Region-based Content Enhancement for Efficient Video Analytics at the Edge

Weijun Wang, *Institute for AI Industry Research (AIR), Tsinghua University;* Liang Mi, Shaowei Cen, and Haipeng Dai, *State Key Laboratory for Novel Software Technology, Nanjing University;* Yuanchun Li, *Institute for AI Industry Research (AIR), Tsinghua University;* Xiaoming Fu, *University of Göttingen;* Yunxin Liu, *Institute for AI Industry Research (AIR), Tsinghua University*

https://www.usenix.org/conference/nsdi25/presentation/wang-weijun

This paper is included in the Proceedings of the 22nd USENIX Symposium on Networked Systems Design and Implementation.

April 28–30, 2025 • Philadelphia, PA, USA

978-1-939133-46-5

Open access to the Proceedings of the 22nd USENIX Symposium on Networked Systems Design and Implementation is sponsored by

جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

Region-based Content Enhancement for Efficient Video Analytics at the Edge

Weijun Wang[1*] Liang Mi[2*†] Shaowei Cen[2†] Haipeng Dai[2] Yuanchun Li[1] Xiaoming Fu[3] Yunxin Liu[1‡]
[1]*Institute for AI Industry Research (AIR), Tsinghua University*
[2]*State Key Laboratory for Novel Software Technology, Nanjing University*
[3]*University of Göttingen*

Abstract

Video analytics is widespread in various applications serving our society. Recent advances of content enhancement in video analytics offer significant benefits for the bandwidth saving and accuracy improvement. However, existing content-enhanced video analytics systems are excessively computationally expensive and provide extremely low throughput. In this paper, we present region-based content enhancement, that enhances only the important regions in videos, to improve analytical accuracy. Our system, RegenHance, enables high-accuracy and high-throughput video analytics at the edge by 1) a macroblock-based region importance predictor that identifies the important regions fast and precisely, 2) a region-aware enhancer that stitches sparsely distributed regions into dense tensors and enhances them efficiently, and 3) a profile-based execution planer that allocates appropriate resources for enhancement and analytics components. We prototype RegenHance on five heterogeneous edge devices. Experiments on two analytical tasks reveal that region-based enhancement improves the overall accuracy of 10-19% and achieves 2-3× throughput compared to the state-of-the-art frame-based enhancement methods.

1 Introduction

Video cameras are widespread in our society, with numerous installations in major cities and organizations [3, 4, 12, 13, 16]. These cameras continuously collect vast amounts of video data for various applications such as traffic control [25], school security [42], crime investigation [15, 83], sports refereeing [5, 14], and more [17, 86]. Significant advances in deep neural networks (DNNs) for vision processing offer a tremendous opportunity to AI-powered automatic *video analytics* [44,45,56,89,92,101]. A typical video analytics pipeline follows the distributed paradigm as AI methods often require high computing power that cameras cannot offer [1, 18]. Cameras capture and deliver the live videos to the edge in proximity for real-time processing. However, the outdated hardware of cameras in use [62] and extremely limited uplink bandwidth between the camera and the edge [21, 43, 101] results in limited video quality and thus low analytical accuracy.

Content enhancement offers a promising solution to tackle this issue. It delivers a remarkable accuracy improvement [68,70,80,82,84,96,100] and bandwidth saving [41,60,79,93–95,97] by leveraging neural enhancement models (*e.g.*, super-resolution [24, 64], generative adversarial network [30, 48], image restoration [63]) to enhance the informative details of video frames prior to feeding them into the final analytical models. Different from previous model optimization methods (*e.g.*, model merging [55, 71], model updating [60, 74], and model switching [59, 95]), data-optimization content enhancement does not require modifying user-provide models [23, 31, 73]. Besides, even if the city government updates current low-end cameras to the latest ones that can offer high-quality video, content enhancement can still improve the details of small objects or blurred content.

Unfortunately, naively employing content enhancement in practice is excessively computationally expensive. Such straightforward ways of enhancements not only cause high latency but also compete for computational resources with final analytical models. For example, applying enhancement on one tail-accuracy frame or executing generative adversarial networks on hard-recognized human faces causes hundreds to thousands of milliseconds of latency [84, 96]. The state-of-the-art method, *selective enhancement* [93, 95], yields some throughput improvement by enhancing only sampled frames but results in substantial accuracy reduction (more in § 2.2).

Goal and observations. This paper asks the following research question: Can we integrate content enhancement into video analytics at the edge in a high-throughput manner? We argue that yes, but it requires a rethink of enhancement. Our key observations are that 1) the time cost of enhancement positively correlates to the input size, and 2) the regions after enhancement that benefit analytical accuracy only occupy a small portion in each frame (more details in §2.3). Therefore, we aim to develop a *region-based content-enhancement* method which only enhances the beneficial regions.

To this goal, we face three non-trivial challenges.

(C1) How to identify beneficial regions fast and accurately? Simply relying on DNN-based identification is too slow to the extent that it even exceeds the time required for the entire frame enhancement.

(C2) How to efficiently enhance regions? Characteristics of enhancement and heterogeneous accuracy gain of regions necessitate a novel enhancement mechanism that differs from the previous frame-based enhancement [60, 93–95].

*Weijun Wang and Liang Mi contributed equally to this work.
†This work was done while Liang Mi and Shaowei Cen were interns at the Institute for AI Industry Research (AIR), Tsinghua University.
‡Corresponding author: Yunxin Liu<liuyunxin@air.tsinghua.edu.cn>.

(C3) How to allocate resources among components? To maximize the overall performance, limited resources on edge must be best allocated among system components; however, typical schedulers cause imbalances among components, which in turn leads to noticeable throughput degradation.

RegenHance. This paper presents RegenHance, a system that efficiently identifies and enhances the most beneficial regions in video, enabling high-accuracy and high-throughput analytics at the edge. RegenHance addresses the above challenges by following techniques:

Macroblock-based region importance prediction. We propose a predictor to fast and accurately identify the regions that yield the highest accuracy improvements in videos (§3.2). We first analyze the advantages of predicting the region importance at the macroblock (MB, the video encoding unit) level and establish a metric to precisely measure the accuracy gain (importance) of each MB after enhancement in each frame. Next, we develop an ultra-lightweight prediction model and a prediction results reuse algorithm, achieving high throughput. Experiments show our method can predict the MB importance in 30 frames per second on a single CPU thread.

Region-aware enhancement. We design an enhancer that efficiently enhances the Tetris-like irregular regions composed of important macroblocks (§3.3). It prioritizes and selects the Top-K MBs of all video streams in order of their importance scores (K is estimated by §3.4). Then, considering the unique characteristic of the time cost of enhancement (§2.3) and the sparse distribution of important MBs (§3.2), the region-aware enhancer can be formulated as a two-dimensional bin-packing problem that minimizes the input size of the enhancement model to optimize the throughput. We propose a greedy algorithm that fast stitches the irregular regions into dense tensors before forwarding them into the enhancement model.

Profile-based execution planning. To balance the limited resources among components on the edge device, we propose an execution planer (§3.4), that profiles the capacity of the given edge device, and allocates resources among components by determining the parameters (*e.g.*, batch size of the input) for each component (*e.g.*, decoder, MB-based region importance prediction, region-aware enhancer, and analytical models) to maximize the end-to-end throughput.

We summarize our key contributions as follows:

- To the best of our knowledge, we are the first to identify the bottleneck of content-enhanced video analytics and propose a new idea: region-based content enhancement.
- We prototype RegenHance that enables region-based content enhancement video analytics at the edge. It involves three components, MB-level region importance prediction, region-aware enhancement, and profile-based execution planning.
- We implement RegenHance and conduct evaluations for two popular analytical tasks with real-world videos on five heterogeneous devices. Experimental results show that RegenHance improves 10-19% accuracy and achieves 2-3× throughput compared to the state-of-the-art methods. Our code is now available at https://github.com/mi150/RegenHance.

This work does not raise any ethical issues.

2 Motivation and Challenges

This section explores three questions: (1) How well are selective enhancement techniques in video analytics (§2.2)? (2) How much potential improvement can region-based content enhancement achieve (§2.3)? (3) What challenges must be tackled (§2.4)?

2.1 Background

Typical video analytics platforms [6, 9, 39], to support real-time applications, provide services on edge servers close to cameras that allow users to register custom video analytics jobs. To register a job, users specify video sources, analytical tasks, performance targets, and optionally upload their analytical models [23, 31, 73]. The camera captures and encodes multiple images into a video stream (*e.g.*, 30 images into a 1-second video chunk set up in prior studies [44, 62, 101]), and then transmits to the edge for analytics. Following the same scope, this paper also concentrates on optimizing the analytics platforms on edge servers.

Content-enhanced video analytics [41, 60, 68, 85, 93–97, 100] is a mainstream data optimization method, that utilizes DNN models to enhance low-resolution frames to high-definition ones. Pre-trained DNNs learn a pixel generator mapping neighboring pixels' values to the generated one from training data, then deployed in runtime systems analyzing real-world videos for accuracy improvement and bandwidth saving.

Selective enhancement is the state-of-the-art method to improve the throughput of content-enhanced Internet video streaming [93, 95, 104]. In this method, a few sampled frames (called anchors) are enhanced by the super-resolution model, while the remaining frames are quickly up-scaled by reusing enhanced ones via codec information. Such reuse causes video quality loss, due to the rate-distortion problem in codec [78], accumulated across consecutive non-enhanced frames, and thus the sampled frames need to be selected carefully.

2.2 Limitation of Frame-based Enhancement

Motivational study. To measure the performance of selective enhancement, we benchmark three analytics methods. Taking object detection as an example, 1) *Only infer* method purely detects objects in each frame, 2) *Per-frame SR* method applies super-resolution on each frame and then detects objects on them (as the ground truth). 3) *Selective SR* [95] applies super-resolution on a set of selected frames and reuses on other ones, then detects objects on all frames; this method will select enough frames until meeting preset target accuracy (*e.g.*, 90%). For fairness, all methods run on an edge

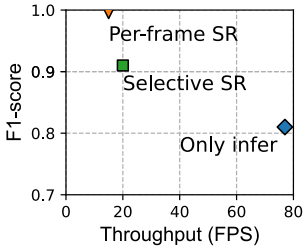


Figure 1: Performance of the frame-based methods. The state-of-the-art selective SR provides poor E2E throughput.

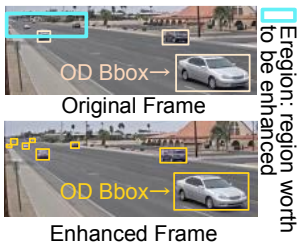


Figure 2: An example eregion worth to be enhanced for object detection (OD) bounding with a rectangle box.

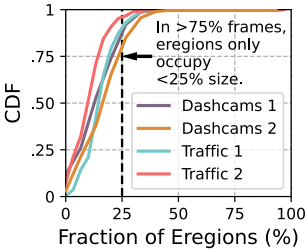


Figure 3: Distribution of eregions in various videos. In large amounts of frames, eregions occupy only a small portion.

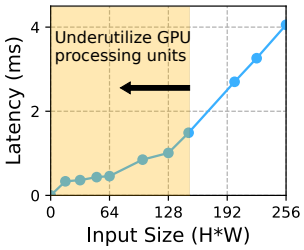


Figure 4: Latency of enhancement. The same H*W (*e.g.*, 64*64) input, no matter pixel values, yields the same latency.

device equipped with an NVIDIA T4 GPU [10] that upscales a 640 × 360 video to a 1920 × 1080 version via EDSR model [64] and detects objects with YOLO [46]. Both models compile with TensorRT [11].

Result: selective enhancement is too heavy for video analytics. As shown in Fig. 1, per-frame SR improves >10% accuracy but reduces >76% end-to-end throughput than only infer. Normalized with per-frame SR, selective SR indeed improves throughput from 15 to 20 (>33%) fps, but still far from the throughput of only infer and results in an undeniable accuracy drop. This is because, compared to human perception, analytical models are more sensitive to blur and distortion caused by accumulated loss of reuse. Small changes in several pixel values may flip the analytics result inversely [29]. So when presetting the accuracy target as 90%, selective SR methods choose 27-61 anchor frames in each 120-frame chunk on average. Such 24-51% fraction is much higher than that select and enhance only 2-13% frames in video applications for human vision.

2.3 Potential Improvement of Region-based Content Enhancement

Above *frame-based* optimizations fail to offer efficient analytics because they evenly enhance every pixel in each frame. We argue that this is unnecessary and causes a significant waste of computational resources.

Our first observation is that *the regions that are worth being enhanced in each frame occupy only a small portion.* In this paper, we name the region that provides higher analytical accuracy after enhancement, *Eregion*. As visualized in Fig. 2, the frame after enhancement (bottom) can provide more detected objects, and the eregion (blue rectangle) is a simple region (discussed more in Appx. C.1). Fig. 3 analyzes the distribution of eregions in experimental videos for Fig. 1. In >75% of frames over videos recording various scenarios, eregions for object detection task only occupy 10-25% of the spatial area of a frame; while for semantic segmentation, as illustrated in Appx. C.1, only 10-15% area in 70% frames is eregions. With careful design, eregions could minimize the enhanced area while offering comparable accuracy to per-frame SR. So ideally, only eregions should be enhanced in video analytics.

The second observation is that *the time cost of enhancement DNNs is positively correlated with input size.* As shown in Fig. 4, along with the enlarging input size, the enhancement latency initially experiences a gradual rise and then scales proportionally with the input size after making full use of processing units. This unique characteristic originates from the nature of enhancement models that generate new pixels from neighboring ones with the learned mapping. Thus, we must minimize the size to decrease enhancement latency.

Our idea. With the above two observations in mind, we propose *region-based content enhancement* that aims to only enhance the eregions to maximize the throughput of content enhancement while providing comparable accuracy to per-frame enhancement. The question next is how to fully explore the potential room for improvement.

2.4 Challenges to Achieve Region-based Content Enhancement

Building an effective system based on region-based content enhancement involves three challenges.

C1: How to fast and accurately identify eregions on original frames? Enhancing only the eregions, as shown in Fig. 5, does save significant (2.4×) time cost. Unfortunately, such an oracle cannot be reached as eregions are calculated from the already-enhanced frames. During real-time analytics, only the original frames are accessible, thus we have to predict eregions on the original frames. Naively using a DNN-based method to identify regions, *e.g.*, DDS [44] identifies the Region of Interest (RoI) with a Region Proposal Network (RPN), can not meet our requirements. The imprecise identification spends too much time on the enhancement of unimportant regions in Fig. 5, and the high computing cost of the selection methods themselves (*e.g.*, the RPN), in turn, harms the possible time saving of region-based enhancement.

C2: How to enhance eregions in high throughput? High-throughput region-based enhancement is hardly achieved by existing frame-based enhancers [84, 93–97]. Three new contradictions that frame-based methods have never encountered

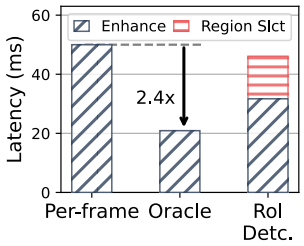
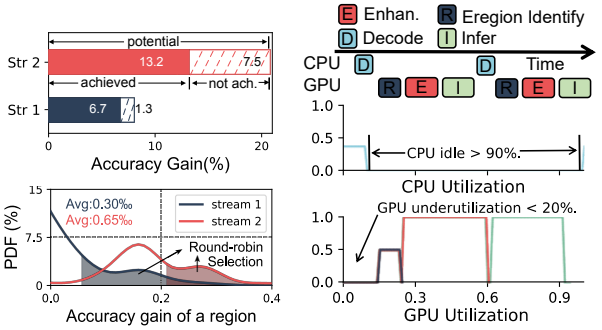


Figure 5: Region-based enhancement saves remarkable (2.4×) latency, but prior region selection methods themselves (*e.g.*, RoI detection [44]) produces too high computing cost.



(a) Round robin across streams (b) Sequential exec. of components

Figure 6: Limitation of region-agnostic resource management.

obstruct region-based enhancement. 1) Eregions are often in irregular size and sparsely distributed in each frame, but the enhancement models (DNNs) only accept rectangle (matrix) input. 2) One natural option, as DDS [44] used, is setting non-eregion pixels to black (zero value in the matrix); and yet it leads to the same latency as enhancing the original frame due to the pixel-value-agnostic characteristic of enhancement latency as demonstrated in Fig. 4. 3) Batch execution of enhancement DNNs benefits higher throughput [76], but requires every input matrix in the same size.

C3: How to allocate resources among components? To optimize the overall performance (accuracy and throughput), particularly for all video streams, 1) eregions must be carefully allocated across streams, and 2) computational resources must be balanced among computing components. However, this is challenging due to the heterogeneity of regions and components. First, the accuracy gains and enhancement overhead of regions are heterogeneous. Second, all runtime components on the edge server, including decoding, eregion identification, enhancement, and analytical inference, compete for computational resources, and their throughputs are heterogeneous. Using previous schedulers [8, 95] causes noticeable degradation in accuracy and throughput.

To demonstrate this, we benchmark a strawman scheduler that 1) parallelizes per-stream decoding on multiple CPU threads, 2) forwards decoded frames to GPU in a round-robin manner, 3) applies RegenHance's region enhancement strategy but pipelines computing components at batch size of four. Two streams that contain different sizes of eregions are delivered to an edge server equipped with an NVIDIA T4 GPU [10]. We measure the accuracy gain per stream and the resource utilization of computing units.

This strawman region-agnostic scheduler fails to achieve high accuracy gain and throughput. Fig. 6(a) (top) shows the potential (the accuracy of per-frame SR minus only infer), achieved (solid part), and not achieved (dotted part) accuracy gain of two streams. There is a remarkable gain (7.5%) not achieved in Stream 2 as the round-robin manner results in an even chance for enhancement across streams. The distributions of the region's accuracy gain in each stream are highly heterogeneous as the two curves shown in Fig. 6(a) (bottom). Suppose a scheduler enables Stream 2 to enhance more regions that provide higher gain (namely, in Fig. 6(a) (bottom), decrease some grey area under Stream 1 to fill the blank under Stream 2 into red), it can improve the overall accuracy. On the other hand, inappropriate execution plans underutilize the hardware, leading to low end-to-end throughput and, in turn, hindering higher accuracy improvement. As illustrated in Fig. 6(b), the region-agnostic scheduler leaves >90% CPU and >15% GPU idle time. The unsuitable batch size results in the Eregion Identification utilizing only <50% computing resource GPU, hence blocking the subsequent enhancement and analytics.

3 RegenHance Design

RegenHance aims to enhance only the regions (Eregions) that best benefit analytical accuracy, so as to maximize the overall accuracy gain among all streams in the optimal end-to-end throughput on a given edge server. To do so, it addresses the above challenges by three techniques: 1) a lightweight but precise algorithm to identify eregions on macroblock (MB) granularity, 2) a high-throughput enhancer that together considers the heterogeneous accuracy gain across streams and unique characteristics of enhancement, and 3) a holistic resource manager generating execution plan for all components on given edge servers. Fig. 7 depicts the design of RegenHance. We implement the techniques with three new components: MB-based region Importance Prediction, Region-aware Enhancement, and Profile-based Execution Planning.

3.1 Overview Workflow

RegenHance takes the original ingest videos and outputs high-resolution frames for downstream analytical tasks. During the offline phase, Profile-based Execution Planning ❶ profiles the budget of processors on the given edge server with the registered analytical tasks and optional uploaded corresponding models by users; next, ❷ generates the execution plan, *i.e.*, specifies every runtime component's execution hardware and their allocated resource meeting best pipelining and parallelization, to maximize the end-to-end throughput under the

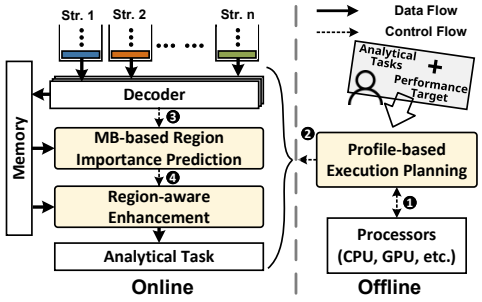


Figure 7: RegenHance overview.

constraints of performance targets (*e.g.*, accuracy and latency) that users specify (§3.4).

During the online phase, the decoder decodes compressed video streams in parallel to RGB frames and stores them in memory. MB-based region importance prediction ❸ selects frames during decoding, loads them from memory, predicts their MBs importance, and appends importance into MBs' indexes (§3.2). Region-aware enhancement ❹ aggregates and sorts all MBs over all streams based on their importance, constructs top N ones into regions, stitches them into dense tensors, and efficiently executes enhancement (§3.3). Lastly, forward the enhanced frames to analytical models for inference.

3.2 MB-based Region Importance Prediction

In order to identify eregions quickly and accurately, we propose an MB-based region importance prediction. It contains a lightweight predictor estimating MB importance in each frame (in spatial dimension) and an MB importance reuse algorithm among continuous frames (in temporal dimension).

3.2.1 Spatial MB Importance Predictor in Each Frame. Eregions can be arbitrary shapes, we need to determine the granularity to construct these regions first. One natural way is pixel granularity. The pixel-grain method, of course, provides the most precise importance prediction but costs too much computing resources (as demonstrated in Fig. 5).

Inspired by the video codec knowledge [45, 53], we argue that setting macroblock (MB) as the elementary unit of eregions is both efficient and accurate. Macroblock serves as the elementary unit for applying the quantization parameter (QP) to control the compression level of video quality. For example, in H.264 [7], frames are divided into an array of 16×16-pixel MBs, and each MB is assigned with different QPs to distribute more bits to regions demanding higher visual quality and fewer bits to less crucial regions.

After setting MB as a unit, our problem transformed to: Given a video frame f containing a set of macroblocks MB, select partial MBs MB_s to be enhanced such that the accuracy of downstream analytical task is maximized:

$$max \quad Acc(I(SR(MB_s)+IN(\overline{MB_s})), I(SR(f))),$$

where $Acc(\cdot)$ is the accuracy of analytical task $I(\cdot)$ (*e.g.*, F1-score of object detection), $SR(\cdot)$ and $IN(\cdot)$ are the super-resolution and the bi-linear interpolation with the same enlarge factor, $\overline{MB_s}$ indicates the unselected MBs.

To select beneficial MBs, we need to quantify their importance. Ideally, MBs after enhancement leading to larger variations in inference results and greater differences in pixel values are more important. With this intuition, we utilize the following "importance" metric looking at the gradient of the accuracy with respect to changes in the pixels in each MB, and the magnitude of change in those pixel values due to enhancement.

$$\sum_{i \in MB} || \underbrace{\frac{\partial Acc(I(IN(f)), I(SR(f)))}{\partial IN(f)}\Big|_i}_{\text{accuracy's gradients at pixel } i} ||_1 \cdot || \underbrace{(SR(f)_i - IN(f)_i)}_{\text{pixel value distance at } i} ||_1,$$

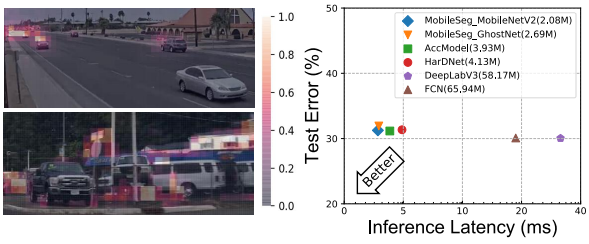
where i is a pixel within the certain MB and $||\cdot||_1$ is the L1-norm. This metric assigns an importance score to each MB. Fig. 8(a) shows the heat map of MB importance on the same frame with Fig. 2. Their similarity demonstrates the MB importance is a good representation of eregions*.

It seems straightforward to calculate the MB importance according to the above importance metric. Unfortunately, it needs the frames already enhanced. Such a chicken-egg paradox makes us predict the MB importance in the original frames with a learning-based method. We construct the training set by enhancing all video frames and calculating the importance metrics on each of them with one forward and backward propagation of the final analytical model. This resulting importance value of each MB is one item in the ground truth matrix of each frame, Mask*.

Next, we consider this problem a segmentation task and draw inspiration from their model designs. Our key observation is that the MB importance prediction resembles the problem of image segmentation. Image segmentation aims to semantically segment an image by assigning each pixel one predefined label, while MB importance prediction gives each MB an importance score. With this in mind, the MB importance prediction can be approximated as an image segmentation problem by boiling the importance value down to multiple importance levels. Suppose setting ten levels in this paper, MB importance prediction targets to assign each MB an importance level, like classification in image segmentation. Appx. B demonstrates such approximation yields good performance. This observation enables us to harness various techniques tailored for MB importance prediction, including learning-based semantic segmentation.

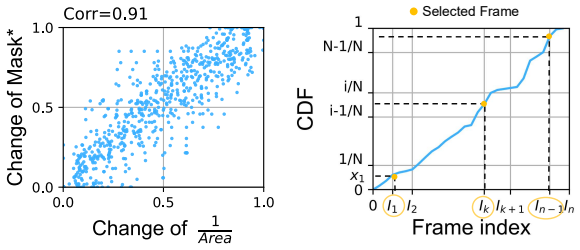
Aiming to predict MB importance precisely at high throughput, we train an ultra-lightweight segmentation model with the above importance metric. We retrained six models using

*Unlike the saliency map in computer vision that captures which pixel values have more influence on the DNN output, our loss function captures how the enhancement or non-enhancement of an MB (its content quality change) specifically changes the DNN inference accuracy.



(a) Heat map of MB importance (b) Model selection

Figure 8: Macroblock-level region importance prediction.



(a) Correlation ratio (b) Frame selection with CDF

Figure 9: Temporal MB importance reuse across frames.

the cross-entropy loss with piecewise Mask* (as importance level) to support MB importance prediction. Six models are an ultra-lightweight model MobileSeg [81] with two backbones, two lightweight models AccModel [45] and HarDNet [32], and two heavyweight models FCN [66] and DeepLabV3 [33]. As shown in Fig. 8(b), ultra-lightweight models provide almost the same accuracy as heavyweight ones while offering 4-18× throughput. This can be attributed to the significantly relaxed complexity of the MB-grained segmentation compared to traditional image segmentation. Predefined MB size in video codec supports this point. The 16×16-MB H.264 [7] codec makes 1920×1080 labels that output in traditional image segmentation models decrease to 120×68 ones in MB-grained. As the best performance of MobileSeg, we select it as our MB importance predictor. At the offline phase, RegenHance fine-tunes the predictor with the Mask* generated by user-uploaded analytical models.

3.2.2 Temporal MB Importance Reuse among Continuous Frames. Reusing DNN outputs on similar frames is popular [45, 90, 100] as discussed in §5. Reusing content of enhanced frames in region-based enhancement like selective SR will cause dramatic accuracy loss (demonstrated in §2.2), but the MB importance value is reusable. Hence, RegenHance predicts the MB importance of a set of frames and reuses their outputs on other frames, to offer the best approximation of per-frame MB importance prediction.

Our key choice is using an ultra-lightweight operator to represent the MB importance change then only predicts MB importance of the frames with large changes. We compared many lightweight features and proposed $\frac{1}{Area}$ operator (details in Appx. C.2). Area operator captures the large blocks in images [52, 62], whereas $\frac{1}{Area}$ captures the change of small objects just as the important MBs in Fig. 8(a) needed. Statistical analysis in Fig. 9(a) shows it is a nice representation to estimate the change of Mask* (with 0.91 correlation).

With the $\frac{1}{Area}$ operator, denoted by Φ, RegenHance selects frames to be predicted by the cumulative distribution function (CDF) of $\Delta\Phi$ between continuous frames in each chunk. It first accumulates the feature change with the following function during decoding a n-frame chunk.

$$\mathbf{S} = Norm(\Delta\Phi(Res_{Y_1}), \cdots, \Delta\Phi(Res_{Y_{n-1}})),$$

where Res_{Y_i} is Y-channel of each frame's residual[†], $\Delta\Phi(Res_{Y_i}) = \Phi(Res_{Y_{i+1}}) - \Phi(Res_{Y_i})$, $Norm(\cdot)$ is L1-normalization. Then, it selects N frames based on the CDF $\mathbf{M}$, where $\sum_i M_i = 1$, calculated from $\mathbf{S}$. As illustrated in Fig. 9(b), the y-axis is divided into N even intervals; and in each interval, it selects a value, *e.g.*, x_1, and then its corresponding frame index, *e.g.*, I_1, on the x-axis is a selected frame. The other frames in each interval reuse the predicted result of this frame. In multiple streams, the number of selected frames for given stream j is allocated by the ratio $\frac{\sum_i \Delta\Phi_{i,j}}{\sum_j \sum_i \Delta\Phi_{i,j}}$, and its total number is determined by the profile-based execution planning (§3.4). Fig 10 depicts the MB-based importance prediction first selects frames and then predicts their MB importance.

3.2.3 Discussion. *Generality of importance metric.* In this paper, each analytical task must retrain a specific MobileSeg for importance prediction, as the metric equation relies on the downstream models. We omit a general importance metric design in this paper because 1) the current specific metric offers good results (as shown in Fig. 8(a)) and 2) the fine-tuning time only costs 4 minutes on eight RTX3090 GPUs. We will explore a general metric in future work.

3.3 Region-aware Enhancement

To achieve optimal overall accuracy over all streams and end-to-end throughput, we propose a region-aware enhancement consisting of a cross-stream MB selection and a region-aware bin packing algorithm.

3.3.1 Cross-stream MB Selection. To maximize the overall accuracy improvement, RegenHance chooses the best MB st among all streams that offer the highest total accuracy gain. As shown in Fig. 10, it constructs a global queue that aggregates and sorts MBs from all streams in order of the importance (level) in the MB index $\{stream_{id}, frame_{id}, loc_x, loc_y, importance\}$, where loc_x, loc_y indicates coordination of the MB in the frame.

Next, it selects the top N MBs and delivers their indexes to the region-aware bin packing algorithm for further processing. The number of selected MBs is estimated as follows,

$$\max_N MB_{size} \cdot N \leq H \cdot W \cdot B,$$

[†]Residual is data existing in video codec representing the difference data between the original and compressed video frames.

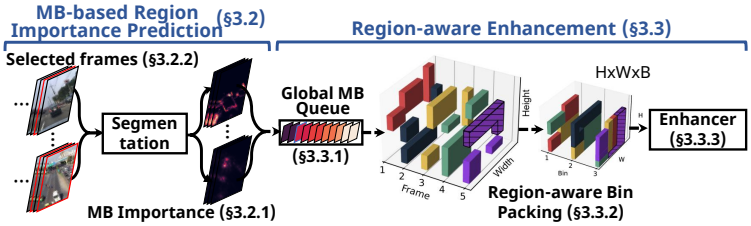


Figure 10: Runtime components of RegenHance.

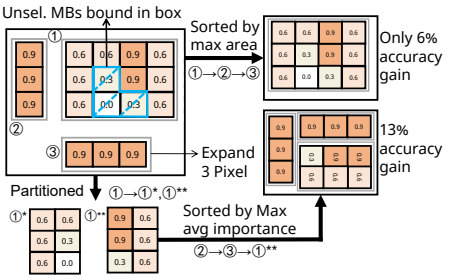


Figure 11: Large region is not always desirable. Classic large-item-first policy packs the MBs that are unselected but bounded in the large box into the bin (upper half), leading to a 7% accuracy drop compared to our importance-first policy (lower half).

Algorithm 1 Region-aware Bin Packing

Input: Indexes of MBs, B Bins of $H \times W$ size
Output: Packing plan of MBs

```
1:  function PACKING(MBs, Bins)
2:      freeareas = Bins                    ▷ Initialize the list of free area as Bins
3:      regions = REGIONPROPS(MBs)
4:      boxes = BOUND(regions)
5:      boxes = PARTITION(boxes)            ▷ Partition big boxes to small ones
6:      SORT(boxes, reverse = True, order = Σ_{MB∈box} MB.importance / |{MB|MB∈box}|)
7:      for box in boxes do
8:          for farea in freeareas do
9:              if ROTATEPACKING(box, farea) then
10:                 UPDATE(farea, freeareas); break
11:         boxes.POP(box)
12: function ROTATEPACKING(box, farea)
13:     if farea.w≥box.w and farea.h≥box.h then return True
14:     else if farea.w≥box.h and farea.h≥box.w then return True
15:     else return False
16: function UPDATE(farea, freeareas, box)
17:     box.APPEND(loc)                     ▷ Append the placed location of box in Bins
18:     ifareas = INNERFREE(farea, box)     ▷ Find rest free areas in box after
                                              packing farea (more in Appx. A)
19:     freeareas = freeareas ∪ ifareas \ {farea}
```

where MB_{size} is the size of a MB, *e.g.*, 16×16 in H.264. H, W, B are the optimal height, width, and batch size for the enhancement model preset in the execution plan (§3.4).

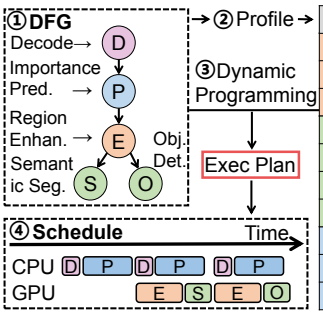
3.3.2 Region-aware Bin Packing. Considering the sparse distribution of selected MBs, the unique characteristics of enhancement models (requires rectangle input and latency is pixel-value-agnostic proportional to the input size), and the helpful but complex batch execution, we propose a region-aware bin packing algorithm to construct selected MBs into irregular regions and stitch them into dense tensors.

The problem is formulated as a *two-dimensional bin-packing problem* [28] that packs the maximum number of selected MBs into given bins. The input are MB indexes and the number of bins B and their size $H \times W$, the output is the packing plan of MBs. We process MB indexes, not real images, to avoid frequent memory I/O. It is known to be NP-hard. Prior methods [37, 38, 76] batch the standard rectangle DNN inputs and thus can not handle irregular regions. To motivate our design, we first analyze two strawmen: 1) MB packing: directly packing selected MBs after expanding three pixels in each direction[‡] into bins, and 2) Irregular region packing: packing the (irregular) regions consisting of connected and selected MBs after extension into bins [67]. MB packing packs too many unimportant or repeated pixels due to extension, thus under-utilizing given bins compared to the irregular manner; whereas the high complexity of existing irregular region packing algorithms causes more than one order of magnitude of time cost than MB packing (more in Appx. C.4).

To strike a better balance between bin utilization and algorithm efficiency, we propose region-aware bin packing in Alg. 1. Its key design choices are: 1) bounding the irregular regions in rectangle boxes for efficient packing plan search (line #3-5); 2) cutting large boxes into small ones and sorting boxes in order of importance density, *i.e.*, the average importance of all MBs in it, for high bin utilization (line #6). Fig. 11 exemplifies that this prioritization leads to a higher (13%) accuracy gain, compared to 6% of traditional large-item-first (max-area-first) policies [57] (more results refer to Fig. 23). Region-aware bin packing first constructs regions by calculating the connected components of selected MB in line #3, and bounds each region in a rectangular box with extension (line #4), *e.g.*, ①②③ in Fig. 11. Next, it partitions the boxes larger than the preset size (line #5) in case of importing too many unimportant MBs, *i.e.*, box ① cut to ①* and ①**. Then, line #6 sorts boxes according to the priority we propose, *i.e.*, not (①,②,③) of max-area-first policy but (②,③,①**). Lastly, it iteratively packs them into bins. In each iteration (line #7-11), it rotates and packs a box into a free area in a bin, updates the box's placing location and the list of free areas, and then deletes the box from the boxes. For example, the "L" region in Fig. 10 and ①** in Fig. 11 is rotated and packed into bins.

3.3.3 Enhancement with Super-resolution Model. Based on the execution plan across various devices (§4.2), except for Jetson AGX Orin equipped with unified memory, the other devices need to transfer real frames from main memory to GPU memory. To save time, we hide this transfer at the same time as the MB selection and bin packing procedure. This idea works because, after processing the real frames by MB-

[‡]Such expansion can avoid the MB/region boundaries causing too many jagged edges and blocky artifacts [45] when pasting enhanced content back to the bi-linear-interpolated frames. For more results refer to Appendix C.3.



Model@HW	Bat	Cos	TPS
SR@CPU	1	270	3
SR@GPU	1	7	142
	4	27	148
OD@GPU	1	2	500
	4	7.8	512
SS@GPU	1	2.7	370
	4	8.3	481
P@CPU	16	27	592
P@GPU	1	1.7	588
	4	4.2	952

Figure 12: An example of execution planning process seven 30-fps streams s.t. <1s latency & >0.9 accuracy performance target. The throughputs (TPS) of one model executing on one hardware (Model@HW) in the right table are profiled with batch size (Bat) and time cost (Cos). The red boxes highlight the execution plan of each component of the given dataflow graph (DFG).

based region importance prediction, as shown in Fig. 7, all modules solely deal with MB indexes before super-resolution. Consequently, we stitch the real-content regions into tensors (bins) following the packing plan on the GPU, subsequently enhancing and stitching them back to bi-linear-interpolated non-regions for final analytics.

3.4 Profile-based Execution Planning

Given an edge server equipped with R computational resource (*i.e.*, R processing time of processors under 100% utilization) and the user's analytical jobs, profile-based executing planning aims to allocate the most appropriate resources to each component, maximizing the end-to-end throughput subject to performance targets. The problem is formulated as follows,

$$\max \ \ T_{e2e}, \ \ s.t. \sum R_u \leq R \ \ \forall u \in V(G).$$

where G is the dataflow graph (DFG) of components; u is the node in graph G and R_u indicates its allocated resource.

We present a profile-based execution planning, as the example of one CPU plus one GPU depicted in Fig. 12, that ① parses the DFG of user-specified analytical tasks, ② runs workload (*e.g.*, one-minute user-specified streams) on all components, including user-uploaded models to profile their capacity on all accessible hardware, ③ generates the execution plan that satisfies user-specified performance targets, and ④ loads each component into corresponding hardware. Our crucial choice is to assign each component's input tensor size (*e.g.*, batch size) for resource allocation. Batching execution, *i.e.*, grouping input matrices into one, is well known to achieve high processor utilization by DNNs; it also allows the inference engines to achieve various throughputs by adjusting different batch sizes [36, 37, 76].

As the DFG of any job, naturally, is a directed acyclic graph (DAG), we can always leverage *dynamic programming* to solve this optimization problem [75]. Define $T_u(r)$ as the maximum throughput of components represented by u and the subtree at u within the resource budget r, which numerically equals the minimum node in each path. For non-leaf node u, the algorithm allocates a resource budget r' for node u and at most $r-r'$ for the subtree, and it then enumerates all $r \leq R$ to find the optimal allocations that assign optimal batch size b for each node. Formally, $\forall (u,v) \in E(G)$,

$$T_u(r) = \max_{r:r\leq R} \left\{ \min(\max_{b:c_u(b)\leq r'} \frac{b}{c_u(b)}, T_v(r-r')) \right\}.$$

where $c_u(b)$ is the resource cost of node u at b batches. RegenHance allocates the least resources for analytical models that satisfy the user's latency target and then assigns other components' batch sizes with this equation.

In general, the optimal solution always converges to the allocation that won't be bottlenecked by any node; in other words, each node in the graph generates the same throughput. Therefore, when users' registrations change frequently, the execution plan must be generated quickly. To this end, we will explore learning-related methods like online deep reinforcement learning [26] and combinatorial-optimization-related methods like local search [50] in future work.

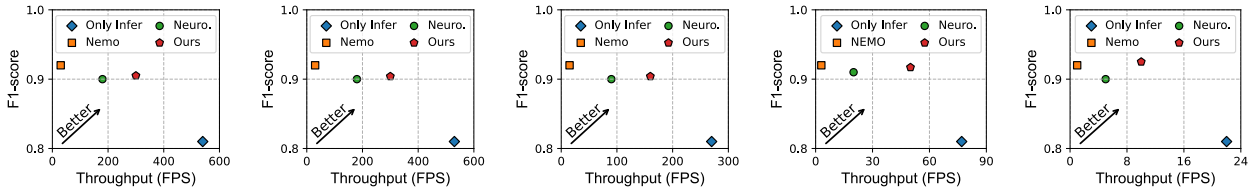
4 Evaluation

We evaluate RegenHance with two video analytics tasks on five heterogeneous devices. Our key findings are:

- RegenHance improves 10-19% accuracy and achieves 2-3× end-to-end throughput compared to the state-of-the-art frame-based enhancement methods. (Sec. 4.3)
- RegenHance shows robust effectiveness on various devices with heterogeneous computational resources, different analytical tasks and models, diverse workloads and performance targets, and various resolutions. (Sec. 4.3)
- MB-based Region Importance Prediction and Region-aware Enhancement bring remarkable accuracy and throughput benefit, and Profile-based Execution Planning boosts great resource utilization and throughput. (Sec. 4.4)

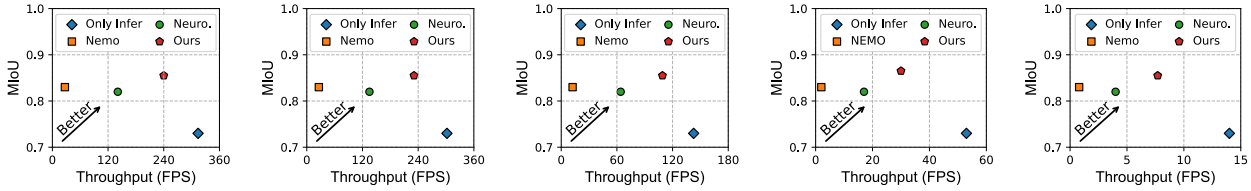
4.1 Implementation

We implement RegenHance upon commercial frameworks including FFmpeg (v4.4.2) [19], Pytorch (v1.8.2) [20], Paddleseg (v2.7.0) [27], ONNX, OpenVINO (v2023.0.1) [22], TensorRT (v8.4.2.4) [11], and the mostly code is implemented by Python (v3.8). The MB-based region importance prediction and the region-aware enhancer are implemented as follows. (1) RegenHance implements the MB importance predictor by retraining MobileSeg (MobileNetV2 backbone) with importance metric and prunes its 50% parameters by L1-Norm pruner. It is further exported to the ONNX version (using `paddle.onnx.export` API) for efficient running on Intel CPU with OpenVINO Runtime, and exported to the TensorRT version for NVIDIA GPU (using `trtexec` library). If not mentioned, all TensorRT models are set as `FP16` in dynamic shape version, and the engine files are exported and inferred by PyCUDA (v2022.2). (2) We modify `ff_h264_idct_add` API in



(a) 4090 (b) A100 (c) 3090Ti (d) T4 (e) Jonson AGX Orin

Figure 13: Accuracy and throughput comparison over various devices (Object detection).



(a) 4090 (b) A100 (c) 3090Ti (d) T4 (e) Jetson AGX Orin

Figure 14: Accuracy and throughput comparison over various devices (Semantic segmentation).

Tasks	Metric	Models	Dataset
Object Detection	F1-score Stream #	Mask R-CNN (Swin) YOLO	YODA VideoClips
Semantic Segmentation	mIoU Stream #	FCN HarDNet	Cityscape BDD100K

Table 1: Summary of downstream tasks and datasets.

FFmpeg to extract the residual for temporal MB importance reuse. (3) RegenHance uses a pre-trained super-resolution model [64] as enhancer, and also converts it to ONNX version and TensorRT version for efficiency.

Loading models into memory can cost hundreds of milliseconds to seconds. Therefore, at runtime, RegenHance preloads the DNN into the specified processors and then invokes it. RegenHance consists of ∼5.3K lines of code.

4.2 Experimental Setup

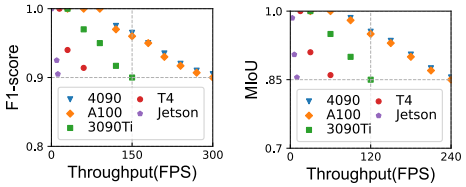
Downstream tasks and dataset. As summarized in Tab.1, we select two downstream analytical tasks, object detection and semantic segmentation, to evaluate the performance of RegenHance as they play the core role in a wide range of high-level tasks. Object detection aims to identify objects of interest (*i.e.*, locations and classes) on each video frame; its accuracy is measured by average *F1-score* in each stream with Intersection over Union (IoU) threshold at 0.5. Semantic segmentation labels each pixel with one class, and we measure its accuracy with *mIoU*. We also evaluate their *throughput*, *i.e.*, number of streams that can be processed in real-time.

Both of the tasks were separately tested with two models (light- and heavyweight) on two video sets. For the object detection, we used the Yoda dataset [88] and gathered 120 video clips from YouTube containing various scenes with diverse characteristics, *i.e.*, time, illumination, objects' density and speed, and road type. The labels to train importance predictor are generated with Mask R-CNN (Swin backbone) on per-frame enhancement since its SOTA performance; if users provide their self-tailored analytical models, we use their output as labels. Here, we use YOLO as an example. We will release this dataset after this paper is accepted. For semantic segmentation, we utilized the BDD100K [2] and Cityscape [35] public dataset. We re-encoded these datasets into 360P resolution, 30 fps, and 1024kbps bitrate videos in H.264 to avoid the impacts of various video codecs.

Devices. We conducted experiments on five heterogeneous devices divided into four categories. (1) As a comparison, we deployed and tested RegenHance on a cloud server with an NVIDIA A100 GPU and Intel(R) i9-12900K CPU. (2) As one of the most popular configurations of edge servers, we conducted experiments on an edge equipped with an NVIDIA Tesla T4 and Intel i7-8700 CPU. (3) To explore the capabilities of gaming cards and their generation gap, we tested an NVIDIA RTX4090 and an NVIDIA RTX3090Ti, respectively, with an Intel i9-13900K. (3) For the embedded edge, we used an NVIDIA Jetson AGX Orin 64GB as the platform.

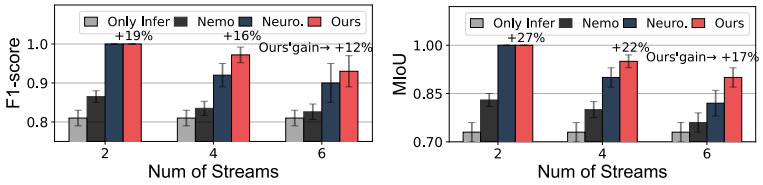
Baselines. To show the superiority brought by RegenHance, we compare it with four different baselines:

- **Only infer** directly applies analytical DNNs on each original frame without enhancement.
- **NeuroScaler [95]** is the state-of-the-art frame-based enhancement method. It first enhances the anchors and reuses their quality gain on non-anchors, then infers all frames. It fast selects anchors in a heuristic manner.
- **Nemo [93]** also only enhances the anchors and reuses their quality gain, then infers all frames; but it iteratively selects the best anchors based on enhancement results.



(a) Object detection (b) Semantic segmentation

Figure 15: TPT-ACC tradeoff on different devices.



(a) Object detection (b) Semantic segmentation

Figure 16: Accuracy comparison over various stream numbers.

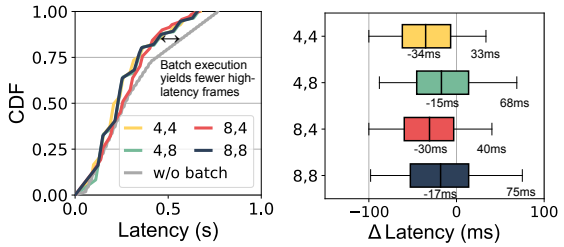


Figure 17: *Left*: Latency of each frame with diverse batch sizes (Infer#, SR#). *Right*: The latency diff between execution w/ and w/o batch of each frame ($\Delta Lat = Lat_{batch}(i) - Lat_{w/o\ batch}(i)$).

Metric	360P	720P
BW (Mbps)	0.96	3.00
Max Stream #	11	10
GPU Usage (SR)	0.23	0.17
Acc Gain	9%	7%

Table 2: Performance trade-off under different video resolutions.

Method	TPT (FPS)
Per-frame SR	95
PF + Plan.	111
PF + Pred. + Plan.	111
PF + Pred. + Enhan.	179
RegenHance	300

Table 3: End-to-end throughput (FPS) breakdown of RegenHance.

4.3 End-to-End Performance

This section reports the E2E performance of RegenHance on various devices. If not mentioned, all results are tested on RTX4090 with 1s latency, 90% object detection, and 88% semantic segmentation (10% gain compared to only infer).

Performance on heterogeneous devices. As shown in Fig. 13 and Fig. 14, RegenHance achieves high accuracy and high throughput simultaneously on various devices. It of course cannot reach the throughput of only infer due to the additional enhancement time cost but provides significant throughput gain compared to SOTA NEMO and NeuroScale. On average, RegenHance outperforms their throughput by 12× and 2.1× in object detection and 11× and 1.9× in semantic segmentation, respectively. This larger performance gain in semantic segmentation stems from its heightened sensitivity to visual details. RegenHance can keep this superiority on all five heterogeneous devices because profile-based execution planning always generates the optimal throughput plans.

Trade-off between throughput and accuracy. RegenHance creates a trade-off space between accuracy and throughput, as shown in Fig. 15; it can offer service on edge servers with heterogeneous resources. Edge servers equipped with higher resources produce larger trade-off space. For example, RegenHance supports object detection for ten streams (300fps) with 91% accuracy on RTX4090 or A100 GPU; and if the accuracy constraint is more strict, RegenHance will make corresponding adjustments and serve the maximum number of streams, *e.g.*, six 95%-accuracy streams. On devices with lower available resources, *e.g.*, on NVIDIA T4 and Jetson AGX Orin, although the maximum frame rate decreases, RegenHance still delivers remarkable throughput under different accuracy targets.

Performance gain with multiple streams. With the increasing number of competing streams, RegenHance always achieves the highest accuracy compared to the other three frame-based enhancement methods. For example, in Fig. 16 tests on RTX4090, RegenHance improves accuracy by 8-14% compared to the selective enhancement in six streams. This is because, in such a highly competitive scenario where each stream is allocated limited resources, our method consistently enhances the most valuable regions; in contrast, the selective baseline and the per-frame baseline waste excessive resources on unimportant content. RegenHance significantly improves the throughput of content-enhanced video analytics.

Frame latency under diverse batch sizes. The latency is defined the same as prior studies (*e.g.*, DDS [44], AWStream [101], and Reducto [62]), *i.e.*, the time from encoding 1s video chunk (30 frames) on cameras to all 30 frames' inference results are available on edge. The left figure of Fig. 17 plots the latency of each frame with different batch sizes under the same workflow. The results imply that batch execution yields fewer high-latency frames because its average latency of 30 frames in each chunk is much lower than executed without batch, which stems from the batch's higher GPU utilization. For a deeper insight, the right figure of Fig. 17 evaluates the latency difference of each frame with and without batch execution. Compared to without batch, execution with batch may cause 75ms latency at most, which is the earliest frame (in video play order) in one batch; but on average, batch execution saves time cost.

Performance gain under different resolutions. When streaming 720p in object detection while 360p for semantic segmentation under a target of 90% accuracy, as shown in Table 2, 360p video costs only 31% bandwidth (0.96Mbps *v.s.*

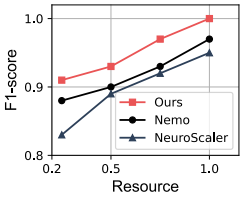


Figure 18: Accuracy on different resources.

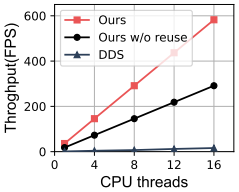


Figure 19: Throughput of region prediction.

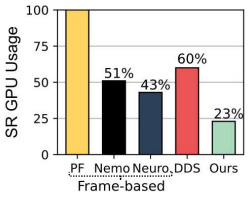


Figure 20: GPU resources usage.

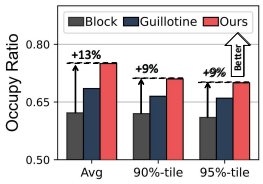


Figure 21: Perf. of different packing policies.

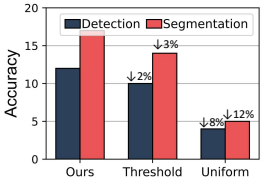


Figure 22: Acc gain of Cross-stream MB sel.

3Mbps) compared to 720p. RegenHance boosts accuracy gain of 360p video from 81% to 90%, while 720p video from 83% to 90%. Higher resolution yields higher accuracy; however, content enhancement can still improve the details of small objects or blurred content. The end-to-end throughput of 720p is almost the same as 360p (11 *v.s.* 10 streams). The reason is that although 720p needs to enhance fewer regions compared to 360p (17% *v.s.* 23% GPU usage for SR), its time cost of other runtime components (*e.g.*, MB importance prediction) is larger than 360p due to the larger input size.

Contributions by individual components. Table 3 shows the throughput breakdown of RegenHance. We tested multiple versions of RegenHance by selectively applying each component. The results show that each component significantly contributes to the improvements in the throughput. The execution planning enables a 1.2× throughput compared to per-frame enhancement by reasonable resource allocation (1st→2nd). Incorporating MB-based region importance prediction without region-aware enhancement fails to improve throughput because filling unimportant regions to black does not alter the enhancement latency (2nd→3rd row). The throughput is increased by 1.6× due to the region-aware enhancement (3rd→4th row) and by 1.7× from the best execution plan of profile-based execution planning (4th→5th row).

4.4 Comprehensive Component-wise Analysis

We provide an in-depth performance analysis of individual system components.

4.4.1 MB-based region Importance Prediction (§3.2) helps RegenHance achieve great throughput improvement while keeping high accuracy.

Accuracy: Fig. 18 illustrates the accuracy gain obtained by different enhancement methods, when assigned the same computational resource to complete object detection on six streams. RegenHance (region-based enhancement) can offer a 3-4% and 4-8% higher gain compared to (frame-based) Nemo and Neuralscaler, respectively, because our predictor identifies the most beneficial region precisely.

Thoughput: As shown in Fig. 19, our ultra-lightweight MB importance predictor can be executed at 30 fps on one single i7-8700 CPU core, which outperforms the RoI selection by RPN in DDS more than 60 times. On GPU, it achieves 973 fps, which outperforms DDS more than 12 times. The reuse contributes 2 times throughput improvement.

Resource saving: Fig. 20 shows that our method reduces 77%, 28%, and 20% GPU usage compared to the frame-based Per-frame enhancement, Nemo, and NeuroScaler, respectively, when enhancing a single stream (30FPS) to achieve an accuracy exceeding 90%. Compared to the RoI selection of DDS, our method saves 37% GPU usage as it identifies the most beneficial region for analytical accuracy more precisely.

4.4.2 Region-aware Enhancement (§3.3) yields significant throughput improvement and overall accuracy gain.

Throughput: Region-aware bin packing maximizes RegenHance's throughput with a high stability and occupy ratio, and thus effectively reducing the additional overhead of SR. We conducted experiments repeated 1,000 times by randomly shuffling the order of six video streams, and comparing the occupy ratio, *i.e.*, the ratio of selected MBs occupying all enhanced content, with the classic Guillotine policy [57] and the Block policy, *i.e.*, MB packing. Fig. 21 shows that the average, 90%-tile, and 95%-tile occupy ratio difference; our packing policy gets the highest occupy ratio of 75% that outperforms the comparisons by up to 13%, 9%, and 9%.

Accuracy: Cross-stream MB selection, especially our custom sorting order, leads to remarkable accuracy gain by considering the heterogeneous region accuracy gain across streams. We compared our MB selection with the Uniform method, which allocates the same MB number to each stream, and the Threshold method sets a fixed threshold, 0.5, to select MB importance for all streams. As shown in Fig. 22, our method outperforms the Uniform and Threshold methods by 8-12% and 2-3% accuracy gain, respectively. Fig 23 demonstrates the superiority of our custom sorting priority. Compared to the traditional large-item-first (max-region-first in our context) packing policy, our method yields a 50% accuracy gain.

4.4.3 Profile-based Execution Planning (§3.4) profiles different devices and allocates devices and resources to different components and models based on the results, thereby maximizing throughput.

Resources allocation: Profile-based execution planning makes RegenHance achieve the best performance on given devices and workloads by avoiding bottlenecked by any component. Fig. 24 visualizes the computational resource allocation of two different object detection models on i9-13900K+RTX4090 providing satisfied performance as Fig. 13(a). With distinct workloads of YOLOv5s (16.9GFLOPs)

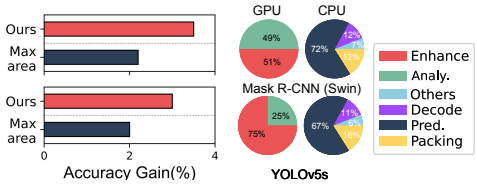


Figure 23: Acc gain of our MB selection.

Figure 24: Execution plan for different workload.

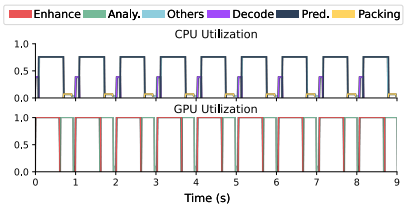


Figure 25: GPU & CPU usage.

Component	RRobin	Ours
MB Prediction	194	533
Enhancement	80	186
Analytics	220	186
Throughput	**80**	**186**

Table 4: Throughput (FPS) gain to region-agnostic strawman in §2.4.

and Mask R-CNN (Swin backbone, 267GFLOPS), it allocates more resources for the analytical task.

Resource usage: Fig. 25 shows the real-time utilization of CPU and GPU when detecting objects on six video streams. We used Nvidia Nsight to monitor GPU utilization and HTOP for CPU utilization. The GPU can reach full load approximately 95-99% of the time, while the CPU can reach high load around 81% of the time. This result indicates that the execution plan achieves an efficient GPU-CPU corporation.

Throughput: We avoid the enhancement bottleneck introduced by the round-robin manner (§2.4), which allocates equal resources to each component. The results in Table 4 indicate that our approach achieves 2.3× throughput.

Scalability and initialization: There are two types of preparation time in our system. If only the ingest video streams change, it needs approximately 0.6-2 seconds for initialization depending on the specific device and model requirements. When given a new device, it needs to take approximately 1-3 minutes to execute the Profile-based Execution Planning.

5 Related Work

Video analytics is facilitated by the advances of deep learning. However, the high analytical accuracy from deep learning comes at a prohibitive computing cost that most end devices cannot afford [87]. For example, today's deployed traffic cameras cost only $40 to $200, equipped with a single-core CPU that provides very scarce computational resources. To tackle this issue, today's video analytics apply a *distributed* architecture [45, 51, 58, 62, 65, 69, 91, 98–100, 102, 103].

Video compression saves bandwidth resources by adjusting video encoding configurations (*e.g.*, resolution, bitrate, and frame rate). Like traditional video streaming protocols like WebRTC [47] and DASH [77], distributed video analytics must tackle bandwidth constraints. Prior studies, no matter server-feedback accurate configuration setup [56, 69, 101, 103] or fast autonomy by video source (camera) itself [45, 65, 89, 91], addressed this issue well, saving considerable bandwidth resources but causing accuracy drop.

Selective inference, also called frame/image filtering, filters similar frames/images out to meet low resource cost and high throughput. Cameras leverage pixel-level difference [34, 40, 62] to filter frames for bandwidth saving; edge servers use light DNNs to select the frames to be detected by heavy DNNs, then reuse their output on filtered frames for throughput increment [51, 58, 98, 100, 102]. Selective inference achieves satisfactory throughput and resource-saving but lower accuracy. DDS [44] iteratively detects the selected images to avoid accuracy drop, but its twice cross-network transmission and detection imports too much delay. Turbo [68] seeks the same goal by enhancing frames on idle GPU slots. Differing from it, RegenHance boosts the enhancement method and fully uses heterogeneous resources on edge.

Model optimization meets high throughput by network pruning [61] and weights quantization [54], and achieves high accuracy by knowledge distillation [49]. However, the compressed model is easily affected by data drift, where the live video data diverges from the training data, leading to accuracy degradation [74]. To tackle this, continuous learning [60, 74], model switching [59, 95], and model merging [55, 71] are proposed by previous studies. Intuitively, continuous learning and model switching leverage small DNNs for high throughput and improve accuracy by matching the weights in DNN to the changed real-world videos. Model merging differs from traditional parameter sharing in one model [72]. It explores structural similarities between concurrent models and enables multiple models to share the same network components, *e.g.*, the backbone [71], by retraining.

6 Conclusion

In this paper, we look at content enhancement in video analytics applications. We found that frame-based content enhancement wastes too much computation on the analytics-agnostic image. We presented the region-based content enhancement technique and well-matched region-aware resource scheduler and implemented RegenHance. In our evaluation using five heterogeneous devices, we show that RegenHance can deliver an order of magnitude than frame-based content-enhanced video analytics.

Acknowledgements. We thank the anonymous NSDI reviewers for their constructive comments. This work was supported by Carbon Neutrality and Energy System Transformation (CNEST) Program, Tsinghua University (AIR)-AsiaInfo Technologies (China), Inc. Joint Research Center for 6G Network and Intelligent Computing, NSFC (No. 62272261, 62402280, U22A2031) the Fundamental Research Funds for the Central Universities (No. 2024300349), EU Horizon CODECO projects (No. 101092696), Shuimu Tsinghua Scholar Program (No. 2023SM201).

References

[1] Axis. axis for a safety touch at the grey cup festival. https://www.axis.com/files/success_stories/ss_stad_greycup_festival_58769_en_1407_lo.pdf.

[2] Bdd100k dataset. https://www.vis.xyz/bdd100k/.

[3] British transport police: Cctv. http://www.btp.police.uk/adviceandinformation/safetyonandneartherailway/cctv.aspx.

[4] Can 30,000 cameras help solve chicago's crime problem? https://www.nytimes.com/2018/05/26/us/chicago-police-surveillance.html.

[5] Global sports analytics market size report, 2021-2028. https://www.grandviewresearch.com/industry-analysis/sports-analytics-market.

[6] Google cloud vision. https://cloud.google.com/vision.

[7] H.264 specification. https://www.itu.int/rec/T-REC-H.264.

[8] Kubeedge. https://github.com/kubeedge/kubeedge.

[9] Microsoft rocket. https://www.microsoft.com/en-us/research/project/live-video-analytics/.

[10] NVIDIA T4. https://www.nvidia.com/en-us/data-center/tesla-t4/.

[11] NVIDIA TensorRT. https://developer.nvidia.com/tensorrt.

[12] One legacy of tiananmen: China's 100 million surveillance cameras. https://www.wsj.com/articles/BL-CJB-22562.

[13] One surveillance camera for every 11 people in britain, says cctv survey. https://www.telegraph.co.uk/technology/10172298/One-surveillance-camera-for-every-11-\people-in-Britain-says-CCTV-survey.html.

[14] Robo scores world cup to have 'robot linesmen' for first time in history as fifa ready to say trials have worked. https://www.thesun.co.uk/sport/18807051/world-cup-robot-linesman-fifa-qatar/.

[15] Video analytics applications in retail - beyond security. https://www.securityinformed.com/insights/co-2603-ga-co-2214-ga-co-1880-ga.16620.html/.

[16] Video analytics market - growth, trends, covid-19 impact, and forecasts (2022 - 2027). https://www.mordorintelligence.com/industry-reports/video-analytics-market.

[17] Video analytics traffic study creates baseline for change. https://www.govtech.com/analytics/Video-Analytics-Traffic-Study-Creates\-Baseline-for-Change.html.

[18] Wyze. wyze camera. https://www.safehome.org/homesecurity-cameras/wyze/.

[19] Ffmpeg: A complete, cross-platform solution to record, convert and stream audio and video. https://ffmpeg.org/, 2021.

[20] Pytorch: Official website of the python platform. https://pytorch.org/, 2021.

[21] Speedtest market report. https://www.speedtest.net/reports/zh/, 2021.

[22] Openvino™ toolkit: An open source toolkit that makes it easier to write once, deploy anywhere. https://www.intel.com/content/www/us/en/developer/tools/openvino-toolkit/overview.html, 2023.

[23] Neil Agarwal and Ravi Netravali. Boggart: Towards General-Purpose acceleration of retrospective video analytics. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 933–951, 2023.

[24] Namhyuk Ahn, Byungkon Kang, and Kyung-Ah Sohn. Fast, accurate, and lightweight super-resolution with cascading residual network. In *European Conference on Computer Vision (ECCV)*, 2018.

[25] Ganesh Ananthanarayanan, Paramvir Bahl, Peter Bodik, Krishna Chintalapudi, Matthai Philipose, Lenin Ravindranath, and Sudipta Sinha. Real-time video analytics: The killer app for edge computing. *Computer*, 50(10):58–67, 2017.

[26] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[27] PaddlePaddle Authors. Paddleseg, end-to-end image segmentation kit based on paddlepaddle. https://github.com/PaddlePaddle/PaddleSeg, 2019.

[28] Judith O Berkey and Pearl Y Wang. Two-dimensional finite bin-packing algorithms. *Journal of the Operational Research Society*, 38:423–429, 1987.

[29] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.

[30] Adrian Bulat and Georgios Tzimiropoulos. Super-fan: Integrated facial landmark localization and super-resolution of real-world low resolution faces in arbitrary poses with gans. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[31] F Cangialosi, N Agarwal, V Arun, J Jiang, S Narayana, A Saarwate, and R Netravali. Privid: Practical, privacy-preserving video analytics queries. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2022.

[32] Ping Chao, Chao-Yang Kao, Yu-Shan Ruan, Chien-Hsiang Huang, and Youn-Long Lin. Hardnet: A low memory traffic network. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3552–3561, 2019.

[33] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[34] Tiffany Yu-Han Chen, Lenin Ravindranath, Shuo Deng, Paramvir Bahl, and Hari Balakrishnan. Glimpse: Continuous, Real-Time Object Recognition on Mobile Devices. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 155–168, Seoul, South Korea, 2015.

[35] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[36] Weihao Cui, Mengze Wei, Quan Chen, Xiaoxin Tang, Jingwen Leng, Li Li, and Mingyi Guo. Ebird: Elastic batch for improving responsiveness and throughput of deep learning services. In *IEEE International Conference on Computer Design (ICCD)*, pages 497–505, 2019.

[37] Weihao Cui, Han Zhao, Quan Chen, Hao Wei, Zirui Li, Deze Zeng, Chao Li, and Minyi Guo. DVABatch: Diversity-aware Multi-Entry Multi-Exit batching for efficient processing of DNN services on GPUs. In *USENIX Annual Technical Conference (ATC)*, pages 183–198, 2022.

[38] Weihao Cui, Han Zhao, Quan Chen, Ningxin Zheng, Jingwen Leng, Jieru Zhao, Zhuo Song, Tao Ma, Yong Yang, Chao Li, et al. Enable simultaneous DNN services based on deterministic operator overlap and precise latency prediction. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 1–15, 2021.

[39] Jason Jinquan Dai, Ding Ding, Dongjie Shi, Shengsheng Huang, Jiao Wang, Xin Qiu, Kai Huang, Guoqiong Song, Yang Wang, Qiyuan Gong, Jiaming Song, Shan Yu, Le Zheng, Yina Chen, Junwei Deng, and Ge Song. Bigdl 2.0: Seamless scaling of ai pipelines from laptops to distributed cluster. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21407–21414, 2022.

[40] Xin Dai, Xiangnan Kong, Tian Guo, and Yixian Huang. Cinet: Redesigning deep neural networks for efficient mobile-cloud collaborative inference. In *SIAM International Conference on Data Mining (ICDM)*, pages 459–467, 2021.

[41] Mallesham Dasari, Arani Bhattacharya, Santiago Vargas, Pranjal Sahu, Aruna Balasubramanian, and Samir R Das. Streaming 360-degree videos using super-resolution. In *IEEE Conference on Computer Communications (INFOCOM)*, pages 1977–1986, 2020.

[42] Sokemi Rene Emmanuel Datondji, Yohan Dupuis, Peggy Subirats, and Pascal Vasseur. A survey of vision-based traffic monitoring of road intersections. *IEEE Transactions on Intelligent Transportation Systems*, 17(10):2681–2698, 2016.

[43] Sina G. Davani and Nabil J. Sarhan. Experimental analysis of optimal bandwidth allocation in computer vision systems. *TCSVT*, 31(10):4121–4130, 2021.

[44] Kuntai Du, Ahsan Pervaiz, Xin Yuan, Aakanksha Chowdhery, Qizheng Zhang, Henry Hoffmann, and Junchen Jiang. Server-Driven Video Streaming for Deep Learning Inference. In *ACM Special Interest Group on Data Communication (SIGCOMM)*, pages 557–570. ACM, 2020.

[45] Kuntai Du, Qizheng Zhang, Anton Arapin, Haodong Wang, Zhengxu Xia, and Junchen Jiang. Accmpeg: Optimizing video encoding for video analytics. In *Machine Learning and Systems*, 2022.

[46] Glenn Jocher et. al. ultralytics/yolov5: v6.0 - YOLOv5n 'Nano' models, Roboflow integration, TensorFlow export, OpenCV DNN support, October 2021.

[47] Google. Webrtc official website. https://webrtc.org/, 2011.

[48] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

[49] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[50] Holger H Hoos and Thomas Stützle. *Stochastic local search: Foundations and applications*. Elsevier, 2004.

[51] Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodik, Shivaram Venkataraman, Paramvir Bahl, Matthai Philipose, Phillip B Gibbons, and Onur Mutlu. Focus: Querying Large Video Datasets with Low Latency and Low Cost. In *USENIX Symposium on Operating System Design and Implementation (OSDI)*, page 19, 2018.

[52] Weiming Hu, Nianhua Xie, Li Li, Xianglin Zeng, and Stephen Maybank. A survey on visual content-based video indexing and retrieval. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 41(6):797–819, 2011.

[53] Jinwoo Hwang, Minsu Kim, Daeun Kim, Seungho Nam, Yoonsung Kim, Dohee Kim, Hardik Sharma, and Jongse Park. Cova: Exploiting compressed-domain analysis to accelerate video analytics. In *USENIX Annual Technical Conference (ATC)*, pages 707–722, 2022.

[54] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2704–2713, 2018.

[55] Angela H. Jiang, Daniel L.-K. Wong, Christopher Canel, Lilia Tang, Ishan Misra, Michael Kaminsky, Michael A. Kozuch, Padmanabhan Pillai, David G. Andersen, and Gregory R. Ganger. Mainstream: Dynamic Stem-Sharing for Multi-Tenant video processing. In *USENIX Annual Technical Conference (USENIX ATC 18)*, pages 29–42, 2018.

[56] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. Chameleon: scalable adaptation of video analytics. In *ACM Special Interest Group on Data Communication (SIGCOMM)*, pages 253–266, August 2018.

[57] J. Jylänki. A thousand ways to pack the bin-a practical approach to two-dimensional rectangle bin packing. *http://clb. demon. fi/files/RectangleBinPack. pdf.*, 2010.

[58] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. NoScope: optimizing neural network queries over video at scale. *Proceedings of the VLDB Endowment*, 10(11):1586–1597, August 2017.

[59] Mehrdad Khani, Ganesh Ananthanarayanan, Kevin Hsieh, Junchen Jiang, Ravi Netravali, Yuanchao Shu, Mohammad Alizadeh, and Victor Bahl. RECL: Responsive resource-efficient continuous learning for video analytics. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 917–932, 2023.

[60] Jaehong Kim, Youngmok Jung, Hyunho Yeo, Juncheol Ye, and Dongsu Han. Neural-enhanced live streaming: Improving live video ingest via online learning. In *ACM Special Interest Group on Data Communication (SIGCOMM)*, pages 107–125, 2020.

[61] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[62] Yuanqi Li, Arthi Padmanabhan, Pengzhan Zhao, Yufei Wang, Guoqing Harry Xu, and Ravi Netravali. Reducto: On-Camera Filtering for Resource-Efficient Real-Time Video Analytics. In *ACM Special Interest Group on Data Communication (SIGCOMM)*, pages 359–376, 2020.

[63] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1833–1844, 2021.

[64] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1132–1140, 2017.

[65] Luyang Liu, Hongyu Li, and Marco Gruteser. Edge Assisted Real-time Object Detection for Mobile Augmented Reality. In *ACM International Conference on Mobile Computing and Networking (MobiCom)*, pages 1–16, August 2019.

[66] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[67] Eunice López-Camacho, Gabriela Ochoa, Hugo Terashima-Marín, and Edmund K Burke. An effective heuristic for the two-dimensional irregular bin packing problem. *Annals of Operations Research*, 206:241–264, 2013.

[68] Yan Lu, Shiqi Jiang, Ting Cao, and Yuanchao Shu. Turbo: Opportunistic enhancement for edge video analytics. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 263–276, 2022.

[69] Marwa Meddeb. Region-of-interest-based video coding for video conference applications. page 172.

[70] Junhyug Noh, Wonho Bae, Wonhee Lee, Jinhwan Seo, and Gunhee Kim. Better to follow, follow to be better: Towards precise supervision of feature super-resolution for small object detection. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9724–9733, 2019.

[71] Arthi Padmanabhan, Neil Agarwal, Anand Iyer, Ganesh Ananthanarayanan, Yuanchao Shu, Nikolaos Karianakis, Guoqing Harry Xu, and Ravi Netravali. Gemel: Model merging for memory-efficient, real-time video analytics at the edge. In *USENIX Symposium on Network System Design and Implementation (NSDI)*, 2023.

[72] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International conference on machine learning (ICML)*, pages 4095–4104. PMLR, 2018.

[73] Rishabh Poddar, Ganesh Ananthanarayanan, Srinath Setty, Stavros Volos, and Raluca Ada Popa. Visor:Privacy-Preserving video analytics as a cloud service. In *USENIX Security Symposium (Security)*, pages 1039–1056, 2020.

[74] Bhardwaj Romil, Xia Zhengxu, Ananthanarayanan Ganesh, Jiang Junchen, Shu Yuanchao, Karianakis Nikolaos, Hsieh Kevin, Bahl Paramvir, and Stoica Ion. Ekya: Continuous learning of video analytics models on edge compute servers. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2022.

[75] Dasgupta Sanjoy, Papadimitriou Christos, and Vazirani Umesh. *Algorithms*. 2008.

[76] Haichen Shen, Lequn Chen, Yuchen Jin, Liangyu Zhao, Bingyu Kong, Matthai Philipose, Arvind Krishnamurthy, and Ravi Sundaram. Nexus: A gpu cluster engine for accelerating dnn-based video analysis. In *ACM Symposium on Operating Systems Principles (SOSP)*, pages 322–337, 2019.

[77] Thomas Stockhammer. Dynamic adaptive streaming over http– standards and design principles. In *Pro. of MMSys*, pages 133–144, 2011.

[78] Gary J Sullivan and Thomas Wiegand. Rate-distortion optimization for video compression. *IEEE Signal Processing Magazine*, 15(6):74–90, 1998.

[79] Lin Sun, Weijun Wang, Tingting Yuan, Liang Mi, Haipeng Dai, Yunxin Liu, and Xiaoming Fu. Biswift: Bandwidth orchestrator for multi-stream video analytics on edge. *arXiv preprint arXiv:2312.15740*, 2024.

[80] Weimin Tan, Bo Yan, and Bahetiyaer Bare. Feature super-resolution: Make machine see more clearly. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3994–4002, 2018.

[81] Shiyu Tang, Ting Sun, Juncai Peng, Guowei Chen, Yuying Hao, Manhui Lin, Zhihong Xiao, Jiangbin You, and Yi Liu. Pp-mobileseg: Explore the fast and accurate semantic segmentation model on mobile devices. *arXiv preprint arXiv:2304.05152*, 2023.

[82] Li Wang, Dong Li, Yousong Zhu, Lu Tian, and Yi Shan. Dual super-resolution learning for semantic segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3774–3783, 2020.

[83] Weijun Wang, Tingting Yuan, Minghao Han, Meng Li, Han Zhang, Yu Ma, Sripriya Srikant Adhatarao, and Xiaoming Fu. Poster: A real-time social distance measurement and record system for covid-19. In *ACM International Conference on Embedded Wireless Systems and Networks*, page 179–180, 2021.

[84] Yiding Wang, Weiyan Wang, Duowen Liu, Xin Jin, Junchen Jiang, and Kai Chen. Enabling edge-cloud video analytics for robotics applications. *IEEE Transactions on Cloud Computing*, 2022.

[85] Yiding Wang, Weiyan Wang, Junxue Zhang, Junchen Jiang, and Kai Chen. Bridging the {Edge-Cloud} barrier for real-time advanced vision analytics. In *USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, 2019.

[86] Fabian Wölk, Tingting Yuan, Krisztina Kis-Katos, and Xiaoming Fu. Measuring consumption changes in rural villages based on satellite image data—a case study for thailand and vietnam. In *IEEE International Conference on Mobility, Sensing and Networking (MSN)*, pages 600–607, 2021.

[87] Carole-Jean Wu, David Brooks, Kevin Chen, Douglas Chen, Sy Choudhury, Marat Dukhan, Kim Hazelwood, Eldad Isaac, Yangqing Jia, Bill Jia, Tommer Leyvand, Hao Lu, Yang Lu, Lin Qiao, Brandon Reagen, Joe Spisak, Fei Sun, Andrew Tulloch, Peter Vajda, Xiaodong Wang, Yanghan Wang, Bram Wasti, Yiming

Wu, Ran Xian, Sungjoo Yoo, and Peizhao Zhang. Machine learning at facebook: Understanding inference at the edge. In *IEEE Symposium on High Performance Computer Architecture (HPCA)*, pages 331–344, 2019.

[88] Zhujun Xiao, Zhengxu Xia, Haitao Zheng, Ben Y Zhao, and Junchen Jiang. Towards performance clarity of edge video analytics. In *IEEE/ACM Symposium on Edge Computing (SEC)*, pages 148–164, 2021.

[89] Xiufeng Xie and Kyu-Han Kim. Source Compression with Bounded DNN Perception Loss for IoT Edge Computer Vision. In *ACM International Conference on Mobile Computing and Networking (MobiCom)*, pages 1–16, 2019.

[90] Mengwei Xu, Mengze Zhu, Yunxin Liu, Felix Xiaozhu Lin, and Xuanzhe Liu. DeepCache: Principled Cache for Mobile Deep Vision. In *ACM International Conference on Mobile Computing and Networking (MobiCom)*, pages 129–144, 2018.

[91] Shuochao Yao, Jinyang Li, Dongxin Liu, Tianshi Wang, Shengzhong Liu, Huajie Shao, and Tarek Abdelzaher. Deep compressive offloading: Speeding up neural network inference by trading edge computation for network latency. In *ACM conference on Embedded Networked Sensor Systems (SenSys)*, pages 476–488, 2020.

[92] Juncheol Ye, Hyunho Yeo, Jinwoo Park, and Dongsu Han. Accelir: Task-aware image compression for accelerating neural restoration. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18216–18226, 2023.

[93] Hyunho Yeo, Chan Ju Chong, Youngmok Jung, Juncheol Ye, and Dongsu Han. Nemo: enabling neural-enhanced video streaming on commodity mobile devices. In *ACM International Conference on Mobile Computing and Networking, (MobiCom)*, pages 1–14, 2020.

[94] Hyunho Yeo, Youngmok Jung, Jaehong Kim, Jinwoo Shin, and Dongsu Han. Neural adaptive content-aware internet video delivery. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 645–661, 2018.

[95] Hyunho Yeo, Hwijoon Lim, Jaehong Kim, Youngmok Jung, Juncheol Ye, and Dongsu Han. Neuroscaler: Neural video enhancement at scale. In *ACM Special Interest Group on Data Communication (SIGCOMM)*, page 795–811, 2022.

[96] Juheon Yi, Sunghyun Choi, and Youngki Lee. EagleEye: wearable camera-based person identification in crowded urban spaces. In *ACM International Conference on Mobile Computing and Networking (MobiCom)*, pages 1–14, April 2020.

[97] Juheon Yi, Seongwon Kim, Joongheon Kim, and Sunghyun Choi. Supremo: Cloud-assisted low-latency super-resolution in mobile devices. *IEEE Transactions on Mobile Computing*, 21(5):1847–1860, 2020.

[98] Mu Yuan, Lan Zhang, Fengxiang He, Xueting Tong, and Xiang-Yang Li. Infi: End-to-end learnable input filter for resource-efficient mobile-centric inference. In *ACM International Conference on Mobile Computing And Networking (MobiCom)*, page 228–241, 2022.

[99] Mu Yuan, Lan Zhang, Xuanke You, and Xiang-Yang Li. Packetgame: Multi-stream packet gating for concurrent video inference at scale. In *ACM Special Interest Group on Data Communication (SIGCOMM)*, 2023.

[100] Tingting Yuan, Liang Mi, Weijun Wang, Haipeng Dai, and Xiaoming Fu. Accdecoder: Accelerated decoding for neural-enhanced video analytics. In *IEEE Conference on Computer Communications (INFOCOM)*, 2023.

[101] Ben Zhang, Xin Jin, Sylvia Ratnasamy, John Wawrzynek, and Edward A. Lee. AWStream: adaptive wide-area streaming analytics. In *ACM Special Interest Group on Data Communication (SIGCOMM)*, pages 236–252. ACM, 2018.

[102] Tan Zhang, Aakanksha Chowdhery, Paramvir Victor Bahl, Kyle Jamieson, and Suman Banerjee. The design and implementation of a wireless video surveillance system. In *ACM Conference on Mobile Computing and Networking (MobiCom)*, pages 426–438. ACM, 2015.

[103] Wuyang Zhang, Zhezhi He, Luyang Liu, Zhenhua Jia, Yunxin Liu, Marco Gruteser, Dipankar Raychaudhuri, and Yanyong Zhang. Elf: Accelerate high-resolution mobile deep vision with content-aware parallel offloading. In *ACMInternational Conference on Mobile Computing and Networking (MobiCom)*, pages 201–214, 2021.

[104] Zhengdong Zhang and Vivienne Sze. Fast: A framework to accelerate super-resolution processing on compressed videos. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 19–28, 2017.

[105] Yuan Zhi, Zhan Tong, Limin Wang, and Gangshan Wu. Mgsampler: An explainable sampling strategy for video action recognition. In *IEEE/CVF International conference on Computer Vision (ICCV)*, pages 1513–1522, 2021.

A Functions invoked in Algorithm 1

Algorithm 2 InnerFree(farea,box)

Input: farea,box
Output: the rest free area in box

```
function InnerFree(farea, box)
    m,n,dp=box.w,box.h,List[m][n]
    Init(dp)
    for j in n do
        up,down,stk=List[m],List[m],Stack()
        for i in m do
            if stk is not None and left[stk.top()][j] >= left[i][j] then
                down[stk.pop()] = i
            if stk is not None then
                up[i] = stk.top()
            stk.push(i)
        for i in m do
            height = down[i] - up[i] - 1
            area = height * left[i][j]
            get the free area with max area
    return max free area
```

B Performance of Importance Level Approximation

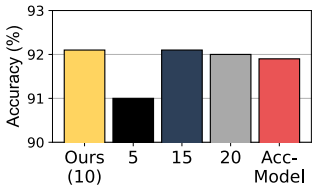


Figure 26: Importance level classification of MBs (*i.e.*, MB-grained image segmentation assigns each MB an importance level) yields comparable (even better) accuracy than AccModel, which predicts exact importance value.

Approximating the MB importance prediction as an image segmentation problem to assign each MB an importance level (class) is effective and produces good performance. We have trained four MB importance predictors (in MobileSeg architecture) with 5, 10, 15, and 20 important levels, respectively. Fig. 26 compares the final accuracy of inference task when predicting the exact MB importance (AccModel) and important levels. Experiment results demonstrate that important level classification yields precise MB importance prediction and hence high accuracy provided the level number is not very coarse, saying 5. The efficiency of this importance level classification is because such shallow model architecture is better at more manageable classification tasks (*e.g.*, segmentation) than regression tasks (*e.g.*, AccModel predicts the exact importance value). RegenHance set level number as 10.

C Additional Results

C.1 Example Eregions and the Distribution

Example eregions of object detection in §2.3 and semantic segmentation in Fig. 27 are generated by simply bounding the different analytical results between original and enhanced frames with a rectangle. Note that these example eregions are not the eregions used in RegenHance as they still contain much unnecessary content; this is why we propose to set fine-grained MB as the construction unit of eregions in RegenHance.

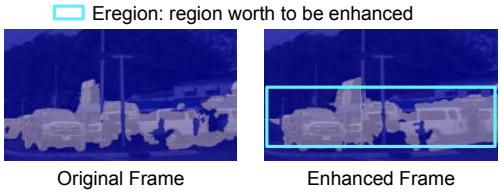


Figure 27: Example eregion worth to be enhanced for semantic segmentation (SS) bounding with a rectangle box.

In semantic segmentation, as illustrated in Fig. 28, only 10-15% area in 70% frames are eregions.

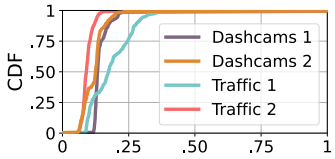
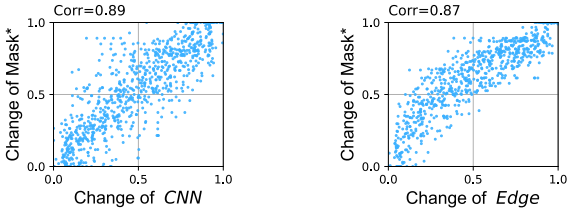


Figure 28: Distribution of eregions in Semantic segmentation.

C.2 Performance of Diverse Operators



(a) CNN [105] (b) Edge

Figure 29: Correlation ratio between the change of features.

As shown in Fig. 29, one-layer CNN [105] and the Edge operator (*i.e.*, edge detector in computer vision community) offer lower correlation ratios compared to the $\frac{1}{Area}$ operator, *i.e.*, 0.91. Fig. 30 exemplifies the characteristics of Area and $\frac{1}{Area}$. Images in the top raw demonstrate that the Area operator captures the change of large blocks, while the values of $\frac{1}{Area}$

change very small. On the contrary, $\frac{1}{Area}$ captures the change of small objects in two bottom images as RegenHance needs.

(a) $\frac{1}{Area}$: 0.002, Area: 0.665

(b) $\frac{1}{Area}$: 0.307, Area: 0.035

Figure 30: Correlation ratio between the change of features.

C.3 Performance *v.s.* Expanding Pixels

Pixel expanding in each direction can avoid the MB/region boundaries causing too many jagged edges and blocky artifacts when pasting enhanced content back to the bi-linear-interpolated frames. But it also introduces extra enhancement costs. We measure the accuracy gain and extra latency of the object detection task, as shown in Fig. 31. To balance the accuracy and the enhancement cost, we expand three pixels in this paper because the MB selection (§3.3.1) can control the number of enhanced MB to adjust the performance of RegenHance.

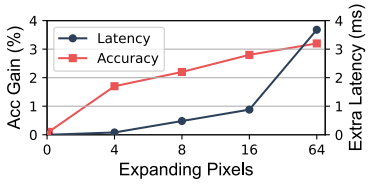


Figure 31: Both accuracy gain and enhancement cost increase with more expanding pixels. RegenHance expands 3 pixels in each direction surrounding every region.

C.4 Performance of MB Packing and Irregular Packing

Our region-aware packing algorithm spends almost the same time as MB packing and provides a close occupy ratio of irregular packing. As shown in Fig. 32, MB packing yields a low occupy ratio as packing too many unimportant or repeated pixels, while irregular packing leads to a large time cost of packing plan search.

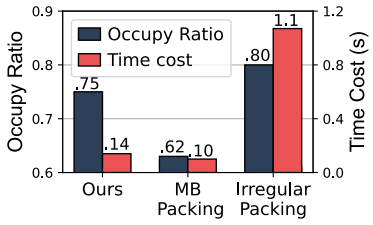


Figure 32: Our packing algorithm achieves a very good balance between bin utilization (occupy ratio) and the time cost of packing plan search.

C.5 An Example of Intermediate Results of RegenHance in Object Detection Task

Fig. 34 shows an example of packing five frames into one bin and pasting enhanced regions in the stitching image back to the interpolated frames. In particular, the region ④ is cut into two smaller ones by our region-aware bin-packing algorithm for better bin utilization. The green boxes in the left column bound the objects can be detected after enhancing the stitching image, *i.e.*, enhancing the selected regions. To the right column, region-based enhancement offers comparable accuracy with entire-frame enhancement.

C.6 Performance under Various Workloads and Users' Latency Targets

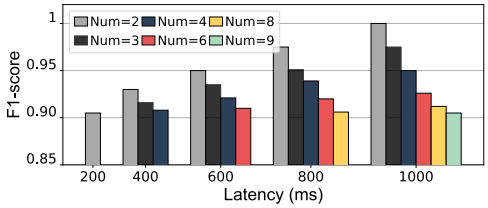


Figure 33: RegenHance supports various user-specified latency targets with adaptive batch sizes under different workloads (*i.e.*, stream numbers)

RegenHance supports various latency targets by automatically adjusting every component's batch size. Fig. 33 demonstrates that RegenHance can well analyze two 30-fps streams (60 frames) under 200ms latency budget and nine streams under 1s. The batch sizes of components change under different workloads. For example, under a 400ms budget, when stream numbers increase from 2 to 4, the batch sizes of (enhanced model, analytical model) are (8,4), (4,8), and (4,8); this implies more resources are allocated from region enhancement to final inference when workload increases, and hence lower accuracy. Note that the batch sizes of any components are no more than 8 among all latency targets; namely, in each batch, the earliest input won't wait for the last one more than 75ms as shown in Fig. 17.

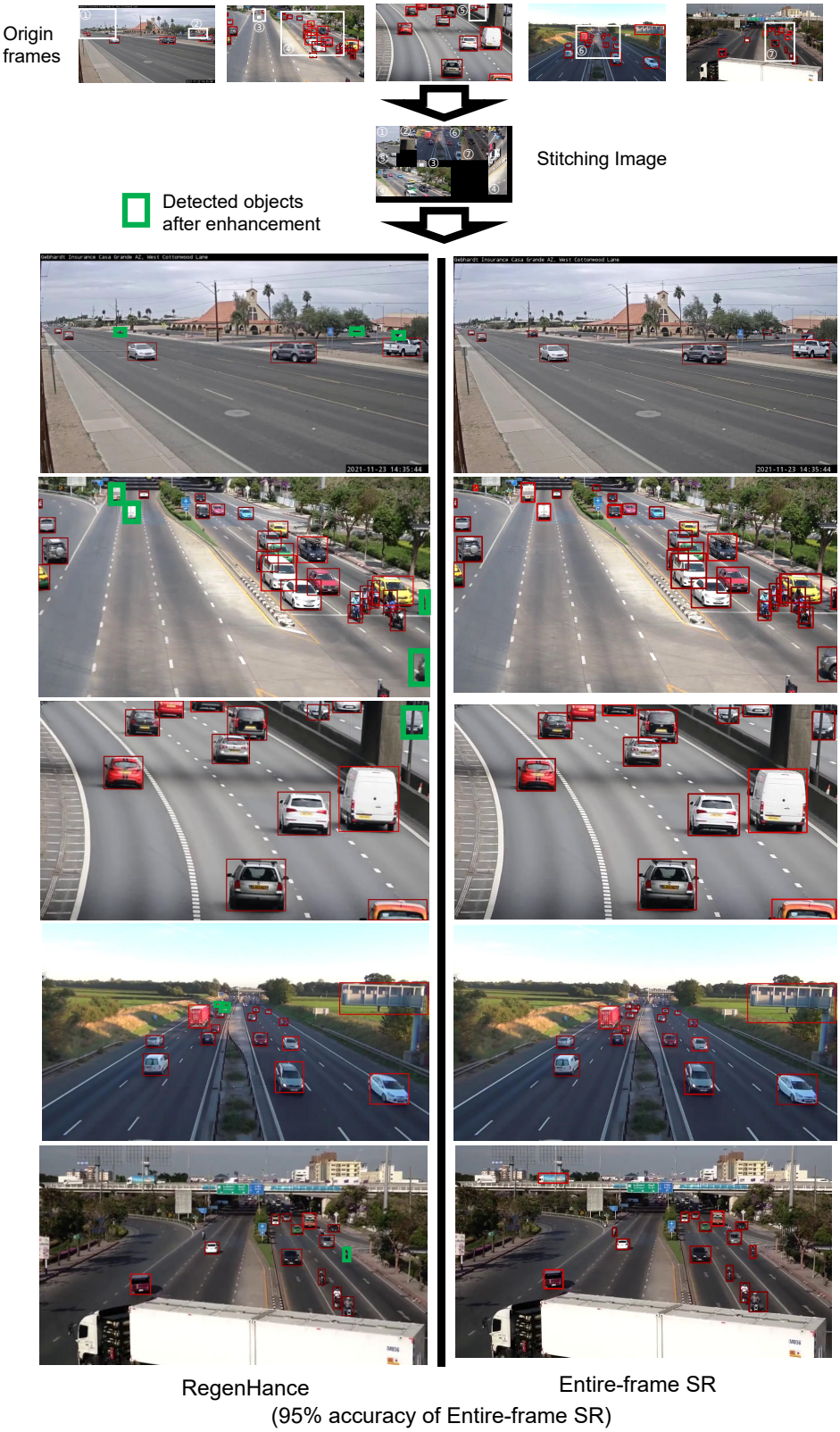


Figure 34: Examples of running super-resolution on stitched regions and their inference results compared to entire-frame enhancement.

Figure 35: Representative mask*.