Hierarchical Head Design for Object Detectors

Shivang Agarwal, Frédéric Jurie

Hierarchical Head Design for Object Detectors

Shivang Agarwal
Normandie Univ., UNICAEN, ENSICAEN, CNRS.
email: shivang.agarwal@unicaen.fr

Frederic Jurie
Normandie Univ., UNICAEN, ENSICAEN, CNRS.
email: frederic.jurie@unicaen.fr

***Abstract*—The notion of anchor plays a major role in modern detection algorithms such as the Faster-RCNN [1] or the SSD detector [2]. Anchors relate the features of the last layers of the detector with bounding boxes containing objects in images. Despite their importance, the literature on object detection has not paid real attention to them. The motivation of this paper comes from the observations that (i) each anchor learns to classify and regress candidate objects independently (ii) insufficient examples are available for each anchor in case of small-scale datasets. This paper addresses these questions by proposing a novel hierarchical head for the SSD detector. The new design has the added advantage of no extra weights, as compared to the original design at inference time, while improving detectors performance for small size training sets. Improved performance on PASCAL-VOC and state-of-the-art performance on FlickrLogos-47 validate the method. We also show when the proposed design does not give additional performance gain over the original design.**

***Index Terms*—2D Object Detection, Computer Vision, Anchors, Deep Learning**

I. Introduction

Object detection has been revolutionized by the introduction of convolutional neural networks [3]. While the first DCNN-based object detectors were applying fine-tuned classifiers on each possible location of the image in a sliding window manner [4], or on some specific regions of interest [5], recent pipelines operate in an end-to-end way, relying on fully convolutional networks jointly selecting the regions of interests and scoring them. Two main streams of detectors co-exist at the moment, with strong influence, *i.e.* single stage detectors (*e.g.* [2], [6] and their follow-ups) as well as two-stage detectors (*e.g.* [1] and their follow-ups) which are slower but more accurate.

Both lines of research rely on the definition of so-called *anchors* [1]. Anchors are the reference boxes having multiple pre-defined positions, scales and aspect ratios, chosen to cover target object candidates of different sizes and shapes all over the image. The purpose is to learn to relate the feature maps of the networks with image bounding boxes, which allows to implement object/non-object criterion as well as bounding box regression for objects.

Generally, lots of anchors are defined at each of the spatial location of the final feature maps of the network. During training some of them are chosen as positives on the condition that they have Intersection over Union (IoU) greater than a threshold, u_h, with the candidate object. The rest are either ignored or used for classifying background. An object of certain size and aspect ratio can be learned by an anchor which is closest to its shape. But it doesn't benefit other anchors as they need to "see" the same object in dimensions which is closer to their shape in order to learn to detect it. The effect is more pronounced in the case of small-scale datasets (*i.e.* datasets with less number of total objects) where each anchor gets to learn from still smaller set of candidate objects. This situation can be improved if some of the weights can be shared among anchors. This will allow the anchors that have "seen" an object to share their trained features with the anchors that have "not seen" that object.

Now the question arises on how to share these weights? We observed that deep convolutional neural networks (DCNNs) are implicitly hierarchical by design. This property allows them to learn to make relations between salient features that are present at distant spatial locations in an image. We used this nice property of hierarchy *explicitly*, in the design of a novel detector head, which will allow the sharing of weights according to various intuitive configurations.

We hypothesize that this head should allow each anchor to have a part of the parameters, which are specific and represent the characteristics which are unique, trained together with other part of the parameters that are shared and represent common properties. The weights for the shared and the specific part of the head can be automatically adjusted during training according to the requirements of the dataset. Thus, it allows anchors to assist each other.

Moreover, the backbone networks can be pre-trained on ImageNet classification task and then used for the object detection task. But since anchors cannot be pre-trained during classification they need to be randomly initialized during training, which means they need to start learning from scratch. This strategy acts as a supplementary training for anchors which have "not seen" the objects of other shapes.

To see the applicability of this head we experimented with three datasets (FlickrLogos-47 [7], PASCAL-VOC [8] and MS-COCO [9]), that are vastly different in number of annotations and images. The amount of improvement increases with decrease in number of annotated images. We also trained with different networks (RetinaNet [10] and Inception-V2 [11]) to show that the proposed design is general and can be used with different kinds of networks.

The rest of the paper is as follows. In the next section we talk about the seminal works that have been done in the domain of anchors for object detection. In Section III we present our technique of hierarchical anchors and different possible designs, while in Section IV we experimentally validate the proposed detector head. Section V concludes the paper.

II. Related Work

Object detection is one of the most important tasks among the computer vision community, and, on this basis, has received a lot of attention. Among the recent detectors, the most successful are the RCNN [5] and variants: Fast-RCNN [12], Faster-RCNN [1], R-FCN [13] and the very recent Mask RCNN [14]. Among the single step detectors, we can also mention the SSD [2] and YOLO [6] detectors. The recent DSSD [15] approach allows to add contextual information into state-of-the-art SSD detectors, by the means of deconvolution layers introducing additional large scale context. FSSD [16] is a Feature Fusion Single Shot Multibox Detector enhancing the feature fusion mechanisms of SSD; features from different layers with different scales are concatenated together, followed by some down-sampling blocks to generate new feature pyramid. Finally, YOLO9000 [17] proposes various improvements to the YOLO detection method.

The recent literature on object detection is certainly too large to be covered exhaustively in this section. The reader interested in having more details can read the survey by Agarwal *et al* [3] and the benchmark study done by Huang *et al*. [18] comparing most of the recent detectors. Best performing detectors at the moment are RefineDet [19], RetinaNet [10], CornerNet [20] or M2Det [21].

While the literature on object detection is very large, the literature on anchors is surprisingly very limited. Anchors were first proposed by Ren *et al*. in [1] for learning region proposals, in the Region Proposal Network. They were later adopted by single-stage detectors [2] also for directly predicting the object instances without having any proposal. In most of the cases, detectors contain predefined set of anchor boxes, with different positions, sizes and aspect ratios.

Redmon and Farhadi [17] have proposed to choose anchor characteristics by running k-means clustering on the training set bounding boxes, to automatically find good anchor parameters. As a distance metric in k-means, they use a measure based on the IoU between regions. This idea has been extended in [22] where the authors complemented the standard loss function with an extra online clustering term measuring the distance between anchors shapes and ground-truth shapes, while in [23] the scale of the anchors are dynamically refined during learning. In [24] anchors are dynamically generated from box priors. However, none of these works hierarchically combine the different anchors nor share their parameters.

Anchors can be translation invariant, or not. YOLO [6] is not translation invariant, which means a lot of anchors at different positions are needed to cover the image. More recent approaches such as [1], [17], based on fully-convolutional networks, allow by construction, to share the parameters across different positions, thus avoiding to explicitly define large number of anchors with shared weights. Anchors can also share their weights across different final feature maps, as in the approach of Lin *et al*. [10].

More recently, Zhong *et al.* [22] proposed to dynamically learn anchor shapes, *i.e.* to learn what are the best center, width and height for each anchor, instead of using fixed positions/scales.

III. Hierarchical Head with Anchor Sharing

This section starts by going into detail of the functioning of the head used in literature (*e.g.* [1], [2]) and then describe the proposed hierarchical head. We assume multiple final feature maps for training the anchors (similar to the proposed methodology in SSD). We also explore the various ways in which our head can be designed based on the intuition we have developed. All the anchors and related terminology correspond to one final feature map and the head attached to it unless mentioned otherwise.

A. General Head

A head is a convolutional layer which takes a $3 \times 3 \times c_k$ (width, height and number of channels) input from each spatial location of each of the k-th final convolutional layers of the base network and the added layers, of size $m_k \times n_k$, in a sliding window manner, and produces two outputs. The first output is used for calculating classification losses and the second output is used for regression losses on a per anchor basis.

More formally, if x represents the feature layers for a given position of the sliding window and $\mathcal{A}$ represents the set of anchors at each spatial location, the head regresses the position p_{a_i} and class scores c_{a_i} of the anchor a_i, where $i \in [1, |\mathcal{A}|]$ as :

$$\begin{cases} c_{a_i} = f^c(W^c_{a_i} x + b^c_{a_i}) \\ p_{a_i} = f^p(W^p_{a_i} x + b^p_{a_i}) \end{cases} \tag{1}$$

where f^c and f^p are activation functions for classification and regression respectively.

Both SSD [2] and Faster-RCNN [1] consider each anchor as being independent and the weights $W^c_{a_i}, W^p_{a_i}$ of each anchor are learned independently. Though the weights can be shared between different feature maps, there is no weight sharing between anchors within a layer.

Each anchor a_i is assigned to a layer (*e.g.* k), and has a scale s and aspect ratio r associated with it. These two parameters define the initial starting point for regression for each anchor.

B. Hierarchical Head

In addition to the convolution layer output of a general head, in our hierarchical head, we compute convolutional output on the same $3 \times 3 \times c_k$ input from the feature map J more number of times. These J units can then be designed to partially or fully share their weights with the anchors $\mathcal{A}$ before calculating losses. In the new design Eq. 1 can be rewritten as:

$$\begin{cases} c_{a_i} = f^c(W^c_{a_i} x + b^c_{a_i} + \sum_{j=1}^{J} (W^c_{R_j} x + b^c_{R_j}) \times \mathbf{M}_{ji}) \\ p_{a_i} = f^p(W^p_{a_i} x + b^p_{a_i} + \sum_{j=1}^{J} (W^p_{R_j} x + b^p_{R_j}) \times \mathbf{M}_{ji}) \end{cases} \tag{2}$$

where R_j is the additional unit, $W^c_{R_j}$ and $b^c_{R_j}$ are the weights and biases for learning the classifier and $W^p_{R_j}$ and $b^p_{R_j}$ are

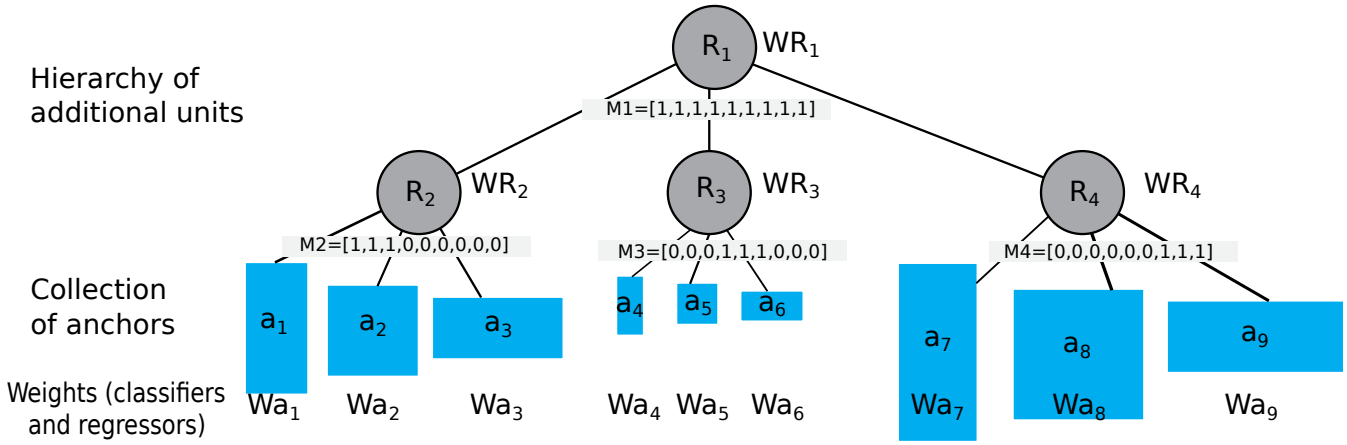

Fig. 1. Graphical presentation of the proposed hierarchical head. In the configuration shown, R_2, R_3 and R_4 are responsible for learning common characteristics between anchors of different aspect ratios and same size. R_1 is responsible for learning common characteristics for all types of anchors.

the weights and biases for learning the regressor for unit R_j. $\mathbf{M}$ is a matrix of J rows and $|\mathcal{A}|$ columns specifying the design of the hierarchical head. $\mathbf{M_{ji}} = 1$ indicates j^{th} unit and i^{th} anchor share weights and vice-versa (see Figure 1 for an illustration). An example design can be as follows:

$$\mathbf{M} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Here, a total of nine anchors are sharing features with four units. First unit, R_1, shares weights with all the anchors. R_2 shares weights with first three anchors and so on. There are a lot of designs possible in theory but we look at the ones which make an intuitive sense. Instead of randomly sharing the weights of a unit, it is more interesting to share the weights across anchors of different scales or anchors of different aspect ratios. It corresponds to the case illustrated by Figure 1.

To simplify the understanding of each configuration analyzed in this work we have clubbed all the anchors and units in different levels. All the anchors responsible for learning unique characteristics are put in a separate level. All the units that partially share the weights with some of the anchors are marked in a separate level. All the units that fully share their weights with all the anchors are marked in a separate level (Figure 2).

a) Hierarchical Head Designs: We propose many designs for matrix $\mathbf{M}$. The one already discussed is named as $9An\{9,3,1\}As$. An intuitive way to read it is a configuration trained for 9 anchors with 1 unit in the bottom layer (R_1), 3 units in the second layer (R_2, R_3, R_4) and 9 units in the last layer ($a_1, a_2 ..., a_9$) which is also the output layer of the network. 'As' indicates sharing is performed over different aspect ratios. In Figure 2, one unit (R_1) is responsible for learning characteristics that are similar to all the 9 anchors in $\mathcal{A}$ is shown at bottom level. Units R_2, R_3 and R_4 are responsible for learning characteristics that are similar to anchors along different aspect ratios are shown in second level.

TABLE I
THE DESIGNS OF MATRIX $\mathbf{M}$ USED IN THE EXPERIMENTS ALONG WITH THE CORRESPONDING CONFIGURATION NAMES. THE CONFIGURATION FOR $9An\{9,3,1\}As$ IS EXPLAINED IN SECTION III-B. OTHER DESIGNS ARE SHOWN HERE.

Configuration	M								
9An{9,3,1}S	R_1	1	1	1	1	1	1	1	1	1
	R_2	1	0	0	1	0	0	1	0	0
	R_3	0	1	0	0	1	0	0	1	0
	R_4	0	0	1	0	0	1	0	0	1
9An{9,1}Both	R_1	1	1	1	1	1	1	1	1	1
9An{9,3}As	R_1	1	1	1	0	0	0	0	0	0
	R_2	0	0	0	1	1	1	0	0	0
	R_3	0	0	0	0	0	0	1	1	1
9An{9,3}S	R_1	1	0	0	1	0	0	1	0	0
	R_2	0	1	0	0	1	0	0	1	0
	R_3	0	0	1	0	0	1	0	0	1

The units $a_1, a_2, ..., a_9$ are responsible for learning only the unique characteristics for each anchor are shown on the top.

Similarly sharing can be done over scales too, where each unit from second layer is added to each anchor in top layer with different scales but same aspect ratio. This configuration is denoted by 'S' (Table I). 'Both' denotes sharing is done over scales as well as aspect ratios. $9An\{9,1\}$'Both' is a possible example. If nothing is mentioned outside the curly brackets, it means the configuration uses a general head. All the designs for $\mathbf{M}$ used in the experiments are as described in Table I.

b) Possible Hierarchical Structures: In this paper we have explored hierarchical structures up to two levels plus the final prediction level. It is also possible to mix and merge the anchors in other ways. For instance, we can share anchors with aspect ratio greater than one in one set and fewer than one in another set. In this case the vertical objects can be shared separately from the horizontal objects. Also adding further levels can make the structure more granular. But since these other methods do not seem to have a strong intuition behind them we did not train them for this work.

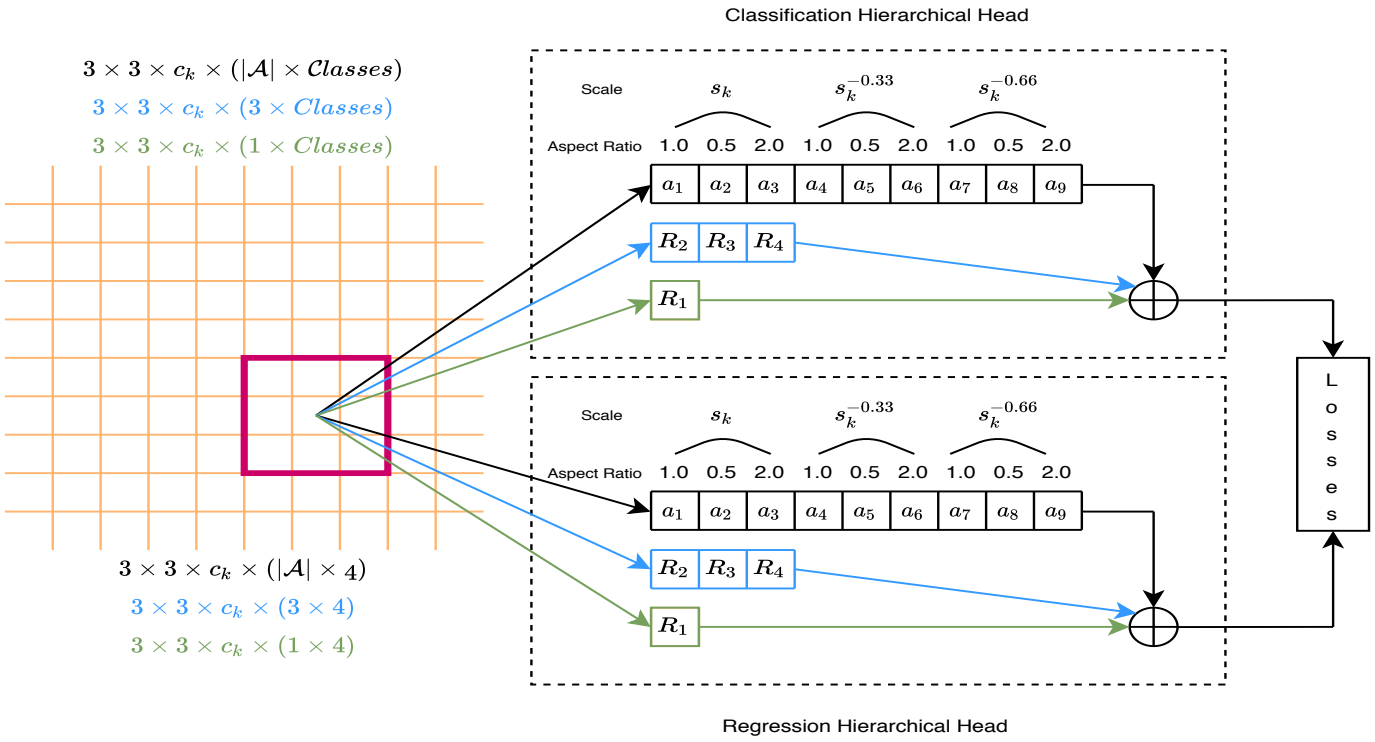

Fig. 2. Hierarchical head applied on a feature map k. J extra convolutions $(3+1)$, of same kernel size are applied in a sliding window fashion. Right-hand side of the figure shows additional units for classification as well as regression. The box shows additional units and final prediction layers for a single class and single window for the purpose of simplicity. The $\bigoplus$ box implements the design **M**. Best viewed in color.

The hierarchical design is general and can be applied to all the networks used for detection and single stage as well as two-stage detectors. In two stage detectors the same design can also be applied to generate proposals at the RPN stage [1] where objectness with a binary class label is predicted for each anchor.

c) Discussion about the total number of parameters: At first glance, Eq. 2, the hierarchical head, seems to have more parameters than the general head defined by Eq. 1. Although this is true at the time of training, this is not true during inference. Since, there are no non-linearities involved in the proposed head and it only consists of linear multiplications and additions of the extra unit(s), each anchor's individual weights can be recalculated by computing $W'_{a_i} = W_{a_i} + \sum W_{R_j} \times \mathbf{M}_{ji}$, at test time. This can be done for the classifier as well as the regressor. W'_{a_i} has the same number of parameters as W_{a_i}. The same logic holds for the biases. Under this form, the sharing can be seen as an inter-anchor regularization mechanism.

IV. Experimental Results

A. Anchor Design

In an SSD [2] type detection architecture, K feature maps are used for detecting objects. Each feature map is scanned in a sliding window approach with a kernel of size 3×3. The total number of anchor boxes per spatial location in a feature map is $|\mathcal{A}|$, where $\mathcal{A}$, the complete set of anchors, decides the number of possible predictions per position of the sliding window. To validate the hierarchical head, we trained the same configurations for each dataset as already described in Table I. Three aspect ratios $\{1.0, 0.5, 2.0\}$ with three sizes $\{s_k, 2^{-0.33} * s_k, 2^{-0.66} * s_k\}$, where $k \in [1, K]$, gives the desired nine anchors. The anchors between different final feature maps are not shared.

B. Datasets

We used three datasets for experimentally validating our proposed head. We give brief details of these datasets and present how they differ in characteristics in Table II. We present these figures after the image resizing done for experiments. For PASCAL-VOC dataset images are resized to 300×300 and 512×512. For FlickrLogos-47 images are resized to 1024×1024. The figures for MS-COCO are presented without any resizing. The table gives a clear idea on the difference of the scale of the datasets and the distribution of object instances according to aspect ratios and size.

a) PASCAL-VOC [8]: The VOC-07 trainval and VOC-12 trainval sets consist of 16551 images in total and VOC-07 test set consists of 4952 images containing 20 classes. The VOC-07+12 train set consists of 8218 images and VOC-07+12 validation set consists of 8333 images. We used VOC-07+12 train/validation sets to choose the best configuration of the hierarchical head and used VOC-07+12 trainval/test sets for

TABLE II

CHARACTERISTICS OF THE DATASETS USED. FIRST THREE COLUMNS INDICATE TOTAL NUMBER OF IMAGES, TOTAL CLASSES AND TOTAL NUMBER OF ANNOTATED OBJECTS RESPECTIVELY IN THE PASCAL-VOC07+12 TRAINVAL SET AND FLICKRLOGOS-47 TRAIN SET. THE NEXT THREE INDICATE THE NUMBER OF OBJECTS THAT ARE VERTICALLY (ASPECT RATIO LESS THAN 0.75), EQUALLY (ASPECT RATIO BETWEEN 0.75 AND 1.5) AND HORIZONTALLY (ASPECT RATIO GREATER THAN 1.5) ORIENTED. THE LAST THREE GIVES THE NUMBER OF SMALL (SMALLER THAN 32^2), MEDIUM (BETWEEN 32^2 AND 96^2) AND LARGE (GREATER THAN 96^2) OBJECTS. FIGURES ARE AFTER PERFORMING THE SAME RESIZING AS DONE FOR EXPERIMENTS (EXCEPT FOR MS-COCO).

				Aspect Ratio			**Size**		
Dataset	**# Imgs**	**# Cls**	**# Objs**	**Vertical**	**Square**	**Horizontal**	**Small**	**Medium**	**Large**
VOC (SSD300)	16,551	20	47,223	24,852	16,763	5,608	9,035	16,710	21,478
VOC (SSD512)	16,551	20	47,223	24,864	16,749	5,610	3,669	13,110	30,444
Flickr (SSD1024)	833	47	1,936	497	607	832	91	651	1,194
MS-COCO	118,287	80	860,001	367,803	297,061	195,137	267,700	300,129	292,172

TABLE III

MEAN AVERAGE PRECISION PERFORMANCE, REPORTED ON PASCAL-VOC07+12 VALIDATION SET (TRAINED ON VOC07+12 TRAIN SET) AND PASCAL-VOC07 TEST SET (TRAINED ON VOC07+12 TRAINVAL SET) RESPECTIVELY, FOR ALL THE CONFIGURATIONS. BEST PERFORMANCES ARE MARKED IN BOLD.

	SSD300		**SSD512**	
Configuration	train / val	trainval / test	train / val	trainval / test
9An{9}	64.70	72.32	67.00	75.74
9An{9,1}Both	65.75	72.75	68.33	76.63
9An{9,3}S	65.48	72.68	68.48	76.27
9An{9,3}As	**65.85**	**73.29**	67.90	76.33
9An{9,3,1}S	65.35	72.71	**68.66**	**76.60**
9An{9,3,1}As	65.78	73.15	68.24	76.14

evaluating the final performance. There are an average of 2.85 annotations per image.

b) FlickrLogos-47 [7]: The FlickrLogos-47 train set consists of just 833 images and test set consists of 1402 images distributed over 47 classes. There are an average of 2.32 annotations per image in the train set. There is no official validation set for this dataset.

c) MS-COCO [9]: The MS-COCO train set consists of 118,287 images distributed over 80 classes. There are an average of 7.27 annotations per image and 1021 images without a single annotation. Validation set consists of 5,000 images.

C. Experiments on PASCAL-VOC

a) Implementation details: We used Inception-V2 [11] as our base network on this dataset. Our code is based on the Tensorflow Object Detection API [18]. We optimized the network using RMSprop optimizer and trained for a total of $120k$ iterations for training on the trainval set. The learning rate was set to 10^{-3} in the beginning and then divided by a factor of three at $60k$, $80k$, $90k$ and $100k$ iterations. Our network was pre-trained on the ILSVRC CLS-LOC dataset [25]. For the train set a total of $60k$ iterations were used with the same initial learning rate divided by a factor of three at $30k$, $40k$, $45k$ and $50k$ iterations.

We experimented with two image sizes, 300×300 and 512×512, as in the original SSD [2] paper. They are called VOC-SSD300 and VOC-SSD512 respectively in this work. The batch size is set to 32. Random horizontal flip and SSD style random crop [2] were used for data augmentation. We used Mixed_4c and Mixed_5c as final feature maps for the box predictors to predict detection boxes. We also added four more layers. The anchors having an Intersection-over-Union (IoU) greater than 0.5 were considered to be positive anchors while anchors having IoU less than 0.5 were considered as negative ones. The losses used were sigmoid for classification and smooth L1 for localization with equal weights. We also performed Online Hard Example Mining (OHEM) [26] with a negative to positive ratio of $3:1$, which is the usual practice. During inference time detection boxes were suppressed with an IoU threshold of 0.5 using Non-Maximal Suppression (NMS). Here, the scale of the default boxes for each feature map is computed as:

$$s_k = s_{min} + \frac{s_{max} - s_{min}}{K-1}(k-1) \qquad \text{where } k \in [1, K] \quad (3)$$

where s_{min} and s_{max} are the minimum and maximum scales respectively. s_{min} and s_{max} were set to 0.2 and 0.95 respectively. All the experiments are performed using GeForce GTX 1080 Ti GPUs.

b) Results: Since we cannot ascertain in advance which configuration of hierarchical head will work best for the test set, we first use the validation set to choose the best configuration. Whichever configuration gives the best result on validation can then be used to train on trainval set and obtain final performance on test set. Table III presents the results for VOC-SSD300 and VOC-SSD512 respectively. Performance for the test set, other than the baseline and the best configuration on the validation set, are greyed out and used for informational purposes only.

VOC-SSD300 works best with $9An\{9,3\}As$ in the trainval-validation setting, therefore, we consider the performance of the same configuration on trainval-test setting. Gain over the standard head (72.32 MAP) is of 0.97 points. In case of VOC-SSD512, $9An\{9,3,1\}S$ gives best performance on validation set. We achieve a gain of 0.86 points over the standard head. This performance is not far from the actual best performance ($9An\{9,1\}Both$) in trainval-test setting. Therefore, we can use the same hierarchical configuration expecting average good performance on test set. VOC-SSD300 has more variation over different aspect ratios than different sizes (Table II), therefore, this might be the reason that sharing anchors over different aspect ratios seems to work better for it. In case of VOC-SSD512, there is more variation over scale and sharing over different scales works better. It can be seen here that the performance for a hierarchical head is always better than the general head.

D. Experiments on FlickrLogos-47

a) Implementation details: We trained RetinaNet [10] on Resnet-50 backbone without the attached sub-network on top of it. Resnet-50 is indeed the network frequently used for this dataset. Apart from the three final layers from the lateral part of the network, we used two additional coarser layers for prediction. We trained for a total of $300k$ steps starting with a learning rate of 0.0001 and decaying it exponentially by a factor of 0.8 after every $25k$ step. The images were resized to 1024×1024 and a batch size of 2 was used. NMS threshold was set to 0.1 and the scales used were of sizes $\{32, 64, 128, 256, 512\}$ for each layer respectively. The rest of the details are same as used in PASCAL-VOC experiments.

b) Results: Mean average precision performance for all the configurations have been presented in Table IV. We achieved state-of-the-art performance on this dataset to the best of our knowledge. Using a general head achieves a performance of 73.84%, but using a hierarchical head improves this performance in the range of 1.5 to 2.7 points. It can be noted that the best results are obtained when an additional unit that shares the weight with all the anchors is present. When this common unit is not used, the results drop by 0.5 to 1.5 points. A possible explanation can be that logos have a lot more likelihood of intra-class rotation and variation in sizes is already abundant from Table II.

The scale of this dataset is much smaller as compared to PASCAl-VOC. In case of less data, it is even more beneficial to share characteristics between anchors. We can see here as well that the performance for a hierarchical head is always better than the general head.

E. Experiments on MS-COCO

a) Implementation details: Our implementation for this dataset was based on the code given by Wu *et al.* [27]. We used a Retina Network [10] with ResNet-50 as the backbone with the attached sub-network on top of it. We used warm-up learning for the first 1000 iterations by increasing the learning rate linearly from 0.0002 to 0.01. Then, we decreased the learning rate after $210k$ iterations and $250k$ iterations by a factor of 10. We trained for a total of $270k$ iterations. The batch size used was 16, trained synchronously over four GPUs. We used a focal loss with $\alpha = 0.25$ and $\gamma = 2.0$. Five feature maps were used for detection with s_k set to $\{32, 64, 128, 256, 512\}$ for each feature map respectively. NMS threshold was set to 0.5. The rest of the details can be found from [27].

b) Results: Table V gives the results for all the configurations. As is evident from the table there are no major gains for different configurations and also there is no decline in the performance. The reason behind this can be that there are enough examples present for each type of anchor for each class, which is also evident from Table II. The effect of supplementary learning saturates in presence of abundant examples and there are no additional performance gains in case of hierarchical head.

TABLE IV
MEAN AVERAGE PRECISION PERFORMANCE, REPORTED ON FLICKRLOGOS-47 TEST SET (TRAINED ON FLICKRLOGOS-47 TRAIN SET), FOR ALL THE CONFIGURATIONS. BEST PERFORMANCE IS MARKED IN BOLD.

Configuration	FlickrLogos-47
9An{9}	73.84
9An{9,1}Both	76.52
9An{9,3}S	76.04
9An{9,3}As	75.37
9An{9,3,1}S	**76.56**
9An{9,3,1}As	76.42

TABLE V
MEAN AVERAGE PRECISION PERFORMANCE (COCO STYLE AP AT IOU= .50 : .05 : .95), REPORTED ON MS-COCO VAL SET (TRAINED ON MS-COCO TRAIN SET), FOR ALL THE CONFIGURATIONS.

Configuration	MS-COCO
9An{9}	37.32
9An{9,1}Both	37.25
9An{9,3}S	37.42
9An{9,3}As	37.35
9An{9,3,1}S	37.51
9An{9,3,1}As	37.33

V. CONCLUSIONS AND FUTURE WORK

This paper has proposed a novel head for SSD detectors, which allows each anchors to embed some specific information, not shared with other anchors, jointly with some more general information shared with other anchors. As a result, anchors are able to gain from the training of other anchors when the data is scarce. The proposed method was shown to have no extra features at inference time. It also improved performance on the PASCAL-VOC dataset and achieved state-of-the-art performance on FlickrLogos-47 dataset. There was no performance gain found in MS-COCO dataset, along with no decline. Therefore, the proposed head can be used in all the problems with performance gains to be expected in small-

scale datasets. The gain in the smallest dataset was found to be the highest.

In the future we would like to come up with a technique to decide the hierarchical relationship automatically during learning instead of assigning zero or one weight in the design matrix. Proposing more such explicit hierarchies for other parts of the detection model can also prove to be a worthwhile direction.

VI. Acknowledgments

This project was partly funded by the DGA through the RAPID-DRAAF project.

References

[1] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., 2015, pp. 91–99. [Online]. Available: http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks

[2] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," in *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I*, ser. Lecture Notes in Computer Science, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., vol. 9905. Springer, 2016, pp. 21–37. [Online]. Available: https://doi.org/10.1007/978-3-319-46448-0_2

[3] S. Agarwal, J. O. du Terrail, and F. Jurie, "Recent advances in object detection in the age of deep convolutional neural networks," *CoRR*, vol. abs/1809.03193, 2018. [Online]. Available: http://arxiv.org/abs/1809.03193

[4] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Is object localization for free? - weakly-supervised learning with convolutional neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society, 2015, pp. 685–694. [Online]. Available: https://doi.org/10.1109/CVPR.2015.7298668

[5] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*. IEEE Computer Society, 2014, pp. 580–587. [Online]. Available: https://doi.org/10.1109/CVPR.2014.81

[6] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 779–788. [Online]. Available: https://doi.org/10.1109/CVPR.2016.91

[7] C. Eggert, D. Zecha, S. Brehm, and R. Lienhart, "Improving small object proposals for company logo detection," in *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval, ICMR 2017, Bucharest, Romania, June 6-9, 2017*, B. Ionescu, N. Sebe, J. Feng, M. A. Larson, R. Lienhart, and C. Snoek, Eds. ACM, 2017, pp. 167–174. [Online]. Available: https://doi.org/10.1145/3078971.3078990

[8] M. Everingham, L. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, 2010. [Online]. Available: https://doi.org/10.1007/s11263-009-0275-4

[9] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," in *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, ser. Lecture Notes in Computer Science, D. J. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., vol. 8693. Springer, 2014, pp. 740–755. [Online]. Available: https://doi.org/10.1007/978-3-319-10602-1_48

[10] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 2017, pp. 2999–3007. [Online]. Available: https://doi.org/10.1109/ICCV.2017.324

[11] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 2818–2826. [Online]. Available: https://doi.org/10.1109/CVPR.2016.308

[12] R. B. Girshick, "Fast R-CNN," in *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. IEEE Computer Society, 2015, pp. 1440–1448. [Online]. Available: https://doi.org/10.1109/ICCV.2015.169

[13] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: object detection via region-based fully convolutional networks," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, Eds., 2016, pp. 379–387. [Online]. Available: http://papers.nips.cc/paper/6465-r-fcn-object-detection-via-region-based-fully-convolutional-networks

[14] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 2017, pp. 2980–2988. [Online]. Available: https://doi.org/10.1109/ICCV.2017.322

[15] C. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "DSSD : Deconvolutional single shot detector," *CoRR*, vol. abs/1701.06659, 2017. [Online]. Available: http://arxiv.org/abs/1701.06659

[16] Z. Li and F. Zhou, "FSSD: feature fusion single shot multibox detector," *CoRR*, vol. abs/1712.00960, 2017. [Online]. Available: http://arxiv.org/abs/1712.00960

[17] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 6517–6525. [Online]. Available: https://doi.org/10.1109/CVPR.2017.690

[18] 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," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 3296–3297. [Online]. Available: https://doi.org/10.1109/CVPR.2017.351

[19] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li, "Single-shot refinement neural network for object detection," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. IEEE Computer Society, 2018, pp. 4203–4212. [Online]. Available: http://openaccess.thecvf.com/content_cvpr_2018/html/Zhang_Single-Shot_Refinement_Neural_CVPR_2018_paper.html

[20] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIV*, ser. Lecture Notes in Computer Science, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., vol. 11218. Springer, 2018, pp. 765–781. [Online]. Available: https://doi.org/10.1007/978-3-030-01264-9_45

[21] Q. Zhao, T. Sheng, Y. Wang, Z. Tang, Y. Chen, L. Cai, and H. Ling, "M2det: A single-shot object detector based on multi-level feature pyramid network," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2019, pp. 9259–9266. [Online]. Available: https://doi.org/10.1609/aaai.v33i01.33019259

[22] Y. Zhong, J. Wang, J. Peng, and L. Zhang, "Anchor box optimization for object detection," in *IEEE Winter Conference on Applications of Computer Vision, WACV 2020, Snowmass Village, CO, USA, March 1-5, 2020*. IEEE, 2020, pp. 1275–1283. [Online]. Available: https://doi.org/10.1109/WACV45572.2020.9093498

[23] Q. Yuan, B. Zhang, H. Li, Z. Wang, and Z. Luo, "A single shot

text detector with scale-adaptive anchors," *CoRR*, vol. abs/1807.01884, 2018. [Online]. Available: http://arxiv.org/abs/1807.01884

[24] T. Yang, X. Zhang, Z. Li, W. Zhang, and J. Sun, "Metaanchor: Learning to detect objects with customized anchors," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 318–328. [Online]. Available: http://papers.nips.cc/paper/7315-metaanchor-learning-to-detect-objects-with-customized-anchors

[25] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li, "Imagenet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015. [Online]. Available: https://doi.org/10.1007/s11263-015-0816-y

[26] A. Shrivastava, A. Gupta, and R. B. Girshick, "Training region-based object detectors with online hard example mining," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 761–769. [Online]. Available: https://doi.org/10.1109/CVPR.2016.89

[27] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.