Hierarchical Modified Fast R-CNN for Object Detection

Arindam Chaudhuri
Samsung R & D Institute Delhi India, NMIMS University Mumbai India
E-mail: arindamphdthesis@gmail.com, arindam.chaudhuri@nmims.edu

Keywords: object recognition, image classification, Fast R-CNN, MS-COCO, CIFAR100, VisualQA

Received: September 3, 2021

In object detection there is high degree of skewedness for objects' visual separability. It is difficult to distinguish object categories which demand dedicated classification. The deep convolutional neural networks (CNNs) are trained as N-way classifiers. As such considerable work is required towards leveraging hierarchical category structures. We present here Modified Fast region-based CNN (Mod Fast R-CNN) and Hierarchical Modified Fast region-based CNN (HMod Fast R-CNN) with deep CNNs being embedded considering categorical hierarchy. The easy classes are separated through coarse classifiers. The difficult classes are classified by fine classifiers. HMod Fast R-CNN is trained by initial components training which follows fine-tuning globally using multiple group discriminant analysis. The regularization is done using coarse category consistency. For large-scale recognition tasks, scalability is done considering conditional execution of fine category classifiers and layer parameters compression. Using MS-COCO (benchmark) CIFAR100 and VisualQA datasets we obtain good results. We build several different HMod Fast R-CNN versions where standard CNNs top-1 error is reduced significantly. HMod Fast R-CNN's performance superiority with other object detectors on PASCAL VOC 2007 and VOC 2012 datasets are also highlighted.

Povzetek: Predstavljena je metoda hierarhičnih hitrih R-CNN za detekcijo objektov.

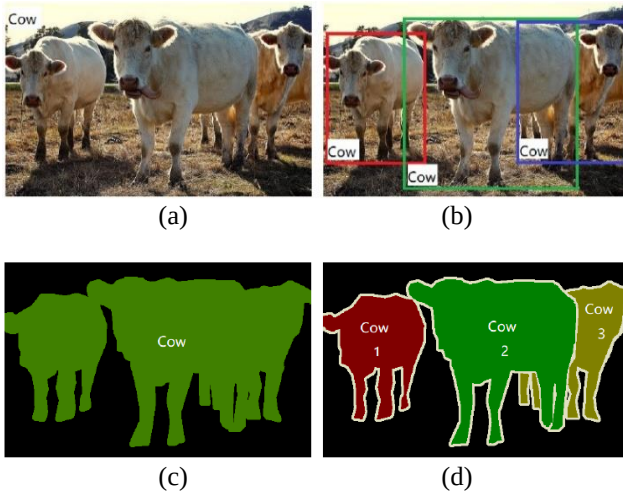
1 Introduction

In computer vision there are several fundamental visual recognition problems such as image classification [1], object detection and instance segmentation [2], [3] and semantic segmentation [4] as shown in Figure 1. Image classification recognizes objects in semantic categories from given image as shown in Figure 1 (a). Object detection recognizes object categories and predicts each object's location considering bounding box as shown in Figure 1 (b). Semantic segmentation predicts pixel wise classifiers in order to assign specific category label to each pixel. It thus provides rich image understanding as shown in Figure 1 (c). However, semantic segmentation does not distinguish between multiple objects of same category. At intersection of object detection and semantic segmentation viz instance segmentation where different objects are identified and assigned to a separate categorical pixel-level mask as shown in Figure 1 (d).

Since the birth of convolutional neural networks (CNN) image classification [5] and object detection [2], [6] problems have received a high degree of accuracy [5], [7]. Almost all available object detection techniques [2], [6], [8], [9] work in multi-stage slow and inelegant pipelines. The complexity arrives from detection which requires accurate object localization leading towards (a) processing of numerous candidate object locations and (b) achieving precise localization for candidate object locations which provide only rough localization. The solution for these problems has often struggled to achieve good speed, accuracy and simplicity.

The region-based convolutional neural network (R-CNN) [2] has achieved brilliant accuracy in object detection. However, it has certain drawbacks [2], [6], [8], [9] such that (a) training is performed through pipeline with multiple stages; (b) appreciable space and time complexity is involved and (c) object detection process happens slowly. R-CNN works slowly as each object's CNN forward pass happens without any computation sharing. By sharing computation, spatial pyramid pooling networks (SPPN) [8] speeds up R-CNN. The input convolutional image's feature map is computed by SPPN. Then each object is classified through feature vector taken from shared feature map. Considering an object, extraction of features happens through max-pooling feature-map's portion within object with fixed output size. As in spatial pyramid pooling (SPP), concatenation and pooling are performed for multiple sizes output. SPPN enhances R-CNN considerably at test time. Due to fast object feature extraction, training time is less.

This work is motivated from success achieved in designing CNN hierarchically considering integration of category hierarchy and linear classifiers. CNN models are continuously upgraded through enhancement of their components such as pooling layers [10], activation units [11], [12] and nonlinear layers [13]. These developments have improved CNN's training and learning processes. This work improves Fast R-CNN's performance considerably. The hierarchical model is built layer-wise considering Fast R-CNN as basic building block. There exist a wide variety of structures with categorical



(a) (b)

(c) (d)

Figure 1: Visual recognition tasks in computer vision (a) image classification (b) object detection (c) semantic segmentation (d) instance segmentation.

hierarchy [14]. The classification with linear classifiers having high number of classes is performed through classifiers' taxonomy. Here classifiers are verified considering test image which scales in sub-linear manner against number of classes [15], [16]. The hierarchy learning is either pre-specified [17], [18], [19] or achieved in top-down and bottom-up manner [20], [21], [22], [23], [24], [25], [26]. Hierarchical classifiers in [27] and [28] have reached considerable speedup bearing some accuracy loss. The initial work on category hierarchy for CNN is available in [29]. [30] achieves good accuracy with training images subset with re-labeled internal nodes in class tree hierarchy. [31] uses CNN hierarchy with scalability and has good classification performance.

Considering above motivation in this work, Modified Fast R-CNN (Mod Fast R-CNN) and Hierarchical Modified Fast R-CNN (HMod Fast R-CNN) [32] methods are proposed. HMod Fast R-CNN performs object detection by hierarchical learning in order to classify objects and refine them. This work looks towards development of HMod Fast R-CNN which integrates deep CNNs alongwith category hierarchy. The algorithm streamlines training process towards R-CNN based object detectors [2], [8]. The image classification task is decomposed into two steps. The weighted coarse component Mod Fast R-CNN classifier separates easy classes. The complex classes are directed towards weighted fine components which takes care of classes with confusion. HMod Fast R-CNN is build considering Fast R-CNN building block through module design principle. The building blocks are considered to be as one of the top ranked single Fast R-CNN. The coarse-to-fine classification is adopted here. Then fine category classifiers predictions are integrated as possibilistic means which takes care of inherent data uncertainty. The proposed architecture is evaluated through MS-COCO [33], CIFAR100 [34] and VisualQA [35] datasets. A comparative analysis of HMod Fast R-CNN with respect to other detectors such as deformable part model (DPM), all versions of you only look once (YOLO), single shot multibox detector (SSD) etc is performed on PASCAL VOC 2007 [36] VOC 2012 [37] datasets alongwith an error analysis. HMod Fast R-CNN achieves less error considering memory footprint increase as well as classification time. The schematic representation of HMod Fast R-CNN based prediction system [32] is given in Figure 2 in Appendix. This paper is structured as follows. The section 2 presents an overview of related work in object detection. The computational methodology is highlighted in section 3. The section 4 presents experimental results. Finally, in section 5 conclusion is given.

2 Related work

In this section we present significant developments in deep learning-based object detection in past few years. A good detection algorithm comes with strong semantic cues understanding and spatial information about image. Object detection is fundamental step towards many computer vision applications such as face recognition [38], [39], [40], pedestrian detection [41], [42], [43], video analysis [44], [45] and logo detection [46], [47], [48].

Initially object detection pipeline was divided into three steps viz (i) proposal generation (ii) feature vector extraction and (iii) region classification. During proposal generation objective was to search locations in image viz regions of interest (RoI) which contain objects. An intuitive idea is to scan whole image with sliding windows [49], [50], [51], [52], [53]. Input images are resized into different scales and multi-scale windows are used to slide through these images in order to capture information about multi-scale and different aspect ratios of objects. In next step on each image's location, a fixed-length feature vector is obtained considering sliding window in order to capture discriminative semantic information of covered region. This feature vector is encoded by several low-level visual descriptors [54], [55], [56], [57]. These descriptors have shown good robustness towards scale, illumination and rotation variance. Finally, region classifiers are learned to assign categorical labels to regions covered. Here support vector machines (SVM) [58] have been used because they offer good performance on small scale training data. Alongwith this classification techniques such as cascade learning [59], bagging [60] and adaboost [61] have also been used in region classification which have provided considerable improvements in detection accuracy.

The successful traditional object detection methods have focused towards designing feature descriptors in order to obtain embedding for RoI. With good feature representations and robust region classifiers impressive results [62], [63] are achieved on PASCAL VOC dataset [64]. The deformable part-based machines (DPMs) [65] learn and integrate multiple part models with deformable loss. They mine hard negative examples with latent SVM for discriminative training. Between 2008 to 2012 PASCAL VOC's progress based on these traditional methods showed incremental progress for building complicated ensemble systems. These reflected limitations of traditional detectors. These limitations are

reflected during proposal generation, feature descriptors and detection pipeline. In proposal generation huge number of redundant proposals are generated with many false positives during classification. The window scales designed manually and heuristically could not match objects well. The feature descriptors are hand-crafted based on low level visual cues [5], [56], [66] making them difficult to capture representative semantic information in complex contexts. In each step of detection pipeline is designed and optimized separately as a result of which global optimal solution could not be obtained.

After CNN's considerable success for image classification [1], [67], object detection also achieved remarkable progress on deep learning [2], [68], [69] based solutions. The newer detection algorithms outperformed traditional ones by huge margins. In [68] deep CNN model is optimized using stochastic gradient descent (SGD) via backpropagation. It provided good performance on digit recognition. However, deep networks are plagued with certain limitations like lack of large-scale annotated training data causing overfitting, limited computation resources and weak conceptual support compared to SVMs. In [5] a deep CNN is trained with ImageNet dataset which showed significant improvement on Large Scale Visual Recognition Challenge (ILSVRC) in comparison to all other approaches. After this success deep learning methods have been quickly adapted to other vision tasks where they have shown promising results over traditional methods. As compared to hand-crafted descriptors in traditional detectors, deep CNN generate hierarchical feature representations from raw pixels to high level semantic information. This is learned automatically from training data and shows more discriminative expression capability in complex contexts. Deep CNN also achieve better feature representation with large datasets. In traditional visual descriptors learning capacity is fixed and no improvement is reached as more data becomes available.

The major contributions in deep learning-based object detection can be categorized into three groups viz detection components, learning strategies and applications and benchmarks [32]. The detection components include detection settings, detection paradigms and backbone architectures. The learning strategies comes with training and testing stages. The applications and benchmarks include commonly used applications and public benchmark datasets. In present scenario, deep learning-based object detection frameworks can be divided into two categories viz two-stage detectors and one-stage detectors. The two-stage detectors cover R-CNN [2] and its variants [66], [70], [71] and one-stage detectors has YOLO [72] and its variants [73], [74]. The one-stage detector makes categorical object prediction on each location of feature maps without cascaded region classification step. The two-stage detectors use proposal generator in order to generate a sparse set of proposals and extract its features. This is followed by region classifiers which predict category of proposed region. The one-stage detectors are more time-efficient and have greater application towards real-time object detection. The two-stage detector achieve better detection performance and report state-of-the-art results on benchmark datasets.

Single shot detector (SSD) [74] has been one of the significant developments in object detection methods in past decade. It does not resample pixels or features for bounding boxes. By eliminating bounding boxes, it provides improvements considering speed at which object detection activities are performed. It provides high accuracy detection for low resolution images. For real time object detection YOLO [72] occupies a very prominent place. It is a state-of-the-art object detection system whose speed and accuracy has grown over the years. Till date we have 5 versions of YOLO [72], [75], [76], [77], [78] each of which supersedes previous versions. Apart from speed and accuracy some of biggest advantages of initial two versions of YOLO viz YOLO and YOLOv2 or YOLO9000 [72], [73] include network understandability towards generalized object representation and smaller architecture. The third version of YOLO, YOLOv3 [75] is extremely fast and accurate. It uses few tricks to improve training with increased performance including multiscale predictions, better backbone classifier and much more. The fourth version of YOLO, YOLOv4 [76] offers improved performance above previous versions. It provides superfast training and accurate object detection. It has also been verified for the influence of state-of-the-art bag-of-freebies and bag-of-specials object detection methods during detector training. The modified state-of-the-art methods include cross iteration batch normalization and path aggregation network which are more efficient and suitable for single GPU training. Finally, fifth version of YOLO, YOLOv5 [77] has also been launched with exceptional improvements. This version outperforms all previous versions with EfficientDet average precision and higher frames per second. Some of the recent major developments of deep learning-based object detection methods include [78], [79], [80], [81], [82], [83]. The Table 1 in Appendix highlights significant state-of-the-art research works in deep learning-based object detection. It is to be noted that in each case best possible results are presented.

3 Computational methodology

In this section computational framework of Mod Fast R-CNN and HMod Fast R-CNN [32] are presented. In subsection 3.1 Mod Fast R-CNN architecture with training is discussed. This is followed by HMod Fast R-CNN architecture with training in subsection 3.2. In subsection 3.3 HMod Fast R-CNN for detection is discussed.

3.1 Modified Fast R-CNN with training

Mod Fast R-CNN architecture is adopted from [32] with [70] as baseline method having certain variations. The architecture is highlighted in Figure 3 in Appendix. The entire image and objects' set forms input towards Mod Fast R-CNN network. The convolutional feature map is produced through processing of entire image alongwith convolutional and max pooling layers. Considering

feature map, a fixed length feature vector is extracted for each object's RoI pooling layer.

The sequence of fully connected (fy_ct) layer takes each feature vector as input. From fy_ct output is fed in 2 sibling output layers producing softmax probability estimates. The softmax probability are estimated with respect to OB object classes. This considers catch-all background class and another layer which has 4 real valued output numbers. For each of OB classes, 4 values' set are encoded considering bounding box refined positions. The max pooling is used for RoI pooling layer in order to convert its features into small feature map considering $Ht \times Wh$ fixed spatial extent. Here Ht and Wh are layers' hyper parameters not dependent on any specific RoI. RoI is rectangular window with convolutional feature map. A 4-tuple (rw, cm, ht, wh) defines an RoI where (rw, wh) is top left with ht and wh as its height and width respectively. The $ht \times wh$ window is divided by RoI max pooling into $Ht \times Wh$ sub-window grids having $\frac{ht}{Ht} \times \frac{wh}{Wh}$ size approximately. Then sub-window values are max-pooled into each grid cell output. Towards each feature map channel independent pooling is applied. RoI layer has 1 pyramid level. It is a special case of SPP layer in SPNN [8]. For experiments 6 pre-trained ImageNet [69] networks with 8 max pooling layers and between 8 and 18 convolutional layers are used. There are 3 transformations for Mod Fast R-CNN network with pre-trained network initialization. RoI pooling layer replaces last max pooling layer. It is configured as $Ht \times Wh$. This is followed by 1000-way ImageNet classification training for network's last fully connected and softmax layers. There are $A + 1$ categories for fully connected layers, softmax layers and bounding-box regressors which are specific to category. The network is updated to absorb 2 data inputs.

Mod Fast R-CNN uses backpropagation to train all network weights. Below SPP layer, weight updation is not possible as SPP layer's backpropagation is not effective. This inefficiency is spread across receptive field spanning entire input image starting from each RoI. The training inputs are large as forward pass processes entire receptive field. The feature sharing is used during training. For each image, RoIs are sampled hierarchically through I and then $\frac{R}{I}$ images for Mod Fast R-CNN training SGD mini-batches. In forward and backward passes, computation and memory are shared for RoIs from same image. Taking small I reduces computation of mini-batch. It slows convergence of training as same image RoIs are correlated. Significant results are achieved using $I = 2$ and $R = 128$ with less SGD iterations. Here training process is synchronized through fine-tuning which optimizes softmax classifier and bounding box regressors [2], [8].

In Mod Fast R-CNN 2 sibling output layers are used. The initial output is discrete probability distribution per RoI considering $A + 1$ categories which is $prob = (prob_0, \dots\dots, prob_A)$. For fully connected layer, $prob$ is calculated for softmax considering $A + 1$ outputs. The 2^{nd} sibling layer has bounding-box regression offsets outputs for A object classes as $v^a = (v_x^a, v_y^a, v_{wh}^a, v_{ht}^a)$. The parameterization for v^a is given in [2]. Here v^a specifies translation (scale-invariant) and height shift (log-space) with respect to the object. For each training RoI labeling is done considering ground-truth class u and ground-truth with bounding-box regression for v. For each labeled RoI, there is joint classification for training and bounding-box regression with respect to multitask loss L:

$$L(p, u, v^u, s) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(v^u, s) \quad (1)$$

For true class u, log loss is $L_{cls}(p, u) = -\log p_u$. L_{loc} is second task loss which is specified considering true bounding-box regression target tuples such that $s = (s_x, s_y, s_{wh}, s_{ht})$ with predicted tuple $v^a = (v_x^a, v_y^a, v_{wh}^a, v_{ht}^a)$ for class u. When $u \geq 1$ $[u \geq 1] = 1$ else 0 is inversion bracket indicator function. Background class with catch-all convention is marked as $u = 0$. L_{loc} is ignored with background RoIs having no ground-truth bounding box. The bounding-box regression loss is:

$$L_{loc}(v^u, s) = \sum_{i \in (x,y,wh,ht)} smooth_{L_1}(v_i^u - s_i) \quad (2)$$

In equation (2) smooth function is:

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2 & |x| < 1 \\ |x| - 0.5 & ow \end{cases} \quad (3)$$

In $smooth_{L_1}(x)$, loss L_2 in Mod Fast R-CNN and SPPN [8] is more outliers' sensitive than robust loss L_1. The loss L_2 needs to be carefully tuned in terms of learning rates to prevent gradients exploding with unbounded training as regression targets. This sensitivity is eliminated through equation (3). In equation (1) balance between L_1 and L_2 is controlled by λ. With $\lambda = 1$ ground-truth regression targets $s_i \sim N(0,1)$. The class-agnostic object network is trained using loss factor [69]. The localization and classification are separated by 2-network system. The images $(N = 2)$ selected uniformly at random are used from SGD minibatch created at fine tuning. The dataset is permuted to perform in iterations. From each image 64 RoIs are sampled considering mini-batches of $R = 128$ size. 25% of RoIs are taken from objects which have intersection over union (IoU) overlap having ground-truth bounding box ≥ 0.5 [2].

$$\frac{\partial L}{\partial x_i} = \sum_r \sum_j [i = i * (r, j)] \frac{\partial L}{\partial y_{rj}} \quad (4)$$

The partial derivative $\frac{\partial L}{\partial y_{rj}}$ accumulates if i is selected as argmax considering y_{rj} through max pooling for each mini-batch RoI r and for pooling output unit y_{rj}. Using backwards function of layer over RoI pooling layer, partial derivatives $\frac{\partial L}{\partial y_{rj}}$ are calculated. Considering softmax classification and bounding-box regression for fully connected layers, an initialization is done through zero-mean Gaussian distributions. Here standard deviations are taken as 0.01 and 0.001 for both cases with 0 as the bias initialization. For weights learning rate is 1 per layer and for biases learning rate is 2 per layer considering all layers. The global learning rate is 0.001. Trainval SGD is executed for 30000 minibatch iterations when training on

PASCAL VOC 2007 or VOC 2012. Then learning rate is lowered to 0.0001 and training is done for next 10000 iterations. SGD is executed for more iterations, when training is done on larger datasets. For weights and biases, momentum is 0.9 and parameter decay is 0.0005. Brute-force learning and image pyramids are used to achieved scale invariant object detection. These approaches used here are taken from [8]. During training and testing for brute-force approach each image is being processed at predefined pixel size. Using training data, network learns scale-invariant object detection. From an image pyramid, approximate scale-invariance to network is provided by multi-scale approach. Each object proposal is scale normalized approximately through image pyramid at test time. Each time when an image is sampled, pyramid scale is randomly sampled at multi-scale training. The detection considers running forward pass. Here objects are assumed to be precomputed as Fast R-CNN network where it is fine-tuned. The network input is image or image pyramid as well as R objects list towards score. R is typically taken as 2000, though cases are there when it is about 45000 at test time. Using image pyramid, each RoI is placed to scale such that scaled RoI is near to 224^2 pixels [8]. Considering each test RoI r forward pass output is posterior probability distribution $prob$ with predicted bounding-box set offsets relative to r for each A classes which gets its refined bounding-box prediction. For each object class k through estimated probability $Prob(class = k|r) \triangleq pk,$ a detection confidence is assigned to r. Then for each class using Fast R-CNN algorithm [84] non-maximum suppression is performed independently.

The time spent for calculating convolutional layers is greater than fully connected layers considering whole-image classification. The processing time for number of RoIs is large enough for detection. It is about 50% of forward pass time required for calculating fully connected layers [84], [85]. By compressing large fully connected layers with truncated singular value decomposition (SVD) easy acceleration is achieved. Each layer is parameterized by $u \times v$ weight matrix W which is approximately factorized as $W \approx U\sum_t V^T$.Here U is $u \times t$ matrix constituting $W's$first t left-singular vectors, $\sum_t$ is $t \times t$ diagonal matrix with $W's$ top t singular values and V is $v \times t$ matrix constituting W' s first t right-singular vectors. The parameter count is reduced from uv to $t(u + v)$ through truncated SVD. This works well when $t < min(u, v)$. Corresponding to W single fully connected layer network is compressed by replacing 2 fully connected layers with no in between non-linearity. With no biases, weight matrix $\sum_t V^T$ is used for first few layers and with original biases linked with W, U is used for second few layers. As RoIs number grows, good speedups are achieved through this compression.

3.2 Hierarchical modified fast R-CNN with training

Now architecture of HMod Fast R-CNN [32] is presented. Based on the success of Mod Fast R-CNN [32], HMod Fast R-CNN is discussed in this section. The image dataset has images $\{x_i, y_i\}_i$ with x_i and y_i representing image data and label respectively. The dataset $\{S_j^f\}_{j=1}^{Ct}$ contains Ct fine categories of images. The category hierarchy with A coarse categories $\{S_a^{ct}\}_{a=1}^{A}$ is used towards formation of learning process. HMod Fast R-CNN emulates category hierarchy structure with coarse categories making up fine categories.

As shown in Figure 4 [32] in Appendix end-to-end classification happens here. It consists of 5 components viz (a) high-level feature extraction layer (b) low-level feature extraction layer (c) weighted coarse component independent layers $\{B^a\}_{a=1}^{A}$ (d) weighted fine component independent layers $\{F^a\}_{a=1}^{A}$ and (e) possibilistic averaging layer. The extraction layers are present on leftmost side of Figure 4. They take raw image pixel as input and extract high-level features followed by low-level features. The configuration of extraction layers is kept same as preceding layers with respect to building block net. The weighted coarse component independent layers assign weight factor to each of A layers and gives coarse prediction based on best weight achieved. The probabilities in weighted coarse category provide: (a) weight factor towards combining predictions which fine category components make and (b) consider threshold conditional executions of fine category components are enabled for which coarse probabilities are quite large. The independent layers are represented considering weighted fine category classifiers set $\{F^a\}_{a=1}^{A}$ where weighted fine category predictions are made by each classifier. Each weighted fine category component classifies small categories set accurately. As such from here fine prediction is produced with respect to partial categories set. When partial set do not have probabilities of other fine categories, they are taken as zero. From building block Mod Fast R-CNN layer configurations are copied. However, in final classification layer filter numbers are taken as partial set size.

The common layers are shared for both weighted coarse category and fine category components. This is because of reasons stated here. The preceeding layers in deep networks [63] respond towards low-level features which are class-agnostic for example corners and edges. The class-specific features are extracted from rear layers. The preceding layers are shared by both coarse and fine components as for both coarse and fine classification tasks low-level features are useful. The floating-point operations network execution memory footprint is considerably reduced. HMod Fast R-CNN parameters are also decreased which is vital towards network's training. Finally, there is a possibilistic averaging layer where fine category and coarse category predictions are received and converted to possibilistic measures through equation (5). Then a weighted average is produced as final prediction result. It is to be noted that merging part plays a significant role in averaging layer of HMod Fast R-CNN. The weighted factors are decided based on certain heuristics [32]. In initial iterations weight factors are decided based on dataset considered. The distribution of coarse-grained and fine-grained images are considered in deciding

weighted factors in later iterations. This helps in reaching best possible results in final prediction. The possibilistic measures handles inherent uncertainty in data better than probabilistic values

$$possb(x_i) = \frac{\sum_{a=1}^{A} possb(B_{ia}) possb_a(x_i)}{\sum_{a=1}^{A} possb(B_{ia})} \quad (5)$$

In equation (5) $possb(B_{ia})$ is possibility of coarse category a considering image x_i which is predicted through coarse category component B and $possb_a(x_i)$ is prediction achieved through fine category component F^a. Considering building block Mod Fast R-CNN, layer configurations for both coarse and fine category components are reused. The flexibility in modular design gives best module Mod Fast R-CNN as building block.

As fine category components are inserted into HMod Fast R-CNN, parameters in rear layers increases linearly with respect to coarse categories. This increases training complexity as well as overfitting risk considering same amount of training data. Within stochastic gradient descent mini-batch, training images are routed probabilistically towards various fine category components. To ensure parameter gradients larger minibatch are required in fine category components which are estimated through quite large number of training samples. The training memory footprint is increased by large training mini-batch but training process is considerably slow. HMod Fast R-CNN training is decomposed into several steps as shown in Figure 5 [32].

Algorithm: HMod Fast R-CNN training algorithm
Procedure: HMod Fast R-CNN training
Step 1: Pre-train HMod Fast R-CNN
Step 1.1: Initialize weighted coarse category components
Step 1.2: Pre-train weighted fine category components
Step 2: Fine tune complete HMod Fast R-CNN

Figure 2: HMod Fast R-CNN training algorithm.

HMod Fast R-CNN is sequentially pre-trained for coarse and fine category components. First a building block Mod Fast R-CNN F^p is pretrained through training set. There is a resemblance in building block Mod Fast R-CNN with preceding and rear layers in coarse category component. As a result of this for initialization purpose, weights of F^p are placed into coarse category component. Fine category components $\{F^a\}_a$ are independently pre-trained in parallel. Each F^a specializes towards classification of fine categories considering coarse category S_a^{ct}. Thus, pre-training of each F^a uses images $\{x_i | i \in S_a^{ct}\}$ with coarse category S_a^{ct}. The initialization is done for shared preceding layers which are kept fixed now. All rear layers are initialized for each F^a except last convolutional layer through writing learned parameters from pre-trained model F^p.

3.3 Hierarchical modified fast R-CNN for Detection

After HMod Fast R-CNN [32] is trained, detection is performed. This section highlights this issue. The complete HMod Fast R-CNN is fine-tuned when coarse and fine category components are appropriately pre-trained. Every fine category component is directed towards classifying fixed fine categories subset, when learning is done for category hierarchy and associated mapping $\boldsymbol{P^0}$. The coarse categories semantics predicted through coarse category component must remain consistent coarse category component during fine-tuning. The consistency term in coarse category is included in order to regularize multiple group discriminant loss. The mapping $\boldsymbol{P}$: $[\mathbf{1}, \boldsymbol{Ct}] \mapsto [\mathbf{1}, \boldsymbol{A}]$ which is fine-to-coarse in nature paves a way towards specification of target coarse category distribution $\{\boldsymbol{t_a}\}$. Here $\boldsymbol{t_a}$ is placed as fraction for all training images within coarse category $\boldsymbol{S_a^{ct}}$ with assumption that distribution for coarse categories over training dataset is near to that in trained mini-batch:

$$t_a = \frac{\sum_{j|a\in F(j)} |S_j|}{\sum_{a'=1}^{A} \sum_{j|a\in F(j)} |S_j|} \quad (6)$$

For fine-tuning HMod Fast R-CNN final loss function is:

$$Loss = -\frac{1}{n}\sum_{i=1}^{n} \log\left(possb_{y_i}\right) + \frac{\lambda}{2}\sum_{a=1}^{A}\left(t_a - \frac{1}{n}\sum_{i=1}^{n} B_{ia}\right)^2 \quad (7)$$

Here training mini-batch size is n and regularization constant $\lambda = 20$. As fine category components are added into HMod Fast R-CNN, rear layers with parameters, memory footprint and execution time variables are linearly scaled with coarse categories. In order to scale HMod Fast R-CNN to large-scale visual recognition, layer parameter compression techniques and conditional execution are used. It is not required to test all fine category classifiers for given image because they have weights B_{ia} which are not significant as shown in equation (7). The final predictions are negligible here. HMod Fast R-CNN classification is accelerated through conditional executions of top weighted fine components. Thus, B_{ia} is given a threshold using $B_t = (\beta A)^{-1}$ and reset $B_{ia} = 0$ when $B_{ia} < B_t$. The evaluation is not done for fine category classifiers with $B_{ia} = 0$. With HMod Fast R-CNN rear layers parameter in classifiers of fine category is directly proportional to number of coarse categories. In order to reduce memory footprint compression of layer parameters is done at test time.

The product quantization approach is chosen to compress parameter matrix $W \in R^{m\times n}$ by partitioning as segments having width s horizontally such that $W = \left[W^1, \ldots\ldots, W^{\left(\frac{n}{s}\right)}\right]$. K-means then clusters rows into $W^i \forall i \in \left[1, \left(\frac{n}{s}\right)\right]$. A compression factor of $\frac{32mn}{\left(32kn+\frac{8mn}{s}\right)}$ is achieved through storing cluster indices which are near at 8-bit integer matrix $I \in R^{m\times\left(\frac{n}{s}\right)}$ with cluster centers in floating number matrix $C \in R^{k\times n}$. The hyperparameters for parameter compression are (s, k).

4 Experimental results

The results from various experiments performed are presented in this section. HMod Fast R-CNN is evaluated

on benchmark datasets MS-COCO [33] and CIFAR100 [34], [86] as well as VisualQA [35]. It is implemented through Caffe [87]. Back propagation [32] is used towards network training. NVIDIA Tesla V100 card is used to simulate all test experiments.

MS-COCO [33] is large-scale object detection, segmentation and captioning dataset. It comes with several prominent features such as object segmentation, recognition in context and super pixel stuff segmentation. It consists of 330000 images with more than 200000 labeled images. It has 1.5 million object instances with 80 object categories, 91 stuff categories, 5 captions per image and 250000 people with key points. Figures 6(a) and 6(b) consider 10 superior overlapping coarse categories. The coarse category optimal number depends on dataset. There is also impact within categories inherent hierarchy.

There are 100 natural image classes in CIFAR100 [34], [86] dataset. The prepared dataset comprises of 50000 and 10000 images for training and testing respectively. The dataset pre-processing is done using contrast normalization globally and ZCA-cor whitening. For training image patches of 30×30 size is flipped and cropped randomly. A 4 stacked layer NIN network is adopted which is denoted as CIFAR100-NIN and placed in HMod Fast R-CNN's building block. The preceeding layers from $conv1$ to $pool1$ are shared by components from weighted fine category. These are responsible towards 10% and 35% of total parameters and floating-point operations respectively. The rest layers are considered as independent layers. In order to construct category hierarchy, 10000 images are chosen at random and taken as heldout set considering training set. There is a visual similarity for fine categories considering similar coarse categories. Pre-training is done for rear layers of fine components category. The initial learning rate is 0.05. This decreases by factor 10 for every 6000 iterations. With mini-batches of 256 size, fine-tuning is done with respect to 20000 iterations. Here initial learning rate is 0.005. This decreases by factor 10 for every 10000 iterations. 10view testing [32] is used towards evaluation. Six 30×30 patches (with 5 corner patches and 1 center patch) alongwith their reflections (horizontal) and predictions (average) are extracted. HMod Fast R-CNN has lower testing error than CIFAR100-NIN.

With category hierarchy construction, clustering algorithm adjusts coarse category. When hyperparameter γ is varied, coarse categories can be made overlapping or disjoint. Their impacts are investigated on classification error. The experiments are performed with $5, 10, 16$ and 20 coarse categories with varying the values of γ. Figures 7(a) and 7(b) consider 10 superior overlapping coarse categories achieved with $\gamma = 6$. The coarse category optimal number and γ depend on dataset. They are also impacted within categories inherent hierarchy.

In comparison with building block net, shared layers usage results in sublinear computational complexity and memory footprint of HMod Fast R-CNN considering fine category classifiers. HMod Fast R-CNN consumes less than four times memory as building block net with no compression of parameters considering 10 fine category classifiers with respect to MS-COCO and CIFAR100-NIN. Tables 2 and 3 highlight significance of classification error, memory footprint and net execution time. Using pre-trained building block net, HMod Fast R-CNN is structured with coarse category and all fine category components which use independent preceding layers initialization. The central cropping is used with single-view testing with slight error increase. The memory footprint and testing time is considerably reduced through shared layers.

By varying hyperparameter β fine category components are affected considerably. The tradeoff exists between execution time and classification. For fine category when more components are executed higher accuracy is achieved through large β values. As shown in Tables 2 and 3 there is slight error increase when conditional executions are enabled through $\beta = 6$. HMod Fast R-CNN achieves 3 times testing time as compared with building block net. The fine category HMod Fast R-CNNs with independent layers from $conv2$ to $conv6$ are compressed and memory footprint reduces from 448 MB to 269 MB with slight error increase. As highlighted in Tables 2 and 3 HMod Fast R-CNN memory footprint is nearly 2 times in comparison with building block model. As a result of this, it is mandatory to compare strong baseline with identical complexity for HMod Fast R-CNN.

CIFAR100-NIN is adapted with doubled filters for all convolutional layers. This results in memory footprint increase by more than 3 times. This is denoted as CIFAR100-NIN-double. The error is higher than HMod Fast R-CNN but lower than building block net. Conceptually HMod Fast R-CNN differs from model averaging [32]. With model averaging full category sets are classified for all models. There is an independent training for each model. As different initializations are used, predictions are different. Partial category sets are classified for each classifier in fine category HMod Fast R-CNN. In order to make comparison between HMod Fast R-CNN and model averaging, 2 CIFAR100-NIN networks are trained independently. This is followed by their prediction average which is treated as final prediction. Tables 4 and 5 show that HMod Fast R-CNN achieves lower error. It is noted that HMod Fast R-CNN bears orthogonality towards model averaging. There is a considerable performance enhancement for HMod Fast R-CNN ensembles. It is fine-tuned using multiple group discriminant analysis in order to verify coarse category consistency term effectiveness in equation (7). Tables 4 and 5 show higher testing error for HMod Fast R-CNN is fine-tuned considering consistency in coarse category. There is considerable performance improvement for HMod Fast R-CNN using MS-COCO and CIFAR100 datasets. In order to further support the experimental hypothesis some results on Visual QA dataset are highlighted in Tables 6 and 7.

A comparative performance analysis of Mod Fast R-CNN and HMod Fast R-CNN [32] with Fast R-CNN, YOLO, Fast YOLO, YOLOv3, YOLOv4, YOLOv5, DPM and SSD on PASCAL VOC 2007 and VOC 2012 datasets

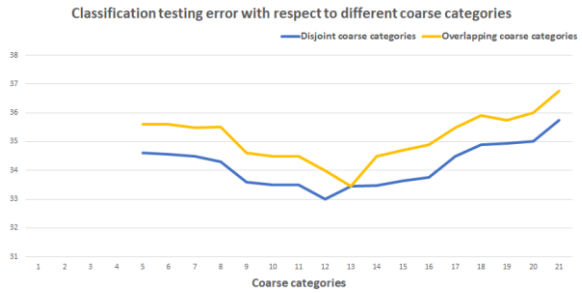


Figure 3: Testing error (10-view) against number of coarse categories with MS-COCO dataset.

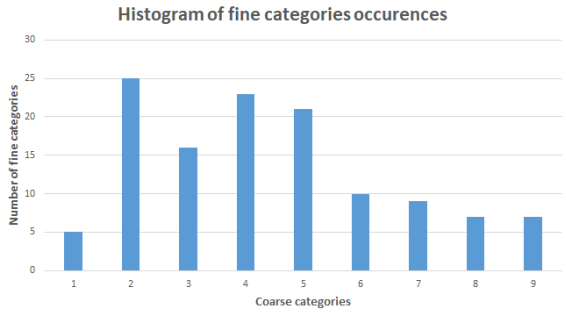


Figure 4: Overlapping coarse categories with respect to fine category occurrences with MS-COCO dataset.

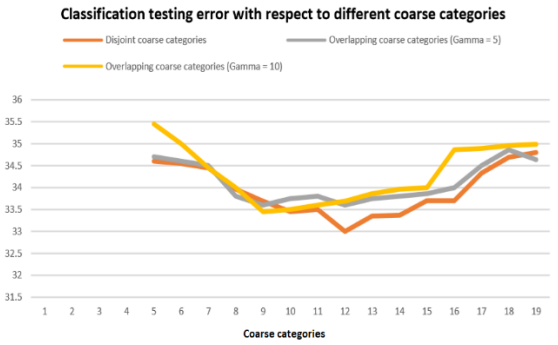


Figure 5: Testing error (10-view) against number of coarse categories with CIFAR100 dataset.

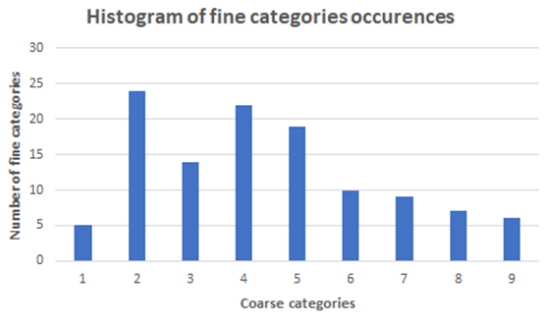


Figure 6: Overlapping coarse categories with respect to fine category occurrences with CIFAR100 dataset.

is presented in Table 8. In order to achieve better results few object detectors are trained through union of PASCAL VOC 2007 and VOC 2012 datasets. An error analysis of HMod Fast R-CNN [32] with Fast R-CNN and all versions of YOLO on same dataset is shown in Figure 8. Here localization and background errors percentage in top N detection category wise is highlighted. In order to further strengthen the results a comparative analysis of Mod Fast R-CNN and HMod Fast R-CNN with state-of-the-art methods is presented in Table 9.

Before concluding this section, we throw some light on design evaluation of HMod Fast R-CNN. In this direction, several experiments are performed to achieve optimal performance for HMod Fast R-CNN. However, there remains certain questions which needs to be discussed. Some of these aspects have been addressed here and rest of them form the future scope of work.

The first question is: Is training using multi-tasking helpful? The multi-task training is always useful because

Model	Top-1, Top-5	Mem (MB)	Time (s)
Base: CIFAR100-NIN	31.90	186	0.05
Mod Fast R-CNN w/o SL	31.87	736	2.00
HMod Fast R-CNN w/o SL	31.72	1250	2.36
HMod Fast R-CNN	**31.36**	**455**	**0.31**
HMod Fast R-CNN + CE	**31.21**	**448**	**0.14**
HMod Fast R-CNN + CE + PC	**31.05**	**270**	**0.14**

Table 1: Testing errors, memory footprint and testing time – building block nets and HMod Fast R-CNN: Comparative analysis on MS-COCO dataset (mini-batch size (for testing) = 100; SL = Shared layers, CE = Conditional execution, PC = Parameter comparison).

Model	Top-1, Top-5	Mem (MB)	Time(s)
Base: CIFAR100-NIN	33.96	186	0.05
Mod Fast R-CNN w/o SL	33.90	736	2.00
HMod Fast R-CNN w/o SL	33.69	1250	2.37
HMod Fast R-CNN	**33.34**	**455**	**0.27**
HMod Fast R-CNN + CE	**33.19**	**448**	**0.10**
HMod Fast R-CNN + CE + PC	**31.05**	**270**	**0.10**

Table 2: Testing errors, memory footprint and testing time – building block nets and HMod Fast R-CNN: Comparative analysis on CIFAR100 dataset (mini-batch size (for testing) = 100; SL = Shared layers., CE = Conditional execution, PC = Parameter comparison).

Method	Error
Model averaging (2 CIFAR100-NIN nets)	36.05
CIFAR100-NIN-double	34.24
Base: CIFAR100-NIN	33.96
Mod Fast R-CNN (no fine tuning)	32.66
Mod Fast R-CNN (fine tuning without CCC)	32.09
Mod Fast R-CNN (fine tuning with CCC)	31.87
HMod Fast R-CNN (no fine tuning)	**32.34**
HMod Fast R-CNN (fine tuning without CCC)	**32.05**
HMod Fast R-CNN (fine tuning with CCC)	**31.84**

Table 3: Testing errors (10-view) on MS-COCO.

Method	Error
Model averaging (2 CIFAR100-NIN nets)	36.05
CIFAR100-NIN-double	34.24
Base: CIFAR100-NIN	33.96
Mod Fast R-CNN (no fine tuning)	33.66
Mod Fast R-CNN (fine tuning without CCC)	33.09
Mod Fast R-CNN (fine tuning with CCC)	31.90
HMod Fast R-CNN (no fine tuning)	**33.34**
HMod Fast R-CNN (fine tuning without CCC)	**33.05**
HMod Fast R-CNN (fine tuning with CCC)	**31.86**

Table 4: Testing errors (10-view) on CIFAR100 dataset (CCC = coarse category consistency).

Model	Top-1, Top-5	Mem (MB)	Time (s)
Base: Prior VisualQA	34.03	186	0.05
Mod Fast R-CNN w/o SL	33.96	736	2.07
HMod Fast R-CNN w/o SL	33.87	1250	2.39
HMod Fast R-CNN	**33.72**	**455**	**0.27**
HMod Fast R-CNN + CE	**33.22**	**448**	**0.10**
HMod Fast R-CNN + CE + PC	**31.06**	**270**	**0.10**

Table 5: Testing errors, memory footprint and testing time – building block nets and HMod Fast R-CNN: Comparative analysis on VisualQA dataset (mini-batch size (for testing) = 100; SL = Shared layers, CE = Conditional execution, PC = Parameter comparison).

Method	Error
d-LSTM+n-I Visual QA	34.69
Base: Prior Visual QA	34.03
Mod Fast R-CNN (no fine tuning)	33.96
Mod Fast R-CNN (fine tuning without CCC)	33.36
Mod Fast R-CNN (fine tuning with CCC)	32.00
HMod Fast R-CNN (no fine tuning)	**33.86**
HMod Fast R-CNN (fine tuning without CCC)	**33.26**
HMod Fast R-CNN (fine tuning with CCC)	**31.98**

Table 6: Testing errors (10-view) on VisualQA dataset (CCC = coarse category consistency).

Object Detectors	Training	mAP	FPS
100 Hz DPM	2007	16.0	100
Fast R-CNN	2007+2012	70.0	0.5
Faster R-CNN	2007+2012	70.7	0.5
YOLO	2007+2012	72.7	155
YOLOv2	2007+2012	75.5	45
YOLOv3	2007+2012	76.5	31
YOLOv4	2007+2012	77.6	27
YOLOv5	2007+2012	78.6	26
Mod Fast R-CNN	2007+2012	81.06	0.5
HMod Fast R-CNN	2007+2012	**87.6**	**0.3**

Table 7: A comparative performance analysis of HMod Fast R-CNN vs other object detectors on PASCAL VOC 2007 and 2012 datasets (2007+2012: union of VOC2007 trainval and test and VOC 2012 trainval).

there is no need to manage sequentially-trained tasks pipeline. This potentially improves accuracy results as there is an influence among the tasks considering shared representation which CNN uses here. The classification loss is one such measure which baseline network uses during training. Another useful measure used here is multi-task loss. It is observed that there is an improvement of pure classification accuracy with respect to only classification training through multi-task training.

The second question is: Is brute-force scale invariance always useful here? The brute-force scale invariance is achieved here through single scale and multi scale (using image pyramids). The scale of image is specified as its shortest side length. Here single and multi-scale pyramids have produced good results. There are certain instances where single scale has shown best tradeoff between speed and accuracy considering very deep models.

The third question is: Is more training data required to verify the results? As a rule of thumb, when trained with large datasets, performance of object detector improves. The same verdict is true here. Here as and when training data volumes are increased object detection performance grows considerably. Further heterogeneity in training data helps network towards learning capability generalization.

The fourth question is: Is using more object proposals always better? Object detectors use two types of proposal viz object proposals with sparse set and dense set. Here dense set proposals have worked well. This has considerably improved HMod Fast R-CNN object detection accuracy. As proposals have a pure computational role, increasing their number/image have produced good results.

The fifth question is: What is the optimal number of layers required in HMod Fast R-CNN to achieve best performance? This depends on object detection dataset where HMod Fast R-CNN architecture is evaluated. This aspect in non-trivial in nature and there is no thumb rule.

Object Detectors	Significant Results
SAF R-CNN	AMR: 9.32
Deep Network Cascades	AMR: 31.11; FPS: 15
LOGO-Net	mAP: 69.9
SCL	mAP: 16.3
SL^2	mAP: 46.9
Local Structured HOG-LBP	mAP: 34.3
Faster R-CNN	mAP: 70.7
Fast R-CNN	mAP: 70.0
FPN	mAP: 59.1
YOLO	mAP: 72.7; FPS: 155
YOLO9000/YOLOv2	mAP: 75.5; FPS: 45
SSD	mAP: 76.8; FPS: 22
YOLOv3	mAP: 76.5; FPS: 31
YOLOv4	mAP: 77.6; FPS: 27
YOLOv5	mAP: 78.6; FPS: 26
Low-cost ISS	mAP: 99.4
ADS + Hardw Accelerators	mAP: 83.64; FPS: 30
xYOLO	mAP: 68.22; FPS: 9.66
Grape Disease Detection	mAP: 95.57
Mod Fast R-CNN	mAP: 81.06; FPS: 0.5
HMod Fast R-CNN	mAP: **87.6**; FPS: **0.3**

Table 8: A comparative performance analysis of HMod Fast R-CNN vs significant state-of-the-art object detection methods (AMR: Average Miss Rate; mAP: Mean Average Precision).

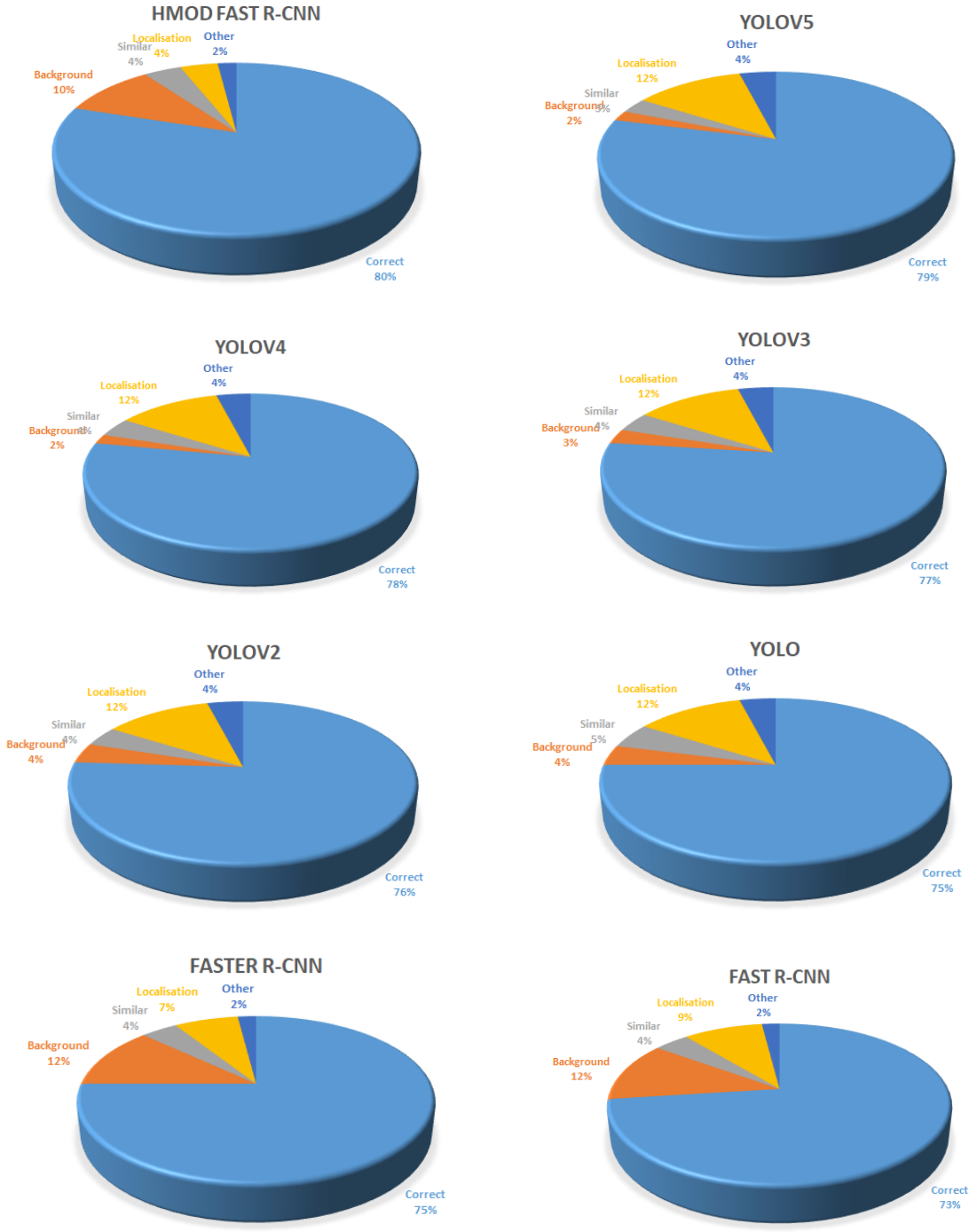


Table 9: An error analysis of HMod Fast R-CNN vs Faster R-CNN, Fast R-CNN, YOLO, YOLOv2, YOLOv3, YOLOv4, YOLOv5 on PASCAL VOC 2007 and 2012 datasets.

Here prior experience in building network architecture with image datasets has produced good results.

5 Conclusion

In this work, we presented HMod Fast R-CNN which is hierarchical updated version of Mod Fast R-CNN. It improves Fast R-CNN's architecture considerably. The computational system comprises of extraction layers, weighted coarse and fine component layers and possibilistic averaging layer. The possibilistic averaging layer converts fine category and coarse category probabilistic predictions into possibilistic measures which is weighted average and considered as final prediction result. The possibilistic measures effectively address inherent uncertainty in data. The experimental results with MS-COCO, CIFAR100 and VisualQA datasets provide several new insights. This fact is highlighted using four variant building block nets. The proposed network's performance superiority over other object detectors on MS-COCO, PASCAL VOC 2007 and VOC 2012 datasets are also illustrated. HMod Fast R-CNN architecture can be

further extended with more than five levels. This will improve experimental results in terms of object detection accuracy as well as accelerates overall process considering theoretical viewpoints. The future work looks towards developing HMod Fast R-CNN with more layers and verifying results with significant image datasets.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. Deep residual learning for image recognition. IEEE Conference on Computer Vision and Pattern Recognition, 770–778, 2016. https://doi.org/10.1109/CVPR.2016.90

[2] Ross Girshick, Jeff Donahue, Trevor Darrell and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. IEEE Conference on Computer Vision and Pattern Recognition, 580–587, 2014. https://doi.org/10.1109/CVPR.2014.81

[3] Kaiming He, Georgia Gkioxari, Piotr Dollár and Ross Girshick. Mask R-CNN. IEEE International Conference on Computer Vision, 2961–2969, 2017. https://doi.org/10.1109/ICCV.2017.322

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFS. arXiv, arXiv:1412.7062, 2014.

[5] Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton. ImageNet classification with deep convolutional neural networks. International Conference on Neural Information Processing Systems, 25:1097–1105, 2012. https://doi.org/10.1.1.299.205

[6] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus and Yann LeCun. OverFeat: Integrated recognition, localization and detection using convolutional networks. arXiv, arXiv:1312.6229, 2014.

[7] Yann LeCun, Bernhard Boser, John Denker, David Henderson, Robert Howard, William Hubbard and Lawrence Jackel. Backpropagation applied to handwritten zip code recognition. Neural Computation, 1(4): 541–551, 1989. https://doi.org/10.1162/neco.1989.1.4.541

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. arXiv, arXiv: 1406.4729, 2014.

[9] Yukun Zhu, Raquel Urtasun, Rusian Salakhutdinov and Sanja Fidler. segDeepM: Exploiting segmentation and context in deep neural networks for object detection. arXiv, arXiv:1502.04275, 2015.

[10] Mathew D. Zeiler and Rob Fergus. Stochastic pooling for regularization of deep convolutional neural networks. arXiv, arXiv:1301.3557, 2013.

[11] Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville and Yoshua Bengio. Maxout networks. International Conference on Machine Learning, 28(3):1319–1327, 2013. https://doi.org/10.5555/3042817.3043084

[12] Jost Tobias Springenberg and Martin Riedmiller. Improving deep neural networks with probabilistic maxout units. arXiv, arXiv:1312.6116, 2013.

[13] Min Lin, Qiang Chen and Shuicheng Yan. Network in network. arXiv, arXiv:1312.4400, 2013.

[14] Anne-Marie Tousch, StěPhane. Herbin and Jean-Yves Audibert. Semantic hierarchies for image annotation: A survey. Patten Recognition, 45(1):333–345, 2012. https://doi.org/abs/10.1016/j.patcog.2011.05.017

[15] Samy Bengio, Jason Weston and David Grangier. Label embedding trees for large multi-class tasks. International Conference on Neural Information Processing Systems, 1:163–171, 2010. https://doi.org/abs/10.5555/2997189.2997208

[16] Tianshi Gao and Daphne Koller. Discriminative learning of relaxed hierarchy for large-scale visual recognition. IEEE Conference on Computer Vision and Pattern Recognition, 2072–2079, 2011. https://doi.org/10.1109/ICCV.2011.612648

[17] Marcin Marszalek and Cordelia Schmid. Semantic hierarchies for visual object recognition. IEEE Conference on Computer Vision and Pattern Recognition, 1–7, 2007. https://doi.org/10.1109/CVPR.2007.383272

[18] Nakul Verma, Dhruv Mahajan, Sundararajan Sellamanickam and Vinod Nair. Learning hierarchical similarity metrics. IEEE Conference on Computer Vision and Pattern Recognition, 2280–2287, 2012. https://doi.org/10.1109/CVPR.2012.6247938C

[19] Yangqing Jia, Joshua T. Abbott, Joseph. Austerweil, Tom Griffiths and Trevor Darrell. Visual concept learning: Combining machine vision and bayesian generalization on concept hierarchies. International Conference on Neural Information Processing Systems, 2:1842–1850, 2013. https://doi.org/10.5555/2999792.2999818

[20] Ruslan Salakhutdinov, Antonio Torralba and Josh Tenenbaum. Learning to share visual appearance for multiclass object detection. IEEE Conference on Computer Vision and Pattern Recognition, 1481–1488, 2011. https://doi.org/10.1109/CVPR.2011.5995720

[21] Gregory Griffin and Pietro Perona. Learning and using taxonomies for fast visual categorization. IEEE Conference on Computer Vision and Pattern Recognition, 1–8, 2008. https://doi.org/10.1109/CVPR.2008.4587410

[22] Marcin Marszałek and Cordelia Schmid. Constructing category hierarchies for visual recognition. Proceedings of European Conference on Computer Vision, IV:479–491, 2008. https://doi.org/10.1007/978-3-540-88693-8_35

[23] Li-Jia Li, Chong Wang, Yongwhan Lim, David M. Blei and Li Fei-Fei. Building and using a semantivisual image hierarchy. IEEE Conference on Computer Vision and Pattern Recognition, 3336–3343, 2010. https://doi.org/10.1109/CVPR.2010.5540027

[24] Hichem Bannour and Cěline Hudelot. Hierarchical image annotation using semantic hierarchies. ACM

International Conference on Information and Knowledge Management, 2431–2434, 2012. https://doi.org/10.1145/2396761.2398659

[25] Jia Deng, Sanjeev Satheesh, Alexander C. Berg and Fei. Li. Fast and balanced: Efficient label tree learning for large scale object recognition. International Conference on Neural Information Processing Systems, 1:567–575, 2011. https://doi.org/10.5555/2986459.2986523

[26] Josef Sivic, Bryan C. Russell, Andrew Zisserman, William T. Freeman and Alexei A. Efros. Unsupervised discovery of visual object class hierarchies. IEEE Conference on Computer Vision and Pattern Recognition, 1–8, 2008. https://doi.org/10.1007/s11263-009-0271-8

[27] Jia Deng, Jonathan Krause, Alexander C. Berg and Li Fei-Fei. Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition. IEEE Conference on Computer Vision and Pattern Recognition, 3450–3457, 2012. https://doi.org/10.1109/CVPR.2012.6248086

[28] Baoyuan Liu, Fereshteh Sadeghi, Marshall Tappen, Ohad Shamir and Ce Liu. Probabilistic label trees for efficient large scale image classification. IEEE Conference on Computer Vision and Pattern Recognition, 843–850, 2013. https://doi.org/10.1109/CVPR.2013.114

[29] Nitish Srivastava and Russ Salakhutdinov. Discriminative transfer learning with tree-based priors. International Conference on Neural Information Processing Systems, 2:2094–2102, 2013.

[30] Jia Deng, Nan Ding, Yangqing Jia, Andrea Frome, Kevin Murphy, Samy Bengio, Yuan Li, Hartmut Neven and Hartwig Adam. Large-scale object classification using label relation graphs. European Conference on Computer Vision, I:48–64, 2014. https://doi.org/10.1007/978-3-319-10590-1_4

[31] Tianjun Xiao, Jiaxing. Zhang, Kuiyuan. Yang, Yuxin Peng and Zheng Zhang. Error driven incremental learning in deep convolutional neural network for large-scale image classification. ACM International Conference on Multimedia, 177–186, 2014. https://doi.org/10.1145/2647868.2654926

[32] Arindam Chaudhuri. Some insights and observations on real time object detectors considering several benchmarks. Technical Report, Samsung R & D Institute Delhi, India, 2021.

[33] MS-COCO dataset: https://cocodataset.org

[34] CIFAR100 dataset: https://web.stanford.edu/~hastie/CASI_files/DATA/cifar100.html

[35] VisualQA dataset: https://visualqa.org/download.html

[36] PASCAL VOC 2007 dataset: http://host.robots.ox.ac.uk/pascal/VOC/voc2007/

[37] PASCAL VOC 2012 dataset: http://host.robots.ox.ac.uk/pascal/VOC/voc2012/

[38] Yi Sun, Ding Liang, Xiaogang Wang and Xiaoou Tang. Deepid3: Face recognition with very deep neural networks. arXiv, arXiv:1502.00873, 2015.

[39] Yi Sun, Xiaogang Wang and Xiaoou Tang. Deep learning face representation by joint identification-verification. arXiv, arXiv:1406.4773v1, 2014.

[40] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj and Le Song. Sphereface: Deep hypersphere embedding for face recognition. IEEE Conference on Computer Vision and Pattern Recognition, 6738–6746, 2017. https://doi.org/10.1109/CVPR.2017.713

[41] Xiaodan Liang, Shengmei Shen, Tingfa Xu, Jiashi Feng and Shuicheng Yan. Scale-aware fast R-CNN for pedestrian detection. IEEE Transactions on Multimedia, 20(4):985–996, 2018. https://doi.org/10.1109/TMM.2017.2759508

[42] Jan Hosang, Mohamed Omran, Rodrigo Benenson and Bernt Schiele. Taking a deeper look at pedestrians. IEEE Conference on Computer Vision and Pattern Recognition, 4073–4082, 2015. https://doi.org/10.1109/CVPR.2015.7299034

[43] Anelia Angelova, Alex Krizhevsky, Vincent Vanhoucke, Abhijit S. Ogale and Dave Ferguson. Real-time pedestrian detection with deep network cascades. British Machine Vision Conference, 32.1–32.12, 2015. https://doi.org/10.5244/C.29.32

[44] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar and Li Fei-Fei. Large-scale video classification with convolutional neural networks. IEEE Conference on Computer Vision and Pattern Recognition, 1725–1732, 2014. https://doi.org/10.1109/CVPR.2014.223

[45] Hossein Mobahi, Ronan Collobert and Jason Weston. Deep learning from temporal coherence in video. ACM International Conference on Machine Learning, 737–744, 2009. https://doi.org/10.1145/1553374.1553469

[46] Steven C. Hoi, Xiongwei Wu, Hantang Liu, Yue Wu, Huiqiong Wang, Hui Xue and Qiang Wu. Logo-net: Large-scale deep logo detection and brand recognition with deep region based convolutional networks. arXiv, arXiv:1511.02462, 2015.

[47] Hang Su, Xiatian Zhu and Shaogang Gong. Deep learning logo detection with data expansion by synthesizing context. arXiv, arXiv:1612.09322v3, 2017.

[48] Hang Su, Shaogang Gong and Xiatian Zhu. Scalable deep learning logo detection. arXiv, arXiv: 1803.11417, 2018.

[49] Andrea Vedaldi, Varun Gulshan, Manik Varma and Andrew Zisserman. Multiple kernels for object detection. IEEE International Conference on Computer Vision, 606–613, 2009. https://doi.org/10.1109/ICCV.2009.5459183

[50] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. IEEE Conference on Computer Vision and Pattern Recognition, 1–1, 2001. https://doi.org/10.1109/CVPR.2001.990517

[51] Hedi Harzallah, Frederic Jurie and Cordelia Schmid. Combining efficient object localization and image classification. IEEE International Conference on Computer Vision, 237–244, 2009.

https://doi.org/10.1109/ICCV.2009.5459257

[52] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. IEEE Conference on Computer Vision and Pattern Recognition, 886–893, 2005. https://doi.org/10.1109/CVPR.2005.177

[53] Paul Viola and Michael J. Jones. Robust real-time face detection. International Journal of Computer Vision, 57(2):137–154, 2004. https://doi.org/10.1023/B:VISI.0000013087.49260.fb

[54] David G. Lowe. Object recognition from local scale-invariant features. IEEE International Conference on Computer Vision, 2:1150–1157, 1999. https://doi.org/10.1109/ICCV.1999.790410

[55] Rainer Lienhart and Jochen Maydt. An extended set of Haar like features for rapid object detection. IEEE International Conference on Image Processing, 1: 900–903, 2002. https://doi.org/10.1109/ICIP.2002.1038171

[56] Herbert Bay, Tinne Tuytelaars and Luc Van Gool. SURF: Speeded up robust features. European Conference on Computer Vision, 404–417, 2006. https://doi.org/10.1007/11744023_32

[57] Marti A. Hearst, Susan T. Dumais, Edgar Osuna, John Platt and Bernhard Scholkopf. Support vector machines. IEEE Intelligent Systems and their Applications, 13(4):18–28, 1998. https://doi.org/10.1109/5254.708428

[58] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. Journal of Artificial Intelligence Research, 11:169–198, 1999. https://doi.org/10.1613/jair.614

[59] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. ACM International Conference on Machine Learning, 148–156, 1996. https://doi.org/10.5555/3091696.3091715

[60] Yinan Yu, Junge Zhang, Yongzhen Huang, Shuai Zhang, Weiqiang Ren, Chong Wang, Kaiqui Huang and Tieniu Tan. Object detection by context and boosted HOG-LBP. European Conference on Computer Vision on PASCAL VOC Workshop, 2010.

[61] Pedro Felzenszwalb, Ross Girshick, David McAllester and Deva Ramanan, Discriminatively trained mixtures of deformable part models, European Conference on Computer Vision on PASCAL VOC Workshop, 2008.

[62] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn and Andrew Zisserman. The PASCAL visual object classes (VOC) challenge. International Journal of Computer Vision, 88(2):303–338, 2010. https://doi.org/10.1007/s11263-009-0275-4

[63] Pedro Felzenszwalb, Ross Girshick, David McAllester and Deva Ramanan. Object detection with discriminatively trained part-based models. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(9):1627–1645, 2010. https://doi.org/10.1109/TPAMI.2009.167

[64] David G. Lowe. Distinctive image features from scale-invariant key points. International Journal of Computer Vision, 60: 91–110, 2004. https://doi.org/10.1023/B:VISI.0000029664.99615.94

[65] Timo Ojala, Matti Pietikainen and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(7):971–987, 2002. https://doi.org/10.1109/TPAMI.2002.1017623

[66] Shaoqing Ren, Kaiming He, Ross Girshick and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. arXiv, arXiv:1506.01497, 2015.

[67] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. Competition and Cooperation in Neural Networks, 267–285, 1982. https://doi.org/10.1007/978-3-642-46466-9_18

[68] Yann LeCun, Lĕon Bottou, Yoshua Bengio and Patrick Haffner. Gradient-based learning applied to document recognition. Proceedings of IEEE, 86(11):2278–2324, 1998. https://doi.org/10.1109/5.726791

[69] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. IEEE International Conference on Computer Vision and Pattern Recognition, 248–255, 2009. https://doi.org/10.1109/CVPR.2009.5206848

[70] Ross Girshick. Fast R-CNN. arXiv, arXiv:1504.08083, 2015.

[71] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan and Serge Belongie. Feature pyramid networks for object detection. IEEE Conference on Computer Vision and Pattern Recognition, 2117–2125, 2017. https://doi.org/10.1109/CVPR.2017.106

[72] Joseph Redmon, Santosh Divvala, Ross Girshick and Ali Farhadi. You only look once: Unified, real-time object detection. IEEE Conference on Computer Vision and Pattern Recognition, 779–788, 2016. https://doi.org/10.1109/CVPR.2016.91

[73] Joseph Redmon and Ali Farhadi. YOLO9000: Better, faster, stronger. IEEE Conference on Computer Vision and Pattern Recognition, 6517-6525, 2017. https://doi.org/10.1109/CVPR.2017.690

[74] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu and Alexander C. Berg. SSD: Single shot multibox detector. European Conference on Computer Vision, 21–37, 2016. https://doi.org/10.1007/978-3-319-46448-0_2

[75] Joseph Redmon and Ali Farhadi. YOLOv3:An incremental improvement. arXiv, arXiv: 1804.02767v1, 2018.

[76] A. Bochkovskiy, C.-Y. Wang and H.-Y. M. Liao. YOLOv4: Optimal speed and accuracy of object detection. arXiv, arXiv:2004.10934v1, 2020.

[77] YOLOv5: https://github.com/ultralytics/yolov5
[78] Zaid S. Sabri and Zhiyong Li. Low-cost intelligent surveillance system based on Fast CNN. PeerJ Computer Science, 7:e402, 2021. https://doi.org/10.7717/peerj-cs.402
[79] Vittorio Mazzia, Francesco Salvetti, Aleem Khaliq and Marcello Chiaberge. Real-time apple detection system using embedded systems with hardware accelerators: An edge AI application. IEEE Access, 8:9102–9114, 2020. https://doi.org/10.1109/ACCESS.2020.2964608
[80] Zhengyi Luo, Austin Small, Liam Dugan and Stephen Lane. Cloud chaser: Real time deep learning computer vision on low computing power devices. arXiv, arXiv:1810.01069v2, 2020.
[81] Daniel Barry, Munir Shah, Merei Keijsers, Humayun Khan and Banon Hopman. xYOLO: A model for real-time object detection in humanoid soccer on low-end hardware. arXiv, arXiv:1910.03159v1, 2019.
[82] Shekofa Ghoury, Cemil Sungur and Akif Durdu. Real-time disease detection of grape and grape leaves using Faster R-CNN and SSD MobileNet architectures. International Conference on Advanced Technologies, Computer Engineering and Science, Alanya, Turkey, 2019.
[83] Anil Kumar, Praneeth Chowdhary and Govinda Rao. Smart embedded device for object and text recognition through real-time video using Raspberry PI. International Journal of Engineering and Technology, 7(4):556–562, 2019. http://dx.doi.org/10.14419/ijet.v7i4.19.27959
[84] Dumitru Erhan, Christian Szegedy, Alexander Toshev and Dragomir Anguelov. Scalable object detection using deep neural networks. arXiv, arXiv:1312.2249, 2014.
[85] Matthew. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. European Conference on Computer Vision, I:818–833, 2014. https://doi.org/10.1007/978-3-319-10590-1_53
[86] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report, Computer Science Department, University of Toronto, Toronto, Canada, 2009.
[87] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. ACM International Conference on Multimedia, 675–678, 2014. https://doi.org/10.1145/2647868.2654889

Appendix

Reference	Year	Object Detectors	Significant Results
Li et al [41]	2018	SAF R-CNN	AMR: 9.32
Angelova et al [43]	2015	Deep Network Cascades	AMR: 31.11; FPS: 15
Hoi et al [46]	2015	LOGO-Net	mAP: 69.9
Su et al [47]	2017	SCL	mAP: 16.3
Su et al [48]	2018	SL^2	mAP: 46.9
Yu et al [60]	2010	Local Structured HOG-LBP	mAP: 34.3
Ren et al [66]	2015	Faster R-CNN	mAP: 70.7
Girshick et al [70]	2015	Fast R-CNN	mAP: 70.0
Lin et al [71]	2017	FPN	mAP: 59.1
Redmon et al [72]	2016	YOLO	mAP: 72.7; FPS: 155
Redmon et al [73]	2017	YOLO9000/YOLOv2	mAP: 75.5; FPS: 45
Liu et al [74]	2016	SSD	mAP: 76.8; FPS: 22
Redmon et al [75]	2018	YOLOv3	mAP: 76.5; FPS: 31
Bochkovskiy et al [76]	2020	YOLOv4	mAP: 77.6; FPS: 27
Jocher et al [77]	2020	YOLOv5	mAP: 78.6; FPS: 26
Sabri et al [78]	2021	Low-cost ISS	mAP: 99.4
Mazzia et al [79]	2020	ADS + Hardw Accelerators	mAP: 83.64; FPS: 30
Barry et al [81]	2019	xYOLO	mAP: 68.22; FPS: 9.66
Ghoury et al [82]	2019	Grape Disease Detection	mAP: 95.57

Table 10: Significant state-of-the-art research works in deep learning-based object detection (AMR: Average Miss Rate; mAP: Mean Average Precision).

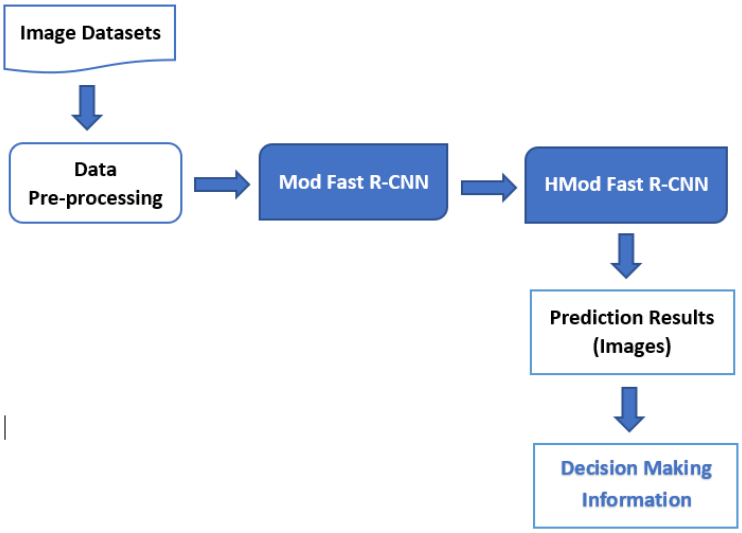


Figure 7: Prediction framework through HMod Fast R-CNN.

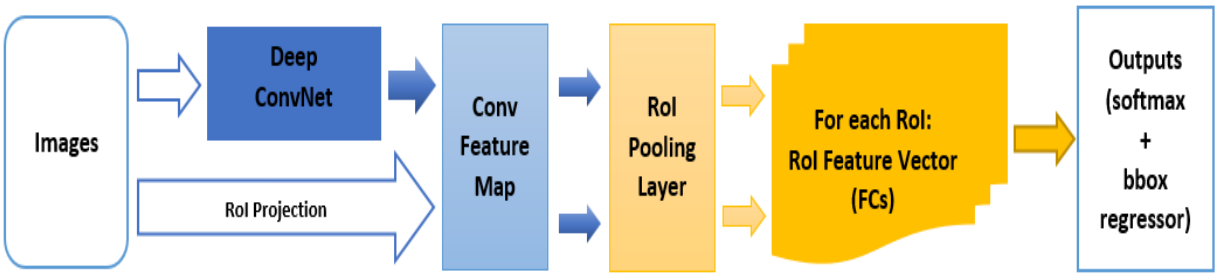


Figure 8: Architecture of Mod Fast R-CNN.

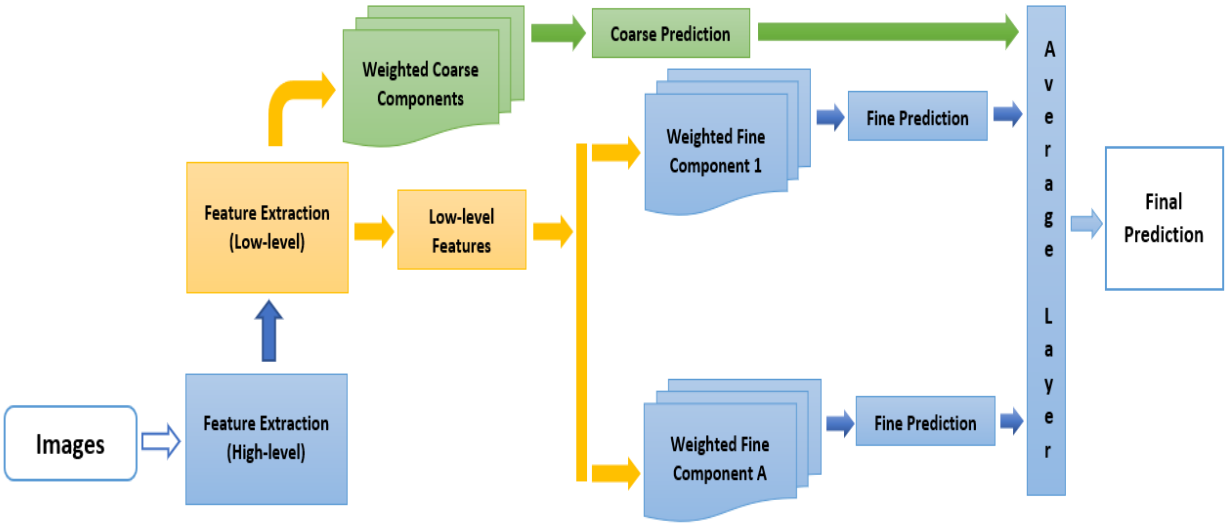


Figure 9: Architecture of HMod Fast R-CNN.