Google Open Images 2019 - Object Detection solution

Adam Jochna

First of all I want to thank Google for hosting this competition. It was great fun to compete with other kagglers and push myself to the limit. Let me first describe how I worked out my solution. Three months ago I didn't have much experience with kaggle, besides basic things like Digit Recognizer and Scikit-learn stuff I have never used NN in my life.I have set myself a goal to do my best at this particular competition. I have spend first days by searching current architectures for NN object detection. Oh boy remembering this names was hard not to mention architectures inside those detectors. In many places I've read that FRCNN was current state of the art. So I decided to download ZFTurbo implementation and see how it works. Unfortunately at that time my GPU was Radeon HD 7770, so I tried setting up virtual machine with GCP credits. I used Nvidia V100 and 300$ disappeared quickly. I decided to buy my own GPU. So now I have GTX 1080 ti. Next 2 weeks I spend on seeing how things work and making first model and submission. This implementation was great, but not prepared for old cpu like mine. So I had to optimize things to run multi-threaded. This big dataset was not something I could download straight to my 250GB HDD. So I decided to download images in fly. Also I have 8Gb of RAM so I couldn't load whole dataset info to my script during training.

1.FRCNN resnet50

Those are the things I changed:
1.Downloading image from web preparing data for training based on image annotations, sampling images based on classes sampling history,training on GPU happened simultaneously this reduced training time on image from 6s to 0.8s
2.Also I didn't like the sampler, original implementation would load all 600 classes and iterate over them, so next image would be sampled if it contains one class, next image next class. This was not effective because lot of classes was 'common' you don't have to sample image if it contains class 'Person' because if you sample 'rare' class this image very likely also contains 'Person'. So I decided to sample as follows:
-make list of all classes occurances initialize with 0
-take class with lowest number of occurrences in training Images already used(initial all 0)
-sample image with this class and for all bounding boxes in image add occurance to list
-repeat
This makes for example common class 'Person' never sampled by sampler although present in sampled images.Looking back I would change one thing. I would split my initial model to 3 models based on classes distribution in training data. First model would train only on classes which have <200 instances. Second model only on classes which have 200-1000 instances. Third >1000.
This way training all the data I wouldn't overfit so badly on classes wich have <200 instances in training data. Because this sample technique chooses minimal class it can happen that class with 50 images can be sampled 2000 times like class with 50000 images. This way

you cannot control loss because on some classes you overfit badly and on some classes you make progress so loss cannot be trusted.
3.Third thing I changed was Softmax to Cross Entropy for each logit separately in the classification head.Softmax is function which has property that sum of all logits = 1. This implies that classes have to be exclusive which is not the case in this competition. For example Car Vehicle. This confuses model if it sees Car it tries to predict maximum value for Car Vehicle and Land Vehicle but it is constrained by sum = 1 so for example it gives 0.33 for each of this classes and 0 to rest sigmoid cross-entropy is not constrained so Car and Vehicle are evaluated separately. Also I modified expanding parents before predictions in ground truth boxes.If there is a Car there is also a Vehicle that model should predict.

2.Inception_resnet_v2

So, at this moment my solution gave me 0.37634mAP public LB. I saw long list of people with exact same score 0.42496 public LB. I thought it came from some kernel. I checked and It came from this model https://tfhub.dev/google/faster_rcnn/openimages_v4/inception_resnet_v2/1 , also one thing I noticed from my experiments with first model on validation mAP, setting score threshold low increases mAP significantly. But model from tfhub was frozen with 300 boxes (if i remember correctly) so I unfroze it and set limit to 2000 boxes this gave me 0.45250mAP 0.027mAP, also I noticed that classes which are not leafs in hierarchy tree have low mAP so I tried expanding classes which are higher in hierarchy for every box with score>0.02 this gave me 0.46226mAP boost of 0.01mAP at this moment i started to think of an ensemble.

3.YOLO v3

The third model I have used was YOLO v3 I downloaded version trained for Open Images Dataset, set threshold very low because I noticed that YOLO gives much smaller score for boxes.
It has to do with the way YOLO calculates the score, in this case score is conditional probability of class in box given there is an object in box where in other models it is probability of class in box

4.Ensemble

Now I have predictions from 3 models I tried several ensembling techniques including changing probability distributions of models. I noticed that those distributions were different, so when applying non maximum suppression model with higher score can make lower score box disappear although this lower score box might have better fit. So after trying several increasing functions mapping [0,1]->[0,1] I found 3 functions that work best and at that moment my best ensemble gave me 0.47595 public LB.I wasn't satisfied with NMS I knew that there are cases where there are overlapping boxes from the same class and both are correct. I tried soft-NMS with gaussian and linear function but It increased my score insignificantly. So I abandoned this idea but I was constantly thinking about better ensembling technique, especially that I've read on kaggle that someone used some heuristic to make ensemble other than NMS and soft-NMS.

5.Plateau

At this moment I tried multiple other things which none of worked.I knew that 3 models is too few to achieve something in this competition so I tried using YOLO9000 which was trained for classification and object detection on Imagenet, this was never done before so I thought I can benefit from different dataset. I made for most labels by hand (7h of searching text) map from 9k YOLO9000 labels to 500 OID labels. I've got big hopes for this model but got only like 0.02mAP. Still don't know why.

6.150cls FRCNN resnet50

At this moment I noticed that some classes have very low AP, I thought that I can do better with simple FRCNN so I trained for 5 days only on this 150 worst classes. This model was best for some classes even better than Inception and YOLO.

7.Searching kaggle discussion is important

Multiple times when I had no idea where to improve I've read all discussions(literally all) on this competition discussion tab. One time I stumbled upon discussion about human parts. This dataset is inconsistent and not all classes instances are always labeled. But in this thread there was list of all images where all human body parts are annotated so I trained third model only on this images without any sampler just randomly selecting images from this list. mAP on human parts improved but deadline was close and I didn't have time to finish it.

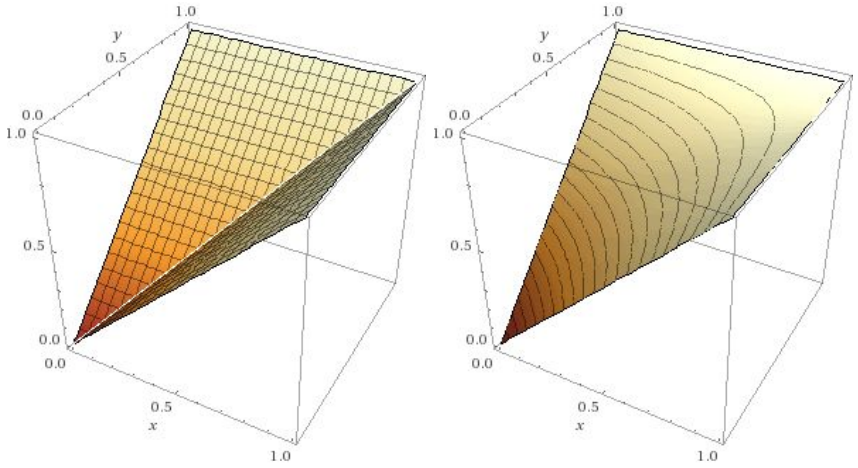
8. Last year solution

I saw ZFTurbo solution on his github and tried using it in my ensemble I predicted on validation set and saw that on some classes it is better and on some it is worse than my ensemble. I predicted on FRCNN resnet 101 and 152 backbones and made 3 predictions csvs: resnet101,resnet152 and one by merging this two using ZFTurbo script.Worth noticing is that ZFTurbo solution included only leafs in hierarchy tree so without parent classes.

9.Ensemble idea

While searching web for ideas is saw post on medium about ensembling technique using graphs and cliques it contained only pseudo code so I quickly wrote implementation.
https://medium.com/synapse-blog/tech-deep-dive-object-detection-ensembles-as-graph-cliques-a7f7d33b5477
I really liked idea that you can use multiple models and if some model produces overlapping boxes clique will be constructed with the box with better IOU score, while preserving the overlapped box. I was sceptical about this score function for merged boxes f(p1,p2)=1-(1-p1)(1-p2) but after while i noticed that if one probability is big second is very small it acts as f(p1,p2) ~= max(p1,p2) also if two probabilities are small it acts as f(p1,p2) ~= p1+p2 and when two probabilities are big it has property
1>= f(p1,p2) >> max(p1,p2) all of this properties can be seen when expanding brackets f(p1,p2)= p1 + p2 - p1*p2
but I had more than 2 models so if clique has more than 2 boxes i just take all scores and set new score as ScoreNew=1-(1-box1_Score)*(1-box2_Score)*...*(1-boxN_Score) this function has similar desired properties as f(p1,p2) just with more variables the most important thing was that small probabilities are added when making new score and big probabilities get even

bigger.Below you can see two functions f(x,y)=max(x,y) and f(x,y)=1-(1-x)(1-y) these properties can bee seen comparing graphs.



This ensemble solution gave me 0.51379 public LB. Taking the best model for each class including clique ensemble as separate model gives 0.51568 public LB script for ensembling is public here
https://github.com/AdamJochna/EnsembleCliques/blob/master/clique_ensemble_github.py
here a is summary of performance of my models evaluated on validation data:

	evYOLOv3	evFRCNN	evInception	evalFRCNN150cls	evalZFturbo152	evalZFturbo101	evalZFturboMerg	evalClique_0_5_1600
mean	0.450036	0.541005	0.585250	0.353207	0.555808	0.537605	0.569271	0.664244
25%	0.263620	0.369057	0.431222	0.052908	0.360436	0.351705	0.381415	0.534642
50%	0.478589	0.564903	0.601785	0.381233	0.620959	0.601222	0.647032	0.697603
75%	0.644308	0.726529	0.746722	0.553145	0.805396	0.772255	0.810806	0.828206

Ideas for next competitions:
-replacing resnet50 with better backbone, at the beginning when I was learning I didn't know the difference between different architectures, now I know that resnet50 was bad idea I probably would choose NASNet