Machine Learning Final Project Report: Understanding and Extending YOLOX

YUZHOU QIN, SUSTech
SICHENG ZHOU, SUSTech

ACM Reference Format:
Yuzhou Qin and Sicheng Zhou. 2024. Machine Learning Final Project Report: Understanding and Extending YOLOX. 1, 1 (January 2024), 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 RESEARCH PROBLEM

1.1 Introduction to Object Detection and Tracking

Object detection and tracking are fundamental tasks in computer vision that play a crucial role in enabling machines to interpret and interact with the visual world. These tasks involve the identification and monitoring of objects within an image or a sequence of images, providing essential information for a myriad of applications ranging from autonomous vehicles and surveillance systems to augmented reality and human-computer interaction.

Object detection refers to the process of locating and classifying objects within an image, often bounding them with a rectangular box and assigning a corresponding label. Complementary to object detection, object tracking involves the continuous monitoring of identified objects across multiple frames or in a video stream. The objective is to maintain the identity of objects as they move and undergo changes in appearance, scale, or orientation.

The challenges in object detection and tracking are diverse, encompassing issues such as handling occlusions, variations in lighting conditions, and real-time processing requirements. As computer vision technology advances, researchers continually seek innovative approaches to improve the accuracy, efficiency, and robustness of these tasks.

1.2 Introduction to YOLOX

You Only Look Once X (YOLOX) stands as a significant milestone in the evolution of object detection algorithms within the field of computer vision.

Traditional object detection methods often face trade-offs between accuracy and speed, limiting their applicability in real-time scenarios. YOLOX adopts an "extreme" approach by focusing on a single detection level, streamlining the detection process for efficiency without compromising precision. This unique design allows YOLOX to achieve impressive real-time processing capabilities, making it well-suited for applications such as video analysis, autonomous vehicles, and interactive systems.

One of the distinguishing features of YOLOX is its anchor-free design, which eliminates the need for predefined anchor boxes. This design choice contributes to its adaptability and robustness across various object scales and aspect ratios. Additionally, YOLOX introduces the concept of a decoupled head, separating object classification and bounding box regression, facilitating more flexible model architecture customization.

YOLOX has gained attention for its real-time processing capabilities and competitive accuracy, making it a promising candidate for various applications.

2 RESEARCH GOALS AND OBJECTIVES

The overarching goal of this project is to comprehensively understand and extend the capabilities of the YOLOX object detection and tracking framework.

In this project, we will conduct a literature review on object detection and tracking methods, emphasizing the evolution of YOLOX and its comparative advantages over existing tgitechniques. We will thoroughly study the research paper and codebase of YOLOX to gain a comprehensive understanding of its architecture, design philosophy, and implementation details. Then we will extend the applicability of YOLOX by training and testing the model on a different dataset, evaluating the model's generalization capabilities across diverse datasets.

3 RELATED WORK

3.1 Object Detection

Object detection is a fundamental task in computer vision, playing a crucial role in various applications such as autonomous driving, surveillance, and human-computer interaction. Over the years, numerous methods have been proposed to address the challenges of accurately localizing and classifying objects within an image.

Classic approaches such as R-CNN (Region-based Convolutional Neural Network) and its variants, including Fast R-CNN and Faster R-CNN, laid the foundation for modern object detection techniques. These methods introduced the concept of region proposal networks and demonstrated significant improvements in both accuracy and efficiency.

The evolution of object detection continued with the advent of single-shot detectors (SSD) and You Only Look Once (YOLO) series. YOLO, in particular, introduced a real-time object detection paradigm by dividing the image into a grid and predicting bounding boxes and class probabilities directly. The trade-off between speed and accuracy became a central theme, leading to the development of YOLO variants like YOLOv2, YOLOv3, and the latest YOLOv4.

Authors' addresses: Yuzhou Qin, SUSTech, qinyz2021@mail.sustech.edu.cn; Sicheng Zhou, SUSTech, zhousc2021@mail.sustech.edu.cn.

XXXX-XXXX/2024/1-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

3.2 Object Tracking

Object tracking extends the capabilities of object detection by enabling the continuous monitoring of objects across consecutive frames. Various tracking algorithms have been proposed to address the challenges of occlusion, scale variations, and appearance changes.

Traditional tracking methods include correlation filters, such as MOSSE (Minimum Output Sum of Squared Error), and Kalman filters, which maintain a dynamic state estimate for each object. More recently, deep learning-based trackers, such as GOTURN (Generic Object Tracking Using Regression Networks) and MDNet (Multi-Domain Network), have demonstrated superior performance in handling complex scenarios.

4 RESEARCH DESIGN AND METHODS

We download the code from Megvii-BaseDetection/YOLOX on github [3], a popular YOLOX open source repository which already have 8.8k stars. After installing the dependent packages following the instruction, we can either download the pretrained model and run the demo, or do the reproduction.

4.1 Inference Demo

This repository provides several pre-trained models of different sizes which can be directly used to do inference. Here we chose YOLOX-s[4], a model with the least number of parameters and the fastest inference speed, to conduct tests on other datasets.

To do inference, we can run tools/demo.py and use -n or -f to specify the detector's configuration. YOLOX-s has its' default configuration which can be specified by the -n yolox-s command, while other reproducing works may need self-written configuration files.

4.2 Reproduce on COCO and VOC Dataset

The repository provides an API, allowing users to change hyper-parameters as well as train on custom datasets. The user should instantiate a Data class describing the dataset, an Evaluator class to estimate model performance, and an Exp class to control all the hyper-parameters.

The Data object should implement three functions, __getitem__, pull_item, and load_anno. __getitem__ method returns image and label according to the given index. pull_item and load_anno methods are used in Mosiac and MixUp augmentations.

The Exp class extends the yolox.exp class. In the __init__ method, the user can specify the model hyper-parameters, including the number of classes, the network depth and width, and the data augmentation probability. After that, it should instantiate the dataset object and the evaluator object. A detailed description of the evaluator is given in the next subsection.

4.3 Evaluation

The user can design their estimation methods by extending the evaluator class.

It first initializes lists to store image IDs and prediction data, then iterates over the dataset (self.dataloader), performing inference on batches of images. During this process, it records inference time and, if applicable, non-max suppression (NMS) time.

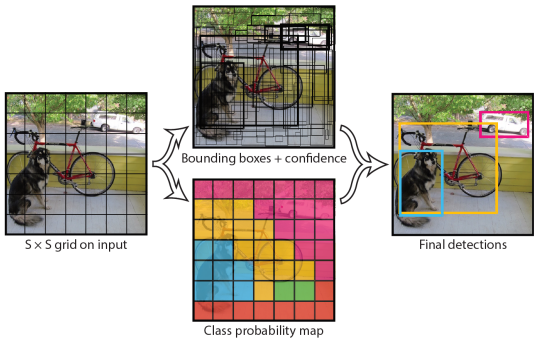

Fig. 1. YOLO Detection Process

After inference, the outputs are processed by the postprocess function, which applies confidence thresholding, filtering out detections with confidence scores below a certain threshold, and NMS, keeping the detection with the highest confidence score and removing all other detections that have a high overlap (as measured by the Intersection over Union, or IoU) with it.

In the end, the evaluator compares the new inference result with the ground truth annotations and gives out per-class AP and AR tables.

4.4 Self Gathered Dataset

In this project, we collect dataset in our campus and do inference on using the provided pretrained yolox_s model. We found that the model has some shortcomings in recognizing road signs. The COCO dataset only has the stop sign category related to road signs, so the model is not able to recognize images of parking lots, speed limit signs, etc. As a result, we used the Labelme[5], a popular annotation tool, to label the dataset we collected and retrained the model with this part of the data.

5 RESULTS

5.1 Literature Review

This section is the literation review we conducted on YOLOX[2]. It should be noted that YOLOX is obtained by improving YOLOv3 as the baseline, therefore before take a deeper look into YOLOX, we need to investigate the methods used in YOLOv3[7].

5.1.1 Overview. YOLO's CNN splits the input image into grids, and each grid predicts the bounding box and category. The overall process is shown in the figure 1.

In bounding box prediction, each grid should not only predict a bounding box position, but a corresponding confidence score as well. The bounding box is denoted by (x, y, w, h). The accuracy of the bounding box can be characterized by the intersection over union (IOU) of the predicted box and the ground truth. In classification, each grid has to predict the probability of which class the bounding box belongs to.

5.1.2 Data Augmentation. YOLOX uses two data augmentation methods: Mosaic and MixUp. Mosaic is to take 4 images and combine them into a single image, which is done by resizing each of the four

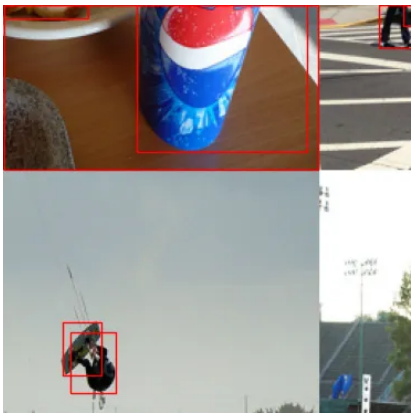

Fig. 2. Final Mosaic Image

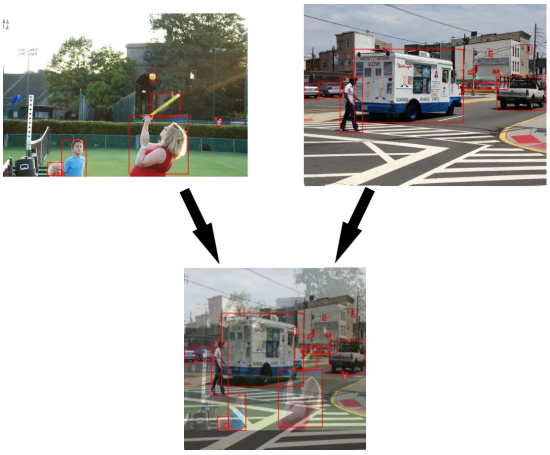

Fig. 3. MixUp

images, stitching them together, and then taking a random cutout of the stitched images. MixUp averages two images together based on a weighting parameter λ:

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j, \quad \text{where } x_i, x_j \text{ are raw input vectors} \tag{1}$$

$$\tilde{y} = \lambda y_i + (1-\lambda)y_j, \quad \text{where } y_i, y_j \text{ are one-hot label} \tag{2}$$

Figure 2 and 3 are examples of those two augmentation methods. But the MixUp and Mosaic methods are closed for the last 15 epochs in YOLOX model, probably due to the fear of excessive data augmentation. Additionally, after using strong data augmentation, it is found that ImageNet pre-training is no more beneficial, so they train all the models from scratch.

5.1.3 From Sliding Window to Anchor-free. Sliding window is the most primitive object detection method. It means that given a fixed size window, it slides step by step from left to right and from top to bottom according to the set pace, and input each window into a convolutional neural network for prediction and classification. This method has two disadvantages. First, the window size is fixed, so it is not suitable for objects with large deformation. Second, there are many windows and large amount of computation.

Region proposal is the core idea in R-CNN family. RPN(Regional Proposal Network) is not responsible for image classification, it is only responsible for selecting candidate regions in the image that may belong to one class of the data set, and then the candidate regions generated by RPN are input into the classification network for final classification. However, this method still has the problem that a window can only detect a single object and cannot solve the multi-size problem.

The Anchor Box is defined by the aspect ratio of the border and the area of the border. According to the Anchor Box, a series of bounding boxes can be generated at any position of the image. YOLOv3 uses K-means clustering to learn different anchors from the training set. There are also some problems with anchor-based methods. Firstly, anchors are obtained by performing cluster analysis before training, which are domain-specific and less generic. Secondly, the anchor mechanism increases the complexity of the detection head, as well as the number of predictions per image.

YOLOX adopts the center-based anchor-free method, which uses the center point or the center target region to define the positive sample, and then predicts its distance to the four edges of the target. It assigns the center position of each object as a positive sample and predefines a scale range to specify the FPN level of each object.

5.1.4 From Coupled Head to Decoupled Head. The coupled head usually feeds the feature map output from the convolutional layer directly into several fully connected layers or convolutional layers to generate the output of the target location and category. YOLOv3 used coupled head. For the detection task of coco80, each anchor will generate a prediction result of $h \times w \times 85$ dimensions, of which obj (distinguish between foreground and background) occupies 1 channel, reg (coordinate) occupies 4 channels, and cls (predict which class among 80 classes) occupies 80 channels.

The decoupled head extracts the target position and category information separately, learns them separately through different network branches, and finally fuses them. YOLOX first uses 1×1 convolution to unify the original feature maps with different channel numbers to 256, and then uses two parallel branches, each of which uses 3×3 convolution.

The comparison between coupled head and decoupled head can be seen in Figure 4.

5.2 Code Understanding

5.2.1 Inference. The image_demo function is used to work with a single image or an image directory, while the imageflow_demo function is used to process a video or camera stream. They first take a list of images then perform inference and visualization on each image.

The inference function is the core of the model inference. It takes an image input and converts it into a tensor, which is then passed through the model to make a prediction. As described in the paper above, the inference result is a bound box with a class label. The visual function receives the prediction results and image information, and then draws the predicted bounding boxes and category labels on the original image.

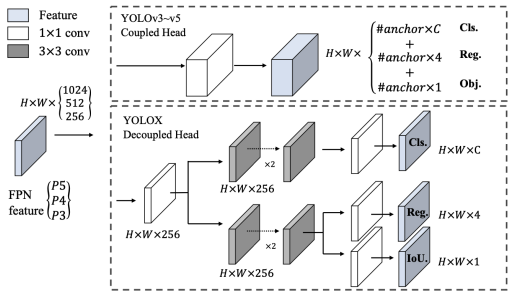

Fig. 4. Comparison between Coupled Head and Decoupled Head

5.2.2 Train. The main function configures the environment for distributed training, and creates a YOLOX trainer using the experiment configuration.

The trainer first performs initialization on model, optimizer and data loader according to the user-specified parameters in Exp object. The optimizer is initialized and the resume_train method is called to check whether the specified checkpoint file is presented. During the training procedure, data augmentation is closed in the last 15 epochs, so there is a no_aug variable indicating whether data augmentation is needed, which is set in the initialization step of data loader. It then comes to the data prefetcher, which can speed up data loading. If the occupy parameter is set, the function calls the occupy_mem method to occupy some memory. If distributed training is used, the model is wrapped as a DDP object. If the exponential moving average of the model is used, a ModelEMA object is created and the number of updates is set.

The model performs the following steps in each iteration: preprocessing of the data, forward propagation of the model, back propagation of the loss, parameter update of the optimizer, update of the EMA model, update of the learning rate, and update of the measurer. First, take the next batch of input and target data from the prefetcher and transfer them to the specified data type. Next, it turns off the gradient computation on the target data and then preprocesses the input data and uses the context manager to perform the forward propagation of the model, which enables mixed-precision training. It then takes the total loss from the output of the model. After that, the gradient of the optimizer is then cleared, and the gradient scaler scales and backpropagates the gradient of the loss. If the exponential moving average of the model is used, the model updates the EMA model and updates the learning rate of the learning rate scheduler.

5.3 Experiments

We deployed our environment on the cluster in HPC Lab. The hardware and software configuration is given in Table 1. We trained three models separately on COCO dataset[6], VOC dataset[1], and SUSTech dataset collected by ourselves, then compare their performance especially on autonomous driving related detection.

5.3.1 COCO Dataset. First, we reproduce on the COCO dataset which is used in the original paper. The COCO (Common Objects in Context) dataset is a large-scale benchmark dataset widely used in computer vision research for object detection, segmentation, and captioning tasks. It contains over 330,000 images with annotations for 80 object categories and 91 "stuff" categories. These annotations include object detection with bounding box coordinates, instance segmentation with detailed annotations, and 5 captions per image, which describe the scene.

Classification	Version
CPU	Intel(R) Xeon(R) Gold 5320 CPU @ 2.20GHz
GPU	A100
Memory	125G
GPU Driver	525.125.06
CUDA	12.0.

Table 1. Hardware and Software Configurations

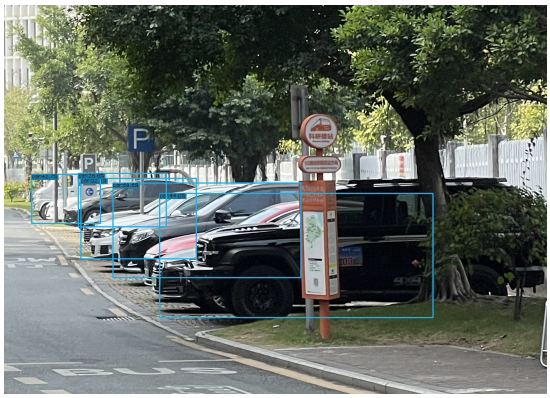

Fig. 5. YOLOX-COCO Result: Good at Cars

In this part, we reuse the training hyperparameters of yolox_s to do reproduction. After training, we used our collected data as test case input. We found this model performs good at detecting vehicles, but the support of the dataset for landmark recognition was not perfect. Fig 5 shows the excellent performance of this model in vehicle recognition. In the case where there are many overlapping cars, the YOLOX-COCO model can still successfully label each car separately. However, in Fig 6, although three person are annotated correctly, the big speed limit sign was not detected.

5.3.2 VOC Dataset. We also tried the VOC dataset to test the scalability of the YOLOX model. The PASCAL VOC (Visual Object Classes) dataset is a well-established set of standardized image datasets for object class recognition. It has been instrumental in advancing computer vision research, primarily through its annual competitions from 2005 to 2012. The dataset consists of images from 20 different object categories, ranging from animals like cats and dogs to vehicles like airplanes and bikes, as well as everyday objects like bottles and chairs.

Here we instantiate Dataset class, Evaluator class and Exp class as described above. However, the training result is not as good as YOLOX-COCO model. This is probably due to the fact that we did

Fig. 6. YOLOX-COCO Result: Bad at Signs

Fig. 7. YOLOX-SUSTech Result: Good at Signs

not tune the model further, and yolox_s is a small model but has undergone a lot of fine tuning. Since our goal of this project is to apply YOLOX to object detection related to autonomous driving, we chose to label the newly collected campus road data set and then add this data to the YOLOX-COCO model for training.

5.3.3 SUSTech Dataset. The dataset for this part of the experiment consists of two parts. Firstly, we collected some picture data related to road signs and vehicles on the school road and annotated them. Second, due to the small scale of our own labeled dataset, and in order to make the model better in image recognition related to autonomous driving, we extracted the data related to vehicle and outdoor in the COCO dataset as part of this training data.

We have seen earlier that YOLOX-COCO achieves good results on car and person detection. Also, since the COCO dataset itself has 80 classes, this model generalizes very well for many different scenarios. So, we choose to conduct a new training based on the training of the YOLOX-COCO model. The new training set contains 33825 images with 298460 annotations, while the test set is consist of 1450 images with 12970 annotations. The new training iterates over the original model for 30 more iterations, with data augmentation turned off for the last 10.

After training, we let YOLOX-SUSTech make predictions on the test set. We find that the new model makes great progress in the recognition of both road signs and vehicles. Comparing Fig 6 and Fig 7, it can be found that the original YOLOX-COCO model can only recognize three people, but the YOLOX-SUSTech model can recognize the road sign in the middle, and when recognizing the person riding the scooter in the middle, not only the Person but also the motocycle are marked. We also compare other vehicles' AP in Fig 8, from which we can see that YOLOX-SUSTech model performs better in most of the cases.

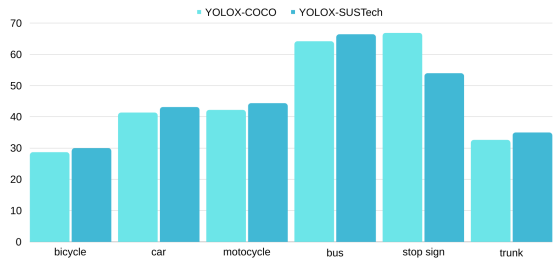

Fig. 8. YOLOX-COCO AP vs YOLOX-SUSTech AP

6 STAFFING PLAN

Yuzhou Qin. Responsible for conducting the literature review, understanding the YOLOX codebase, and reproducing.

Sicheng Zhou. Collaborates with the lead researcher in understanding the YOLOX codebase and contributing to the reproduction efforts. Takes an active role in preparing the SUSTech dataset.

7 TIMELINE

Week 1-2: Background Research and Literature Review. Conduct an in-depth literature review on YOLOX, focusing on the original research paper, relevant conference proceedings, and open-source contributions. Summarize key insights, strengths, and weaknesses identified from the literature.

Week 3-4: Code Understanding and Reproduction. Thoroughly study the YOLOX codebase, gaining a clear understanding of its architecture, key components, and implementation details. Begin the process of code reproduction or successful code execution on the original dataset used in the YOLOX research paper.

Week 5: Dataset Preparation and Initial Testing. Collect SUSTech dataset in our campus and test the performance of our models using this dataset. Label it and train YOLOX model on the SUSTech and VOC dataset.

Week 6: Evaluation and Final Report. Evaluate the YOLOX model's performance based on benchmarking metrics, including accuracy, precision, and recall. Compile final results and insights into a comprehensive report. Discuss any challenges encountered during the reproduction and testing process.

8 CONCLUSION

In this project, we successfully trained three models on three datasets, with a primary focus on the recognition of autonomous driving related images. These models are based on the YOLOX object detection and tracking framework, which we thoroughly investigated and extended in this project. Our research included an in-depth literature

review on object detection and tracking methods, with particular attention to the evolution and advantages of YOLOX over other techniques. By delving into the YOLOX research paper and codebase, we gained a comprehensive understanding of its architecture, design philosophy, and implementation details.

Subsequently, we applied this knowledge to train YOLOX on varying datasets, which allowed us to evaluate the model's generalization capabilities across different scenarios. The results from these experiments demonstrate that YOLOX, with its anchor-free approach, decoupled head, and SimOTA label assignment strategy, can achieve state-of-the-art performance. Notably, our adaptations of YOLOX have led to improvements in autonomous driving related images recognition, showcasing the model's robustness and effectiveness in practical applications.

REFERENCES

[1] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. 2015. The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision* 111, 1 (Jan. 2015), 98–136.
[2] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. 2021. YOLOX: Exceeding YOLO Series in 2021. *arXiv preprint arXiv:2107.08430* (2021).
[3] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. 2021. *YOLOX repo on Github.* https://github.com/Megvii-BaseDetection/YOLOX
[4] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. 2021. *YOLOX-s.* https://github.com/Megvii-BaseDetection/YOLOX/releases/download/0.1.1rc0/yolox_s.pth
[5] labelmeai. 2024. *Labelme.* https://github.com/labelmeai/labelme
[6] Microsoft. 2017. *Commen Objects in Context.* https://cocodataset.org/#home
[7] Joseph Redmon and Ali Farhadi. 2018. YOLOv3: An Incremental Improvement. (2018). arXiv:1804.02767 [cs.CV]