# **Introduction**

## **1.1. Objective**

This project is trying to use YOLO to detect several different targets among 15000 images including biker, car, pedestrian, traffic Light, traffic Light-Green, traffic Light-Green Left, traffic Light-Red, traffic Light-Red Left, traffic Light-Yellow, traffic Light-Yellow Left, and truck in images.

Those objects will be classified and localized by the YOLO algorithm with a probability to which a certain object belongs and its corresponding bounding box. YOLOv3 will be used in this project because it is fast, accurate, robust, and fully developed.

15000 images with 512\*512 pixels from Udacity self-driving car dataset in roboflow will be used for a pre-trained model which is trained on big datasets, such as COCO dataset, VOC dataset. The pre-trained model will be used as an initialization of the training.

## **1.2. Significance**

Autonomous driving will be the leading way of transportation in the future, because it lowers the risk of making mistakes, increases stability and safety, and provides passive protection beyond human cognition. In the meanwhile, an automatic system with dedicatedly constructed planning strategies improves the efficiency of road planning, decision making, and so on and so forth, liberating drivers while increasing comfortability. The current autonomous driving systems rely highly on visual information to make decisions, e.g., detecting multiple objects simultaneously, estimating the depth map of an image, semantically segmenting multiple instances. Another important aspect of autonomous driving is to make decisions in real-time, as the lower the latency is, the safer the system will be. Therefore, recognizing multiple targets in real-time is highly demanded in the decision-making of self-driving systems.

This leads us to the YOLO (you only look once) algorithm which has the capability of detecting multiple objects in real-time with reasonable performance. The Original YOLO paper demonstrates its ability on performing object detection on videos with 45 FPS, and even 150 FPS with a tiny version. At that time, the other state-of-the-art methods, e.g., Faster R-CNN is only able to process images at 7 FPS. The essential idea of YOLO is that it provides a unified neural network to train and do inference instead of a multi-stage system inferring on a neural network several times. The motivation to do so is that feeding data to a (convolutional) neural network is an extremely time-consuming process even on GPU. The R-CNN-based methods are either running a neural network multiple times during inference, or running multiple neural networks to do regional proposals and detection, separately. The inherent computation of selecting regional proposals is the key factor that R-CNN based method can not run inference in a fast matter. The millisecond-level inference enables YOLO to be way better than human cognitions whose reaction time is always at half-second.

Furthermore, not only is YOLO fast, it is also accurate in terms of the mean average precision (mAP). The original paper demonstrates its mAP is doubled compared to the other state-of-the-art work. For autonomous driving, precision is of equal importance as real-time compatibility. The higher accuracy the algorithm is able to achieve, the less risk the driver will face. For instance, it is disastrous for an autonomous driving system to fail to detect a pedestrian, a car in front of a driver. YOLO makes big progress in real-time detection. It is the most popular and reliable way to detect multiple targets in real-time based on the image information obtained from the camera, that is also the reason why we choose this system as a main approach to the detection in the project. A combination of YOLO, Lidar, Radar, and HD(high-definition map) leads to a reliable, safe auto-driving system that will bring huge pleasure and convenience for humen.

In this project, the main methodology behind the YOLO will be reviewed in detail. In the meanwhile, some previous and current state-of-the-art work will also be reviewed in a brief manner, which may lead us to the motivation of YOLO. Finally, the experiments on a custom dataset were performed to demonstrate the performance of the YOLO algorithm in terms of mean average precision and visual inspection.

# **2. Methodology**

## **2.1. Current State of Research**

Object detection is one of the most significant and hottest high-level vision tasks which empowers a variety of applications in autonomous driving, surveillance, machine inspection, and etc. The fundamental problem of object detection is to classify potential objects within an image while localizing them simultaneously. The former task categorizes the objects within an image or a patch of an image, if patch-based object detection methods were considered. In general, classification is not an overwhelmingly intimating task, the convolutional neural network (CNN) revolutionizes the accuracy of object classification year by year, e.g., ImageNet[7], AlexNet[7], ResNet[8], etc. On the contrary, localization is comparatively harder to achieve even in a fully supervised manner. There are three main reasons that contribute to that: 1) the deformation of a variety of objects, 2) the (partial) occlusion of objects under a variety of physical condition alterations; 3) the appearance of the same objects in different scales.

Even before the emergence of convolutional neural networks, there were many state-of-the-art methods that always used hand-crafted features (e.g., spatial representation, local statistics) to represent an object. The Haar-based rectangular features were firstly used to detect faces, which characterizes the representation of the face by finding that the intensities of the chuck are different from those of the other regions, and areas between eyes are darker than other areas. By taking advantage of that observation, Viola et al [17] proposed using the gradient of the intensity in neighboring regions along with a cascaded AdaBoost classifier to detect the face. Later on, some features based on local statistics are used to detect people or some other objects. Dalal & Triggs [18] proposed using histograms of oriented gradients (HoG) features for human detection based on the local statistics of a human object. Felzenszwalb et al[19]., proposed using the deformable representation of the object as the feature to detect a variety of objects not stick to a certain one. Their hypothesis is that an object always has a hierarchical representation, e.g, the face contains eyes, mouth, nose which each of them is also a single object which can be identified.

    However, the hand-crafted features sometimes are biased in some cases, especially when an occlusion exists. Ideally, features ought to be learned from the data itself. Therefore, we only focus on CNN-based methods in this paper. But notice that Transformer[9] and Mixer[10] are also other powerful architectures other than CNN, which imposes different mechanisms to represent spatial features.

At the very early stage of the research in object detection, people always constructed a two-stage system: the first is to train a classifier based on sub-patches of images where a certain object is labeled; the second is to perform inference with the trained classifier at different scale (from coarse to fine) and aggregate the results at different scales as the final localization, because objects are always at different scales. One of the most widely-used methods is sliding windows at different scales, which sweeps overlapping patches of an image with a set of scaled windows and performs inference at each window. This method is computationally expensive in reality especially in the cases where the classifier is (convolutional) neural network. Another issue of this method is the ill-posedness of precise localization on deformable objects. The Overfeat method proposed by Sermanet et al[4] mitigates the inefficiency in the computation of the sliding window by applying the sliding window directly over the feature map of the CNN instead of over the input images. The key idea is that convolution can preserve spatial information. Performing a sliding window over the input image and feeding the CNN is equivalent to feeding the whole image to the CNN and performing a sliding window over the feature map which can be done by another convolutional layer. This method, however, still needs to perform coarse-to-fine inference which means running the network several times. Girshick et al.[11], mitigate this issue by proposing the Regional proposal method named R-CNN. The essential idea is to reduce the number of potential patches which contain objects by performing a preliminary segmentation over the raw image. By doing so, only the patches (~2k) potentially contain the objects that will be feeding to the neural network to extract the feature and then train a support vector machine (SVM) to classify the image based on the extracted features. Even though this method reduces the potential patches containing objects to the network, it runs a network for each patch which is time-consuming. The fast R-CNN[12] and faster R-CNN[13] aim to mitigate this issue by changing the way to do regional proposals. The fast R-CNN feeds the whole image in a CNN and performs regional proposals over the feature map meaning only one network is running for each image. The faster R-CNN proposed a regional proposal network (RPN) to do the regional proposal which essentially takes advantage of the fact that the extracted feature used for regional classification can also be used to do a regional proposal.

YOLO (You Only Look Once) is a powerful new technology in computer vision. It only applied one single neural network to the full image which improved the speed and brought the objects-detection to real-time. As its name suggests, it can identify multiple specific objects in videos, live feeds, or images by only “looking” at the images at once.

## **2.2. The development of YOLO**

YOLOv1 is a powerful objective detection method that was proposed by R. Joseph[1] in 2015. Different with deformable parts models (DPM) using sliding windows slides the entire image, and R-CNN using region proposal method to generate all potential bounding boxes and classify on the objects. YOLOv1 only includes a simple convolutional network that simultaneously predicts multiple bounding boxes and their class probabilities[1]. That is also the reason why it is fast and can be used as real-time objective detection. The main disadvantage of YOLOv1 is that its fast identify speed leads to it underperforming other detection methods in accuracy.

YOLOv1 used all information of the entire image and all objects. Each image is divided into S\*S grid, the grid which includes the center of one objective will be used to do the detection. There are many bounding boxes and their confidence scores for one grid. The confidence is . IOU is the intersection over union, which is a method to measure the accuracy of detections. The equation of IOU is: . The confidence score will be the IOU between the predicted box and the ground truth box[1], which means the area of overlap divided by the area of union between the predicted box and the ground truth box. There are 5 parameters to help to locate the object,  and the confidence of each object.  is the center of the bounding box,  is the width and is the height of the bounding box.

## **2.3. Hypothesis & Questions to Answer**

The accuracy used to be a weakness of YOLO, especially for small objects that appear in groups, such as flocks of birds[5]. However, it can be a trade-off between the speed and accuracy. In[3], it said the trade-off between speed and accuracy can be simply changed by the size of the model. The way that making the model both high speed and high accuracy will be one of questions that need to be answered in the further research. In addition, the comparison and development for YOLOv1 to YOLOv5 will be summarized. The approaches and attempts of improving YOLO is another question that needs to be answered in the future work.

## **2.4. Data Collection**

## **2.5. Model Architecture**

Diagram

Description automatically generated

## **2.6. Training Loss**

The unified loss is optimized by a specially constructed squared error which includes: the confidence score loss, the bounding box coordinate predictions loss, and classification loss. The equation of unified loss is show below:

## 

## **2.7. Evaluation Metric**

Conventionally, evaluation of predicted bounding boxes revolves around the intersection over union (IoU) metric. The IoU of a predicted bounding box $A$ and the ground truth (target) boxes $B$ is the area of overlap that divides the area of union.

Generally, we threshold the IoU score with a constant threshold to keep those bounding boxes with the highest probability meaning a predicted bounding box is considered correct at a certain threshold if its IOU is greater than the threshold. But for each bounding box, the IoU threshold may vary, which is in the range [0.5, 0.75] in our task. Therefore, we sweep over a  range of  IoU thresholds and for each of them, which is calculated at threshold values in the range [0.5, 0.75] with a step size of 0.05,  and compute their average.  The set of predictions for a single image is scored by the average of the precision values for each threshold:

**Text

Description automatically generated with medium confidence**

# **3. Results and Discussion**

YOLOv3 will be used to detect multiple objects in this project. YOLOv3 is a fully developed, fast, and accurate real-time objects detection system. In mAP(mean average precision) measured at 0.5 IOU, YOLOv3 is on par with Focal Loss but about 4x faster[6]. Which inspired by ResNet and FPN (Feature-Pyramid Network) architectures, YOLO-V3 uses a feature extractor called Darknet-53 (it has 52 convolutions) that contains skip connections (like ResNet) and 3 prediction heads (like FPN) — each processing the image at a different spatial compression[3].

Since the training of YOLO is computationally expensive, a pre-trained model which is trained in a big dataset with multiple targets will be found in the first step. The online sources of the YOLOv3 system will be used. Then,  80% of the 15000 images will be used to train the model, 10% will be used in the validation test. Mean average precision(mAP) will be used as the evaluation criterion. After that, I will try to use some different hyperparameters to train the model several times until I get a good mean average precision value. Finally, the last 10% of the dataset will be used to test the YOLO model. The output of mean average precision evaluates the accuracy of the model. From now to Nov 12, I will learn the way to achieve YOLOv3, its background knowledge, and the development from YOLOv1 to YOLOv5. Then, it will take around 2 weeks to set up, debug, train, and test the model. Finally, one week will be left before the deadline to work on constructing the report.

This project is a practice of using the YOLO system to do multiple objective detections. There is not much innovation in this work. Since I and my advisor work on small datasets and are still in the step of developing our own algorithm, there is not a very good application by using machine learning methods for my own research till now. However, I am sure we will use machine learning to improve our algorithm in the future. The reason why I chose YOLO as the topic of my final project is that I used to work for a technology company which is focused on autonomous driving during the summer after I got my master degree. I had no idea about machine learning and worked in a quality engineering position at that time, and felt very curious how they dealt with it.

# **4. Announcement**

This report concludes knowledge from published papers. It is only used to learn and application practice of YOLO, and will not be used for publishing.

# **References**

[1] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

[2] <https://pjreddie.com/darknet/yolo/>

[3] <https://towardsdatascience.com/yolo-v3-explained-ff5b850390f>.

[4] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229.

[5] <https://medium.com/@anand_sonawane/yolo3-a-huge-improvement-2bc4e6fc44c5>

[6] Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

[7] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. International journal of computer vision, 115(3), 211-252.

[8] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[9] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

[10] Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., ... & Dosovitskiy, A. (2021). Mlp-mixer: An all-mlp architecture for vision. arXiv preprint arXiv:2105.01601.

[11] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587).

[12] Girshick, R. (2015). Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).

[13] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 28, 91-99.

[14] <https://blog.paperspace.com/how-to-implement-a-yolo-object-detector-in-pytorch/>

[15] <https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52>

[16] <https://sandipanweb.wordpress.com/2018/03/11/autonomous-driving-car-detection-with-yolo-in-python>

[17] Rapid Object Detection using a Boosted Cascade of Simple Features

[18] histograms of oriented gradients for human detection

[19] Object Detection with Discriminatively Trained Part Based Models