Improve KITTI Cyclist Detection with ResNeSt and NAS-FPN based Faster R-CNN

Zhen Li
lizhenpi@stanford.edu

Yijing Bai
byj@stanford.edu

Linkun Chen
lkchen@stanford.edu

Abstract

Object detection is one of the most important problems in Autonomous Driving. Recognizing smaller targets like pedestrians and cyclist are especially important as they typically don't perform as well as recognizing cars in modern technologies. With recent backbone network improvement published based on Faster R-CNN, including ResNeSt and NAS-FPN, it opens up a new opportunity to improve the 2D object detections to a higher level. In this paper we focus on improving the cyclists class classification quality on KITTI dataset by applying the state-of-the-art ResNeSt, NAS-FPN on top of the traditional Faster R-CNN structure, and doing model pretraining using Tsinghua-Daimler Cyclist Detection Dataset. With all the improvements, We improved the moderate difficulty cyclist detection accuracy from 45.1% to 63.8% without decreasing the accuracy of other classes. Our work is hosted online [10].

1. Introduction

Autonomous driving has been a focus of attention in recent years, and has the great potential in changing people's lives dramatically. However, many state-of-the-art systems are largely optimized for detection accuracy on vehicles , while the detection accuracy on cyclist and other smaller targets are relatively low. There could be several reasons, including generally fewer training data exist for cyclists in real driving environment, and the traditional Faster R-CNN structure doesn't perform well on multi-scale detection [9].

In this project we focus on improving the object detection accuracy of cyclist on KITTI benchmark [12]. The input to our algorithm is a 2D road image that is captured on the road by autonomous car camera, we use a improved Faster R-CNN [13] to output a list of predicted object appeared in the image, including the positions of the targets and their predicted classification class. We used ResNet based Faster R-CNN as baseline, applied the recently published ResNeSt [19] with FPN [9] on KITTI dataset 2D object detection problem, and further improve the cyclists detection quality by applying NAS-FPN [3], pretraining with Tsinghua-Daimler Cyclist Detection Dataset [8] and fine tuning.

2. Related Work

Deep ConvNet object detections Deep ConvNet showed dramatic improvements in object detection tasks in recent years after the work of R-CNN [6], Fast R-CNN [4] and Faster R-CNN [13]. Especially the work of Faster R-CNN proposed region proposal network which enables end-to-end training, achieves a great trade-off between accuracy and speed. Due to the requirements for low decision latency in autonomous driving task, Faster R-CNN has been widely used for driving time object detection [9]. Recently ResNeSt improves the Backbone ResNet structure in Faster R-CNN by introducing a Split-Attention block, which achieves state-of-the-art object detection performance on MS-COCO [19].

Multi-scale detection Recognizing objects of different scales has been a challenge in the Faster R-CNN structure since the bounding boxes are only pooled from the last convolutional layer rather than being wrapped into a canonical size. The last convolutional layer may not contain enough information to make decision when the boxes are too small [9]. To solve this problem, many different structures are proposed to pool Region of Interests from different convolutional layers, including Scale Dependent Pooling [17], Context-aware RoI pooling [7], SSD [11] and FPN (Feature Pyramid Networks) [9]. Among those structures, FPN showed the best performance on various tasks. Furthermore, NAS-FPN [3] was introduced recently as a AutoML searched FPN structure that works similarly with FPN, but better. It consists of a combination of top-down and bottom-up connections to fuse features across scales, which is the state-of-the-art structure for multi-scale detection.

Multi-scale detection for Autonomous Driving There are many efforts specifically focusing on improve the cyclist or other smaller scale object detection in autonomous driving tasks. From network structure perspective, SINet [7], SDP [17] and MS-CNN [2] focused on extract multi-layers features into RoI pooling, we will use FPN and NAS-FPN to solve this specific problem in a similar way. From data per-

spective, Tsinghua-Daimler Cyclist Detection Benchmark [8] is introduced to provide more labeled cyclist data in autonomous driving situation.

3. Methods

In this project we improve the cyclist detection accuracy on KITTI dataset in 3 ways, 1. We improve the backbone ResNet network by replace ResNet with ResNeSt network structure. 2.We implement NAS-FPN in Detectron2, replacing FPN to improve the multi-scale detection. 3. We pretrain Faster R-CNN on Tsinghua-Daimler cyclist dataset. Our improved architecture is shown in Figure 1.

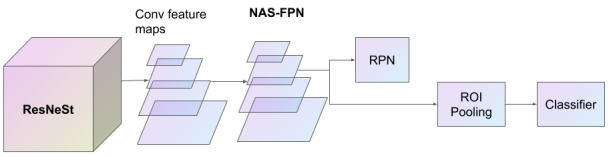

Figure 1. Architecture of our methods

3.1. ResNeSt as Backbone network

ResNeSt [19] is a newly published backbone network structure that improves the quality of ResNet on various classification and detection tasks, including on MS-COCO detection task.

The ResNet originally designed for image classification, which has limited receptive field size. Besides, it lacks cross-channel interactions. ResNeSt proposes a versatile network with cross-channel representations without needs of additional computation than existing ResNet-variants. The large scale benchmarks the author conducted shows the models with ResNeSt backbone achieves state of the art performance on several tasks like images classification, object detection, instance segmentation and semantic segmentation.

It introduces a Split-Attention Block, which divide the feature map into several splits, and then use a weighted combination of the splits to produce the output feature representation for each group. The weights are chosen based on global contextual information. The structure of Split-attention block is shown in Figure 2. Based on this block, the feature is divided into several groups, the number is given by a hyperparameter K. The groups are called cardinal groups. They use $\hat{U}^k = \sum_{j=R(k-1)+1}^{Rk} U_j$ denote the k-th cardinal group, where $\hat{U}^k \in R^{H \times W \times C/K}$ for $k \in 1, 2, ...K$. H, W, C are block output width, height, channel respectively. For global contextual information, it's calculated by global average pooling. $s_c^k = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} \hat{U}_c^k(i, j)$ is for c-th component. The weighted fusion of the group is using channel-wise soft attention, please checkout the original paper for details. In the ResNeSt block, it concatenates the cardinal groups along the channel dimension. Figure 3 shows the structure of a ResNeSt block.

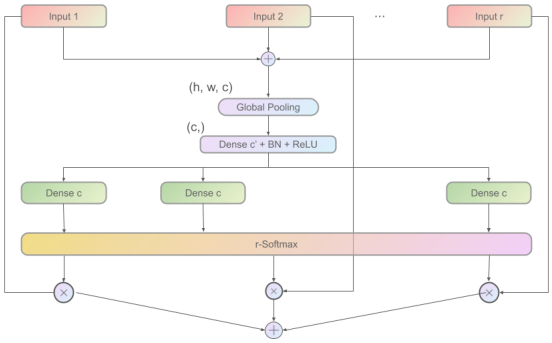

Figure 2. Split-Attention Block

We apply the ResNeSt Faster R-CNN implementation based on detection2 provided by the ResNeSt authors to KITTI dataset, and measure the improvement on top of ResNet.

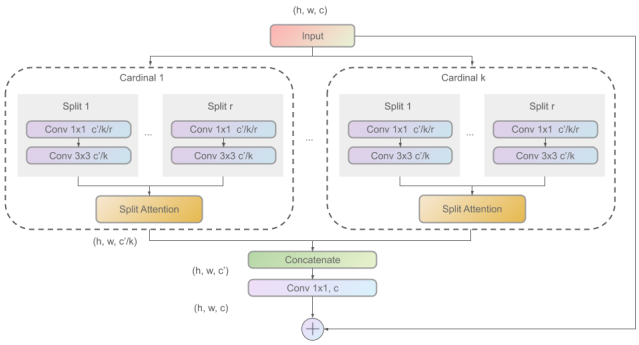

Figure 3. ResNeSt Block

3.2. NAS-FPN to replace FPN

One of the reasons that cyclists or other smaller scale objects detection doesn't perform as well as car class in autonomous driving is that the traditional Faster R-CNN RoI pooling only extract features from the last feature layer, while it loses the features at the earlier layers with higher resolution. Feature Pyramid Network (FPN) [9] solves this problem by combining features from consecutive layers, thus semantics in lower resolution can combine with details in higher resolution. In practice, FPN works very well

capturing the rich semantic meaning at different convolutional layers for multi-scale detection, and has been used widely as a important component in object detection tasks [3]. As FPN is already implemented as part of the option in Detectron2 object detection framework, we tried applying FPN on top of the traditional Faster R-CNN to measure the improvement of cyclist detection

On top of FPN, Google AutoML team recently published NAS-FPN [3], which is a neural network searched structure that further improves the performance of FPN. With the goal of improving the cross-scale connections in FPN that only connect consecutive layers, connections in NAS-FPN are discovered by Neural Architecture Search using training accuracy as reward signal. As a straightforward architecture replacement, it outperforms FPN and achieves state-of-the-art results on many object detection tasks including MS-COCO.

With the assumption that using NAS-FPN will also helps multi-scale detections on KITTI dataset, we replace the FPN with NAS-FPN in our model. In their original paper [3] they proposed 5 different NAS-FPN architectures that was proven to have progressively higher reward during training. We pick the specific NAS-FPN architecture that gave the highest AP among experiments, and reproduce the implementation in pytorch Detectron2 framework based on existing tensorflow implementation [18]. The architecture of NAS-FPN we use is in Figure 4.

3.3. Data augmentation and Pretraining

Lack of cyclist training data in KITTI dataset is another reason for relatively lower cyclist detection accuracy. On the KITTI object detection benchmark rank board, it is common for car detection accuracy to get higher than 90%, but for cyclists accuracy it is common to see number around 50% and 60%. After looking into the KITTI dataset, we notice the imbalance number of labeled data. Specifically, there are 28742 labeled cars, but only 1627 cyclists labels in the KITTI Vision Benchmark Suite [12]. Given the extremely small number of labeled cyclist data, it will be much harder for model to learn a high accuracy cyclist detection.

We decide to feed model more labeled data of cyclist to better learn the cyclist detection. We found the Tsinghua-Daimler cyclists Benchmark [8] to be very promising, which consists of 9741 labeled images and 22161 annotated cyclists also captured from a driving car's camera. Since this dataset only have labeled data of cyclist but didn't label cars, pedestrians and other classes, we cannot simply merge it with KITTI dataset. Thus we decided to pretrain our model on Tsinghua-Daimler cyclist dataset, and use the pre-trained weight as the initialization weights for training on KITTI dataset. Experiments showed that pretraining obviously speedup the Kitti cyclist training, and also increases the cyclist accuracy on the trained model.

Other than pretraining, we also did data augmentation by adding horizontally flipped the images & labels, and resized images & labels in training using Detectron2 framework.

4. Dataset and Features

We are using the object detection benchmark from The KITTI Vision Benchmark Suite [12]. KITTI dataset is a broadly used open-source dataset to evaluate visual algorithms on a driving scene considered representative for autonomous driving. it contains 7481 images for training and 7518 images for testing, comprising a total of 80,256 labeled objects. In this project, We only use the 2D images input data, and won't use the depth data from Laserscanner like Velodyne point clouds. A example image can be seen at Figure 11.

In addition, we used The Tsinghua-Daimler cyclists Benchmark[8] for cyclist pre-training, which has 9741 training images in it with annotations for cyclist. This dataset is also captured on a driving scene, but most of the images are captured inside Tsinghua University, which have much fewer cars and much more pedestrians and cyclists comparing to Kitti dataset. A example image can be seen at Figure 5.

4.1. Dataset format

All images in KITTI benmark are 8-bit color RGB and saved as png format, in the shape of 1224 * 370. There are 9 classes in total, *Car, Pedestrian, Cyclist, Truck, Van, Tram, Person sitting, Misc* and *Dont care*. In the training set, the specific label distributions are:

Label	Count
Pedestrian	4487
Truck	1094
Car	28742
Cyclists	1627
Dontcare	11295
Misc	973
Van	2914
Tram	511
Person sitting	222

There are several labels on each image, labels signals includes *class type, truncated score, occluded state, 2D bounding box, 3D object dimensions, 3D object location* and *rotation angle*.

The images in The Tsinghua-Daimler cyclists Benchmark are 1024 * 2048 8-bit color LDR format. In train split, there are 9741 images with annotations all for cyclist. The label contains *class type (cyclist only), 2D bounding box and occluded state*.

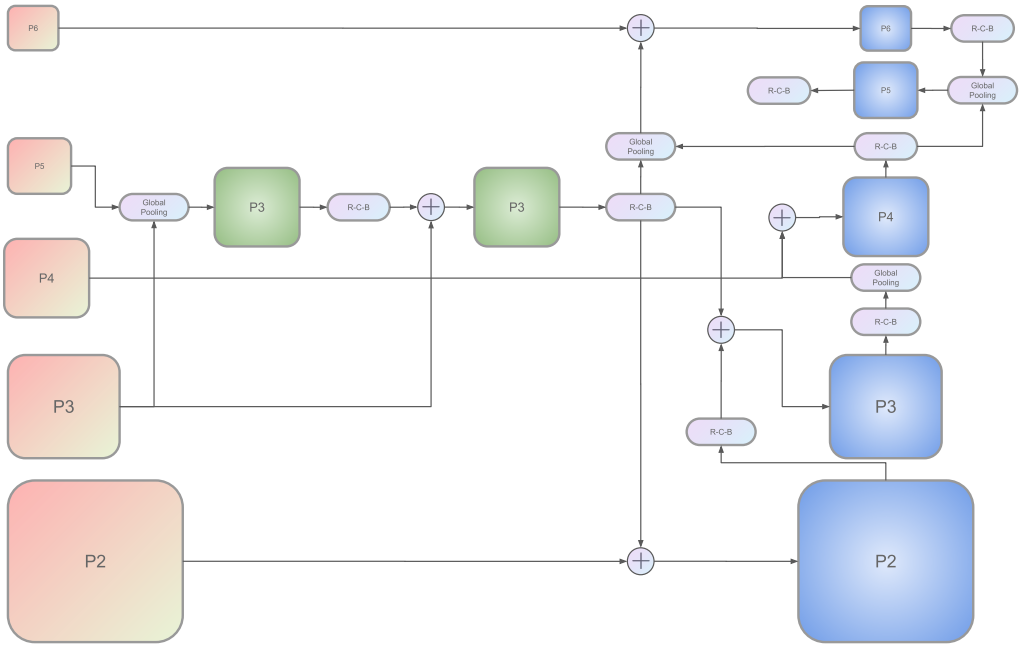

Figure 4. Architecture of NAS-FPN, R-C-B denotes ReLU-Conv-BatchNorm layers.

Figure 5. Sample Image of Tsinhua-Daimler dataset

4.2. Metrics

We evaluate the object detection performance using the well established average precision (AP) metric as defined by KITTI [12]. Detections are iteratively assigned to ground truth labels starting with the largest overlap, measured by bounding box intersection over union. In KITTI setup it require true positives to overlap by more than 50% and count multiple detections of the same object as false positives. The metric is divided into *easy, moderate* and *hard* cases based on the height of the 2D bounding box of object instances, occlusion and truncation level [12].

4.3. Data conversion

Detectron2 doesn't support the KITTI data format and Tsinghua-daimler dataset format by nature. We used Visual Object Dataset converter [15] to convert the KITTI to Pascal 2012 format, and further implemented conversion from Tsinghua-daimler data format to Pascal 2012 format and KITTI format. We used the Pascal data-set configurations in Detectron2 [16] for training, and KITTI data format for eval to get the KITTI required evaluation numbers.

4.4. Validation

Detectron2 doesn't support validation statistics [14], we need to implement our own validation loss function based on Detectron2. Specifically we added a eval loss hook to the trainer, so that trainer will run inference on the entire validation set every certain iterations and report validation loss to prevent overfitting.

5. Experiments

5.1. Experiments setup

We run the experiments on GCP with one GPU (Nvidia Tesla T4). We split the KITTI training dataset into 5415 training images and 2065 validation images using the same split method proposed by [5], ensuring that images from the same sequence are not present in both training and validation sets. Due to KITTI platform has stopped accepting test submissions, we further split the validation set into 1032 validation images and 1022 test images for cross-validation.

We also did data augmentation by adding horizontally flipped the images & labels, and resized images & labels into training using Detectron2 framework. And we also only train for 3 major classes *car*, *cyclist* and *pedestrian* as these 3 classes are the major classes reported in the KITTI results, and we found that model started to optimize for

other unimportant classes like *don't care* and sacrificing major classes in the later iterations.

5.2. Hyperparameters

We inherited many hyperparameters from the default settings in Detectron2 config with the assumption that most of them should be a good default choice for object detection. For optimizer we used SGD with momentum as it is the default setup for Detectron2 and was used in the state-of-the-art results from ResNeSt object detection [19]. The major hyperparameters we tuned for are learning rate, batch size, learning rate decay factor γ, learning rate decay timing and number of iterations.

To tune the hyperparameters, we first started with small dataset with 100 pictures. For learning rate, we tried the learning rate at different order of magnitudes, the order of 0.1 makes the training gradient explode. Then we tried 0.02 and its close learning rate 0.01, 0.03. As most of the configs in Detection2 sets the learning rate 0.02. We run 100 iterations with 0.01, 0.02, 0.03 to check the loss curves shown in figure 6. 0.03 lr is too high as after several iterations, it makes the loss increased. Hence we chosen 0.02 as learning rate. For learningrate decay factor γ, we notice around 10k iterations, the loss curve start to flat, hence we choose it as the first point to decay the learning rate.

The batch size is mainly depends on the GPU memory. We use the maximum batch size that our GPU can train, with NAS-FPN, the batch size is 4. With FPN, the batch size is 8.

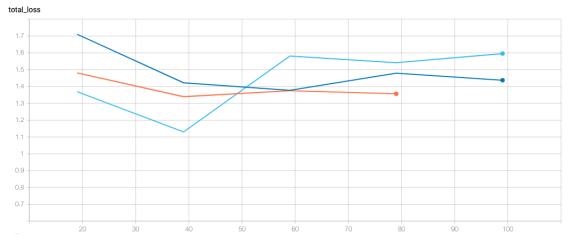

Figure 6. Loss curve with 0.01(dark blue), 0.02(orange), 0.03(light blue).

The parameters used in our experiments are showed as follows:

Parameter	Value
optimizer	sgd
momentum	0.9
batch size	4/8
learning rate	0.02
γ	0.2
lr decay steps	(10000, 16000)

5.3. Weights Initialization

We used the existing pretrained weights on ImageNet as the initialization weights for our Faster R-CNN. Specifically for all the ResNet based model, we used the ImageNetPretrained MSRA model provided by Detectron2. For all ResNeSt based model, we used the the pretrained Faster R-CNN weights provided by ResNeSt author. For all the experiments using Tsinghua-Daimler dataset pretrained model, we first train a model on Tsinghua-Daimler dataset, and chose the weights that performs best on test set to be the initilization weights for training on KITTI dataset.

5.4. ResNeSt vs ResNet

We used ResNeSt as our backbone convolutional feature extraction network for all following experiments, and we did experiments measuring ResNeSt performs comparing to ResNet performance on KITTI dataset. The results are shown in Table 1 and 2.

		Mod.	Easy	Hard
ResNeSt50	car	85.49	92.62	72.06
	cyclists	**52.48**	64.45	50.82
	pedestrian	67.58	79.60	62.90
ResNet50	car	84.90	94.97	71.26
	cyclists	**45.06**	67.38	43.15
	pedestrian	70.94	84.41	65.77

Table 1. ResNeSt50 comparing to ResNet50.

		Mod.	Easy	Hard
ResNeSt101	car	88.13	98.03	72.90
	cyclists	**53.41**	66.18	51.65
	pedestrian	72.54	85.77	67.26
ResNet101	car	85.65	94.00	72.26
	cyclists	**40.45**	60.17	39.86
	pedestrian	70.45	84.22	65.65

Table 2. ResNeSt101 comparing to ResNet101.

We can see that, in both 50 layers and 101 layers backbone network, ResNeSt outperforms ResNet by 1-3% on car, pedestrian accuracy, and 5% on cyclist accuracy. This matches our expectation after seeing the ResNeSt performance increased reported on MS-COCO [19]. We think this improvement is also due to the Split-Attention mechanism in ResNeSt, since it's better supporting the cross-channel interactions.

5.5. NAS-FPN vs FPN vs Single Feature Map

We compared the performance of using different ways to extract features for Region Proposal Network, including the single feature map used in original Faster R-CNN, FPN (Feature Pyramid Network) and NAS-FPN. As shown in Table 3, FPN outperforms single feature map (C4 in the table) on both car detection and cyclist detection. This matches

our expectation as FPN has been proved to better resolve the multi-scale detection problem.

		Mod.	Easy	Hard
FPN	car	84.90	94.97	71.26
	cyclists	**45.06**	67.38	43.15
	pedestrian	70.94	84.41	65.77
C4	car	82.44	92.15	69.67
	cyclists	**43.03**	67.64	42.31
	pedestrian	69.93	82.72	64.65

Table 3. KITTI results ResNet50 + C4 and ResNet50 + FPN.

We also compared the performance of NAS-FPN and FPN. We ran two experiments on ResNet101 with FPN and NAS-PFN, the result is in table 4. NAS-FPN improved 2.58% cyclist detection accuracy on ResNet101. Besides, as can be seen in Figure 7, accuracy on FPN increases faster due to the pretraining weights provided by [19]. We notice that NAS-FPN converge slower comparing to FPN, and it's likely because the weights this NAS-FPN are newly implemented and trained from scratch, while the weights for FPN are inherited from a trained model. However, NAS-FPN reached a higher accuracy eventually on cyclist that none of our training using FPN can outperform, this matches the observation in [3] that NAS-FPN outperforms FPN in many detection tasks with the right hyperparameters.

		Mod.	Easy	Hard
NAS-FPN	car	82.36	94.09	68.85
	cyclists	**43.03**	67.64	42.31
	pedestrian	69.93	82.72	64.65
FPN	car	83.53	92.56	69.69
	cyclists	**40.45**	60.17	39.86
	pedestrian	70.45	84.22	65.65

Table 4. KITTI results ResNet101 + FPN and ResNet101 + NAS-FPN.

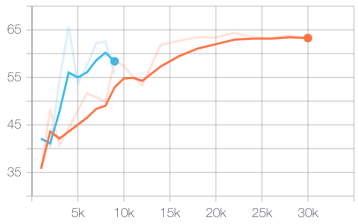

Figure 7. Cyclist AP curve for on KITTI validation set using ResNeSt101 with NAS-FPN (orange) and FPN (blue).

5.6. Pretraining

We pretrained the model on Tsinghua-Daimler cyclist dataset use the same training setup, and use the trained weights as the initialization weights for training on KITTI dataset to understand whether pretraining can transfer the learnt cyclist detection from Tsinghua-Daimler dataset to KITTI. For performance on Tsinghua-Daimler Dataset we can see from Table 6 that our model achieved 77% AP on cyclist detection.

		Mod.	Easy	Hard
NAS-FPN	car	86.31	94.24	71.22
	cyclists	**63.28**	74.53	61.44
	pedestrian	74.54	86.35	69.75
FPN	car	87.31	95.69	74.01
	cyclists	**58.38**	75.16	56.22
	pedestrian	72.48	85.22	67.58

Table 5. KITTI results ResNeSt101 + NAS-FPN + Pretrain vs ResNeSt101 + FPN + Pretrain.

	Mod.	Easy	Hard
ResNeSt + NAS-FPN	**77.14**	86.86	74.73
ResNeSt + FPN	72.83	82.67	70.69

Table 6. Results on Tsinghua-Daimler cyclist dataset

We compared model with and without pretrained weights on ResNeSt + FPN and ResNeSt + NAS-FPN. As we see from Figure 8, Table 7 and Table 8, model with pretraining outperforms model without pretraining by more than 10% on cyclist AP after converge, which demonstrates that the pretraining weights on Tsinghua-Daimler dataset are successfully improving the cyclist detection in KITTI dataset. Also from Figure 8 we can see that the model without pretraining started with a very low (10%) cyclist accuracy, while the pretrained model can get a 40% accuracy in the first few iterations. This is also a evidence that the pretraining is helping the training. We observed similar patterns in all 3 rounds of experiments that we conducted.

Another factor we notice is that, either with or without pretraining, the accuracy for cyclist will stop increasing or even start decreasing in the very later iterations while the car accuracy keep increasing. We suspect that's because of too few cyclist labeled exists in KITTI dataset, thus learning better cars detections can better reduce the total loss comparing to keep high accuracy cyclist detections. We think mix the Tsinghua-Daimler dataset together with KITTI dataset will likely resolve this issue, the idea will be discussed in Future Work section.

5.7. Stacked NAS-FPN

Hinted by [3], we also experimented with 3 and 5 stacked NAS-FPN with ResNeSt, since the output of NAS-FPN does not change the original resolution of each layer. Stacking multiple NAS-FPN should be able to help improving the accuracy according to [3]. But we found they are harder

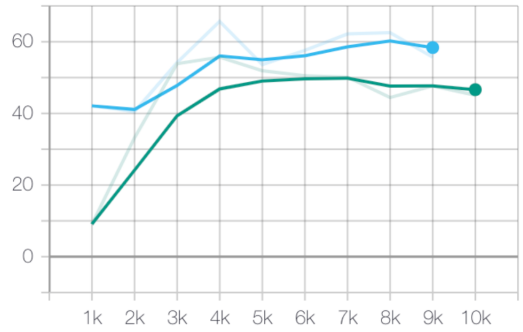

Figure 8. Cyclist AP curve for on KITTI validation set using ResNeSt + FPN with pretraining (blue) and without pretraining (green).

	FPN	**Mod.**	Easy	Hard
pretrain	car	87.31	95.69	74.01
	cyclists	**58.38**	75.16	56.22
	pedestrian	72.48	85.22	67.58
no-pretrain	car	88.13	98.03	72.90
	cyclists	**53.41**	66.18	51.65
	pedestrian	72.54	85.77	67.26

Table 7. KITTI results using ResNeSt + FPN, the cyclist performance of pretrained model outperforms no-pretrain by 5%.

	NAS-FPN	**Mod.**	Easy	Hard
pretrain	car	86.31	94.24	71.22
	cyclists	**63.28**	74.53	61.44
	pedestrian	74.54	86.35	69.75
no-pretrain	car	84.19	91.76	69.91
	cyclists	**50.08**	68.48	48.22
	pedestrian	72.25	84.35	67.54

Table 8. KITTI results using ResNeSt + NAS-FPN, we can see that pretrained model outperforms no-pretrain by 13%.

to train comparing to single layer NAS-FPN. In Figure 9, the loss curve of 3 and 5 stacked NAS-FPN(In green) drops slower than the loss curve of single NAS-FPN(in grey). The result also does not exceed the single layer NAS-FPN.

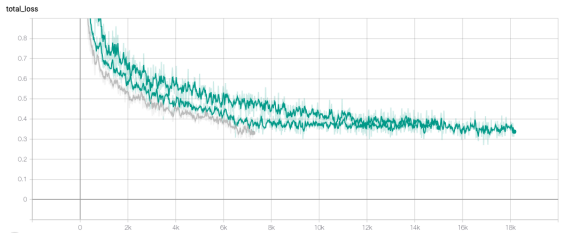

Figure 9. 3 and 5 stacked NAS-FPN compared to single NAS-FPN.

Given constrained resources, also due to no given weights to use, we didn't spend much time fine tuning ResNeST with multi-stack NAS-FPN.

6. Conclusion

In this project we combined ResNeSt, NAS-FPN and Tsinghua-Daimler cyclist dataset pretraining to improve the cyclist detection accuracy on KITTI dataset. The model with all technology applied improved the cyclist detection accuracy from our baseline 45% to 63% without decreasing the accuracy of other classes. A example of predicted image can be seen in Figure 11. As we can see from Figure 10, the biggest gain comes from:

Using Tsinghua-Daimler Cyclist dataset for pretraining gives 5% improvement. This demonstrated that when the dataset is imbalanced, it is helpful to pretrain the model with more labeled data for the class that has relatively fewer labels. The transfer learning not only speeds up the training for that class, but also improves the final accuracy. One observation we found is that with large number iterations model will still starting optimizing for the class that dominates the dataset and not optimizing too much for the minority class, so the max iteration numbers need to be tuned carefully when using the pretrained weights.

Replacing FPN with NAS-FPN on top of pretrained weights gives also 5% improvement. There are also improvements brought by replacing ResNet with ResNeSt. This two results demonstrated that the state-of-the-art backbone network improvement proved to be working on other dataset can also improve the performance on KITTI dataset. We expect this kind of improvement can also be seen in other networks and structures as they can be easily used as a replacement of ResNet and NAS-FPN. However, we also observed that these modern structures are relatively more complex thus harder to train in our experiments.

We notice that there's a big accuracy gap (8%) between ResNet + FPN and ResNeSt + FPN, but we don't expect such a big performance gain by simply replacing ResNet with ResNeSt. We suspect this is because we didn't spend enough time fine tuning the model based on ResNet, and the ideal results for ResNet should be much closer to ResNeSt, due to the time constraint of this project, this will also be explored in our future work.

7. Future Work

For future work, we would like to train KITTI dataset and tsinghua cyclist dataset together instead of training tsinghua cyclist dataset and using the pretrained weights to train KITTI dataset. Given that the number of labeled cars are much more than cyclist in KITTI dataset, we observed that in the late period of training the model tends to optimize more for car detection instead of cyclist as discussed above. Thus we want to mix up more cyclist examples into training following the method described in [1]. Specifically,

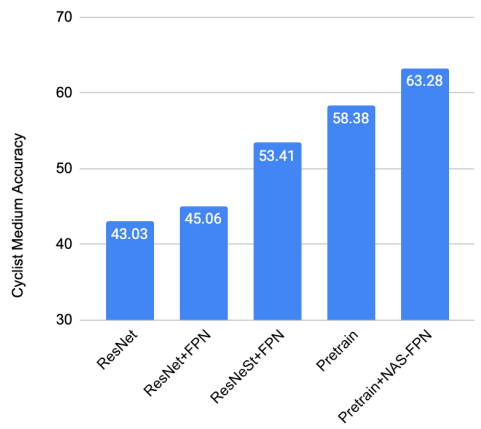

Figure 10. Performance gain from different method.

since tsinghua cyclist dataset have no labeled data for cars and pedestrians, we need to change the model to ignore these two classes when the source image is from tsinghua dataset, and only calculate loss and gradients for boxes that predicted to be cyclist. Using this method, the network can be trained normally on KITTI dataset, and only optimize for cyclist on Tsinghua-Daimler dataset. We expect this can further improve the cyclist accuracy with the right normalization on two dataset.

Besides, NAS-FPN introduced more new layers, especially with 3, 5, 7 stacked NAS-FPN, which leads the model easier to overfit and harder to train. In original paper[3], they apply DropBlock with size 3x3 after the batch normalization layers, which also improve the performance. We would like to add DropBlock in our NAS-FPN implementation to see if it boost the detection accuracy.

We would also like to try more hyperparameters setup like using Adam optimizer, try different normalization methods and different weight decay setup.

8. Contributions

Used code (with local modification):

- Facebook Detectron2 [16]
- ResNeSt Implementation on Detectron2 [19]
- vod-converter for converting KITTI format to PASCAL 2012 [15]
- Followed tensorflow NAS-FPN implementation [18]

Zhen Li

- Research related work and state-of-the-art backbone improvements, main author for final report.
- Converted & trained Tsinghua-daimler dataset. Do pretraining, experiment pretrain vs no-pretrain on ResNeSt + FPN and ResNeSt + NAS-FPN.
- Implemented validation loss, tuned data augmentation config.

Yijing Bai

- Implemented the evaluation code converting the inference result to KITTI format to run the evaluation, experiment various ResNet + C4, FPN.
- Implemented NAS-FPN with help from Linkun and stacked NAS-FPN, experiment ResNet101 + FPN/NAS-FPN.
- Conducted various hyperparameter tuning.

Linkun Chen

- Build and maintain training pipeline as well as visualization tool.
- Implemented NAS-FPN with Yijing, experiment ResNeST50/101 + multi-stack/vanilla FPN/NAS-FPN.
- FPN/NAS-FPN hyperparameter tuning.

References

[1] Tobias Brosch and Ahmed Elshaarany. Object detection and classification on heterogeneous datasets.

[2] Quanfu Fan Rogerio S. Feris Cai, Zhaowei and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. *European conference on computer vision*, pages 354–370, 2016.

[3] Tsung-Yi Lin Ghiasi, Golnaz and Quoc V. Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[4] Ross Girshick. Fast r-cnn. *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2016.

[5] David Martín Guindel, Carlos and José María Armingol. Joint object detection and viewpoint estimation using cnn features. *IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pages 145–150, 2017.

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

[7] Xuemiao Xu Yongjie Xiao Hao Chen Shengfeng He Jing Qin Hu, Xiaowei and Pheng-Ann Heng. Sinet: A scale-insensitive convolutional neural network for fast vehicle detection. *IEEE transactions on intelligent transportation*, 2018.

[8] Fabian Flohr Yue Yang Hui Xiong Markus Braun Shuyue Pan Keqiang Li Li, Xiaofei and Dariu M. Gavrila. A new benchmark for vision-based cyclist detection. *IEEE Intelligent Vehicles Symposium (IV)*, pages 1028–1033, 2016.

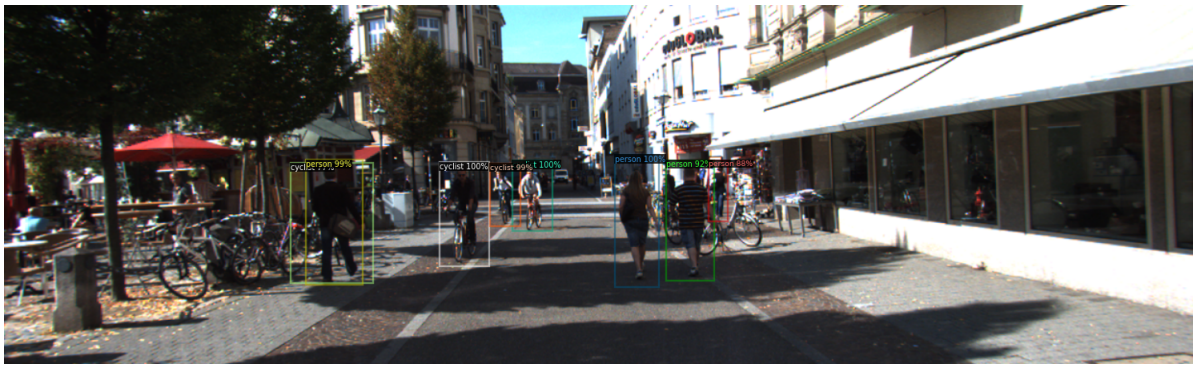

Figure 11. Predicted results for final model.

[9] Dollár P. Girshick R. He K. Hariharan B. Belongie S Lin, T. Y. Feature pyramid networks for object detection. *IEEE conference on computer vision and pattern recognition*, 2017.

[10] Linkun C., Zhen L., and Yijing B. `https://github.com/lk-chen/cs231n_project`, 2020.

[11] Dragomir Anguelov Dumitru Erhan Christian Szegedy Scott Reed Cheng-Yang Fu Liu, Wei and Alexander C. Berg. Ssd: Single shot multibox detector. *European conference on computer vision*, pages 21–37, 2016.

[12] Karlsruhe Institute of Technology (KIT). Kitti vision benchmark suite, 2012. http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=2d.

[13] Kaiming He Ross Girshick Ren, Shaoqing and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, pages 91–99, 2015.

[14] Facebook research. `https://github.com/facebookresearch/Detectron/blob/master/FAQ.md#q-how-do-i-compute-validation-ap-during-training`, 2017.

[15] Karl Rosaen. Visual object dataset converter. `https://github.com/umautobots/vod-converter/tree/29e16918145ebd97e1692ae8e7ef3dc4da242a88`, 2017.

[16] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[17] Wongun Choi Yang, Fan and Yuanqing Lin. Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[18] YangXue. `https://github.com/DetectionTeamUCAS/NAS_FPN_Tensorflow`, 2019.

[19] Zhang Z Zhu Y Zhang Z Lin H Sun Y He T Mueller J Manmatha R Li M Zhang H, Wu C. Resnest: Split-attention networks. *arXiv*, 2020.