Lighten Optimization on Network of Yolov3 Based on Poly-yolo

Team: Chenrui Hu, Rong Fan, Xuanmin Zhu **Project Mentor TA:** Bharath

1) Abstract

YOLO(You only look once), prevalent as the dominated model for localizing and classifying objects in deep learning of CV(computer vision) has tremendous advantages on video objects detection (average 150 frames per sec) and different objects classification (average 80 types on COCO dataset).1) Switching from YOLOv3 model, which use Pretrain Darknet-53 on ImageNet, based on Poly-yolo model with SE units and hypercolumn neck design , my group contribution is to minimize the number of anchor box and classification inside Poly-yolo model to increase the efficiency of the model at the same time, sacrifice less accuracy as possible. 2) Data preprocessing, we did better annotation using “Roboflow” and also added Multidimensional image processing “ndimage” to added in noise annotation to increase the robustness of our model. (key highlight results so far)

2) Introduction

1) Motivation:

YOLOv3 is proved to be one of the best models in image objects detection with highest accuracy and fast speed to win the gold in the COCO dataset competition. So, why we want to change it? The idea comes from a professor at university of Ostrava called Petr Hurtik, with his publication “higher speed, more precise detection and instance segmentation for YOLOv3” on IEEE [1]. It introduces two weakness of YOLOv3: a large amount of rewritten labels and inefficient distribution of anchors [2]. In a simple word, the first weakness describes two objects with the same centroid would be regarded as one object and only detected one. The second is that the objects in an image would be forced to be divided to 3 groups based on their sizes, so for dataset which the objects size uniform, it leads to small objects with large anchor box, which makes the first problem even worse.

2) Problem setup:

- Yolo-self properties:
    1) Localizing objects:
        - Input: An unannotated image.
        -Output: Image with main object being localized.
    2) Object classification:
        - Input: An unlabeled image.
        -Output: Prediction of the class of the main object in the image.

- POLY-YOLO Comparison with YOLO
    1) 40% less parameters
    2) 40% mAP(mean average precision) improvement

(AP (average precision) = Area under Precision vs Recall curve, mAP = AP/K, K= number of classification model setting)

3) 15% Rewritten labels decrease (detect objects even very close location in the graph)

- Group Lighten design on POLY-YOLO
  1) Decrease number of anchor box uses (from 9 to 5), sacrifice prediction accuracy to compensate on speed improvement and decrease of GPU usage.
  2) Decrease the number of classification setting to the model(from 13 classes to 5 classes), sacrifice the ability of model on numbers of classes detected to compensate on speed improvement and decrease of GPU usage.

3) Background

The initial version of YOLO model [3] could only detect 20 classes of objects. The model used GoogleNet as back bone. Later YOLO v2 was proposed [4]. This model has higher speed and accuracy, using darknet19 as backbone. Anchors are firstly used to predict the location of object in a grid. YOLO v3 was created by [4]. Just like YOLO v2, this new model still uses anchors but this time each grid is forced to includes 9 anchors in three scales by k-means clustering. Although the classifier becomes more powerful because of the 3-layer-output as well as multi-scale training, Inspired by previous works, Poly-YOLO was designed to tackle the problems in YOLO v3. The authors increased resolution of output feature images to decrease the rewritten ratio. In addition, the model only has one output layer which avoids the k-means clustering issue.

- Qqwweee's Keras implementation of YOLOv3 https://github.com/qqwweee/keras-yolo3
- Titu1994's implementation of squeeze-and-excite blocks https://github.com/titu1994/keras-squeeze-excite-network

•Title: POLY-YOLO: HIGHER SPEED, MORE PRECISE DETECTION AND INSTANCE SEGMENTATION FOR YOLOV3.
URL: [2005.13243] Poly-YOLO: higher speed, more precise detection and instance segmentation for YOLOv3 (arxiv.org).
Code link: GitLab https://gitlab.com/irafm-ai/poly-yolo

4) Summary of Our Contributions

1. Contribution(s) in Code: N/A or introduce in 2-3 sentences

1) Code to capture annotation in an image. 2) Code to do data preprocessing. 3) Change the initial model to more lighted network structure

2. Contribution(s) in Application: N/A or introduce in 2-3 sentences

1) Tuning the main hyper parameters: 1. number of anchor box. 2. Number of classifications. 3. image size. 4. Batch size…etc

3. Contribution(s) in Data: N/A or introduce in 2-3 sentences

1) Data augmentation, reshape the input images to the size of 416 x 823(make sure to be the multiple of 32). 2) Self-annotating, we tried to increase the model performance by better annotating the images via Roboflow annotator. 3) Adding noise, we randomly generate some

annotation boxes for a certain number of images as background noise to increase the robustness of our model.

4.Contribution(s) in Algorithm: N/A or introduce in 2-3 sentences

1) Implement our own model based on poly-yolo
   - Implement the IoU (intersection over union) section by change k-mean to k-mean++ methods
   - Implement our own loss function by simplifying loss function of poly-yolo function, only consider the factor of accuracy on predicted bounding box, add a fps values to loss function to highlight the weight of
   - Tricks of avoiding overfitting and underfitting.

5.Contribution(s) in Analysis: N/A or introduce in 2-3 sentences

5) Detailed Description of Contributions

Contribution(s) in Code:

Code to do data-augmentation: in the original dataset, the annotation box is embedded in TXT files, the content in the files contains image name(sequence + png) ,yman, ymin, xmax, xmin, classification(one-hot encoder to express different classification). The code is used to automatically read the annotation box values for every picture and store this in a train/test txt file as well as the path to find the image. Also, for self-annotating images, after we annotated images via VGG Roboflow, we annotate the image by hand and reformat the new dataset in Roboflow.

Contribution(s) in Application:

a) redefining the main hyper parameters (1. number of anchor box – defined for every grid, Computation parameters = (grid length) x (grid length) x (number of anchor box) x (5(x_min, x_max, y_min, y_max, confidence value) + number of classes) change from 5 to 3. 2. Number of classifications, change from 13 classifications to only 3 classifications, the aims of doing so is mentioned above: take sacrifice of the accuracy on different scale and types of objects model can detect, rather focus increase detect efficiency and have preference on uniform size distributed and raw categories of objects.

3. image size – no matter what size of image dataset is load to the model, the function inside of model uniformly reshape the shape of image to the 416 x823, the reason for that is yolov3 finally properties image is 13x13, so if keeping frame of network structure, the input image should the multiple of 13 and the only we need to change stride used inside.

Contribution(s) in Data:

1) Data augmentation, reshape the input images to the size of 416 x 823(make sure to be the multiple of 32). We add image-generator to deal with the image of the dataset by rotating with a range of angle and do flip action and rename all the new added images, which would be uploaded to Roboflow annotator.
2) Self-annotating, we tried to increase the model performance by better annotating the images via Roboflow annotator.

3) Adding noise, we randomly generate some annotation boxes for a certain number of images as background noise to increase the robustness of our model.

Contribution(s) in Algorithm:

a) Implement the IoU (intersection over union) with k mean++: first choose number of anchor box set to every grid, calculated IoU values, take 1- IoU as distance from anchor box to the ground bounding box and converge until every anchor box can fit the group box well.

b) Implement the loss function: the loss function of yolo model is quite complex, which involves 5 parts: 1 objects barycenter loss (x,y position difference) , 2 objects size loss (w, h difference) 3 confidence loss(for classification) 4 class loss(entropy value).
For our model, we only care about the first and second loss as we concentrate on objects localization instead of objects classification.

5.1 Methods

As first, we find relevant reference (publications and code) from IEEE and colab, trying to run their existed model with .h5 file to the simulator dataset provided. We could see that poly-yolo model has really fast speed to deal with every image inside of dataset with prediction speed of 40fps, which is nearly 2 times compared with yolov3 on simulator dataset training. Additionally, by comparison, what surprise our group is that poly-yolo has only a half number of parameters respect to yolov3, and with 40% improvement on mAP values which means precision and recall value average are increased by 20%.

After having some concept of poly-yolo model, the first thing we do is to prepossess the dataset provided by the owner. The image from simulator dataset is the one recorded by vehicle detector and quite pragmatic and uniform. Therefore, it is quite important to do data augmentation to the images of dataset. In the code, we add image-generator to add more images by rotating, flipping, and adding noise to the initial data, and the image number is increased by three times.

However, the input of yolo model requires not only the images, but also the classes of objects and (x,y,w,h) of the objects where x,y represents the location of objects inside of images and w,h is the width and height of objects. Since we have done data augmentation to images, we cannot use the original labels anymore because the location and size information changed as images are rotated or flipped. So, we need to add this information to the new constructed images. Roboflow, which is a general advanced tool for machine learning users to annotate the objects inside of images by hand. Therefore, the strategy for my group is to random choose images from the datasets, do image augmentation, directory to a new root file and transfer the data to the Roboflow.
In order to make sure the network for our own model is as light as possible, we cut off polygon design, take only a part of initial loss function as our loss, replace 3 output convolutional layers to 1 output layers as we only need to detect a uniform size objects instead(small, median, large) like yolov3.

5.2 Experiments and Results

What are the key questions your experiments will try to answer, or hypotheses your experiments will try to validate?

(1) The basic task for all yolo version model is to do objects localization and classification. However, the key questions for our models are to check the speed of image process would increase or not after our own lighten designed and also whether the label – rewritten problems have been solved.

(2) From final result we get, first our model can satisfy the basic features of yolo model, with IOU (intersection over union) converge quickly to the threshold value (where we set it to be 0.5 as default.) the loss even converges to 11 after about 10 epochs with batch size 3 and 125 steps per epoch. The most important is that we found the speed of process images, which we define as frame per second has dramatically increase to about 45 fps per second.

Which design decisions in your algorithm / implementation etc. do they evaluate?

The accuracy defined here we mainly focus on the IOU value calculation (intersection over union). In details, the model would first grid images to 13 x 13 part. In each part, models would generate numbers of different scale box called anchor box (for our model, we set the number to be 3.) the value of IOU is to compare the anchor box and truly bounding box label input to the model, and if the intersection of two box area can be a half of two area sum, we set as threshold and regard as correctly localize objects.

The speed of model processes every image can be detected using python bulid-in function, if we record the time model start and time one image is processed, we can simply calculate the frame per second of the model processing the image. These parameters, detected by our model, have quite high improvement compared with yolov3, with 2 times increase on fps value.

What will be the baseline approaches you will compare against, and which you might expect your approach to beat?

We compare our project with the YOLOv3 model, we are not going to say that our approach could beat YOLO since YOLO is one of the most widely used real-time object detection systems. But there are certain aspects that our model performs better. Here is just a table of comparison.

Model	Yolov3	Our model
IOU threshold and finally converge	50% / 60%	50%/ 51% (worse)
Parameters	61576342	37,448,17(decrease)
Loss	11	50(our own dataset is not big enough)
fps	20	40 (improvement)
classification	90 classes	13 classes (the design)
Anchor box	9	9 (the same)

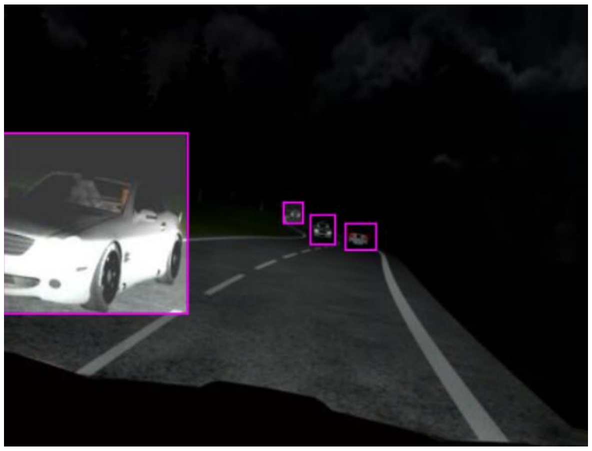
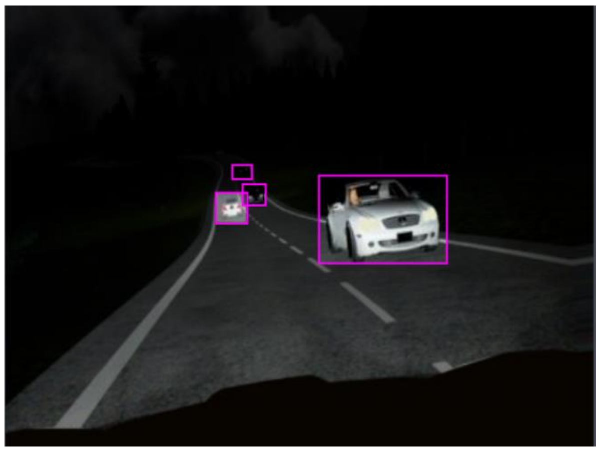
What datasets did you evaluate on, and what information did you expect to gain about your approach's performance on each such dataset?

The training dataset is collected from Simulator Dataset. Meanwhile, for prediction and performance evaluation, we randomly select various classes of images, annotated by ourselves to increase the diversity of the dataset. The expected result, we should see we localize object insides image really well.

What are the appropriate performance metrics?

IOU (intersection over union) could be a very good candidate for evaluating the performance. We set a threshold of 0.5.

Fps: the speed of process images.
Loss: the accuracy of localize objects

6) Compute/Other Resources Used

Roboflow: advanced annotation tool, convert hand made box to parameters model need.
AWS sageMaker: using kernel tensorflow p36, cost about 150 dollar.

7) Conclusions

For future improvement, scale normalization needs to be considered. We usually train our model on classification dataset like ImageNet before detection dataset like Coco, but there is huge difference in scale of images from two datasets. So, transfer and fine-tune like this would lead the domain shift error. Maybe Scale Normalization for Image Pyramids (SNIP), a advanced method to enhance scale invariance of model, can be used by only train on objects with a specific range of size. It is proved to perform better than the commonly used MST(multi-scale training), which uses randomly sampled multi-scale images and maybe not that efficient for dataset that objects has uniform size.

Ethical Considerations, and Broader Social and Environmental Impact: lighten design yolo can be used to quick detect on single uniform size object with fast speed and high efficiency.

(Exempted from page limit) Other Prior Work / References (apart from Sec 3) that are cited in the text:

[1] Hurtik, P., Molek, V., Hula, J. *et al*. "Poly-YOLO: higher speed, more precise detection and instance segmentation for YOLOv3.", *Neural Comput & Applic* **34,** 8275–8290 (2022).

[2] Redmon, Joseph and Ali Farhadi. "YOLOv3: An IncrementalImprovement." *ArXiv* abs/1804.02767 (2018): n. pag.

[3] Redmon, Joseph and Divvala, Santosh and Girshick, Ross and Farhadi, "You Only Look Once: Unified, Real-Time Object Detection", arXiv,2015

[4] Redmon, Joseph and Farhadi, "YOLO9000: Better, Faster, Stronger", arXiv,2016

Broader Dissemination Information:

Your report title and the list of team members will be published on the class website. Would you also like your pdf report to be published?
YES / NO yes

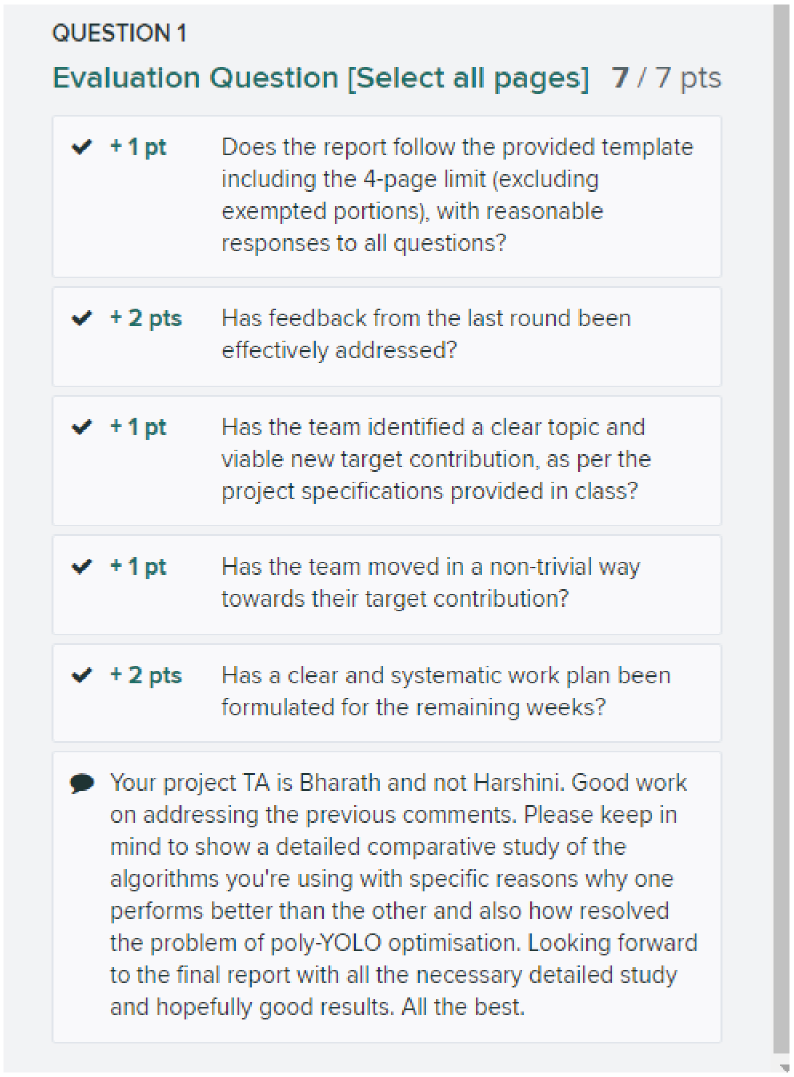
If your answer to the above question is yes, are there any other links to github / youtube / blog post / project website that you would like to publish alongside the report? If so, list them here.

- <github-link >: < https://github.com/chenrui-hu1/cis519.git>
  ...

(Exempted from page limit) **Work Report: This may look like your GANTT chart from the midway report, with more completed steps now. Okay to modify.** (Mark completed steps in green, as shown here. For convenience, you may split into two charts, one till Nov 8, and another for after Nov 8, placed one below the other.)

		Wk5				Wk6				Wk7				Wk8				Wk9			
		OCT																	NOV		
PERSON (S)	TASK (S)	S 3	M 4	W 6	Th 7	S 10	M 11	W 13	Th 14	S 17	M 18	W 20	Th 21	S 24	M 25	W 27	Th 28	S 31	M 1	W 3	Th 4
Name 1	Task 1		🟩																		
Name 2, Name 1	Task 2		🟩																		
Name 3	Task 3		🟩			🟩															
Name 3, Name 2	Task 4											🟩									
Name 2, Name 3	Task 5								🟩												
...	Task 6								🟩												
...	Task 7										🟦										
...	Task 8															🟩					
...	Task 9																		🟦		
...	Task 10																				

(Exempted from page limit) Attach your midway report here, as a series of screenshots from Gradescope, starting with a screenshot of your main evaluation tab, and then screenshots of each page, including pdf comments. This is similar to how you were required to attach screenshots of the proposal in your midway report.



QUESTION 1

Evaluation Question [Select all pages] 7 / 7 pts

✓ + 1 pt Does the report follow the provided template including the 4-page limit (excluding exempted portions), with reasonable responses to all questions?

✓ + 2 pts Has feedback from the last round been effectively addressed?

✓ + 1 pt Has the team identified a clear topic and viable new target contribution, as per the project specifications provided in class?

✓ + 1 pt Has the team moved in a non-trivial way towards their target contribution?

✓ + 2 pts Has a clear and systematic work plan been formulated for the remaining weeks?

Your project TA is Bharath and not Harshini. Good work on addressing the previous comments. Please keep in mind to show a detailed comparative study of the algorithms you're using with specific reasons why one performs better than the other and also how resolved the problem of poly-YOLO optimisation. Looking forward to the final report with all the necessary detailed study and hopefully good results. All the best.

YOLO to YOLO3 Algorithm analysis and Poly Yolo endeavor for optimization

Team: Chenrui Hu, Rong Fan, Xuanmin Zhu **Project Mentor TA: Harshini**

1) Introduction

The input of the system is the stream of images and output of the system is object detection which involve objects classifications and object locations in the image. Evaluation can be done from different perspectives,

- mAP (mean Average Precision), below will use mathematical equation to express the meaning of this:

1. First step: calculate Intersection Over Union (IOU)

$$IOU = \frac{area(B_P \cap B_{gt})}{area(B_P \cup B_{gt})}$$

   Where B_{gt} is the Ground Truth box of the objects in the image and B_P is the predicted box.

2. Definition of TP, FP, FN, TN

   True Positive (TP): IOU > $IOU_{threshold}$ (default 0.5), every unit Ground Truth only check once.

   False Positive (FP): IOU < $IOU_{threshold}$ (default 0.5), number of box satisfies this equation.

   False Negative (FN): no detected GT

   True Negative (TN): no use in mAP

3. Precision and Recall

   Precision = $\frac{TP}{TP+FP} = \frac{TP}{all\ detections}$, where all detections is the number of anchors set in system

   Recall = $\frac{TP}{TP+FN} = \frac{TP}{all\ ground\ truths}$, all ground truths is the number of GT.

4. According precision and recall to calculate the P-R curve and AP is the area under the curve and mAP is mean of all P-R curve area.

- FPS (frames per second) one advantage of yolo or poly-yolo is that the speed of dealing images is guaranteed and the speed is fast enough even to deal with images in the video.

- Yolo Loss function (too complex to explain, given by reference)

$$\begin{aligned} loss_{N1} = {} & \lambda_{box} \sum_{i=0}^{N1\,X\,N1} \sum_{j=0}^{3} 1_{ij}^{obj} [(t_x - t_x')^2 + (t_y - t_y')^2] \\ & + \lambda_{box} \sum_{i=0}^{N1\,X\,N1} \sum_{j=0}^{3} 1_{ij}^{obj} [(t_w - t_w')^2 + (t_h - t_h')^2] \\ & - \lambda_{obj} \sum_{i=0}^{N1\,X\,N1} \sum_{j=0}^{3} 1_{ij}^{obj} \log(c_{ij}) \\ & - \lambda_{nobj} \sum_{i=0}^{N1\,X\,N1} \sum_{j=0}^{3} 1_{ij}^{obj} \log(1 - c_{ij}) \end{aligned}$$

$$-\lambda_{class} \sum_{i=0}^{N1 X N1} 1_{ij}^{obj} \sum_{j=0}^{3} \sum_{c \in classes} [p'_{ij}(c) \log\left(p_{ij}(c)\right) + (1 - p'_{ij}(c) \log(1 - p_{ij}(c))]$$

$$LOSS = loss_{N1} + loss_{N2} + loss_{N3}$$

Where λ is the weight, 1_{ij}^{obj} is the function that when True Positive, it equals to 1; when false negative, it equals to 0.

Motivation: generally speaking, mAP defines the percentage of number of images correctly detected in unit time, FPS define the speed of image detection and loss function defines the accuracy of object location and classification. The plan right now for our project is first upload the pretrained poly- yolo model to the colab and add GridSearchCV pacakage to the model to adjust parameters inside like batch size in batch normalization layer, epoch size in pipeline to both decrease the validation error and increase converge speed. Main problem comes from coding like fix different package version, adjust initial code to combine our own GridSearchCV…etc.

2) How We Have Addressed Feedback From the Proposal Evaluations

The feedback from proposal is main about we need to clarify the reason why our creative work make sense and what is our data contribution.

The problem of using yolo3 to recognize images in dataset like coco is that labels are highly likely to be rewritten when applying this algorithm to the real-world data which is not that uniformly distributed as well as having similar size. To train a network with dataset more like the real-world data, we need to modify the last convolutional layer of the original darknet53 network. Although we have not realized this modification, we are working on that and start to test the performance to small changes and evaluate our modification method.

As for the latter problem, after comparing the distance between bbox for coco dataset and Simulator datasets, we decide to use Simulator as our dataset. This is because the uniformly distributed objects in coco dataset led to low label rewriting rate so that we cannot see the efficacy of using Poly-YOLO algorithms. In the Simulator datasets, bbox of objects are distributed randomly and some of the bbox are really close to each other. Based on this dataset, we implemented data augmentation and generated image batch that is 10 times larger than the original one.

3) Prior Work We are Closely Building From

A. Gitlab reference.
   Title: POLY-YOLO: HIGHER SPEED, MORE PRECISE DETECTION AND INSTANCE SEGMENTATION FOR YOLOV3.
   URL: [2005.13243] Poly-YOLO: higher speed, more precise detection and instance segmentation for YOLOv3 (arxiv.org).
   Code link: IRAFM AI / Poly-YOLO - GitLab https://gitlab.com/irafm-ai/poly-yolo

Contribution: Given complete code for poly-yolo using keras. Results of training on 3 different datasets (Coco, IDD, simulator, sythetic) are given, 3 different function model (poly- yolo-lite, poly-yolo, poly-yolo-polygon) are shown. What we do now is just to select poly-yolo-lite pretrained model(h5), add GridSearchCV to adjust some parameter.

4) What We are Contributing

Contributions in Algorithm:

We aimed to optimize the backbone of YOLO3 to decrease the bias on IoU (Intersection Over Union) based on images with almost same scale objects. So far, we have recurred Poly-YOLO algorithm. Although in the next milestone we had planned to focus on optimizing the backbone to decrease its demand on GPU, it seems like a tough work. This is because after considering the complexity and performance of current Poly-YOLO algorithm and do evaluation of the amount of work we need to complete to realize that optimization, we find out that it takes long time due to larger number of convolutional layers to be considered and huge amounts of parameters to be tune.

Contributions in Code:

Since the whole project will be written in Python, our creative work like doing data augmentation and probably optimize the network will become our contributions in code.

Contributions to Data:

In order to mitigate the effect of overfitting problem caused by small dataset, we use some bag of freebies like data augmentation and add attention module as bag of specials. These tools will improve the generalization ability of our model. Relating data augmentation tools includes rotate, shift, shear, zoom, flip .etc.

5) Detailed Description of Each Proposed Contribution, Progress Towards It, and Any Difficulties Encountered So Far

5.1 Methods

- For already existed model, the first thing we are going to do is to do model rebuilt, as poly-yolo model is really complex and advanced model, we have to format our own text file to record parameters insight of this model. For an instance, we have to create our own anchors.txt to record the size of anchor box according to k-mean algorithm provided by reference, save the prediction results and test error to folders and print out the fps of poly-yolo on dealing with stream of images.
  Secondly, we would like to adjust some parameters insight of model, for details, what we have already finished is to adjust the batch size of batch normalization layer to get fast converge speed and we find the batch size to 4 comfortable to the GPU of our own computer. For pipeline process, we test epoch size from 5 to 15, finally finding validation loss would have lowest value about 11 at the epoch size equals to 5.
  For future work, we would like to consider other parameter like the number of anchor box, the number vertex of polygon, the number of classification labels…etc. However, each influence of variation is oblique and the meaning for changing it also needs to be discussed.

- For data contribution, we attempt to do data augmentation, we would like to try to flip images horizontally and vertically, rotating it by 180 degrees, scaled outward or inward, Crop and add some Gaussian Noise…etc. Additionally, we would like to try to use Conditional GANs to the rescue to make deep change to the image datasets. Finally, data augmentation and generated image batch that is 10 times larger than the original one, and we would confirm whether it would give us lower loss in final results.
- For algorithm contribution, the main idea to optimize the algorithm is based on same concept from the generation of poly-yolo. Yolov3 has nearly perfect performance on Coco datasets, however, dataset itself has own distribution, so in practice, if dataset distribution changes, the final training effect would change correspondingly. As a result, we would change the structure of neural network to adapt to different distributions of datasets.
  For details, Coco datasets is close to M ~ U(0, r), where r is the resolution ratio of the image and a uniform distribution between bounds given by 0 and r, this is to say The size distribution of the object is a uniform distribution with a boundary of 0 to r, then the K-mean method to format anchor box is reasonable, However, for poly-yolo, the dataset it uses satisfies that M ~ N (0.5r, r), where N (0.5r, r) is normal distribution with mean μ = 0.5r and standard deviation σ^2 = r is a more realistic case, which causes that most of the boxes will be captured by the middle output layer (for the medium size) and the two other layers will be underused.
  Based on yolov3 and poly-yolo, our strategy is to find a dataset which is different from yolov3(Coco) and poly_yolo dataset, for instance, we can find the images where object in images is binomial distributions and then adjust the K-mean algorithm for having just two large ranges and one threshold values to split. (Compared with yolov3 and poly-yolo where v3 has no split and no threshold while poly-yolo has 3 splits and 2 threshold values).
  For output layer, yolov3 has 3 output layers for big, medium, and small objects while poly-yolo has only one output layer because the size of object in dataset is almost the same, if we could find a dataset that the objects in image has two different sizes and has binomial distributions, then, we can has two output layers corresponding and result would be perfect.

5.2 Experiments and Results (same as section one, be short)

- Check mAP values
- Check Loss function (if neural network adjust, loss function change corresponds, however, complexity is very high, may not finish)
- Check fps

6) Risk Mitigation Plan

Considering the difficulty to revise the current Poly-YOLO network, our goal is mainly about recurring the current algorithm and applying it to more real-world data with data preprocessing like data augmentation as well as analyze the test result by comparing the performance with and without data augmentation. Based on this, we will properly tune the hyperparameters to get a better performance and analysis reason of such modification. If we have extra time, we will consider optimizing the whole algorithm. Then if the optimization fails, we will also systematically analyze the reason and give relating evaluation results.

(Exempted from page limit) Supplementary Materials if any (but not guaranteed to be considered during evaluation):