Final Report

"African Wildlife Animal Detection"

Yunming Bai

February 28, 2021

1 Introduction

Object detection refers to a collection of related tasks for identifying objects in images or videos. The aim is to find all the objects of interest in the image and determine their 2D location. The region of interest is often defined by a bounding box. As one of the fundamental computer vision problems, object detection can provide valuable information for semantic understanding of images and videos and is related to many applications, including human behavior analysis [1], face recognition [2], and autonomous driving [3]. In this project, one animal detection system is implemented based on YOLO V1 and the African Wildlife animal dataset. The system can be used in autonomous vehicles to reduce collisions between vehicles and animals on roads. Moreover, as an efficient method to analyze the images collected by Motion-sensor cameras in natural habitats, this system can also help the study of wildlife animals and improve our ability to conserve ecosystems.

2 Related Work

Nowadays, deep learning methods in the field of target detection are mainly divided into two categories: target detection algorithms of two stages and target detection algorithm of One Stage. Two-stage methods first generate potential bounding boxes in an image and then run a classifier on these proposed boxes. After classification, post-processing is used to refine the bounding boxes, eliminate duplicate detections, and rescore the boxes based on other objects in the scene. This type of model mainly include R-CNN [4], Fast RCNN [5], Faster R-CNN [6]. The One-stage detector, on the other hand, directly transforms the target border positioning problem into the regression problem and requires only a single pass through the neural network. One-stage frameworks mainly include YOLO [7], Multi-Box [8]. In this project, YOLO is implemented to solve this animal detection problem because of its simple structure and fast detection speed.

3 Dataset

In this project, the Kaggle African Wildlife animal dataset is used. [9] Four animal classes commonly found in nature reserves in South Africa are represented in this data set: buffalo, elephant, rhino, and zebra. This data set contains 370 images for each animal class collected via Google's image search function and labeled for object detection.

The images of the original dataset have different aspect ratios and contain at least one example of the specified animal class. Moreover, multiple instances of animals can exist in a single image with and without overlapping. Figure 1 shows some examples from the dataset with ground truth bounding boxes.
During the training phase, experiments show that the number of images is limited to train a model with good generalization performance. Therefore, for each class 100 more images are collected and manually labeled in the YOLO format using the graphical image annotation tool LabelImg. [10] Most of the collected images contain only one instance from different perspectives or multiple instances which are well separated. The reason for this specific collection is that these easily detected scenarios are the basis for complicated situations, for example, multiple instances with overlapping. The collected images are all used for training and none of the collected images is already included in the test dataset, which ensures that none of the images in the test dataset is previously seen during the training process. The final dataset is split into training and test datasets with 1595 and 220 images respectively. The test dataset contains not only images with a single instance but also with multiple instances with an intersection, which ensures a thorough and reasonable evaluation of the trained model.

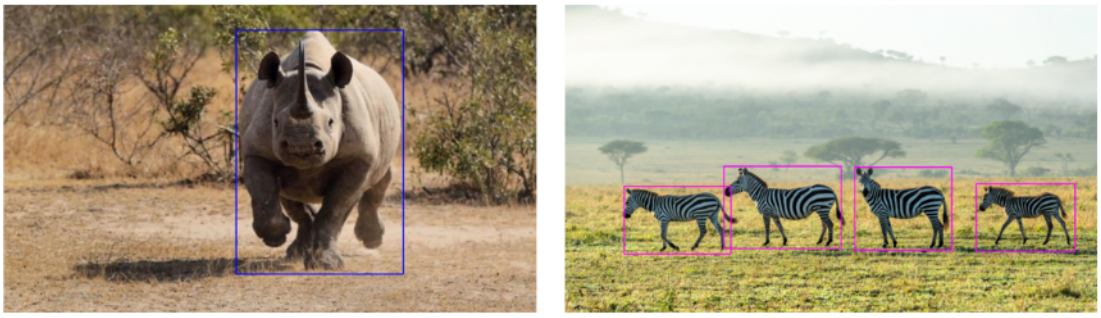


Figure 1: Images from Wildlife animal dataset with groundtruth bounding boxes

4 Method

4.1 Network Design

The architecture of YOLO V1 has 24 convolutional layers followed by 2 fully connected layers. In the original paper, this network is pretrained on Imagenet for two weeks. Considering the large model and high training effort, a transfer learning-based YOLO is implemented in this project. The whole Model structure is shown in Figure 2. Firstly, the initial part of pretrained Resnet50 is used as a feature extractor, which includes the convolutional layers and pooling layers before fully connected layers. Resnet50 [11] is chosen in this project because of its great performance on classification problems on ImageNet. The output of the feature extractor has a dimension of 2048. Secondly, this intermediate representation is passed through two fully connected layers with 1024 and

$5 \times 5 \times 14$ hidden neurons respectively. The first fully connected layer is followed by a ReLU activation function and dropout layer of 0.3. In this model, only FC layers are trained to predict the locations of the bounding boxes and the corresponding class.
The output of the model is defined as $5 \times 5 \times 14$. YOLO divides the input image into an S $\times$ S grid. If the center of an object falls into an S $\times$ S grid cell, that grid cell is responsible for detecting that object. For each grid cell, the model predicts B bounding boxes, confidence for those boxes, and C class probabilities. Each bounding box consists of 5 predictions: x, y, w, h, and confidence. The (x, y) coordinates represent the center of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image. These predictions are encoded as an S $\times$ S $\times$(B * 5 + C) tensor. Considering that the chosen dataset has only 4 classes, C is 4 in this project. S is set as 5 considering that the density of animals in the chosen dataset is not quite high and each grid predicts 2 bounding boxes.

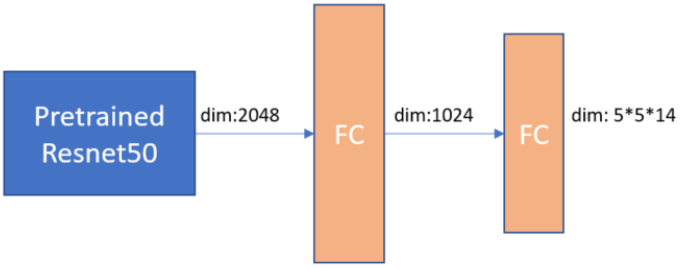


Figure 2: Model Structure

4.2 loss function

In this project, the multi-part loss function in the original paper is applied, which consists of 5 parts. In equation 1 $\mathbf{1}_i^{obj}$ denotes if object appears in cell i and $\mathbf{1}_{ij}^{obj}$ denotes that the jth bounding box predictor in cell i is "responsible" for that prediction. In this project, YOLO predicts B=2 bounding boxes per grid cell. During the training phase, the bounding box predictor which has the highest IOU with the ground truth is responsible for each object and included in the loss function.
The first two parts are losses for coordinate predictions: x, y, width, and height. The third part is related to the confidence score. The fourth part is introduced to avoid predicting a bounding box without an object. The last term represents the loss for the predicted class. The loss from bounding box coordinate predictions is weighted with λ_{coord} and the loss from confidence predictions for boxes that don't contain objects is weighted with λ_{noobj}. In this project, these hyperparameters are chosen according to the original paper as λ_{coord}=5 and λ_{noobj}=0.5. The loss function only penalizes classification error if an object is present in that grid cell. It also only penalizes bounding box coordinate error if that predictor is "responsible" for the ground truth box.

$$\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbf{1}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] +$$
$$\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbf{1}_{ij}^{obj} [(\sqrt{w}_i - \sqrt{\hat{w}_i})^2 + (\sqrt{h}_i - \sqrt{\hat{h}_i})^2] +$$
$$\sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbf{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 + \tag{1}$$
$$\lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbf{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2 +$$
$$\sum_{i=0}^{s^2} \mathbf{1}_{i}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2$$

5 Experiments

During the training process, the batch size is set as 32. The model is trained for 50 epochs and the Adam optimizer is used. For the first 30 epochs, the learning rate is $2e-5$ and for the later 20 epochs, the learning rate is decreased to $1e-5$. Figure 3 represents the loss and mean average precision on train and test dataset respectively. Mean average precision is calculated at IOU threshold 0.4, which means the predicted bounding box which shares an IOU with the ground truth box bigger than 0.4, is classified as true positive. The final mean average precision on the test dataset converges to 0.48.

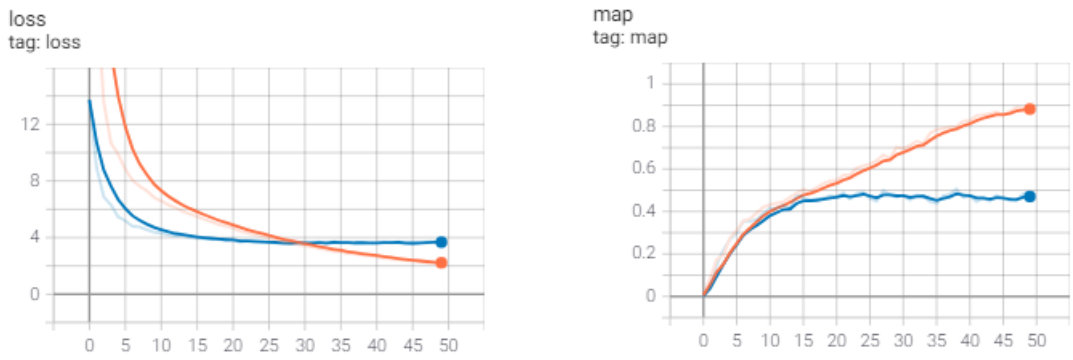


Figure 3: loss and mean average precision (blue: train, orange: test)

The trained model has limited generalization performance. To reduce overfitting, the following approaches have been tried. Firstly, the model complexity is reduced. The

number of hidden neurons in the first fully connected layer after pretrained Resnet50 is reduced to 512. Secondly, dropout rate is increased from 0.3 to 0.5. Thirdly, the l2 regularization term is introduced in the loss function. All these approaches don't significantly reduce the overfitting. Therefore, the possible approach is to collect more data. Although we have collected 100 more images, the dataset is still not large enough to solve the object detection problem. This problem seems to be reasonable because this specific dataset is manually collected and annotated.

6 Result and evaluation

6.1 Qualitative evaluation

The qualitative evaluation will be done by simple visualization. All the following images are from the test dataset.

6.1.1 image with single instance

The model performs generally good on images with only one instance. The bounding box and class of the instance are correctly predicted. (e.g. Figure 4 (a), (b), (c), (d)) Figure 4 (e), (f) show some unperfect result. The predicted bounding box in Figure 4 (e) is shifted compared to the ground truth and the predicted bounding box in Figure 4 (f) is much larger than the ground truth. The localization performance of the model still needs to be improved. One possible solution to this problem is adjusting λ_{coord} in the loss function, which is the weight assigned to coordinate prediction (x, y, weight, and height). Generally, the model output reasonable results on images with one image instance.

6.1.2 image with two instances

The performance of the model on images with two instances depends on the intersection of two instances. If two instances are separated (Figure 5 (a), (b)) or there is only small intersection between them (Figure 5 (c), (d)), the model is able to output reasonable bounding boxes. However, if the overlapping of two instances is large (Figure 5 (f), (g), (h)), the model tends to output one large bounding box, which contains both instances.

6.1.3 image with multiple instance

The performance of the model on multiple images with multiple instances is shown in Figure 6. The performance of the model is limited. It has localization problem (Figure 6 (a), (b)) and problem of predicting large bounding boxes (Figure 6 (c), (d)). One possible approach to improve the performance of the model on two and multiple instances is to increase the number of grid cells. It seems that the grid cell of 5×5 can handle simple scenarios (e.g. image with a single instance or image with two separated instances). However, it is not enough to output reasonable bounding boxes in complicated situations. Therefore, increase the number of grid cells is one idea to improve the model. Another

reason for the limited performance is that not enough complicated images with two or multiple images are collected in the training data. Therefore, collecting more data is also necessary to improve the model.

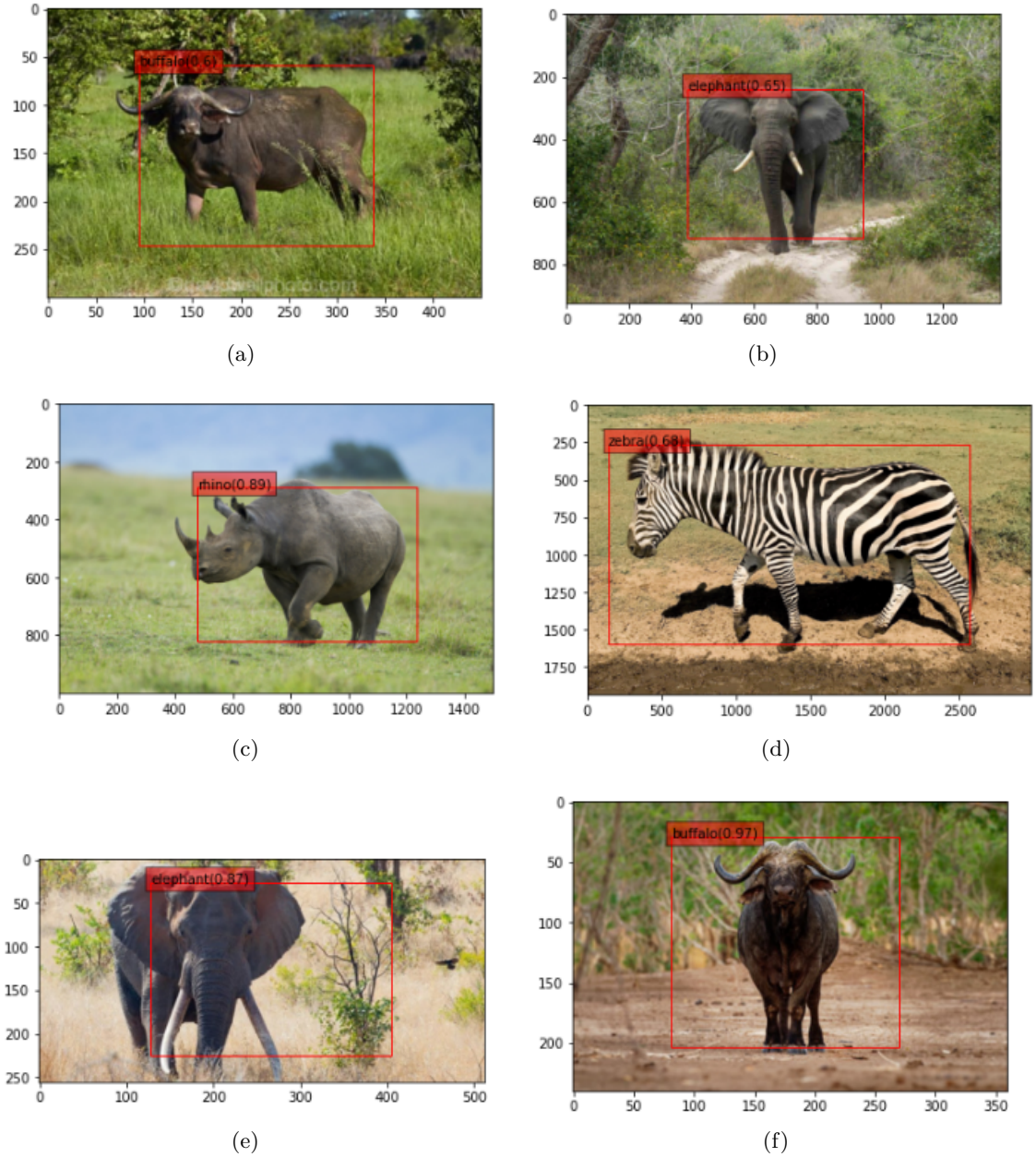


Figure 4: Images with single instance

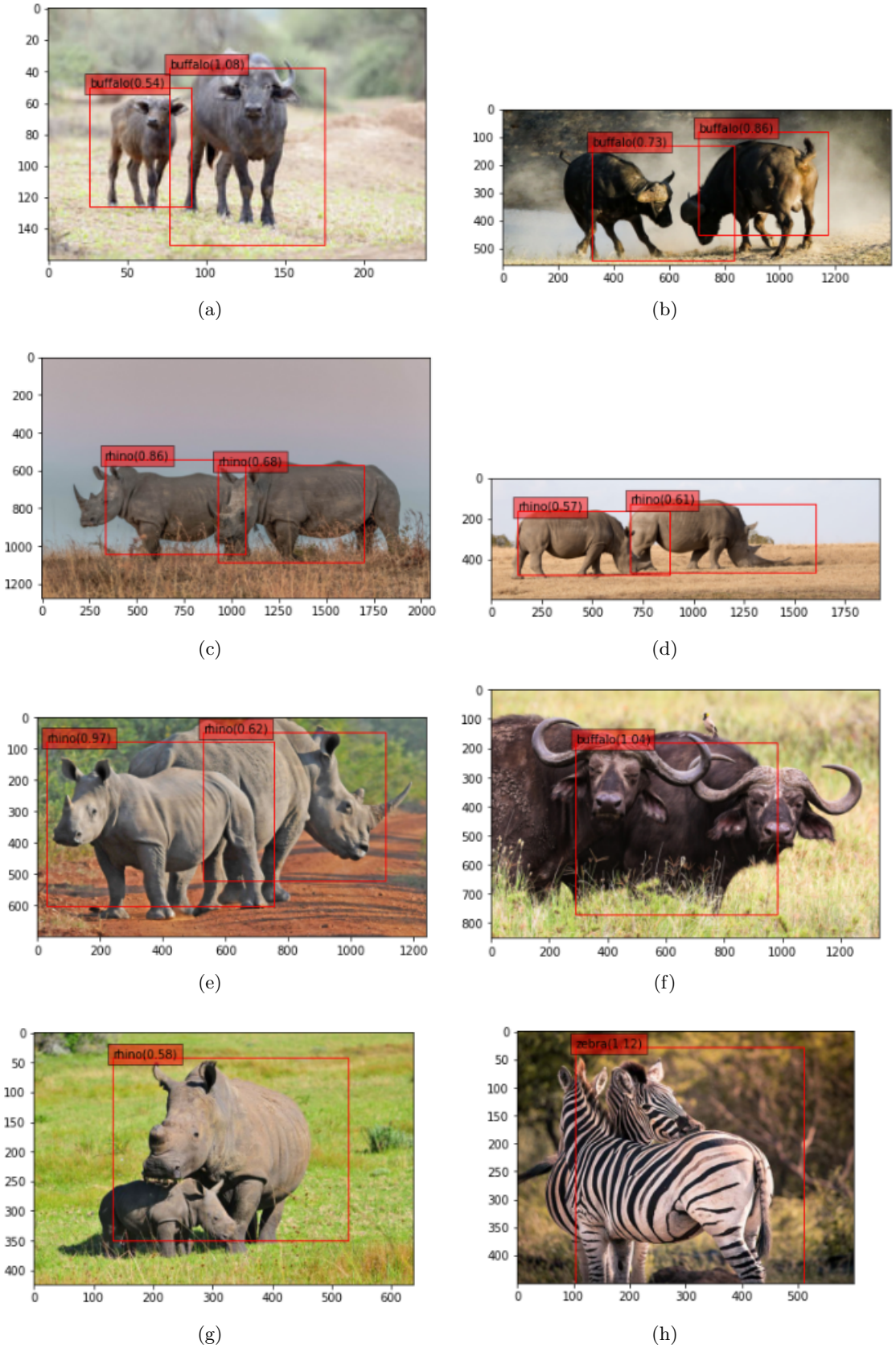
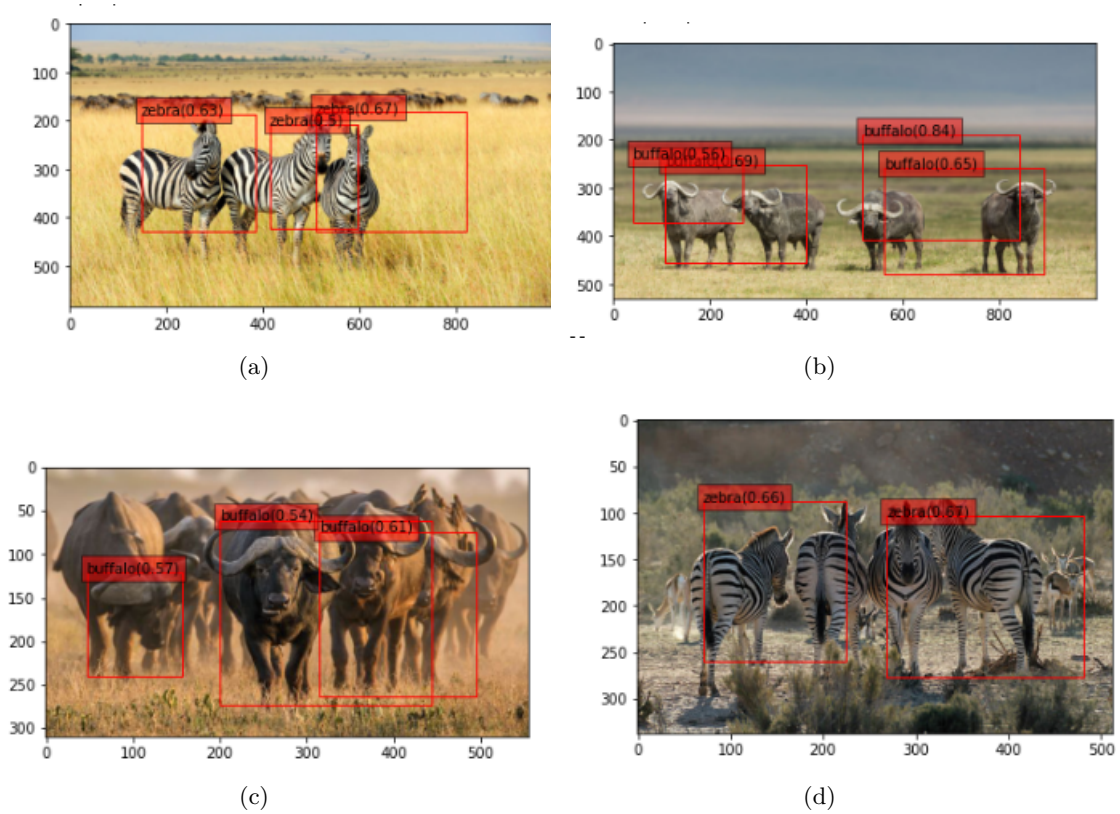


Figure 5: Images with two instances



(a) (b) (c) (d)

Figure 6: Images with multiple instances

6.2 Quanitative evaluation

For quantitative analysis, MAP (mean average precision) is used to evaluate the trained model. It is the most common metric used in deep learning to evaluate object detection models. Firstly all bounding box predictions on the test set are achieved and filtered by NMS (non-max suppression). Then each bounding box will be classified as either true positive or false positive. If one predicted bounding box shares no IOU (intersection over union) with any ground truth box bigger than a certain threshold, it will be classified as false positive, otherwise, it will be recognized as true positive. Then, all the predicted bounding boxes will be sorted in descending order according to their confidence score, which indicates their confidence that there is one object inside the bounding box. Thirdly, the precision and recall will be calculated as we go through all the outputs and the area under the precision-recall curve will be computed. The mentioned work is only related to one specific class. After repeating the mentioned procedure for all classes, the mean average precision under one certain IOU threshold will be achieved. Finally, we need to redo all the computations for many IoU thresholds.
The final result is shown in Table 1. With the increase of IOU threshold, which implies the higher requirement on localization accuracy, the mean average precision gradually

drops. The possible reasons may be the limited number of grid cells and the lack of training images with two or multiple instances and increasing the weights for coordinate prediction in loss function, increasing the number of grid cells and collecting more data may improve the localization performance of the model.

IOU Threshold	0.3	0.35	0.4	0.45	0.5	0.55	0.6
MAP	0.54	0.52	0.48	0.43	0.37	0.32	0.27

Table 1: Mean average precision evaluation

7 Conclusion

In this project, a transfer learning-based YOLO is implemented. The network consists of two parts. A pretrained Resnet50 is used as a feature extractor and the following two fully connected layers are trained to predict the location and class of objects. Considering that object detection is a complicated problem, which includes not only classification but also localization, a large amount of data is necessary for the good performance of the model. Therefore, in addition to the Kaggle African Wildlife animal dataset, 400 images are manually collected and annotated. Experiments show that the trained model performs generally well on images with a single instance. However, the performance on complicated scenarios (e.g. images with multiple instances and instances with large intersections) is limited qualitatively and quantitatively. More improvements on complicated scenarios can be expected by increasing the number of grid cells and collecting more data.

References

[1] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: realtime multi-person 2d pose estimation using part affinity fields. *IEEE transactions on pattern analysis and machine intelligence*, 43(1):172–186, 2019.

[2] Zhenheng Yang and Ramakant Nevatia. A multi-scale cascade fully convolutional network face detector. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 633–638. IEEE, 2016.

[3] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE international conference on computer vision*, pages 2722–2730, 2015.

[4] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[5] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[6] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.

[7] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[8] Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. Scalable object detection using deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2147–2154, 2014.

[9] african-wildlife-dataset. `https://www.kaggle.com/biancaferreira/african-wildlife` Accessed January 29, 2021.

[10] Labelimage. `https://github.com/tzutalin/labelImg` Accessed February 28, 2021.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.