Fast Object Detection in High-Resolution Videos

Ryan Tran, Atul Kanaujia, Vasu Parameswaran
Percipient.ai, Santa Clara, CA

Abstract

Despite the rapid evolution of video resolutions and progress on object detection algorithms, processing high resolution videos has had three main challenges so far. Firstly, it is non-trivial to use existing tracking algorithms to extend an object detection framework for efficient processing of high resolution videos. In theory, fully convolutional CNN architectures in most existing deep learning models allow any input resolution to be processed. However, in practice, inferencing on high resolution images decoded from a video incurs significant computational costs, making it impractical for real-time applications. Secondly, most tracking approaches typically require the entire frame to be decoded. Relatively little work has gone into object detection directly on compressed data, which include rich temporal cues that can be exploited to reduce the computational cost at inference time. Thirdly, most of these approaches require labeled data for training models, thereby limiting their adoption.

We tackle all the three challenges in our framework by incorporating forward and backward motion cues from the compressed video to dramatically increase the processing speed of a pretrained baseline object detector, without any loss of accuracy. Our training is based on knowledge transfer from the baseline detector as a teacher network, thereby forgoing the need for any labeled data. Finally, the models are agnostic to teacher network architecture, and can be used to improve efficiency of any object detector. Our results show a speed gain of 3x to 20x compared to a frame-by-frame detector, depending upon input data resolution.

1. Introduction

The proliferation of smartphones and security cameras has resulted in a significant increase in the number of high resolution videos. With the advent of deep learning algorithms, object detection methods have become considerably more accurate and robust in their ability to learn and identify complex spatial patterns in videos. Most existing live camera processing and alerting frameworks process data at low resolutions (typically 512×512) even though fully convolutional networks allow inputs to be processed at any resolution. This results in the loss of the ability to detect smaller sized objects, which may be critical for many applications. Object detectors trained for variable input resolutions (e.g. up to 1024×1024) have the ability to process inputs at much high resolutions, allowing smaller sized objects to be detected. However their use as-is requires higher computational cost, continuous decoding of large images, and demands significantly high throughput from the hardware. Classical approaches of splitting video streams into parallel low-resolution streams[23] or pyramidal processing also do not scale well due to significantly higher processing costs. A natural solution therefore is to process a high resolution video sparsely, and apply fast tracking algorithms to propagate detections in the intermediate frames.

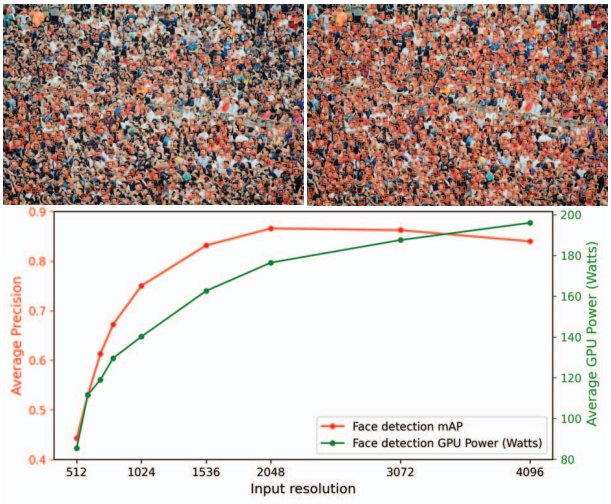

Figure 1: Processing input images at high resolutions has clear benefits of enabling user to detect small objects. *(top)* Processing a 6000×4000 image at 512 resolution, detects significantly fewer faces(222) compared to detector at 2048 resolution shown on the right(894). *(bottom)* Average Precision(AP) accuracy of an SSD[15] face detector on a set of ultra high resolution images (6000×4000) progressively improves with the detector resolution, although it comes with a cost of higher power and resource requirements.

In the past, researchers have attempted to incorporate

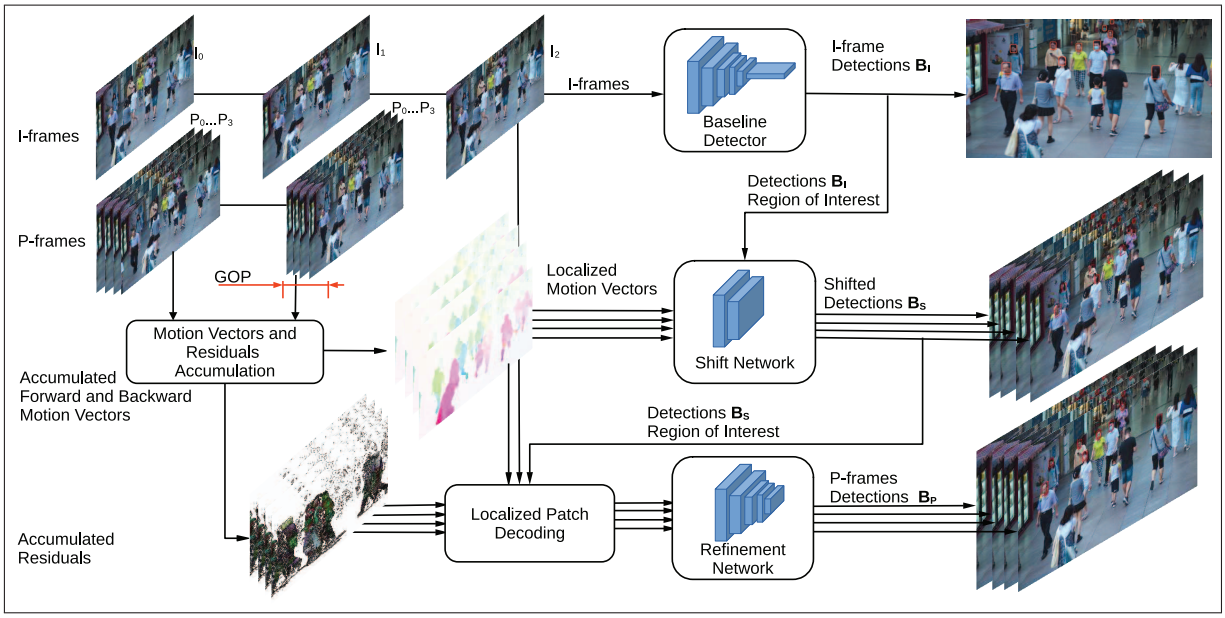

Figure 2: Overview of the entire system for processing high resolution videos in compressed domain. Videos are encoded as series of interleaved I-frames and P-frames. Only I-frames are processed using the baseline detector. Motion vectors and the residuals associated with the GOP P-frames are accumulated in forward and backward directions to enable parallel processing. The Shift network uses accumulated motion vectors from the preceding and succeeding I-frames to predict shifts in the bounding boxes. The Refine network refines both the set of bounding boxes from the localized reconstructed patches.

temporal context for tracking detections, although standard sequence modeling approaches like recursive nets and LSTMs have not been widely applied, mainly due to their high computational costs and memory requirement. While a majority of approaches require the frames to be decoded, some have attempted to run inference directly in the compressed domain. Video compression algorithms are fairly mature and highly optimized for storage and bandwidth. Decoding the entire video frame and processing each frame individually is redundant and wasteful of computing resources, especially if the changes are limited to small localized regions in the scene.

In this paper, we introduce a novel framework that uses a generic full-frame object detector (referred to as the *baseline detector*) that detects one or more object classes in an image, and two customized lightweight neural networks called the *Shift* network and the *Refine* network. The baseline detector may be optimized for high resolution frames such as HD or 4k-UHD. The Shift and Refine networks on the other hand are much faster than the baseline network, and do not require the entire image to be decoded. Our framework speeds up video processing by depoying the computationally expensive baseline detector on a sparse set of frames, and propagating the detections to obtain refined detections on the intermediate frames using the Shift and Refine networks. While the Refine network is critical to overcoming the lack of rich motion information in the compressed video (see figure 3), the Shift network ensures that decoding a frame patch is only needed in a localized region, which amplifies speed gains. The video decoding process is therefore tightly coupled with the baseline detector, requiring only sparse and localized reconstructions in the majority of frames. The framework thus facilitates processing very high resolution videos without the need to decode the entire frame, while maintaining the detection accuracy of the baseline detector. The Shift and Refine networks are trained on the output distribution of the baseline detector. The training happens seamlessly on unlabeled representative data in a fully unsupervised fashion, using teacher-student knowledge distillation. Figure 2 shows the overview of the system. To summarize, our work demonstrates that (1) spatially localized processing of high resolution videos using tight coupling of video decoding and processing can yield significant performance gains without any loss of accuracy, (2) knowledge distillation based learning can be effectively used to train the lightweight Shift and Refine networks from mature object detectors, thus eliminating the need for labeled data, and (3) a generic, detector agnostic pipeline can be used for fast processing of high resolution videos.

2. Related Work

Methods seeking to speed up computer vision tasks in video typically run a more expensive model on a sparse set of frames, and fill in the rest of the frames by tempo-

rally propagating results using changes in the feature maps or motion and optical flows. The task of action recognition in compressed domain has been somewhat successful,(e.g. [33][24][35][29][25]). Wu *et al* [33] was one of the early works incorporating accumulated motion vectors and residuals in a loosely coupled framework for fast inference on videos. They independently trained separate models for I-frames, motion-vector frames and residual frames. The action recognition scores from individual models were aggregated by simply summing individual model scores. DMC-net [24] is a lightweight generator network to reconstruct flow-like signals from low resolution, imprecise motion vector signals in compressed videos. Its authors demonstrated significant speed gains at the same level of accuracy, compared to frameworks that used optical flow. Two-stream convolution networks[25] have found success in action recognition but are inherently slow due to optical flow computation. Zhang *et al* [35] improved it 27× by replacing optical flows with the motion vectors from the compressed videos which tend to be much coarser and noisy. Context and motion decoupling [10] advocates use of motion vector cues in the videos for embedding high-level motion representations in their learning framework using self-supervision. While methods for action recognition on compressed data may inspire ideas for object detection, they cannot be used directly for object detection, which requires spatially finer grained propagation.

Object detection in video has also seen some success [14][36][16][8][4][5][6]. Information from the previous frames can be incorporated as either multi-level feature maps from different stages of the deep network inference pipeline or detections (of which, tracking is an example). Liu *et al* [14] have proposed a tracking based detector that employs CNN to perform appearance based data association in non-key frames that is 6x faster than the detector. Their framework is jointly trained with the detector and critically relies on existence of labeled training data in videos. It is infeasible to extend it to new objects without retraining with the detector. A large number of works (e.g. [36],[4]) employ pixel tracking (flows or MHI) to propagate features maps to non-key frames. These frameworks are orthogonal to our approach which uses existing video encoding to propagate detections. Luo *et al* [16] built a scheduling Siamese network to adaptively switch between detection or tracking depending on the contents. Wang *et al* [30] building upon [31] design a light-weight memory network (LSTM) to propagate features across multiple scales. They use attention to extract relevant regions in the feature map for propagation. In [32] the authors train a network to propagate features from motion vectors and residuals in the P-frames. They propagate features across P-frames (short term) and use optical flow across I-frames (long term). All of these methods require labeled data for training. DeltaCNN [20] is

Figure 3: Comparison of optical flow(second row)[11] and forward motion vectors(third row) extracted from P-frame for three frames. Fourth row illustrates backward motion vectors from the successive I-frame. Evidently flow provides far more precise information to accurately track targets compared to motion vectors from the video compression algorithms. Motion vector patterns are noisy, discontinuous and overlapping in the presence of multiple objects. The Shift network is trained to estimate motion patterns of a target by amplifying the motion cues and overcoming the noise embedded in these signals.

a framework for processing changes in the inputs sparsely. It extends deep network operations to support incremental changes (delta) as input to produce changes in the output. Linear operations can handle deltas while non-linear operations need dense-accumulated inputs to generate deltas in the output. DeltaCNN can only work with static cameras and not moving cameras. Followup work [19] extended DeltaCNN for panning camera motion.

Our work, though not tracking, falls in the category of propagating detections using motion vectors encoded in videos. We employ both forward and backward motion vector cues to propagate detections. This minimizes the risk of missing out targets appearing in the non-key(I) frames of the video. One of the earliest methods methods [34] attempted to use motion vectors embedded in MPEG compressed videos to track a moving object and developed an algorithm to continue tracking it even when the object has stopped moving. Motion vector interpolation and prediction based methods [28][12][27][1] essentially employ simple approaches to use noisy motion cues from the video compression and are not robust to objects moving in a crowd or with complex patterns. These frameworks do not support moving camera and cannot handle changes in the size of the target due to camera motion. Knowledge transfer using teacher-student training has been widely applied in var-

ious domains of vision (a good survey is [7]). We employ teacher-student learning to train lightweight models from a pre-trained object detector. To our knowledge, ours is the first approach to use teacher-student learning for propagating object detections. Processing high resolution videos has been challenge for deep learning and only a few works exist addressing it [23]. Our work makes processing very high resolution practically feasible for a broad category of detection frameworks.

3. Parallelized Video Processing

Video compression: We work with the H.264 encoding format which is representative of efficient compression algorithms used in modern video encoders [18]. In a typical video, frames are encoded as series of interleaved I-frames, P-frames and B-frames. An I-frame (intra-coded) is a compressed image which is *independent* of the rest of the frames in the video. The compression in I-frame exploits spatial redundancy in the pixels to encode blocks of image in the frequency domain (DCT), similar to JPEG level compressions. P-frames (predictive) and B-frames (bi-predictive) are motion compensated, differential frames from the nearby P/B/I-frame. Group of P/B-frames (Group Of Pictures or GOP) between consecutive I-frames, $\mathbb{N}_G$, is a critical parameter to balance accuracy and speed gains feasible in our framework. An H.264 video has variable GOP size depending on the content and motion magnitude captured by the P-frames between a pair of I-frames, $\{I_0, P_0^0, P_1^0, \ldots, I_1, P_0^1, P_1^1, \ldots\}$[1] A P-frame P_0 is composed of a motion vector frame M_0 (encoded as macro blocks) and entropy-encoded residual frame R_0. Bi-directional predictive B-frames use backward motion compensation in addition to forward motion compensation modeled by the P-frame. We omit further analysis of B-frames, without loss of generality, as they can be handled in the same way in reverse temporal direction.

Motion vectors are extracted as a 2d matrix [26][2] of variable sized macro blocks in M_0. We maintain both forward and backward sets of motions vectors using previous and next I-frame respectively. The missing motion vector information from the last P-frame to the next I-frame is approximated as zeros without any effect on the accuracy. For each macro block, we maintain the source frame (previous and next I/P-frame), the center of a macro block in both the source and destination frame, as well as the width and height of the macro block. Temporal dependency between consecutive P-frames is removed by accumulating the changes for each consecutive P-frame and using only the corresponding I-frame as the reference source frame. This requires a fast pass across $\mathbb{N}_G$ P-frames to accumulate the motion vectors and the residuals. For each P-frame,

[1] We will drop superscript from the P-frame in future as it always refers to the previous I-frame.

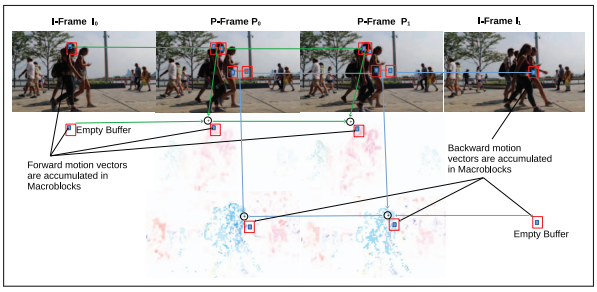

Figure 4: Motion vector accumulation, both in forward and backward directions, happens by passing an empty buffer for each macro block from I-frame across consecutive P-frames, and adding motion vectors for every pair of source and destination frame. Unlike accumulation proposed in the [33] that moves the motion vector tail for every pair of frames, our implementation fixes the tail and only strengthens the vector during accumulation.

we maintain a 2-channel image of motion vectors, denoting how far the corresponding pixel-coordinate in the I-frame has shifted along the x and y axes. Accumulated shift at each pixel co-ordinate is computed by summing up the x/y motion vectors of all the macro blocks covering that coordinate. Macro blocks are not guaranteed to be discrete in the source frame. This means that our accumulation algorithm can copy an I-frame coordinate to multiple destination locations which results in more than one macro block trying to increment the accumulated shift for an I-frame coordinate. This happens for less than 5% of the total pixels, and can be safely ignored when using multiple threads for accumulation.Figure 4 illustrates the accumulation process.

4. Shift Network

The Shift network is by design, a motion prediction function $\mathcal{F}_S$ to infer target motion using accumulated motion vector patterns in the P-frames. This lightweight model takes as input, the list of RoI (region of interest) from the current frame, for each bounding box detected in the referenced I-frame. The motion vector patterns in the RoI capture the trajectory of the target in a video. For each bounding box **B**=[cx, cy, w, h], the Shift network uses `RoIAlign`[22] to predict a three dimensional vector Δ**B**=[Δcx, Δcy, Δs] as the shift vector along the x and y axes, and the scale space s. As the motion vectors in the P-frames are defined over macro blocks of pixels 8×8, the resolution of the motion vector image is much lower than that of the input frame. We take advantage of this by pooling the pixels into 8×8 buckets during the accumulation pass. This vastly cuts down on the computational cost while having minimal effect on detection accuracy. Target translation in an image is a composite of target motion and

camera motion, which is only approximately captured by motion vector patterns in the P-frame. Figure 3 illustrates the lack of precision in the motion vectors when compared to the optical flows[11] in the same frame.

Learning using Knowledge Distillation: $\mathcal{F}_S$ is learned as a two-layer shallow network using a teacher student framework. Specifically, supervisory soft tracking labels from the targeted detector are used to steer the training of the Shift network. The input motion vectors are globally normalized for the entire frame. This enables better discrimination between motion patterns due to target motion and the camera ego-motion. Labels for the Shift network are relative offsets ΔB_P of the detection bounding box B_P in the current P-frame, that corresponds to the bounding box B_I in the source I-frame. Offsets of centers and scale in the corresponding I and P-frames of size $[\mathrm{I}_h, \mathrm{I}_w]$ are modeled as

$$\begin{aligned} \Delta\mathrm{cx} = (\mathrm{cx}_P - \mathrm{cx}_I)/\mathrm{I}_w, \Delta\mathrm{cy} = (\mathrm{cy}_P - \mathrm{cy}_I)/\mathrm{I}_h \\ \Delta\mathrm{s} = \log\left(\frac{\sqrt{\mathrm{w}_P\mathrm{h}_P}}{\sqrt{\mathrm{w}_I\mathrm{h}_I}}\right) \end{aligned} \tag{1}$$

Soft labels from the teacher network in the current frame are obtained by tracking the bounding box B_I in the reference I-frame. For training, we use single object tracking[13] based on a pretrained Siamese-RPN framework to independently track each of the bounding boxes, followed by matching using the Hungarian algorithm with IoU (Intersection over Union) as the metric. For each pair of boxes that share the same tracklet ID such that the first box is in the I-frame B_I and the second is in the P-frame B_P, we use ΔB_P = B_P - B_I as the target label with an associated weight $\mathcal{M}(B_I, B_P)$. The weight denotes the matching score

$$\frac{(\mathcal{P}_D(B_P) + \mathcal{P}_T(B_I, B_{I,P}) + \alpha\mathcal{P}_{IoU}(B_{I,P}, B_P))}{(2.0 + \alpha)} \tag{2}$$

where $B_{I,P}$ is the intermediate tracked box from the I-frame, $\mathcal{P}_D$ is the teacher network detection confidence, $\mathcal{P}_T$ is the accumulated tracking confidence normalized to 1.0, and $\mathcal{P}_{IoU}$ is the box overlapping score with $\alpha < 1.0$. We give high weights to detection and tracking score compared to the degree of overlap when assessing validity of a label. This is to filter out cases where the observed target may be getting occluded or losing track. Also, more importantly, motion vector cues are imprecise and cannot be expected to track bounding boxes with high overlapping ratio. Training loss for the student network $\mathcal{F}_S$ is modeled as $\mathcal{M}(B_I, B_P)$ * $\mathtt{smooth}_{\mathrm{L1}}(\Delta B, \Delta B_P)$.

5. Refine Network

Coarse motion cues from the P-frames result in the Shift network producing imprecise target localizations. Additionally, targets may also get occluded or move out of the scene. The Refine network utilizes patches reconstructed from accumulated residuals and motion vectors to further refine the shifted bounding boxes. It has both classification and regression heads. The classification head attempts to infer possible disappearance of the object due to occlusion or its leaving the scene. The regression head predicts a refined bounding box relative to the shifted bounding box B_S using the reconstructed (decoded) patch. For a shifted bounding box B_S obtained from the Shift network, we use decoded image patch $\mathcal{F}_{expand}(B_S; W)$ as inputs to the refine model. More context around the bounding box B_S by rescaling its size by W improves the refinement accuracy. Patch reconstruction uses `RoIAlign` to extract variable sized patches from accumulated residuals, motion vectors and the reference I-frame, and resizes them to fixed sized inputs for processing using the Refine network. Additionally, the confidence scores of the detections predicted by the baseline detector in the I-frame are used as informative features in the refinement model training. These scores are concatenated with the patch feature in the input.

Learning using Knowledge Distillation: Detections from the teacher network with scores greater than 0.01 are used as soft labels for training the Refine network. These labels are matched to the shifted detections B_S of the referenced I-frame using a custom overlapping metric that emphasizes the fraction of the pseudo-label area covered by the shifted bounding box.

$$\mathcal{O}_P = \frac{B_S \cap B_P}{\mathrm{Area}(B_P)} \cdot \frac{B_S \cap B_P}{B_S \cup B_P} \tag{3}$$

This metric intentionally down weighs the need for B_S to have similar size as B_P, but favors enclosing larger context for refinement by the Refine network. Note that B_S could be large and enclose multiple labels. In those cases we prefer closest labels as discussed later in this section. Associations with $\mathcal{O}_P < 0.25$ are treated as negative examples. We allow any B_P to match to multiple B_S.

The classification head is trained to detect occlusion or disappearance of shifted box. We use teacher detection score $\mathcal{P}_D$ to create soft labels for the knowledge transfer. The classification loss function is formulated as:

$$\mathcal{L}_{conf}(B, B_P) = -\sum_{k\epsilon\mathrm{Classes}} \mathcal{P}_D^k \log\left(\frac{\exp(c^k)}{\sum_{i\epsilon\mathrm{Classes}}\exp(c_i)}\right) \tag{4}$$

where c^k are the classification scores for the examples from the Refine network. Negative training labels are also generated by randomly sampling boxes of variable sizes around a valid target or as hard negatives during the training process.

The regression head is trained to output offsets relative to the center of the shifted bounding box $\mathcal{F}_{expand}(B_S; W)$ with the pseudo labels as ΔB_P = B_P - B_S. The regression loss is defined only for positive samples and is re-weighted

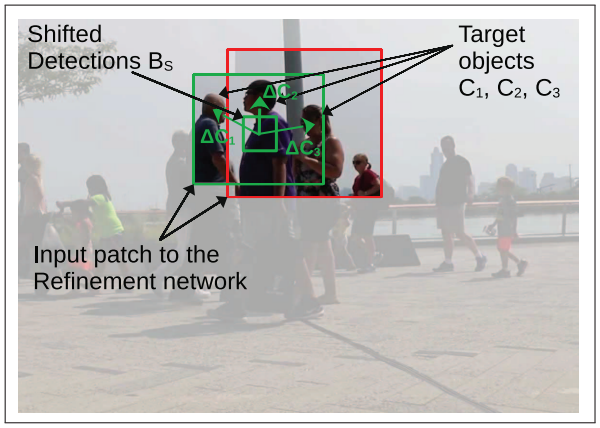

Figure 5: Center prior is used to train models that favor refined boxes that are close to the center of the input patch. Here we illustrate an example of ambiguity due to presence of multiple target objects (faces) in the input patch. Amongst the three possible targets, the prior favors target two as $\Delta C_2 < \Delta C_1 < \Delta C_3$

by $\mathcal{P}_D$:

$$\mathcal{L}_{loc}(B, B_P) = \mathcal{P}_D \text{smooth}_{\text{L1}}(\Delta B, \Delta B_P) \quad (5)$$

Where ΔB is the offset relative to B_S as outputted from the refine model. An appropriately rescaled image patch from $\mathcal{F}_{expand}(B_S; W)$ is critical to disambiguate cases when the patch contains multiple possible targets in the patch. In those cases we add a prior term to favor bounding boxes closer to the center of the patch:

$$\mathcal{L}_{center}(B) = \beta \|(\Delta \text{cx}, \Delta \text{cy})\|_2 \quad (6)$$

β is given a low weight of 0.05 and $W = 1.5$ in our training. In addition to the shifted predictions from the Shift network B_S, we sample random locations around B_P, and use those to train the Refine model. This is part of the data augmentation process.

6. Experiments

The experiments were designed to assess the ability of the shift/refine pipeline to boost the processing speed of an object detector, while maintaining accuracy. We ran our experiments on two classes of objects: *faces* and *persons*.

Inference Pipeline: Our highly parallelized inference pipeline, shown in figure 6, incrementally propagates localized changes relative to the detections in the I-frame obtained from the baseline detector. Video decoding is a critical bottleneck, and is therefore run as a separate process. P-frames processing has referential dependencies on the preceding and succeeding I-frames, and are executed concurrently after the accumulation step. Process 2 runs accumulation of motion vectors and residuals on the P-frames, while process 1 runs the baseline detector on the I-frames. Process 3 runs the Shift and Refine networks on the P-frames and has a blocking dependency on both process 1 and process 2.

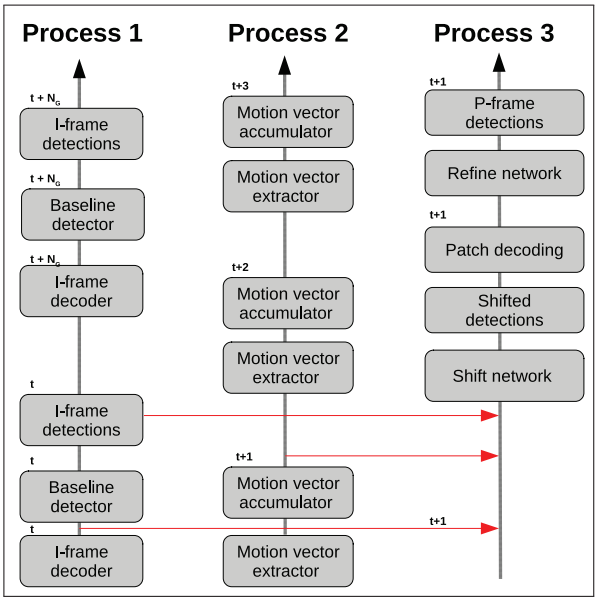

Figure 6: Overview of the multiprocess inference pipeline. Arrows indicate synchronization and blocking calls to retrieve data. Process 3 occasionally gets blocked on process 1 due to the slower baseline detector.

Detector	Training data (unlabeled)	HD Test data 1920×1080	4K Test data 3840×2160
Face(static)	156,592	36,344	7,576
Face(moving)	416,432	1,800	9,106
Person(static)	122,235	11,106	–
Person(moving)	315,962	3,141	–

Table 1: Datasets used for evaluating our framework. We experimented with both stationary and moving camera, and show the number of frames in each set and different resolutions. We used entire labeled data for testing.

Network Implementation Details For baseline detectors, we used a standard SSD[15] with resnet 50-layers feature extractor for the faces and YOLOv3[21] for the persons. We vary the GOP size $\mathbb{N}_G$ in our experiments to study its effect on speed and accuracy. The original video does not need to be encoded with a particular GOP size. Rather, when processing a video with a large GOP size, we reconstruct a source I-frame after the $\mathbb{N}_G$ P-frames, and run the motion vector accumulation relative to it. The Shift network is trained as a two-layer network with 256 channels. Inputs to the Shift network are variable sized bounding boxes, resized to a fixed patch size. The patch is normalized to preserve magnitude and orientation of the motion vectors,

Res.	512 × 512				1920 × 1080				3840 × 2160			
Detector (Batch)	Detect (*infer*)	Refine (*infer*)	Detect (*e2e*)	Refine (*e2e*)	Detect (*infer*)	Refine (*infer*)	Detect (*e2e*)	Refine (*e2e*)	Detect (*infer*)	Refine (*infer*)	Detect (*e2e*)	Refine (*e2e*)
Face(1)	56	54(1.0x)	56	60(1.0x)	12	52(4.1x)	12	41(3.4x)	3.3	52(15.3x)	3.4	20(6.2x)
Face(8)	99	70(0.7x)	65	56(0.8x)	13	68(5.0x)	13	42(3.2x)	3.4	68(19.7x)	3.4	22(6.2x)
Person(1)	35	90(2.5x)	37	88(2.4x)	16	78(5.1x)	15	64(4.3x)	4.4	52(11.9x)	4.3	28(6.5x)
Person(8)	54	130(2.4x)	56	87(1.5x)	17	122(7.3x)	17	63(3.8x)	4.7	66(14.3x)	4.6	29(6.3x)

Table 2: *e2e* = End-to-end, *infer* = inference. Speed gains (number in parenthesis) achieved by our framework relative to the baseline detector. We compare the face and person detector and refinement pipeline speed for *inference* and *end-to-end* processing. End-to-end processing includes video decoding in addition to neural network inference.

and its size is determined from the mean aspect ratio of the object. For faces the fixed input patch size for the Shift network is 24×16 while for persons, we used 44×16. Both forward and backward motion vectors are used in the same Shift network to make multiple predictions. The Refine network is single feature map with the backbone feature extractor as `Mobilenet0.25` [9]. Inputs to the Refine network are the fixed size cropped patches of the decoded P-frame. Decoding of localized patches happens in the GPU using `RoIAlign` to extract regions from the source I-frame, motion vector and residual components of the P-frame. Patches are all resized to fixed size (128×128) and processed in batches of 128 for training. The top most layer emits a 1×1 feature map of depth 256. The training in the Refine network used extensive data augmentation to overcome variations due to JPG artifacts, color changes, lighting and brightness changes. In addition, small translation perturbations were added in the input patches for robustness to noisy predictions from the Shift network. Both networks were trained independently using SGD with learning rate set to 0.001, learning rate decay to 0.1, and weight decay to 0.0005. We train the models for 100 epochs. We used a small validation set to determine optimal values for these parameters, and fixed them thereafter. Also, we used detection confidences as soft pseudo-labels for knowledge transfer. Both forward and backward detections from the Shift network are concatenated and refined by the Refine network. The final set of detections are obtained by applying NMS, that filters out the duplicate detections.

Evaluation Dataset: Our baseline detectors are fully convolutional (Single Shot Detector[15]) and have been trained at varying resolutions. In our experiments the detectors were used to process high resolution imagery with input resolution ranging from 512×512 to 3840×2160. For persons, we used a combination of both MOT17Det and MOT20Det [17] dataset that provided rich set of HD 1920×1080 videos with labeled persons for evaluation. For faces, we used WILDTRACK [3] dataset that included multiple HD resolution videos with labeled faces. For UltraHD videos, we labeled faces in various Youtube videos. A summary of total frames and videos used in our framework is listed in the table 1. For motion vector and residuals extraction we used ffmpeg[26][2]. We developed our custom implementation for running motion vector and residual accumulation on P-frames.

Algorithm Video	No Shift +Refine	Mean Shift	Learn Shift	Shift +Refine	Detect per frame
Static					
512	0.257	0.446	0.451	**0.592**	0.495
HD	0.374	0.640	0.658	**0.736**	0.714
4K	0.404	0.668	0.682	**0.789**	0.770
Moving					
512	0.125	0.363	0.369	**0.703**	0.650
HD	0.175	0.533	0.534	0.828	**0.902**
4K	0.154	0.512	0.523	0.767	**0.849**
All	0.397	0.670	0.683	**0.791**	0.772

Table 3: Comparison of face detection accuracy (AP) for different stages of the inference pipeline. *Mean shift* refers to shifting bounding boxes using average motion vector in the contextual patch. *Learn shift* column does not refine the detections using Refine network.

Speed and accuracy comparison: Table 2 shows the speed gains achieved using our framework for different video resolutions. In this experiment, we used the same detector to process the videos of different resolutions. Notice that the gain factors are dramatically higher for higher resolution videos, thus demonstrating the framework as a powerful enabling technology for processing high resolution videos. Detector accuracy is measured as area under the precision-recall curve or average precision(AP). We conducted several ablation studies to assess effects of various components of our inference pipeline. Table 3 shows the AP obtained by different stages of our processing pipeline on test videos of varying resolutions. The first column shows accuracy when no shift and refinement processing is performed. The bounding boxes stay in the same location as detected in the I-frames. The last column shows face detector accuracy when applied for every frame of the video. Notice that for stationary camera videos, temporal motion cues could cause shift and refinement to outperform the original detector's accuracy. This is not unusual for static cameras where

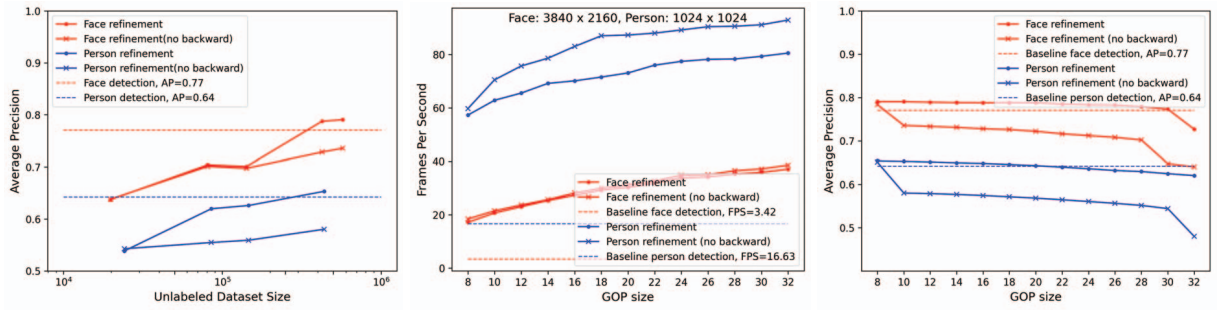

Figure 7: *(left)* Accuracy of the detectors improves as the number of unlabeled training examples increases. Backward motion cues are critical for achieving better accuracy compared to the baseline models. The observed non-monotonic increase in the accuracy of the face refinement plot is due to noisy labels from the baseline model. *(center)* Increasing the GOP size increases the processing speed for both detectors due to better parallelization, *(right)* although at the cost of reduced accuracy.

Algorithm Video	No Shift +Refine	Learn Shift	Shift +Refine (forward)	Shift +Refine (both)	Detect per frame
Static					
512	0.517	0.577	0.575	**0.645**	0.563
832	0.528	0.610	0.580	**0.654**	0.637
1K	0.534	0.582	0.581	**0.655**	0.642
Moving					
512	0.393	0.503	0.639	0.649	**0.657**
832	0.418	0.526	0.649	0.696	**0.720**
1K	0.415	0.493	0.650	0.703	**0.724**
All	0.526	0.573	0.580	**0.653**	0.642

Table 4: Comparison of person detection accuracy (AP) for different stages of our inference pipeline. We compare the effect of forward and backward motion cues on the inference. Fourth and fifth columns show the accuracy without and with backward motion cues respectively.

Detector Memory(Mb)	Refine Memory(Mb)	Detector Power(Watts)	Refine Power(Watts)
1065.0	81.0	191.7	92.0

Table 5: Comparison of per frame GPU memory and power usage for inference (batch size 1) with baseline detector vs. refinement network.

tracking is more accurate than occasionally inconsistent, frame-by-frame object detection when used to process high resolution imagery. We observed somewhat poor accuracy for high resolution moving camera videos for faces. This is primarily attributed to lot of small faces, at far distance, undergoing significant shift due to rotational motion of the camera. This issue is less evident for persons as shown in table 4. For persons, we only show accuracy of the inference pipeline when the I-frames are processed at 512, 832 and 1024 resolutions. Figure 7*(left)* illustrates a key result and contribution of our framework. We conducted end-to-end experiments for training shift and refinement models for the face and person detector with increasing amounts of unlabeled data. As the plots show there is a clear trend of improving accuracy when more unlabeled data is used in the teacher-student training framework. The dotted plot shows per-frame detection accuracy of the baseline face and person detector. The GOP size is a critical parameter in our framework and determines the tradeoff between speed and accuracy. Large GOP size allows greater parallelization at the cost of accuracy. Figure 7*(center)* and *(right)* illustrates how the performance varies with increasing GOP size for face and person detector. Dotted plots denote the performances of the baseline detector. The decrease in AP for large GOP size is due to the need for better models that could handle larger magnitude of motion flows from farther I-frames.

7. Conclusion and Future Work

This work demonstrates a novel approach for increasing processing speeds of object detection models for high resolution videos, without sacrificing accuracy, using a generic framework of Shift and Refine networks. The work employs motion cues encoded in the compressed videos to interpolate detections between sparsely processed I-frames. Speed gains are achieved by processing localized patches as opposed to the entire frame, and parallel processing of the P-frames after removing their sequential dependencies and making them reference only the nearest I-frames. The framework has widespread applicability due to lack of the need for labeled data for training the Shift and Refine networks. They are trained using unlabeled, representative videos, using a novel teacher-student based knowledge transfer learning. Finally, we demonstrated that the framework works with any generic object detector. Future work includes narrowing down the accuracy gap for moving camera videos containing fast motions by enhancing the accuracy of the Shift network.

References

[1] Saeed Ranjbar Alvar and Ivan V Bajić. MV-YOLO: Motion vector-aided tracking by semantic object detection. In *2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–5. IEEE, 2018. 3

[2] L. Bommes, X. Lin, and J. Zhou. Mvmed: Fast multi-object tracking in the compressed domain. pages 1419–1424, 2020. https://github.com/LukasBommes/mv-extractor. 4, 7

[3] Tatjana Chavdarova, Pierre Baqué, Andrii Maksai, Stéphane Bouquet, Cijo Jose, Louis Lettry, Francois Fleuret, Pascal Fua, and Luc Van Gool. Wildtrack: A multi-camera hd dataset for dense unscripted pedestrian detection. IEEE Conference on Computer Vision and Pattern Recognition, pages 5030–5039, New York, 2018. IEEE. 7

[4] Kai Chen, Jiaqi Wang, Shuo Yang, Xingcheng Zhang, Yuanjun Xiong, Chen Change Loy, and Dahua Lin. Optimizing video object detection via a scale-time lattice, 2018. 3

[5] Yihong Chen, Yue Cao, Han Hu, and Liwei Wang. Memory enhanced global-local aggregation for video object detection, 2020. 3

[6] Yiming Cui. Dfa: Dynamic feature aggregation for efficient video object detection, 2022. 3

[7] Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, mar 2021. 4

[8] Fei He, Naiyu Gao, Jian Jia, Xin Zhao, and Kaiqi Huang. QueryProp: Object query propagation for high-performance video object detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(1):834–842, jun 2022. 3

[9] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. 7

[10] Lianghua Huang, Yu Liu, Bin Wang, Pan Pan, Yinghui Xu, and Rong Jin. Self-supervised video representation learning by context and motion decoupling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13886–13895, 2021. 3

[11] Lingtong Kong, Chunhua Shen, and Jie Yang. Fastflownet: A lightweight network for fast optical flow estimation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021. 3, 5

[12] Athindran Ramesh Kumar, Balaraman Ravindran, and Anand Raghunathan. Pack and detect: Fast object detection in videos using region-of-interest packing. *CoRR*, abs/1809.01701, 2018. 3

[13] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8971–8980, 2018. 5

[14] Qiankun Liu, Bin Liu, Yue Wu, Weihai Li, and Nenghai Yu. Real-time online multi-object tracking in compressed domain, 2022. 3

[15] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot MultiBox detector. pages 21–37, 2016. 1, 6, 7

[16] Hao Luo, Wenxuan Xie, Xinggang Wang, and Wenjun Zeng. Detect or track: Towards cost-effective video object detection/tracking, 2018. 3

[17] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. MOT16: A benchmark for multi-object tracking. *arXiv:1603.00831 [cs]*, Mar. 2016. arXiv: 1603.00831. 7

[18] Leandro Moreira. Digital video introduction, 2016. https://github.com/leandromoreira. 4

[19] Mathias Parger, Chengcheng Tang, Thomas Neff, Christopher D. Twigg, Cem Keskin, Robert Wang, and Markus Steinberger. Motiondeltacnn: Sparse cnn inference of frame differences in moving camera videos, 2022. 3

[20] Mathias Parger, Chengcheng Tang, Christopher D Twigg, Cem Keskin, Robert Wang, and Markus Steinberger. Deltacnn: End-to-end cnn inference of sparse frame differences in videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12497–12506, 2022. 3

[21] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018. 6

[22] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. 4

[23] Vit Ruzicka and Franz Franchetti. Fast and accurate object detection in high resolution 4k and 8k video using GPUs. In *2018 IEEE High Performance extreme Computing Conference (HPEC)*. IEEE, sep 2018. 1, 4

[24] Zheng Shou, Xudong Lin, Yannis Kalantidis, Laura Sevilla-Lara, Marcus Rohrbach, Shih-Fu Chang, and Zhicheng Yan. Dmc-net: Generating discriminative motion cues for fast compressed video action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 3

[25] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'14, page 568–576, Cambridge, MA, USA, 2014. MIT Press. 3

[26] Suramya Tomar. Converting video formats with ffmpeg. *Linux Journal*, 2006(146):10, 2006. 4, 7

[27] Julian True and Naimul Khan. Motion vector extrapolation for video object detection. *arXiv preprint arXiv:2104.08918*, 2021. 3

[28] Takayuki Ujiie, Masayuki Hiromoto, and Takashi Sato. Interpolation-based object detection using motion vectors for embedded real-time tracking systems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 616–624, 2018. 3

[29] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Val Gool. Temporal segment net-

works: Towards good practices for deep action recognition. In *ECCV*, 2016. 3

[30] Shiyao Wang, Hongchao Lu, and Zhidong Deng. Fast object detection in compressed video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7104–7113, 2019. 3

[31] Shiyao Wang, Yucong Zhou, Junjie Yan, and Zhidong Deng. Fully motion-aware network for video object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 542–557, 2018. 3

[32] Xinggang Wang, Zhaojin Huang, Bencheng Liao, Lichao Huang, Yongchao Gong, and Chang Huang. Real-time and accurate object detection in compressed video by long short-term feature aggregation. *Computer Vision and Image Understanding*, 206:103188, 2021. 3

[33] Chao-Yuan Wu, Manzil Zaheer, Hexiang Hu, R. Manmatha, Alexander J. Smola, and Philipp Krähenbühl. Compressed video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 3, 4

[34] Takanori Yokoyama, Toshiki Iwasaki, and Toshinori Watanabe. Motion vector based moving object detection and tracking in the mpeg compressed domain. In *2009 Seventh International Workshop on Content-Based Multimedia Indexing*, pages 201–206. IEEE, 2009. 3

[35] Bowen Zhang, Limin Wang, Zhe Wang, Yu Qiao, and Hanli Wang. Real-time action recognition with enhanced motion vector cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2718–2726, 2016. 3

[36] Xizhou Zhu, Jifeng Dai, Lu Yuan, and Yichen Wei. Towards high performance video object detection, 2017. 3