classification.
- Keep a good balance between speed and accuracy.

Based on the above requirements, we finally chose YOLOv3 (Redmon and Farhadi 2018) as the core machine learning part of our app. Compared with other Machine Learning detection systems, YOLOv3 can realize real-time detection and in the meanwhile keep relatively high Mean Average Precision (mAP). More details of qualitative and quantitative comparisons can be found in Other Methods and Experiments and Results section.

YOLOv3

You Only Look Once(YOLO) was first introduced by (Redmon et al. 2015). After that, YOLO has been improved over the years and the latest version is YOLOv3 (Redmon and Farhadi 2018). In this section, we will take a closer look at its technical details.

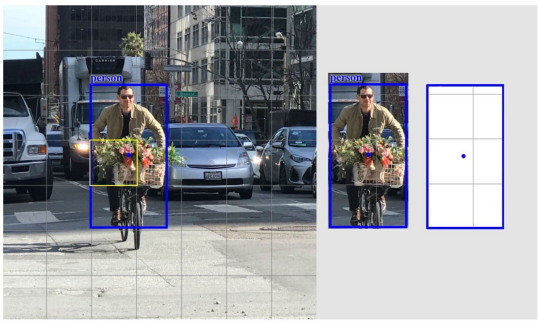

Figure 3: YOLOv3 divides an image into $N * N$ grid cells.

Grid Cell

YOLOv3 divides the image into $N * N$ grid cells. Each cell draws B bounding boxes at different scales and classifies object type inside each box (Fig. 3). Every bounding box consists of the following parameters:

- x, y, w and h, which are related to the coordinates of the box's center as well as the width and height of the box.
- The probability that the box contains an object $P(Object)$, which is called objectness.
- Class scores $P(Class_i|Object)$ for C class types.

The final prediction of YOLOv3 is encoded as a $N * N * (B * (4 + 1 + C))$ tensor.

Network Design

As shown in Fig. 4, a CNN with 53 layers is used to perform feature extraction from images. The features are then fed into a regression to make predictions.

Loss Function

The loss function of YOLOv3 consists of:

- Localization Loss
  The localization loss measures the errors in the location and size of the predicted boundary box.

$$loss_{loc} = \lambda_{loc} * (MSE(x, \hat{x}) + MSE(y, \hat{y}) + MSE(w, \hat{w}) + MSE(h, \hat{h})) \quad (1)$$

  where MSE is mean squared error

- Objectness Loss
  The objectness loss measures the error in predicting whether a bounding box contains objects.

$$loss_{obj} = \lambda_{obj} * BCE(P(Object), \hat{Object}) \quad (2)$$

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 4: YOLOv3 network for feature extraction.

where BCE is binary cross-entropy

- Classification Loss
  The classification loss reflects the accuracy of classifying the correct object type given the object in the box.

$$loss_{class} = \lambda_{class} * CE(P(Class_i|Object), \hat{Class_i}) \tag{3}$$

where CE is cross-entropy

The final multi-part loss for YOLOv3 neural network is:

$$loss_{final} = loss_{loc} + loss_{obj} + loss_{class} \tag{4}$$

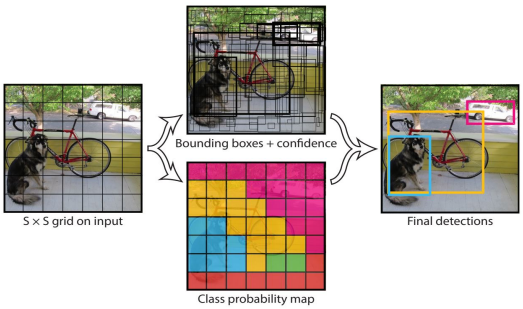

Figure 5: YOLOv3 system working process.

Inference

At this inference stage, final class-specific confidence for each bounding box is computed (Fig. 5):

$$conf_{class} = P(Class_i|Object) * P(Object) = P(Class_i) \tag{5}$$

To make sure that one object is bounded by exactly one box, YOLOv3 uses non-maximal suppression. Here is its implementation in (Redmon et al. 2016):

- Sort the predictions by the class-specific confidence scores.
- Start from the prediction with the highest score, ignore or remove any current prediction if there exists a previous prediction that has the same class label and $IoU^2 > 0.5$ with the current prediction.
- Repeat the previous step until all predictions are checked.

Interesting Behavior of YOLOv3

YOLOv3 is extremely fast. (Redmon and Farhadi 2018) designs the model as a unified single neural network whose performance can be optimized end-to-end directly instead of a complicated pipeline. During prediction, the bounding box coordinates and class probabilities come straight from image pixels in a single shot.

Previous region-proposal methods limit the classifier to some specific regions. Compared with them, YOLOv3 accesses to the whole image so that it demonstrates fewer background errors during prediction.

Most object detection systems assume that class labels are mutually exclusive. However, in reality, a "pedestrian" could also be a "child". In YOLOv3, logistic classifiers replace softmax function used in other systems so that all class scores $P(Class_i|Object)$ don't necessarily sum up to 1 and an object can be multi-labeled (Hui 2018).

YOLOv3 and Our App

Based on assessment of several existing object detecting models, we decided to use YOLOv3, as it met our expectations and fit the requirements of our proposed application.

YOLOv3 runs significantly faster than other detection methods and it can model human eyes quite well in terms of our app functionality. It helps *Visionary* render real-time object recognition and classification, which is extremely significant in assisting visually impaired people to observe the surroundings. YOLOv3 is strong in a detection metric of mAP at $IOU = 0.5$. According to (Russakovsky, Li, and Fei-Fei 2015), it is even difficult for even human beings to distinguish a bounding box with IOU of 0.3 from one with IOU of 0.5. Hence it is sufficient to use YOLOv3 in our app.

YOLOv3 reasons globally about the image when making predictions (Redmon and Farhadi 2018). Since we are detecting the person with the closest distance to the user, it is good to have a comprehensive overview of the surroundings. It is also easily extensible, as we can further predict objects other than persons in the same frame. Furthermore, YOLOv3 learns generalizable representations of objects, so it is less likely to break down when applied to new domains

[2]IOU is union intersection between ground truth and predicted bounding box

or rare objects (Redmon and Farhadi 2018). However, due to hardware limitations, we are not able to produce fast outputs as presented in their paper using our own laptops and phones but the model is theoretically competitive.

Improvements on Current Model

Since our app cares about only the human object closest to the user, there might be inconsistency in the assumption when YOLOv3 uses bounding box. For instance, a person stretching his arms may actually be more distant but still has larger bounding box than a nearer person standing straight. In order to cope with this issue, alternatives to bounding box method could be attempted. A new approach called Mask R-CNN was proposed by (He et al. 2017), which predicts an object mask in parallel with bounding box. Incorporating object mask in the current model would effectively reduce the probability of making wrong assumption, and hence improve application performance.

Other Machine Learning Methods

This section describes other possible methods and compares them with YOLOv3.

R-CNN

As shown in Fig. 6, R-CNN performs selective search[3] on the input images and extracts 2000 bottom-up region proposals. These 2000 candidates are fed into a large convolutional neural network(CNN) that outputs a 4096-dimensional feature vector. Then the extracted features are fed into an SVM[4] to classify the presence of the object within that region proposal. Post-processing is used to refine the bounding boxes and eliminate duplicate detections.

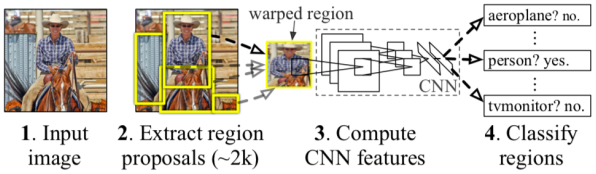

Figure 6: Architecture of R-CNN.

Although R-CNN achieves a mAP of 53.7% on PASCAL VOC 2010, the model needs to examine 2000 region proposals and spends around 47s on each test image. Therefore, R-CNN cannot be implemented in real time.

Fast R-CNN

According to Fig. 7, Fast R-CNN fits the entire image into a fully convolutional neural network to generate a feature map. It then performs selective search algorithm on the feature map to identify region proposals and uses RoI pooling[5] layer to reshape them into the required size. The network has two output vectors per RoI: namely softmax probabilities and per-class bounding-box regression offsets.

Empirically, although Fast R-CNN offers speed and accuracy improvement over R-CNN, it still falls short of real-time detection.

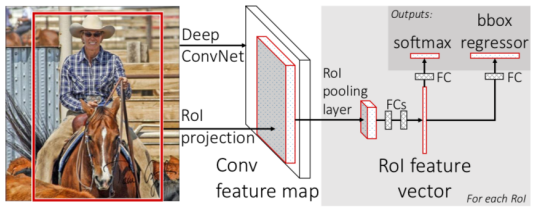

Figure 7: Work flow of Fast R-CNN. The architecture is trained end-to-end with a multi-task loss.

Faster R-CNN

Faster R-CNN employs a region proposal network to replace the selective search algorithm for generating region proposals (Ren et al. 2015).

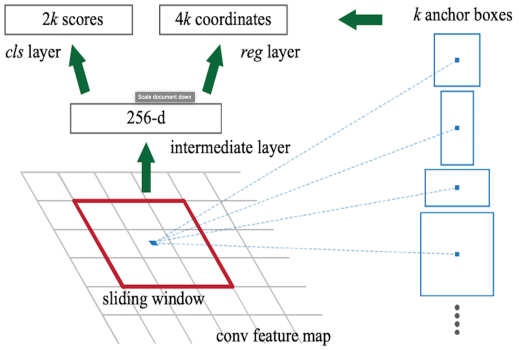

Figure 8: Region proposal network of Faster R-CNN.

The region proposal network is a convolutional neural network, which takes in feature map as an input. A 3×3 window with depth K slides through the feature map, outputting a vector with 256 features for each window. These features are fed into 2 fully-connected layers, box-regression layer and box-classification layer to compute the boundary box. Then, the output region proposals are fed into the RoI pooling layer in the Fast R-CNN algorithm (Fig. 8).

Faster R-CNN reduces the object detection time to 0.2 seconds, as the time-consuming step of selective search is replaced by region proposal network. Compared to YOLOv3, it has higher correctness but longer detection time.

[3]Selective Search is a region proposal algorithm that generates hierarchical grouping of similar regions in terms of color, texture, size and shape compatibility.

[4]Support Vector Machine is a supervised learning method that looks at data and sorts it into one of two categories

[5]A pooling layer which performs max pooling on inputs of non-uniform sizes and produces a small feature map of fixed size.

Deformable Parts Model

DPM can detect and localize objects with a mixture of the graphical model of deformable parts (Felzenszwalb et al. 2010). The model contains three important components. First, a root filter constructs a detection window that approximately covers the locations of the objects. Then, smaller parts of the object will be covered by multiple part filters. These part filters are learned at twice the resolution of the root filter. Finally, a spatial model is used to score the locations of part filters relative to the root.

Fig.9 illustrates how DPM matching process works in details. However, the performance of DPM in some experiments (Fig. 11) shows that the speed and accuracy are worse than YOLOv3.

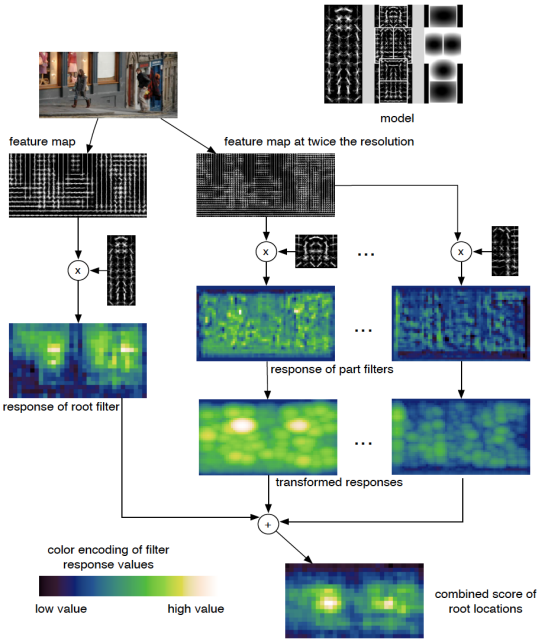

Figure 9: DPM maching process

RetinaNet

2017 introduced RetinaNet, a one-stage detector with a new-designed focal loss as loss function, ResNet+FPN as the backbone for feature extraction, plus two task-specific sub-networks for classification and bounding box regression.

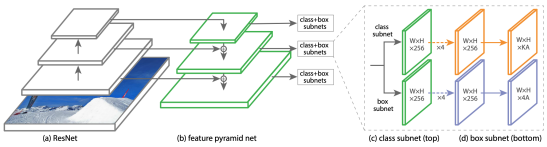

Figure 10: RetinaNet architecture

Focal loss is a loss function resembling cross entropy loss, but it is reshaped to down-weight easy example and focus training on hard negative samples. Feature Pyramid Network (FPN) is originally a two-stage detector providing both bottom-up and top-down pathways (Fig. 10). Bottom-up pathway extracts features, and top-down pathway combines features with layers in bottom-up pathway. Classification sub-net predicts the presence of objects and box regression sub-net computes the bounding box locations.

Compared to YOLOv3, RetinaNet has slightly higher prediction accuracy, but clearly longer detection time.

Experiments and Results

This section shows the results of experiments and quantitative comparisons with other methods.

After applying YOLO, R-CNN and DPM on the VOC 2007, Picasso, and People-Art Datasets, the result shows that YOLO has the best performance(Fig. 11) in AP[6] value which means it has the best accuracy among the 3 models.

	VOC 2007	Picasso		People-Art
	AP	AP	Best F_1	AP
YOLO	**59.2**	**53.3**	**0.590**	**45**
R-CNN	54.2	10.4	0.226	26
DPM	43.2	37.8	0.458	32

Figure 11: Quantitative results on the VOC 2007, Picasso, and People-Art Datasets.[8]

According to the result of applying Faster R-CNN, RetinaNet and YOLOv3 on MS CoCo Dataset, RetinaNet has the best accuracy (Fig. 12). However, in inference time aspect, YOLOv3 is much faster than RetinaNet (Fig. 13).

	backbone	AP	AP_{50}	AP_{75}
Two-stage methods				
Faster R-CNN+++	ResNet-101-C4	34.9	55.7	37.4
Faster R-CNN w FPN	ResNet-101-FPN	36.2	59.1	39.0
Faster R-CNN by G-RMI	Inception-ResNet-v2	34.7	55.5	36.7
Faster R-CNN w TDM	Inception-ResNet-v2-TDM	36.8	57.7	39.2
One-stage methods				
RetinaNet	ResNet-101-FPN	39.1	59.1	42.3
RetinaNet	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4

Figure 12: Quantitative results on MS CoCo.

Method	mAP	time
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

Figure 13: Inference time on MS CoCo.

[6] AP is average precision

[8] F1 score is a measure of a test's accuracy

Real-world Dataset

Since our app mainly focuses on human object detection, we are keen to find datasets that involve moving person objects and use them to train and test our algorithm specifically on top of the pre-trained model. Followed are some relevant real-world datasets that can be useful:

- Zero-Occlusion-Object-Tracking-Data-Set[9]
- Scramble Crosswalks[10]
- Help Blind Community to walk[11]
- Run or Walk[12]

Future Work

In the future, *Visionary* can be further developed to provide more reliable, accurate and complete services.

The framework should be further optimized for phone and achieve faster real-time output. Now, due to hardware constraints of the laptops, our demonstration still lags behind real-time performance. But a good existing example is app *iDetection* available on Apple App Store. This app simply builds YOLOv3 on iPhone and it works with low latency. Ideally, *Visionary* is expected to be developed to perform real-time tasks on phone.

Currently, *Visionary* outputs voice when bounding box size increases. But when a pedestrian is very far from the user, *Visionary* should ignore the person even if he is approaching. To resolve this problem and offer a more complete solution, camera needs to take both RGB and depth information. With depth images, the distance from the user to the person can be retrieved and *Visionary* won't warn the user when obstacle is far away. On Apple Store, a tool *Depth Cam - Depth Editor* capturing depth images is found. Thus, we will integrate its functionality to *Visionary*.

Team work

Work Allocation

Team Member	Role
Jin Shuyuan	Researched on YOLOv3 and set up project code
Mou Ziyang	Researched on other models and made comparison with YOLOv3
Tian Xin	Researched on DPM model and compare their performance
Tian Xueyan	Researched on YOLOv3 and datasets
Wang Tengda	Researched on other models and made comparisons
Zhao Tianze	Application developer

Table 1: Team members' roles.

Reflections

- **Jin Shuyuan** I feel lucky to work in this team because everyone is trying to contribute. And I found that Deep Learning takes rigorous experiments and experiences.
- **Mou Ziyang** I find it was exciting to find out how fast machine learning models are updated.
- **Tian Xin** Using machine learning models to solve real world problems is meaningful.
- **Tian Xueyan** I was driven to look into details of algorithms and I found it rewarding that I learned how to evaluate models and how to associate models with real-world problems.
- **Wang Tengda** A fruitful experience researching on various machine learning models, making comparisons and applying them to solve real-world problems.
- **Zhao Tianze** Compared with other algorithms, machine learning usually produce high accuracy result and could be optimized to an acceptable short time complexity.

[9]https://github.com/csbuja/Zero-Occlusion-Object-Tracking-Data-Set

[10]http://data-lahub.opendata.arcgis.com/datasets/ladot::scramble-crosswalks

[11]https://www.kaggle.com/hamzafar/look4me

[12]https://www.kaggle.com/vmalyi/run-or-walk

References

Felzenszwalb, P.; Girshick, R.; McAllester, D.; and Ramanan, D. 2010. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32(9):16271645.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. B. 2017. Mask R-CNN. *CoRR* abs/1703.06870.

Hui, J. 2018. Real-time object detection with yolo, yolov2 and now yolov3.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, 2980–2988.

Redmon, J., and Farhadi, A. 2018. Yolov3: An incremental improvement. *CoRR* abs/1804.02767.

Redmon, J.; Divvala, S. K.; Girshick, R. B.; and Farhadi, A. 2015. You only look once: Unified, real-time object detection. *CoRR* abs/1506.02640.

Redmon, J.; Divvala, S.; Girshick, R.; and Farhadi, A. 2016. You only look once: Unified, real-time object detection. CVPR 2016 Talk Slides.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, 91–99.

Russakovsky, O.; Li, L.; and Fei-Fei, L. 2015. Best of both worlds: Human-machine collaboration for object annotation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2121–2131. United States: IEEE Computer Society.

2017. Who we serve.