2016 23rd International Conference on Pattern Recognition (ICPR)
Cancún Center, Cancún, México, December 4-8, 2016

MRCNN: A Stateful Fast R-CNN

Using temporal consistency in R-CNN for video object localization and classification

Philippe Burlina
Applied Physics Laboratory and Dept. of Computer Science
Johns Hopkins University

***Abstract*— Deep convolutional neural networks (DCNNs) perform on par or better than humans for image classification. Hence efforts have now shifted to more challenging tasks such as object detection and classification in images, video or RGBD. Recently developed region CNNs (R-CNN) such as Fast R-CNN [7] address this detection task for images. Instead, this paper is concerned with video and also focuses on resource-limited systems. Newly proposed methods accelerate R-CNN by sharing convolutional layers for proposal generation, location regression and labeling [12][13][19][25]. These approaches when applied to video are stateless: they process each image individually. This suggests an alternate route: to make R-CNN stateful and exploit temporal consistency. We extend Fast R-CNNs by making it employ recursive Bayesian filtering and perform proposal propagation and reuse. We couple multi-target proposal/detection tracking (MTT) with R-CNN and do detection-to-track association. We call this approach MRCNN as short for MTT + R-CNN. In MRCNN, region proposals -- that are vetted via classification and regression in R-CNNs -- are treated as *observations* in MTT and propagated using assumed kinematics. Actual proposal generation (e.g. via Selective Search) need only be performed sporadically and/or periodically and is replaced at all other times by MTT proposal *predictions*. Preliminary results show that MRCNNs can economize on both proposal and classification computations, and can yield up to a 10 to 30 factor decrease in number of proposals generated, about one order of magnitude proposal computation time savings and nearly one order magnitude improvement in overall computational time savings, for comparable localization and classification performance. This method can additionally be beneficial for false alarm abatement.**

Keywords—Region CNNs in Video, Deep Learning, ConvNets, Region Proposals, Fast R-CNN.

I. INTRODUCTION

Looking back at over two decades of work in visual detection and recognition, algorithms have used varied strategies. Non-exhaustive examples include approaches leveraging statistical moments, histogram of gradients/features, decomposable part models, or combination and ensembles thereof accrued with various machine learning techniques (e.g. [1][2][3][4][5]). These have led to varying degrees of success, some yielding very good performance for specific object detection (e.g. face [20]). But progress in convolutional neural nets (CNNs) exploiting deeper networks, large training datasets, novel non-linear layers, GPU processing, and improved techniques for ConvNet training, have recently led to significant performance improvements for general purpose image classification. It is now possible to achieve certain tasks, e.g. whole image classification, with accuracy on par or better than humans (see [6][14][15][16][17][18]).

The focus has now shifted from classifying dominant objects in images to more challenging problems such as detecting, localizing and classifying individual objects in images. Recently such efforts have produced techniques with notable accuracy/speed improvements [7][12][13][19][25].

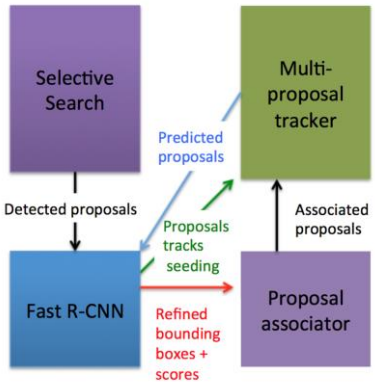

Figure 1 Flow diagram for MRCNN: (top-left): region proposals are periodically generated via Selective Search (or alternates such as Edge Boxes [27] or Bing [26]). (Bottom-left): proposals are input to Fast R-CNN which outputs scored regions that are filtered and non-maximum suppressed, producing vetted object detections that initiate tracks in MTT (upper-right). Then, proposal generation need only be computed sporadically or periodically as MTT instead provides predicted object positions as proposals input to Fast-RCNN, leading to computational savings and false alarm abatement. (Bottom-right): a Hungarian algorithm associates scored proposals output by Fast R-CNN with tracked objects in MTT.

Our work is motivated by the use of R-CNN for applications not in images but in video. It is also guided by the need for computational efficiency when implementation cannot afford the use of high-end GPUs (e.g. NVIDIA Titan) or GPU clusters, but instead is bound to utilize small embedded systems or systems on a chip (SOCs) such as NVIDIA TK1/TX1s [21][22]. These use a small GPU and processor(s) with reduced computational capabilities more consistent with deployment in smart phones or vision systems embedded in smart cameras or drones.

For real-time applications, recent progress in R-CNNs was aimed at addressing computational bottlenecks, including proposal generation [12] and proposal scoring. This has recently led to extending R-CNNs to ConvNet designs that

share convolutional layers for objectness scoring, classification and bounding box regression, where training is done with the multiple tasks of proposal generation and classification [12]. Such work includes: fast RCNN [13], Faster RCNN [12], Yolo [19] and SSD [25]. These approaches have proven to be elegant, efficient and fast, that is, consistent with video-rate processing (e.g. 5 fps for Faster R-CNN) when used with high-end GPUs (e.g. NVIDIA Titan or K40s). Additionally, accuracy was kept on par or exceeded former best of breed algorithms for Pascal VOC challenges.

When applied to video, a common feature shared by all these methods is that they are *stateless*: they all process each image individually. While there is virtue in doing stateless processing (it leads to simpler systems with no error propagation), taking video applications into account, we propose instead a different path that exploits temporal consistency and promotes proposal reuse and correction. This stateful approach has potentially key additional benefits such as incorporating object kinematics for proposal prediction and performing label association from frame to frame. At the time of submission of this work, and to the best of our knowledge, our approach was the first and only to propose a stateful R-CNN in video object detection. Very recently however, a new approach exploiting temporal tracking was reported in [32].

The method reported here exemplifies this stateful approach to video R-CNN. While it specifically extends Fast R-CNN [7][13], it is a general blueprint for augmenting other recent R-CNN approaches such as Faster R-CNN, Yolo or SSD [12][19][25]. Our approach uses Bayesian Filtering in a multiple target observer/predictor tracker (MTT) framework to propagate proposals that are vetted by the Fast R-CNN stage. It uses a Hungarian algorithm for overall K nearest neighbor optimization (KNN) to associate R-CNN vetted proposals to proposal tracks in MTT. In preliminary experiments, we show that this method provide benefits for computational savings while exhibiting classification behavior qualitatively similar to the baseline Fast R-CNN. This work is further described in the following sections: Section II details the method, Section III reports on experiments, and Section IV discusses and compares the approach with recent developments in R-CNNs including limitations and possible extensions.

Figure 2 Comparing proposals generated and used in the baseline Fast R-CNN approach (left) and MRCNN (right) where only tracked object proposals and their *bred children* are used.

II. METHOD

Our approach extends Fast R-CNN and focuses on allowing propagation, association and prediction of vetted object proposals, and is described next:

A. Overall Approach

The method leverages Fast R-CNN inside a Bayesian multiple target tracking (MTT) and data association framework. Video sequences are decomposed into cycles of fixed length including the first frame (*interframe*) and subsequent *intraframes*. Processing works as pictured in Figure **1**: at the start of each cycle, proposals are generated in interframes via Selective Search (SS) (top-left). These are fed (black arrow) to a Fast R-CNN (lower-left) that generates outputs including (red arrow) object detections with regressed positions and label for each input proposal. R-CNN output detections are trimmed down via a down-selection filtering step including non-maximum suppression and are used to initiate tracks (green arrow) in the Bayesian MTT (top-right) framework. Inside video cycles and for each intraframe: proposal generation via SS is turned off and instead the MTT generates predicted proposals (blue arrow). Each such prediction breeds *child* proposal replicas with varying aspect ratios around each *parent* proposal. For intraframes or interframes, all proposals are then input to Fast R-CNN, whose outputs are considered *observations* for MTT and then passed on (red arrow) to the association stage, associating tracks to the vetted detections output by Fast R-CNN. These associated observations are finally fed to the Bayesian MTT for track *correction*. The MTT then produces new *predicted* proposals for the next intraframe (blue arrow), completing the loop. Details follow.

B. Implementation Details

Rescaling: input video frames are rescaled to 640 by 480 in fashion similar to Fast RCNN's rescaling.

Video Cycles: processing cycles span N frames (decomposed into a first interframe followed by N-1 intraframes). N is varied in practice in the range N=5 to N=15 for videos taken at 30 fps. In experiments reported below we use N=10 for custom sequences, and N=5 for ImageNet VID sequences.

Bayesian Filtering: detected objects bounding boxes (including position, width and height) output by Fast R-CNN are used as input to MTT: these initiate tracks (for interframes) or provide track observations (for intraframes). Tracking consists of one state vector per proposal track, with no interaction among tracks. The MTT uses linear Kalman Filters (KF) [29] for each track. Each KF assumes a uniform motion for each object and includes two entries for the extremal (upper left and lower right) points in the object bounding boxes. Each extremal point subsystem has equations (*i* and *j* are the time and track indices):

$$X_{i+1,j} = A\,X_{i,j} + N_p;\ X = [x, y, x', y'] \quad (1)$$

$$A = [1,0,1,0;\ 0,1,0,1;\ 0,0,1,0;\ 0,0,01] \quad (2)$$

$$Y_{i,j} = B\,X_{i,j} + N_o \quad (3)$$

$$B = [1,0,0,0;\ 0,1,0,0] \quad (4)$$

Above, X and Y are state and observation vectors for the extremal points of the object bounding box. N_p and N_o are the process and observation noises. In experiments below the covariance matrices of observations are initialized to 3E-3.I for process and 3E-5.I for observations, with I the identity matrix.

Association/Hungarian Algorithm: A distance matrix is computed where entries encodes all image domain distances (in

pixels) between the centroids of tracks' bounding boxes and the Fast R-CNN detected objects' bounding boxes. Association is obtained by minimizing the overall cost (sum of distances) for associating each detection to track. The Munkres Hungarian algorithm [28] is used for computing the optimal association.

Proposal Breeding: Augmenting or reducing space margins around proposal bounding boxes can have a virtuous (because it adds space/context around the object) or deleterious effect (because it adds confounding nearby objects) on CNN classification performance. These effects depend on the image, the object class and training exemplars. Producing variants of base proposals bounding boxes with multiple aspect ratios allows for *context plurality* by enabling inclusion or exclusion of additional space around the object. Because of this we perform *spawning* (replication) of MTT–predicted proposals. For intraframes only, each prediction proposal obtained from a tracked object (called *the parent*) is replicated into *children* proposals forming an expanded proposal set that is then input to Fast R-CNN. *Children* proposal replicas are made up of rescaled and translated copies of the *parent*. For translated replicas, we form a grid of new offset positions around the predicted proposal center. In experiments below we use a set of offsets OS = [-10, 0, 10] (in pixels) along both the x and y directions. Therefore a number M_t = 9 of new translated children replicas are generated around each parent. For scaled replicas, proposals with modified aspect ratios are generated whereby the parent bounding boxes dimensions (w, h) are rescaled as (w_{new}, h_{new}) where $w_{new}=(1+p_w).w$ and $h_{new} = (1+p_h).h$. Two sets of values are used for scale breeding. For *large breeding* we have (p_w, p_h) in {-p, 0, p} x {-p, 0, p} (in our implementation we chose p=0.2), producing M_s = 9 scaled children replicas. We also employ a smaller scale replication scheme (*small breeding)* where we use (w_{new}, h_{new}) in {(0.05, -0.05), (0.10, -0.10)}, in this case producing M_s = 2 scaled children replicas. Combining rescaling and translation, for small breeding the total number of replicated children per original parent is $M_s.M_t$ = 18, while for large breeding $M_s.M_t$ = 81 (Figure 2).

Classes: Fast R-CNN is used and comes pre-trained on Pascal VOC classes. This includes 21 classes of which 20 are actual object classes and one class is used for the background.

Post-Processing: post processing of objects output from Fast R-CNN includes doing down-selection filtering and other steps entailed in association and MTT correction. Down-selection filtering uses: non-maximum suppression which involves removing overlapping detected objects of the same class when these have an intersection over union exceeding a set threshold T_{IOU}, filtering out objects that are not in an interest list L_I, filtering out objects whose score is less than a detection threshold T_D, and filtering out occasional malformed bounding boxes from R-CNN. Numerical values used for the experiments below are T_D=0.5 and T_{IOU}=0.3. In the experiments below L_I={person, monitor, chair, dog, car, plant, bottle} for the custom sequences experiment and L_I={bicycle, airplane, boat, car, motorbike, train, horse, dog, bird, bus} for the ImageNet VID experiment.

Proposals: for interframes, objects are found via selective search. Proposals with area less than $T_S = 40^2$ are removed. While different tuning parameters were tested for SS [24], in experiments below k_{vals} was set to 50 and 200, and the max number of iterations for blob merging was set to 50.

III. EXPERIMENTS

Datasets and Platform: We used ten custom video sequences with everyday objects, some static and some dynamic. These sequences did not include ground truth. These can be seen in [30]. We also tested using 72 ImageNet VID video sequences.

Implementation: MRCNN is implemented using the following components: OpenCV 3.1 Python bindings; Fast R-CNN [31]; SciPy's implementation of Munkres' algorithm; OpenCV 3.1 Kalman filter; and DLIB's implementation of Selective Search. Given our interest in testing on computationally-limited systems, MRCNN was deployed and tested on an NVIDIA Jetson TK1 for the custom sequences. Specifications of the TK1 include: the Tegra system on a chip which has four ARM processors, a Keppler architecture GPU, and a somewhat limited memory (2 Gb). (TK1's successor, the TX1 has improved but still limited capabilities compared to high-end GPUs.) Because of this we use AlexNet as a ConvNet in Fast R-CNN. VGG16, while having better performance for recognition and also an option for R-CNN, cannot be run on the TK1 due to memory limitations. For VID experiments we deploy MRCNN on a 6-core I7 workstation with an NVIDIA 980 GPU and 4 Gb VRAM.

Performance Evaluation: We profile MRCNN by computing the number of proposals generated per frame and the average processing time per frame for the following steps: proposal generation, labeling, post-processing, and overall processing. These metrics are reported in Tables II-IV.

Custom Sequences: comparisons are made for methods with and without MTT: method (a) is a baseline method using only SS + Fast RCNN and is reported in Table 1; two MRCNN approaches, both with N=10, including method (b) working with large breeding and method (c) with small breeding, are reported in Table II and Table III. Table IV provides a summary of gains over all sequences and methods. For custom sequences, accuracy is not reported quantitatively due to lack of ground truth. Instead, five of these sequences with overlaid result bounding boxes are available in [30] for judging accuracy. We also show a selection of intraframes from these sequences with overlaid results in Figures 3, 4 and 5 comparing all three methods. Visual inspection shows that the methods with and without MTT have qualitatively comparable accuracy performance with slight detection degradation or improvement depending on sequences/frames.

ImageNet VID Sequences: to profile time along with detection accuracy, we test on a set of 72 sequences from VID using MRCNN and N=5. Comparing MRCNN vs. Fast R-CNN as a baseline (all numbers are per frame): the average number of proposals generated for MRCNN is 196.604 (vs. 782.767 for baseline); proposal generation time is 0.088s (vs. 0.371s), labeling time is 0.044s (vs. 0.068s), post-processing time is 0.001s (vs. 0.001s), and total time is 0.133s (vs. 0.440s); (precision, recall) values averaged over all frames/sequences

are (42.666%, 75.721%) for MRCNN (vs. (44.519%, 72.983%)). Thus MRCNN provides computational saving with similar detection performance.

IV. Discussion and Comparison to other methods

Factors Affecting Performance: Closer visual inspection of the custom sequence results reveal small detection differences likely due to the following: the main drawback of MRCNN is *potential error propagation*, i.e. if an object is not scored high enough in an interframe it remains undetected/unlabeled during the rest of the cycle. This interframe failure depends on performance of both the interframe proposal generation and labeling methods (here SS and Fast R-CNN). This potentially limits the length N of the cycle. Additionally, *breeding may sometimes negatively affect detection*. In particular rescaling/dilation of the bounding box during breeding can sometime interfere with non-maximum suppression and eliminate certain detections. Conversely, it can also be seen from the videos that MRCNN is able to propagate forward proposals that are sometimes lost by the baseline algorithm either due to SS or Fast R-CNN breakdowns. There is also a visible positive effect in MRCNN abating false alarms as seen in Figures 3-5. Other errors may arise, as for example on the painting where the DCNN can have issues finding persons in artwork.

Values reported in the Tables II-V show there is a significant decrease in generated proposals in MRCNN (10 to over 30 times less). Less proposals generated has a positive effect on detection: MRCNN may avoid redundant object detection (e.g. for Seq. 7 in Figures 3-5, we see a smaller number of detections being produced per single object, the plant for example). Having lesser but better proposals has also a positive incidence on the value of the scoring threshold T_D. With fewer proposals generated, this acts as a false alarm mitigation scheme; a lower threshold on the score can then be used. This was also informally tested but is not reported here. With fewer proposals generated, processing time savings with factors up to one order of magnitude were observed.

SS tuning entailing higher numbers of proposals could have been considered—leading to higher relative computing savings in MRCNN. However increasing the number of proposals may eventually lead to swamping the classifier [12] and impact classification accuracy. Lastly, the reduction in proposals can be seen to impact labeling time. Method (c) labeling time is almost an order of magnitude better than (a) and proposal generation time is also similarly reduced. For VID, a higher N (e.g. 10) can be used for improved savings.

Comparisons to other work: Detection and classification using CNNs has *recently* first relied on *region proposal* generation. The main thrust behind proposal generation was to avoid exhaustive approaches where sliding windows are used and sub-images passed on to CNNs for label scoring. Region proposal methods have been well researched in the past few years. A possible taxonomy [12] separates methods that use selection criteria (e.g. using edge distributions) applied over sliding windows (e.g. edge boxes [25]) and methods that use segmentation and aggregation (e.g. Selective Search [24]). Several best of breed methods are reported and their performance tested in [23].

Needless to say, methods using region proposal showed computational gains over those using naïve sliding windows. Region CNN (*R-CNN*)—when first published in [7]—also showed substantial accuracy gains for detection and recognition over prior Pascal VOC leaderboard entries. R-CNN works by first finding region proposals via Selective Search in quality mode generating about 2K proposals per image. These are then fed to a CNN. *Traditional* networks (e.g. AlexNet or VGG16) are fine tuned to produce N+1 SoftMax outputs (N=20 Pascal VOC classes plus one background class). In one mode of operation, the method uses universal features out of the fine tuned network run on fixed input size images (e.g. 227x227 sized images for AlexNet) leading to 4096 features. If the detected proposal window is highly rectangular this results in substantial warping prior to classification. One against all SVMs are trained and SVM scores for each class are input to non maximum suppression. The method performs dilation of proposal bounding boxes found by the region proposal stage to include context in the warped window. Using high-end GPUs, [7] cites run times per image that improve on methods such as DPM but are still not consistent with video processing. R-CNN can take up to over 40s per image on high-end GPUs (e.g. NVIDIA K40).

Fast R-CNN's main focus is to address time/computational cost [13]. One reason R-CNN is slow is because it does not share computations and does forward ConvNet computation for each proposal. Instead, the idea is to compute convolutional layers output once over the entire image and feed the output to a max-pooling layer. In Fast RCNN, the network first processes the entire image. A second CNN block ingests that output to produce a SoftMax output and regressed bounding box location. This second CNN has a ROI pooling layer for each proposal. It, in turn, outputs regression quadruplets for position, and a SoftMax distribution vector output. Therefore each image entails a fixed computation cost (the upstream feed forward of the entire image through the first CNN) and a cost per proposal (the connected layers following proposal pooling). Each proposal is generated via Selective Search as in R-CNN.

Faster RCNN [12] addresses the cost of proposal generation, the main bottleneck in Fast RCNN, using a combined CNN strategy for both proposal generation, scoring and position regression. Faster R-CNN has two components: one is a region proposal network (RPN) made up of a fully convolutional network that generates proposals meant to replace selective search. The RPN uses a sliding window and for each anchor position generates nine different proposals with set aspect rations that are then passed to the rest of the network. The second CNN component is identical to the Fast R-CNN detector ROI pooling and fully connected layers.

Yolo is another more recent [19] approach based on related design principle of sharing network components where a network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. Speed gains are also obtained by dividing the image into a preset grid where each grid cell predicts a set number of bounding boxes. For each bounding box, prediction includes position and confidence score. Some limitations according to [19] are that small objects may not be detected and that it imposes fixed sizes on bounding boxes grids.

The above mentioned methods show significant improvements in computation time using two techniques: (1) sharing of network to do proposal generation, proposal refinement, and label scoring and (2) gridding the image and controlling the proposal location and size. These are in-image and stateless methods. Ours is a temporal consistency method that is orthogonal and complementary in design—and could be used to augment the above methods. This would provide computational benefits. It would also provide value in terms of usage of tracking kinematics, association of detection, and, as discussed above, would entail additional benefits in terms of better proposals and false alarm mitigation.

V. CONCLUSION

We describe MRCNN, a framework that—unlike prior art—exploits temporal consistency for efficient R-CNN in video. We show that MRCNN uses a significantly lower number of proposals compared to a baseline (Fast R-CNN) method, with comparable detection/classification performance. This confers computational gains for proposal generation and labeling time. Better/fewer proposals generated can lead to false alarm abatement. While this paper uses a MTT framework in combination with Fast R-CNN the design principle applies to, and similar benefits could be realized with, other object detection approaches such as Yolo or SSD. Finally, the framework addresses association which is not considered in prior methods.

ACKNOWLEDGMENTS

This work was supported by internal JHU/APL research funding. Assistance from N. Joshi and N. Fendley for the VID sequences testing is gratefully acknowledged.

TABLE I. COMPUTATIONAL PROFILE FOR BASELINE (SS+FAST RCNN)

seq	MTT	frames	prop	pt (s)	lt (s)	ppt (s)	tt (s)
seq0	0	185	3656.362	1.601	4.273	0.007	5.881
seq1	0	390	3965.264	1.806	4.676	0.006	6.488
seq2	0	448	3483.444	1.681	4.046	0.006	5.733
seq3	0	164	4790.067	2.028	5.831	0.007	7.866
seq4	0	165	4685.139	2.005	5.753	0.007	7.764
seq5	0	1091	4739.085	2.038	5.739	0.007	7.783
seq6	0	660	4776.403	2.039	5.976	0.007	8.022
seq7	0	507	4837.205	2.012	6.252	0.007	8.270
seq8	0	297	4742.219	2.011	6.192	0.006	8.210
seq9	0	870	4757.663	2.020	6.057	0.007	8.084
mean	0	477.7	4443.285	1.924	5.479	0.006	7.410

TABLE II. COMPUTATIONAL PROFILE FOR MRCNN/LARGE BREEDING

seq	MTT	frames	prop	pt (s)	lt (s)	ppt (s)	tt (s)
seq0	1	185	589.984	0.204	2.028	0.015	2.247
seq1	1	390	347.128	0.175	1.779	0.009	1.963
seq2	1	448	275.739	0.155	1.298	0.007	1.459
seq3	1	164	609.701	0.218	2.153	0.015	2.386
seq4	1	165	480.618	0.196	2.108	0.011	2.315
seq5	1	1091	468.370	0.197	1.331	0.010	1.537
seq6	1	660	459.715	0.191	1.263	0.010	1.463
seq7	1	507	498.002	0.204	1.441	0.011	1.656
seq8	1	297	346.229	0.175	1.498	0.008	1.681
seq9	1	870	518.146	0.207	1.430	0.012	1.650
mean	1	477.7	459.363	0.192	1.632	0.010	1.835

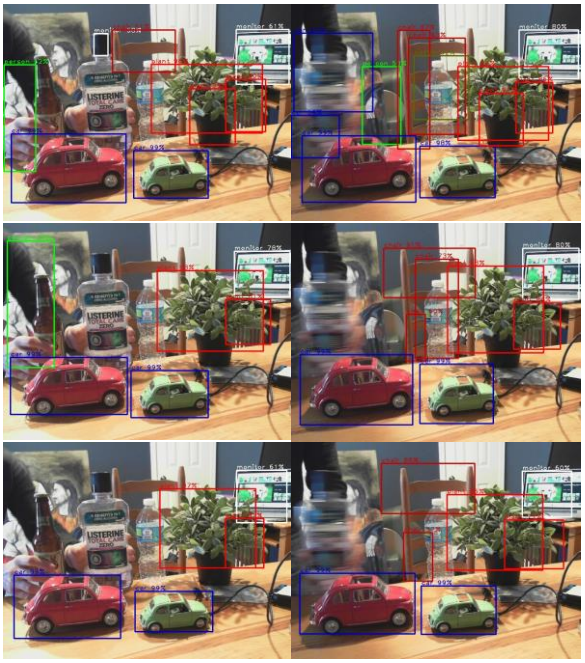

Figure 3 Comparison between (top) baseline (middle) MRCNN large breeding and (bottom) MRCNN small breeding for 2 intraframes (right: 31, left: 42) in sequence 7

Figure 4 Comparison between (top) baseline (middle) MRCNN large and (bottom) MRCNN small for 2 intraframes (right: 95, left: 107) in sequence 7

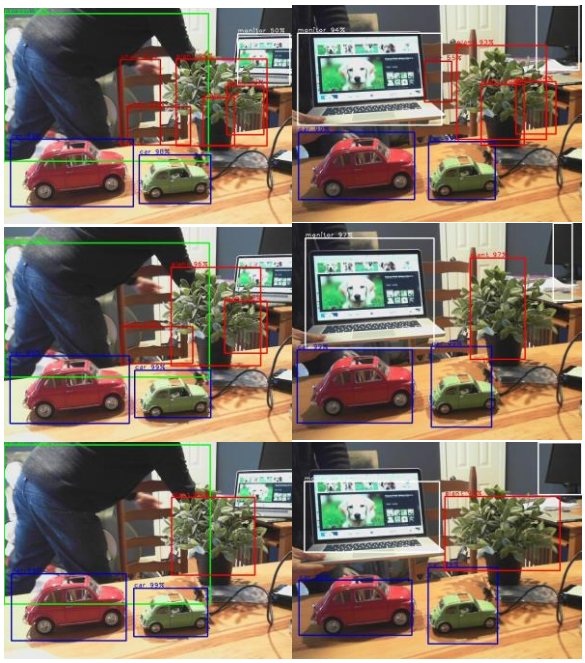

Figure 5 Comparison between (top) baseline (middle) MRCNN large and (bottom) MRCNN small for 2 intraframes (right: 116, left: 169) in sequence 7

TABLE III. COMPUTATIONAL PROFILE FOR MRCNN/SMALL BREEDING

seq	MTT	frames	prop	pt (s)	lt (s)	ppt (s)	tt (s)
seq0	1	185	154.335	0.109	0.638	0.008	0.755
seq1	1	390	102.467	0.107	0.641	0.004	0.752
seq2	1	448	82.058	0.100	0.617	0.003	0.720
seq3	1	164	164.037	0.123	0.749	0.008	0.881
seq4	1	165	134.691	0.119	0.725	0.006	0.849
seq5	1	1091	132.937	0.116	0.632	0.005	0.754
seq6	1	660	124.070	0.113	0.614	0.005	0.732
seq7	1	507	139.582	0.117	0.643	0.006	0.765
seq8	1	297	105.320	0.112	0.719	0.004	0.835
seq9	1	870	142.432	0.117	0.676	0.007	0.799
mean	1	477.7	128.192	0.113	0.665	0.005	0.784

TABLE IV. COMPUTATIONAL GAINS FOR PROPOSALS GENERATED AND PROCESSING TIMES

methods resource usage ratio	prop	pt (s)	lt (s)	ppt (s)	tt (s)
rcnn / mrcnn-large	9.67	10.02	3.36	0.60	4.04
rcnn / mrcnn-small	34.66	16.98	8.23	1.07	9.45

REFERENCES

[1] D. Lowe. Distinctive Image features from scale invariants keypoints. *IJCV*, 2004.

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

[3] Rajagopalas, A. N., Burlina, P., & Chellappa, R. Detection of people in images. IEEE International Joint Conference on Neural Networks, 1999.

[4] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *TPAMI*, 2010.

[5] Parizi, S. N., Oberlin, J. G., & Felzenszwalb, P. F. Reconfigurable models for scene recognition. CVPR, 2012.

[6] Krizhevsky, A., Sutskever, I. and Hinton, G. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems. 2012.

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

[8] Banerjee, A., Burlina, P., & Diehl, C. A support vector method for anomaly detection in hyperspectral imagery. IEEE Transactions on Geoscience and Remote Sensing, 2006.

[9] Burlina, P., DeMenthon, D., & Davis, L. S. (1992, May). Navigation with uncertainty: Reaching a goal in a high collision risk region. IEEE ICRA, 1992.

[10] Juang, R., & Burlina, P. Comparative performance evaluation of GM-PHD filter in clutter. IEEE International Conference on Fusion, 2009.

[11] Banerjee, A., & Burlina, P. Efficient particle filtering via sparse kernel density estimation. IEEE Trans. Image Proc., 19(9), 2480-2490, 2010.

[12] Ren, S., He, K., Girshick, R., & Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. In Advances in Neural Information Processing Systems (pp. 91-99). 2015.

[13] Girshick, Ross. Fast R-CNN. ICCV, 2015.

[14] Szegedy, Christian, et al. Going deeper with convolutions. CVPR. 2015.

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

[16] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R. and LeCun, Y., Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229, 2013.

[17] Russakovsky, Olga, et al. Imagenet large scale visual recognition challenge. International Journal of Computer Vision 115.3, 2015.

[18] Razavian, Ali, et al. CNN features off-the-shelf: an astounding baseline for recognition. CVPR Workshops. 2014.

[19] Redmon, J., et al. You only look once: Unified, real-time object detection. arXiv preprint arXiv:1506.02640 2015.

[20] Viola, P, and Jones, M.J. "Robust real-time face detection." International journal of computer vision, 2004.

[21] Ukidave, Yash, et al. Performance of the NVIDIA Jetson TK1 in HPC. IEEE Int. Conf. on. Cluster Computing, 2015.

[22] Paolucci, Pier Stanislao, et al. Power, Energy and Speed of Embedded and Server Multi-Cores applied to Distributed Simulation of Spiking Neural Networks: ARM in NVIDIA Tegra vs Intel Xeon quad-cores." arXiv preprint arXiv:1505.03015, 2015.

[23] Hosang, J., Benenson, R. and Schiele, B., How good are detection proposals, really?. *arXiv preprint arXiv:1406.6962*, 2014.

[24] Uijlings, J.R., van de Sande, K.E., Gevers, T. and Smeulders, A.W., 2013. Selective search for object recognition. International journal of computer vision, 104(2), pp.154-171. 2013.

[25] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., & Reed, S. SSD: Single Shot MultiBox Detector. arXiv preprint arXiv:1512.02325, 2015.

[26] Cheng, Ming-Ming, et al. BING: Binarized normed gradients for objectness estimation at 300fps. CVPR. 2014.

[27] Zitnick, C.L. and Dollár, P., Edge boxes: Locating object proposals from edges. In Computer Vision–ECCV 2014. Springer International Publishing. 2014.

[28] Munkres, James. Algorithms for the assignment and transportation problems. Journal of the Society for Industrial and Applied Mathematics 5.1 32-38. 1957

[29] Bar-Shalom, Yaakov. Multitarget-multisensor tracking: advanced applications. Norwood, MA, Artech House, 1990, 391 p. 1 1990)

[30] https://drive.google.com/file/d/0B3nkZdEsK1qIbVV1VXFLZ1hrd2s/view?usp=sharing

[31] https://github.com/rbgirshick/fast-rcnn

[32] K. Kang, W. Ouyang, H. Li, X. Wang; Object Detection from Video Tubelets with Convolutional Neural Ntworks, CVPR 2016.