TURBO: Opportunistic Enhancement for Edge Video Analytics

Yan Lu*, Shiqi Jiang†, Ting Cao†, Yuanchao Shu†

*New York University, †Microsoft Research

jasonengineer@hotmail.com,{shijiang,ting.cao,yuanchao.shu}@microsoft.com

ABSTRACT

Edge computing is being widely used for video analytics. To alleviate the inherent tension between accuracy and cost, various video analytics pipelines have been proposed to optimize the usage of GPU on edge nodes. Nonetheless, we find that GPU compute resources provisioned for edge nodes are commonly under-utilized due to video content variations, subsampling and filtering at different places of a video analytics pipeline. As opposed to model and pipeline optimization, in this work, we study the problem of opportunistic data enhancement using the non-deterministic and fragmented idle GPU resources. In specific, we propose a task-specific discrimination and enhancement module, and a model-aware adversarial training mechanism, providing a way to exploit idle resources to identify and transform pipeline-specific, low-quality images in an accurate and efficient manner. A multi-exit enhancement model structure and a resource-aware scheduler is further developed to make online enhancement decisions and fine-grained inference execution under latency and GPU resource constraints. Experiments across multiple video analytics pipelines and datasets reveal that our system boosts DNN object detection accuracy by 7.27 – 11.34% by judiciously allocating 15.81 – 37.67% idle resources on frames that tend to yield greater marginal benefits from enhancement.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; • **Computing methodologies** → **Computer vision**.

KEYWORDS

Edge Computing, Deep Neural Networks, Video Analytics, Object Detection, Opportunistic Enhancement

ACM Reference Format:

Yan Lu*, Shiqi Jiang†, Ting Cao†, Yuanchao Shu†. 2022. TURBO: Opportunistic Enhancement for Edge Video Analytics. In *ACM Conference on Embedded Networked Sensor Systems (SenSys '22), November 6–9, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3560905.3568501

SenSys '22, November 6–9, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9886-2/22/11…$15.00
https://doi.org/10.1145/3560905.3568501

1 INTRODUCTION

Video analytics has drawn a significant attention over the past couple years due to the growing presence of cameras and rapid developments on artificial intelligence. In order to preserve privacy and lower the total cost of ownership for video analytics, edge compute devices are predominantly used at customer's premises for video ingestion and processing [5, 7, 33, 36, 37, 42, 50, 55, 60].

Edge devices are known to be resource-constrained. Over the years, a considerable amount of literature has been published on the design and implementation of efficient edge video analytics systems. Examples include but are not limited to cascaded and adaptive analytics pipeline [2, 3, 16, 27, 56], multi-capacity neural networks [17, 18, 21], memory-efficient deep neural network (DNN) inference [4, 39], continuous learning of models [38, 45], low-cost analytics across cameras [26] and hierarchical clusters [23]. In contrast to the plethora of research on the optimizations of video analytics pipelines (VAPs) and DNNs, in this paper, we seek to answer the question that *given an optimized VAP*, how idle compute resources on edge, if present, can be harnessed to further improve the overall analytics accuracy.

The rationale behind this question is two-fold. First, by studying canonical VAPs on real-world datasets, we noticed that there is a decent amount of idle GPU compute resources on the edge due to video content changes and widely-used subsampling techniques [2, 3, 23, 27, 33]. For instance, in a cascaded analytics pipeline, heavyweight DNN is called upon only when a lightweight CPU-based background subtraction module detects motion in certain areas, resulting in less but video-dependent and fluctuating GPU usage. Similarly, a vehicle counting and recognition pipeline could generate much less DNN inference requests at times of low traffic volume. The same observation holds true to a wide range of VAPs given that edge machine is commonly used to process multiple camera streams and tends to be provisioned for scenarios of the worst-case workload. Second, we found that despite of decent overall accuracy provided by VAPs on target video inputs, there always exists a small portion of frames where VAPs perform poorly. This can be due to many reasons, including the low quality of the image (*e.g.,* occlusion, blur, low lighting), and the lack of representative training data for the DNN. Regardless of the cause, analytics accuracy could be largely improved from effective enhancements on such hard samples.

To reap the benefits of idle GPU resources and further improve the performance of an existing VAP, we introduce TURBO, an opportunistic enhancement framework which *selectively* enhances incoming frames based on GPU resource availability and characteristics of the DNN model used in a VAP. Design of TURBO faces three challenges. First, it is non-trivial to reliably and efficiently

identify frames that tend to yield inferior performance on downstream DNNs. The reason is simply because inference performance depends both on frame contents and on the DNN used in a VAP. For example, an object that is easily recognizable by a DNN detector could become ambiguous in a couple frames when lighting condition changes. Likewise, the definition of *hard* could vary significantly between a YOLOv3 [43] model pre-trained on COCO dataset [35] and a Faster-RCNN [44] model pre-trained on a private dataset. Second, it is technically challenging to improve the performance of an existing DNN on hard samples without sacrificing its accuracy on relatively easy ones. End-to-end model optimizations (*e.g.,* retraining or fine-tuning the entire model for hard samples) could lead to overfitting or bias, and is also prohibitively expensive in terms of both compute and annotation cost. At times model adaptation and retraining could even become infeasible when proprietary data, models, and techniques (*e.g.,* training oprimizations, third-party software, specialized accelerators) are used. Third, idle GPU resources from running VAPs are *non-deterministic* and *fragmented*. Hence, enhancement at runtime requires the awareness of resource availability as well as an elastic and fine-grained execution mechanism.

In Turbo, we tackle these challenges by making the following three contributions.

- We propose a task-specific discrimination and enhancement module based on generative adversarial networks (GAN). The module is trained by a novel model-aware adversarial training mechanism, which as a result, provides a *discriminator* that effectively identifies hard samples for a particular DNN, and a *generator* that makes image inputs more amenable to the downstream DNN in an efficient manner.
- We devise an enhancement execution module, achieved by an elastic structure design of the GAN model and a resource-aware scheduler, to best utilize the fragmented GPU compute resources. Specifically, the module maximizes the overall analytics accuracy by running a pre-trained multi-exit GAN model at different enhancement levels on selected frames under given latency and resource constraints.
- We fully implement our solution and evaluate it on two large-scale real-world video datesets. Results from three VAPs show that with the same computation hardware, Turbo improves average analytic accuracy by 9.02%, 11.34%, and 7.27% for three different detection models from harvesting idle resources. Turbo's code and datasets are available at: *aka.ms/turbo-project*.

In what follows, we use the object detection, a pivotal component in various video analytics systems, as a canonical application to motivate and describe the design of Turbo. Turbo can be easily extended to other kinds of heavyweight video DNN workloads as we only rely on DNN output and do not make any assumption on the inner workings of the model.

2 MOTIVATION AND BACKGROUND

In this section, we present the opportunities and challenges in the opportunistic enhancement for edge video analytics.

2.1 Edge Video Analytics Pipelines

Edge devices are being used increasingly for video analytics. Given the nature of limited compute and network resources, video analytics task typically uses a cascaded pipeline which consists of a series of modules on the decoded frames of the video stream. Fig. 1 demonstrates a VAP, where multiple cameras are connected to an edge node, and on it downstream modules like DNN-based object detection are performed. Before arriving at heavyweight DNNs, video frames are typically processed using techniques like temporal pruning (*e.g.,* sub-sampling based on pixel differences between frames), spatial pruning (*e.g.,* region cropping and background extracting), and model pruning (*e.g.,* gating networks and cascaded models) [57]. Such modules result in more efficient but dynamic GPU usage which are content-dependent.

To examine the performance of edge VAPs and their corresponding resource utilization in real deployments, we conduct a measurement study with two canonical pipelines, Glimpse [7] and Vigil [50]. Glimpse uses temporal pruning and sends frames to the downstream object detection model only when movements are detected between two frames. Vigil, on the other hand, adopts model pruning and sends out only images that contain objects detected by a lightweight local model. We execute these two pipelines on UA-DTRAC [52], a traffic video dataset with rich annotations. In all experiments, we use EfficientDet-D0 [49] as the object detection model, and process 4 video streams simultaneously on an Azure Stack Edge Pro [10] equipped with a Nvidia Tesla T4 GPU [14]. We use the Streaming Multiprocessor (SM) Activity reported by Nvidia DCGM [12] to characterize the GPU utilization. SM activity is defined as the fraction of time at least one warp was active on a SM, averaged over all SMs. It is a finer-grained metric than the GPU utilization number reported by *nvidia-smi*, which is the ratio of time the graphics engine is active.

Fig. 2 illustrates the actual workloads and the corresponding GPU utilization of a selected VAP. Due to pruning, only a portion of frames are eventually sent to GPU for processing. Overall, we observe that GPU throughput varies greatly over time, from 11 infer/sec to 78 infer/sec. In particular, for Glimpse, throughput goes beyond 50 infer/sec for only 7.26% of the time. Similarly, throughput of Vigil stays below 45 infer/sec for 19.03% of the time.

Not surprisingly, the GPU usage also fluctuates due to workload variations. Specifically, there appears more than 43% and 60% idle resources on average for Vigil and Glimpse, respectively. Furthermore, the appearances of idle resources are non-deterministic and fragmented since they are highly related to video contents. It is a fleeting opportunity to harvest the idle resources and in turn improve the analytic accuracy.

We also study the performance of different object detection models, including EfficientDet [49], Faster-RCNN [44] and YOLOv3 [43], on each individual frame of a selected trace from UA-DTRAC. In Fig. 3, we notice that the mean averaged precision (mAP) [35] varies dramatically over time. For Faster RCNN, while more than half of the frames (425) yield accuracy higher than 55.05%, due to low mAP scores on a small set of hard frames, the averaged mAP across all frames is only 52.72%. In fact, the averaged mAP of the bottom 5%

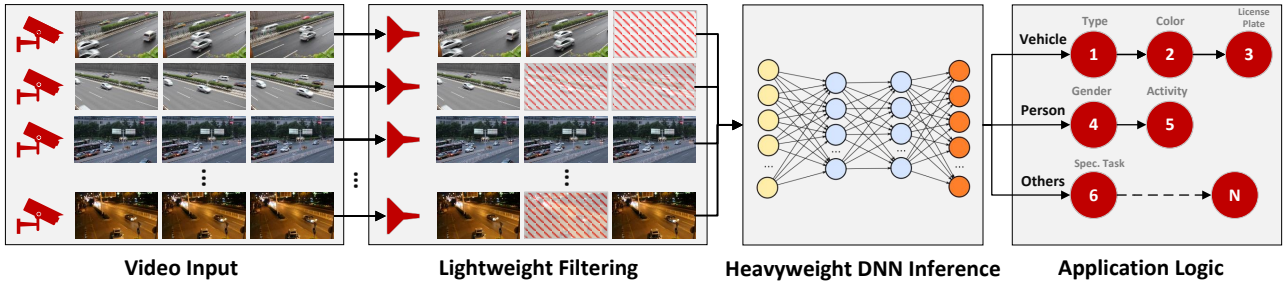
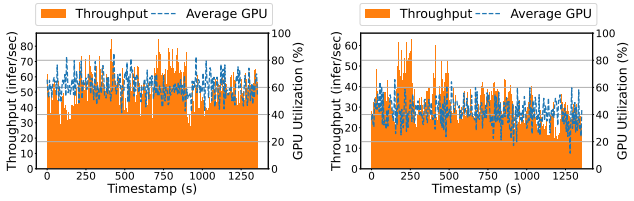


Figure 1: Illustration of a sample cascaded edge video analytics pipeline. In this paper, we use the object detection as an example of the heavyweight DNN tasks.



(a) Vigil (b) Glimpse

Figure 2: Dynamic video analytics workloads on an edge device shared by multiple streams.

of the frames (45) is as low as 37.13%. Similar results are observed on both beefier and wimpier models. For example, the average mAP of YOLOv3 can be increased by 10.63% if the mAP of the bottom 5% frames is improved by 15.51%.

2.2 Challenges of Opportunistic Enhancement

To improve the performance of a VAP on hard samples, one might employ a DNN model optimization approach by collecting all hard samples, annotating and using them to retrain or fine-tune the model. The method, however, falls short for two reasons. First, data collection and annotation process could be computationally expensive and the fragmented idle GPU resources makes the DNN model training challenging. Second, VAPs in real deployment might consist of black-box DNN models pre-trained on proprietary datasets. Without the details (*e.g.,* DNN architecture and weights), one can hardly update the model. In fact, even for a model well pre-trained and fine-tuned, there still exist hard samples and under-utilized compute resources given the nature of a pipeline and content variations. As such, we shift our focus to opportunistically enhancing input images of a video analytics system.

Image enhancement has been extensively studied in both computer vision and systems communities [20, 32, 34, 34, 41, 59, 61, 63, 64]. Enhancement methods, such as super resolution, deblurring and dehazing, look for ways to restore corrupted details in raw captured images. Intuitively, one can apply off-the-shelf image enhancements to improve the quality of frames for edge video analytics.

To this end, we select six state-of-the-art image enhancement methods, namely super resolution [34], dehaze [41], deblur [63], denoise [34], relight [20, 61] and derain [63], and apply them on

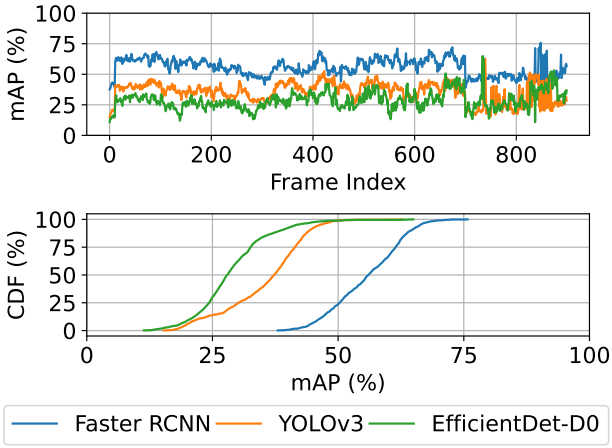


Figure 3: mAP of each individual frame from the selected video trace of UA-DTRAC using different models.

images of the selected video trace from UA-DTRAC. Similarly, we use EfficientDet as our detection model. We compare the mAP on each individual raw and enhanced image, and present the results in Fig. 4.

As can be seen, none of selected image enhancement methods unanimously improves the quality of image and makes all frame samples easier for the detector. In fact, enhanced images lead to worse detection accuracy on some easy samples. The rationale behind that is general purpose image enhancements are usually designed for human visual perception and trained on manually labelled dataset. In real deployments, however, the cause of hard samples with respect to a particular downstream task can be far more complex. For instance, environmental changes (*e.g.,* clouds, glare) could result in the drastic lighting condition change within a few seconds. Other common factors include object movements and changes of object sizes. For example, a car gets harder to be detected when it suddenly accelerates or moves away from the camera. In summary, reconstructed details from a single or general purpose image enhancement methods are not sufficiently discriminative for the heavyweight DNN model used in a specific VAP.

Key Takeaways:

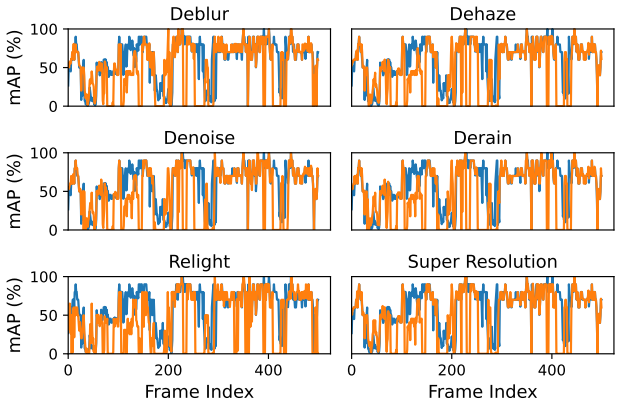


Figure 4: mAP on every raw *(blue)* and enhanced *(orange)* image of a selected video trace.

- We see a considerable amount of idle GPU compute resource exists in different edge VAPs. However, idle resource availability is highly dynamic and fragmented.
- Overall accuracy of a video analytics task can be boosted significantly when the quality of a small portion of frames improves. However, such hard samples are non-uniformly distributed in the time domain and are hard to predict.
- Running off-the-shelf opportunistic enhancement methods, *i.e.,* image enhancement in a naive way is inappropriate. It is expensive and could even adversely impact object detection performance. On the contrary, a fine-grained, task-specific enhancement is desired.

In what follows, we propose Turbo, a GAN-based task-specific enhancement model and an online execution scheduler which gracefully select and transform hard frames by harvesting the highly dynamic idle GPU resources.

3 DESIGN OVERVIEW

In Turbo, we design three key modules, namely discriminator, enhancer, and scheduler. Fig. 5 demonstrates the architecture overview of Turbo, which employs opportunistic enhancement in two phases. In the offline phase, we make attempts to train a discriminator and an enhancer (*i.e.,* the Generator in Fig. 5), which are tailored for the downstream detector. The trained discriminator is thus able to classify if an incoming frame can be well detected. For those hard frames, the trained enhancer provides additional processing which introduces more discriminative details to make the frame more amenable to the detector. Turbo also provides multiple enhancement levels. In the online phase, we inject the trained discriminator and enhancer into the VAP, without modifying any other existing modules. The resource-aware scheduler buffers incoming frames and selectively executes the enhancer at different levels within the resource budget, so as to achieving the best overall detection accuracy.

To train the discriminator and the enhancer jointly in the offline phase, Turbo's enhancement module builds on top of recent advances in generative adversarial networks (GAN). Unlike traditional general purpose GAN-based image enhancements, Turbo proposes a task-specific GAN architecture and a model-aware adversarial training mechanism. This GAN aims to improve semantic details for the downstream tasks instead of improving the interpretability or perception of information in images for human viewers.

To enable fine-grained enhancement at runtime, we devise a multi-exit GAN structure and an adaptive scheduler to decide how the multi-exit GAN is executed on frames so as to maximize the overall object detection accuracy. Due to video content variations and subsampling of the VAP, heavyweight object detection workloads vary over time and would likely under-utilize the GPU resources provisioned upfront for most of the time. Thus, Turbo's scheduler firstly determines the resource budget by quantifying the number of frames reaching the object detector, and then uses the discriminator in GAN to classify frames. Based on the classified difficulties and resource budget, a combinatorial optimization problem is formulated to decide at what level the enhancement model is executed on incoming frames.

4 ADVERSARIAL LEARNING-BASED ENHANCEMENT

GAN [19, 25, 30, 66] is widely used for the image enhancement and synthesis. GAN adopts *adversarial training* techniques [19] to learn a *generator* (G) and a *discriminator* (D) simultaneously.

In GAN-based image enhancement, G is responsible for generating synthetic high-quality images, whereas D takes as input both synthetic and real high-quality images, and is trained to distinguish between these two sources. Since G and D play a competing and continuous game, in which G is learning to produce more and more realistic high-quality images, and D is learning to be better and better at distinguishing synthetic data from real data, GAN-based image enhancement gains both generative and discriminative abilities at the end of training [34, 47, 53, 58, 61]. Such ability well serves the purposes of identifying and transforming hard frame samples to ones that are more amenable to the object detector in a video analytics pipeline.

Despite superior accuracy, GAN is known to be hard to use in reality [54]. The reason is two-fold. First, *general purpose, large capacity* GAN model training is challenging and prone to mode collapse, non-convergence and instability due to inappropriate design of network architecture, misuse of objective function and optimization algorithms [9, 34, 41, 61, 63]. Second, prohibitively high inference cost hinders real-world deployment of GAN. Specifically, GAN's generator typically uses a encoder-decoder architecture where latent features are firstly extracted by encoder and then processed by stacked up-sampling layers in decoder for semantic details recovery. Running G in a naive way on frames in a VAP could incur a prohibitively high latency.

To accelerate the inference pipeline and enable fast adaptation on new testing data, we propose a *detector-specific* GAN architecture and a *model-aware* adversarial training mechanism (Fig. 6). As opposed to building a general purpose model, our GAN architecture is tailored for the object detection model used in a video analytics pipeline. This way, we effectively reduce the complexity of GAN

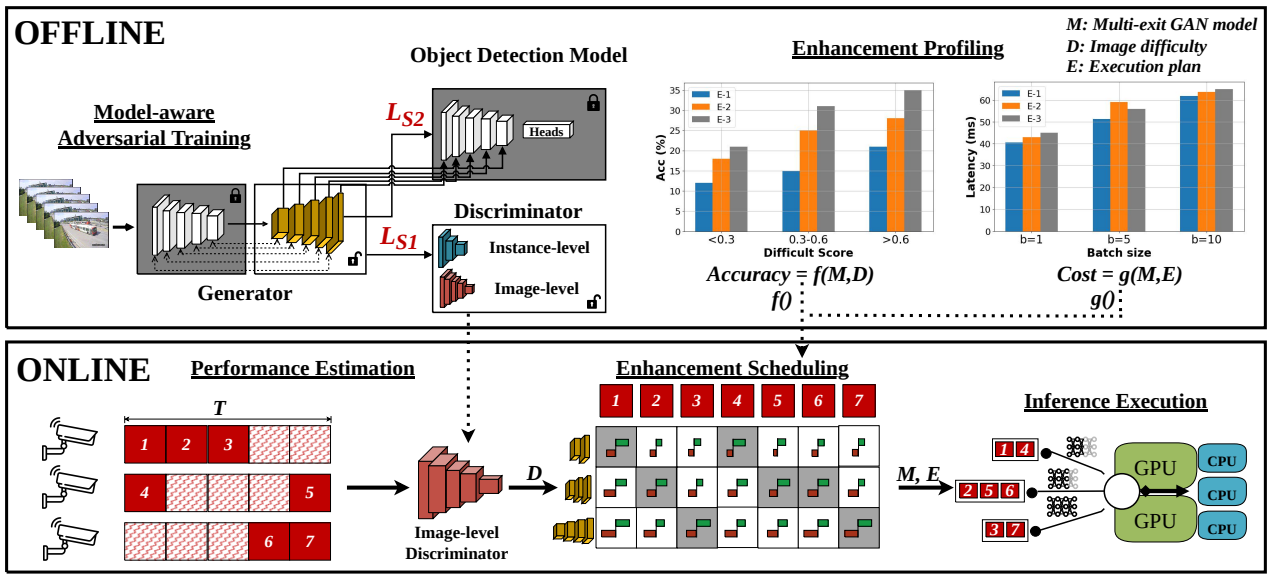


Figure 5: TURBO system overview.

training by learning a general G on a similar public dataset and fine-tuning its discriminator only on testing data. To further reduce training complexity and cost, we replace the encoder in G with backbone layers of the detector since the latter has already been trained for extracting feature embedding. In addition, we introduce a new multi-exit structure between two models which provides more flexibility and fine-grained trade-off between inference latency and accuracy.

4.1 GAN Architecture

We follow the common practice [19, 25, 30, 61, 66] to design the overall architecture of GAN. However, two key changes are introduced to make our model more suitable for opportunistic enhancement in video analytics.

First, inspired by [61], we design our GAN consisting of one single G and two Ds, a frame-level D_f and an instance-level D_i. D_f is applied on the whole frame to examine if it is hard or not, whereas D_i is applied onto each individual object instance in one frame. The insight behind this design is that one frame could contain hard instances of various types, thus need to be treated differently from each other. For the architecture of D_i and D_f, we use two and one convolutional layers with three fully connected layers for D_i and D_f, respectively.

Second, we reuse the backbone of the downstream DNN in G. We design the G as a U-Net [46] architecture, which contains an encoder and a decoder. For a hard frame, the encoder firstly extracts the feature maps, after which the decoder synthesizes an easier frame with more discriminative features from that. Intuitively, the encoder plays an exact role of the backbone network in a detector. Hence, we directly replace the encoder with detector's backbone including its weights. Such a design brings two benefits, compared to training G from scratch: 1) enhancements based on feature maps extracted by detector's backbone yields higher object detection accuracy. 2) training only the decoder would make G learn faster as it can focus on reconstructing semantic details instead of learning a feature extractor and a reconstructor together. In a U-Net, a skip connection between the encoder and the decoder is used to reconstruct semantic details better. In specific, the decoder layer leverages a corresponding extracted feature embedding from the encoder layer to improve semantic details. To implement this idea, we set the shape of the decoder layer output to be the same as the corresponding feature embedding extracted from the backbone.

4.2 Two-stage Model Training

We propose a two-stage training process, as illustrated in Fig. 6. In stage one, we try to train a GAN, and in the stage two, we empower the GAN with the multi-exit capability to fit the dynamic idle compute resources. We begin with the stage one.

To train the GAN, a dataset containing training frames is required. We would discuss the training dataset selection in §4.4. Here given a set of frames, we firstly identify easy and hard samples using the downstream object detector. Based on the observations that DNNs always perform uncertainly on hard samples, we leverage detector's predicted confidence score as an indicator. Specifically, for frame F, we calculate its difficulty score θ_F by averaging the confidence scores of all its Region-of-Interests (RoIs),

$$\theta_F = \frac{1}{N}\sum_{i=0}^{N}\sigma_i, \tag{1}$$

where σ_i is the confident score of the i_{th} RoI in the frame F.

We select frames with θ_F lower than a threshold, empirically set to 0.6, as hard samples, and the remaining as easy ones. By using the selected hard and easy samples, we train the frame-level discriminator D_f and the instance-level discriminator D_i. We update their weights via the back-propagation from the following loss functions,

$$\begin{aligned} L_f &= \mathbb{E}_{x\sim p_e(x)}[logD_f(x)], \\ L_i &= \mathbb{E}_{x\sim p_e(x)}[logD_i(x)], \end{aligned} \tag{2}$$

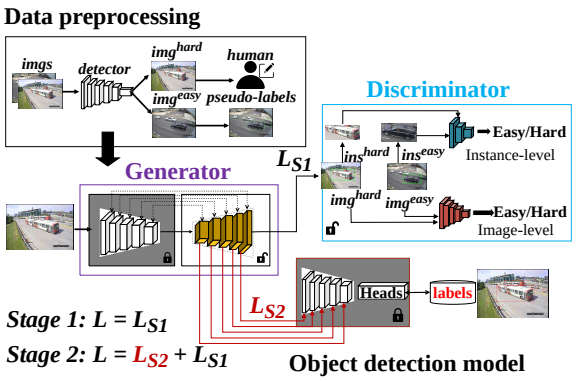


Figure 6: Two-stage training strategy in Turbo. Stage one trains the generator and discriminators whereas stage two fine-tunes the generator with the multi-exit mechanism.

where L_f and L_i are the loss functions for D_f and D_i, respectively. p_e denotes the data distribution mixed by real and synthetic easy frames. Note that we do not need the annotations of frames to train the discriminators.

Once D_f and D_i are trained, we use them to train the generator G. The goal is to make G take as the input a hard frame, and output a *synthetic* easy frame, which D is not able to distinguish from the real easy ones. In particular, we make use of the following adversarial loss in the stage one,

$$L_{S1} = L_f + \mathbb{E}_{z\sim p_h(z)}[log(1 - D_f(G(z))] + L_i + \mathbb{E}_{z\sim p_h(z)}[log(1 - D_i(G(z))] \quad (3)$$

where p_h represents the distribution of hard frame. We then run the adversarial training process, update G and Ds in the alternating and competing manner, until reaching the convergence.

4.3 Multi-exit Structure

After the GAN is trained, we connect the output of the decoder with the detector in stage two. Next, we fine-tune the GAN using the annotations of frames without modifying the detector. More specifically, we go through all the frames. For each frame, we run the generator followed by the detector. According to the detection results and the frame annotations (ground truth), we calculate the detection loss L_d. Based on that we fine-tune and update the decoder of the GAN.

As discussed in §3, we propose to enable the elastic execution for the enhancement module, to best utilize the dynamic idle resources. To this end, we introduce a multi-exit structure. Such a design is due to two rationales: 1) by skipping a different number of layers, the enhancement module would result in multiple accuracy-latency profiles, which are helpful for a flexible scheduling; 2) Even for hard frames, they have various difficulty scores. Some of them might be relatively amenable to the detector, a lightweight enhancement is totally enough for a better detection.

Specifically, we add connections between the feature maps of the κ_{th} layer of the decoder and the $(\beta - \kappa)_{th}$ layer of the detector's backbone, as shown in Fig. 6, where β is the total layer number of the backbone. The connections naturally work because we reuse the backbone as the encoder and design the decoder accordingly. As a result, the shape of the connected layers is the same. Furthermore, in our design, not only $(\beta - \kappa)$ layers in the decoder can be skipped, κ layers in the detector's backbone can be skipped as well.

Next, we fine-tune the GAN with the multi-exit structure. As we expect all exits to be effective for the final detection results, we compute the detection loss for each exit by comparing its predictions and the annotations, and then average the detection losses of all exits. In this stage, since G is updated, we add L_{S1} to the training process as an regularization term. In sum, the overall loss of the stage two L_{S2} is formulated as,

$$L_{S2} = \frac{1}{\beta}\sum_{\kappa=0}^{B} L_d^{\kappa} + L_{S1} \quad (4)$$

where κ denotes the exit's index, L_d is a standard loss of the object detection, and β is the layer number of the backbone.

The multi-exit structure offers several enhancement options with different levels of discriminative characteristics, processing latency and accuracy gain. In §5, we propose an adaptive scheduler to take advantage of this feature.

4.4 Pre-training Dataset Selection

In the offline phase of Turbo, we pre-train the GAN-based enhancement module, which requires the labelled training data. Intuitively, the performance of Turbo is highly related to the dataset selection.

A straightforward way to build the training dataset is to collect historical frames from the target scenes, and label them either manually or by a golden model. This approach, however, falls short on efficiency in real deployments. Collecting and labeling a number of frames require mass of human efforts, which could largely impair the practicality.

To this end, we introduce an alternative dataset selection strategy. We propose to use public datasets that contains rich annotations on scenes similar to the target environment. For example, if the deployment focuses on traffic analytics (*e.g.*, vehicle counting), the pre-train can be done using BDD100K [62], which contains more than 100 million annotated frames captured from driving cars. This strategy turned out to work extremely well as in contrast to limited types of hard samples in the target scene, the pre-training process benefits way more from the much larger variety of samples in massive public datasets. In the evaluation (§6.2), we would show more details about the accuracy gains of the dataset selection strategy.

5 ADAPTIVE ENHANCEMENT SCHEDULING

To take full advantage of idle resources, we design the adaptive scheduler, which makes the online decisions of applying the most suitable enhancement levels for the incoming frames, maximizing overall detection accuracy within the resource availability. To achieve this, we firstly profile the enhancement module in terms of the latency cost and the accuracy gain, in the offline phase. Then the profilings as well as the pre-trained GAN are deployed on the target edge device. Based on the profilings, in the online phase, we

conduct the scheduling. We formulate the scheduling as an optimization problem, and we introduce the heuristic solution to solve it.

5.1 Enhancement Profiling

Thanks to the multi-exit design (§4.3), Turbo provides multiple enhancement levels. Such the diversity makes more rooms for the enhancement scheduling. In order to determine the best enhancement level, we need to get a sense of the latency-accuracy trade-off of executing the pre-trained GAN at different exits.

Profiling the latency of the multi-exit GAN is simple. we run different levels of the enhancement module using various batch sizes in an exhaustive manner. We execute 100 runs and make use of the averaged latency. Finally We obtain I_κ, which stands for the expected inference latency when executing the κ_{th} level enhancement,

$$I_\kappa = \mu_D + \varepsilon_\kappa + \nu_\kappa, \tag{5}$$

where μ_D is the inference latency of the frame-level discriminator D_f (§4.1), which is a constant. ε_κ donates the latency of the generator G (§4.1) when κ layers are used. ν_κ donates the latency of the downstream detector when κ layers are skipped.

Since we can execute the enhancement with the same level at batch, thus we notate I^n_κ where n is the batch size. Note that $\kappa = 0$ represents no enhancement would be applied, and $I_0 = \mu_D + \nu_0$.

To profile the accuracy gains of the the multi-exit GAN, we bucketize the frames from the training dataset based on their difficulty scores θ according to Equation 1. We set the bucket granularity to 0.1. Then we execute the base detector on every frame, obtain the accuracy without the enhancement, *i.e.,* mAP. Next we execute each enhancement level of the GAN on every frame, and obtain the corresponding mAP improvement compared to the base detector. Finally we average the mAP of frames in the same bucket, and obtain P^θ_κ, which stands for the expected accuracy gains when applying the κ_{th} level enhancement on a frame with the difficulty of θ. Note that $\kappa = 0$ represents no enhancement would be applied, and $P_0 = 0$.

Note that the profiling is a one-time effort and we put in the offline phase.

5.2 Heuristic-based Optimization

With the latency profile I^n_κ and the accuracy profile P^θ_κ, we formulate the scheduling as an optimization problem. Given M frames streamed from multiple cameras, they are required to be processed within T, which is the latency constraint of the VAP. Due to the filtering modules, only m frames need to be processed where $m < M$. The scheduling purpose is to generate an enhancement plan. In particular, for each frame $x, x \in m$, we determine a κ, to maximum the total accuracy gains of all the m frames, while the total latency is not beyond the constraint T. Specifically we formulate it as,

$$\max \quad \sum P^{\theta'_x}_\kappa, x \in m, \tag{6}$$

$$s.t. \quad f(\sum I_\kappa) \le T, \tag{7}$$

where θ'_x is the estimated difficulty score of the frame x, using the frame-level discriminator D_f. The function $f(\cdot)$ is to organize the frames assigned by the same enhancement level to execute in a batch.

Suppose there are m frames and β enhancement levels in total. Then the search space of the optimization problem would be β^m. Particularly, this problem is one kind of non-linear generalized assignment problem (GAP) [15]. It is known to be NP-hard. Though the search space κ might be limited in real deployments (4-5), it still brings the scheduling overhead. To this end, in Turbo we devise sub-optimal solution with two heuristics: 1) the pre-trained multi-exit GAN has a monotonic characteristic that the higher enhancement level is involved, the more accuracy improvement is achieved (see §6.3). 2) Applying enhancement on the hardest frames tends to yield higher marginal accuracy gains.

As such, we follow a prune-and-search approach in three steps. 1) We assign all the m frames with the maximum κ. 2) Since this enhancement plan would most likely violate latency constraint T, so we select the frame that has the minimal marginal accuracy gain, and assign $\kappa - 1$. 3) We repeat the prior step until the T is met. Once the enhancement plan is determined, we execute each frame according to the plan.

6 EVALUATION

In this section, we present evaluation results of Turbo with three canonical VAPs on two real-world video datesets.

6.1 Experimental Setup

Video Analytics Pipeline (VAP). Various kinds of video analytics pipelines have been developed to strike the balance between inference accuracy and compute/network cost [2, 7, 22, 27, 29, 33, 50, 57]. In our experiments, we adopt three canonical cascaded VAPs with the object detection as the downstream task.

(1) **Glimpse** [7] measures inter-frame difference, and sends only frames that contain new objects to the object detector down the pipeline.

(2) **Vigil** [50] runs a lightweight object detection model firstly, and sends only frames that contain most objects to the edge cluster for the heavyweight DNN inference.

(3) **NoScope** [29] identifies frames with significant pixel changes, and runs a cheap DNN to select frames with low confidence, then calls the edge-based heavyweight object detection.

Dataset. We evaluate Turbo on two traffic video datasets, UA-DETRAC [52] and AICity [51]. The videos are captured by the surveillance cameras on streets. UA-DETRAC consists of 10 hours of videos captured at 24 locations with 25 FPS and the resolution of 960×540. AICity contains more than 3 hours of videos at 10 FPS with resolution of at least 1920×1080. UA-DETRAC includes three types of vehicles annotations, including car, bus and van. AICity consists of two types of vehicles annotations, which are car and van. In total there are 128, 800 frames and 1, 103, 160 annotations for our evaluation.

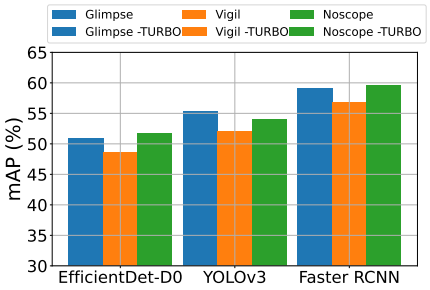


Figure 7: Overall mAP on UA-DETRAC of the selected VAPs and the enhanced VAPs by Turbo, with different object detectors, on T4 GPU.

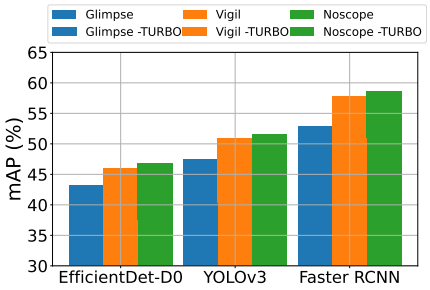


Figure 8: Overall mAP on AICity of the selected VAPs and the enhanced VAPs by Turbo, with different object detectors, on T4 GPU.

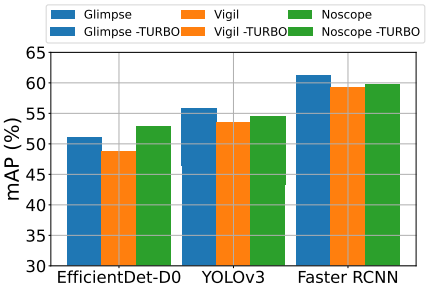


Figure 9: Overall mAP on UA-DETRAC of the selected VAPs and the enhanced VAPs by Turbo, with different object detectors, on V100 GPU.

Targeted Detector	EfficientDet-D0	YOLOv3	Faster-RCNN
# Param. of D	136,555	663,147	2,764,907
# Param. of G	9,142,073	30,251,419	37,568,837
# Hard Samples	1, 017, 875	1, 387, 283	827, 583
# Easy Samples	10, 982, 125	10, 612, 717	11, 172, 417

Table 1: Training details of the GANs for EfficientDet-D0, YOLOv3 and Faster RCNN on BDD100K dataset.

In our experiments, we also use BDD100K [62] to pre-train our GAN enhancement module. BDD100K contains 100K driving videos collected from more than 50K rides. The videos are not from the surveillance cameras, but the targets are vehicles as well. Therefore we use it as the pre-training dataset to verify our designs in §4.4.

Object Detection Models. We train and evaluate our GAN enhancement module upon three popular DNN object detectors, including YOLOv3 [43], Faster RCNN [44] and EfficientDet [49]. All of models are pre-trained on COCO dataset [35].

Metrics. We use the mAP to measure the analytics accuracy of the selected VAPs. We use the streaming multiprocessors (SM) utilization [13], a fine-grained metric for GPU utilization, to quantify the compute resource utilization.

Test platforms. We make use of two platforms, the Azure Stack Edge Pro [10] with a NVIDIA Tesla T4 GPU [14], and a virtual machine equipped with a NVIDIA Tesla V100 GPU. T4 has 320 Tensor cores and 16GB GPU memory, V100 GPU has 640 Tensor cores and 16GB GPU memory.

Implementation. We build and train the models in Turbo with TensorFlow [1], and execute the inference using NVIDIA Triton Inference server [11]. GPU SM utilization is collected using NVIDIA DCGM [12]. We implement an application to simulate video streams from multiple cameras over HTTP. In our evaluation, we feed four video streams to an edge node.

To train the GAN on BDD100K, we firstly use the particular pre-trained detectors to pick hard samples as discussed in §4.2. For instance, we get 1, 017, 875 hard samples and 10, 982, 125 easy samples for EfficientDet-D0. We then train the discriminator, followed by the generator. We set the training epoch to 200. Finally, we fine-tune the generator with the multi-exit structure for an extra 100 epochs. More training settings for EfficientDet-D0, YOLOv3 and Faster-RCNN are listed in Table 1. Source code and deployment instructions are also available at: *aka.ms/turbo-project*.

6.2 End-to-End Evaluation

Fig. 7 illustrates the overall performance of Turbo on UA-DETRAC with T4 GPU, compared to the selected VAPs integrated with three DNNs. Overall, Turbo improves the absolute mAP by 9.02%, 11.34% and 7.27% on average for EfficientDet-D0, YOLOv3 and Faster RCNN across the selected VAPS. Specifically, for Vigil integrated with YOLOv3, Turbo can bring about 9.35% mAP improvement. Without the enhancement, the Glimpse with Faster RCNN achieves 53.42% mAP, and Turbo can bring 5.70% more mAP to the VAP.

The averaged mAP improvements of the VAPs, Glimpse, Vigil and NoScope across different detectors are 8.32%, 9.20% and 8.66%, respectively. Turbo achieves the higher mAP improvements in Vigil and NoScope. It is because that VAPs with model-based pruning are prone to feed more hard frames, so are benefiting from Turbo much more than temporally filtered frames from Glimpse.

Performance over different datasets. In addition to UA-DETRAC, we also evaluate Turbo on AICity dataset as well. Shown in Fig. 8, on the AICity dataset, Turbo performs even better than on UA-DETRAC. For instance, with and without the Turbo enhancement, Glimpse integrated with YOLOv3 could obtain 38.71% and 47.32% mAP, respectively. It is because that the pre-trained detectors have poor performance on AICity, while Turbo significantly enhances these imperfect detectors. Overall the averaged mAP improvements of Glimpse, Vigil and NoScope across different detetectors on AICity are 12.46%, 9.71% and 8.42%, respectively.

Performance over different devices. Different devices might expose different idle resources. To show the effectiveness of Turbo on other computing devices, we evaluate Turbo on UA-DETRAC dataset with V100. As shown in Fig. 9, Turbo achieves a higher mAP improvement than T4 (Fig. 7) for three object detectors. Especially on Faster RCNN, Turbo boosts 4.45% absolute mAP on average. It is because that idle resources on V100 is much more than T4.

Performance over different throughout. To understand how Turbo harvest the idle resources in fine-grained ways, we introduce the performance over different throughout. As described in §2.1, throughput is the real arriving rate of video streams, it represents the number of frames the VAP need to process per second. According to our evaluation, the maximum processing capacity of T4 for YOLOv3 is 84 inference per second (infer/sec). Fig. 10 illustrates the obtained mAP of the enhanced VAPs under different throughout. Turbo perform significantly well when the throughout is low since more idle resources are harvested to enhance the VAP. Particularly, 13.75% absolute mAP improvement can be obtained for all of VAPs when the throughput is lower than 21 infer/sec.

Performance over more baselines. In addition to the raw VAP, we also introduce two more baselines, which are potentially used as the opportunistic enhancement approaches, the image enhancement and the model switching.

For the baseline of image enhancement, we replace the generator of the GAN with the image enhancement models. We use three popular image enhancement models, including deblurring [63], dehazing [41] and super resolution [34]. We also add an oracle image enhancement, which would select the best image enhancement model for each frame. Fig. 11 shows the evaluation results of Turbo compared to the baseline of image enhancements. Although the oracle enhancement improves 3.34% average mAP, it is still much lower than Turbo, which gets 13.51% mAP improvement.

For the baseline of model switching, we runs our scheduling algorithm on a model zoo including, EfficientDet-D0, D4 and D7, to use the larger model to harvest the idle resources. Fig. 12 shows the evaluation results of Turbo compared to the model switching baselines. Turbo achieves more than 12.42% mAP than the best model switching baseline.

GPU utilization and overhead. Turbo harvests the idle GPU resources to enhance the VAP. Therefore we evaluate the GPU utilization. As shown in Fig. 13, Turbo successfully utilizes more idle GPU resources, resulting in the higher GPU utilization rate. Particularly, Turbo brings 25.42% more GPU utilization on the T4 GPU.

Specifically, the discriminator cost around $2ms$ on T4, and the full generator would only cost $27.31ms$. As the reference, YOLOv3 would cost $35.71ms$ per inference on the T4 GPU.

6.3 Evaluation of Multi-Exit GAN

Next we break down Turbo and evaluate key components in detail.

Effectiveness of the pre-training strategy. To show effectiveness of our data selection on GAN pre-training, we show mAP of our GAN pre-trained on two datasets, shown in Fig. 14. In specific, we train our GAN for EfficientDet-D0 on the target dataset and a public dataset separately. In our experiments, we select BDD100K [62] as the public dataset to train GAN.

It is interesting to note that GAN pre-training on target datasets are hard to preserve mAP improvements for all detection models. But pre-training GAN on a large-scale public dataset achieves a stable mAP improvement on different models or datasets. It is because that target datasets cannot provide enough hard samples for training Turbo's GAN. For instance, there are only 960 hard samples on the target scenes (AICity) and 65,540 hard samples on BDD100K for Faster RCNN.

Effectiveness of the model-aware adversarial training. We propose a model-aware adversarial training, therefore we compare with a model-agnostic adversarial training strategy. It selects frames with more small objects as hard samples, since the scale-variant is one of challenges for locating objects. Objects with small scales are hard to be annotated in existing large-scale labeled training datasets (*e.g.,* Microsoft COCO [35], UA-DETRAC [52] and MIO-TCD [65]).

We set the same initial model architectures and parameters for both training strategies. Model-agnostic training fine-tunes all layers end-to-end instead of freezing backbone layers. As in Fig. 15, model-aware training only requires 1/3 time of model-agnostic training but achieves 15%, 18% and 22% more mAP improvements for EfficientDet-D0, YOLOv3 and Faster RCNN.

Effective pre-trained frame-level discriminator With an effective multi-exit GAN, Turbo needs a accurate and robust frame-level discriminator to predict difficult-scores for a new input frame. To examine how the pre-trained frame-level discriminator perform on the unseen data, we firstly group frames by its predicted difficult-scores and use ground-truth to get real distributions of hard samples in each group.

As shown in Fig. 16, we observe that almost all easy frames (the 1^{st} group) are classified correctly and more than 80% hard frames in the last group are assigned to correct labels by our pre-trained frame-level discriminator. Although $15\% - 18\%$ easy frames of the last group are assigned to hard samples, the number of frames in UA-DETRAC is small (about 150) and they are randomly distributed in $56,340$ testing frames.

Effectiveness of the multi-exit GAN. Using deeper layers of a good multi-exit G on hard images should achieve higher accuracy, and our heuristic search is based on this monotonic characteristic. To verify it, we run the pre-trained multi-exit GAN on all hard frames and record overall mAP improvements for each exiting layer. We notice that the mAP can be improved monotonically via using a deeper enhancement layer, as shown in Fig. 17.

Since a deeper enhancement layer adds more semantic details of input frames, it makes the detector improve performance on the false negatives (missing objects). But on the false positives (incorrect predicted bounding boxes), image enhancement based techniques might be not working, because optimized detectors are very confident on their predicted outputs and its outputs follow a U-shaped distribution. Many bounding boxes are assigned either low (< 0.2) or high (> 0.8) confidences. To decrease the false positives, predictions with the high confidence should be refined, which requires the retraining on the labeled data. Thus, continuous training might be a new opportunity for Turbo to improve the accuracy in further.

6.4 Evaluation of Adaptive Enhancement Scheduling

In the end, we evaluate our scheduler. Because our solution is based on the heuristic idea and achieves sub-optimal trade-off between

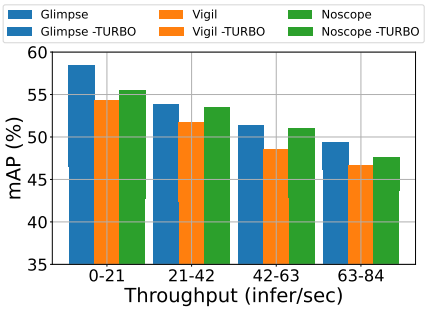


Figure 10: mAP of the selected VAPs and the enhanced VAPs by Turbo with YOLOv3 under different throughput, using T4 GPU on UA-DTRAC.

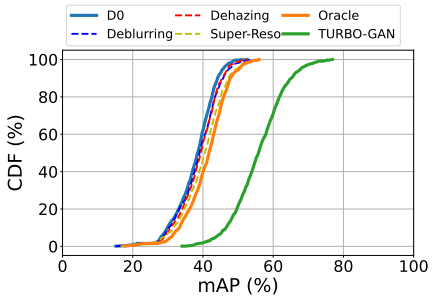


Figure 11: The mAP distribution of frames in UA-DETRAC obtained by the Turbo, compared to different image enhancement baselines.

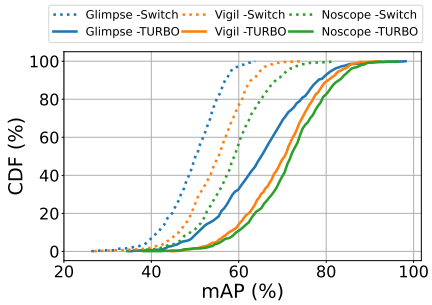
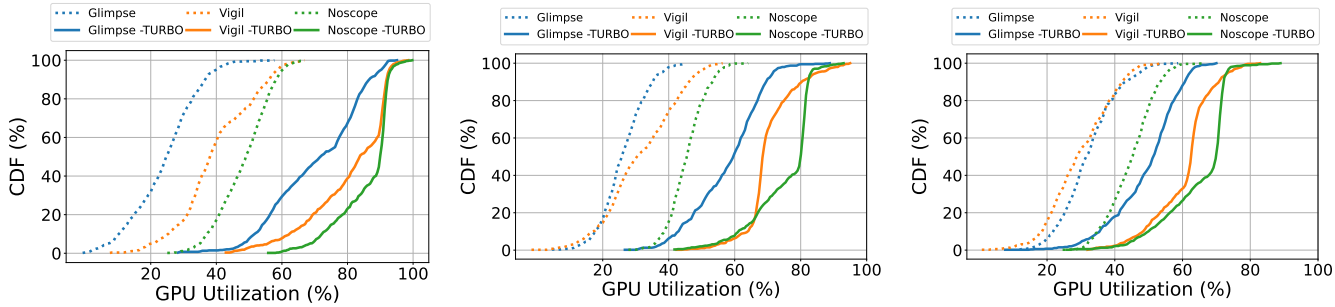


Figure 12: The mAP distribution of frames in UA-DETRAC obtained by the Turbo, compared to different model switching baselines.



(a) EfficientDet-D0 (b) YOLOv3 (c) Faster RCNN

Figure 13: GPU utilization of Turbo with the selected VAPs integrated with different object detectors on T4.

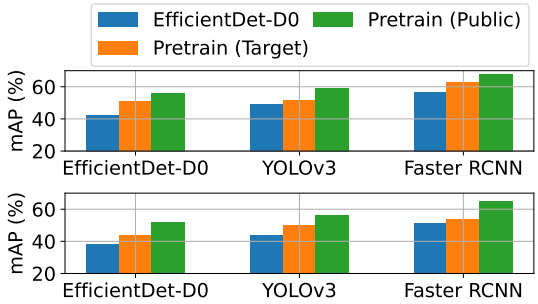


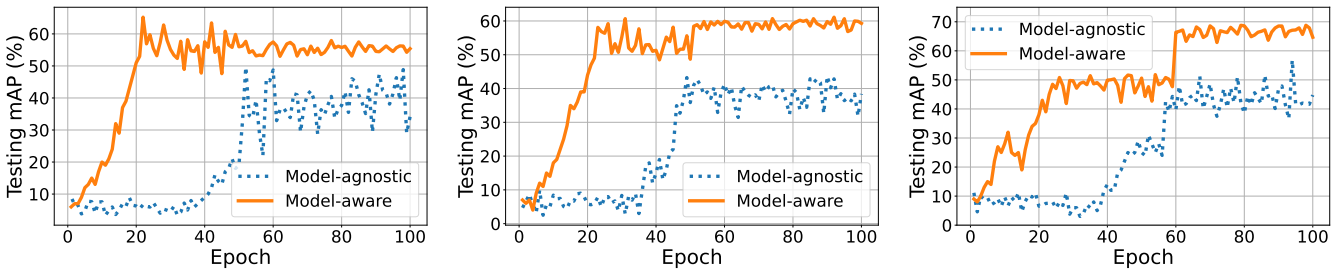
Figure 14: Achieved mAP of Turbo pre-trained on target dataset and Public dataset.

the accuracy and the latency, we compare Turbo with a brute-force search method, notated as *upper*. As shown in Fig. 18, Turbo is lower than the upper-bound method by average 3.26%, 1.95% and 3.63% mAP for EfficientDet-D0, YOLOv3 and Faster RCNN respectively. To measure searching cost on different scheduling methods, we present the distribution of execution time for three detectors on Glimpse, Vigil and Noscope in Fig. 18(d), Fig. 18(e) and Fig. 18(f) respectively. We find that even though a brute-force search can achieve the best accuracy but its latency of searching is much larger than Turbo. Thus, our scheduling algorithm is an efficient and effective method because it can find a enough good solution within about 20*ms*.

7 RELATED WORK

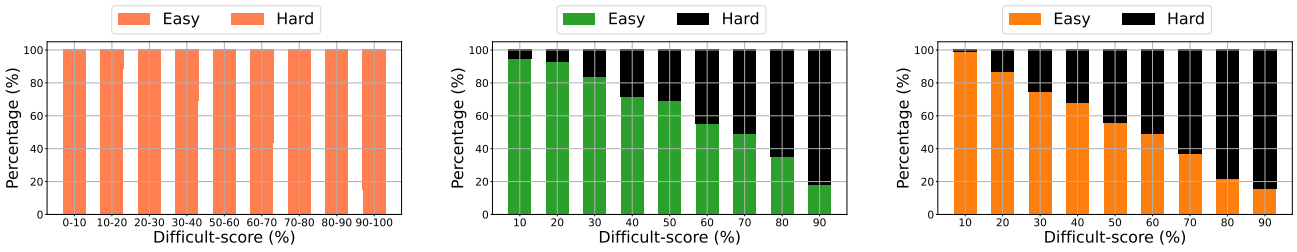
Edge Video Analytics Pipelines. Edge video analytics systems have been widely deployed and have became the solution to many large-scale safety and management tasks. Most systems [6–8, 22, 23, 27, 29, 33, 36, 48, 50] adopt an edge-cloud architecture [57] where camera or far edge nodes are responsible for processing simple tasks (*e.g.,* video compression and temporal filtering) and network edge servers and cloud maintain a deep neural network to provide accurate analysis on input videos [5, 37]. However, inference accuracy is often limited by the unstable network links and resource-constraint edge servers.

To balance the inference accuracy and resource (compute/network) cost, many cascade video analytics pipelines are proposed. They leverage video processing heuristics to design pruning methods in order to save compute and network costs. The first category is based on temporal consistency of videos and seeks to skip processing on similar frames by examining inter-frame differences [6, 7, 22, 23, 27, 29, 33, 36, 50]. Although these methods are effective to reduce network cost, a cheap tracking model is required to be deployed



(a) EfficientDet-D0 (b) YOLOv3 (c) Faster RCNN

Figure 15: The test mAP of each training epoch using model-aware and model-agnostic training strategies.



(a) EfficientDet-D0 (b) YOLOv3 (c) Faster RCNN

Figure 16: An analysis on the pre-trained image-level discriminator on unseen frames.

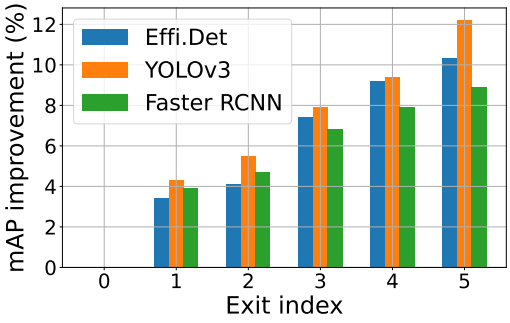
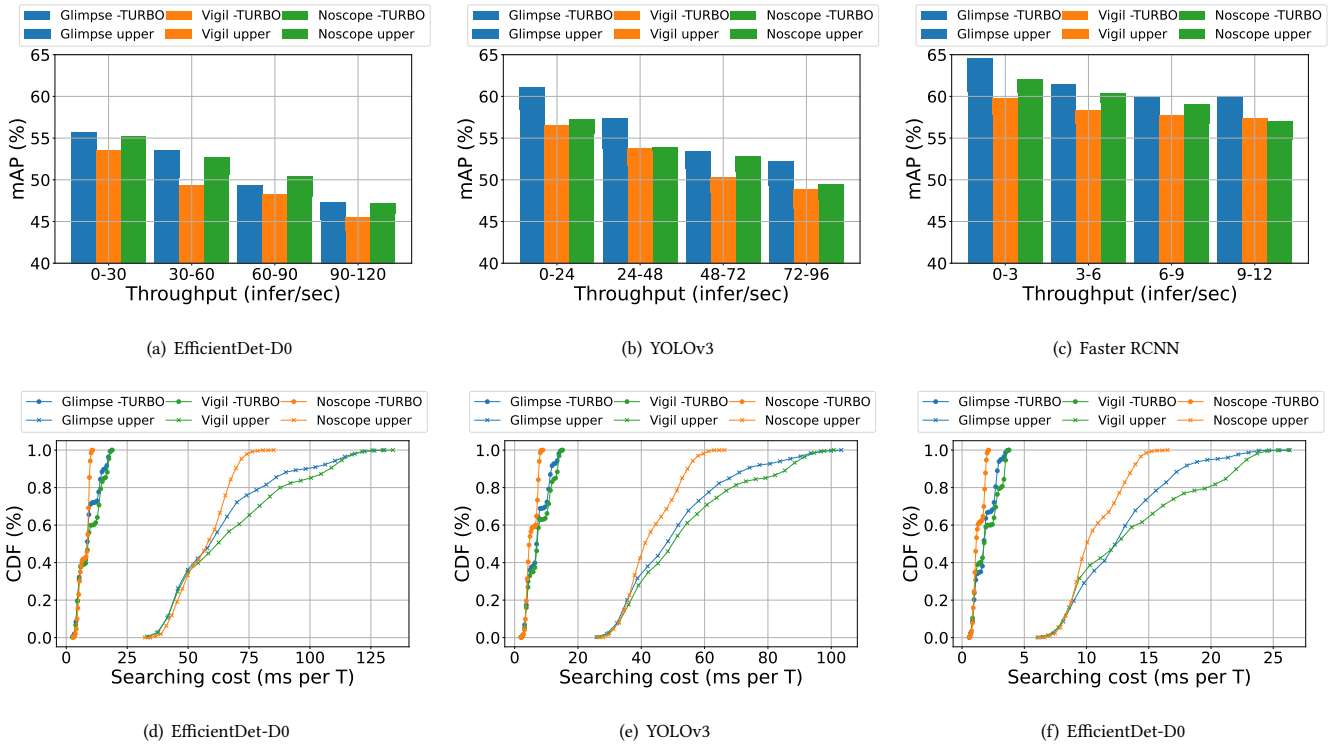


Figure 17: Achieved mAP improvement of each exit point.

on camera or far edge for inference on filtered frames. To further reduce communication cost, spatial [6, 22, 27, 50] and model [22, 27, 29, 48] pruning methods are proposed in VAPs. Both methods need to deploy a cheap deep neural network on edge devices and use it to select region-of-interests (RoIs) or uncertain frames for server's inference. In practice, three pruning methods are often used together (*e.g.*, Noscope [29] integrates temporal and model pruning to filter video frames). Based on this cascade design, network edge servers only need to process few frames in a video. However, a over-provisional compute resource is often assigned because it can meet all requests' requirements. Thus, how to leverage this existing dynamic idle compute resources to improve inference is ignored by current video analytics systems. To bridge this gap, we design Turbo to enhance hard samples in an adaptive manner for higher overall accuracy.

Besides the pruning methods, advanced VAPs are proposed with model switching or model mixture techniques, *e.g.*, Chameleon [27] and Remix [28]. They better utilize the compute resources and improve the analytics accuracy despite the existence of idle resources. In addition, such VAPs' performance highly depends on the DNN models in the pipeline. For each model, its hard samples are still not well handled. Hence, Turbo is complementary to these advanced VAPs.

Image Enhancement. Image enhancement is a well-studied problem in low-level computer vision (CV) tasks and is also named image restoration [31]. In many benchmarks [24, 40], they are explicit grouped by the corresponding image noise: image deblurring [63], image deraining [63], image denosing [34], image dehazing [41], relight [20, 61] and super-resolution [34]. It aims to restore raw images from images mixed by noises and is always seen as a data prepossessing step for downstream CV tasks. Because almost vision models (*e.g.*, YOLOv3 [43] and EfficientDet-v0 [49]) are pre-trained on cleaned images (*e.g.*, Microsoft COCO [35]), their accuracy are easy to be degraded by natural image noise. For example, vehicle detectors are often degraded by image blurring when cars speed up. Besides, existing benchmarks only provide labeled training data for a single noise instead of mixed noise. Thus, we need to download different pre-trained models for processing different image noise. But in real world applications, an video may contain more than two or three noises. For instance, traffic videos collected in a rush hour may contain blurring and low-light cases. Thus, using them in VAPs is not easy because it requires VAPs select suitable models on any frames. Fortunately, we find that adversarial training can provide



(a) EfficientDet-D0 (b) YOLOv3 (c) Faster RCNN

(d) EfficientDet-D0 (e) YOLOv3 (f) EfficientDet-D0

Figure 18: Accuracy gains and search costs of the Turbo scheduler, compared with the brute-force searching based scheduler (denoted as *upper*).

not only image generator but also image discriminator, which is highly matching VAP's need on frames' selection. Thus, we choose to integrate existing VAPs with a GAN-based image enhancement model [61].

8 DISCUSSION AND FUTURE WORKS

In this section, we discuss several design considerations in Turbo.

The generality of Turbo. Though Turbo works when idle compute resources appear on edge, the idle resources are common in real deployments, as revealed in §2. The reason is simply because the compute power on edge should always be provisioned for the peak workload, which however could not be reached for the majority of running time, due to the applied sub-sampling techniques, in turn making rooms for Turbo. Furthermore, since Turbo is a plug-and-play solution without modifying the existing models, we could easily switch off Turbo if the workload keeps high.

The overhead of Turbo. Two extra modules are introduced to the VAP by Turbo, a discriminator and an enhancer. The discrimination overhead cannot be ignored. Nonetheless, we make attempts to design a light-weighted but effective discrimination module. According to our measurement, the discriminator in Turbo costs less than 2*ms* on Nvidia Tesla T4. The enhancer, on the other hand, is scheduled to execute using the idle compute resources, thus we argue the overhead of the enhancer is negligible.

More advanced VAPs. We propose Turbo to improve the inference accuracy via executing a GAN-based enhancement module by harvesting the dynamic idle compute resource for edge VAPs. We notice that the achieved accuracy gain highly depends on the filtering methods. For example, the model-based filtering tends to bring higher accuracy improvements than temporal filtering-based VAPs. Thus, an interesting solution is to use adaptive hyper-parameters to adjust the filtering rate and the accuracy gain together. To scale Turbo to spatial filtering-based VAPs, we need to develop a region-level GAN to enhance semantic details in relevant RoIs. In the future, we will put more efforts to scale Turbo to more VAPs with advanced filtering modules.

9 CONCLUSION

In this paper, we propose Turbo, an opportunistic image enhancement framework which takes advantages of the over-provisioned GPU resources at runtime to improve the overall video analytics accuracy. Turbo first designs a task-specific GAN and trains it with the model-aware adversarial training strategy. Such a method allows the GAN to intelligently identifies model-specific hard samples and applies enhancements at various granularity. At runtime, an enhancement execution scheduler is developed to assign the most suitable enhancement level to each image to achieve the best overall accuracy within a given resource availability. We evaluate Turbo on a real-world traffic video dataset with three canonical video analytics pipelines. Turbo improves the absolute mAP by 9.02%, 11.34% and 7.27% on average for EfficientDet-D0, YOLOv3 and Faster RCNN, respectively.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. (2016). arXiv:cs.CV/1603.04467

[2] Ganesh Ananthanarayanan, Victor Bahl, Landon Cox, Alex Crown, Shadi Nogbahi, and Yuanchao Shu. 2019. Demo: Video Analytics - Killer App for Edge Computing. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*. ACM, New York, NY, USA, 695–696.

[3] Ganesh Ananthanarayanan, Yuanchao Shu, Mustafa Kasap, Avi Kewalramani, Milan Gada, and Victor Bahl. 2020. Live Video Analytics with Microsoft Rocket for reducing edge compute costs. https://www.microsoft.com/en-us/research/publication/live-video-analytics-with-microsoft-rocket-for-reducing-edge-compute-costs/. (2020).

[4] Padmanabhan Arthi, Agarwal Neil, Iyer Anand, Ananthanarayanan Ganesh, Shu Yuanchao, Karianakis Nikolaos, Xu Guoqing Harry, and Netravali Ravi. 2023. GEMEL: Model Merging for Memory-Efficient, Real-Time Video Analytics at the Edge. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, Boston, MA, USA.

[5] Victor Bahl. 2022. Microsoft and AT&T demonstrate 5G-powered video analytics. (2022). https://azure.microsoft.com/en-us/blog/microsoft-and-att-demonstrate-5gpowered-video-analytics/

[6] Zhang Ben, Jin Xin, Ratnasamy Sylvia, Wawrzynek John, and Lee Edward A. 2018. AWStream: Adaptive Wide-Area Streaming Analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM)*. Association for Computing Machinery, New York, NY, USA, 236–252.

[7] Tiffany Yu-Han Chen, Lenin Ravindranath, Shuo Deng, Paramvir Bahl, and Hari Balakrishnan. 2015. Glimpse: Continuous, Real-Time Object Recognition on Mobile Devices. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys)*. Association for Computing Machinery, New York, NY, USA, 155–168.

[8] Pakha Chrisma, Chowdhery Aakanksha, and Jiang Junchen. 2018. Reinventing Video Streaming for Distributed Vision Analytics. In *Proceedings of the 10th USENIX Conference on Hot Topics in Cloud Computing (HotCloud)*. USENIX Association, Boston, MA, USA, 1.

[9] Guo Chunle, Li Chongyi, Guo Jichang, Loy Chen Change, Hou Junhui, Kwong Sam, and Cong Runmin. 2020. Zero-reference deep curve estimation for low-light image enhancement. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE, Seattle, WA, USA, 1780–1789.

[10] Microsoft Corporation. 2022. Azure Stack Edge documentation. (2022). https://docs.microsoft.com/en-us/azure/databox-online/

[11] Nvidia Corporation. 2019. Triton Inference Server. https://github.com/triton-inference-server/server. (2019).

[12] Nvidia Corporation. 2021. NVIDIA Data Center GPU Manager. https://github.com/NVIDIA/DCGM. (2021).

[13] Nvidia Corporation. 2022. CUDA Tookit Documentation. https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html. (2022).

[14] NVIDIA Corporation. 2022. NVIDIA T4. (2022). https://www.nvidia.com/en-us/data-center/tesla-t4/

[15] Claudia D'Ambrosio, Silvano Martello, and Michele Monaci. 2020. Lower and upper bounds for the non-linear generalized assignment problem. *Computers & Operations Research* 120 (2020), 104933.

[16] Shohei Enomoto and Takeharu Eda. 2021. Learning to Cascade: Confidence Calibration for Improving the Accuracy and Computational Cost of Cascade Inference Systems. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press, Virtual, 7331–7339.

[17] Biyi Fang, Xiao Zeng, Faen Zhang, Hui Xu, and Mi Zhang. 2020. FlexDNN: Input-Adaptive On-Device Deep Learning for Efficient Mobile Vision. In *IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, Virtual, 84–95.

[18] Biyi Fang, Xiao Zeng, and Mi Zhang. 2018. NestDNN: Resource-Aware Multi-Tenant On-Device Deep Learning for Continuous Mobile Vision. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom)*. ACM, New York, NY, USA, 115–127.

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger (Eds.), Vol. 27. Curran Associates, Inc., Montreal, Quebec, Canada.

[20] Chunle Guo, Chongyi Li, Jichang Guo, Chen Change Loy, Junhui Hou, Sam Kwong, and Runmin Cong. 2020. Zero-Reference Deep Curve Estimation for Low-Light Image Enhancement. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Seattle, WA, USA, 1780–1789.

[21] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. 2022. Dynamic Neural Networks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 44, 11 (2022), 7436–7456.

[22] Zhang Haoyu, Ananthanarayanan Ganesh, Bodík Peter, Philipose Matthai, Bahl Victor, and Freedman Michael. 2017. Live Video Analytics at Scale with Approximation and Delay-Tolerance. In *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation (NSDI)*. USENIX Association, Boston, MA, USA, 377–392.

[23] Chien-Chun Hung, Ganesh Ananthanarayanan, Peter Bodik, Leana Golubchik, Minlan Yu, Paramvir Bahl, and Matthai Philipose. 2018. Videoedge: Processing camera streams using hierarchical clusters. In *IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, New York, NY, USA, 115–131.

[24] Andrey Ignatov, Radu Timofte, et al. 2018. PIRM challenge on perceptual image enhancement on smartphones: report. In *European Conference on Computer Vision (ECCV) Workshops*. Springer, Munich, Germany, 0–0.

[25] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE, Honolulu, HI, USA, 1125–1134.

[26] Samvit Jain, Xun Zhang, Yuhao Zhou, Ganesh Ananthanarayanan, Junchen Jiang, Yuanchao Shu, Paramvir Bahl, and Joseph Gonzalez. 2020. Spatula: Efficient cross-camera video analytics on large camera networks. In *IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, New York, NY, USA, 110–124.

[27] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. 2018. Chameleon: Scalable Adaptation of Video Analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM)*. Association for Computing Machinery, New York, NY, USA, 253–266.

[28] Shiqi Jiang, Zhiqi Lin, Yuanchun Li, Yuanchao Shu, and Yunxin Liu. 2021. Flexible High-Resolution Object Detection on Edge Devices with Tunable Latency. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking (MobiCom '21)*. 559–572.

[29] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. 2017. NoScope: Optimizing Neural Network Queries over Video at Scale. *Proc. VLDB Endow.* 10, 11 (2017), 1586–1597.

[30] Tero Karras, Samuli Laine, and Timo Aila. 2019. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*. IEEE, Long Beach, CA, 4401–4410.

[31] Ravneet Kaur and Er. Navdeep Singh. 2014. Image Restoration - A Survey. *IOSR Journal of Computer Engineering* 16 (2014), 107–111.

[32] Royson Lee, Stylianos I. Venieris, Lukasz Dudziak, Sourav Bhattacharya, and Nicholas D. Lane. 2019. MobiSR: Efficient On-Device Super-Resolution through Heterogeneous Mobile Processors. In *The 25th Annual International Conference On Mobile Computing And Networking (MobiCom)*. ACM, Los Cabos, Mexico, Article 54, 16 pages.

[33] Yuanqi Li, Arthi Padmanabhan, Pengzhan Zhao, Yufei Wang, Guoqing Harry Xu, and Ravi Netravali. 2020. Reducto: On-Camera Filtering for Resource-Efficient Real-Time Video Analytics. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*. Association for Computing Machinery, New York, NY, USA, 359–376.

[34] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. 2021. SwinIR: Image Restoration Using Swin Transformer. In *IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*. IEEE, Virtual, 1833–1844.

[35] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollar. 2014. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision (ECCV)*. Springer, Zurich, Switzerland, 740–755.

[36] Liu Luyang, Li Hongyu, and Gruteser Marco. 2019. Edge Assisted Real-Time Object Detection for Mobile Augmented Reality. In *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom)*. ACM, New York, NY, USA, Article 25, 16 pages.

[37] Nick McQuire. 2022. How developers can benefit from the new 5G paradigm. (2022). https://azure.microsoft.com/en-us/blog/how-developers-can-benefit-from-the-new-5g-paradigm/

[38] Khani Mehrdad, Ananthanarayanan Ganesh, Hsieh Kevin, Jiang Junchen, Netravali Ravi, Shu Yuanchao, Alizadeh Mohammad, and Bahl Paramvir. 2023. RECL: Responsive Resource-Efficient Continuous Learning for Video Analytics. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, Boston, MA, USA.

[39] Arthi Padmanabhan, Anand Padmanabha Iyer, Ganesh Ananthanarayanan, Yuanchao Shu, Nikolaos Karianakis, Guoqing Harry Xu, and Ravi Netravali. 2021. Towards Memory-Efficient Inference in Edge Video Analytics. In *Proceedings of the 3rd ACM Workshop on Hot Topics in Video Analytics and Intelligent Edges (HotEdgeVideo)*. ACM, New York, NY, USA, 31–37.

[40] Young Peter, Lai Alice, Hodosh Micah, and Hockenmaier Julia. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics* 2 (2014), 67–78.

[41] Xu Qin, Zhilin Wang, Yuanchao Bai, Xiaodong Xie, and Huizhu Jia. 2020. FFA-Net: Feature fusion attention network for single image dehazing. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press, New York, NY, USA, 11908–11915.

[42] Xukan Ran, Haolianz Chen, Xiaodan Zhu, Zhenming Liu, and Jiasi Chen. 2018. DeepDecision: A Mobile Deep Learning Framework for Edge Video Analytics. In *IEEE Conference on Computer Communications (INFOCOM)*. IEEE, Honolulu, HI, USA, 1421–1429.

[43] Joseph Redmon and Ali Farhadi. 2018. YOLOv3: An Incremental Improvement. (2018). arXiv:cs.CV/1804.02767

[44] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., Montreal, Quebec, Canada.

[45] Bhardwaj Romil, Xia Zhengxu, Ananthanarayanan Ganesh, Jiang Junchen, Shu Yuanchao, Karianakis Nikolaos, Hsieh Kevin, Bahl Paramvir, and Stoica Ion. 2022. Ekya: Continuous learning of Video analytics models on Edge compute servers. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, RENTON, WA, USA, 119–135.

[46] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi (Eds.). Springer, Cham, 234–241.

[47] Li Ruoteng, Cheong Loong-Fah, and Tan Robby T. 2019. Heavy Rain Image Restoration: Integrating Physics Model and Conditional Adversarial Learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE, Long Beach, CA, USA, 1633–1642.

[48] Haichen Shen, Seungyeop Han, Matthai Philipose, and Arvind Krishnamurthy. 2017. Fast Video Classification via Adaptive Cascading of Deep Models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Honolulu, HI, USA, 3646–3654.

[49] Mingxing Tan, Ruoming Pang, and Quoc V. Le. 2020. EfficientDet: Scalable and Efficient Object Detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE, Seattle, WA, USA, 10781–10790.

[50] Zhang Tan, Chowdhery Aakanksha, Bahl Paramvir (Victor), Jamieson Kyle, and Banerjee Suman. 2015. The Design and Implementation of a Wireless Video Surveillance System. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking (MobiCom)*. ACM, New York, NY, USA, 426–438.

[51] Zheng Tang, Milind Naphade, Ming-Yu Liu, Xiaodong Yang, Stan Birchfield, Shuo Wang, Ratnesh Kumar, David Anastasiu, and Jenq-Neng Hwang. 2019. CityFlow: A City-Scale Benchmark for Multi-Target Multi-Camera Vehicle Tracking and Re-Identification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Long Beach, CA, USA, 8797–8806.

[52] Longyin Wen, Dawei Du, Zhaowei Cai, Zhen Lei, MingChing Chang, Honggang Qi, Jongwoo Lim, MingHsuan Yang, and Siwei Lyu. 2020. UA-DETRAC: A New Benchmark and Protocol for Multi-Object Detection and Tracking. *Computer Vision and Image Understanding* 193 (2020), 102907.

[53] Du Wenchao, Chen Hu, and Yang Hongyu. 2020. Learning Invariant Representation for Unsupervised Image Restoration. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE, Seattle, WA, USA, 14483–14492.

[54] Lilian Weng. 2019. From gan to wgan. (2019). arXiv:cs.CV/1904.08994

[55] Hao Wu, Xuejin Tian, Minghao Li, Yunxin Liu, Ganesh Ananthanarayanan, Fengyuan Xu, and Sheng Zhong. 2021. PECAM: Privacy-Enhanced Video Streaming and Analytics via Securely-Reversible Transformation. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking (MobiCom)*. ACM, New York, NY, USA, 229–241.

[56] Zeng Xiao, Fang Biyi, Shen Haichen, and Zhang Mi. 2020. Distream: Scaling Live Video Analytics with Workload-Adaptive Distributed Edge Intelligence. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems (SenSys)*. ACM, New York, NY, USA, 409–421.

[57] Zhujun Xiao, Zhengxu Xia, Haitao Zheng, Ben Y. Zhao, and Junchen Jiang. 2021. Towards Performance Clarity of Edge Video Analytics. In *IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, San Jose, CA, USA, 148–164.

[58] Pan Xingang, Zhan Xiaohang, Dai Bo, Lin Dahua, Loy Chen Change, and Luo Ping. 2022. Exploiting Deep Generative Prior for Versatile Image Restoration and Manipulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 44, 11 (2022), 7474–7489.

[59] Juheon Yi, Sunghyun Choi, and Youngki Lee. 2020. EagleEye: Wearable Camera-Based Person Identification in Crowded Urban Spaces. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking (MobiCom)*. ACM, New York, NY, USA, Article 4, 14 pages.

[60] Shanhe Yi, Zijiang Hao, Qingyang Zhang, Quan Zhang, Weisong Shi, and Qun Li. 2017. LAVEA: latency-aware video analytics on edge computing platform. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing (SEC)*. IEEE, New York, NY, USA, Article 15, 13 pages.

[61] Jiang Yifan, Gong Xinyu, Liu Ding, Cheng Yu, Fang Chen, Shen Xiaohui, Yang Jianchao, Zhou Pan, and Wang Zhangyang. 2021. EnlightenGAN: Deep light enhancement without paired supervision. *IEEE Transactions on Image Processing (TIP)* 30 (2021), 2340–2349.

[62] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. 2020. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*. IEEE, Seattle, WA, USA, 2636–2645.

[63] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. 2021. Multi-Stage Progressive Image Restoration. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Virtual, 14821–14831.

[64] Wuyang Zhang, Zhezhi He, Luyang Liu, Zhenhua Jia, Yunxin Liu, Marco Gruteser, Dipankar Raychaudhuri, and Yanyong Zhang. 2021. Elf: Accelerate High-Resolution Mobile Deep Vision with Content-Aware Parallel Offloading. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking (MobiCom)*. ACM, New York, NY, USA, 201–214.

[65] Luo Zhiming, Branchaud-Charron Frédéric, Lemaire Carl, Konrad Janusz, Li Shaozi, Mishra Akshaya, Achkar Andrew, Eichel Justin, and Jodoin Pierre-Marc. 2018. MIO-TCD: A New Benchmark Dataset for Vehicle Classification and Localization. *IEEE Transactions on Image Processing (TIP)* 27, 10 (2018), 5129–5141.

[66] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision (ICCV)*. IEEE, Venice, Italy, 2223–2232.