Team Pullingos: Multi-Object Detection

Venkatesh Prasad Venkataramanan
A53318036
vvenkata@ucsd.edu

Shakeel Mansoor Shaikna
A53315574
samansoo@ucsd.edu

Siddarth Meenakshi Sundaram
A53299801
simeenak@ucsd.edu

Zirui Wang
A53279727
ziw007@ucsd.edu

Abstract

Multi-Object detection is an integral part of various advanced tasks like self-driving technology, face recognition, object counting and so on and hence its importance has grown exponentially. Huge improvements have been made in the past decade in this field. There have been new architectures and algorithms to detect multiple objects simultaneously and the accuracy has improved by a huge margin in the past few years. Although the accuracy has improved over the years, we find in this project that there can be limitations to algorithms and that the improvements made are not universal. We prove that ResNeXt algorithm isn't always better for image detection compared to ResNet as claimed and that ResNet may be better than ResNeXt in smaller datasets.

1 Introduction

Multi-Object Detection is a field that is undergoing rapid developments with it being the core of many emerging technologies like self-driving cars[1], face recognition, surveillance and so on. Deep Learning is the most advanced technique to detect objects accurately and instantly. To study object detection in detail, in this project, we have experimented on Deep Learning networks of Faster R-CNN[2] and Cascade R-CNN[3] with ResNet and ResNeXt backbones.

We have uploaded the codes in the following github repository. The electronic website to access this repository is at :

`https://github.com/venkateshprasad23/Multi-Object-Detection`

The repository will contain the training data, instructions to reproduce the experiments, demos along with the codes themselves.

1.1 Dataset

We use the PASCAL VOC 2012 challenge dataset[4] which is a standard dataset used for Object detection. Pascal VOC Dataset consists of 20 classes such as aeroplanes, car, bus, person, etc.

This dataset comes with its own set of limitations.

- The challenge in this project is the availability of very small dataset to train consisting of about 5717 images to train and we use the validation dataset with about 5823 images available in the VOC 2012 dataset to test.
- As we use the validation data to test, we do not have separate data to perform validation during training. One normally uses Dropout[5], Batch Normalization[6] to prevent the

network from overfitting, but the lack of a validation set prevents us from using such techniques.

1.2 Performance Metric

There are two common metrics used in object detection - mAR and mAP. We have used mAP in this project due to the highly reliable data available from past experiments on PASCAL VOC 2012 dataset already.

- **mAR**: mean Average Recall (mAR) is a numerical metric that can be used to compare detector performance. In essence, AR is the recall averaged over all and can be computed as two times the area under the recall-IoU curve. Mean average recall is defined as the mean of AR across all 20 classes.
- **mAP**: Similar to Average Recall, If you have a Average Precision score of close to 1.0 then there is a high likelihood that whatever the classifier predicts as a positive detection is in fact a correct prediction. mean Average Precision is defined as the mean of AP across all 20 classes. As we are using mAP to compare the performance of the experiments with the baseline models, we will explain in-detail about this metric in Section 3.

1.3 Work flow

In Section 2, we will discuss the methods used (Faster R-CNN, Cascade R-CNN), their architectures and the modifications we make to run our task efficiently. ResNet[7], residual network was developed in 2016 and ResNeXt[8], an Aggregated Residual Transformations for Deep neural network was developed in 2016, in which X which suggests the next dimension. In the above paper[8], ResNext50 is said to outperform not only ResNet50[7] - ResNet with 50 layers, but also ResNext101 - ResNeXt with 101 layers.

In Section 3, we will describe the settings used in running the experiments such as dataset, training parameters, validation and testing procedure (data split, evolution of loss with number of iterations, performance metrics used, etc.)

In Section 4, we will obtain our results, using GPU from the UCSD DSMLP Cluster[9]. We used a single GPU for training and as the load in the Cluster is high, we train the models for 4 epochs to compare the performance between the models. The ResNet backbone model is used as the baseline to compare the performance of the ResNeXt and the modified ResNeXt models. After collecting the results in this section, we will then go on to tabulate it.

In the Section 5, we will summarise the work done and infer the results. We will then conclude with work that can be done in future using this project as an inspiration.

2 Description of the Method

2.1 Algorithms

In object detection, an intersection over union (IoU) threshold is required to define positives and negatives. An object detector, trained with low IoU threshold, e.g. 0.5, usually produces noisy detections. However, detection performance tends to degrade with increasing the IoU thresholds. Two main factors are responsible for this: 1) over-fitting during training, due to exponentially vanishing positive samples, and 2) inference-time mismatch between the IoUs for which the detector is optimal and those of the input hypotheses. A multi-stage object detection architecture, Faster Region-based Convolutional Neural Network(Faster RCNN)[2] and the Cascade R-CNN[3] is proposed to address these problems along with Region Proposal Networks.

2.1.1 Faster Region-based Convolutional Neural Network (Faster R-CNN)

The Faster Region-based Convolutional Network(Faster R-CNN)[2] is a combination between the RPN and the Fast R-CNN model. A Region Proposal Network (RPN) is inserted after the last convolutional layer. Region proposals detected with the selective search method is computationally expensive. [2] has introduced Region Proposal Network (RPN) which is trained to generate region

proposals directly, predict bounding boxes and detect objects. Next stage in this algorithm consists of Region of Interest (RoI) pooling, classifier and bbox regressor and are used just like Fast R-CNN[10].

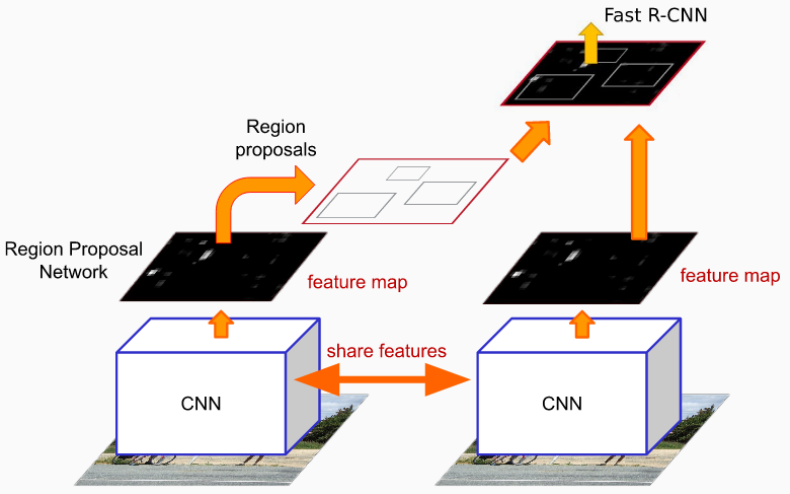

Figure 1: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network. Source: [2]

A CNN model takes the entire image as input and produces feature maps. A window of size 3x3 slides over the feature maps and outputs a features vector linked to two fully-connected layers, one for box-regression location and one for box-classification into object or background. A maximum of k regions is fixed, thus the output of the box-regression layer has a size of 4k (coordinates of the boxes,their height and width) and the output of the box-classification layer a size of 2k ("objectness" scores to detect an object or not in the box). The k region proposals detected by the sliding window are called anchors. When the anchor boxes are detected, they are selected by applying a threshold over the "objectness" score to keep only the relevant boxes. These anchor boxes and the feature maps computed by the initial CNN model feeds a Fast R-CNN model. Faster R-CNN uses RPN to avoid the selective search method, it accelerates the training and testing processes, and improve the performances. The RPN uses a pre-trained model over the ImageNet dataset for classification and it is fine-tuned on the PASCAL VOC dataset. Then the generated region proposals with anchor boxes are used to train the Fast R-CNN. This process is iterative.

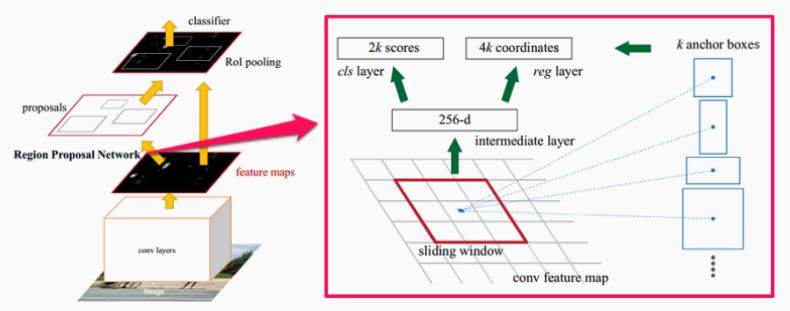

Figure 2: Detecting the anchor boxes for a single 3x3 window. Source: [2]

2.1.2 Cascade R-CNN

Cascade R-CNN[3] consists of a sequence of detectors trained with increasing IoU thresholds, to be sequentially more selective against close false positives. The detectors are trained stage by stage, leveraging the observation that the output of a detector is a good distribution for training the next higher quality detector. The resampling of progressively improved hypotheses guarantees that all detectors have a positive set of examples of equivalent size, reducing the overfitting problem. The

same cascade procedure is applied at inference, enabling a closer match between the hypotheses and the detector quality of each stage.

The two-stage architecture of the Faster-RCNN is improved in this algorithm. The first stage is a proposal sub-network ("H0"), applied to the entire image, to produce preliminary detection hypotheses, known as object proposals. In the second stage, these hypotheses are then processed by a RoI detection sub-network ("H1"), denoted as detection head. A final classification score ("C") and a bounding box ("B") are assigned to each hypothesis.

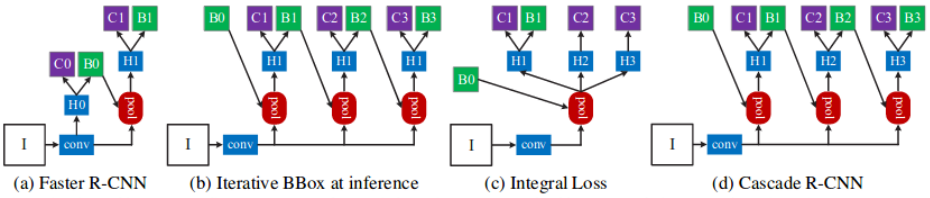

Figure 3: The architectures of different frameworks. "I" is input image, "conv" backbone convolutions, "pool" region-wise feature extraction, "H"network head, "B" bounding box, and "C" classification. "B0"is proposals in all architectures. Source: [3]

A simple implementation of the Cascade R-CNN is shown to surpass all single-model object detectors on the challenging COCO dataset. Experiments also show that the Cascade R-CNN is widely applicable across detector architectures, achieving consistent gains independently of the baseline detector strength.

2.2 Network Architecture

For an object detection task, we have to rely on transfer learning. We use backbone networks pre-trained on the ImageNet challenge, and re-purpose them for smaller training sets. A number of successful networks have been created over the years.

The curse of dimensionality mandates that one must use a network of lesser number of layers in order to guard against overfitting. Since we have an extremely small dataset, it is not advisable to use a network with over 100 layers, and hence we decided to test relatively shallow networks, consisting of 50 layers at max. ResNet, the winner of the ILSVRC2015[11] is an extremely capable and versatile network. ResNeXt is the runner-up in ILSVRC2016[11] introduced to outperform ResNet.

In this project, we test the above two backbone architectures, namely ResNet and ResNeXt.

2.2.1 ResNet

A residual neural network (ResNet)[7] is an artificial neural network (ANN) of a kind that builds on constructs by utilizing skip connections, or shortcuts to jump over some layers. Typical ResNet models are implemented with double- or triple- layer skips that contain nonlinearities (ReLU) and batch normalization in between. An additional weight matrix may be used to learn the skip weights. In the context of residual neural networks, a non- residual network may be described as a plain network.One motivation for skipping over layers is to avoid the problem of vanishing gradients, by reusing activations from a previous layer until the adjacent layer learns its weights. During training, the weights adapt to mute the upstream layer, and amplify the previously-skipped layer. In the simplest case, only the weights for the adjacent layer's connection are adapted, with no explicit weights for the upstream layer. This works best when a single nonlinear layer is stepped over, or when the intermediate layers are all linear. If not, then an explicit weight matrix should be learned for the skipped connection. Skipping effectively simplifies the network, using fewer layers in the initial training stages. This speeds learning by reducing the impact of vanishing gradients, as there are fewer layers to propagate through. The network then gradually restores the skipped layers as it learns the feature space. Towards the end of training, when all layers are expanded, it stays closer to the manifold and thus learns faster. A neural network without residual parts explores more of the feature space. This makes it more vulnerable to perturbations that cause it to leave the manifold, and necessitates extra training data to recover.

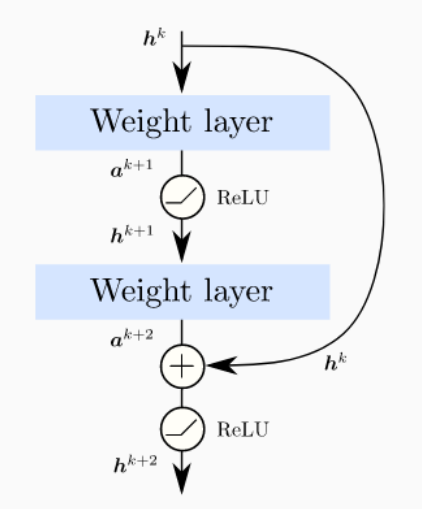

Figure 4: Skips in ResNet architecture

2.2.2 ResNeXt

ResNeXt[8] is constructed by repeating a building block that aggregates a set of transformations with the same topology. The simple design results in a homogeneous, multi-branch architecture that has only a few hyper-parameters to set. This strategy exposes a new dimension, which is called "cardinality" (the size of the set of transformations), as an essential factor in addition to the dimensions of depth and width. On the ImageNet-1Kdataset, increasing cardinality is able to improve classification accuracy. Moreover, increasing cardinality is more effective than going deeper or wider when the capacity is increased.

ResNeXt won 2nd place in ILSVRC 2016 classification task and also showed performance improvements in Coco detection and ImageNet-5k set than their ResNet counterpart.

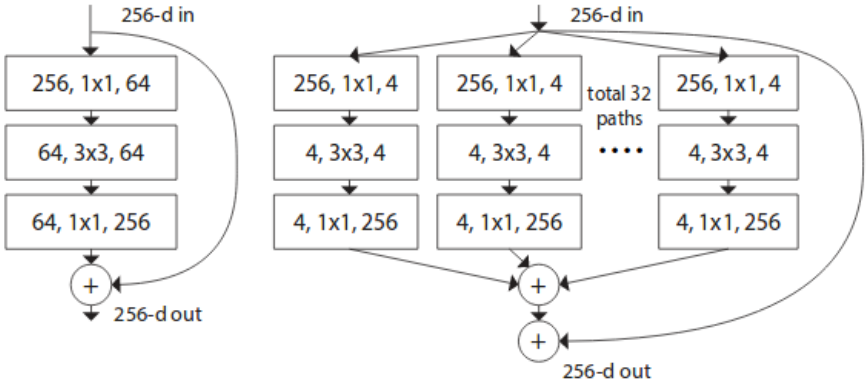

Figure 5: Left: A block of ResNet 14[7]. Right: A block of ResNeXt with cardinality=32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels). Source: [8]

Figure 5 distinguishes between a simple ResNet block and ResNeXt block. It follows a split-transform-aggregate strategy. The number of paths inside the ResNeXt block is defined as cardinality. In Figure 5 Cardinality C=32. All the paths contain the same topology. ResNeXt tries to embed more subspaces compared to its ResNet counterpart. Both the architectures have different width. Layer-1 in ResNet has one convolutional layer with 64 width, while layer-1 in ResNext has 32 different convolutional layers with 4 width (32*4 width). ResNet parameters = 256*64+3*3*64*64+64*26) ResNeXt parameters = C*(256*d+3*3*d*d+d*256), with C=32 and d=4.

Despite the larger overall width in ResNeXt, both architectures have the same number of parameters(about 70k)

2.3 Performance Metric

We use the mAP metric to evaluate the performance of our experiments.

The metric that is used in our tasks is 'mAP'[12], which stands for 'mean Average Precision'. It is a numerical value from 0 to 1. The higher values are typically better, but its value is different from the accuracy metric in classification. Each bounding box will have a score associated (likelihood of the box containing an object). The average precision (AP) is the area under the Precision Recall curve. First the AP is computed for each class separately, and then averaged over the 20 classes to get mAP.

Figure 6: Intersection over Union(IoU), Source: [13]

Note that a detection is a true positive if it has an 'intersection over union' (IoU) with the ground-truth box greater than some threshold (usually 0.5).

2.4 MMDetection

mmdetection[14] is an open source object detection toolbox based on PyTorch. It is a part of the open-mmlab project developed by Multimedia Laboratory, CUHK. The model components are categorized into 4 types in mmdetection,

- Backbone: A backbone is usually an FCN network to extract feature maps, e.g., ResNet, MobileNet. A backbone is a convolutional network which is usually pre-trained on ImageNet dataset. It is used to extract feature maps. The hidden layers in the pre-trained backbone can be frozen and re-trained for our purposes using the concept of transfer learning. Some examples include ResNet, HRNet, ResNeXt,etc.
- Head: Head is the component for specific tasks, e.g., bbox prediction and mask prediction. Component for bbox prediction is usually a linear regressor, and for region prediction, it is usually a Support Vector Machine(SVM) classifier.
- Neck: A neck is the component between backbones and heads, e.g., Feature Pyramid Network[15], PAFPN, etc.
- RoI Extractor: RoI Extractor is the part for extracting RoI features from feature maps, e.g., RoI Align.

3 Experimental Setting

3.1 Dataset

The Pascal VOC dataset which was used for the 2012 challenge was used in our experiments. The Pascal VOC Dataset consists of 20 classes. The train/val data has 11,530 images containing 27,450 ROI annotated objects and 6,929 segmentations. For our purpose, we would be using only the training data(5717 images) for training and validation data(5823 images) for testing. Such a small training set and the lack of the validation set poses an extremely difficult challenge.

3.2 Description of the Experiments

We perform two experiments to compare the performance of ResNet50 and ResNeXt50.

3.2.1 Experiment 1 (Faster R-CNN)

As seen in section 2.4, mmdetection requires the specification of the following configurations. We are comparing the two backbones of ResNet50 and ResNeXt50, keeping the neck as FPN, head as RPN for region extraction and SharedFCBBoxHead[14] as the head for bounding boxes, and SingleRoIExtractor for extracting RoIs of bounding boxes.

Table 1: Training Parameters

Training Parameters	Experimental Conditions
Optimizer	SGD, Momentum = 0.9, weight decay = 0.0001
Learning Rate Scheduler	lr = 0.01, 'step' policy
Number of Epochs	4

3.2.2 Experiment 2 (Cascade R-CNN)

For the purpose of comparison, we are retaining the same network architecture as Experiment 1. We are comparing the performance of the two backbones, namely ResNet50 and ResNeXt50, on the two algorithms, Faster RCNN and Cascade RCNN.

Table 2: Training Parameters

Training Parameters	Experimental Conditions
Optimizer	SGD, Momentum = 0.9, weight decay = 0.0001
Learning Rate Scheduler	lr = 0.01, 'step' policy
Number of Epochs	4

3.3 Validation and Testing procedure

The Pascal VOC dataset which was used for the 2012 challenge was used in our experiments. The Pascal VOC Dataset consists of 20 classes. The train/val data has 11,530 images containing 27,450 ROI annotated objects and 6,929 segmentations. Since there is no separate data for testing on the UCSD DSMLP cluster, we would be using only the training data(5717 images) for training and validation data(5823 images) for testing.

The plots for evolution of losses with the number of iterations has been plotted for all the cases. Particularly, we are concerned with the regression loss for the prediction of bounding boxes, and the classification loss for the "objectness" of a particular region in an image.

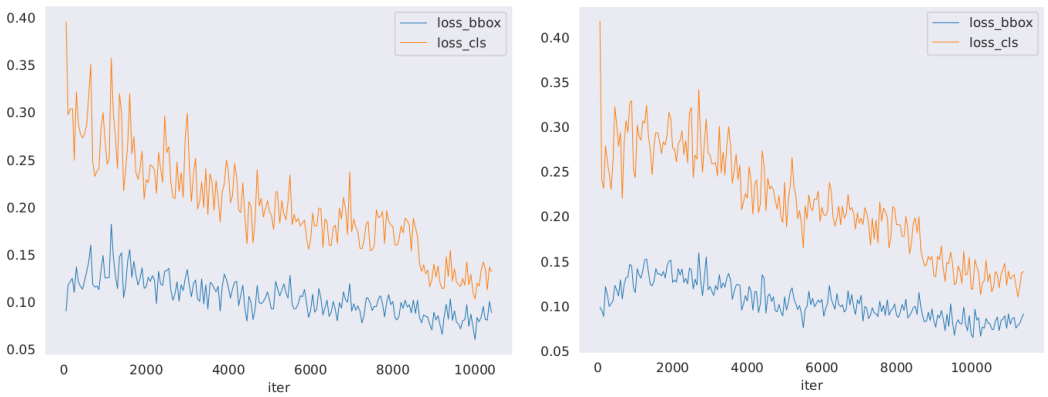

Figure 7: Faster R-CNN. Left: ResNet-50. Right: ResNeXt-50

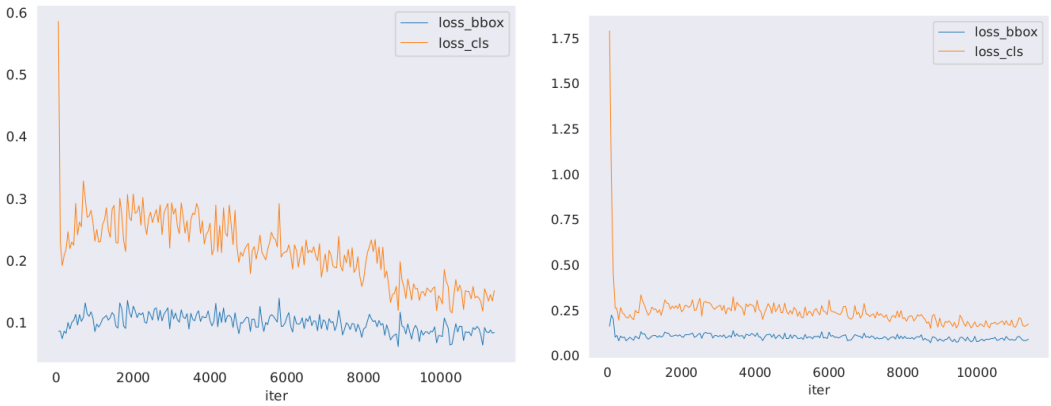

Figure 8: Cascade R-CNN. Left: ResNet-50. Right: ResNeXt-50

4 Results

Performance of these techniques has been benchmarked by calculating the class-wise AP(Average Precision) and then averaging over all classes to find the mAP. They have been curated in Table 3.

We make the following observations. We can see that, in both the algorithms, ResNeXt50 performs worse than ResNet50, even though its training accuracy was higher. Also, in all 4 cases, the class "Person" has a very high average precision. This is as a result of higher number of annotations for that particular class in the VOC 2012 dataset. Out of the four models which were considered, ResNet50 with Faster R-CNN has the highest mAP. Hence, it is the best performing model for our purpose as seen in Figure 9.

The training accuracy for ResNeXt 50 is higher than that of ResNet 50, for both the algorithms, namely Faster R-CNN and Cascade R-CNN. This was expected, since it is an improvement of the ResNet architecture. But, the problem arises since its Mean Average Precision on the testing set is lower than that of ResNet 50. This is a clear indication of overfitting. Since we are using an extremely small dataset, this was expected.

5 Discussion, Conclusion and Future works

We can observe from the above results that the ResNet50 algorithm produces better results than ResNeXt50 algorithm in both experiments. From the previous statement, we can say that ResNeXt backbone does not always produce better results than ResNet as stated in [8]. We can hence conclude that the new and cutting-edge networks and methods developed may not be better than the existing networks in an universal manner and that each networks has its pros and cons and are efficient in different tasks with varying levels of accuracy.

In this project, the challenge of using a smaller dataset was taken as a setting of the task to better choose our algorithms and backbones. We learn that knowing the algorithms inside-out is necessary to apply them to obtain superior results. Each algorithm will work better with different parameters/settings for different datasets and tasks. We must be able to apply different algorithms based on the task at hand.

The major constraint using Pascal VOC dataset is that we could not use Batch Normalization or Dropout due to the lack of validation dataset. We know that ResNeXt model overfits as the training loss of ResNeXt is lower but the error in the test is higher compared to ResNet. In future, a larger dataset such as COCO, available with a separate training, validation and test dataset maybe be used to test our hypothesis and possibly prevents overfitting.

The object detection task has several parameters that can be improved upon, like accuracy, inference speed, computational complexity, etc. Recent research has been aligned towards reducing computational overhead, in order to commercialise AI applications for devices with limited computational capability, but, here, by focusing on accuracy, we strive to improve the accuracy of these computationally simpler approaches.

Table 3: Class-wise AP and model-wise mAP

	Average Precision			
Class	Faster R-CNN		Cascade R-CNN	
	ResNet50	ResNeXt50	ResNet50	ResNeXt50
aeroplane	0.785	0.685	0.708	0.58
bicycle	0.639	0.63	0.587	0.477
bird	0.66	0.585	0.566	0.405
boat	0.372	0.315	0.332	0.202
bottle	0.492	0.491	0.454	0.345
bus	0.747	0.649	0.692	0.613
car	0.721	0.688	0.669	0.64
cat	0.808	0.714	0.649	0.554
chair	0.428	0.423	0.325	0.238
cow	0.38	0.242	0.281	0.225
diningtable	0.4	0.345	0.306	0.197
dog	0.725	0.609	0.563	0.437
horse	0.54	0.473	0.453	0.349
motorbike	0.716	0.667	0.672	0.433
person	0.818	0.819	0.798	0.773
pottedplant	0.392	0.372	0.307	0.104
sheep	0.623	0.504	0.541	0.412
sofa	0.48	0.438	0.359	0.255
train	0.633	0.515	0.586	0.354
tvmonitor	0.636	0.589	0.559	0.417
mAP	0.600	0.538	0.520	0.400

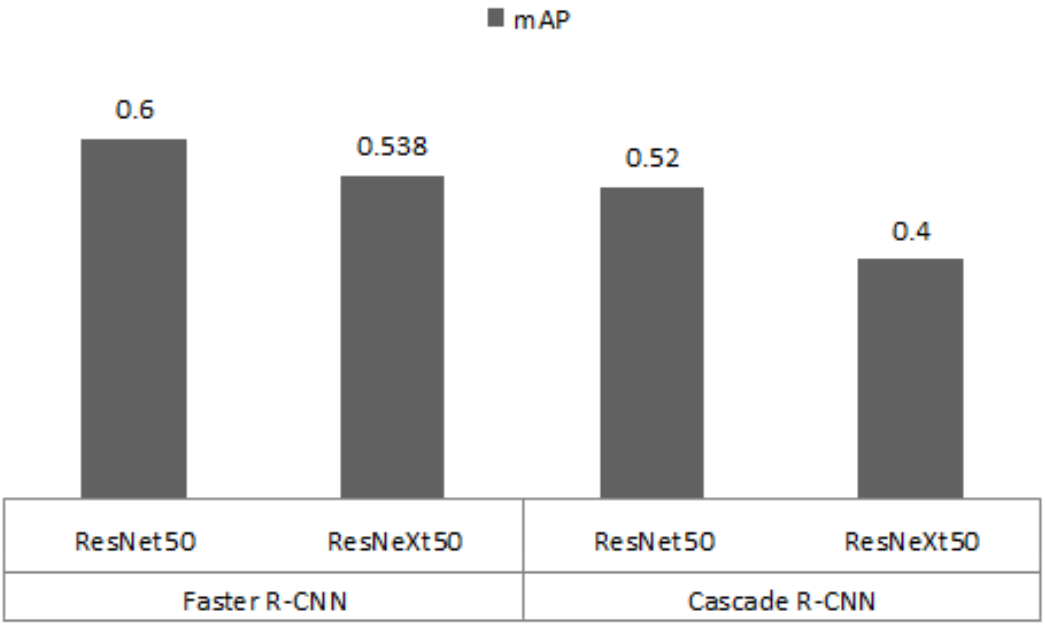

Figure 9: Comparison of mAP

References

[1] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.

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

[3] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162, 2018.

[4] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[5] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[7] 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.

[8] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[9] UCSD DSMLP Cluster. (accessed November, 2019). `https://datahub.ucsd.edu/`.

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

[11] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[12] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. A support vector method for optimizing average precision. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 271–278. ACM, 2007.

[13] Rajalingappaa Shanmugamani. *IoU - Deep Learning for Computer Vision*, (accessed December, 2019). `https://www.oreilly.com/library/view/deep-learning-for/9781788295628/a5ce2fa2-8c67-4ead-a9bd-a2d07b5f3fa8.xhtml`.

[14] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[15] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.