Detection of Objects Through Hierarchical Version of Fast Region-Based Convolutional Neural Networks

Arindam Chaudhuri

April 30, 2021

Detection of objects through hierarchical version of fast region-based convolutional neural networks

FirstName Surname†
Department Name
Institution/University Name
City State Country
email@email.com

ABSTRACT

In object detection there is high degree of skewedness for objects' visual separability. It is difficult towards distinguishing certain categories demanding dedicated classification. The training for the deep convolutional neural networks (CNNs) is performed through N-way classifiers. Considerable work needs to be done for leveraging structures in hierarchical category. We present here hierarchical fast region-based CNNs (Hrch Fast RCNNs) where deep CNNs are embedded considering hierarchy as categorical. The easy classes are separated through classifiers in coarse category. The difficult classes are classified by classifier in fine category. The training in Hrch Fast R-CNN is achieved by initial training of the components which follows fine-tuning globally using multiple group discriminant analysis. The regularization is done using consistency in coarse category. For large-scale recognition tasks, scalability is done considering conditional execution of classifiers in fine category and compression in layer parameters. Using CIFAR100 datasets as benchmark we obtain good results. We build four different Hrch Fast R-CNN where standard CNNs top-1 error are reduced significantly.

CCS CONCEPTS

• Computing Methodologies • Artificial Intelligence • Computer Vision

KEYWORDS

Object detection, CNN, Recognition, Classification, Scalability

1 Introduction

The image classification [1] and object detection [2], [3] tasks have shown a high degree of accuracy [1], [4] from convolutional neural networks (CNN). Most of the available techniques [2], [3], [5], [6] work in multi-stage slow and inelegant pipelines. The complexity arrives from detection which requires accurate object localization leading to: (a) processing of numerous candidate object locations and (b) achieving precise localization for candidate object locations which provide only rough localization. The solutions for the stated problems often struggle to achieve good speed, accuracy and simplicity.

The region-based convolutional neural network (R-CNN) [2] has reached brilliant accuracy in detection of objects through deep CNN towards object classification. However, it has certain drawbacks [2], [3], [5], [6]: (a) the training is performed through pipeline with multiple stages (b) there is space and time complexity involved and (c) detection of objects happen slowly. R-CNN works slowly as each object's CNN forward pass happens without sharing of computation. By sharing computation, the spatial pyramid pooling networks (SPNN) [5] speeds up the 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 through max-pooling feature-map's portion within the object with fixed output size. As in spatial pyramid pooling, 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 also reduced.

In this paper, hierarchical version of fast region-based CNN (Hrch Fast R-CNN) is proposed towards object detection that hierarchically learns to classify objects and refine them. This work looks towards the development of Hrch Fast R-CNN which integrates deep CNNs alongwith category hierarchy. The algorithm streamlines the process for training towards CNN-based object detectors [2], [5]. The image classification task is decomposed into two steps. The weighted coarse component R-CNN classifier separates classes which are easy. The complex classes are directed towards weighted fine components which takes care of classes with confusion. Hrch Fast R-CNN is build considering R-CNN building block through module design principle. The building blocks are considered to be as one of the top ranked single R-CNN. The coarse-to-fine classification is followed here. Then fine category classifiers predictions are integrated as possibilistic means which

*Article Title Footnote needs to be captured as Title Note
†Author Footnote to be captured as Author Note

WWW'19, May, 2019, San Francisco, USA
© 2018 Copyright held by the owner/author(s). 978-1-4503-0000-0/18/06...$15.00

takes care of the inherent data uncertainty. The proposed architecture is evaluated through CIFAR100 dataset [7]. Hrch Fast R-CNN achieves less error with respect to memory footprint increase as well as time of classification. The schematic representation of the Hrch Fast R-CNN based prediction system is given in Figure 1.

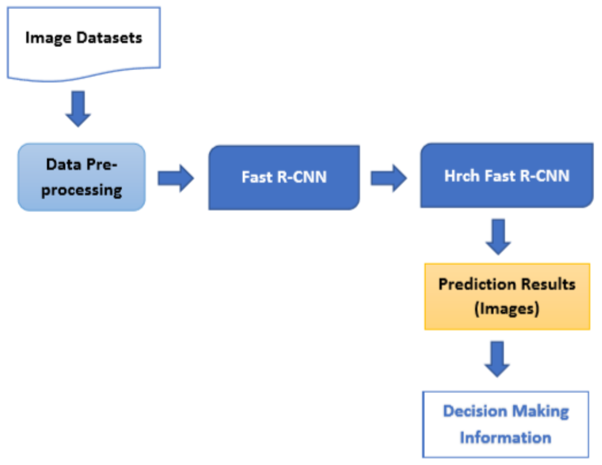

Figure 1: Prediction framework through Hrch Fast R-CNN

This paper is presented as: section 2 presents the motivation for this work. The Fast R-CNN architecture and training is given in section 3. The Hrch Fast R-CNN is highlighted in section 4. This is followed by Hrch Fast R-CNN detection in section 5. The section 6 places the experimental results. Finally, in section 7 conclusions are given.

2 Motivation for this Work

The motivation for this work is taken from the success achieved in designing CNN hierarchically with respect to the integration of category hierarchy and linear classifiers. CNN models are continuously upgraded through enhancement of their components such as pooling layers [8], activation units [9], [10] and nonlinear layers [11]. These developments have improved CNN training and learning processes. This work improves the performance of CNN model considerably. The hierarchical model is built layer-wise considering the building block as the basic CNN model. Several building blocks are placed to form the hierarchical version of deep CNN.

There exit a wide variety of structures with categorical hierarchy [12]. The classification with linear classifiers having high number of classes is generally performed through classifiers' taxonomy. Here the classifiers are verified with respect to test image which scales in sub-linear manner against number of classes [13], [14]. The hierarchy learning is either pre-specified [15], [16], [17] or achieved in top-down and bottom-up manner [18], [19], [20], [21], [22], [23], [24]. The hierarchical classifiers in [25] and [26] have reached considerable speedup bearing some accuracy loss. The initial work on category hierarchy for CNN is available in [27]. [28] achieves good accuracy considering training images subset that are re-labeled with internal nodes in class tree hierarchy. [29] uses CNN hierarchy with scalability and has good classification performance over CNN.

3 Fast R-CNN Architecture and Training

This section presents Fast R-CNN architecture adopted from [30] as the baseline method with subtle variations. The Fast R-CNN architecture is highlighted in Figure 2. The entire image and objects' set forms the input towards Fast R-CNN network. The convolutional feature map is produced through the processing of the entire image alongwith various convolutional and max pooling layers. Considering the feature map, a fixed length feature vector is extracted for each object's region of interest (RoI) pooling layer. The sequence of fully connected (fy_ct) takes each feature vector as input. From fy_ct the output is fed in 2 sibling output layers producing softmax probability estimates. The softmax probability are estimated with respect to OB object classes considering catch-all background class as well as another layer that has 4 real valued numbers as output. For each of the OB classes, 4 values' set are encoded considering bounding box positions which are refined. The max pooling is used for RoI pooling layer in order to convert its features into small feature map considering fixed $Ht \times Wh$ fixed spatial extent. Here Ht and Wh are the hyper parameters of layers not dependent on any specific RoI. RoI is just a rectangular window with convolutional feature map. A 4-tuple (rw, cm, ht, wh) defines an RoI where (rw, wh) is the 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 $ht/Ht \times wh/Wh$ size (approx). 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 spatial pyramid pooling layer in SPNN [5]. 6 pre-trained ImageNet [31] networks (with 8 max pooling layers and between 8 and 18 convolutional layers) are used to perform the experiments. There are 3 transformations for Fast R-CNN network with pre-trained network initialization. RoI pooling layer replaces the 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 and softmax layers as well as bounding-box regressors which are category specific. The network is updated to absorb 2 data inputs.

Fast R-CNN uses backpropagation to train all network weights. Below spatial pyramid pooling layer, weight updation is not possible as SPP layer's backpropagation is not effective. This inefficiency is spread across receptive field spanning the entire input image starting from each RoI. The training inputs are large as the forward pass processes the entire receptive field. The feature sharing is used during training. For each image, RoIs are sampled hierarchically through I and then R/I images for Fast R-CNN training stochastic gradient descent (SGD) mini-batches. In forward and backward passes, computation and memory are shared for RoIs from same image. Taking small I reduces the 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 the training process is synchronized through fine-tuning which optimizes softmax classifier and bounding box regressors [2], [5].

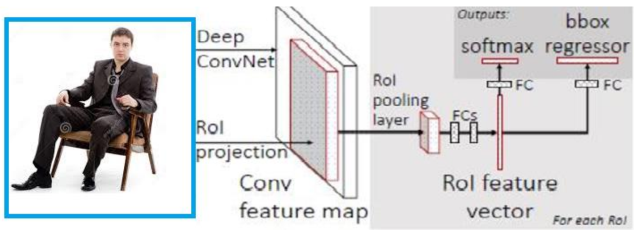

Figure 2: Architecture of Fast R-CNN

In Fast R-CNN 2 sibling output layers are used. The initial output is discrete probability distribution per RoI considering $A+1$ categories which is $prob = (prob_0, \ldots\ldots, prob_A)$. For fully connected layer, $prob$ is calculated for softmax considering $A+1$ outputs. The 2nd sibling layer has the following 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 a 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 the 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 the iverson 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 loss (bounding-box regression) 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) the 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 R-CNN and SPPN [5] 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) the 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 the loss factor [31]. 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 the 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]. The examples marked with foreground object class as $u \geq 1$ make the RoIs. The rest RoIs are sampled considering objects with maximum IoU at ground truth in $[0.1, 0.5)$ [5]. These form examples (background) and are marked as $u = 0$. A 0.1 low threshold acts as hard mining heuristic [33]. The training images are flipped horizontally considering probability 0.5. Augmentation of data is not used. Derivatives are routed through RoI pooling layer using backpropagation. All images are treated independently at forward pass since it assumes only one image per mini-batch ($N = 1$) with $N > 1$. Let $x_i \in \mathbb{R}$ as i^{th} activation input towards RoI pooling layer. Also assume y_{rj} as the layers' j^{th} output from r^{th} RoI. The RoI pooling layer computes $y_{rj} = x_{i*(r,j)}$ with $i * (r, j) = \operatorname{argmax}_{i' \in \mathcal{R}(r,j)} x_{i'}$. $\mathcal{R}(r, j)$ denotes input set's index for sub-window considering the max pooling of output unit y_{rj}. Several different outputs y_{rj} are assigned single x_i. The partial derivative of loss function is computed through RoI pooling layer's backwards function considering each input variable x_i as:

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

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 the backwards function of layer over RoI pooling layer, the 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 mini-batch iterations when training on VOC07 or VOC12. Then the 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 the momentum is 0.9 and parameter decay is 0.0005. Brute-force learning and using image pyramids are used to achieved scale invariant object detection. These approaches used here are taken from [5]. 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 small amount of running forward pass when objects are assumed to be precomputed as Fast R-CNN network 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 [5]. Considering each test RoI r, the output of forward pass 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 R-CNN algorithm [33], 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 the forward pass time required for calculating fully connected layers [33]. By compressing the large fully connected layers with truncated 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, Σ_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 is less than $min(u, v)$. 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, the 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 the RoIs number grows, good speedups are achieved through this compression.

4 Architecture of Hierarchical Fast R-CNN with Training

In this section architecture of Hrch Fast R-CNN is presented. Based on the success of Fast R-CNN, Hrch 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 $\left\{S_j^f\right\}_{j=1}^{Ct}$ contains Ct fine categories of images. The category hierarchy with A coarse categories $\{S_a^{ct}\}_{a=1}^{A}$ is used to form the learning process. Hrch Fast R-CNN emulates category hierarchy structure with coarse categories making up the fine categories.

As shown in Figure 3 end-to-end classification happens here. It consists of 5 components: (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 the leftmost side of figure 2. They take raw image pixel as input and extract the high-level features followed by the low-level features. The configuration of extraction layers is kept same as the preceding layers with respect to the building block net. The weighted coarse component independent layers assign weight factor to each of the A layers and gives the coarse prediction based on the best weight achieved. The weighted fine component independent layers assign weight factor to each of the grouped A layers and gives the fine prediction for each group based on the best weight achieved. Both these layers reuse rear layers' configuration considering the CNN's building block such that $\left\{B_{ia}^f\right\}_{a=1}^{Ct}$ with respect to image x_i. To reach the prediction $\{B_{ia}\}_{a=1}^{A}$ with respect to the coarse categories an intermediary layer is placed which transforms fine towards coarse predictions through function $F: [1, Ct] \longmapsto [1, A]$.

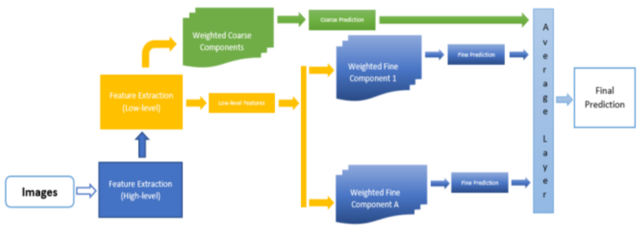
Weighted Coarse Components
Coarse Prediction
Weighted Fine Component 1
Fine Prediction
Feature Extraction (Low-level)
Low-level Features
Average Layer
Final Prediction
Images
Feature Extraction (High-level)
Weighted Fine Component A
Fine Prediction

Figure 3: Hrch Fast R-CNN architecture

The probabilities in weighted coarse category probabilities provide: (a) the weight factor towards combining predictions which the fine category components make and (b) considering the threshold conditional executions of fine category components are enabled for which the 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 the partial set do not have probabilities of other fine categories, they are taken as zero. From the building block Fast R-CNN, the layer configurations are copied. However, in final classification layer the filter numbers are taken as partial set size.

Common layers are shared for both weighted coarse category and fine category components. This is because of the reasons stated here. The preceeding layers in deep networks [34] respond towards low-level features (class-agnostic) for example corners and edges but class-specific features are extracted from rear layers. The preceding layers is 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. The Hrch Fast R-CNN parameters are also decreased which is vital towards the 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 the final prediction result. The possibilistic measures handles the inherent uncertainty in data better than probabilistic values.

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

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

As the fine category components are inserted into Hrch Fast R-CNN the parameters in rear layers increases linearly with respect to the coarse categories. This increases training complexity as well as overfitting risk considering the same training data amount. 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. Hrch Fast RCNN training is decomposed into several steps as shown in the Algorithm below.

```
**Algorithm: Hrch Fast R-CNN training algorithm**
**Procedure:** Hrch Fast R-CNN training
    **Step A:** Pre-train Hrch Fast R-CNN
        **Step A.1:** Initialize weighted coarse category components
        **Step A.2:** Pre-train weighted fine category components
    **Step B:** Fine-tune the complete Hrch Fast R-CNN
```

Hrch Fast R-CNN is sequentially pre-trained for coarse and fine category components. First a building block Fast R-CNN F^p is pre-trained through training set. There is a resemblance in building block Fast R-CNN with preceding and rear layers in coarse category component. As a result of this for initialization purpose, the weights of F^p are placed into coarse category component. Fine category components $\{F^a\}_a$ can be 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 $\{x_i | i \in S_a^{ct}\}$ considering coarse category S_a^{ct}. The initialization is done for the 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.

5 Hierarchical Fast R-CNN Detection

Once Hrch Fast R-CNN is trained the detection is performed. This section highlights this issue. The complete Hrch 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 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 the multiple group discriminant loss. The mapping $F: [1, Ct] \longmapsto [1, A]$ which is fine-to-coarse in nature paves a way towards specification of target coarse category distribution $\{t_a\}$. Here t_a is placed as fraction for all training images within coarse category S_a^{ct} with assumption that distribution for coarse categories over training dataset is near to that in trained mini-batch such that:

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

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

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

Here training mini-batch size is n and regularization constant $\lambda = 20$. As fine category components are added into Hrch Fast R-CNN, rear layers with parameters, memory footprint and execution time variables are linearly scaled with coarse categories. In order to scale Hrch 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. Hrch 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 Hrch Fast R-CNN rear layers parameter in classifiers of fine category is directly proportional to the 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 the parameter matrix $W \in R^{m \times n}$ by partitioning as segments having width s horizontally such that $W = \left[W^1, \dots \dots, W^{(n/s)}\right]$. 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 $I \in R^{m \times (n/s)}$ with cluster centers in floating number matrix $C \in R^{k \times n}$. The hyperparameters for parameter compression are (s, k).

6 Results from Experiments

The results from experiments are presented in this section. Hrch Fast R-CNN is evaluated on the benchmark dataset CIFAR100 [7], [35]. Hrch Fast R-CNN is implemented through Caffe [36]. Back propagation [1] is used to train the network. NVIDIA Tesla V100 card is used to simulate all the test experiments.

There are 100 classes of natural images in CIFAR100 dataset. The dataset is composed of 50000 and 10000 images for training and testing respectively. The pre-processing of the dataset is done using contrast normalization (globally) and ZCA-cor whitening. For training the 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 the building block of Hrch Fast R-CNN. 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 of the layers are considered as independent layers. In order to construct the category hierarchy, 10000 images are chosen at random and taken as heldout set considering the training set. There is a visual similarity for the fine categories considering the similar categories which are coarse. Pre-training is done for rear layers for fine components category. The learning rate considered initially as 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. The learning rate considered initially as 0.005 here. This decreases by factor 10 for every 10000 iterations. 10view testing [1] 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. Hrch Fast R-CNN achieves lower testing error than CIFAR100-NIN.

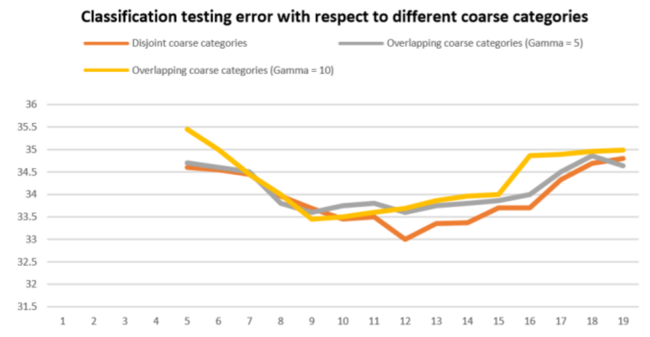

Figure 4(a): Testing error (10-view) against number of coarse categories with CIFAR100 dataset

When the category hierarchy is constructed, the clustering algorithm adjusts the coarse category number. When the hyperparameter γ is varied, the coarse categories can be made overlapping or disjoint. Their impacts are investigated on the classification error. The experiments are performed with 5, 10, 16 and 20 coarse categories with varying the values of γ. Figures 4(a) and 4(b) show that considering 10 coarse categories (overlapping) superior are achieved with $\gamma = 6$. The coarse category optimal number and γ depend on the dataset. They are also impacted through within categories inherent hierarchy.

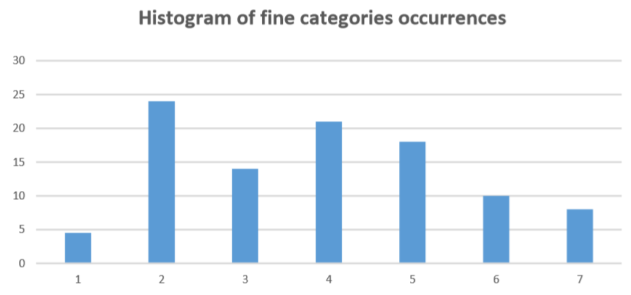

Figure 4(b): Overlapping coarse categories with respect to fine category occurrences

In comparison with building block net, the shared layers usage results in sublinear computational complexity and memory footprint of Hrch Fast R-CNN considering fine category classifiers number. Hrch 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 CIFAR100-NIN. Table 1 highlights the significance of classification error, memory footprint and net execution time. Using the pre-trained building block net, Hrch Fast R-CNN is built with coarse category and all fine category components which use independent preceding layers initialization. The central cropping is used with single-view testing where there is a slight increase in error. The memory footprint and testing time is considerably reduced through shared layers.

By varying hyperparameter β the fine category components number which are executed are affected considerably. The trade-off exists between time of execution time and error of classification. For fine category when more components are executed higher accuracy is achieved through large β values. As shown in Table 1, there is slight error increase when conditional executions are enabled through $\beta = 6$. Hrch Fast R-CNN achieves 3 times testing time as compared with building block net. The fine category 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 Table 1 Hrch Fast R-CNN memory footprint is nearly 2 times in comparison to the building block model. As a result of this it is mandatory to compare strong baseline with identical complexity for Hrch Fast R-CNN.

Model	Top-1, Top-5	Mem (MB)	Tim (s)
Base: CIFAR100-NIN	33.96	186	0.04
Hrch Fast R-CNN w/o SL	33.69	1350	2.37
Hrch Fast R-CNN	33.34	455	0.27
Hrch Fast R-CNN + CE	33.19	448	0.10
Hrch Fast R-CNN + CE + PC	31.05	269	0.10

Table 1: Testing errors, memory footprint and testing time - building block nets and Hrch Fast R-CNN: Comparative analysis on CIFAR100 dataset (mini-batch size (for testing) = 100; SL = Shrd. lyrs., CE = Cond. exec., PC = Param. comp.)

CIFAR100-NIN is adapted with doubled filters for all convolutional layers. This results in increase of memory footprint by more than 3 times. This is denoted as CIFAR100-NIN-double. The error is higher than Hrch Fast R-CNN but lower than building block net.

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

Table 2: Testing errors (10-view) on CIFAR100 dataset (CCC = coarse category consistency)

Conceptually Hrch Fast R-CNN differs from model averaging [1]. With model averaging full category sets are classified for all models. There is an independent training for each model. As different initializations are used the predictions are different. Partial category sets are classified for each classifier in fine category in Hrch Fast R-CNN. To make comparison between Hrch Fast R-CNN and model averaging, 2 CIFAR100-NIN networks are trained independently followed by their prediction average which is treated as final prediction. Table 2 shows that Hrch Fast R-CNN achieves lower error. It is noted that Hrch Fast R-CNN bears orthogonality towards model averaging. There is a considerable performance enhancement for Hrch Fast R-CNN ensembles. Hrch Fast R-CNN is fine-tuned using multiple group discriminant analysis in order to verify the coarse category consistency term effectiveness in equation (7) (loss function). Table 2 shows that higher testing error for Hrch Fast R-CNN is fine-tuned considering consistency in coarse category. There is a considerable performance improvement for Hrch Fast R-CNN using CIFAR100 datasets.

Before concluding this section, we throw some light on the design evaluation of Hrch Fast R-CNN. In this direction, several experiments were performed in order to achieve the optimal performance for Hrch Fast R-CNN. However, there remains certain questions which needs to be discussed. Some of these aspects have been addressed here and the rest of them form the future scope of work. The first question is: *Does training using multi-tasking is helpful*? The multi-task training is always useful because there is no need to manage sequentially-trained tasks pipeline. It potentially improves accuracy results as there is an influence among the tasks considering the shared representation which CNN use over here. The classification loss is one such measure which the baseline network uses during training. Another useful measure used here is the multi-task loss. It is observed here 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 both single scale and multi scale (using image pyramids). The scale of the image is specified as its shortest side length. It has been observed that both single and multi-scale has produced good results here. There are certain instances where single scale has shown the best tradeoff between speed and accuracy considering the very deep models. The third question is: *Is more training data required to verify the results*? As a rule of thumb, when trained with larger datasets, the performance of the object detector improves. The same verdict is true here. Here as and when the training data volumes are increased the object detection performance grows considerably. Further the heterogeneity in the training data helps the networks to generalize its learning capability appreciably. The fourth question is: *Is using more object proposals always better*? Generally, the object detectors use two proposal types viz object proposals with sparse set and dense set. Here dense set proposals have worked well. This has considerably improved Hrch Fast R-CNN object detection accuracy. As the proposals play here a pure computational role, increasing their number per image has produced good results.

7 Conclusion

In this work, we presented Hrch Fast R-CNN which is the hierarchical updated version of Fast R-CNN. It improves deep CNN 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 CIFAR100 dataset provide new insights. This fact is highlighted using four variant building block nets. Hrch Fast R-CNN architecture can be further extended with more than five levels. This will improve the experimental results in terms of object detection accuracy as well as accelerated the overall process considering the theoretical viewpoint. The future work looks towards developing Hrch Fast R-CNN with more layers and verifying the results with other image datasets such ImageNet, MS-COCO, MNIST, VisualQA etc.

REFERENCES

[1] Krizhevsky, I. Sutskever and G. Hinton, 2012 ImageNet classification with deep convolutional neural networks, Advances in Neural Information Processing Systems, https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[2] R. Girshick, J. Donahue, T. Darrell and J. Malik, 2014 Rich feature hierarchies for accurate object detection and semantic segmentation, IEEE Conference on Computer Vision and Pattern Recognition, https://dl.acm.org/citation.cfm?id=2679851

[3] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus and Y. LeCun, 2014 OverFeat: Integrated recognition, localization and detection using convolutional networks, arXiv: 1312.6229, https://arxiv.org/pdf/1312.6229.pdf

[4] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard and L. Jackel (1989). Backpropagation applied to handwritten zip code recognition. Neural Computation, 1(4), 541–551.

[5] K. He, X. Zhang, S. Ren and J. Sun, 2014 Spatial pyramid pooling in deep convolutional networks for visual recognition, arXiv: 1406.4729, https://arxiv.org/pdf/1406.4729.pdf

[6] Y. Zhu, R. Urtasun, R. Salakhutdinov and S. Fidler, 2015 segDeepM: Exploiting segmentation and context in deep neural networks for object detection, arXiv: 1502.04275, https://arxiv.org/pdf/1502.04275.pdf

[7] M. D. Zeiler and R. Fergus, 2013 Stochastic pooling for regularization of deep convolutional neural networks, International Conference on Learning Representations, http://www.matthewzeiler.com/wp-content/uploads/2017/07/iclr2013.pdf

[8] CIFAR100 dataset: https://web.stanford.edu/~hastie/CASI_files/DATA/cifar-100.html

[9] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville and Y. Bengio, 2013 Maxout networks, International Conference on Machine Learning, https://dl.acm.org/citation.cfm?id=3043084

[10] J. T. Springenberg and M. Riedmiller, 2013 Improving deep neural networks with probabilistic maxout units, arXiv: 1312.6116, https://arxiv.org/pdf/1312.6116.pdf

[11] M. Lin, Q. Chen and S. Yan, 2013 Network in network, arXiv: 1312.4400, https://arxiv.org/pdf/1312.4400.pdf

[12] A. M. Tousch, S. Herbin and J. Y. Audibert (2012). Semantic hierarchies for image annotation: A survey. Pattern Recognition, 45(1), 333-345.

[13] S. Bengio, J. Weston and D. Grangier, 2010 Label embedding trees for large multi-class tasks, Advances in Neural Information Processing Systems, https://papers.nips.cc/paper/4027-label-embedding-trees-for-large-multi-class-tasks.pdf

[14] T. Gao and D. Koller, 2011 Discriminative learning of relaxed hierarchy for large-scale visual recognition, International Conference on Computer Vision, https://dl.acm.org/citation.cfm?id=2356555

[15] M. Marszalek and C. Schmid, 2007 Semantic hierarchies for visual object recognition, IEEE Conference on Computer Vision and Pattern Recognition, https://ieeexplore.ieee.org/abstract/document/4270297

[16] N. Verma, D. Mahajan, S. Sellamanickam and V. Nair, 2012 Learning hierarchical similarity metrics, IEEE Conference on Computer Vision and Pattern Recognition, https://www.computer.org/csdl/proceedings-article/2012/cvpr/288P2C26/12OmNrMZppY

[17] Y. Jia, J. T. Abbott, J. Austerweil, T. Griffiths and T. Darrell, 2013 Visual concept learning: Combining machine vision and bayesian generalization on concept hierarchies, Advances in Neural Information Processing Systems, http://papers.nips.cc/paper/5205-visual-concept-learning-combining-machine-vision-and-bayesian-generalization-on-concept-hierarchies.pdf

[18] R. Salakhutdinov, A. Torralba and J. Tenenbaum, 2011 Learning to share visual appearance for multiclass object detection, IEEE Conference on Computer Vision and Pattern Recognition, https://dl.acm.org/citation.cfm?id=2191903

[19] G. Griffin and P. Perona, 2008 Learning and using taxonomies for fast visual categorization, IEEE Conference on Computer Vision and Pattern Recognition, https://authors.library.caltech.edu/18774/1/Griffin2008p85632008_Ieee_Conference_On_Computer_Vision_And_Pattern_Recognition_Vols_1-12.pdf

[20] M. Marszałek and C. Schmid, 2008 Constructing category hierarchies for visual recognition, European Conference on Computer Vision, https://link.springer.com/chapter/10.1007/978-3-540-88693-8_35

[21] L. J. Li, C. Wang, Y. Lim, D. M. Blei and L. Fei-Fei, 2010 Building and using a semantivisual image hierarchy, IEEE Conference on Computer Vision and Pattern Recognition, http://vision.stanford.edu/publications.html

[22] H. Bannour and C. Hudelot, 2012 Hierarchical image annotation using semantic hierarchies, ACM International Conference on Information and Knowledge Management, https://dl.acm.org/citation.cfm?id=2398659

[23] J. Deng, S. Satheesh, A. C. Berg and F. Li, 2011 Fast and balanced: Efficient label tree learning for large scale object recognition, Advances in Neural Information Processing Systems, https://papers.nips.cc/paper/4212-fast-and-balanced-efficient-label-tree-learning-for-large-scale-object-recognition.pdf

[24] J. Sivic, B. C. Russell, A. Zisserman, W. T. Freeman and A. A. Efros, 2008 Unsupervised discovery of visual object class hierarchies, IEEE Conference on Computer Vision and Pattern Recognition, https://www.di.ens.fr/willow/pdfs/sivic08.pdf

[25] J. Deng, J. Krause, A. C. Berg and L. Fei-Fei, 2012 Hedging yourbets: Optimizing accuracy-specificity trade-offs in large scale visual recognition, IEEE Conference on Computer Vision and Pattern Recognition, http://vision.stanford.edu/documents/DengKrauseBergFei-Fei_CVPR2012.pdf

[26] Liu, F. Sadeghi, M. Tappen, O. Shamir and C. Liu, 2013 Probabilistic label trees for efficient large scale image classification, IEEE Conference on Computer Vision and Pattern Recognition, https://www.cv-foundation.org/openaccess/content_cvpr_2013/papers/Liu_Probabilistic_Label_Trees_2013_CVPR_paper.pdf

[27] N. Srivastava and R. Salakhutdinov, 2013 Discriminative transfer learning with tree-based priors, Advances in Neural Information Processing Systems, https://papers.nips.cc/paper/5029-discriminative-transfer-learning-with-tree-based-priors.pdf

[28] J. Deng, N. Ding, Y. Jia, A. Frome, K. Murphy, S. Bengio, Y. Li, H. Neven and H. Adam, 2014 Large-scale object classification using label relation graphs, European Conference on Computer Vision, https://link.springer.com/chapter/10.1007/978-3-319-10590-1_4

[29] T. Xiao, J. Zhang, K. Yang, Y. Peng and Z. Zhang, 2014 Error driven incremental learning in deep convolutional neural network for large-scale image classification, ACM International Conference on Multimedia, https://dl.acm.org/citation.cfm?id=2654926

[30] R. Girshick, 2015 Fast R-CNN, arXiv: 1504.08083v2, https://arxiv.org/pdf/1504.08083.pdf

[31] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li and L. Fei-Fei, 2009 ImageNet: A large-scale hierarchical image database, IEEE Conference on Computer Vision and Pattern Recognition, http://www.image-net.org/papers/imagenet_cvpr09.pdf

[32] Erhan, C. Szegedy, A. Toshev and D. Anguelov, 2014 Scalable object detection using deep neural networks, arXiv: 1312.2249, https://arxiv.org/pdf/1312.2249.pdf

[33] P. Felzenszwalb, R. Girshick, D. McAllester and D. Ramanan (2010). Object detection with discriminatively trained part-based models. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(9), 1627–1645.

[34] M. Zeiler and R. Fergus, 2014 Visualizing and understanding convolutional networks, European Conference on Computer Vision, https://link.springer.com/chapter/10.1007/978-3-319-10590-1_53

[35] A. Krizhevsky and G. Hinton, 2009 Learning multiple layers of features from tiny images. Technical Report, Computer Science Department, University of Toronto, Toronto, Canada.

[36] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama and T. Darrell, 2014 Caffe: Convolutional architecture for fast feature embedding, ACM international conference on Multimedia, https://dl.acm.org/citation.cfm?id=2654889