ABSTRACT

Object detection in videos is an important task in computer vision for various applications such as object tracking, video summarization and video search. Although great progress has been made in improving the accuracy of object detection in recent years due to the rise of deep neural networks, the state-of-the-art algorithms are highly computationally intensive. In order to address this challenge, we make two important observations in the context of videos: (i) Objects often occupy only a small fraction of the area in each video frame, and (ii) There is a high likelihood of strong temporal correlation between consecutive frames. Based on these observations, we propose Pack and Detect (PaD), an approach to reduce the computational requirements of object detection in videos. In PaD, only selected video frames called anchor frames are processed at full size. In the frames that lie between anchor frames (inter-anchor frames), the detections from the previous frame are used in the detector, resulting in lower computational requirements. PaD leverages two key opportunities in the context of object detection, the proposed algorithm expands the ROIs greedily to provide additional background around each object to the detector. PaD can use any underlying neural network architecture to process the full-size and reduced-size frames. Experiments using the ImageNet video object detection dataset indicate that PaD can potentially reduce the number of FLOPS required for a frame by 4x. This leads to an overall increase in throughput of 1.25× on a 2.1 GHz Intel Xeon server with a NVIDIA Titan X GPU at the cost of 1.1% drop in accuracy.

KEYWORDS

Object Detection, Neural Network, Temporal Correlation, Object occupancy, Region-of-Interest packing

1 INTRODUCTION

The task of object detection in videos [3, 15–17, 23, 30, 38, 44–46] has been gaining attention in recent years. It serves as an important preprocessing task for object tracking and for several other video processing tasks such as video summarization and video search. Many object detection applications require video frames to be processed in real-time in resource-constrained environments. It is thus imperative to design systems that can detect objects in videos accurately, but also in a computationally efficient manner.

Still-image object detection has been studied extensively in the past. The accuracy and speed of still-image object detection have improved by leaps and bounds in recent years due to advances in deep convolutional neural networks (CNN). Recent CNN-based object detectors include Faster-RCNN [36], SSD [24], YOLO [33–35] and RFCN[4]. These still-image object detectors can be extended for object detection in videos by using them on a per-frame basis. However, this is inefficient as there is a strong temporal correlation between frames in a video. This temporal redundancy can be leveraged either to improve the accuracy or speed of object detection. In the recent past, there have been several attempts to improve the accuracy of object detection in videos by either integrating the bounding boxes [10, 16, 17, 39] or features [11, 15, 44, 45] across frames. However, there has not been enough attention on leveraging this temporal redundancy to improve speed. Some exceptions to this norm are [3, 23, 30, 38, 46]. In this work, we propose a method, Pack and Detect (PaD), for fast object detection in videos that can work with any underlying object detector.

PaD leverages two key opportunities in the context of object detection in videos. First, the objects of interest often occupy only a small fraction of an image. Second, there is a strong correlation between successive frames in a video. In PaD, only selected frames called anchor frames are passed in their entirety to the underlying object detector. In frames that lie between anchor frames (inter-anchor frames), the detections from the previous frame are used to identify ROIs in the image where an object could potentially be located. The ROIs are packed in a reduced-size image that is fed into the detector, resulting in lower computational requirements.

We propose a ROI packing algorithm based on the following criteria:

(1) Each ROI is expanded to provide as much background context as possible to maintain the accuracy of the detector.

(2) There is minimal loss of resolution and no change in aspect ratio to maintain the accuracy of the detector.

(3) Each object is present in a unique ROI.

(4) The space in the reduced-size frame is used as efficiently as possible.
We evaluate PaD by implementing it on top of the SSD300 object
detector and evaluating it with the ImageNet video object detection
dataset. Our results indicate that PaD reduces the FLOP count for
reduced-size frames by around 4×. Overall, PaD achieves 1.25×
increase in throughput with only a 1.1% drop in accuracy.

2 RELATED WORK
Object Detection in Videos
The temporal redundancy present in videos has been exploited
before to improve the accuracy and speed of object detection. In
[10, 16, 17, 39], the aggregation of information from neighbouring
frames is done at the bounding box level to improve accuracy. In
[16], per-frame object detection is combined with multi-context
suppression, motion-guided propagation and object tracking to im-
prove detection accuracy. In [10, 39], non-maximum suppression is
done over bags of frames. In [11, 15, 44, 45], integration of the CNN
features across neighbouring frames is used to improve accuracy. In
[15], a CNN is combined with a Long Short Term Memory (LSTM)
to obtain temporal features for object detection. In [11, 44, 45], the
features from neighbouring frames are aggregated together using
optical flow information to improve feature quality. These meth-
ods [10, 15–17, 39] pose large computational requirements, making
them often unsuitable for real-time processing.

On the other hand, [3, 23, 30, 38, 46] are relatively faster methods
aimed at object detection in videos. The methods in [3, 23, 38] are
faster by virtue of using a faster still-image object detector or an
efficient backbone network. In [46], the feature maps from selected
anchor frames are transferred to neighbouring frames by warping
them with optical flow information, leading to reduced computa-
tion. In [30], neighbouring frames are subtracted to give rise to a
spare input that is processed with a sparsity-aware hardware
accelerator [8] to achieve computational savings.

PaD differs vastly from the prior methods proposed to speed-up
video object detection. PaD can be used alongside previous methods
such as [46] and on top of existing efficient object detectors that
operate on a per-frame basis [3, 23, 38]. Moreover, PaD does not
require any specialized hardware accelerator like in [30] to obtain
computational savings.

Efficient Neural Networks
Several efforts have attempted to reduce the computational re-
quirements of neural networks. Quantization with retraining was
shown to improve the efficiency of neural network implementations
in [41]. Deep compression [9] combined pruning, trained quanti-
zation and weight compression and demonstrated large speedups
on a custom hardware accelerator [8]. Subsequent efforts have ex-
plored structured sparsity [42] by pruning filters [7, 21, 25, 28, 42]
of a CNN. MobileNet [12] replaces the standard convolution with a
combination of depth-wise and point-wise convolution to reduce
computation. SqueezeNet [13] uses network architecture modifications
to reduce the number of computations and memory. Scalable-
effort classifiers reduce computational requirements by first using
lower-complexity classifiers to process an input and subsequently
using higher-accuracy classifiers only when needed [40]. A similar
approach is taken by Big-Little networks [31]. Conditional computa-
tion [2] selectively activates certain parts of the network depending
on the input. The policy for deciding which parts of the network
to activate is learnt using reinforcement learning. Dynamic deep
neural networks (D2NN) [22] work in a similar manner to condi-
tional computation and turn on/off regions of the network using
reinforcement learning. DyVEDeep [6] reduces computations in
neural networks dynamically by using three strategies - satura-
tion prediction and early termination, significance driven selective
sampling and similarity-based feature map approximation.

The above methods focus on modifications to the network to
reduce computations and achieve speedup. In this work, we take a
complementary approach and compress the inputs that we feed into
the network. Hence, PaD is orthogonal to most existing techniques
and can be used in combination with them.

Visual Attention Mechanism
Inspired by human vision, there have been several attempts [1, 14,
18, 19, 29, 32] to reduce computation by processing an image as a
sequence of glimpses rather than as a whole. The notion of a foveal
glimpse is somewhat similar to the idea of ROI discussed here. How-
ever, there are several important differences. A foveal glimpse is a
high resolution crop of an important region in the image that is
crucial to the task at hand. In our work, we pack all the ROIs
together in a single frame and do not process them sequentially.
Further, the location of ROIs is inferred from the detections in the
previous frame in a video and does not need an attention mecha-
nism. Also, a foveal glimpse obtains crops by extracting pixels close
to the location target at high resolution and pixels far from the lo-
cation target at low resolution. We do not employ multi-resolution
processing. Hence, our work, although inspired from the notion of
foveal attention is considerably different.

Multiple Object Tracking
Object detection in video is a precursor to the problem of multiple
object tracking. Once the objects are detected in the video, the de-
tections are linked together to form a track. This problem is studied
separately from the object detection problem in the literature. The
ImageNet VID dataset used in this work does not have ground truth
labels to measure the tracking metrics. In the MOT challenge [26],
the detections that are input to the tracker are provided with the
dataset. Several popular trackers such as [20, 27, 37, 43] have gar-
nered attention through the challenge. While the use of detection
and tracking to complement each other to improve accuracy or
speed is possible, it is not well studied in the literature. In [5], the
detection and tracking have been used in a complementary fash-
ion to improve the accuracy. In future work, we will explore the
potential of combining detection and tracking to improve speed.

3 MOTIVATION

3.1 Occupancy of objects in frames
PaD leverages the hypothesis that the objects of interest occupy
only a small fraction of the area in the frame. We support this
hypothesis using statistics from a popular video dataset. Figure 1
is a histogram of the object occupancy ratio in the ImageNet VID
validation set containing 555 videos with 176126 frames. From the
figure, we see that the objects occupy only 22.7% of the frame on
average. In a vast majority of the frames, the objects occupy less than 30% of the frame.

3.2 Temporal correlation of object locations across frames

It is well known that successive frames in a video are likely to be highly correlated. We illustrate this through a statistical analysis of the ImageNet VID validation set. Figure 2 presents a histogram of the object occupancy area Intersection over Union (IoU) statistics between consecutive frames in the dataset. In the figure we can clearly see a sharp peak close to 1. On average, the IoU of areas containing objects between consecutive frames is 94.4%.

4 PACK-AND-DETECT: APPROACH AND ALGORITHMS

4.1 Overview

An overview of the PaD approach is presented in Figure 3. Full-sized video frames are processed at regular intervals (by designating the first of every $d$ frames as an anchor frame). In other frames, ROIs are identified based on the locations of the detections from the previous frame. Only detections with a minimum confidence threshold $\tau$ are taken into consideration. An ROI packing algorithm attempts to pack the ROIs into a reduced-size frame. If the packing is successful, then the reduced-size frame is processed instead, giving rise to computational savings. Once the reduced-size frame is processed using the CNN detector, the object locations are mapped back to the original frame. However, if the packing is not successful, then the frame is processed at full size, incurring an overhead due to checking whether ROI packing is possible. We demonstrate that this tradeoff is often favorable, resulting in a net improvement in the speed of object detection.

4.2 ROI packing algorithm

Figure 4 describes the ROI packing algorithm. As a first step in the algorithm, we construct a graph where nodes represent ROIs and an edge connects two nodes if the corresponding ROIs intersect. We find all connected components of this graph. We then find the enclosing bounding box over the union of ROIs in each connected component. We iterate the connected components algorithm until the final bounding boxes do not overlap. This constraint is important because if two bounding boxes overlap, then parts of the same object could be present two or more times in the packed frame. Once the number and size of the bounding boxes are decided, the layout of the bounding boxes is determined by using the algorithm presented in Figure 5. Once the layout is decided, a check is done to see whether the bounding boxes can fit in the layout. If it is not possible to fit the bounding boxes in the layout, the image is processed at full size.

If the bounding boxes can be fit in the layout, a post-processing step is performed as described below. Our experiments indicated that neural network based object detectors are often overfit to the background context of the object to be detected. Consequently, the accuracy of the object detector degrades if there is no background context. To address this challenge, we extend each bounding box to provide as much context as possible to the detector. The algorithm for extending the bounding boxes works as follows. We decide whether to first extend the boxes horizontally or vertically. For
Find all connected components in the ROI graph.

Find the enclosing bounding box around the union of bounding boxes in each connected component and form a new set of bounding boxes.

Do the new set of bounding boxes intersect?

Determine layout of bounding boxes in the reduced-size image.

Check which pairs of bounding boxes can potentially intersect when expanded in either the horizontal or vertical direction.

Decide order of dimensions for expansion.

Extend the bounding boxes along each dimension in chosen order.

Extract patches from the original image according to the final bounding boxes and place them according to the decided layout.

Figure 4: ROI packing algorithm. Sample results of the algorithm provided in Figure 6.

For the sake of discussion, let us assume that the choice is to first extend all the bounding boxes horizontally. We find all the bounding boxes that could potentially intersect when extended horizontally. We extend all bounding boxes horizontally until the layout size is reached or the bounding boxes start intersecting with each other. Then, we repeat the same procedure in the other dimension. Once the final bounding boxes are decided, the corresponding regions in the image are extracted and the reduced-size frame is composed according to the determined layout.

5 EXPERIMENTAL METHODOLOGY

The ImageNet object detection dataset (DET) is a dataset comprising 200 classes of objects that form a subset of the ImageNet 1000 classes. Further, the ImageNet video object detection dataset (VID) comprises of 30 classes of objects from among the DET 200 classes. The ImageNet video object detection (VID) dataset was the most appropriate choice for illustrating the results of our work. The ImageNet VID training set has 3862 video snippets and the ImageNet VID validation set has 555 video snippets. 53539 frames from the DET dataset comprising only of the classes from the VID dataset and 57834 frames from the VID training set were combined to form the final training set in our experiments.

The SSD300 [24] object detector operated on a per-frame basis was used as the baseline for our work. The SSD300 object detector uses VGG16 as feature extractor. The SSD300 pretrained model on the DET dataset was further trained on our training set for 210k iterations with a learning rate of $10^{-3}$ for the first 80000 iterations, $10^{-4}$ for the next 40000 iterations and $10^{-5}$ for the rest of the training. This SSD300 trained model gave a mAP score of 70 on the VID validation set. Further, this model has a network throughput of 47 fps and an overall throughput (including standard pre-processing time) of 18 fps. The SSD300 network processes images at $300 \times 300$ as the name suggests. However, closer observation of the network suggested that the same network can process $150 \times 150$ images as well by stopping processing at the penultimate layer. Hence, we use the same SSD300 network to process both full-size and reduced-size images. In all our experiments, the full size $s_1$ is 300

Figure 5: Procedure to determine layout of ROIs in a frame. Sample outputs, including cases with 1, 2 and 4 non-overlapping bounding boxes, are shown in Figure 6.
and the reduced size $s_2$ is 150. When a $150 \times 150$ sized image is passed on to the SSD300 network, processing is configured to stop at the penultimate layer. All the experiments were performed using the SSD Caffe framework running on a 2.1 GHz Intel Xeon CPU with a Nvidia TITAN X GPU. Code will be released soon. In all the experiments, the batch size was 1 to emulate a real-time processing scenario. The detection threshold $\tau$ used to select ROIs was fixed at 0.2 in our experiments unless explicitly specified otherwise.

## 6 EXPERIMENTAL RESULTS

### Results from sample videos

We show results on processing some sample videos with PaD. Figure 6 provides sample detections with our ROI-packing algorithm. The first column shows frame $i$. The second column shows the ROI packed frame $i+1$ with the detections. The third column shows the original frame $i+1$ with detections transformed from ROI packed frame $i+1$.

![Figure 6: Consecutive frames processed with the ROI packing algorithm. The first column shows frame $i$. The second column shows the ROI packed frame $i+1$ with the detections. The third column shows the original frame $i+1$ with detections transformed from ROI packed frame $i+1$.](image)

In Figure 7, we plot the per-frame time as well as the cumulative time for processing a sample video using PaD. Figure 7 provides sample detections with our ROI-packing algorithm. The first column shows frame $i$. The second column shows the ROI-packed reduced-size frame $i+1$ with the detections. The third column shows the original frame $i+1$ with detections mapped from ROI-packed frame $i+1$. For this experiment, the confidence threshold $\tau$ for selecting a detection as an ROI for the next frame was set to 0.3 for the sake of illustration. All bounding boxes with a minimum threshold of 0.2 are shown in the figure.

![Figure 7: Comparing speed of PaD to the baseline for a sample video: (a) Per-frame processing time and (b) Cumulative processing time](image)

In Figure 7(a), we observe that processing the lower sized frame of $150 \times 150$ is almost $3\times$ faster. When ROI packing fails, there is a slight overhead incurred which is visible towards the end of the video. The average per-frame speedup is around $1.25\times$ and the FLOP reduction on the average is $32\%$. The average overhead incurred for ROI-packing is around $9\%$ of the total time taken. The mAP score drops by $1.1\%$ (from $70.6$ to $69.5$).

### Results over the entire dataset

PaD was run with an inter-anchor distance $d = 5$ and $s_2 = 150$. In Figure 8, we plot the histogram of average per-frame processing time on a video-by-video basis. In other words, the average time taken per frame was obtained for each video and is plotted as a histogram across videos. From the figure, we can clearly see that the average time taken to process a frame is lower using PaD for more videos than the baseline. The average per-frame speedup is around $1.25\times$ and the FLOP reduction on the average is $32\%$. The average overhead incurred for ROI-packing is around $9\%$ of the total time taken. The mAP score drops by $1.1\%$ (from $70.6$ to $69.5$).

### Comparison with a naive ROI-packing algorithm

In order to illustrate the benefits of our ROI-packing algorithm discussed in section 4, we compare the accuracy drop when compared with a naive ROI-packing algorithm.

The naive ROI-packing algorithm can accommodate up to four ROIs just like the sophisticated method. If there are more than four objects in the frame, the frame is processed at full size. Otherwise, the bounding box surrounding each frame is extended by a factor...
of $1.2 \times$ and is treated as an ROI. If there is only one object, the ROI surrounding the bounding box is rescaled to size $s_2 \times s_2$ and is processed by the detector. If there are two objects, the lower sized frame is divided into two columns of size $s_2 \times \frac{s}{2}$. The two ROIs are rescaled to the appropriate sizes and laid out on the lower sized frame. In the case of three or four objects, the lower sized frame is divided into four regions in two columns and two rows of size $\frac{s}{2} \times \frac{s}{2}$. In the case of three ROIs, the ROIs will be rescaled to occupy three of the four regions in the frame and the fourth region will be left blank. In the case four ROIs, the ROIs will be rescaled and fit to these four regions. We do not perform a greedy expansion of the ROIs to provide additional background context. Instead, the ROIs are just expanded by a constant factor of $1.2 \times$ and rescaled to appropriate size.

PaD’s ROI packing method with inter-anchor distance $d = 5$ gave a mAP score of 69.5. With the same parameter setting, the naive ROI packing algorithm gave a mAP score of 56.8. This clearly illustrates the need for an ROI-packing algorithm that preserves the scale and aspect ratio of the ROIs and provides as much background context as possible.

With this setup, we observed $1.25 \times$ speedup with 1.1% drop in accuracy on the ImageNet VID validation set. Further, the time taken to process a lower sized frame is almost $3 \times$ lesser and the FLOP count reduces by $4 \times$.

As part of future work, we plan to incorporate a motion model to obtain the ROIs in the current frame. Incorporating a motion model could also help extend this framework to larger batch sizes. Also, it is possible to use two different models or networks to process larger sized and smaller sized frames. This will help reduce the accuracy drop but will in turn increase the memory footprint. There is an overhead incurred in checking whether the ROIs can fit in the lower sized frame. Currently, we select anchor frames at regular intervals. However, information on whether ROIs were packed successfully in previous frames can help us decide how frequently we select anchor frames. Thus, another line of future work is a dynamic mechanism for selecting anchor frames in order to reduce the overhead. It would be interesting to test PaD in more resource constrained platforms like mobile GPUs and CPUs. We expect the benefits to be more pronounced in such platforms.

7 CONCLUSION AND FUTURE WORK

Still-image object detection has improved by leaps and bounds in recent years due to the success in training and deploying neural networks. However, the opportunities that are available in the context of videos have not been fully exploited. Neural networks are in general very compute-intensive. In this work, we use the opportunities available in the context of videos to speed up and reduce the amount of computation in neural network based object detectors. In the proposed method, called PaD, the full-sized input is only processed in selected anchor frames. In the inter-anchor frames, ROIs are identified based on the locations of objects in the previous frame. These ROIs are packed together in a reduced-size frame that is fed to the CNN object detector. The ROI packing algorithm needs to ensure that the scales and aspect ratios of the objects are preserved and enough background context is provided.

![Figure 8: Histogram of average per-frame processing time on a video-by-video basis](image)

ACKNOWLEDGMENTS

This work was supported by Intel India, the Robert Bosch Centre for Data Science and AI (RBC-DSAI) and the Center for Computational Brain Research (CCBR) at IIT Madras.

REFERENCES

[1] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. 2014. Multiple object recognition with visual attention. arXiv preprint arXiv:1412.7755 (2014).
[2] Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. 2015. Conditional Computation in Neural Networks for faster models. CoRR abs/1511.06297 (2015). arXiv:1511.06297 http://arxiv.org/abs/1511.06297
[3] Xingyu Chen, Zhengxing Wu, and Junzhi Yu. 2018. TSSD: Temporal Single-Shot Object Detection Based on Attention-Aware LSTM. CoRR abs/1803.00197 (2018). arXiv:1803.00197 http://arxiv.org/abs/1803.00197
[4] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. 2016. R-fcn: Object detection via region-based fully convolutional networks. In Advances in neural information processing systems. 379–387.
[5] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. 2017. Detect to track and track to detect. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 3038–3046.
[6] Sanjay Ganapathy, Swagath Venkataramani, Balaraman Ravindran, and Anand Raghunathan. 2017. DyVIDeep: Dynamic Variable Effort Deep Neural Networks. CoRR abs/1704.01137 (2017). arXiv:1704.01137 http://arxiv.org/abs/1704.01137
[7] Jia Guo and Miodrag Potkonjak. 2017. Pruning Filters and Classes: Towards On-Device Customization of Convolutional Neural Networks. In Proceedings of the 1st International Workshop on Deep Learning for Mobile Systems and Applications (EMDK ’17). ACM, New York, NY, USA, 13–17. https://doi.org/10.1145/3089801.3089806
[8] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A. Horowitz, and William J. Dally. 2016. EIE: Efficient Inference Engine on Compressed Deep Neural Network. In Proceedings of the 43rd International Symposium on Computer Architecture (ISCA ’16). IEEE Press, Piscataway, NJ, USA, 243–254. https://doi.org/10.1109/ISCA.2016.30
[9] Song Han, Huizi Mao, and William J. Dally. 2015. Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. CoRR abs/1510.00149 (2015). arXiv:1510.00149 http://arxiv.org/abs/1510.00149
[10] Wei Han, Pooya Khorrani, Tom Le Paine, Prajit Ramachandran, Mohammad Babaeizadeh, Honghui Shi, Jianan Li, Shuicheng Yan, and Thomas S Huang. 2016. Seq-nms for video object detection. arXiv preprint arXiv:1602.08465 (2016).
[11] Conghui Hetang, Hongwei Qin, Shaohui Liu, and Junjie Yan. 2017. Impression Network for Video Object Detection. CoRR abs/1712.05896 (2017). arXiv:1712.05896 http://arxiv.org/abs/1712.05896
[12] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. CoRR abs/1704.04861 (2017). arXiv:1704.04861 http://arxiv.org/abs/1704.04861

This work was supported by Intel India, the Robert Bosch Centre for Data Science and AI (RBC-DSAI) and the Center for Computational Brain Research (CCBR) at IIT Madras.
