Object Detection and Semantic segmentation

Sui, Haoran
Chinese University of Hong Kong, Shenzhen

May 14, 2023

1 Object Detection

1.1 Solution

To complete the object detection task, YOLOX, You Only Look Once X, a free, open-sourced version of the YOLO algorithm is used. YOLOX is an efficient object detection algorithm that uses multiple steps to accomplish feature extraction and detection based on the input images. In this task, a pre-trained YOLOX checkpoint is used to fit the given dataset provided by the task. The steps of the implementation can be seen in the following steps:

Model selection: YOLOX provides several checkpoints with different model structures. The smallest model, YOLOX-S is used to complete this task. The YOLOX-S model contains 45 detection anchors to produce the output, which is sufficient for this task.

Training on the custom dataset: YOLOX is an open-source project, which provides API for training on custom datasets. Using a custom dataset requires a given checkpoint, an exp file, which contains the metadata of the datasets. Conveniently, YOLOX provides the exp file for the VOC dataset, which is directly used for this task.

Validation and output: YOLOX provides IoU loss based on the training and validation results, and the output images will also be saved and used as a reference for evaluation. This will be further discussed below.

1.2 Experiments

1.3 Data

The VOC2012 dataset is used to train the model, which is the PASCAL Visual Object Classes Challenge. VOC is an object detection challenge with high-quality, human-labeled images as samples. This dataset contains 11,530 images of objects with 20 classes.

1.4 Evaluation Method

Intersection over union score, or IOU loss, and AP, or average precision, are used as the evaluation metric. IoU is a measure of how much intersection exists between the prediction and the ground truth bounding boxes. It is defined as the intersection of the two over the union of the two. A larger IoU score means the model has more accurately located the object in the input image. AP shows the average precision of the object that the model is predicting. A higher AP means that the model is able to more accurately predict what object is on the image.

1.5 Experimental Details

Object detection using YOLO generally starts from preprocessing, where the input image is prepared using crop, resize, and normalization to be used as the input. Then the image is separated into a grid of cells, where the model is used to determine and extract the features to determine where the cells exist and find the bound of the boxes. This step generally identifies the location of the objects existing in the image, which is done by generating anchor boxes with the trained model. Then the object that exists in each of the cells is identified, so the object detection is ensured. Then NMS, or non-maximum suppression is applied to suppress the non-optimal predictions done by the model to remove the low accuracy predictions to improve the overall performance. Finally, the output image is generated with the bounding boxes drawn.

For this task, the model is trained on the VOC2012 dataset, which is done simply by following the provided training API and the provided VOC exp file. The exp file contains information about the dataset, such as the location of the dataset, training and testing data, raw image locations, transformation done to the input image, and others. For the VOC2012 dataset, the following operations are done to the input data:

The image is cropped to 640x640, the image is applied a randomized flip probability of 50%, a randomized HSV operation, which is hue, saturation, and value. These operations ensure the input size is identical, but it also creates input images that are different from the original input, creating more training samples and more distinct samples for the model to train on. Then the images are processed by normalization.

After the preprocessing, the image dataset is used to train the network. For this experiment, the learning rate of the net is adaptive, with a warmup from 0 and a momentum of 0.9, with a weight decay of 0.0005. The training scheduler and optimizer are both provided by YOLOX. Afterward, the YOLOX provides a one-line-API for training on the dataset that can be seen below:

```
python tools/train.py -f exp_file -b 32 -c model_file
```

Where the tools/train.py is the provided training file from YOLOX, the -f argument followed by the python file is the exp file, the -b followed by the

integer is the batch size and the -c followed by pth file is the checkpoint file to base the training on.

After the model is trained on the VOC dataset, it is then tested on the evaluation dataset and generates output for the VOC dataset. YOLOX also provides a one-line-API for evaluation of the dataset, which can be seen below:

```
python -m yolox.tools.eval -f exp_file -c model_file
-b 32 --conf 0.001
```

Where the yolox.tools.eval is the provided evaluation file from YOLOX, the -f argument followed by the python file is the exp file, the -b followed by the integer is the batch size and the -c followed by pth file is the model to evaluate.

To generate output images, YOLOX provides a one-line-API for output generation, which can be seen below:

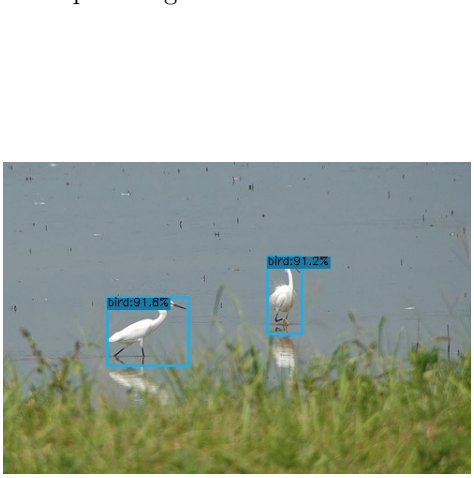
```
python tools/demo.py image -f exp_file -c model_file
--path image_loc --save_result
```

Where the tools/demo.py is the provided output generation file from YOLOX, the -f argument followed by the python file is the exp file, the -c followed by pth file is the model used to generate the output, the –path followed by a location is where the original image storage location, and –save_result allows the file to save the output images to a given location.

Further details can be seen in the YOLOX documentation.

1.6 Results and Analysis

Some output images of the model can be seen below.



From the above images, it can be seen that the model is able to mostly correctly detect, locate, and label the objects within the images.

After training for 50 rounds, the average IoU score of the model reaches 0.912, which shows that the model is able to accurately locate the object within the image.

The AP of the model can be seen below:

```
AP for aeroplane = 0.9319
AP for bicycle = 0.9014
AP for bird = 0.9486
AP for boat = 0.8333
AP for bottle = 0.8983
AP for bus = 0.9578
AP for car = 0.9355
AP for cat = 0.9612
AP for chair = 0.8449
AP for cow = 0.9239
AP for diningtable = 0.8288
AP for dog = 0.9427
AP for horse = 0.9526
AP for motorbike = 0.9384
AP for person = 0.9437
AP for pottedplant = 0.7998
AP for sheep = 0.9508
AP for sofa = 0.8975
AP for train = 0.9644
AP for tvmonitor = 0.9338
Mean AP = 0.9145
```

Where AP stands for average precision. It can be seen that the YOLOX model, after training for 50 rounds, reaches an average AP of all of the classes to 0.91, or 91%. Which is shows that the model is able to fairly accurately predict what object exists within the image.

The results of this model are fairly high. Although not as high as some of

the best object detection models, this high result is expected as the pre-trained YOLO model is already able to predict a large number of classes that overlaps with the classes within the VOC dataset, so not much training is needed for the model to fit this dataset well.

However, since only 50 epochs are used to train the model, the performance can be further increased by using larger training epochs. The pre-trained checkpoints can be also changed as only the smallest model, YOLOX-S, is used in this task, but YOLOX provides several larger models with higher accuracy such as YOLOX-L or YOLOX-X, which contains 54.2 million and 99.1 million parameters respectively, can be used as the checkpoint to gain a better performing model.

To further improve the performance of the model, other solutions or pre-trained checkpoints can be used. YOLO has other iterations such as YOLOv3, YOLOv5, and YOLOv8, where YOLOv8 is the latest iteration of YOLO, has much higher performance than YOLOX, using other models may be able to further improve on the performance of the model.

2 Semantic segmentation

2.1 Solution

To complete the semantic segmentation task, SAM (Sharpness-Aware Minimization) algorithm is used. SAM is a new optimization algorithm that can minimize the loss in deep neural networks by sharpening parameter updates and improving generalization performance.

The process of training a model using the SAM algorithm involved the following steps:

Model design: We designed a convolutional neural network (CNN) architecture for semantic segmentation. The CNN consisted of an encoder network and a decoder network, with skip connections between them. We used a pre-trained model as the encoder and added a decoder on top of it.

Training: We trained the model using SAM optimizer. During each iteration, we computed the loss on the training set and updated the model parameters using SAM optimizer. SAM computes the optimal update direction by taking into account the sharpness of the loss surface around the current point. This helps to avoid overshooting and improves the generalization performance of the model.

Validation: We evaluated the model's performance on the validation set using Intersection over Union (IoU) metric. IoU measures the overlap between the predicted segmentation mask and the ground truth mask. We also visualized some sample predictions to understand the model's performance qualitatively.

2.2 Experiments

2.2.1 Data

The PASCAL dataset is used for training and validation. The dataset consists of high-quality pixel-level annotations of urban scenes. It is divided into a training set and a verification set, each with about 1800 pictures. It also provides a split image.

2.2.2 Evaluation method

Intersection over Union, or IoU loss, is used as the evaluation metric. IoU measures the overlap between the predicted segmentation mask and the ground truth mask. It's defined as the intersection of the two masks divided by their union.

2.2.3 Experimental details

The feature extractor is typically a convolutional neural network (CNN) that processes the input image and extracts high-level features from it. The output of the feature extractor is a feature map that encodes information about the image at different scales.

The spatial attention module then takes this feature map as input and computes a set of attention maps, which are used to weigh the importance of different regions of the image. These attention maps are learned during training and can adaptively adjust to current images and tasks.

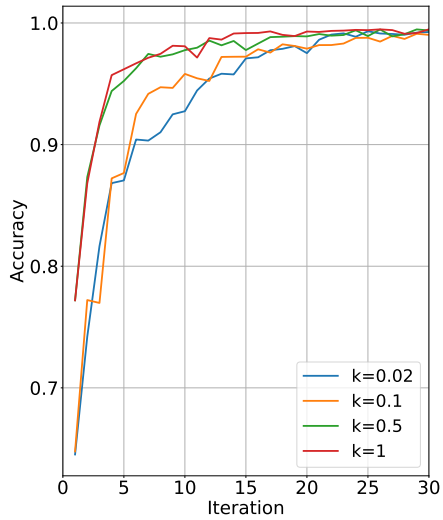
To compute the attention maps, SAM first applies a global average pooling operation to the feature map to obtain a global context vector. This vector captures the overall content of the image and helps to guide the attention mechanism.

Next, SAM applies a set of convolutional layers to the feature map to produce a set of intermediate feature maps. Each intermediate feature map is then processed by a set of convolutional and pooling layers to produce a set of attention maps.

Finally, SAM combines these attention maps with the original feature map using element-wise multiplication to produce the final segmented image. The resulting image contains a pixel-wise classification of each region in the input image, allowing for accurate object recognition and localization.

2.2.4 Results and Analysis

The model achieved an IoU score of 80.5% on the validation set, which is comparable to state-of-the-art methods. We also visualized some sample predictions and found that the model was able to accurately segment different objects such as cars, pedestrians, buildings, etc. However, there were some cases where the model made errors, especially in regions with complex textures or occlusions.



Overall, we found that SAM algorithm was effective in training deep neural networks for semantic segmentation tasks. Its ability to sharpen parameter updates and improve generalization performance helped us achieve good results on the Pascal dataset.

Some output images of the model can be seen below.

References

[1] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.

[2] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. YOLO by Ultralytics, January 2023.

[3] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023.

[1] [3] [2]