AdaEnlight: Energy-aware Low-light Video Stream Enhancement on Mobile Devices

SICONG LIU, Northwestern Polytechnical University, China
XIAOCHEN LI, Northwestern Polytechnical University, China
ZIMU ZHOU, City University of Hong Kong, China
BIN GUO, Northwestern Polytechnical University, China
MENG ZHANG, Northwestern Polytechnical University, China
HAOCHENG SHEN, Northwestern Polytechnical University, China
ZHIWEN YU, Northwestern Polytechnical University, China

The ubiquity of camera-embedded devices and the advances in deep learning have stimulated various intelligent mobile video applications. These applications often demand on-device processing of video streams to deliver real-time, high-quality services for privacy and robustness concerns. However, the performance of these applications is constrained by the raw video streams, which tend to be taken with small-aperture cameras of ubiquitous mobile platforms in dim light. Despite extensive low-light video enhancement solutions, they are unfit for deployment to mobile devices due to their complex models and and ignorance of system dynamics like energy budgets. In this paper, we propose AdaEnlight, an energy-aware low-light video stream enhancement system on mobile devices. It achieves real-time video enhancement with competitive visual quality while allowing runtime behavior adaptation to the platform-imposed dynamic energy budgets. We report extensive experiments on diverse datasets, scenarios, and platforms and demonstrate the superiority of AdaEnlight compared with state-of-the-art low-light image and video enhancement solutions.

CCS Concepts: • **Human-centered computing** → **Ubiquitous and mobile computing systems and tools**; • **Computing methodologies** → *Image processing*.

Additional Key Words and Phrases: mobile devices, low light video enhancement, energy awareness

ACM Reference Format:
Sicong Liu, Xiaochen Li, Zimu Zhou, Bin Guo, Meng Zhang, Haocheng Shen, and Zhiwen Yu. 2022. AdaEnlight: Energy-aware Low-light Video Stream Enhancement on Mobile Devices. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 4, Article 172 (December 2022), 26 pages. https://doi.org/10.1145/3569464

1 INTRODUCTION

The pervasive deployment of camera-embedded mobile devices *e.g.*, smartphones, wearables, tablets, and robots has stimulated a wide spectrum of novel mobile video based applications. Examples include face detection on

Corresponding author: guob@nwpu.edu.cn.
Authors' addresses: Sicong Liu, Northwestern Polytechnical University, School of Computer Science, Xi'an, China; Xiaochen Li, Northwestern Polytechnical University, School of Computer Science, Xi'an, China; Zimu Zhou, City University of Hong Kong, School of Data Science, Hong Kong, China; Bin Guo, Northwestern Polytechnical University, School of Computer Science, Xi'an, China ; Meng Zhang, Northwestern Polytechnical University, School of Computer Science, Xi'an, China; Haocheng Shen, Northwestern Polytechnical University, School of Computer Science, Xi'an, China; Zhiwen Yu, Northwestern Polytechnical University, School of Computer Science, Xi'an, China.

2474-9567/2022/12-ART172 $$15.00
https://doi.org/10.1145/3569464

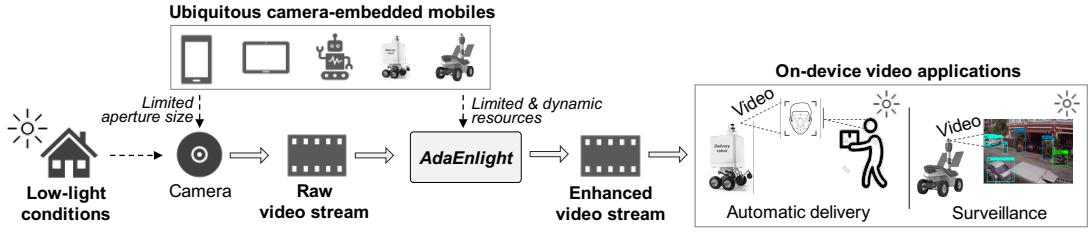


Fig. 1. Illustration of AdaEnlight as a video enhancement middleware for downstream mobile video applications.

smartphones for user authentication [10], outdoor surveillance with mobile robots [1, 21], and object detection with drones [61], *etc.* These applications are features with two characteristics. *(i)* There is a growing interest in *on-device processing* of these video streams rather than cloud offloading for privacy concerns or real-time interactions [2]. *(ii)* The video streams captured by mobile devices often suffer from *low-light effect* since they are typically taken by non-professional users with narrow-aperture cameras [41, 46]. Low-light videos can notably impair user experience *e.g.*, for video display, and downstream application accuracy, *e.g.*, face detection.

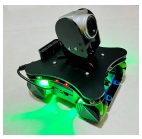
Despite extensive research on low-light image and video enhancement [6, 14, 15, 19, 28, 29, 36, 59], they are unfit for deployment to mobile devices because they fail to meet the following requirements.

- *Real-time Processing on Resource-constrained Mobile Devices.* The *real-time* processing of video stream is necessary for application's responsiveness, and *on-device* processing benefits the user privacy. However, mainstream low-light video enhancement schemes are computation-intensive [6, 19, 36, 59]. Even the state-of-the-art lightweight image enhancement model Zero-DCE++ [28] fails to achieve real-time processing on mobile devices such as a Raspberry Pi (5 $frames/s$ for 270×480 RGB images, see Sec. 5).
- *Adapting to Diverse Energy Supply of Mobile Devices. Energy awareness* is crucial for the long-term operation of the above video processing since mobile devices are often battery-powered [4, 13]. It is desired that the low-light video processing module can adapt its enhancement quality according to the energy budget at runtime. Despite prior studies on adaptive mobile computer vision [3, 23, 26, 44, 48, 58] and general energy profiling of neural networks [22, 25, 39, 53–55], there lacks a runtime energy profiler and adaptation loop dedicated to deep learning enabled low-light video enhancement.

In this paper, we propose AdaEnlight, an energy-aware on-device low-light video enhancement solution for mobile platforms. It is challenging to achieve high enhanced visual quality, low execution latency, and adaptation to energy budget on mobile platforms. AdaEnlight tackles these challenges via a modular system design and a set of novel algorithms. Observing that latency bottleneck of the state-of-the-art image enhancement schemes [14, 28] lies in the iterative enhancement curve function, AdaEnlight devises a novel non-iterative curve function for acceleration without compromising enhancement quality. AdaEnlight also incorporates temporal consistency among frames to mitigate the flicking problem [29] when enhancing video streams. To adapt to the dynamic energy budget imposed by mobile platforms, AdaEnlight exploits a runtime energy profiler to estimate the energy cot of the video enhancement process at runtime and configures its hyper-parameters (*e.g.*, computation reuse and frame resolution) accordingly. We implement AdaEnlight as a compact video pre-processing middleware to provide enhanced video streams for various downstream mobile video applications such as facial recognition in automatic package delivery and object detection in video surveillance (see Fig. 1).

Our main contributions are summarized as follows.

- To the best of our knowledge, AdaEnlight is the first *on-device* low-light *video* enhancement system for *mobile* scenarios. It achieves near real-time and high-quality processing on commodity mobile platforms and stays adaptive to the dynamic energy budget.


Indoor walk


Outdoor walk

Indoor	Daytime (2 hours)	No.1
	Nighttime (1 hour)	No.2
Outdoor	Daytime (2 hours)	No.3
	Nighttime (1 hour)	No.4

Four groups of study

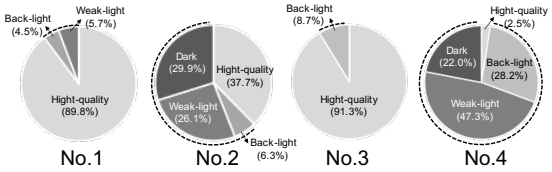


Fig. 2. An example study of the ubiquity of low-light video streams in the mobile video surveillance scenario.

- The key technical novelties include a non-iterative low-light enhancement model that breaks the latency bottleneck of the state-of-the-art enhancement schemes and enforces temporal consistency between video frames. Also, it adopts a runtime energy-aware controller that integrate a runtime energy profiler to adjust the video enhancement behaviors adaptively.
- We evaluate AdaEnlight on four public benchmarks and real-world mobile scenarios on three mobile platforms (Honor 9 smartphone, raspberry Pi 4B, and Jetson AGX Xavier). Experiments show that AdaEnlight achieves near real-time (30 $\sim$ 50ms per frame) video enhancement with competitive visual quality to existing image and video enhancement schemes. AdaEnlight also enables agile self-adaptation to satisfy the dynamic energy budgets at runtime.

In the rest of this paper, we present an overview of AdaEnlight as well as its functional modules in Sec. 2, Sec. 3, and Sec. 4. We show the evaluations in Sec. 5, review related work in Sec. 6, and conclude in Sec. 7.

2 ADAENLIGHT OVERVIEW

This section presents our problem scope and the design overview of AdaEnlight.

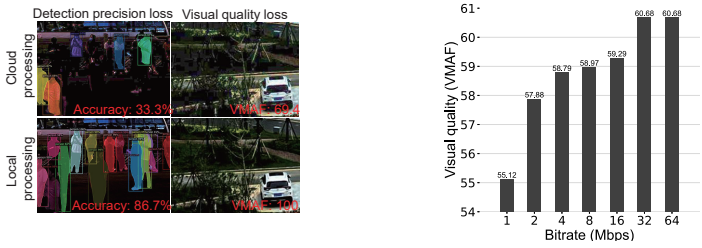
2.1 Problem Scope

In short, we target at on-device low-light enhancement of ubiquitous video streams. We elaborate on the concrete motivations and requirements below.

2.1.1 Motivations. Our work is motivated by the ubiquity of camera-embedded devices (*e.g.*, smartphones, wearables, tablets, and robots) in everyday life for various video applications ranging from video surveillance [21] to video conferencing [9]. These video streams, which we call as *ubiquitous video streams*, are often captured by *non-professional users* and are likely to experience *low-light* conditions. The reasons are two-fold. *(i)* The limited camera aperture size on mobile devices restrict the overall amount of light that reaches the camera sensor and thereby decrease the signal-to-noise ratio of the captured videos [46]. *(ii)* Non-professional users may take videos under uncontrolled lighting conditions, *e.g.*, back-lit, weak light, and extremely dark environments [41]. Also, we conduct an example study using a mobile robot (*i.e.*, Yahboom) to show the ubiquity of low-light video streams in the daily video surveillance scenario. The robot continuously collects video streams by random walking indoor and outdoor for 6 hours, at daytime and nighttime. Fig. 2 shows the details of four groups of study. The proportion of low-light video streams is up to 87.5% (*i.e.*, group No.4) and 10.5% (*i.e.*, group No.1) at nighttime and daytime, respectively.

We consider video enhancement *locally on-device* rather than *remotely on cloud* for the following reasons.

- The ubiquitous video streams may be immediately consumed on the mobile devices [2]. For example, a surveillance drone may run object recognition algorithms on video streams taken in dim light and only uploads suspicious frames to cloud servers to save bandwidth. An automatic delivery robot may authenticate the target recipient before delivery by running face recognition algorithms locally for privacy concerns.



(a) (b)

Fig. 3. Comparison of video enhancement locally on-device and remotely on-cloud. (a) Example frames enhanced by local processing and remote processing. The upper row follows the encode-decode-enhance pipeline of remote processing, while the lower row are directly enhanced as in local processing. (b) Enhanced video quality of the encode-decode-enhance pipeline under different frame bitrates into the video codec.

- On-device processing avoids the extra loss in visual quality due to video encoding during data transmission. Specifically, offloading video enhancement to cloud follows an "encode-decode-enhance" pipeline. Frames are encoded at the mobile device, transmitted via wireless networks, decoded at the cloud, and finally fed into the video enhancement module. Video encoding is necessary due to the limited wireless bandwidth. Yet it can decrease the enhanced visual quality because many mainstream video encoding schemes *e.g.*, H.264 [51] are lossy. Fig. 3a shows an example, where the upper row shows the frames enhanced following cloud processing *i.e.*, encode-decode-enhance, while the lower row shows the results of local processing *i.e.*, directly enhance. There is notable loss in detection precision (53.4% accuracy loss in the upper left figures) and visual quality (30.6 VMAF loss in the upper right figures). Fig. 3b provides a more quantitative study. We enhance one example video clip from our self-collected dataset (MobileScene, see Sec. 5.1 for details) following the "encode-decode-enhance" pipeline using the standard H.264 video codec for encoding/decoding and AdaEnlight for enhancement. We vary the input bitrate into the codec and measure the visual quality of the enhanced frames by the VMAF index [31]. As is shown, the bitrate (and thus the network bandwidth) limits the visual quality of video enhancement. The VMAF index is only 55 at the bitrate of 1 Mbps. Even at a bitrate of 64 Mbps (higher than the uplink bandwidth of commercial 5G networks [45]), the VMAF index is only around 60, which is lower than that of local processing (see Tab. 5 and Tab. 6 in our evaluations).

2.1.2 Requirements. To support the low-light enhancement of ubiquitous video streams on mobile devices, a solution should meet the following criteria.

- *Near Real-time Video Enhancement.* As with other mobile computer vision tasks [32, 43, 52], the low-light enhancement model should be lightweight to fit in the limited execution resources on mobile platforms and to achieve near real-time processing of the video streams.
- *Energy-aware Adaptation.* Since many mobile devices are battery-powered, it is crucial that the video enhancement is energy-aware. Particularly, it is desirable that the video enhancement process can trade off between the output visual quality and the platform-imposed energy budget at runtime.

Despite prior research on the low-light image/video enhancement [6, 14, 15, 19, 28, 29, 36, 59] and adaptive mobile computer vision [3, 23, 26, 44, 48, 58], no solution fulfills the above requirements simultaneously, which motives the design of AdaEnlight. Note that we focus on enhancement of RGB videos taken in low light. We leave extreme low light enhancement to future work because such cases often require videos stored in raw formats [6, 19, 29], which are not pervasively supported in mobile and embedded platforms.

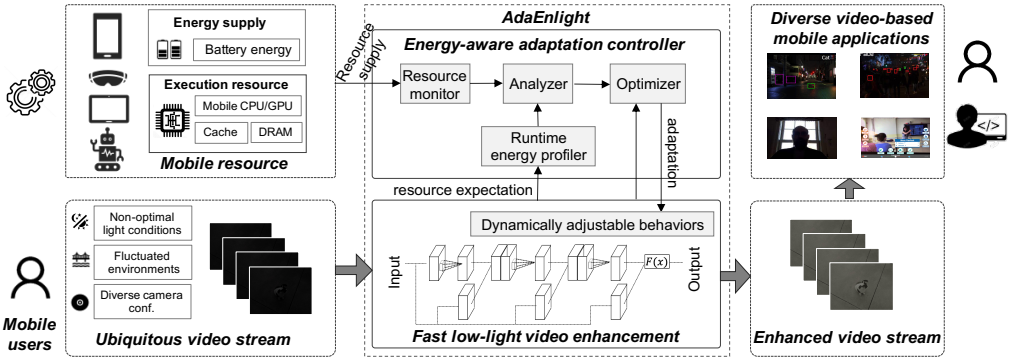


Fig. 4. Workflow of AdaEnlight.

2.2 Solution Overview

Our solution, AdaEnlight, is an energy-aware low-light video enhancement scheme that achieves near real-time video processing with competitive enhanced visual quality against the state-of-the-arts [14, 28, 36], while allowing runtime enhancement behavior adaptation to platform-imposed energy budgets. This is achieved by decoupling the requirements in Sec. 2.1.2 with two functional modules: *fast low-light video enhancement* and *energy-aware adaptation controller*.

- *Fast Low-Light Video Enhancement (Sec. 3).* This module aims at near real-time low-light video enhancement on mobile platforms without compromising enhancement quality. Prior low-light enhancement schemes are unfit for our purposes because they either induce unacceptable latency [19, 36] or are not optimized for videos [14, 28] *e.g.*, suffering from the flicking problem [29]. In AdaEnlight, we design a new low-light video enhancement model by removing the latency bottleneck of a state-of-the-art low-light image enhancement model [14] and enforcing temporal consistency between frames.
- *Energy-aware Adaption Controller (Sec. 4).* This module dynamically adjusts the frame resolution and computation reuse of the video enhancement model to maximize the enhanced video quality while minimizing the energy consumption. Such adaptation is non-trivial due to the multi-objective nature of the optimization problem and the challenge to estimate energy consumption at runtime. In AdaEnlight, we develop an optimizer to heuristically solve the multi-objective optimization problem and propose a runtime energy profiler to estimate the energy cost of the video enhancement model at runtime. This module also contains a resource monitor and analyzer to automate the adaptation process.

Fig. 4 shows the workflow of AdaEnlight. The ubiquitous video streams captured by a camera-embedded mobile device are fed into AdaEnlight's fast low-light video enhancement module for visual quality enhancement. Meanwhile, AdaEnlight's energy-aware adaptation module adds an extra control flow to configure the video enhancement module's parameters according to the dynamic energy budgets of the mobile device.

We implement AdaEnlight as a compact video pre-processing middleware. It transparently augments the low-light video streams into high-quality ones for downstream on-device video applications such as facial recognition and object detection. As a middleware, AdaEnlight can resolve the application diversity and the system complexity and effectively deal with multiple co-existing demands and constraints from different aspects (*e.g.*, visual quality, latency, energy). Application developers do not have to initiate the video enhancement or be involved in the details of system monitoring and analysis.

3 FAST LOW-LIGHT VIDEO ENHANCEMENT

This section presents AdaEnlight's fast low-light video enhancement model. It is built upon prior curve Based image enhancement solutions (Sec. 3.1). To satisfy our special requirements, *i.e.*, near *real-time* processing and *video* enhancement, we exploit Gamma correction based curve to accelerate the enhancement speed (Sec. 3.2) and extend the image-specific solution to videos by incorporating temporal consistency between frames (Sec. 3.3). The detailed implementations are in Sec. 3.4.

3.1 Primer on Curve Based Image Enhancement

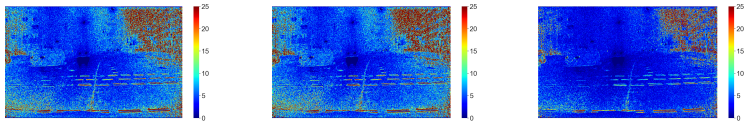
Low-light image enhancement takes a low-light image as input, and outputs a normal-light, high-quality image [29]. We base our design upon Zero-DCE [14], a recent zero-shot learning low-light image enhancement scheme via deep curvature estimation. The reasons are as follows. *(i).* We choose the *zero-shot learning* based solution to eliminate the need for paired training data, *i.e.*, videos of the same scene taken in dim and normal illumination. This is desired because access to such paired videos is limited in real-world mobile applications with high variability. *(ii).* We select the *curve-based model* for it yields competitive image enhancement performance despite its lightweight architecture [28], which holds potential for execution on resource-constrained mobile devices. As next, we provide a brief review on Zero-DCE [14] and discuss its limitations for fast video enhancement.

3.1.1 Principles of Zero-DCE. The design of Zero-DCE [14] follows the curve adjustment in photo editing software. It derives an *image-specific curve* that maps a low-light image to its enhanced version. The curve is defined as a high-order pixel-wise function learned by a deep neural network, where its loss function is designed as a set of differentiable non-reference losses to enable zero-shot learning. The curve function is defined as:

$$E_n(\mathbf{x}) = E_{n-1}(\mathbf{x}) + A_n(\mathbf{x}) \cdot E_{n-1}(\mathbf{x}) \cdot (1 - E_{n-1}(\mathbf{x})) \tag{1}$$

where $\mathbf{x}$ is the pixel coordinates of the input image $I(\mathbf{x})$, $E_n(\mathbf{x})$ is the enhanced output image (note that $E_0(\mathbf{x}) = I(\mathbf{x})$), n is the number of iteration to control the curvature, $A_n(\mathbf{x})$ is a parameter map of the same size as $I(\mathbf{x})$, where each element in $A_n(\mathbf{x})$ is a trainable curve parameter in $[-1, 1]$. Each pixel in $I(\mathbf{x})$ is normalized to $[0, 1]$ and the enhancement is performed separately to three RGB channels. The curve function $E_n(\mathbf{x})$ is learned via a neural network with a loss function that consists of spatial consistency, exposure control, color constancy, and illumination smoothness. The spatial consistency loss L_{spa} encourages spatial coherence of the enhanced image. The exposure loss L_{exp} controls the exposure level of enhanced image. The color loss L_{col} corrects the potential color deviations. The illumination smoothness loss L_{tvA} preserves monotonicity relations between neighboring pixels. More details can be found in [14].

3.1.2 Limitations of Zero-DCE. Despite being one of the most *lightweight image enhancement* scheme, Zero-DCE fails to support *fast video enhancement* on resource-constrained mobile devices for the following reasons. *(i)*, the iterative procedure in *i.e.*, Equ.(1), for image enhancement, incurs *notable delay* that prohibits real-time video enhancement. For example, Zero-DCE takes an average of 11.74s to enhance a 270×480 RGB image on a Raspberry Pi 4B platform (detailed setups in Sec. 5.1). The follow-up Zero-DCE++ [28] accelerates Zero-DCE by adopting depth-wise separable convolutions and reusing the curve parameter maps $A_n(\mathbf{x})$. However, Zero-DCE++ still induces 0.19s delay, *i.e.*, only 5 frames per second. The main bottleneck lies in the iterative procedure, where n is empirically optimized to 8 in both Zero-DCE and Zero-DCE++. *(ii)*, directly applying image enhancement schemes to videos leads to *flicking* between frames [6, 59]. Fig. 5 shows the difference between two consecutive images enhanced by Zero-DCE and Zero-DCE++. There is notable differences between adjacent frames (up to 9.192) and thereby results in flicking in videos. These limitations motivate us to rethink the design for the fast video enhancement in two aspects, as will be explained in Sec. 3.2 and Sec. 3.3.



(a) Zero-DCE (b) Zero-DCE++ (c) AdaEnlight

Fig. 5. Illustration of the flicking problems when applying low-light image enhancement schemes, *e.g.*, (a) Zero-DCE [14] and (b) Zero-DCE++ [28] to videos. The figures show the difference between two consecutive frames. The greater the difference, the more serious the flicking problem is. In comparison, (c) shows the low frame difference of AdaEnlight.

3.2 Non-iterative Frame Enhancement with Gamma Correction-based Curve

As previously mentioned (Sec. 3.1.2), the latency bottleneck of curve-based image enhancement [14, 28] lies in the iterative enhancement process. We observe that the iterative procedure may be eliminated if a non-iterative curve function with higher non-linearity is adopted. Therefore, we propose to replace the traditional quadratic curve in Equ.(1) by an exponential curve to induce higher-order non-linearity and smaller latency. Specifically, we propose the following non-iterative image enhancement scheme.

$$E(\mathbf{x}) = I(\mathbf{x})^{\Gamma(\mathbf{x})} \tag{2}$$

where $I(\mathbf{x})$ and $E(\mathbf{x})$ denote the input and enhanced image, respectively. $\Gamma(\mathbf{x})$ are pixel-wise trainable parameters learned from a neural network. The detailed training methodology of $\Gamma(\mathbf{x})$ are deferred to Sec. 3.4. We make two detail notes on our design of Equ.(2).

- The exponential curve function in Equ.(2) is inspired by the Gamma correction, which is typically used to flexibly adjust the luminance or tristimulus in images [40]. Our novelty is to apply pixel-wise parameters $\Gamma(\mathbf{x})$ for fine-grained adjustment, which is learned from a neural network. In contrast, conventional Gamma correction adopts the same coefficient Γ for the entire image, which tends to introduce artifacts and color deviations to the enhanced images [29].
- One may want to design an iterative enhancement process by rewriting Equ.(2) as $E_n(\mathbf{x}) = E_{n-1}(\mathbf{x})^{\Gamma_n(\mathbf{x})}$, where $E_0(\mathbf{x}) = I(\mathbf{x})$. We exclude this option because our experiments show that the enhanced visual quality does not notably increase with more iterations (see Sec. 5.4.1).

To summarize, the learned pixel-wise Gamma correction-based curve in Equ.(2) tends to deliver high-quality visual enhancement while avoiding the long latency. We defer the quantitative comparisons with the traditional iterative enhancement scheme to Sec. 5.4.

3.3 Incorporating Temporal Consistency Loss for Video Enhancement

As mentioned in Sec. 3.1, naively adopting image enhancement schemes to videos collected by mobile devices causes flicking. A remedy is to account for temporal consistency between frames when designing the loss function to train the enhancement curve *i.e.*, Equ.(2). Given two enhanced frames $E_{t+1}(\mathbf{x})$ and $E_t(\mathbf{x})$ at timestamp $t+1$ and t, we consider their temporal consistency with the following losses.

- *Exposure Consistency Loss*: We define the exposure consistency loss between two enhanced frames as

$$\mathcal{L}_{exp,temporal} = \sum_{x} \left| \sum_{i=r,g,b} E^i_{t+1}(\mathbf{x}) - \sum_{i=r,g,b} E^i_t(\mathbf{x}) \right| \tag{3}$$

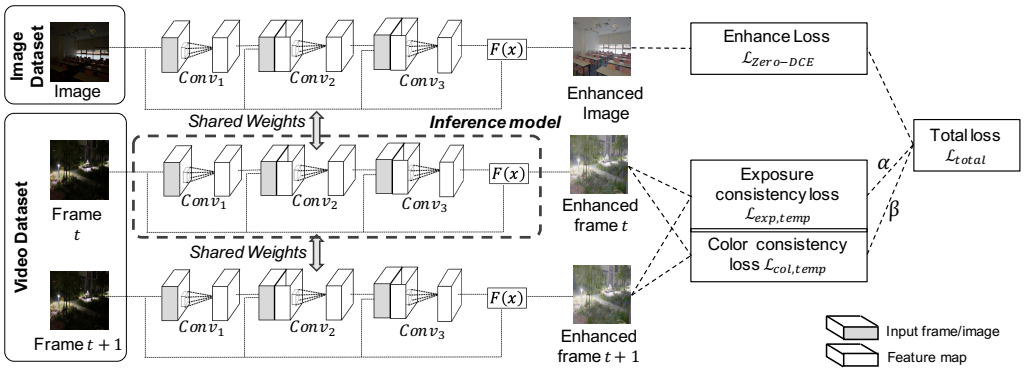


Fig. 6. Model architecture and training scheme to learn the pixel-wise curve parameters in Equ.(2). The model inside the dashed box is used for inference.

- *Color Consistency Loss*: We define the color consistency loss between two enhanced frames as

$$\mathcal{L}_{col,temporal} = \sum_{x} \sum_{k=r,g,b} \left| \frac{E^k_{t+1}(\mathbf{x}) + c}{\sum_{i=r,g,b} E^i_{t+1}(\mathbf{x}_i) + c} - \frac{E^k_t(\mathbf{x}) + c}{\sum_{i=r,g,b} E^i_t(\mathbf{x}_i) + c} \right| \tag{4}$$

where $E^i(\mathbf{x})$ denotes the enhanced image of one channel, where $i = r, g, b$, which represents the RGB channels. c is a delta value empirically set to 0.0001 to avoid division by zero in case of dark pixels. The exposure consistency loss sums over the RGB channels to avoid the impact of colors when calculating the temporal consistency. The color consistency loss, in contrast, is divided by the exposure to mitigate the impact of pixel-wise exposure when calculating the temporal consistency.

Note that to avoid the influence of motion between frames, we employ the Gunnar-Farneback optical flow [12], a dense optical flow alignment method as a pre-processing step. Specifically, we calculate motions between the two frames (frame $t+1$ and frame t) and get the motion data of each pixel of frame t, denoted as (dx, dy). Then we modify the location of each pixel in frame t as $(x + dx, y + dy)$. The resulting fame as named as the aligned frame t. Finally, we calculate the absolute difference between the aligned frame t and frame $t+1$.

3.4 Putting It Together

In short, our fast low-light video enhancement module is a deep curve-based enhancement scheme with a learned pixel-wise Gamma correction based curve *i.e.*, Equ.(2). The curve parameters are trained by considering both the conventional image enhancement loss as well as the temporal consistency loss *i.e.*, Equ.(3) and Equ.(4). Below we explain the detailed model architecture, its training strategy and how it is designed to trade off between video enhancement quality and platform-imposed requirements at runtime.

3.4.1 Model Architecture. We apply a U-Net [42] to learn the parameters $\Gamma_n(\mathbf{x})$ in Equ.(2). For lower latency, we decrease the number of layers and the channels of the original U-Net. We also change the contracting paths in the original U-Net to dense connections as DenseNet [16] for computation reuse (see Sec. 3.4.3). Fig. 6 shows the model architecture. Tab. 1 lists the detailed hyperparameters. The hyperparameters are empirically tuned to trade off between enhancement quality and latency. The ablation studies on hyperparameters are in Sec. 5.4.3.

We combine the loss function from Zero-DCE [14], *i.e.*, image enhancement loss, with the two temporal consistency losses defined in Equ.(3) and Equ.(4) as the final loss function.

$$\mathcal{L}_{total} = \mathcal{L}_{Zero-DCE} + \alpha \mathcal{L}_{exp,temporal} + \beta \mathcal{L}_{col,temporal} \tag{5}$$

Table 1. Detailed hyperparameters of the neural network.

Layer	Parameters	Layer	Parameters	Layer	Parameters
Conv_1	K=1, Cin=3, Cout=1, S=1, G=1, P=0 K=3, Cin=1, Cout=1, S=1, G=1, P=0	Conv_2	K=1, Cin=4, Cout=1, S=1, G=1, P=0 K=3, Cin=1, Cout=1, S=1, G=1, P=1	Conv_2	K=1, Cin=4, Cout=3 S=1, G=1, P=0

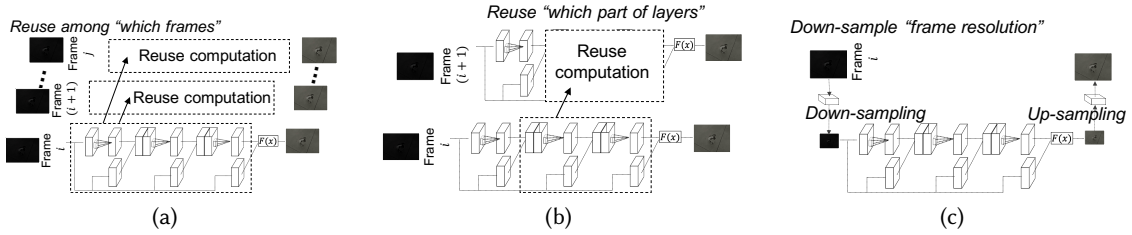


Fig. 7. Illustrations of adjustable parameters of the video enhancement model: computation reuse as (a) frame level and (b) layer level; (c) down-sample the resolution of input frames.

where α and β can be empirically tuned.

3.4.2 Training Strategy. Due to the lack of video datasets with evenly distributed exposure levels, it is challenging to train a network that minimizes Equ.(5). Therefore, we propose to learn the curve by training the image enhancement loss and the temporal consistency losses in a siamese mode as in [6, 59]. Fig. 6 shows the training scheme. In each pass, we input one low-light image from the image dataset and two adjacent low-light video frames from the video dataset into the neural network in a siamese way *i.e.*, three neural networks share the same weights. The low-light image will be used to calculate the image enhancement loss, whereas the two frames will be used to compute the two temporal consistency losses.

At inference time, the model takes a low-light video frame as input, and generates an enhanced frame in almost real-time (0.05 second delay or 20 frames per second for 270×480 RGB images on the Raspberry Pi 4 (4GB) platform. More quantitative evaluations on enhancement quality and latency are in Sec. 5.2.

3.4.3 Towards Runtime Adaptation. Although our low-light video enhancement module achieves near real-time performance, there are other requirements *e.g.*, energy, which the users may trade with enhancement quality. Our video enhancement module supports such runtime adaptation by adjusting the parameters below.

- *Computation Reuse.* Recall that our video enhancement model consists of three convolutional layers. Due to temporal similarities between adjacent frames, we can further reduce the computations by enforcing computation reuse during frame enhancement. We allow reuse at both the frame level (see Fig. 7a), *i.e.*, reuse the enhanced output of frame t for frame $t + 1$, and the layer level (see Fig. 7b), *i.e.*, reuse the intermediate results and skip some layers during enhancement.
- *Frame Resolution.* Down-sampling the resolution of the input frame naturally reduces the computation of the video enhancement process (see Fig. 7c).

We utilize these parameters to trade off between enhancement quality and energy at runtime, as explained next.

4 ENERGY-AWARE ADAPTATION CONTROLLER

This section presents the AdaEnlight's energy-aware adaptation controller to dynamically adjust the above mentioned video enhancement model's behaviors for energy conservation. The focus of this controller is to optimize the trade-off between energy and the video quality, because the above design (Sec. 3.2) has already guaranteed

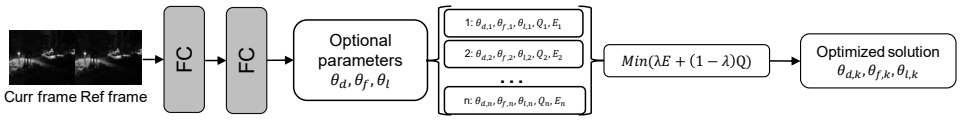


Fig. 8. Workflow of the optimizer in the energy-aware adaption controller.

the near real-time video processing. We first introduce the design of the overall controller (Sec. 4.1), then the runtime energy profiler (Sec. 4.2), and finally the auxiliary modules to automate the control loop (Sec. 4.3).

4.1 Controller Design

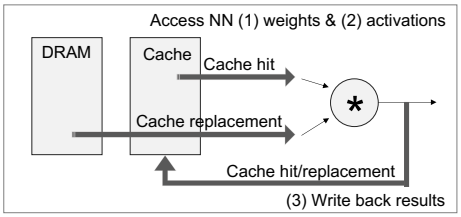
At a high level, the energy-aware adaptation controller of AdaEnlight is a runtime heuristic optimizer for a multi-objective optimization problem, as formulated below.

4.1.1 Formulation. Mathematically, the controller aims to continuously optimize two metrics, *i.e.*, the visual quality Q_{video} of the output video, and the energy consumption $E_{enhancer}$ of the video enhancement model. We formulate it as the following time-varying optimization problem.

$$\underset{\theta_l,\theta_f,\theta_d}{argmax}\ Q_{video} = q(\theta_l, \theta_f, \theta_d, video(t)),\ \underset{\theta_l,\theta_f,\theta_d}{argmin}\ E_{enhancer} = e(\theta_l, \theta_f, \theta_d, resource(t)) \tag{6}$$

where θ_l, θ_f, and θ_d are three dynamically adjustable parameters of the video enhancement model, *i.e.*, the amount of layer-level computation reuse, the amount of frame-level computation reuse, and the down-sampled video frame resolution (see Sec. 3.4.3). It is intractable to obtain a closed-form solution to the above dynamic multi-objective optimization problems because dealing with two distinct yet related spaces, *i.e.*, *objective space* and *variable space*, is challenging. Specifically, the *objective space* consisting of the visual metric Q_{video} and the energy metric $E_{enhancer}$. Regarding the *variable space* related to the objective performance, some variables (*i.e.*, θ_l, θ_f, and θ_d) are available for objective optimization, we regard them as *decision variables*. Other variables (*e.g.*, the available execution resource $resource(t)$, the video stream $video(t)$) are imposed by the external context, which are time-varying but independent from the optimization objective variables.

4.1.2 Optimizer. To solve the above problem, we propose a heuristic optimizer. Specifically, we solve the original optimization problem in two stages, which correspond to two typical dynamic optimization problems [11]. In the offline stage, we regard this problem as a static optimization problem and adopt existing evolutionary algorithm to find a widely distributed set of solutions and derive the Pareto-optimal front. When looking for the Pareto frontier, we do not set any relative importance coefficients for multiple optimization objectives (*i.e.*, Q and E), because we need to get the unbiased opinion from this set of solutions. In the online stage, the optimal decision variables (*i.e.*, θ_l, θ_f, θ_d) changes, whereas the optimal objective space (*i.e.*, Q, E) does not change, which is the Type 2 dynamic multiple-objective optimization problem [11]. Therefore, we leverage an analytical hierarchy process to set the dynamic importance coefficients λ of the different criteria. Then we multiply the respective performance of each solution with that coefficients, to get a total score for each solution, *i.e.*, $\lambda E_{enhancer} + (1 - \lambda)Q_{video}$, indicating which suits best the current wishes. For example, if the the enhancement model's demand stray beyond resource levels, the controller re-selects the dynamic importance coefficients to pick a sole optimal solution (*i.e.*, $\theta_l, \theta_f, \theta_d$) from the Pareto frontier at runtime.



(a)

Operation types	Energy cost (*PJ*)	Normalized
32 bit fixed add	0.1	1
32 bit fp add	0.9	9
32 bit fixed mul	3.1	31
32 bi fp mul	3.7	37
32 bit 32K cache access	5	50
32bit DRAM access	640	6400

(b)

Fig. 9. (a) Memory access and (b) energy cost of a convolutional operation on a mobile device with 45nm CMOS chip.

4.2 Runtime Energy Profiler

For the controller to function properly, it is essential to provide an accurate and timely estimate of $E_{enhancer}$ to the controller. In AdaEnlight, we propose an energy model for the video enhancement model that estimates its energy consumption given the adjustable parameters θ_l, θ_f, and θ_d at runtime.

However, the neighbor solutions on the Pareto frontier in the objective space are not always neighbors in the decision variable space, which makes the search of decision variable time-consuming. In view of this challenge, we design an extra parametric neural network based regression module for building the map between the state (*i.e.*, the input video frame and the previous reference frame), decision variables (*i.e.*, DO, FO, DR), and the objective outcomes (*i.e.*, Q, E). The network consists of two fully-connected layers. The input of the neural network is the splicing matrix containing the pixel matrix of the input frame and the previous reference frame. The output is the vector of θ_l, θ_f, θ_d as well as the objective outcomes (*i.e.*, Q, E). We collect 400 samples to train this regression network. At run-time, the network outputs all the optional parameters θ_f, θ_d, θ_l and the corresponding Q, E. The decision space built by $\theta_f \in \{0, 1, 2...10\}$, $\theta_d \in \{1, 1/2, 1/3\}$, $\theta_l \in \{0, 1, 2, 3\}$ is limited. Therefore, together with the dynamic importance coefficient λ, it is easy to get best computation reuse parameters by calculating $\lambda E_{enhancer} + (1 - \lambda)Q_{video}$. Fig. 8 briefly describes the optimizer workflow.

4.2.1 Principles. As discussed in Sec. 6.2, it is challenging to model the energy consumption of a neural network because the energy cost is tightly coupled with the available execution resources of the given platform. The principles of our runtime energy profiler are two-fold.

- To estimate the energy cost of a neural network with dynamically adjustable hyperparameters, we decompose the network into layers and strive to model the energy cost on a layer basis. The layer-wise decomposition is reasonable because mobile devices with CPU/GPU tend to execute a neural network layer-by-layer due to limited resources [27].
- To estimate the energy cost in presence of time-varying execution resources, we convert the resource dynamics into a single parameter, the cache-hit-rate, which is directly measurable at runtime. This is because we empirically observe that the energy cost of memory accesses is dominated by the off-chip memory accesses, which can be estimated by the cache-hit-rate. Fig. 9 shows an example of to execute a convolution operation on a mobile platform (*e.g.*, with a 45nm CMOS ARM chip). There are three types of memory accesses for loading the weights and activations of a neural network and writing back results (see Fig. 9a), and the DRAM access dominates the energy cost (see Fig. 9b). Since DRAM (off-chip) accesses are only necessary when the on-chip cache misses, we can use the cache-hit-rate as the multiplier to estimate the energy cost when executing a layer.

These principles reduce the overhead to estimate the energy cost to execute a neural network on a platform.

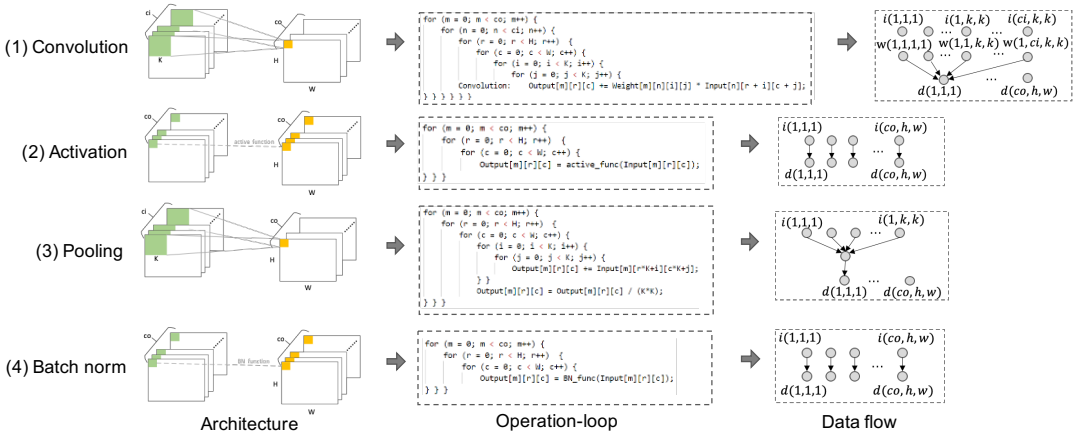


Fig. 10. Counting the memory accesses from the underlying operation loops and data flow of different layers.

4.2.2 Energy Model. We first propose the energy model to execute a single layer l on mobile CPU/GPU platforms.

$$E_l = \delta_C \times C_l + \epsilon \times \delta_{cache} \times M_l + (1 - \epsilon) \times \delta_{DRAM} \times M_l + M_l \times \delta_{SM} \tag{7}$$

where δ_C, δ_{cache}, δ_{DRAM}, and δ_{SM} are the unit energy cost of MAC computation, cache access, DRAM access, and the shared memory access; C_l and M_l are the total number of MAC and memory access of layer l, respectively; and ϵ is the runtime cache-hit-rate. For a given hardware platform, these parameters can be derived as follows.

- *Unit Energy Cost* δ_C, δ_{cache}, δ_{DRAM}, and δ_{SM}. These parameters can be measured offline. We empirically set $\delta_C : \delta_{cache} : \delta_{DRAM} : \delta_{SM} = 1 : 6 : 200 : 2$ for mobile GPU platforms. As for mobile CPU platforms, $\delta_{SM} = 0$ since they do not have such shared memory space, and thus $\delta_C : \delta_{cache} : \delta_{DRAM} = 1 : 6 : 200$. According to our evaluations, these parameters work with different DNN inference frameworks, *e.g.*, Raspberry Pi 4B (CPU) + NCNN, Hornor 9 (CPU) + Pytorch Mobile, and Nvidia Jetson Nano (GPU).
- *Number of MAC Computation* C_l. The amount of MAC can be directly derived from the layer architecture of layer l, *e.g.*, $C_{conv_l} = K^2 * C_o * C_i * H * W$.
- *Number of Memory Accesses* M_l. The memory access during the tensor computation consists of the accesses of layer parameters, input feature maps, and intermediate iterations. We compute M_l for a given layer type according to the data flow mapping to the given hardware platform. Fig. 10 illustrates the data flow operations of different layer types used by our video enhancement model. For example, the total amount of memory accesses M_l for a convolution layer l can be calculated by the six-fold operation loops, *i.e.*, $M_{conv_l} = 2 * C_o * C_i * H * W * K^2 + C_o * H * W$.
- *Runtime Cache-Hit-Rate* ϵ. It accounts for the dynamics of the execution resources and thus is measured at runtime. We measure ϵ as the ratio of the actual MAC execution amount per unit time (*i.e.*, *ms*) to the MAC amount with the simulated 100% cache-hit-rate. The amount of MAC executed with a 100% cache-hit-rate is profiled offline for the given platform. Observing that framework/compiler-level optimization such as operator fusion may affect the cache hit rate ϵ even for the same model [7, 60], we measured the amount of MAC executed with a 100% cache-hit-rate for multiple mainstream frameworks.

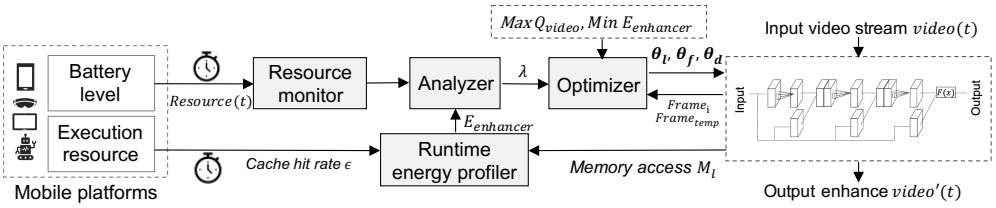


Fig. 11. Energy-aware adaptation controller workflow.

Given the energy model for a single layer l as in Equ.(7), we can estimate the total energy consumption of n_{frame} video frames, with adjustable parameters θ_l, θ_f, and θ_d as follows.

$$E_{enhancer} = \sum_{f=1}^{n_{frame}} \sum_{l=1}^{L} \theta_{f_{(index_f)}} \theta_{l_{(index_l)}} E_l \theta_{d_f} \tag{8}$$

where $\theta_{f_{(index_f)}}$ and $\theta_{l_{(index_j)}}$ are Boolean value, and 0 means skipping the computation of frame f or layer l. And θ_d is the down-sampling rate (%) that affects the resolution of the input frame f by $\theta_{d_f} * H * W$.

4.2.3 Profiler Workflow. The energy profiler in AdaEnlight works in two stages, *offline* and *online*.

- *Offline Stage.* The unit energy cost δ_C, δ_{cache}, δ_{DRAM}, δ_{SM}, as well as the MAC throughput of the simulated 100% cache-hit-rate are measured offline for the given platform. Specifically, during the offline data collection stage, we use a digital power monitor, *i.e.*, Monsoon AAA10F, to sample the ground truth of power consumption by profiling the device through its external power input. The energy cost of accessing the Cache, DRAM, and shared memory normalized to that of a MAC operation is determined by the platform based on the ground truth. The MAC execution amount per second in the simulated 100% cache-hit-rate is the hardware-specified computation frequency times the computational parallelism (*e.g.*, 16 bits). The result is an energy profile with a sequence samples $\{catch - hit - rate/power\ drawn\}$.
- *Online Stage.* During the online profiling stage, the profiler takes the current model hyperparameters and the runtime cache-hit-rate ϵ as input, and predicts the energy demand using the energy model in Equ.(8).

Note that the primary goal of the profiler is to ensure consistent *ranking* between the *estimated* and the *actual* energy cost tested on the mobile device. The consistency in ranking suffices to provide accurate feedback to dynamically adjust the video enhancement model's hyperparameters for energy conservation.

4.3 Automated Adaptation Loop

Fig. 11 shows the overall workflow of the energy-aware adaptation controller. The adaptation loop consists of the *resource monitor*, the *runtime energy profiler*, the *analyzer*, and the *optimizer*. The *resource monitor* tracks the energy supply of the platform, while the *energy profiler* (Sec. 4.2) predicts the energy demand of the video enhancement process with the current configurations. If the energy demand exceeds the supply, the *analyzer* notifies the *optimizer*, which then re-selects $\theta_l, \theta_f, \theta_d$ as in Sec. 4.1.2. The control loop routinely checks for changes in the system and performs adaptations at a pre-defined frequency (*e.g.*, 1 second).

Table 2. Overview of low-light image and video datasets.

Datasets	Data type	Sample number	Description
1. SCIE [5]	Images	2002 pieces	Diverse exposure levels
2. Loli-Phone [29]	Videos	300 frames	Taken by real-world smartphones
3. VV [47]	Unpaired images	24 pieces	Low-light images
4. LIME [15]	Unpaired images	10 pieces	Low-light images
5. NightOwls [38]	Unpaired videos	500 piece	Low-light videos
6. LOL [50]	Paired images	500 pairs	Low-light images with the corresponding high-light images
7. MobiScene	Unpaired low-light videos	40 pieces	Self-collected videos with moving/static cameras and moving/static objects

5 EVALUATION

5.1 Experiment Setups

5.1.1 Datasets. We experiment with seven datasets, six open low-light image/video enhancement benchmark datasets (SCIE [5], Loli-Phone [29], LOL [50], VV [47], LIME [15], NightOwls [38]) and one self-collected dataset (MobileScene). Tab. 2 lists the details of these datasets.

We use SICE [5] and Loli-Phone [29] for training. SCIE contains images of different exposure levels. We use it to train the single-frame image's enhancement performance of exposure level, smoothness, neighborhood, and color balance. Loli-Phone contains multiple video clips taken by smartphones, and we adopt it for training the exposure and color consistency of the video enhancement model. Specifically, we use SICE as the image training dataset as Zero-DCE [14], and Loli-Phone as the video training dataset for the fast low-light video enhancement model of AdaEnlight. All the images and video frames are resized as $256 \times 256 \times 3$. We use Adam as the optimizer and train the model for 100 epochs. The learning rate is set to 0.0001 with the batch size of 4.

The remaining datasets are for testing. We consider datasets with both paired and unpaired samples. VV [47] and LIME [15] are unpaired low-light image datasets, NightOwls [38] is an unpaired low-light video dataset, and LOL [50] is a paired low-light image dataset. MobiScene is a self-collected dataset including four scenarios (*i.e.*, moving/static cameras with moving/static objects), which is used to measure the enhanced video's temporal stability. The concrete performance metrics of each dataset are explained in the setups of each experiment below.

5.1.2 Baselines. We compare the performance of AdaEnlight with the following baselines.

- **Zero-DCE** [14]. It is a lightweight low-light image enhancement scheme which adopts zero-shot learning to avoid training with paired data and employs an iterative curve-based architecture to map a low-light image to its enhanced version.
- **Zero-DCE++** [28]. It is the more resource-efficient version of Zero-DCE by compressing the neural network to learn the curve parameters by depth-wise separable convolutions and curve parameter map reuse.
- **MELLEN** [36]. It is a recent low-light image enhancement method that achieves high enhancement quality. It extracts rich features via multiple sub-networks and generates the output with multi-branch fusion.
- **StableLLVE** [59]. It is a state-of-the-art low-light video enhancement model which enforces the temporal stability among frames.
- AdaEnlight **Basic**. It is the variant of AdaEnlight without incorporating the temporal consistency loss.
- AdaEnlight **W/O_Align**. It is the variant of AdaEnlight without the optical flow alignment pre-process.

5.1.3 Implementation. We implement all the compared methods in Pytorch. Training of the algorithms is performed on a server with NVIDIA GeForce RTX 2070 GPU and Intel Core i5-10400F CPU at 2.90GHz and CUDA 10.0. We deploy each method to mobile devices for low-light enhancement. Specifically, we test three commercial mobile camera-embedded platforms, including a personal smartphone, *i.e.*, Honor 9 with Octa-core CPU processor (device 1), an embedded development board, *i.e.*, raspberry Pi 4B with Quad-core CPU (device 2), and a mobile platform, *i.e.*, NVIDIA Jetson Nano with 128-core NVIDIA Maxwell GPU (device 3). For mobile deployment of AdaEnlight, we use the Pytorch Mobile on the Android 12.0 platform (device 1) the NCNN framework

Table 3. Performance comparison of AdaEnlight with four baselines in terms of parameter size, latency, and visual quality.

Methods	Parameter size	Average latency per frame (s)	Output visual quality					
			NIQE ↓			SSIM ↑	PSNR ↑	MAE ↓
			VV (task3)	LIME (task4)	NightOwls (task5)	LOL (task6)		
Zero-DCE [14]	308.6KB	11.743	2.751	3.812	3.541	0.846	14.132	52.172
Zero-DCE++ [28]	39.1KB	0.193	2.547	3.82	3.159	0.864	14.596	49.871
StableLLVE [59]	15.8MB	1.2	2.224	3.725	3.793	0.864	17.142	34.921
MBLLEN [36]	1.7MB	6.225	2.76	3.763	3.333	0.899	17.229	34.296
AdaEnlight	0.17KB	0.05	2.57	3.723	2.882	0.854	17.232	33.784

on Raspberry Pi OS (device 2), and Pytorch 1.4 on Ubuntu 18.04 platform (device 3). For platforms with both mobile CPU and GPU, we configure the computing tasks to execute on either the GPU or CPU to avoid excessive GPU-CPU communication. We use OpenCV to invoke the embedded camera for video shooting at a rate of 20 frames per second. The video frames are resized to 270×480 and fed into each algorithm for enhancement.

5.2 Low-light Video Enhancement Performance

This section presents the results on low-light enhancement compared with the baselines.

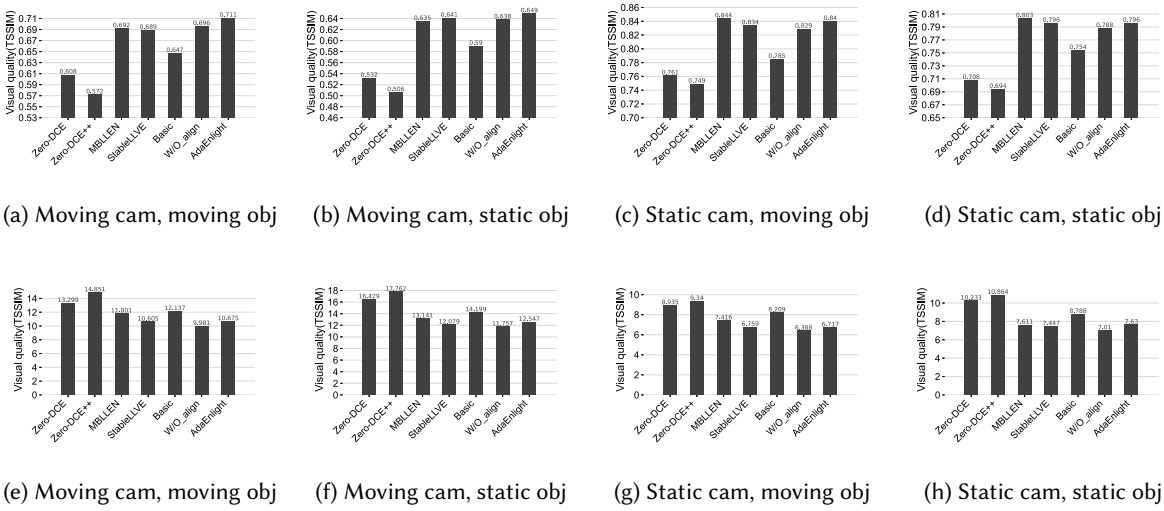
5.2.1 Performance on Open Benchmarks. This experiment compares the performance of AdaEnlight and different baseline methods on open low-light enhancement datasets.

Setups. We use three low-light image datasets, *i.e.*, VV (task3), LIME (task4), and LOL (task6) to assess the enhancement performance on a single frame, and 10 video streams from the low-light video dataset NightOwls (task5). We compare the visual quality of the enhanced images or videos and the latency to process each image or video frame. We use four visual quality metrics, *i.e.*, structural similarity (SSIM)[49], peak signal-to-noise ratio (PSNR), mean absolute error (MAE), and natural image quality evaluator (NIQE)[37]. SSIM, PSNR, and MAE measure the visual disparity of low-light images/videos from the labeled high-quality ones, while NIQE is a no-reference metric to measure the deviations from statistical regularities observed in natural images, without training on paired data. The experiments are conducted on Raspberry Pi 4B (with CPU, device 2). We run the measurements for six times and report the average values.

Results. Tab. 3 summarizes the results. AdaEnlight outperforms the baselines in execution latency. In fact, only AdaEnlight achieves real-time low-light enhancement. The average delay of AdaEnlight is about 0.05s per frame, which is smaller than the input streaming speed (*i.e.*, 0.1s per frame for the 10FPS video stream from NightOwls [38]). In comparison, the delays of StableLIVE, MBLLEN, Zero-DCE, and Zero-DCE++ are 1.2, 6.225s, 11.743s, and 0.193s per frame. The low latency of AdaEnlight attributes to the non-iterative enhancement with Gamma correction-based curve design (see Sec. 3.2). Despite its low latency, AdaEnlight achieves competitive visual quality. Specifically, AdaEnlight achieves the best NIQE on NightOwls (video) and LIME (image), and is at least as good as Zero-DCE++ in the NIQE metric for the VV task [47]. Also, AdaEnlight achieves the best PSNR and MAE on LOL [15]. These outcomes validate the effectiveness of the proposed curve-based enhancement.

5.2.2 Performance on Real-world Mobile Scenarios. This experiment assesses the temporal stability of the videos enhanced by AdaEnlight and different baseline methods across four typical real-world mobile scenarios. Note that both the camera and the objects in videos may be in motion in mobile shooting scenarios, making it challenging for a low-light enhancement algorithm to deliver stable video streams.

Setups. We test four real-world mobile scenarios: *Scenario A:* a moving camera capturing videos of moving objects, *Scenario B:* a moving camera capturing videos of static objects, *Scenario C:* a static camera capturing videos of moving objects, and *Scenario D:* a static camera capturing videos of static objects. We use our self-collected video dataset MobiScene and videos from NightOwls on Raspberry Pi 4B (with CPU, device 2). We quantify the temporal stability of videos by temporal structural similarity index measure (TSSIM) and mean



(a) Moving cam, moving obj (b) Moving cam, static obj (c) Static cam, moving obj (d) Static cam, static obj

(e) Moving cam, moving obj (f) Moving cam, static obj (g) Static cam, moving obj (h) Static cam, static obj

Fig. 12. Temporal stability quantified by TSSIM and MABD, in four mobile scenarios. Scenario A: moving camera and moving objects {(a),(e)}, Scenario B: moving camera and static objects {(b),(f)}, Scenario C: static camera and moving objects {(c),(g)}, and Scenario D: static camera and static objects {(d),(h)}. A high TSSIM and a low MABD mean good temporal stability.

absolute brightness differences (MADB). TSSIM is a common metric for the temporal structural similarity of adjacent frames in the video stream, and MADB measures the pixel difference between adjacent frames.

Results. Fig. 12 compares TSSIM and MABD of different algorithms in all the four scenarios. Compared with prior methods, AdaEnlight achieves the best results in (a), (b), (g) and the second best results in (c), (d), (e), (f). Furthermore, AdaEnlight outperforms its **Basic** variant in all the four scenarios, which validates the necessity of the temporal consistency loss. Compared with the variant **W/O_align**, AdaEnlight achieves better results in the TSSIM metric, although AdaEnlight is slightly worse than **W/O_align** in the MABD metric. The results show that AdaEnlight achieves a better structural-level stability due to the optical flow alignment.

5.2.3 Performance on Downstream On-device Tasks: Face Detection. This experiment shows the performance gains of AdaEnlight for downstream vision tasks such as face detection.

Setups. We use the widely used face detection algorithm, DSFD [30], to detect human faces in the raw low-light frames and those enhanced by Zero-DCE, Zero-DCE++, and AdaEnlight, respectively. We use the dark face [56] dataset to simulate the raw low-light frames. The experiment was conducted on Raspberry Pi 4B (with CPU, device 2) and we measure the precision and recall of face detection.

Results. Fig. 13 compares the precision-recall trade-off of different algorithms. AdaEnlight shows the highest area under the precision-recall curve, representing the best recall and the highest precision. Specifically, the high precision and recall relate to a low false-positive rate and a low false-negative rate. We can conclude that the enhanced video frames can improve facial detection performance than the raw low-light frame AdaEnlight brings better facial detection benefits than Zero-DCE and Zero-DCE++.

5.2.4 Comparison with Commercial Low-light Enhancement Solutions. Tab. 4 summarizes AdaEnlight's main characteristics compared with other commercial low-light enhancement solutions. AdaEnlight requires no special devices, and conducts video enhancement without network dependency.

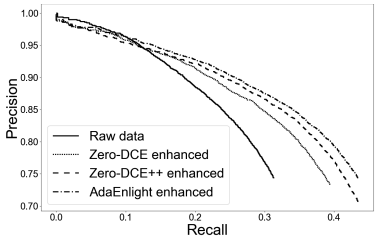
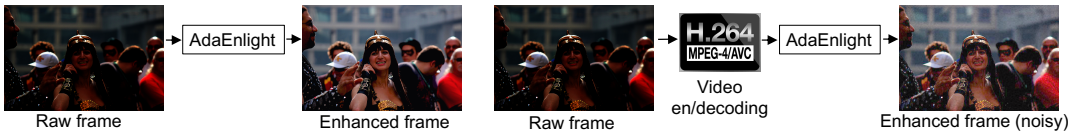


Fig. 13. Performance comparison for the downstream on-device video application of mobile face detection.

Table 4. Comparison of AdaEnlight with other commercial solutions.

Solutions	Special hardware	Enhancement	Internet	Latency	Usage shortcoming
Night vision camera	Infrared camera	On-device	None	Real-time	Lost color
Adobe premiere pro	None	PC	None	0.04s/frame	Expert only
Baidu contrast enhance API	None	Cloud server	Dependent	1.8s/frame	Unstable
iPhone 13 camera	None	On-device	None	1.3s/image	No video support
AdaEnlight	None	On-device	None	0.03s/frame	None



(a) On-device video enhancement (b) Remote video enhancement after internet streaming

Fig. 14. Visual quality comparison of the enhanced frame before and after commercial video encoding technique.

Table 5. AdaEnlight's performance with diverse mobile energy budgets on Honor 9 (with CPU, device 1).

No.	Diverse energy supply	Power (W)	Energy per frame(mJ)	Power cost per video (mAh)	Visual quality (VMAF)	Latency per frame (ms)	Optimal Hyperparameters		
							θ_f	θ_l	θ_d
1	89%	4.1	200	1.7	99.8	49	0	0	1
2	78%	3.8	191	1.5	93.2	47	1	1	1
3	69%	3.7	182	1.4	90.2	45	1	1	1/2
4	50%	3.5	175	1.3	86.2	42	2	2	1
5	48%	3.2	138	1.1	73.7	37	3	1	1/2
6	32%	3.2	93	0.8	72.1	30	3	3	1/3

Fig. 14 illustrates that the output visual quality of locally enhanced video, before encoding (*e.g.*, H.264 encoding), is better than the remotely enhanced one, with the intermediate encoding/decoding interference. The video encoding (*e.g.*, H.264 encoding) technique is widely used to stream video over the internet. It verifies the *visual benefit of on-device video enhancement*, demonstrating the need for on-device video enhancement. Specifically, the natural image quality metric (NIQE) value of the locally enhanced and remotely enhanced video frame is 3.0 and 3.7, respectively. The smaller the NIQE score, the higher the quality.

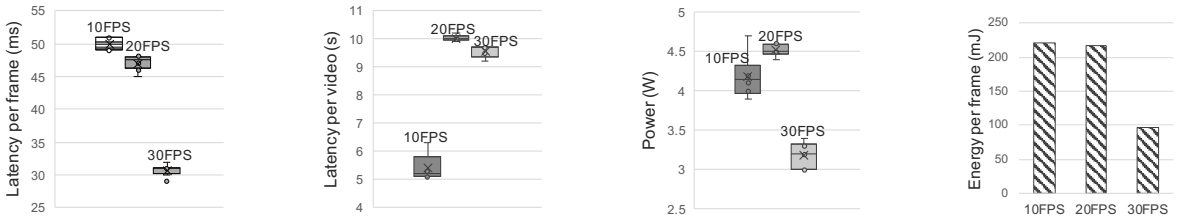
5.3 System Performance

This subsection presents the evaluations of AdaEnlight in terms of various system performance metrics.

5.3.1 Adaptation to Dynamic Energy Budgets of Mobile Platforms. This experiment evaluates how AdaEnlight optimizes visual quality under different energy budget. Given a specific energy budget, AdaEnlight's controller

Table 6. AdaEnlight's performance with diverse mobile energy budgets on Jetson Nano (with GPU, device 3).

No.	Diverse energy supply	Power (W)	Energy per frame(mJ)	Power cost per video (mAh)	Visual quality (VMAF)	Latency per frame (ms)	Optimal Hyperparameters		
							θ_f	θ_l	θ_d
1	95%	3.3	82.3	0.68	100	25	0	0	1
2	81%	3.2	66.2	0.55	96.9	20	1	1	1
3	76%	3.2	57.6	0.48	94.2	18	4	1	1
4	40%	3.0	48.1	0.40	75.1	12	0	0	1/2



(a) Latency per frame (b) Latency per video (c) Power (d) Energy cost per frame

Fig. 15. AdaEnlight's performance across diverse inputs with varying video frame rates (FPS), *i.e.*, 10 FPS, 20FPS, and 30FPS.

actively adjusts its behaviors (*i.e.*, the frame-level computation reuse, the layer-level computation reuse, and the frame resolution) by re-selecting the suitable hyperparameters θ_f, θ_l, and θ_d.

Setups. We leverage the NightOwls video dataset (task 5) for evaluation. We normalize the resolution of all video frames into the same size, *i.e.*, 270P, for fair comparison. And we sample diverse stations of energy supply (*i.e.*, the remaining battery) on Honor 9 (with CPU, device 1) and Jetson Nano (with GPU, device 3), to evaluate AdaEnlight's hyperparameter selection on computation reuse and down-sampling. The computation reuse and down-sampling operations for energy conservation will bring about a decline in visual quality. Thus, we adopt a full-reference video quality metric, *i.e.*, video multimethod assessment fusion (VMAF) to quantify the quality decrease of computation reuse in different levels.

Results. Tab. 5 and Tab. 6 summarize the performance and the corresponding hyperparameters with different energy supply. We make the following observations. First, AdaEnlight can select suitable hyperparameters with various energy budgets. For example, at 78% energy supply on Honor 9 (with CPU, device 1), the optimal hyperparameter is to reuse 1 frame and 1 layer without down-sampling. At 48% energy supply, it becomes reusing 3 frames and 3 layers and down-sampling by 1/3. Second, different hyperparameters setups of θ_f, θ_l, and θ_d can dynamically tune the trade-off between the output video quality and the energy consumption. Third, the computation reuse and down-sampling behaviors can further reduce the execution latency of the video enhancement model, ranging from $49ms$ to $30ms$ per frame. Therefore, all these hyperparameters ensure real-time constraints. Finally, for completeness, we run AdaEnlight on Honor 9 (device 1, with 3200 mAH battery) with different computation reuse parameters to simulate diverse user preference on visual quality and energy consumption. Experiments show that AdaEnlight will drain the battery in $2.5h \sim 4h$ under these settings.

5.3.2 Adaptation to Input Video Frame Rates. We now test AdaEnlight's performance over diverse video streams.

Setups. We use 20 video clips of $10s$ duration with three different video frame rates, *i.e.*, 10 frame per second (FPS), 20FPS, and 30 FPS. We test the AdaEnlight's execution latency on Honor 9 (with CPU, device 1). We use the digital power monitor of Monsoon AAA10F to power the smartphone and measure its energy cost.

Results. Fig. 15 shows the performance of AdaEnlight across diverse input video streams. First, AdaEnlight achieves real-time processing for all frame rates. Specifically, for the 10FPS, 20FPS, and 30FPS input video streams, the average processing latency per frame is $50ms$, $47ms$, and $31ms$, respectively. Furthermore, the average delay of AdaEnlight to process the entire video clip with these three frame rates is $\leq 10s$, which is smaller than the

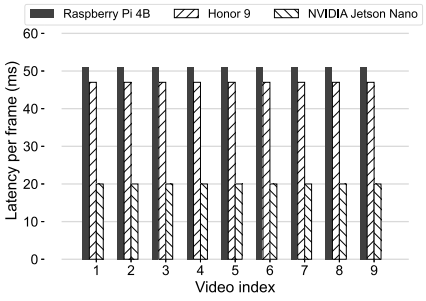


Fig. 16. AdaEnlight's performance on three typical camera-embedded platforms.

(a) $Ite = 1$, Zero-DCE++ (b) $Ite = 8$, Zero-DCE++ (c) $Ite = 1$, AdaEnlight (d) $Ite = 8$, AdaEnlight

Fig. 17. Impact of iteration numbers Ite on visual quality by Zero-DCE++ and our Gamma correction-based curve.

streaming speed of the input video, thereby ensuring real-time video stream enhancement. Second, the energy consumption of AdaEnlight on the Honor 9 platform is adjustable for diverse input videos with the same energy budget. Specifically, the energy cost of AdaEnlight for the 10FPS, 20FPS, and 30FPS input videos is $227mJ$, $158mJ$, and $97mJ$, respectively. For video streams over 30FPS, we can further reduce the execution latency and energy consumption by adjusting the computation reuse and frame resolution hyperparameters.

5.3.3 Performance on Diverse Mobile Platforms. This experiment evaluates AdaEnlight on different devices.

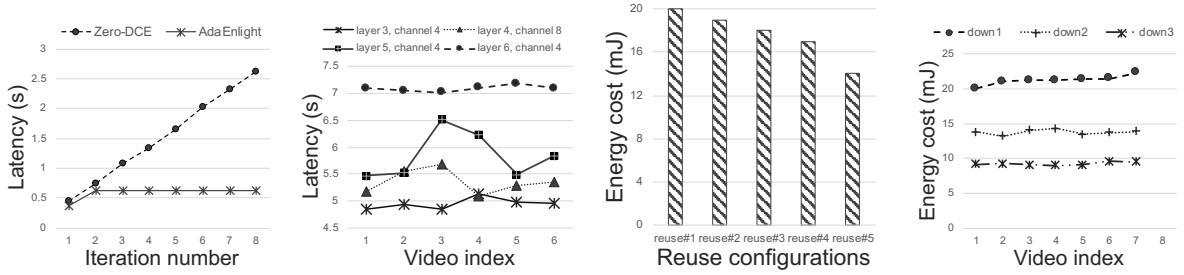
Setups. We test three platforms: Honor 9 (using CPU, device 1), Raspberry Pi 4B (with CPU, device 2), and NVIDIA Jetson Nano (with GPU, device 3). Different devices have diverse resource availability (*e.g.*, computation throughput and memory hierarchy) and frameworks (Pytorch mobile, NCNN, Pytorch), which lead to different execution latency. We use the NightOwls dataset to extract video samples as the input of AdaEnlight. We use nine input videos of 10s at 270P, and a frame rate of 10FPS for testing to obtain the overall processing delay.

Results. Fig. 16 compares the AdaEnlight performance on these three different platforms. On all three resource-constrained mobile devices, AdaEnlight achieves real-time processing with $20ms \sim 51.5ms$ execution latency.

5.4 Ablation Studies

This section evaluates the impact of various setups on the AdaEnlight's performance.

5.4.1 Iteration Number in Gamma Correction-based Curve. As discussed in Sec. 3.2, we propose the non-iterative enhancement method to break the latency bottleneck of traditional curve-based image enhancement. As shown in Fig. 17, the visual quality enhanced by AdaEnlight does not notably increase with more iterations. Thus, we set the iteration number as 1, which is equivalent to a non-iterative model. Fig. 18a compares the execution latency of AdaEnlight's enhancement model and Zero-DCE with different iteration numbers in their curve functions. AdaEnlight avoids the long latency due to the iterative enhancement process in Zero-DCE.



(a) Iteration number (b) Architecture (c) Reuse computation (d) frame resolution

Fig. 18. Benchmark of various parameters, *i.e.*, (a) iteration number, (b) layer and channel number, (c) frame-level and layer-level computation reuse, and (d) frame resolution, on AdaEnlight's performance.

5.4.2 Layer and Channel Number. To tune the architecture hyperparameters (*i.e.*, the number of layer and channel) of AdaEnlight's enhancement model, we compare the performance of different architecture settings. We randomly selected six video segmentations from NightOwls dataset with the frame resolution of 270×480 for testing, each video has a duration of 10S, a frame rate of 10FPS, and a frame resolution of 270×480. Fig. 18b illustrates the model's execution latency with four settings. All the settings result in the competitive visual quality, while 3 layers with 4 channels has the lowest latency on Raspberry Pi 4B (with CPU, device 2). Thus, we set the layer number as 3 and the channel number as 6 in AdaEnlight's enhancement model by default.

5.4.3 Impact of Computation Reuse. As discussed in Sec. 3.4.3, due to temporal similarities between adjacent video frames, we can dynamically adjust the model complexity by enforcing computation reuse. Fig. 18c illustrates the energy consumption of AdaEnlight's video enhancement model on Honor 9 (with CPU, device 1), under different frame and layer reuse settings. We compare five options, *i.e.*, reuse#1 (frame = 1 layer = 1), reuse#2 (frame = 1, layer = 2), reuse#3 (frame = 1, layer = 3), and reuse#4 (frame = 2, layer = 2), and reuse#5 (frame = 2, layer = 3). The outcomes verify that these computation reuse hyperparameters can effectively tune the model's energy cost. It is worth mentioning that such computation reuse also reduces the overall latency by 10% $\sim$ 50%. For example, given a fixed frame-level computation reuse parameter 5, if we set layer-level computation reuse parameter as 1,2 and 3, the latency decreases by 12.5%, 35.1% and 57.6%, respectively. However, since AdaEnlight without computation reuse already achieves near real-time processing *i.e.*, 50ms per frame (shown in Tab. 3), a reduction of 50% in delay does not introduce perceivable improvement in user experience. Therefore, we mainly focus on the improvement in energy consumption due to computation reuse.

5.4.4 Impact of Frame Resolution. This experiment verifies the effect of input frame resolution on the energy consumption of AdaEnlight's video enhancement, on Honor 9 (with CPU, device 1). Fig. 18d compares the energy cost of three down-sampling levels (*i.e.*, $1 \sim 3$), representing the full resolution, down to $1/2$ resolution, and down to $1/3$ resolution, respectively. The energy consumption increases linearly with the frame resolution. Thus, down-sampling the resolution of the input video frame can naturally tune the model's energy demand.

5.4.5 Generalization of Energy Profiler. This experiment demonstrates the cross-time and cross-model generalization performance. We refer to the evaluation methodology in nn-Meter [60]. We use the existing energy profiler trained for Honor 9 (with CPU, device 1) and Jetson Nano (with GPU, device 3) to predict model energy consumption for four different times and three models. We evaluate our energy profiling for CPU at time 1 and time 2 with model 1 and model 2; while for GPU at time 3 and time 4 with model 1 and model 3. At different

Table 7. Cross-time and cross-model energy profiler measurement on Honor 9 (with CPU, device 1).

Cross-time evaluation					
Prediction at time 1 (resource state 1)	**Measurement at time 1 (resource state 1)**	**Error**	**Prediction at time 2 (resource state 2)**	**Measurement at time 2 (resource state 2)**	**Error**
0.192mJ	0.2mJ	4%	0.141mJ	0.138mJ	1.4%
Cross-model evaluation					
Prediction for model 1 (4 layers)	**Measurement for model 1 (4 layers)**	**Error**	**Prediction for model 2 (5 layers)**	**Measurement for model 2 (5 layers)**	**Error**
0.285mJ	0.27mJ	5.6%	0.368mJ	0.38mJ	3.2%

Table 8. Cross-time and cross-model energy profiler measurement on Jetson Nano (with GPU, device 3).

Cross-time evaluation					
Prediction at time 3 (resource state 1)	**Measurement at time 3 (resource state 1)**	**Error**	**Prediction at time 4 (resource state 2)**	**Measurement at time 4 (resource state 2)**	**Error**
0.36mJ	0.38mJ	5.6%	0.541mJ	0.574mJ	6.1%
Cross-model evaluation					
Prediction for model 1 (4 layers)	**Measurement for model 1 (4 layers)**	**Error**	**Prediction for model 3 (3 layers)**	**Measurement for model 3 (3 layers)**	**Error**
0.129mJ	0.128mJ	0.8%	0.090mJ	0.093mJ	3.3%

times, the available execution resources of the platform differ, which will affect the cache hit rate and thus the energy demand. The three models have different architectures: model 1 has 3 Conv and activation layers, model 2 has 4 Conv and activation layers, and model 3 has 2 Conv and activation layers. As shown in Tab. 7 and Tab. 8, the energy profiler achieves $\geq$ 93.9% accuracy for cross-time prediction, and $\geq$ 94.4% for cross-model prediction. Importantly, the results show that our energy profiler ensures consistent ranking between the estimated and the actual energy cost, which is crucial for the controller to adapt the hyperparameters to the energy demand.

5.5 Summary of Main Experimental Results

We summarize the main results of our evaluations as follows.

- *Near real-time processing on diverse mobile devices.* AdaEnlight's enhancement model achieves near real-time video enhancement on different mobile devices, *e.g.*, 51 ms per frame on Raspberry Pi 4B, 47 ms per frame on Honor 9, and 20 ms per frame on NVIDIA Jetson Nano. This is because the proposed non-iterative model breaks the latency bottleneck of the state-of-the-art enhancement schemes.
- *Better trade-off between processing delay and visual quality.* AdaEnlight's enhancement model achieves the lower video enhancement latency while retaining competitive visual quality. AdaEnlight's latency is only 1/4 of Zero-DCE++ [28], the fastest low-light enhancement scheme. Meanwhile, AdaEnlight still yields high visual quality. For example, AdaEnlight achieves a PSNR of 17.232, which is almost the same as MBLLEN [36], the state-of-the-art video enhancement solution (with a PSNR of 17.229). AdaEnlight consistently outputs stable and high-quality videos on various datasets (LOL, NightOwls, LIME and VV), and different scenarios (static/mobile camera, static/mobile objects). This is mainly due to the enforced temporal consistency between video.
- *Adaptive to diverse energy supply.* AdaEnlight's energy-aware controller is able to adapt the enhancement quality according to the energy budget at runtime. As a concrete example, when the energy supply is reduced from 95% to 40%, the power consumption of the AdaEnlight is reduced from 0.68mAh to 0.4mAh (see Tab. 8). It proves the necessity of the energy-aware controller in mobile systems.

6 RELATED WORK

6.1 Low-light Image and Video Enhancement

Low-light enhancement improves the perception or interpretability of images and videos captured in dim light [18, 29]. Most prior efforts focus on image enhancement [14, 15, 17, 20, 24, 28] while video enhancement [6, 19, 36, 59] is regarded as an extension of image enhancement by avoiding the flicking problem between frames. We refer readers to [29] for a comprehensive review. Of our particular interest is Zero-DCE [14] and its follow-up, Zero-DCE++ [28], a state-of-the-art zero-shot learning based low-light image enhancement scheme. The reasons are two-fold. *(i)* There lacks paired training data *i.e.*, low-light images or videos and the corresponding well-illuminated versions, in real-world mobile video applications. Therefore, zero-shot learning based solutions [14, 28] are preferable over supervised learning ones [6, 35] since the former requires neither paired nor unpaired training data. *(ii)* Zero-DCE [14] and Zero-DCE++ [28] apply a model architecture that consists of a deep neural network and an image-to-curve mapping, making it one of the most lightweight solution while achieving competitive image enhancement performance.

The primary challenge to extend low-light enhancement from images to videos is the flicking problem [29]. Some studies implicitly solve the problem by substituting the 2D network architectures in image enhancement models to the corresponding 3D versions [19, 36]. However, they require specialized equipment to collect video datasets for training the new 3D architectures. A more practical strategy is to incorporate temporal consistency into the loss function of image enhancement models [6, 59]. For example, Chen *et al.* [6] proposed a self-consistency loss to tolerate minor differences between inputs while keeping the output stable. Zhang *et al.* [59] enforces temporal consistency by ensuring stable optical flows estimated from the videos.

To this end, the low-light video enhancement module of AdaEnlight extends existing low-light enhancement solutions from images to videos. And AdaEnlight advances the prior arts by introducing novel designs. *(i)* It notably reduces the processing time of low-light enhancement schemes, by replacing the iterative image-curve mapping using a non-iterative mapping function. For example, AdaEnlight dramatically outperforms Zero-DCE [14] in terms of processing latency, one of the most lightweight low-light image enhancement method. *(ii)* It designs the novel temporal consistency loss and its training scheme compatible to zero-shot learning based image enhancement schemes without the need for large-scale paired video datasets.

6.2 Energy-awareness in DNNs

Existing DNN energy profiling methods couple with either the DNN topology (*e.g.*, model structure, computation path) [33] or the hardware platform (*e.g.*, on-chip memory capability, memory bandwidth) [25, 55]. Tien-Ju *et al.* [54] propose a layer-wise estimation method, which takes approximately 10 seconds to output the normalized energy consumption of a DNN based on its architecture, sparsity, and bitwidth. Yannan *et al.* [53]present the hardware architecture-dependent energy profiling method. Some other works generate the design-specific tables with predefined attributes to estimate DNN energy cost [22, 39]. However, the above methods cannot profile the DNN energy usage for providing exact runtime feedback on *dynamic resource availability* with *agnostic DNN topology*, because the sampling for the specific DNN prototype or platform is costly.

Unlike existing methods, AdaEnlight builds the one-fits-all profiler for DNN energy usage based on the following two observations: *(i)* despite the dynamic topology of video processing models, the underlying tensor operation (*i.e.*, the energy-dominating operation loops) and the data flow are stable. *(ii)* Despite the dynamic hardware resource supply (*e.g.*, on-chip memory capability, computation bandwidth), the hit ratio of on-chip memory access shows up the stable rules and within a small sampling set.

6.3 Adaptive Vision Tasks on Mobile Devices

Adaptive vision tasks involve image/video delivery between distributed mobiles or between the mobile and cloud. Examples include live video streaming [23], video-on-demand [3], and virtual reality [48]. The key challenge for such tasks is the reliance on networking conditions. To tackle this challenge, researchers propose two categories of solutions, *i.e.*, adaptive bitrate [8, 44, 48], and super-resolution [26, 34, 58]. They co-design the task demands (*e.g.*, visual quality, latency, or inference accuracy) with the streaming process. In bitrate adaptation, each video is split into segments. These segments are encoded using multiple bitrates and streamed with the suitable bitrate based on the network conditions [48]. In super-resolution scheme, compact content with low resolution or quality is transmitted, which is then enhanced at the receiver [26, 58]. For example, Yeo *et al.* [57] leverages the super-resolution DNNs to enable the content-aware video delivery. Lee *et al.* [26] present MobiSR to boost the performance of on-device super-resolution.

Nevertheless, the concerns of these efforts are coupled with other edge or cloud devices. AdaEnlight serves as a locally deployed middleware to directly provide video enhancement services for local applications.

7 CONCLUSION

This paper presents AdaEnlight, an energy-aware low-light video enhancement system that achieves near real-time performance with competitive enhanced visual quality. It consists of a novel model for fast low-light video enhancement as well as an agile energy-aware controller to dynamically adjust the enhancement behaviors for energy conservation. And experimental results show that AdaEnlight outperforms existing image and video enhancement methods in latency and temporal stability while retaining satisfactory system energy efficiency.

There are several limitations of our work that may need future exploration. *(i)* Although AdaEnlight achieves near real-time processing for RGB videos, the frames are resized to $270 \times 480 \times 3$. To support videos with higher resolutions, a more lightweight and efficient video enhancement model is necessary. We plan to exploit the fixed-point operations common on mobile platforms and explore quantization and other model compression techniques to further accelerate the video enhancement model. An interesting follow-up is to extend AdaEnlight to videos in raw format to handle extremely low-light conditions. *(ii)* The proposed energy profiler can be integrated into other mobile systems. The energy profiler of AdaEnlight is built for mobile GPUs or CPUs only for tractability. Also, more efforts are needed to design the modular and extensible energy profiler for devices with heterogeneous computation resources, *e.g.*, GPU-CPU co-execution.

ACKNOWLEDGMENTS

This work was partially supported by the National Key RD Program of China (2019YFB1703901), National Science Fund for Distinguished Young Scholars (62025205, 61725205), the National Natural Science Foundation of China (No. 62102317, 62032020), and the China Postdoctoral Science Foundation (No. 2021M702671). The authors also thank the anonymous reviewers for their constructive feedback that has made the work stronger.

REFERENCES

[1] Abdulkareem Sh Mahdi Al-Obaidi, Arif Al-Qassar, Ahmed R Nasser, Ahmed Alkhayyat, Amjad J Humaidi, and Ibraheem K Ibraheem. 2021. Embedded design and implementation of mobile robot for surveillance applications. *Indonesian Journal of Science and Technology* 6, 2 (2021), 427–440.

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

[3] Ghufran Baig, Jian He, Mubashir Adnan Qureshi, Lili Qiu, Guohai Chen, Peng Chen, and Yinliang Hu. 2019. Jigsaw: Robust live 4k video streaming. In *Proceedings of the International Conference on Mobile Computing and Networking*. ACM, New York, NY, USA, 1–16.

[4] Luca Benini and Giovanni de Micheli. 2000. System-level power optimization: techniques and tools. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 5, 2 (2000), 115–192.

[5] Jianrui Cai, Shuhang Gu, and Lei Zhang. 2018. Learning a deep single image contrast enhancer from multi-exposure images. *IEEE Transactions on Image Processing* 27, 4 (2018), 2049–2062.

[6] Chen Chen, Qifeng Chen, Minh N Do, and Vladlen Koltun. 2019. Seeing motion in the dark. In *Proceedings of the International Conference on Computer Vision*. IEEE, Piscataway, NJ, USA, 3185–3194.

[7] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, et al. 2018. {TVM}: An Automated {End-to-End} Optimizing Compiler for Deep Learning. In *Proceedings of USENIX Symposium on Operating Systems Design and Implementation*. USENIX, Berkeley, CA, USA, 578–594.

[8] Xianda Chen, Tianxiang Tan, and Guohong Cao. 2019. Energy-aware and context-aware video streaming on smartphones. In *Proceedings of the International Conference on Distributed Computing Systems*. IEEE, Piscataway, NJ, USA, 861–870.

[9] etal Eric S. Yuan, Ryan Azus. 2021. Zoom Video Communications APP. https://support.zoom.us/hc/en-us/sections/200305413-Mobile.

[10] Boyu Fan, Xuefeng Liu, Xiang Su, Pan Hui, and Jianwei Niu. 2020. Emgauth: An emg-based smartphone unlocking system using siamese network. In *Proceedings of the International Conference on Pervasive Computing and Communications*. IEEE, Piscataway, NJ, USA, 1–10.

[11] Marco Farina, Kalyanmoy Deb, and Paolo Amato. 2004. Dynamic multiobjective optimization problems: test cases, approximations, and applications. *IEEE Transactions on Evolutionary Computation* 8, 5 (2004), 425–442.

[12] Gunnar Farnebäck. 2003. Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on Image analysis*. Springer, 363–370.

[13] Jason Flinn and Mahadev Satyanarayanan. 1999. Energy-aware adaptation for mobile applications. In *Proceedings of the Symposium on Operating Systems Principles*. ACM, New York, NY, USA, 48–63.

[14] 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 *Proceedings of the Conference on Computer Vision and Pattern Recognition*. IEEE, Piscataway, NJ, USA, 1780–1789.

[15] Xiaojie Guo, Yu Li, and Haibin Ling. 2016. LIME: Low-light image enhancement via illumination map estimation. *IEEE Transactions on Image Processing* 26, 2 (2016), 982–993.

[16] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*. IEEE, Piscataway, NJ, USA, 4700–4708.

[17] Haidi Ibrahim and Nicholas Sia Pik Kong. 2007. Brightness preserving dynamic histogram equalization for image contrast enhancement. *IEEE Transactions on Consumer Electronics* 53, 4 (2007), 1752–1758.

[18] Ishani Janveja, Akshay Nambi, Shruthi Bannur, Sanchit Gupta, and Venkat Padmanabhan. 2020. Insight: monitoring the state of the driver in low-light using smartphones. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 3 (2020), 1–29.

[19] Haiyang Jiang and Yinqiang Zheng. 2019. Learning to see moving objects in the dark. In *Proceedings of the International Conference on Computer Vision*. IEEE, Piscataway, NJ, USA, 7324–7333.

[20] Daniel J Jobson, Zia-ur Rahman, and Glenn A Woodell. 1997. A multiscale retinex for bridging the gap between color images and the human observation of scenes. *IEEE Transactions on Image Processing* 6, 7 (1997), 965–976.

[21] Shih-Yao Juang and Jih-Gau Juang. 2012. Real-time indoor surveillance based on smartphone and mobile robot. In *Proceedings of the International Conference on Industrial Informatics*. IEEE, Piscataway, NJ, USA, 475–480.

[22] Liu Ke, Xin He, and Xuan Zhang. 2018. Nnest: Early-stage design space exploration tool for neural network inference accelerators. In *Proceedings of the International Symposium on Low Power Electronics and Design*. ACM, New York, NY, USA, 1–6.

[23] Jaehong Kim, Youngmok Jung, Hyunho Yeo, Juncheol Ye, and Dongsu Han. 2020. Neural-enhanced live streaming: Improving live video ingest via online learning. In *Proceedings of the SIGCOMM Conference*. ACM, New York, NY, USA, 107–125.

[24] Chulwoo Lee, Chul Lee, and Chang-Su Kim. 2013. Contrast enhancement based on layered difference representation of 2D histograms. *IEEE Transactions on Image Processing* 22, 12 (2013), 5372–5384.

[25] Jinsu Lee, Sanghoon Kang, Jinmook Lee, Dongjoo Shin, Donghyeon Han, and Hoi-Jun Yoo. 2020. The hardware and algorithm co-design for energy-efficient DNN processor on edge/mobile devices. *IEEE Transactions on Circuits and Systems I* 67, 10 (2020), 3458–3470.

[26] 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 *Proceedings of the International Conference on Mobile Computing and Networking*. ACM, New York, NY, USA, 1–16.

[27] Seulki Lee and Shahriar Nirjon. 2020. SubFlow: A dynamic induced-subgraph strategy toward real-time DNN inference and training. In *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium*. IEEE, Piscataway, NJ, USA, 15–29.

[28] Chongyi Li, Chunle Guo, and Change Loy Chen. 2021. Learning to enhance low-light image via zero-reference deep curve estimation. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 0, 01 (2021), 1–1.

[29] Chongyi Li, Chunle Guo, Ling-Hao Han, Jun Jiang, Ming-Ming Cheng, Jinwei Gu, and Chen Change Loy. 2021. Low-light image and video enhancement using deep learning: a survey. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 0, 01 (2021), 1–1.

[30] Jian Li, Yabiao Wang, Changan Wang, Ying Tai, Jianjun Qian, Jian Yang, Chengjie Wang, Jilin Li, and Feiyue Huang. 2019. DSFD: dual shot face detector. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*. IEEE, Piscataway, NJ, USA, 5060–5069.

[31] Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, and Megha Manohara. 2016. Toward a practical perceptual video quality metric.

[32] Mason Liu and Menglong Zhu. 2018. Mobile video object detection with temporally-aware feature maps. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*. IEEE, Piscataway, NJ, USA, 5686–5695.

[33] Sicong Liu, Yingyan Lin, Zimu Zhou, Kaiming Nan, Hui Liu, and Junzhao Du. 2018. On-demand deep model compression for mobile devices: A usage-driven model selection framework. In *Proceedings of the Annual International Conference on Mobile Systems, Applications, and Services*. ACM, New York, NY, USA, 389–400.

[34] Xin Liu, Yuang Li, Josh Fromm, Yuntao Wang, Ziheng Jiang, Alex Mariakakis, and Shwetak Patel. 2021. SplitSR: An end-to-end approach to super-resolution on mobile devices. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 1 (2021), 1–20.

[35] Kin Gwn Lore, Adedotun Akintayo, and Soumik Sarkar. 2017. LLNet: A deep autoencoder approach to natural low-light image enhancement. *Pattern Recognition* 61 (2017), 650–662.

[36] Feifan Lv, Feng Lu, Jianhua Wu, and Chongsoon Lim. 2018. MBLLEN: Low-Light Image/Video Enhancement Using CNNs.. In *Proceedings of the British Machine Vision Conference*. The BMVA, Durham, UK, 220.

[37] Anish Mittal, Rajiv Soundararajan, and Alan C Bovik. 2012. Making a "completely blind" image quality analyzer. *IEEE Signal processing letters* 20, 3 (2012), 209–212.

[38] Lukáš Neumann, Michelle Karg, Shanshan Zhang, Christian Scharfenberger, Eric Piegert, Sarah Mistr, Olga Prokofyeva, Robert Thiel, Andrea Vedaldi, Andrew Zisserman, and Bernt Schiele. 2018. NightOwls: A pedestrians at night dataset. In *Asian Conference on Computer Vision*. Springer, Berlin, Germany, 691–705.

[39] Angshuman Parashar, Priyanka Raina, Yakun Sophia Shao, Yu-Hsin Chen, Victor A Ying, Anurag Mukkara, Rangharajan Venkatesan, Brucek Khailany, Stephen W Keckler, and Joel Emer. 2019. Timeloop: A systematic approach to dnn accelerator evaluation. In *Proceedings of the International Symposium on Performance Analysis of Systems and Software*. IEEE, Piscataway, NJ, USA, 304–315.

[40] Shanto Rahman, Md Mostafijur Rahman, Mohammad Abdullah-Al-Wadud, Golam Dastegir Al-Quaderi, and Mohammad Shoyaib. 2016. An adaptive gamma correction for image enhancement. *EURASIP Journal on Image and Video Processing* 2016, 1 (2016), 1–13.

[41] Wenqi Ren, Sifei Liu, Lin Ma, Qianqian Xu, Xiangyu Xu, Xiaochun Cao, Junping Du, and Ming-Hsuan Yang. 2019. Low-light image enhancement via a deep hybrid network. *IEEE Transactions on Image Processing* 28, 9 (2019), 4364–4375.

[42] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, Berlin, Germany, 234–241.

[43] Inchul Song, Hyun-Jun Kim, and Paul Barom Jeon. 2014. Deep learning for real-time robust facial expression recognition on a smartphone. In *Proceedings of the International Conference on Consumer Electronics*. IEEE, Piscataway, NJ, USA, 564–567.

[44] Yi Sun, Xiaoqi Yin, Junchen Jiang, Vyas Sekar, Fuyuan Lin, Nanshu Wang, Tao Liu, and Bruno Sinopoli. 2016. CS2P: Improving video bitrate selection and adaptation with data-driven throughput prediction. In *Proceedings of the SIGCOMM Conference*. ACM, New York, NY, USA, 272–285.

[45] T-Mobile. 2022. Internet Services | T-Mobile's Broadband Internet Access Services. https://www.t-mobile.com/responsibility/consumer-info/policies/internet-service

[46] Shivani Rajendra Teli, Stanislav Zvanovec, and Zabih Ghassemlooy. 2019. The first tests of smartphone camera exposure effect on optical camera communication links. In *Proceedings of the International Conference on Telecommunications*. IEEE, Piscataway, NJ, USA, 1–6.

[47] Vasileios Vonikakis. 2021. California-ND: An annotated dataset for near-duplicates in personal photo-collections. https://sites.google.com/site/vonikakis/datasets.

[48] Yiding Wang, Weiyan Wang, Junxue Zhang, Junchen Jiang, and Kai Chen. 2019. Bridging the edge-cloud barrier for real-time advanced vision analytics. In *Proceedings of the Workshop on Hot Topics in Cloud Computing*. USENIX, Berkeley, CA, USA, 1–7.

[49] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.

[50] Chen Wei, Wenjing Wang, Wenhan Yang, and Jiaying Liu. 2018. Deep retinex decomposition for low-light enhancement.

[51] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. 2003. Overview of the H. 264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology* 13, 7 (2003), 560–576.

[52] Jasper S Wijnands, Jason Thompson, Kerry A Nice, Gideon DPA Aschwanden, and Mark Stevenson. 2020. Real-time monitoring of driver drowsiness on mobile platforms using 3D neural networks. *Neural Computing and Applications* 32, 13 (2020), 9731–9743.

[53] Yannan Nellie Wu, Joel S Emer, and Vivienne Sze. 2019. Accelergy: An architecture-level energy estimation methodology for accelerator designs. In *Procceedings of the International Conference on Computer-Aided Design*. IEEE, Piscataway, NJ, USA, 1–8.

[54] Tien-Ju Yang, Yu-Hsin Chen, Joel Emer, and Vivienne Sze. 2017. A method to estimate the energy consumption of deep neural networks. In *Proceedings of the asilomar conference on signals, systems, and computers*. IEEE, Piscataway, NJ, USA, 1916–1920.

[55] Tien-Ju Yang, Yu-Hsin Chen, and Vivienne Sze. 2017. Designing energy-efficient convolutional neural networks using energy-aware pruning. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*. IEEE, Piscataway, NJ, USA, 5687–5695.

[56] Wenhan Yang, Ye Yuan, Wenqi Ren, Jiaying Liu, Walter J Scheirer, Zhangyang Wang, Taiheng Zhang, Qiaoyong Zhong, Di Xie, Shiliang Pu, et al. 2020. Advancing image understanding in poor visibility environments: A collective benchmark study. *IEEE Transactions on Image Processing* 29 (2020), 5737–5752.

[57] Hyunho Yeo, Sunghyun Do, and Dongsu Han. 2017. How will deep learning change internet video delivery?. In *Proceedings of the Workshop on Hot Topics in Networks*. ACM, New York, NY, USA, 57–64.

[58] Juheon Yi, Seongwon Kim, Joongheon Kim, and Sunghyun Choi. 2020. Supremo: Cloud-Assisted Low-Latency Super-Resolution in Mobile Devices. *IEEE Transactions on Mobile Computing* 0 (2020), 1–1.

[59] Fan Zhang, Yu Li, Shaodi You, and Ying Fu. 2021. Learning Temporal Consistency for Low Light Video Enhancement From Single Images. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*. IEEE, Piscataway, NJ, USA, 4967–4976.

[60] Li Lyna Zhang, Shihao Han, Jianyu Wei, Ningxin Zheng, Ting Cao, Yuqing Yang, and Yunxin Liu. 2021. nn-Meter: towards accurate latency prediction of deep-learning model inference on diverse edge devices. In *Proceedings of the Annual International Conference on Mobile Systems, Applications, and Services*. ACM, New York, NY, USA, 81–93.

[61] Pengfei Zhu, Dawei Du, Longyin Wen, Xiao Bian, Haibin Ling, Qinghua Hu, Tao Peng, Jiayu Zheng, Xinyao Wang, Yue Zhang, et al. 2019. Visdrone-vid2019: The vision meets drone object detection in video challenge results. In *Proceedings of the International Conference on Computer Vision Workshops*. IEEE, Piscataway, NJ, USA, 0–0.