https://doi.org/10.2352/EI.2025.37.15.AVM-109

FlexEye - Application Specific Quality-Scalable ISP Tuning

***Sumbal Akram*[1]*, Muhammad Abdullah*[2]*, Khuzaeymah Nasir*[2]*, Shaharyar Yaqub*[2]*, Rehan Ahmed*[1] *Rehan Hafiz*[1]**
[1]*Information Technology University, Lahore, Pakistan*
[2]*10x Engineers, Irvine, CA, USA*

Abstract

As AI becomes more prevalent, edge devices face challenges due to limited resources and the high demands of deep learning (DL) applications. In such cases, quality scalability can offer significant benefits by adjusting computational load based on available resources. Traditional Image-Signal-Processor (ISP) tuning methods prioritize maximizing intelligence performance, such as classification accuracy, while neglecting critical system constraints like latency and power dissipation. To address this gap, we introduce FlexEye, an application-specific, quality-scalable ISP tuning framework that leverages ISP parameters as a control knob for quality of service (QoS), enabling trade-off between quality and performance. Experimental results demonstrate up to 6% improvement in Object Detection accuracy and a 22.5% reduction in ISP latency compared to state of the art. In addition, we also evaluate Instance Segmentation task, where 1.2% accuracy improvement is attained with a 73% latency reduction.

Introduction

With AI becoming ubiquitous, vision system designers are challenged by the resource limitations of edge devices and the resource-intensive nature of deep learning (DL) applications. Emerging computing paradigms such as Green Edge AI advocate a resource-conscious approach to edge AI ensuring sustainable performance [1]. This approach shifts the design objective from solely maximizing intelligence performance (e.g., classification accuracy) to adopting a flexible and quality-scalable compute paradigm that jointly optimizes both accuracy and key design constraints, such as latency and energy budgets. Techniques adopted for incorporating QoS include adaptive selection of ML model depending upon the system constraints [2, 3] and changing the input resolution to tradeoff for energy efficient visual computing [4]. Software-Defined Imaging (SDI) [5], a recently proposed computing paradigm, uses software to control imaging hardware, enabling flexible image processing and real-time adjustments to algorithms, sensor configurations, and workflows. SDI aims to design the hardware–software stack from the sensor, Image Signal Processor (ISP), compute architecture, and OS to allow for reconfigurability and programmability driven by the end applications and device constraints. An ISP converts RAW sensor data to a high-quality image and comprises multiple processing blocks that can be configured via an ISP configuration. As an example Table 1 provides ISP configurations for Fast-OpenISP [6]. This configuration controls the structure, i.e., the number of active ISP blocks and their tunable parameters, and hence provides a potential quality control knob, provided SDI provides ISP reconfigurability. Traditionally, the ISP parameters are fixed after a manual tuning process or optimized automatically [7] aimed at producing the best image quality for humans. With the advent of AI and Deep Learning, there are now a growing number of cameras whose outputs are no longer viewed by humans but rather by AI algorithms such as Image Classification, Object Detection, Segmentation, etc. Thus, an ISP tuned for human perception may not provide optimal results for high-level computer vision tasks [8]. Therefore, there has been a growing interest in tuning the ISP parameters for high-level computer vision tasks. These works include efforts to design end-to-end learnable ISP pipelines [9–11], optimization of ISP for a particular set of CV tasks using genetic algorithms [12, 13], and empirical studies [14–16] on effects of various ISP blocks, possible reductions in ISP blocks and minimum ISP search for a CV task [17]. In Shi et al. [17], ISP structure and parameters are optimized to achieve compact and efficient configurations, improving accuracy and reducing ISP latency under challenging conditions like low-light and noisy images. *Almost all of the schemes aim at providing an ISP setting tuned for either human perception or a particular CV application and do not formally consider utilizing the ISP configurations as a quality control knob to provide quality scalable ISP tuning.*

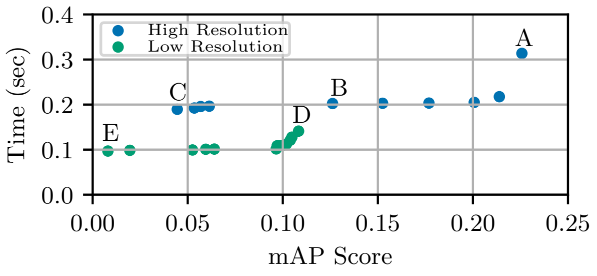

Figure 1: The ISP design space is defined by ISP Latency T (in sec.) and accuracy, evaluated across various ISP configurations for two different resolution settings. This design space allows exploration of the trade-offs between processing speed (latency) and quality (accuracy) for each configuration.

Motivational Analysis: To demonstrate that ISP parameters indeed provide a reliable QoS control knob, we provide the accuracy score (Mean Average Precision (mAP)) for a computer vision application (Object Detection) in Figure 1 for multiple ISP configurations of Fast-OpenISP [6]. The design points are dictated by ISP configuration design space, which we state in Table 1.

For each configuration, an image is generated that is subsequently passed on to an object detector [18], and its mAP score is computed. The design points were found by applying a GA-based optimization for two distinct input resolutions (1164x874 and 582x438). Without the loss of generalizability, we observe that *by tweaking ISP configuration, we can control the QoS of the CV application.* As an example, ISP configuration A provides a higher mAP score, while point C provides a lower mAP with a lower ISP computational time requirement. We also observe that, as expected, lowering the resolution provides another quality knob. *However, more importantly, an ISP configuration with low-resolution input may provide a higher mAP than one with high-resolution input requiring more processing time*. For example, Point C is inferior compared to Point D.

Required: Thus, there is a motivation to jointly explore the combined design space of input resolution and ISP configuration. This enables a graceful degradation of quality as a function of a constraint, such as the ISP latency for our case. For example, in Figure 1 traversing the path of A-B-D-E may provide a better adaptive QoS than a more traditional flow where we switch from a higher resolution to a lower resolution, which may result in sub-optimal adaptive QoS (A-B-C-E).

Our Contribution: In this paper, we present FlexEye, an application-specific, quality-scalable ISP tuning scheme that provides a set of ISP configurations enabling SDI-ready programmable ISPs to dynamically adapt their QoS. Our approach employs a mixed variable GA-based optimization strategy that navigates a multi-objective design space, considering both CV task accuracy and processing time across multiple resolutions to extend the QoS design space. By facilitating adaptive Quality of Service (QoS), our ISP configurations enable ISPs to transition to lower-quality profiles while delivering substantial gains in terms of reducing the ISP latency. The scheme was evaluated for multiple computer vision tasks, and results were compared with state-of-the-art.

Methodology

Problem Formulation

The ISP pipeline f_{ISP} is characterized by its structure Λ and parameters Θ. We assume that the ISP modules are indexed from $0 \dots H$, where H is the total number of ISP parameters that can be configured. If module i is *on*, i.e. $\Lambda_i = 1(\text{True})$, Θ_i represents the configuration parameter of this module. Given a dataset of raw sensor images $\mathbf{R} = \{r_0, r_1, \dots, r_n\}$, the ISP pipeline is used to transform these images into Red-Green-Blue (RGB) images $\mathbf{I} = \{i_1, i_2, \dots, i_n\}$. This transformation is represented as:

$$\mathbf{I} = \{f_{\text{ISP}}(\text{scale}(r_i, \text{res}), \Lambda, \Theta, \text{res}) | r_i \in \mathbf{R}\} \tag{1}$$

The $\text{scale}(r_i, \text{res})$ provides the RAW image at the specified resolution res. The performance of the targeted downstream task is denoted as M. For-example, for object detection, $M(\mathbf{I})$ represents the mAP score for all RGB images in $\mathbf{I}$. We explore the dual-objective space of performance M and the computation time. The premise is that we should be able to trade-off performance with lowering the computation complexity, possibly by turning some of the ISP modules *off*. We denote the computation time of the ISP pipeline as $T(\Lambda, \Theta, \text{res})$.

Our objective is to find the pareto-optimal set of ISP configurations. More concretely, we define the set of all ISP configurations as $\mathbf{C} = \{(\Lambda_i, \Theta_i, \text{res}_i) | i \in \mathbb{N}^0\}$. The configuration c_i dominates the configuration c_j, where $i \neq j$, based on the following conditions:

Dominance Condition For succinct notation, we denote $f_{\text{ISP}}(\text{scale}(r_k, \text{res}_i), \Lambda_i, \Theta_i, \text{res}_i)$ as $f^i_{\text{ISP}}(r_k)$. The dominance relation is defined as:

$$c_i \succ c_j \qquad \text{IFF}$$

$$\begin{gathered} M(\{f^i_{\text{ISP}}(r_k)\}) > M(\{f^j_{\text{ISP}}(r_k)\}) \\ \text{AND} \quad T(\Lambda_i, \Theta_i, \text{res}_i) \leq T(\Lambda_j, \Theta_j, \text{res}_j) \\ \text{OR} \\ M(\{f^i_{\text{ISP}}(r_k)\}) \geq M(\{f^j_{\text{ISP}}(r_k)\}) \\ \text{AND} \quad T(\Lambda_i, \Theta_i, \text{res}_i) < T(\Lambda_j, \Theta_j, \text{res}_j) \end{gathered}$$

where $r_k \in \mathbb{R}$ and configuration i must be better than configuration j in either performance or time.

Pareto-Optimal Set: The goal is to find the pareto-optimal set of ISP configurations:

$$\hat{\mathbf{C}} = \{c_i \in \mathbf{C} | \{c_j \in \mathbf{C} | c_j \succ c_i, i \neq j\} = \emptyset\} \tag{2}$$

i.e. $\hat{\mathbf{C}}$ is composed of all ISP configurations which are not dominated. Through the Genetic Algorithm (GA) based search approach presented in the following section, we intend to find a good approximation of $\hat{\mathbf{C}}$.

Optimization Process

As stated in the previous section, a given ISP configuration is represented as $c(\Lambda, \theta, \text{res})$. In this section, we explain how we optimize the ISP configuration. The optimization is conducted considering two objectives: the application-specific performance metric ($M(\cdot)$), and the execution time of the ISP pipeline ($T(\cdot)$). This optimization is a complicated process due to the inter-dependency of the ISP modules. To address this inter-dependency, we adopt a mixed-variable GA approach.

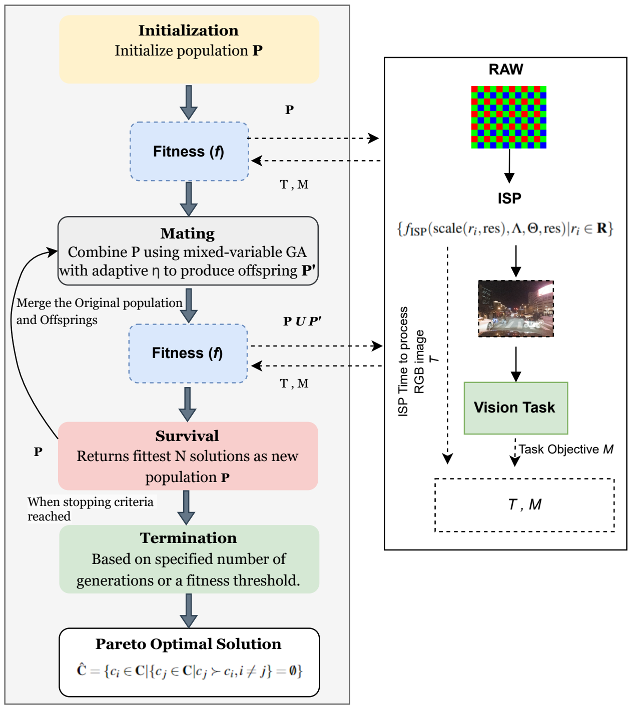
Initialization
Initialize population **P**
P
Fitness (*f*)
T , M
Mating
Combine P using mixed-variable GA with adaptive η to produce offspring **P'**
Merge the Original population and Offsprings
P U P'
Fitness (*f*)
T , M
Survival
Returns fittest N solutions as new population **P**
P
When stopping criteria reached
Termination
Based on specified number of generations or a fitness threshold.
Pareto Optimal Solution
$\hat{\mathbf{C}} = \{c_i \in \mathbf{C} | \{c_j \in \mathbf{C} | c_j \succ c_i, i \neq j\} = \emptyset\}$
RAW
ISP
$\{f_{\text{ISP}}(\text{scale}(r_i, \text{res}), \Lambda, \Theta, \text{res}) | r_i \in \mathbf{R}\}$
ISP Time to process RGB image *T*
Vision Task
Task Objective *M*
T , *M*

Figure 2: FlexEye ISP Tuning Methodology

GA is inspired by the evolutionary process and is good at tackling optimization problems that are not well suited for standard optimization algorithms, in which the objective function is discontinuous or nonlinear. To perform GA based optimization, we use pymoo python library [19].

The individual steps are illustrated in Figure 2 whereas, algorithm 1 provides the concise description of the optimizer. As stated in algorithm, we first **initialize** the population and compute its **fitness**. Fitness has two attributes as stated in the following equations:

$\forall c_i \in \boldsymbol{P_n}$

$$f_i^M = M(\{f_{\mathrm{ISP}}(\mathrm{scale}(r_j, \mathrm{res}), \Lambda_i, \Theta_i, \mathrm{res}_i) | r_j \in \mathbf{R}\}) \tag{3}$$

$$f_i^T = T(\Lambda_i, \Theta_i, \mathrm{res}_i) \tag{4}$$

f_i^M needs to be maximized while f_i^T needs to be minimized. Next, t individuals from the population are selected for mating based on tournament selection method [20]. On the selected parents, **crossover** operation is performed, producing a set of offspring ($\boldsymbol{P'_g}$). Based on the probability p_m the offspring are then **mutated**. Finally, Rank and Crowding Survival (RnC-S) is used to determine the individuals from the combined set of parents and offspring, who will **survive** into the next round of GA. Once the GA **terminates**, due to reaching either the generation threshold or objective space tolerance threshold f_{tol}. f_{tol} is integrated in [19] and is a stopping criterion based on changes in the objective space covered by the returned solutions. The set of individuals in the final generation who are on the Pareto-front (see (2)) is returned as the solution. These are the Pareto-optimal set of ISP configurations.

Algorithm 1 Proposed Algorithm for ISP Optimization

```
Require: Default ISP, Parameters range limits, Maximum number of generations $G$ and fitness tolerance threshold $f_{\mathrm{tol}}$, Population size $N$, Crossover probability$P_c$, number of parents $t$, and $\boldsymbol{\eta}$ array containing $\eta$ values for different sets of generations.
 1: Initialization: Initialize the population $\{P_{0,i}\}_{i=0}^{N-1}$ where the first individual $P_{0,0}$ = default ISP, while other individuals are randomly generated within the stated range limits. Compute the fitness of each individual using equations (3) and (4).
 2: for $g = 0, 1, 2, \ldots, G$ do
 3:     $\eta = \eta_k$ where $k = \lfloor \frac{g}{5} \rfloor$
 4:     Selection: Select fittest $t$ parents from $\boldsymbol{P_g}$ where $t < N$, using tournament selection method as stated in [20]. We call this set $\hat{\boldsymbol{P}}_{\boldsymbol{g}}$
 5:     Crossover: Perform 2-point Crossover on elements pairs in $\hat{\boldsymbol{P}}_{\boldsymbol{g}}$ producing the set of offspring $\boldsymbol{P'_g}$
 6:     Mutation: With a probability $p_m$, modify the individuals in $\boldsymbol{P'_g}$. Bit-flip mutation is used for binary attributes while polynomial mutation is used for integer and floating point attributes. $\eta$ determines the level of variation which is added with lower values affecting higher variation.
 7:     Fitness: Evaluate fitness ((3) and (4)) for each individual in the population $\boldsymbol{P'_g}$.
 8:     Survival: Use RnC-S method to select the new population $\boldsymbol{P_{g+1}}$ from the combined set of current and new individuals ($\boldsymbol{P_g} \cup \boldsymbol{P'_g}$)
 9:     Termination Criteria: If the stopping criteria (e.g., maximum number of generations $G$ or fitness tolerance threshold $f_{\mathrm{tol}}$) are met, then terminate.
10: end for
11: Output: Pareto optimal individuals as stated in Equation 2.
```

Experiments and Results

Our results are structured into three sections. In the first two sections, we evaluate the performance of FlexEye ISP in delivering quality-scalable ISP configurations for two computer vision applications: object detection using YOLOv8n [18] and instance segmentation using SAM [21], both of which are pre-trained on the COCO dataset [22]. All experiments are carried out on a Corei7 machine with an NVIDIA 1080Ti GPU. In the final section, we compare the effectiveness of our scheme against the state-of-the-art automatic ISP tuning scheme of [17].

Camera ISP: ISP used to evaluate the scheme was Fast-OpenISP [6], an efficient version of OpenISP. The details of the ISP pipeline and configurations are provided in Table 1. Our fitness function is multi-objective, considering task fitness $M(\cdot)$ (measured by Mean Average precision mAP for object detection and Mean Intersection over union (mIOU) for segmentation) and computation budget given by ISP latency per image $T(\cdot)$.

Dataset: We validate FlexEye using a low-light raw image dataset sourced from a OnePlus sensor, provided in [9]. This dataset consists of 141 night time driving scenes. We manually annotated each image with ground truth labels and masks for every instance of the Person and Car classes. We consider three resolutions (1164x874, 920x690, 582x438). Stratified sampling is used to get 42 images employed for ISP tuning. The remaining 99 images are used for testing. This split is identical for both object detection and instance segmentation.

FlexEye Hyperparameter Settings: For FlexEye, the population size N is 30, with a maximum of 30 generations G. The optimization will terminate early if the termination criterion based on f_{tol} is satisfied. $\boldsymbol{\eta}$ is set to $[15, 20, 25, 30, 35]$. These settings were chosen based on their balance between convergence speed and exploration of the solution space.

Quality-scalable ISP Tuning for Object Detection

Figure 3 provides the results of Quality-scalable ISP Tuning for Object Detection using FlexEye on training set. The y-axis provides CV task performance M, which provides the mAP50-95 score, a widely used metric for object detection evaluation, while the x-axis provides the ISP processing rate which is the reciprocal of ISP latency $(1/T)$, thus highlighting the trade-off between detection accuracy and ISP processing rate for various ISP configurations. For this experiment, three different input resolutions (1164x874, 920x690, 582x438) were considered for FlexEye, represented by Hi-res, Med-res, and Low-res in the Figure 3. The dashed line represents the quality-scalable ISP configurations returned by FlexEye. We also provide the ISP configurations at the individual Pareto front for each resolution. The FlexEye ISP configuration points (A, B, C, and D) provided on the dashed line show that the combined design space allows us to avoid inferior points offered by individual resolutions. Note that a more traditional approach might have simply reduced the resolution without considering the additional design space offered by the ISP for enhanced QoS. Despite observing distinct performance clusters for each resolution, the design points offered by each resolution may have inferior points compared to those provided by the extended design space enabled through ISP tuning. For example, ISP configuration B is superior to multiple Hi-res ISP configurations. In

Table 1: ISP Configuration settings considered for Fast OpenISP [6] for the Design Space Exploration conducted in this research work. Here $\Lambda = 1$ represents a necessary block while $\Lambda = \{0,1\}$ represents an ISP block that can be turned OFF or ON. Ranges for tunable parameters Θ are also provided in the table, where I and F represent integer and float values, respectively.

index	Module Name	Parameter/s	Λ	Θ
0	Dead Pixel Correction (DPC)	diff_threshold, neighbor_pixels	{1}	I[0,99], I[0,9]
1	Black Level Compensation (BLC)	{r,gr,gb,b}, {α,β}	{1}	{I[0,1023]}, {F[0,1)}
2	Automatic White Balance (AWB)	{r_gain, gr_gain, gb_gain, b_gain}	{1}	{I[0, 1023]}
3	Chroma Noise Filter (CNF)	diff_th, {r_gain, b_gain}	{1}	I[0,39], {I[0,1023]}
4	Color Correstion Matrix (CCM)	-	{0,1}	Fixed
5	Gamma Correction (GAC)	gain, gamma	{1}	I[0, 255], F[0,2)
6	Non-local Means Denoise (NLM)	-	{0,1}	Fixed
7	Bilateral Noise Filtering (BNF)	{intensity_sigma, spatial_sigma}	{0,1}	{F[0,1)}
8	Contrast Enhancement (CEH)	clip_limit	{0,1}	F[0,1)
9	Edge Enhancement (EEH)	edge_gain, {flat_th, edge_th, delta_th}	{1}	I[0,1023], {I[0,9]}, I[0,255]
10	brightness and contrast control (BCC)	{brightness_offset, contrast_gain}	{0,1}	{I[0,255]}
11	False Color Suppresion (FCS)	-	{0,1}	Fixed
12	Hue Saturation Control (HSC)	{hue_offset, saturation_gain}	{0,1}	{I[0,255]}

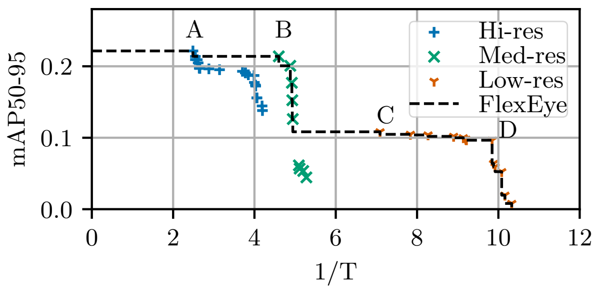

Figure 3: Pareto front of design space explored by FlexEye for quality scalable ISP tuning for Object Detection. The X-axis provides the reciprocal of ISP Latency, and hence, the higher, the better; the Y-axis provides the M score; the higher, the better.

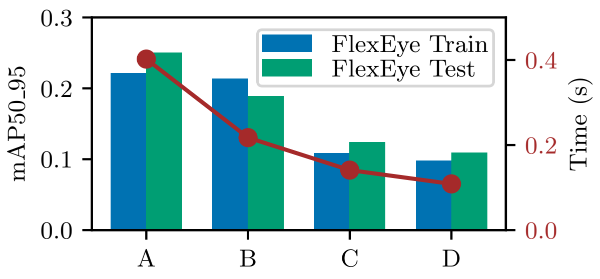

Figure 4: Object Detection: FlexEye enables effective quality scaling of the ISP as a function of ISP Latency (Time).

Figure 4, we provide the train and test performance of FlexEye ISP Configurations A, B, C, and D, and found the results to be consistent with training. As we move from point A to B, the mAP score on test images decreases by only 6.17% with a 45.93% reduction in ISP latency. Similarly, as we move from point C to D, the mAP decreases by only 1.46% for a 22.95% reduction in ISP Latency. Thus, FlexEye configurations provide a graceful quality degradation as the ISP latency is reduced.

Figure 5 compares the object detection results computed over a test image for FlexEye ISP as compared to the default ISP provided by [9]. It can be observed that for each resolution, an ISP configured by FlexEye provides more detections at a lower/comparable time.

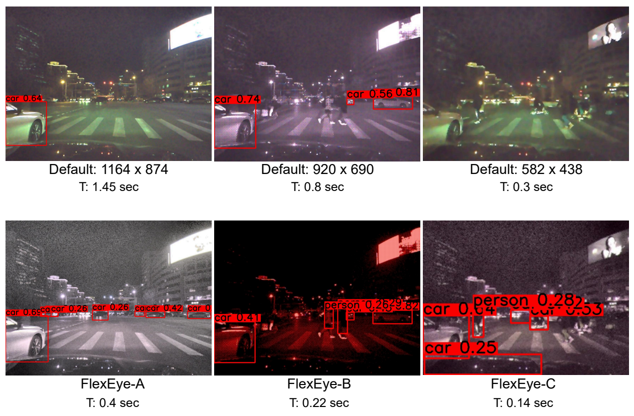

Figure 5: Visual results for Object Detection on images obtained using three FlexEye ISP configurations (A, B, C) compared with that of Default ISP at the three resolutions. Our ISP configurations detect more objects while achieving lower ISP latency.

Quality-scalable ISP Tuning for Segmentation

Figure 6 provides the results of FlexEye's quality-scalable ISP tuning for Instance Segmentation. Here, the y-axis provides the mIOU score. Similar to the last section, three different input resolutions (1164x874, 920x690, 582x438) were considered. The FlexEye ISP configuration points provided on the dashed line show that exploring the combined design space allows us to avoid inferior points offered by individual resolutions. For example, ISP configuration C is superior to a number of Med-res ISP configurations. We provide the performance of segmentation on test images produced by FlexEye ISP configurations in Figure 7, and found the results to be consistent with training. It can be observed that FlexEye configurations indeed provide a graceful degradation in quality as the ISP latency is reduced. For example, as we move from point A to B, the mIOU score for test set decreases by only 2.06% for a 48.59% reduction in time. Similarly, as we move from point C to D, we see 6.55% decrease in mIoU and a 16.48% decrease in ISP Latency. Figure 8 compares the segmentation results computed over a test image for FlexEye ISP as compared to the default ISP provided by [9]. It can be observed that for each resolution, an ISP configured by FlexEye provides better instance segmentation at a lower/comparable time.

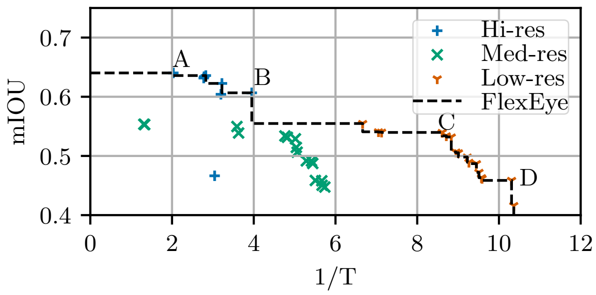

Figure 6: Pareto front of design space explored by FlexEye for quality scalable ISP tuning for Segmentation.

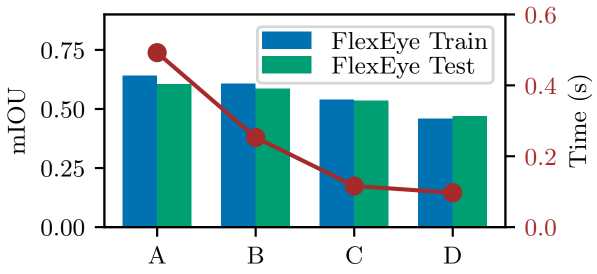

Figure 7: Segmentation: FlexEye enables effective quality scaling of the ISP based on Latency (Time).

Comparison with SoA ISP Tuning Scheme

In this section, we compare the performance of our ISP tuning method to Refactoring ISP [17], a recent approach which improved upon the prior art by proposing automatic search for compact ISP configurations. For completeness, we also compare our performance with the Default ISP configuration provided with the dataset [9]. For a fair comparison, we used the same hyperparameters (population size of 265 and 63 generations) as suggested by [17]. Figure 9 shows the performance comparison in terms of accuracy and processing time for Object Detection on both the training and test datasets. Here, we compare the configurations with the highest mAP50-95 for both FlexEye and Refactoring ISP. Our method achieves the highest mAP, with scores of 22.29% during training and 25.39% during testing, with ISP latency of just 0.46 seconds (1/T=2.17) and 0.31 (1/T=3.2) seconds respectively. Our approach surpasses both the Default ISP and Refactoring ISP configurations by 5.6% and 6%, respectively, while reducing the ISP latency by up to 78% and 22.5% respectively for test set. Similarly, Figure 10 provides the results for segmentation. Here, we compare the configurations with the highest mIOU for both FlexEye and Refactoring ISP. Our method achieves the highest mIOU, with scores of 64.02% during training and 61.1% during testing with an ISP latency of 0.49 sec (1/T =2.04) and 0.35 sec (1/T = 2.86) respectively. On the test dataset, our approach again surpasses both the Default ISP and Refactoring ISP configurations by 1.1% and 1.2%, respectively, while reducing the ISP latency by up to 75.8% and 73%, respectively. Thus, FlexEye ISP configurations consistently deliver better QoS performance for both CV tasks, thereby offering compact and efficient ISP configurations compared to [17]. These results are also supported by Figure 11 which provides the visual results on a test image for both tasks. It can be observed that FlexEye provides better performance compared to the Default, and the Refactoring ISP [17] configurations for both tasks.

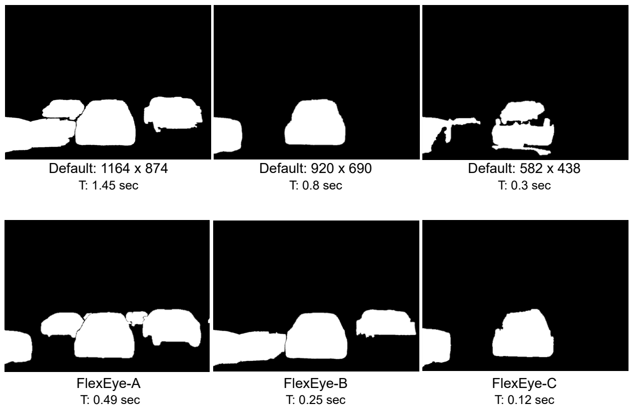

Figure 8: Visual results for Segmentation on images obtained using three FlexEye ISP configurations (A, B, C) compared with that of Default ISP at the three resolutions. FlexEye ISP configurations provide better segmentation quality at lower ISP latency.

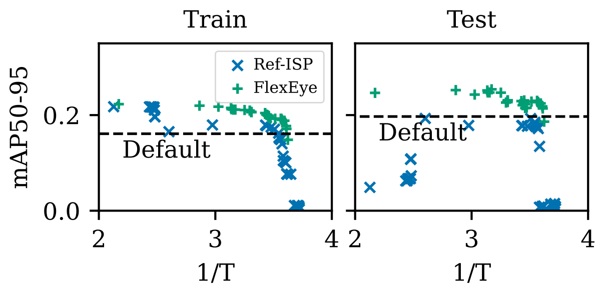

Figure 9: Object Detection: Performance comparison of FlexEye ISP configurations to that of state of the art [17]. ISP latency for Default configuration is 1.45 sec (1/T = 0.68).

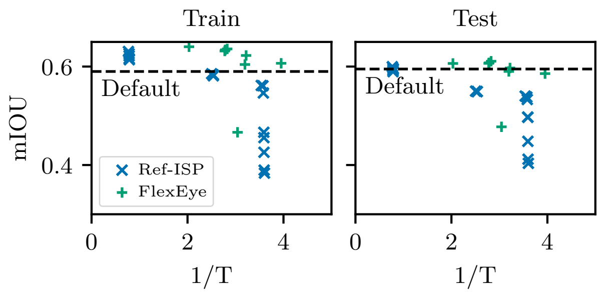

Figure 10: Segmentation: Comparison of FlexEye ISP configurations to that of state of the art [17]. ISP latency for Default configuration is 1.45 sec (1/T = 0.68).

Conclusion and Future Work

This paper incorporates ISP tuning with resolution settings as a control knob for quality scaling. The introduced ISP tuning framework, FlexEye, utilizes a mixed-variable genetic algorithm (GA)-based optimization strategy to explore a multi-objective design space, optimizing computer vision (CV) task accuracy and ISP latency across multiple resolutions. Experimental results on two CV applications were carried out. It was demonstrated that the extended design space offered by our scheme provides ISP configurations that provide object detection with 6% better accuracy while requiring 22.5% lower ISP latency than SOA and

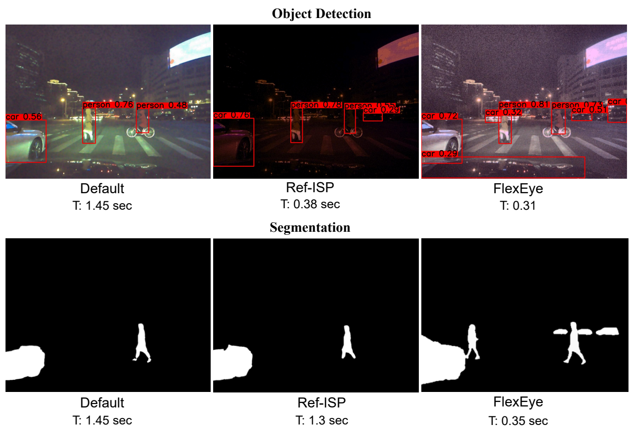

Figure 11: Visual Results for Object Detection and Segmentation comparison with the SOA and Default ISP [9] configurations

5.6% better accuracy while requiring 78% lower ISP latency than default. For Segmentation, FlexEye provided 1.2% better accuracy while requiring 73% lower ISP latency than SOA and 1.1% better accuracy while requiring 75.8% lower ISP latency than default ISP on test set. Experiments also confirmed that our scheme provides a graceful degradation as the ISP latency is constrained. Possible future work involves extending FlexEye to optimize for additional objectives, such as energy efficiency or memory footprint, to further enhance its applicability in resource-constrained environments. Another possibility is to explore the integration of machine learning models to predict optimal ISP configurations based on real-time application needs. Additionally, expanding the evaluation of FlexEye across a broader range of CV tasks and real-world scenarios would provide deeper insights into its scalability and versatility in different use cases.

References

[1] Yuyi Mao, Xianghao Yu, Kaibin Huang, Ying-Jun Angela Zhang, and Jun Zhang. Green edge ai: A contemporary survey. *Proceedings of the IEEE*, 2024.

[2] Vicent Sanz Marco, Ben Taylor, Zheng Wang, and Yehia Elkhatib. Optimizing deep learning inference on embedded systems through adaptive model selection. *ACM Transactions on Embedded Computing Systems (TECS)*, 19(1):1–28, 2020.

[3] Tom Springer, Erik Linstead, Peiyi Zhao, and Chelsea Parlett-Pelleriti. Towards qos-based embedded machine learning. *Electronics*, 11(19):3204, 2022.

[4] Robert Likamwa, Jinhan Hu, Venkatesh Kodukula, and Yifei Liu. Adaptive resolution-based tradeoffs for energy-efficient visual computing systems. *IEEE Pervasive Computing*, 20(2):18–26, 2021.

[5] Suren Jayasuriya, Odrika Iqbal, Venkatesh Kodukula, Victor Torres, Robert Likamwa, and Andreas Spanias. Software-defined imaging: A survey. *Proceedings of the IEEE*, 111(5):445–464, 2023.

[6] Qiu Jueqin. Fast open image signal processor. `https://github.com/QiuJueqin/fast-openISP`, 2021.

[7] G Pavithra and Bhat Radhesh. Automatic image quality tuning framework for optimization of isp parameters based on multi-stage optimization approach. *Electronic Imaging*, 33:1–7, 2021.

[8] Diarmaid Geever, Tim Brophy, Dara Molloy, Martin Glavin, Edward Jones, and Brian Deegan. Isp tuning for improved image quality in machine vision. *Electronic Imaging*, 36:1–8, 2024.

[9] Ke Yu, Zexian Li, Yue Peng, Chen Change Loy, and Jinwei Gu. Reconfigisp: Reconfigurable camera image processing pipeline. In *Proceedings of the ICCV*, pages 4248–4257, 2021.

[10] Ethan Tseng, Felix Yu, Yuting Yang, Fahim Mannan, Karl ST Arnaud, Derek Nowrouzezahrai, Jean-François Lalonde, and Felix Heide. Hyperparameter optimization in black-box image processing using differentiable proxies. *ACM Trans. Graph.*, 38(4):27–1, 2019.

[11] Eli Schwartz, Raja Giryes, and Alex M Bronstein. Deepisp: Toward learning an end-to-end image processing pipeline. *IEEE Transactions on Image Processing*, 28(2):912–923, 2018.

[12] Ali Mosleh, Avinash Sharma, Emmanuel Onzon, Fahim Mannan, Nicolas Robidoux, and Felix Heide. Hardware-in-the-loop end-to-end optimization of camera image processing pipelines. In *Proceedings of the IEEE/CVF CVPR*, pages 7529–7538, 2020.

[13] Mark Buckler, Suren Jayasuriya, and Adrian Sampson. Reconfiguring the imaging pipeline for computer vision. In *Proceedings of the ICCV*, pages 975–984, 2017.

[14] Henryk Blasinski, Joyce Farrell, Trisha Lian, Zhenyi Liu, and Brian Wandell. Optimizing image acquisition systems for autonomous driving. *Electronic Imaging*, 30:1–7, 2018.

[15] Lucie Yahiaoui, Jonathan Horgan, Brian Deegan, Senthil Yogamani, Ciarán Hughes, and Patrick Denny. Overview and empirical analysis of isp parameter tuning for visual perception in autonomous driving. *Journal of Imaging*, 5(10):78, 2019.

[16] Dara Molloy, Brian Deegan, Darragh Mullins, Enda Ward, Jonathan Horgan, Ciaran Eising, Patrick Denny, Edward Jones, and Martin Glavin. Impact of isp tuning on object detection. *Journal of Imaging*, 9(12):260, 2023.

[17] Yongjie Shi, Songjiang Li, Xu Jia, and Jianzhuang Liu. Refactoring isp for high-level vision tasks. In *Proceedings of ICRA*, pages 2366–2372. IEEE, 2022.

[18] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolov8, 2023.

[19] Julian Blank and Kalyanmoy Deb. Pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020.

[20] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[21] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the ICCV*, pages 4015–4026, 2023.

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, and et al. Microsoft coco: Common objects in context. In *Proceedings of ECCV*, pages 740–755. Springer, 2014.

Author Biography

Sumbal Akram (MS 2024, Information Technology University) is a researcher at Vision Processing Lab. She focuses on optimizing ML models, with a particular emphasis on embedded systems and enhancing TinyML solutions for edge devices.
Muhammad Abdullah leads the Imaging team at 10xEngineers. He specializes in developing image and signal processing algorithms, ISP tuning, mathematical modeling focusing on providing end-to-end CV applications for edge devices.
Khuzaeymah bin Nasir is an AI engineer at 10xEngineers. He specializes in optimizing deep learning models for CV applications, focusing on creating efficient solutions for edge devices.
Shaharyar Yaqub is an algorithm engineer at 10xEngineers with expertise in ISP optimization. He focuses on developing frameworks that enable ISP optimizations for vision tasks.
Rehan Ahmed (Ph.D. 2015, University of Wisconsin Madison) is an Assistant Professor at ITU. His main research interests are design/scheduling of cyber-physical and energy harvesting systems.
Rehan Hafiz (Ph.D. 2008, University of Manchester) is a Professor and Director of Vision Processing Laboratory at ITU. His areas of research include Vision System Design, Approximate Computing, and Edge Analytics.