LapNet : Automatic Balanced Loss and Optimal Assignment for Real-Time Dense Object Detection

Florian Chabot Mohamed Chaouch Quoc Cuong Pham
CEA, LIST, Vision and Learning Lab for Scene Analysis
florian.chabot@cea.fr

Abstract

Several modern deep single-stage object detectors are really effective for real time processing [19, 20, 21, 22] but still remain less efficient than more complex ones [17, 28]. The trade-off between model performances and computing speed is an important challenge, directly related to the learning process. In this paper, we propose a new way to efficiently learn a single shot detector providing a very good trade-off between these two factors (Figure 1). For this purpose, we introduce LapNet, an anchor based detector, trained end-to-end without any sampling strategy. Our approach focuses on two limitations of anchor based detector training: (1) the ambiguity of anchor to ground truth assignment and (2) the imbalance between classes and the imbalance between object sizes. More specifically, a new method to assign positive and negative anchors is proposed, based on a new overlapping function called "Per-Object Normalized Overlap" (PONO). This more flexible assignment can be self-corrected by the network itself to avoid the ambiguity between close objects. In the learning process, we also propose to automatically learn weights to balance classes and object sizes to efficiently manage sample imbalance. It allows to build a robust object detector avoiding multi-scale prediction, in a semantic segmentation spirit.

1. Introduction

Object detection has been widely studied and remains a research field of interest for scene understanding application. It needs to face several challenges: small object detection, hardly occluded objects and real time processing. Modern detectors are based on deep convolutional neural networks (CNN) which have shown their effectiveness in several computer vision tasks (classification, segmentation, detection...). These object detectors can be mainly divided into two categories: two-stage and one-stage detectors. Two-stage detectors such as [10, 23] first propose a set of region of interest which are then extracted on deep features map and classified. These methods perform well, but they are time consuming because of the cascaded pipeline. In contrast, one-stage detectors [20, 21, 19, 17, 28] directly provide detection boxes without region extraction and refinement step allowing, for some of them [20, 21, 22, 19], fast processing time in inference phase. However, light and fast single shot methods do not perform as well as heavy single-stage or two-stage ones.

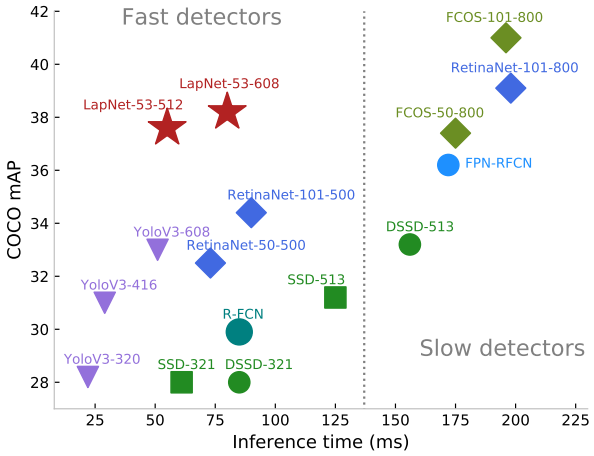

Figure 1. Speed/accuracy trade-off for single shot based detectors on COCO `test-dev` 2017 and TITAN X GPU. We only plot detectors with a computing time less than 200 ms. LapNet gives significant better results than YoloV3 [22], using the same backbone network (Darknet53 [22]) and almost the same computing time. With an input resolution of 608x608, LapNet gives similar performances than RetinaNet [17] with a smaller inference time.

Recent object detection algorithms generally use a set of discrete pre-defined boxes called anchors. The network predicts if an anchor contains an object (positive anchor) or not (negative anchor) as well as regressing the anchor coordinates to fit the object. Generally, the strategy for finding positive and negative anchors during training is based on the absolute overlap criterion: an anchor is considered as positive if a ground truth box overlaps the region above a certain threshold. This thresholding is a way to define what

an object is and what it is not. However, this strategy could be discussed. For instance, due to the discrete property of the anchor set, there is some chance that no anchor overlaps a ground truth box above the threshold, especially for small objects. In addition, an anchor between two objects is assigned only to one of them, even if the anchor highly overlaps both of them. It introduces an ambiguity perturbing model training.

Another general issues when training object detectors are classes imbalance and object sizes imbalance. Better represented classes (large number of training samples) and large objects are generally best trained. Positive anchors are more easily associated to large objects and dominate the loss function. A standard way to address imbalance is positive/negative sampling strategy [23, 19, 8] based on heuristic and hyperparameters tuning. Another way to solve it is to design a specific loss function such as the focal loss [17] which explicitly manages the balance. The objective of the focal loss is to minimize the influence of easy samples (*i.e* background samples) in the total loss. It balances foreground and background samples in an automatic hard data mining way, but it does not provide a global weighting function which explicitly takes into account class representativeness and object sizes.

In this paper, we introduce LapNet, a new single-stage object detector combining real time and efficient detection. LapNet is based on a CNN encoder-decoder architecture which takes as input an image and a set of predefined anchors. It outputs a dense set of bounding boxes with associated per-class probabilities. Our approach focuses on two main contributions.

First, we propose a more flexible way to choose positive/negative anchors. As stated above, the standard absolute overlap (AO) criterion is limited by the number of anchors. To overcome this limitation, we introduce Per-Object Normalized Overlap (PONO). PONO is the overlap between an anchor and a given ground truth box, normalized by the maximum overlap between this ground truth and all anchors. In other words, each ground truth box has at least one anchor which has a PONO value equal to one. It has two advantages compared to the standard absolute overlap criterion. The first advantage is an increased robustness when applying a threshold to select positive and negative anchors. In the case of absolute overlap, when a ground truth box has too small AO values with all the anchors, it is discarded. PONO solves this issue by design. The second advantage is that PONO normalizes matching score without being dependent of matching quality: all objects are given equal importance regardless of their relative absolute overlap values. The finer analysis of the overlapping gives its name to the method (LapNet). In addition, a strategy allowing the network to self-correct the positive and negative anchor assignment during training is proposed. The idea is that a positive anchor could highly overlap almost equally several objects. When such an ambiguity happens, it is difficult to select the right object. Furthermore, the regression part of the network will not be able to fit the correct object. To solve this issue, we propose to consider the anchor as negative using an online ambiguity weighting strategy.

The second contribution is related to the loss function. We propose to densely optimize the network without any sampling strategy. As already discussed, dense optimization for object detection is still difficult because of large imbalance of classes and object sizes. The authors of [14] proposed the homoscedastic loss for automatic loss balancing in the context of multi-task learning. In our work, we generalize this loss function to address general imbalance. In particular, we extend this weighting strategy to automatically learn weights to balance classes and to balance object sizes. Our new methodology is a generic alternative to multi-scale feature pyramid detection process always used in recent litterature [16, 17, 28, 22]. These methods use Feature Pyramid Network (FPN [16]), which consists in predicting boxes for several features levels (multi-scale prediction). During training, FPN based approach balances object size in the network architecture itself: large objects are optimized in the lower pyramid level and small objects in the higher pyramid level. It forces to define which anchors correspond to each level. In this work, we propose a generic weight learning that avoids the use of multi-scale detection and shows its effectiveness.

The two contributions of this work bring LapNet to be comparable to high performing object detectors with a significant reduction of inference processing time.

2. Related work

Given an input image, an object detector returns a list of bounding boxes and associated confidence scores for each class. The main idea is to use a model discriminating image regions to predict regions corresponding to objects. A standard way to scan regions with the model is the sliding window scheme which has been widely used by reference works [30, 7, 4]. With the advent of Deep Learning in computer vision community, sliding window and hand-crafted features for object detection have been surpassed by CNN based methods.

Anchor based detectors. Most CNN based methods [23, 19, 21, 17] use anchors for object detection. These pre-defined boxes are supposed to be representative of the shape of all the objects in terms of scale and ratio. Each 2D position in the CNN output map corresponds to one anchor at this position. The objective is to predict if an anchor contains an object or not as well as the offsets to apply on the anchor to fit the object. During training, positive and negative anchors are chosen based on the overlapping

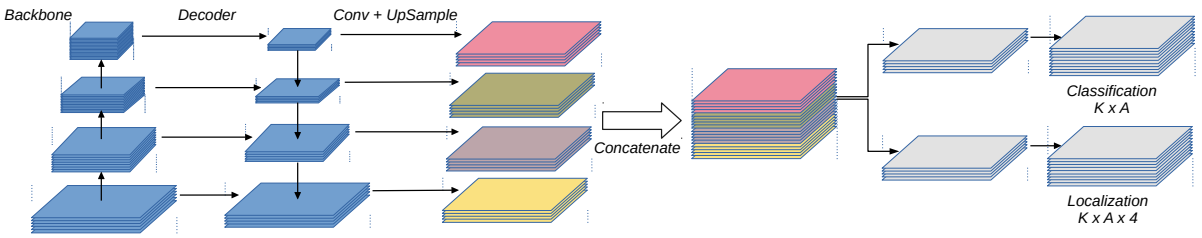

Figure 2. LapNet overview. The input image is passed through a backbone network. A decoder with skip connections is then used to compute feature maps at several resolutions. Each of these features is then processed by four 3x3 convolution and resized to the size of the high resolution feature map. After multi-scale feature concatenation, we use two convolutional heads to provide the two outputs. Each head is composed by four 3x3 convolutions and one final convolutional layers for the output.

criterion. A classification loss (cross entropy) and a regression loss (smoothL1) are often used. One limitation of these methods is related to the positive/negative anchor assignment. Because of the absolute overlapping criterion, some ground truth boxes could not have high overlapping anchors. In this case, anchors around the ground truth are considered as negative. In this paper, we propose to solve this limitation using a more flexible overlapping criterion and an ambiguity weighting allowing less sensitivity to the number of pre-defined anchors.

Two-stage vs single-stage detectors. The first two-stage based detector has been proposed by the authors of [11]. This work introduces the object proposal concept: a multi-scale segmentation algorithm [29] is executed to propose several interesting boxes. These regions are then extracted in the image and forwarded throught a CNN, predicting class scores and regression offsets. This work has been extended in [10], which extracts region directly on deep feature maps (using ROI Pooling) which is more efficient and faster. In 2015, Faster RCNN [23] is introduced, combining both anchor based object proposal generation and region classification in the same deep network. Since, Faster RCNN has been well studied and improved [16, 3, 2] making it the reference method for two-stage detection. However, cascaded methods such as Faster RCNN are time consuming in inference phase. It is why researchers are interrested in single-stage method which has been first introduced in the SSD paper [19]. The authors propose to bypass the region extraction used in two-stage detection system. As a result, the network directly outputs a dense set of boxes and associated per class scores. Since, many approaches extend single shot detection system such as [20, 21, 8, 17, 28]. The main advantage of these methods is processing time: with a reasonable input resolution and a light backbone network, some of these detectors access to real time processing [19, 20, 21, 22]. However, it still exists an important performance gap beetween fast single shot detector and heavy ones such as RetinaNet [17] or the recent FCOS [28]. With LapNet, we provide strong detection results while keeping real time processing.

Sample imbalance. Class imbalance is a well known issue which is not specific to object detection. For instance, Dice loss [26] is often used in medical image segmentation to address large class imbalance. In object detection, object scales and class representativeness are two important things to take care of. To not focus on large objects, some approaches use sampling strategy [23, 19, 8] or bootstrapping [24]. In [17, 28], the focal loss automatically balance foreground and background giving more weight to hard samples. In our work, a generic method for both classes and object scales imbalance is proposed. To our knowledge, no method taking balance as trainable parameters exists. Our approach is the generalization of the homoscedastic loss proposed by [14], in the context on multi-task learning. We hope that it will help further researches on the general problem of sample imbalance.

3. LapNet

3.1. Overview

Given an input image, LapNet predicts a dense set of bounding boxes for each class and the associated confidence scores. These boxes are then filtered using non-maximum suppression to provide final detections. As in [23, 21, 16, 19], LapNet uses anchors corresponding to predefined boxes. More specifically, the network returns (1) the probability of an anchor to contain an object of a specific class and (2) the box offsets to apply on the anchors to fit the objects.

Per-class anchors. Anchors are computed on the training dataset with the K-means based clustering proposed in [21]. Contrarily to related work, anchors are computed for each class independently. In this way, anchors are more specific and closer to the bounding box distribution of the given class. It makes sense in several context, for instance in traffic scene analysis where the different classes do not have the same box ratio distribution (”car” and ”person” classes

do not share the same box shape distribution). We define K as the number of class and A the number of anchors for each class. To simplify, A is the same for each class but in practice, the number of anchor could be different from a class to another. With this formulation, LapNet returns $K \times A$ score maps and $K \times A \times 4$ box offsets (corresponding to the four offsets to apply on an anchor to fit the object *i.e* (dx, dy, dw, dh) as in [10, 23, 17]).

Architecture. LapNet is a fully convolutional network based on an encoder-decoder architecture with skip connections. This design allows to keep important information related to object resolution and contextual information. Unlike FPN based approaches [16, 17, 22], detection is not performed on each resolution of the decoder. Instead, each resolution is passed through four convolutional layers and resized to the size of the high resolution feature map. A concatenation of these feature maps is then performed. The resulting feature is forwarded into a classification head and an offset head to predict outputs. The Figure 2 illustrates the LapNet architecture. This architecture is closer to segmentation network than recent detection network because it does not perform multi-level prediction. Avoiding multi-scale prediction removes ambiguity in ground truth to pyramid level assignment during training. With the strong proposed training process, we show that multi-scale prediction is not necessary.

3.2. Dense ground truth representation

As in previous work, ground truth boxes have to be transformed into something trainable. In particular, LapNet requires dense ground truth targets which are used to compute the loss function. This section explains how the per-object normalized overlap (PONO) is computed. The idea behind is to propose a relative overlapping measure which does not miss small or hardly occluded object when assigning positive and negative anchors. A grid of anchors is first computed using pre-defined anchors (Figure 3b). It consists in a grid of size $H_f \times W_f \times K \times A \times 4$ where H_f and W_f are the height and width of the output feature map. A and K are the number of anchor per class and the number of class respectively.

Anchor to ground truth assignment. Each element of the anchor grid is assigned to a ground truth box using the overlapping criterion: an anchor is assigned to the ground truth box with the higher overlap. This assignment forms assignment clusters where each element of the cluster corresponds to the same ground truth (Figure 3c).

Per-Object Normalized Overlap. Assignment clusters are then used to compute the PONO map O. For each cluster, the higher overlap is computed. Each element of the cluster is divided by its corresponding maximum. With this per-object normalization, the resulting map has at least one element of value 1 for each object (Figure 3d).

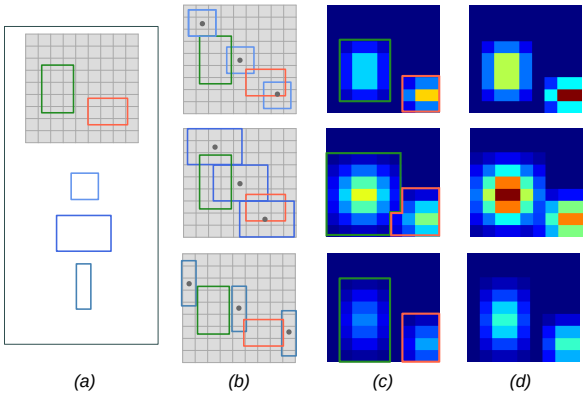

Figure 3. Per-object normalized overlap computation. *(a)* An input image with two ground truth bounding boxes (top) and three predefined anchors (bottom). *(b)* Dense anchor grid representation. *(c)* The anchor to ground truth assignment using overlap, cluster colors (green and orange) correspond the ground truth assigned to each anchor of the grid. *(d)* Resulting per-object normalization overlap. The two ground truth boxes have at least one anchor which has its PONO value equal to one (red color).

At this stage, the PONO map O and the assignment has been computed. It can be seen as an object relative overlapping map which performs a kind of anchor ranking. In other words, the PONO value is defined depending on all anchor overlaps for a given object, resulting in an invariance to object sizes. Thus, the PONO allows to not count as negative small or hardly occluded objects. It will be used as input in the training process described in the next part.

3.3. Balanced training

To train LapNet, two balanced loss functions are used: a localization loss $\mathcal{L}_{loc}$, and a classification loss $\mathcal{L}_{cls}$. They use the pre-computed per-object normalized overlap map O described in the above section. The localization loss is used to regress offsets allowing to fit an anchor to its associated ground truth box. The classification loss is used to predict if an anchor contains an object of a specific class. In this work, we propose a weighted variant of these losses to avoid class and object size imbalance. Manually finding weights to apply on samples or losses for better training is time consuming because hand-crafted weights or sampling strategy require a lot of hyper-parameters tuning. In the following, we describe the general training process (Figure 4) and loss functions. Then, we explain our balanced training strategy which is a fully automatic way to find weights to balance losses, classes and object sizes.

3.3.1 Pixel-wise localization loss

In LapNet, the localization loss uses the overlapping criterion to train box offset regression task. Contrarily to pre-

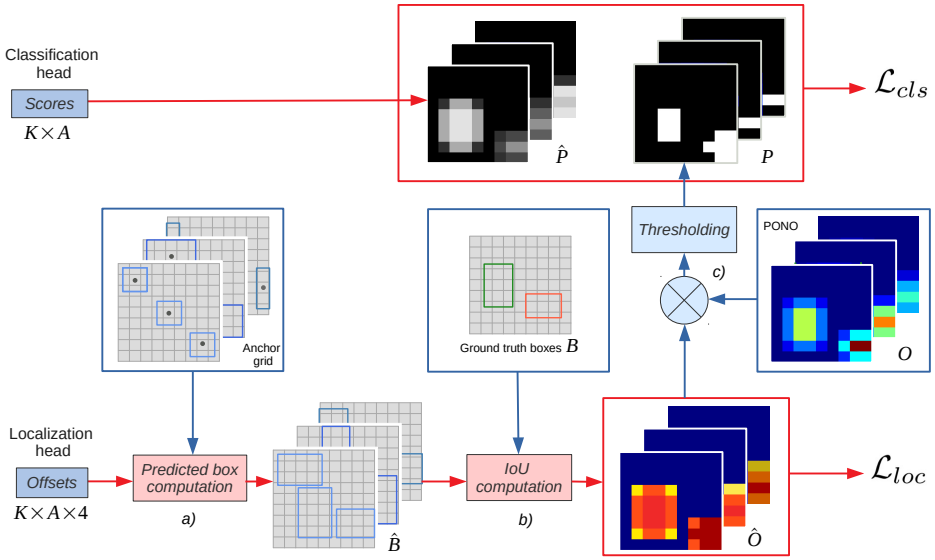

Figure 4. LapNet training. Given an image, Scores and Offsets are the outputs of the network. a) Offsets are applied to the anchor grid that outputs predicted boxes. b) An IoU map $\hat{O}$ is then computed using predicted boxes and associated ground truth boxes. $\hat{O}$ is used to compute the localization loss $\mathcal{L}_{loc}$. c) The thresholded product between the PONO map O and the predicted IoU map $\hat{O}$ gives classification labels to optimize the classification loss $\mathcal{L}_{cls}$. Back-propagation is performed on red flows. Blue flows represent steps for label target computing.

vious anchor based works [23, 21, 20, 17] which directly train the box offsets using SmoothL1 loss, we propose to train box offsets in a latent way. In other words, we do not directly optimize the transformation parameters, but we use the IoU function to find them indirectly. IoU function is a bounded function and is more stable when predicted offsets have to be high. This is particularly true in our case: we use a relative overlap (PONO) to select positive anchors and because of that, positive anchors could be "far" from the object.

With the offsets predicted by the network, predicted boxes (Figure 4a) are generated. It is followed by an IoU computation between predicted boxes and previously assigned ground truth boxes (Figure 4b). For a given class c, a given anchor a at location (i, j), we define the IoU function as follow:

$$\hat{O}_{c,a,i,j} = IoU(B_{c,a,i,j}, \hat{B}_{c,a,i,j}) \tag{1}$$

Where $B_{c,a,i,j}$ is the ground truth box associated to the anchor a of the class c at position (i, j). $\hat{B}_{c,a,i,j}$ is the box predicted by the anchor $i.e$ the anchor with applied offsets. With this notation and the pre-computed PONO map O, we define the pixel-wise localization loss function as follow:

$$\mathcal{L}_{loc}(c, a, i, j) = \begin{cases} \|1 - \hat{O}_{c,a,i,j})\|^2, & \text{if } O_{c,a,i,j} > 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

This loss function allows to learn latent offsets so that the overlapping beetween a predicted box and its associated ground truth box is one. Using IoU function instead of offset regression allows to better fit ground truth boxes. Some works already use it [28, 32] in case of pixel-to-box regression but it is not used for anchor based detector.

3.3.2 Pixel-wise classification loss

A standard way to define labels for classification loss is to threshold the PONO map as in localization loss. As explained in introduction, directly thresholding an absolute or per-object normalized overlap to choose positive and negative anchors could lead to ambiguities for anchors between two close objects. To avoid it, we propose to multiply the PONO map O by the predicted geometric overlap $\hat{O}$ (Figure 4c). For a given anchor a of the class c at position (i, j), we define positive and negative labels as follow:

$$P_{c,a,i,j} = \begin{cases} 1, & \text{if } O_{c,a,i,j} \times \hat{O}_{c,a,i,j} > 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

In case of ambiguity, if the network cannot fit correctly an anchor to its ground truth (because it is too hard or ambiguities between two objects exist), this product will return a low score at this position. So, even if the PONO score is above the threshold of 0.5, the product will return a value under that threshold. This particular anchor is then considered as a negative one. The pixel-wise classification loss is then defined by:

$$\mathcal{L}_{cls}(c, a, i, j) = CE(P_{c,a,i,j}, \hat{P}_{c,a,i,j}) \tag{4}$$

where CE is the standard binary cross-entropy and $\hat{P}_{c,a,i,j}$ is the probability than the anchor a of the class c at position (i, j) contains an object of the class c. In LapNet, the sigmoid activation function is applied on logits to get $\hat{P}$.

3.3.3 Balanced losses

One important challenge in training multi-class models for object detection or segmentation is the class and object scale imbalance. Underrepresented classes in the training dataset could not be learned correctly because of well represented classes. In the same way, large objects dominate the loss function and prevent small objects to be trained efficiently. To solve it, several methods use sampling strategy [23, 16], multi-scale detection head [16, 17, 28, 22], hand-crafted weights [33, 1] or specific loss [17].

In our work, we propose to automatically learn balance weights. Inspired by [14], that is interested in loss balancing for multi-task training, we learn two kinds of weights: loss weights (LW) and anchor-class weights (ACW). Loss weights are used to balance the different loss functions in the total cost in the spirit of [14]. Our main contribution here, is the anchor-class weights allowing to balance classes and object sizes. The idea behind is to consider each anchor-grid map c, a loss independently. In this way, we extend the homoscedastic loss [14] for anchor-class balancing. More formally, we propose the following weighted losses:

$$\mathcal{L}_{loc} = \lambda_{loc} \frac{1}{N^+} \sum_c \sum_a \lambda_{loc}^{c,a} \sum_{i,j} \mathcal{L}_{loc}(c, a, i, j) \quad (5)$$

$$\mathcal{L}_{cls} = \lambda_{cls} \frac{1}{N} \sum_c \sum_a \lambda_{cls}^{c,a} \sum_{i,j} \mathcal{L}_{cls}(c, a, i, j) \quad (6)$$

where λ_{cls}, λ_{loc} are the loss weights and $\lambda_{cls}^{c,a}$, $\lambda_{loc}^{c,a}$ are the anchor-class weights, defined for a given anchor a and a class c. N^+ is the number of positive anchors for the localization task and N is the total number of pixels. In this formulation, all λ weights are trainable variables and we add a regularization loss $\mathcal{L}_{reg}$ to avoid $\lambda = 0$:

$$\begin{aligned} \mathcal{L}_{reg} = & log(\frac{1}{\lambda_{cls}}) + log(\frac{1}{\lambda_{loc}}) \\ & + \frac{1}{KA} \sum_c \sum_a log(\frac{1}{\lambda_{cls}^{c,a}}) + log(\frac{1}{\lambda_{loc}^{c,a}}) \end{aligned} \quad (7)$$

In practice, the network optimize the variables $s = log(\frac{1}{\lambda})$ in the regularization loss which allows better numerical stability and avoid weights to be negative ($\lambda = e^{-s}$). In addition, if the anchor grid corresponding to the anchor a of the class c does not have a least one positive association, corresponding anchor-class weights $\lambda_{cls}^{c,a}$ and $\lambda_{loc}^{c,a}$ are not updated. This is important to avoid huge weights which could bring the network to diverge. In fact, the objective of anchor-class weights is to balance the loss of each anchor grid map. If an anchor grid map only has negative samples, it is an easy case for the network and thus its loss will be very small. In this case, the anchor-class weight increases too much and too fast compared to the other anchor-grid maps containing positive samples.

Finally, the total loss for LapNet training is the sum of the localization, classification and regularization loss. In the next section, we demonstrate the power of this method to efficiently train object detection models.

4. Experiments

We propose some experiments to show the effectiveness of the LapNet training process. In particular, we provide results on two challenging datasets (PASCAL VOC [6], MSCOCO [18]) and ablation studies for our contributions.

Training details. In all experiments, SGD with momentum of 0.9, initial learning rate of 0.005 and polynomial decay policy with power of 0.9 are used. For data augmentation, random scale and horizontal random flipping are applied. We also fix arbitrarily the number of anchors per object to $A = 10$. The trainable variables $s = log(\lambda)$ are all initialized to 1 (we saw that the initial value does not influence performances). Two pre-trained backbone networks, Darknet53 [22] and Inception-Resnet-V2 [27] are used for experiments, both pre-trained on ImageNet [5]. We use 4 GPU resulting in a global batch size of 28 and 16 for Darknet53 and Inception-Resnet-V2 respectively. For Darknet53, the decoder structure proposed by YoloV3 [22] is used and for Inception-Resnet-v2, the decoder structure proposed by [17] is plugged into the backbone. In both cases, the spatial output size of the network is eight times smaller than the input image (feat stride = 8).

4.1. PASCAL VOC

This dataset contains 20 classes of objects and LapNet is trained using the union of 2007 and 2012 `trainval` and tested on 2007 `test`. 120K iterations are runned to get the final result shown in Table 1. We can see that LapNet increases detection results compared to previous works for both backbone networks. In particular, LapNet gives better results compared to the stronger method DSSD [8] which uses multiple training stages to get the final detector. With LapNet, the training is done end-to-end. In the following, an ablation study based on PASCAL VOC is proposed.

PONO and assignment ambiguities. Table 2 gives results showing the effectivness of our Per-Object Normalized Overlap and our ambiguity weighting method. All models are trained using loss weights (LW), anchor-class weights (ACW) and the Darknet53 backbone [22]. In the first column of Table 2, we compare the PONO assignment with the standard absolute overlap (AO) used in many

method	backbone	mAP	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Faster [23]	VGG-16	73.2	76.5	79.0	70.9	65.5	52.1	83.1	84.7	86.4	52.0	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6
Faster [23, 12]	Resnet-101	76.4	79.8	80.7	76.2	68.3	55.9	85.1	85.3	89.8	56.7	87.8	69.4	88.3	88.9	80.9	78.4	41.7	78.6	79.8	85.3	72.0
MR-CNN [9]	VGG-16	78.2	80.3	84.1	78.5	70.8	68.5	88.0	85.9	87.8	60.3	85.2	73.7	87.2	86.5	85.0	76.4	48.5	76.3	75.5	85.0	81.0
R-FCN [3]	Resnet-101	80.5	79.9	87.2	81.5	72.0	**69.8**	86.8	88.5	89.8	**67.0**	88.1	74.5	89.8	90.6	79.9	81.2	53.7	81.8	81.5	85.9	79.9
YOLOv2 [21]	Darknet-19	78.6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SSD300 [19]	VGG-16	77.5	79.5	83.9	76.0	69.6	50.5	87.0	85.7	88.1	60.3	81.5	77.0	86.1	87.5	83.97	79.4	52.3	77.9	79.5	87.6	76.8
SSD512 [19]	VGG-16	79.5	84.8	85.1	81.5	73.0	57.8	87.8	88.3	87.4	63.5	85.4	73.2	86.2	86.7	83.9	82.5	55.6	81.7	79.0	86.6	80.0
SSD513 [8]	Resnet-101	80.6	84.3	87.6	82.6	71.6	59.0	88.2	88.1	89.3	64.4	85.6	76.2	88.5	88.9	87.5	83.0	53.6	83.9	82.2	87.2	81.3
DSSD513 [8]	Resnet-101	81.5	86.6	86.2	82.6	74.9	62.5	89.0	88.7	88.8	65.2	87.0	78.7	88.2	89.0	87.5	83.7	51.1	86.3	81.6	85.7	83.7
LapNet512	Darknet-53	81.7	88.1	88.7	82.0	75.0	65.8	88.1	**91.7**	90.0	65.1	85.9	77.4	88.5	90.8	86.1	**86.2**	51.5	82.7	81.8	89.2	79.4
LapNet512	IncResV2	**83.2**	**89.8**	**89.8**	**83.8**	**76.1**	65.2	**89.8**	90.7	92.0	64.6	**89.8**	**79.0**	**91.8**	**91.8**	**89.5**	84.9	**53.5**	**86.3**	**82.4**	**89.6**	**84.7**

Table 1. Results on the VOC 2007 `test` set. Two-stage detectors are on the top of the table and single-stage at the end. Compared to both detector family, LapNet increases detection results.

Ambiguity Weighting	no	yes
AO	80.3	78.4
PONO	81.1	**81.7**

Table 2. Influence of PONO and geometric ambiguity weighting. AO is the standard absolute overlap. The bottom right result corresponds to the complete LapNet model which outperforms other kind of positive/negative assignment.

λ_{loc}	λ_{cls}	$\lambda_{loc}^{c,a}$	$\lambda_{cls}^{c,a}$	Loss	mAP
fixed = 1	fixed = 1	fixed = 1	fixed = 1	CE	16.4
fixed = 1	fixed = R	fixed = 1	fixed = 1	CE	79.0
fixed = 1	fixed = R	fixed = 1	fixed = 1	FL	71.4
learned	learned	fixed = 1	fixed = 1	CE	81.3
learned	learned	learned	learned	CE	**81.7**

Table 3. Influence of automatic weights learning in the loss equation (5) and (6). We provide results for training variant (handcrafted fixed weights or learned weights) on the four kind of weights. R represents the normalization used for classication loss in RetinaNet [17], see text for detail. CE is the cross entropy and FL the focal loss. The last line corresponds to the proposed LapNet training process.

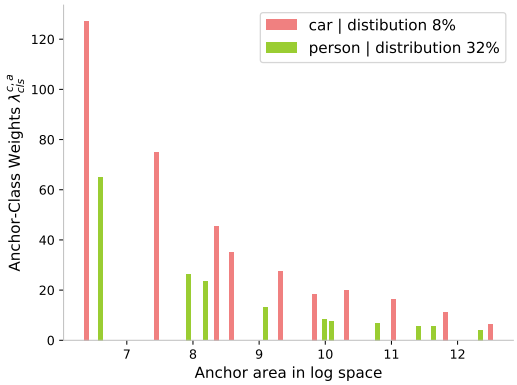

Figure 5. Automatic learned weight analysis on VOC2007-2012 `trainval`. Anchor-class weights of two classes are plotted here (car and person), with $A = 10$. The number of samples over the entire training dataset is reported in the top-right corner. This plot shows that weight values decrease with the anchor size and decrease with the label distribution.

previous methods [23, 17, 19, 8]. Importantly, we see that PONO works better than absolute overlapping (81.1 vs 80.3) which demonstrates that it is a stronger indicator for positive/negative assignment. The last column of Table 2 provides results when ambiguity weighting, described in 3.3.2, is added to choose positive and negative anchors. Adding ambiguity weighting to AO decreases mAP (78.4 vs 80.3). One explanation is that AO already misses some ground truth object in the assignment. Ambiguity weighting makes it worth because it makes the overlapping value lower. This issue does not appear with PONO because it is a more flexible way to associate an overlapping score to an anchor. Finally, PONO and ambiguity weighting increase results (81.7 vs 81.1) showing the effectiveness of the method. Contrarily to previous works [23, 17], we do not need to tune several thresholds to select positive/negative/ignored samples.

Automatic balance. We also study the influence of automatic weight learning for loss, class and anchor balance shown in Table 3. In these results, all models are trained using the PONO map and the ambiguity weighting for positive/negative anchor assignment. The first row is a model where no trainable weights are used, fixed to one. We observe that the model completely failed. The main reason comes from the normalizer N in the equation of the classification loss (5) which is an important parameter. In this work, N is simply the number of pixel (anchors) in the outputs. In other words, we just take the average over all pixels without any focus on positive anchors unlike RetinaNet [17]. The resulting classification loss is therefore very low because of the large number of easy negative samples leading the network to not be trained efficiently. In the second row, weights are also not trainable but we use the normalization method used in RetinaNet [17]. They propose to divide the classification loss by the number of anchor having an overlapping score greater than zeros (noted A^+). With this formulation and the equation (5), we give the constant value $\lambda_{cls} = R = \frac{N}{A^+}$. That improves results, proving that the loss normalizer needs to be chosen carefully. In the third line, we repeat the same

	method	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage	Faster R-CNN w/ TDM [25]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	52.1
	Faster R-CNN w/ FPN [16]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
	Faster R-CNN by G-RMI [13]	Inception-ResNet-v2	34.7	55.5	36.7	13.5	38.1	52.0
	Faster R-CNN w/ TDM [25]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	52.1
Slow Single-stage	DSSD513 [8]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
	RetinaNet [17]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
	RetinaNet [17]	ResNeXt-32x8d-101-FPN	40.8	61.1	44.1	24.1	44.2	51.2
	CornerNet [15]	Hourglass-104	40.5	56.5	43.1	19.4	42.7	**53.9**
	FCOS [28]	ResNet-101-FPN	41.0	60.7	44.1	24.0	44.1	51.0
	FCOS [28]	ResNeXt-32x8d-101-FPN	**42.1**	**62.1**	**45.2**	**25.6**	**44.9**	52.0
Fast Single-stage	SSD513 [19]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
	YOLOv2 [21]	DarkNet-19	21.6	44.0	19.2	5.0	22.4	35.5
	YOLOv3 [22]	DarkNet-53	33.0	**57.9**	34.4	18.3	35.4	41.9
	Lapnet512x512	DarkNet-53	37.6	55.5	40.4	17.6	40.5	**49.9**
	Lapnet608x608	DarkNet-53	**38.2**	56.6	**41.2**	**20.3**	**41.6**	47.5

Table 4. LapNet comparaison results on MSCOCO `test-dev2017` with two different input resolutions. As LapNet has been designed for real-time detection, we provide results with the Darknet53 backbone. Our approach clearly outperforms previous light single-stage object detectors. To get information about time vs accuracy trade-off, see Figure 1.

experiment using focal loss [17] instead of cross entropy. Focal loss brings down results significantly showing that this loss is not enought generic and its efficiency depends on the used training dataset. Importantly, the fourth line of Table 3 shows that when using only automatic loss weights (LW) the model outperforms the model using hand-crafted weights of the second line. That proves that during training, even if classification loss is low because of dividing by N, automatic weights naturally counterbalance that and provide better results. It is an important point because it avoids hyperparameters / heuristic tuning such as burn-in strategy [17, 22] or hand-crafted loss normalizer [17] which are not generic depending on the training dataset properties. Finally, when automatic anchor-class weights (ACW) are added in the training function, that increases mAP again (Table 3, last line). To get a better visualization of these weights, the Figure 5 illustrates some learned ACW. This plot shows that larger is the anchors, smaller is its learned weight and smaller is the label distribution, larger are the weights. This is an expected behavior which could be hardly designed manually but here, all these weights are automatically learned and very efficient. In the following section, we provide results on the MSCOCO dataset to confirm the approach effectiveness and genericity.

4.2. MSCOCO

We train LapNet on the 80 object categories using the 2017 `trainval` split (115K images) and test on 2017 `test-dev` (20K images) by uploading results on the evaluation server. Results presented in Table 4 are obtained using 500k training iterations. The main result is that LapNet widely outperforms the real-time YoloV3 detector [22] using the same backbone network and the same decoder (37.6 of mAP vs 33.0). This important improvement comes from our proposed learning strategy showing that with almost the same computing time (Figure 1), LapNet gives stronger detection results. Compared to previous results on PASCAL VOC, where DSSD [8] and LapNet were compared, we observe that LapNet also outperforms largely this approach with a smaller inference time (Figure 1).

Compared to stronger but time consuming methods such as RetinaNet [17] and FCOS [28], LapNet is less efficient. Using stronger backbone such as ResNeXt [31] and high resolution input image could lead LapNet to better results. In fact, in RetinaNet [17] and FCOS [28], the input image is resized so that its smaller dimension becomes 800. In LapNet, the image is resized so that its larger dimension becomes 608 (or 512). The resizing function of these methods associated with a heavy backbone lead to better results but clearly slow down algorithms. However, we see on Figure 1 that with almost the same input resolution, LapNet clearly outperforms RetinaNet [17] (see RetinaNet-101-500 vs LapNet-53-512).

5. Conclusion

We have introduced LapNet, a real time single-shot detector which gives a very good trade-off between speed and accuracy. The training process is based on flexible positive/negative assignment based on the Per-Object Normalized Overlap and ambiguity weighting. An automatic method is also proposed to balance losses, classes and object sizes for efficient learning. We hope that the genericity of this automatic balance will inspire other research fields in computer vision. Finally, experiments show the influence of our contributions, leading LapNet to be the best compromise between real-time and object detection performances.

References

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *PAMI*, 2017.

[2] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teuliere, and T. Chateau. Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. *CVPR*, 2017.

[3] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. *NIPS*, 2016.

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *CVPR*, 2005.

[5] J. Deng, W. Dong, R. Socher, L. jia Li, K. Li, and L. Fei-fei. Imagenet: A large-scale hierarchical image database. *CVPR*, 2009.

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

[7] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *PAMI*, 2010.

[8] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. Dssd: Deconvolutional single shot detector. *arXivpreprint arXiv:1701.06659*, 2017.

[9] S. Gidaris and N. Komodakis. Object detection via a multi-region and semantic segmentation-aware cnn model. *ICCV*, 2015.

[10] R. Girshick. Fast r-cnn. *ICCV*, 2015.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR*, 2014.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CVPR*, 2016.

[13] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. *CVPR*, 2017.

[14] A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *CVPR*, 2018.

[15] H. Law and J. Deng. Cornernet: Detecting objects as paired keypoints. *ECCV*, 2018.

[16] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. *CVPR*, 2017.

[17] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollr. Focal loss for dense object detection. *ICCV*, 2018.

[18] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollr, and C. L. Zitnick. Microsoft coco: Common objects in context. 2014.

[19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. *ECCV*, 2016.

[20] J. Redmon, S. Divvala, R. Girshick, , and A. Farhadi. You only look once: Unified, real-time object detection. *CVPR*, 2016.

[21] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. *CVPR*, 2017.

[22] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXivpreprint arXiv:1804.02767*, 2018.

[23] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NIPS*, 2015.

[24] A. Shrivastava and H. Mulan. Training region-based object detectors with online hard example mining. *CVPR*, 2016.

[25] A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta. Beyond Skip Connections: Top-Down Modulation for Object Detection. *arXivpreprint arXiv:1612.06851*, 2016.

[26] C. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. *DLMIA*, 2017.

[27] C. Szegedy, S. Ioffe, V. Vanhouck, and A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *ICLR WS*, 2016.

[28] Z. Tian, C. Shen, H. Chen, and T. He. FCOS: Fully convolutional one-stage object detection. *arXiv preprint arXiv:1904.01355*, 2019.

[29] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *IJCV*, 2013.

[30] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *CVPR*, 2001.

[31] S. Xie, R. Girshick, P. Dollr, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. *CVPR*, 2017.

[32] J. Yu, Y. Jiang, Z. Cao, and T. Huang. Unitbox: An advanced object detection network. *ACM*, 2016.

[33] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. *CVPR*, 2017.