Faster R-CNN with Focal loss

(MP 084, Last Minute)

Phu Pham (597254) Tinka Valentijn (521042) Kha Nguyen (600646)

January 31, 2018

Abstract

Faster R-CNN [1] is one of the most efficient detection networks that utilizes and improves the bottleneck from other detection networks such as Fast R-CNN [2]. By introducing Region Proposal Network (RPN), Faster R-CNN is a two-stage detector that can detect objects with state-of-the-art results and close to real-time speed. Shortly after, the recent paper by Tsung-Yi Lin et al. [3] proposed a one-stage detector RetinaNet. RetinaNet was trained with a novel loss function called "focal loss" and it outperforms Faster R-CNN. In this paper, we implemented Faster R-CNN and applied the focal loss function on the RPN to evaluate the performance of Faster R-CNN. We trained our models with 3 different setups: 3-object-class with traditional loss function of Faster R-CNN, 3-object-class with focal loss function on RPN, and 20-object-class with normal loss function on RPN. Our main goal is to understand how Faster R-CNN works and to experiment the performance of Faster R-CNN with different contexts and loss functions. We succeeded in this goal and show that there are several differences in performance between the 3 models.

1 Introduction

Nowadays, there are two types of deep neural networks for object detection, two-stage and one-stage detectors. Of both types we took one state-of-the art algorithm and analyzed them thoroughly. As our two-stage detector we inspected Faster R-CNN and as one-stage detector RetinaNet. We decided to take the Faster R-CNN as base method. We tried to understand how it focuses on classes by comparing the performance on 3 classes to 20 classes. Then, we developed a combination of Faster R-CNN and RetinaNet by applying the focal loss in RetinaNet to the Region Proposal Network (RPN) of Faster R-CNN. The goal of these experiments is to understand how the Faster RCNN and specifically the RPN works.

For this project we used the PASCAL VOC 2012 dataset for object detection. This dataset consists of 11540 annotated images belonging to 20 classes.

2 Related work

Over the last decade several object detection networks have been developed. All top performers are CNN-based. The first breakthrough was the R-CNN[4], which proposes regions from an image, extracts features of each of those regions by applying CNN, and then predicts the class and bounding box with SVM for every region. Though the result was really good, the biggest bottleneck was computation time.

To improve computation time, Girshick proposed Fast R-CNN [2]. Fast R-CNN first extracts the features of the whole image, then proposes regions, and finally classifies those regions. Fast R-CNN mainly focuses on classification. One can freely choose one of the region proposal methods, such as selective search [5]. Though faster than R-CNN, Fast R-CNN is still not really fast.

Hence, Faster R-CNN was proposed [1]. In Faster R-CNN the classification method is the Fast R-CNN, but the novelty is the Region Proposal Network (RPN) that is used to compute the proposed regions. RPN learns regions by using a CNN. A large part of the CNN is shared by the RPN and Fast R-CNN, which speeds up computation significantly and improves performance[1].

However, all these methods use two stages; region proposal, and classification. They still are rather slow. Hence, recently a lot of effort has been put in so called one-stage detectors, which consist of one neural network without the region proposal. Though significant speed-up was reached, the performance decreased[6, 7]. Recently, RetinaNet [3] was proposed. A one-stage detector with better speed and performance than Faster R-CNN. Their main breakthrough is the use of focal loss. The basic idea of focal loss is to make the individual contribution of easy to classify regions to the loss smaller, such that the loss is more focused on the hard and misclassified regions.

During this project our goal was to understand both Faster R-CNN and RetinaNet. We thoroughly read both papers and decided to take Faster R-CNN as a basis for our experiments.

3 Method

We use the Faster R-CNN architecture, see Figure 1 for a schematic overview of the network. As CNN we use VGG-16[8], since it has shown to yield good performance in Faster R-CNN[1] while not having too many parameters. We only use the convolutional layers of VGG-16 and not the fully connected layers. VGG-16 consists of 13 convolutional layers and hence, we shall from now on refer to this network as VGG-13.

We implemented the network in Keras. We reused parts of an already existing implementation[9], but we adjusted it to our own needs and made the code more readable [1], helping in our goal to understand the working of Faster R-CNN.

3.1 Faster R-CNN

3.1.1 Region Proposal Network

The Region Proposal Network first applies a CNN, in our case VGG-13, to the image in order to get a feature map. Then we apply another convolutional layer with a 3x3 filter and 9 channels to this feature map. This layer finds the region proposals. At every sliding window we apply 9 anchors, hence 9 channels. These anchors cover regions of the feature map with 3 different sizes and 3 different scales, making 9 combinations. Due to the 9 anchors, we can capture many different regions. After the convolutional layer, we apply two fully connected layers. One classifies each proposed region in foreground or background. Its output is a probability of the region being foreground. The other fully connected layer computes the bounding box of that region. Hence, its output is the 4 coordinates of the proposed bounding box.

[1] See our Github repository for the code: `https://github.com/Tinkaa/DLproject`

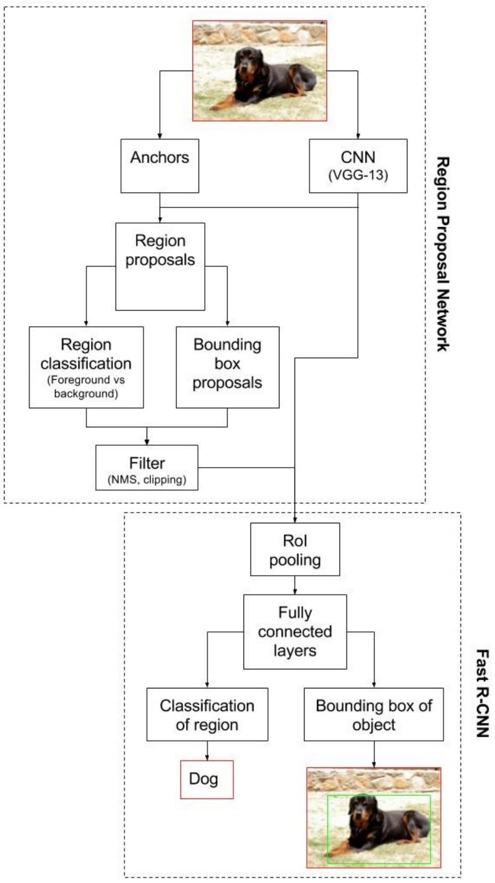


Figure 1: Schematic overview of the Faster R-CNN network.

We now have our proposed regions. On average the size of the feature map is approximately 40x60, this results in approximately 40x60x9 $\approx$ 20K region proposals. However, many of those regions have a very high overlap and hence we use non-maximum suppression (NMS) with an Intersection over Union (IoU) of 0.7. Moreover, during training we only take regions that are fully inside the image, during testing we clip the regions that are partly outside the image. This leaves us with approximately 2K region proposals as input to the classifier, Fast R-CNN.

3.1.2 Fast R-CNN

The input to the Fast R-CNN are the proposed regions and the feature map from the VGG-13. The first layer of the Fast R-CNN is the RoI pooling layer. The proposed regions of the RPN differ in size, but we want them all to have the same size for our classifier. The RoI pooling layer converts each of the regions into a 7x7 feature map. Now we can input these same-sized feature maps into the classifier. Our classifier consists of 2 fully connected layers, each with 4096 nodes. For regularization after each layer a dropout layer is added. After these layers, we have two more classification layers. One applies the softmax function to give the probabilities of the image belonging to the classes, including background. The other classification layer predicts the bounding box coordinates.

3.2 Losses

The Faster R-CNN network uses the traditional cross-entropy loss for both classification phases, and smooth L1 loss for both bounding box regression phases. The choice of loss functions here is logical, but we saw opportunity for improvement after studying how RetinaNet manages to achieve better performance. We took the idea of loss from RetinaNet to apply to Faster R-CNN RPN phase.

3.2.1 Focal Loss

The advantage of RetinaNet over other object detection networks is in its special Focal Loss function[3]. The function has the form

$$FL(p_t) = -\alpha(1 - p_t)^\gamma \log p_t$$

where p_t is

- p, the probability detected for a correct class
- $1 - p$, the probability detected for an incorrect class

The last component $\log p_t$ is the usual cross-entropy for classification problems. RetinaNet takes the cue that, in practice, we have far more negative samples (i.e. background) than positive samples, thus making misclassification incur non-trivial loss, and rare classes are overwhelmed by common classes.

The modulating factor $(1-p_t)^\gamma$ is small for common misclassification, thus downweighting the loss for common classes. For rare classes, p_t is much smaller, thus making modulating factor approach 1, and keep the loss almost unchanged. The gamma exponential is used to control (or focus) the effect of the modulator, hence the name.

Finally, the weighting factor α_t is used to give priority of contribution to loss for each class. Theoretically, we do not need this weighting factor for understanding Focal loss, but it is included in our implementation to have more control over the loss function.

3.2.2 Bring Focal loss to RPN

We notice the similarity between the entire RetinaNet, and the RPN in Faster R-CNN. RetinaNet detects object classes and bounding boxes at every anchor. RPN detects foreground and bounding boxes at every anchor. Both RetinaNet and RPN deal with many background samples. We came up with the idea to replace the crossentropy loss of RPN with the focal loss of RetinaNet and evaluate performance.

We did not bring Focal loss to the detection phase of Faster R-CNN. This phase consumes the proposals from the previous phase which are already only foreground proposals, and thus does not suffer from extreme disproportion of background samples.

4 Data

We use the PASCAL VOC 2012 object detection dataset[10]. This dataset consists of 11540 images containing 27450 objects. Each image has an annotation XML file, which contains the objects with bounding boxes and classes in that image.

In the whole dataset, 5717 images are used for training and the others for validation. For a part of our experiments, we only use images from 3 classes during training(cats, dogs and cars). See Table 1 for the number of images and objects per class in the training and validation set.

The dimension of the images differ, but before inputting the images to our network we reduce the image to a size such that the shortest side is 600 pixels. Apart from that, no preprocessing is done to the images.

	train images	train objects	validation images	validation objects
Car	590	1013	571	1004
Cat	539	605	541	612
Dog	632	756	654	759

Table 1: The number of images and objects in the training and validation set for each of the three classes.

5 Experiments

In our experiments we changed two parts of the model, namely the number of classes and the loss for the RPN. The goal of changing the number of classes, was to get an insight in the working of the Faster R-CNN. We wanted to discover if the Faster R-CNN scores better on the limited number of chosen classes. Our reasoning was that it might be better at those classes, since it could focus all its attention on those classes. On the other hand, it has seen a less diverse range of data and hence it might be worse at detecting unseen objects.

The goal of changing the loss function was to see if we could use the discoveries of RetinaNet to improve Faster R-CNN. We changed the loss function of the RPN classifier to focal loss. In order to conduct both experiments we train 3 different scenarios: 1) Faster R-CNN on all 20 classes, 2) Faster R-CNN on 3 classes, and 3) Faster R-CNN with focal loss on 3 classes.

Next to the 3 experiments, we also tuned different hyperparameters. The main breakthrough of Faster R-CNN is the use of RPN. At each sliding windows on the convolution feature map, there are k **anchors** for possible region proposals. The more anchors for each slider window has, the more likely that objects can be detected. However, the trade-off here is the computation cost. We experimented with different values of k in range from 4 (2 scales, 2 aspect ratios) to 25 (5 scales, 5 aspect ratios). The value k=9 gave acceptable results with reasonable cost. This also agreed with the value set in the original paper of Faster R-CNN[1].

Besides, we also tested various values for Intersection-over-Union (IoU) overlap thresholds. The upper bound value of IoU defines the threshold to select positive samples while the lower bound value defines the threshold to select negative samples. It means that if an anchor has an IoU overlap higher than the upper bound with any ground-truth bounding box, it will be labeled as positive. In contrast, if the IoU overlap is smaller than the lower bound, it will be labeled as negative. In our experiment, we lower the upper bound to around 0.6, the results were worst than the value reported in the original paper which is 0.7[1]. Our model detected the correct class of an object but the predicted bounding box was misplaced or placed far off the ground truth bounding box. The results were understandable as the upper threshold was too low, hence, many anchors were labeled as positive samples and led to misplaced bounding boxes.

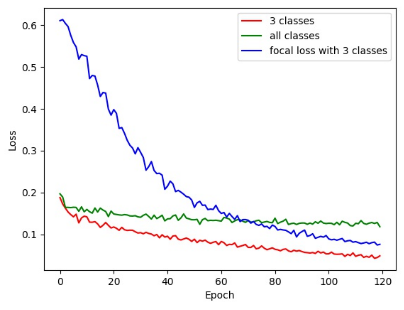
Apart from the anchors and the IoU, we kept all hyperparameters the same as in the original Faster R-CNN and RetinaNet paper[1, 3].

In the official PASCAL VOC competition mean Average Precision (mAP) is used as the performance measure. For the PASCAL VOC 2012 no public annotations are available and results have to be submitted to a server. We made a submission, but unfortunately it gave a not understandable error and the people of VOC were not reachable for help. Hence, we tried writing our own mAP code for testing on the PASCAL VOC 2007. However, since it is rather hard to compute the mAP for object detection and since we only noticed rather late that the server didn't work, we sadly didn't succeed. Hence, we have to compare our models based on the lost functions, the accuracy of the RPN classifier and by visually looking at detection done on images from the test set.

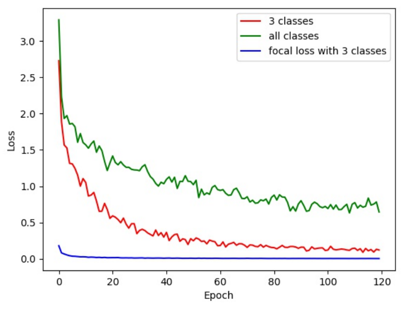
In order to speed up training process, we made use of pre-trained VGG-16 network by loading the weights that were already trained on ImageNet. In the next step, we trained the whole network from end to end with optimal values of anchors per sliding windows (k = 9) and thresholding values (0.3 for lower bound and 0.7 for upper bound). Because of time limitation and lack of computation resources, we could only train our network for 120 epochs, 1000 iterations in each epoch. Moreover, we did not fully utilize the power of GPU computing, our training time was several times longer than the number reported in the Faster R-CNN paper. It took about 25 to 30 hours for training with one scenario on Aalto computing cluster called Triton. We used 1 CPU with 200GB of memory and 1 Tesla P100 GPU with 16GB of memory.

6 Results

We computed the loss for all the 4 loss functions, i.e. the regressor and classifier in the RPN and in the Fast R-CNN.



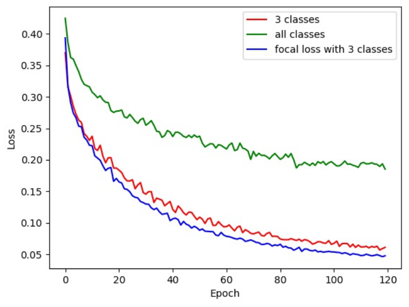
(a) Regressor



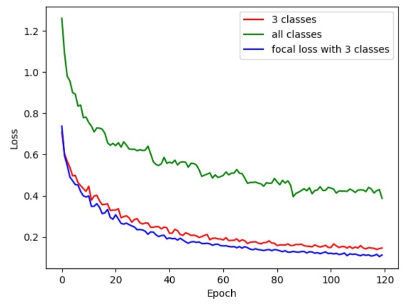
(b) Classifier

Figure 2: RPN loss

Figure 2 indicates the loss of the RPN regressor and RPN classifier as a function of epoch for the 3 scenarios mentioned in section 5. As we can see, the RPN regressor loss for scenarios 1 and 2 with normal loss function started with small value but slowly converge to the optimal. On the other hand, the RPN regressor loss for scenario 3 with focal loss function was high at the beginning but quickly converges to the optimal. For the RPN classifier, loss for scenario 3 was very small and remained almost unchanged during training. In scenarios 1 and 2, the loss for scenario 2 (with 3 classes) converged faster than scenario 1 (all classes).



(a) Regressor



(b) Classifier

Figure 3: Fast R-CNN detector loss

Figure 3 shows the loss of the Fast R-CNN regressor and classifier for the 3 scenarios. In the Fast R-CNN detector, we did not use focal loss. However, the improved accuracy in the RPN due to focal loss seems to have a little effect on the R-CNN detector as well since we can see that the loss here of scenario 3 is slightly smaller than of scenario 2. The loss for scenario 2 and 3 both for the regressor and classifier converged faster than scenario 1. The results are reasonable as scenario 2 and 3 only have 3 classes while scenario 1 has 20 classes.

Moreover, we compute the accuracy of the RPN classifier. See Figure 4 for the development of the accuracy of the RPN over time for all architectures. We can see that scenario 3 reaches the highest accuracy, while scenario 1 is clearly lacking behind.

Lastly, to compare performance we detected and visualized bounding boxes for several images from the test set and compared them. Most detections were rather similar for all 3 models. Especially, for the model with and without focal loss and hence we don't depict them here. However, in some pictures there were interesting differences between the model on all and 3 classes. See Figure 5 and Figure 6 for 3 examples from the 3 and all classes model respectively. These 3 examples depict 3 different scenarios: an image with a class that is part of both models, an image that only belongs to the classes of the all classes model, and an image that doesn't belong to any of the classes of both models.

From these images, it can be seen that the model on all classes sometimes predicts too many bounding boxes, shown by the cat example. On the cow example, the model on all classes predicts correctly the class of one cow, but misses the other cow. On the other hand, the 3 classes model recognizes both cows but predicts them as dogs. In the giraffe example, the 3 classes model recognizes a cat and a dog whereas the all classes model doesn't detect any object.

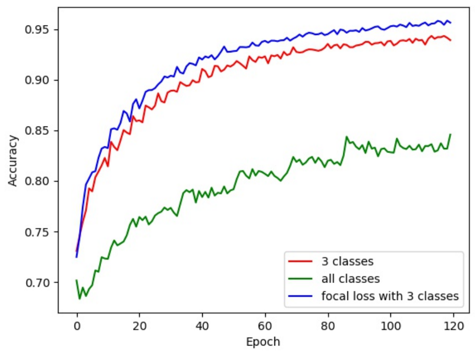


Figure 4: RPN classifier accuracy over the number of epochs for the 3 architectures'

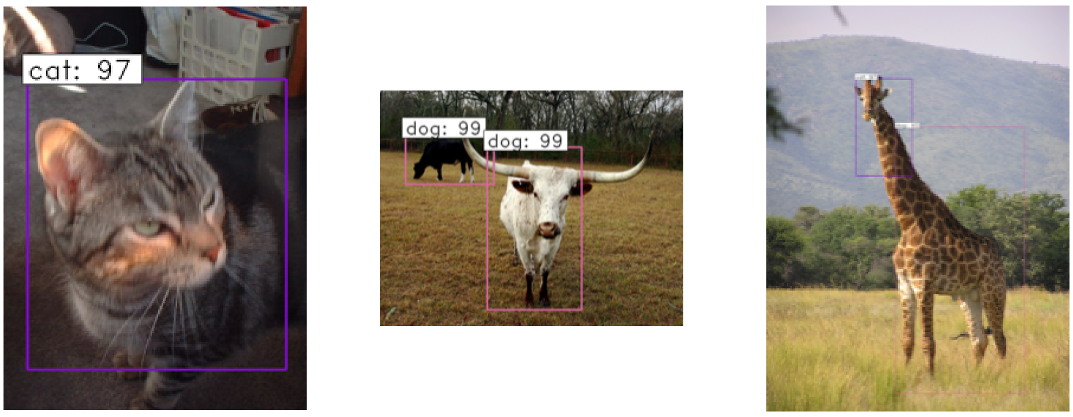


Figure 5: Detections done on 3 images by the model trained on 3 classes

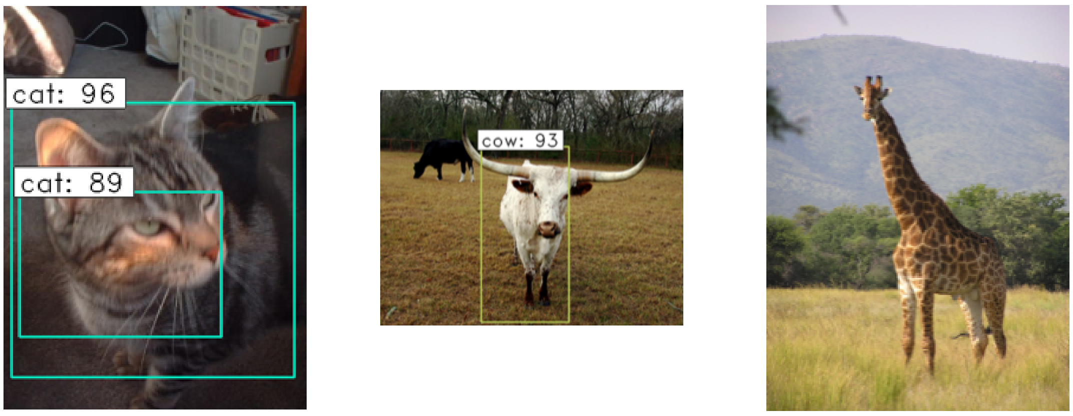


Figure 6: Detections done on 3 images by the model trained on all classes

7 Discussion

7.1 3-class vs. all-class

When comparing the 3-class model and all-class model by looking at the visualized detections, we can conclude that the 3-class model is predicting marginally better on the 3 chosen classes. The all-class model often overestimates the number of bounding boxes in those cases. Apparently, because the 3-class model can focus all its attention on a smaller variation of data, it can predict better on that type of data. However, when showing it pictures of classes it is not supposed to detect, it still detects those classes. Whereas, the all-class model does not detect, e.g. the giraffe, as something else. This is an indication that the all-class model has learned to better recognize what is really one of the classes and what not. This makes sense, since the all-class model has seen a wider variation of data during training. However, when looking at the convergence of the loss it might be the case that the all classes model will also outperform the 3-class model on the 3 classes when training for more epochs.

7.2 Loss functions

From the charts, it seems focal loss improves RPN significantly. However, as explained in the loss function section, focal loss aims to minimize loss for common classes to help the network generalize rare classes better. Naturally, this leads to lower loss in general, but does not indicate better accuracy. However, we can see from the accuracy of the RPN classifier that the Focal loss helps there. Moreover, we do notice a slight decrease in loss for the detection phase in the focal-loss model. This is due to different (or better) proposals from the RPN. Due to the increased accuracy of the RPN and decreased loss of the Fast R-CNN we have reasons to believe that Focal loss is improving the performance of Faster R-CNN. However, to fully judge we would still have to rely on the mean Average Precision (mAP).

8 Conclusions

In this work, we successfully implemented Faster R-CNN with focal loss in the RPN stage. The performance of our models was acceptable on the test dataset even though we trained them for only 120 epochs. Based on the results we got, we are confident that we have achieved our original goal: understand how Faster R-CNN works and the impact of focal loss on the RPN. During this project, we got more insights about one of the most efficient object detection network, its strength and weaknesses, we also understood the differences between one-stage and two-stage detectors by experimenting Faster R-CNN and RetinaNet. By implementing focal loss on RPN, we believed it improved the performance of the detection network as a whole.
We would have better metrics with mAP, but this was challenging because the VOC service cannot verify our results. We implemented our own verification script to discover that we have inconsistencies in data between images and groundtruth annotations, thus making VOC and our verification script fail to run.
All in all, we learned a lot and are happy with what we have achieved during this mini project.

References

[1] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[2] R. Girshick, "Fast r-cnn," *arXiv preprint arXiv:1504.08083*, 2015.

[3] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *arXiv preprint arXiv:1708.02002*, 2017.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[5] J. Hosang, R. Benenson, P. Dollár, and B. Schiele, "What makes for effective detection proposals?" *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 4, pp. 814–830, 2016.

[6] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," *arXiv preprint*, vol. 1612, 2016.

[7] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision.* Springer, 2016, pp. 21–37.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[9] Y. Henon, "Keras frcnn," https://github.com/yhenon/keras-frcnn, 2017.

[10] 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.

9 Roles of the authors

Together we decided upon our goals and the plan. While meeting regularly, we divided the work as follows:

Tinka (the data tinker) Valentijn

- Inspect and prepare data for training and testing
- Understand and clarify research papers to the team
- Visualize the architecture for the networks
- Test the models

Phu (with Tritonjiujutsu) Pham

- Inspect, explain, and reimplement Faster R-CNN in Keras
- Train and test models on Triton
- Collect and analyze test results

Kha (fast as a car) Nguyen

- Implement focal loss for RPN
- Inspect and explain loss functions for Faster R-CNN and RetinaNet
- Implement accuracy metrics collector