










An Edge-Side Real-Time Video Analytics System With Dual Computing Resource Control

Chuang Hu , Rui Lu , Qianlong Sang , Huanghuang Liang , Dan Wang , Senior Member, IEEE, Dazhao Cheng , Senior Member, IEEE, Jin Zhang , Member, IEEE, Qing Li , and JunKun Peng 

Abstract—Video analytics systems conduct video preprocessing to filter out unnecessary frames and model inference using appropriately selected neural networks for high analytics speed. Video preprocessing is instruction-intensive computing (IIC) executed by CPU, and model inference is data-intensive computing (DIC) executed by GPU. In this paper, we show the analytics accuracy of existing systems can largely vary in fields, caused by the *dynamic* IIC and DIC workloads of different contents in applications. Unfortunately, cameras have *fixed* CPU/GPU resources and cannot effectively adapt to workload dynamics. We develop Gemini, a new edge-side real-time video analytics system enhanced by a dual-image FPGA. We take the advantage of negligible image switching time of dual-image FPGAs, pre-configure one CPU image and one GPU image and elastically multiplex the dual CPU-GPU resources in *time* dimension. Gemini requires both hardware and software revisions. In hardware, we overcome challenges of hardware-dependent application development, low communication efficiency between the microprocessor and FPGA, and high programming complexity by hardware abstraction, asynchronous data transfer mechanism and stub-skeleton middleware. In software, we overcome the challenge of adapting to the dynamic workloads by a bandit learning approach. We implement Gemini and show that Gemini can improve the analytics accuracy to 90.35%.

Index Terms—Bandit learning, dual-image FPGA, video analytics, accelerators, middleware.

Manuscript received 5 December 2022; revised 12 April 2023; accepted 27 July 2023. Date of publication 2 August 2023; date of current version 8 November 2023. This work was supported in part by the Zhejiang Lab Open Research Project under Grant K2022PI0AB01, in part by the Special Fund of Hubei LuoJia Laboratory under Grant 220100016, in part by the Fundamental Research Funds for the Central Universities under Grant 2042023kf0132, in part by the National Key R&D Program of China under Grants 2020YFE0200500, GRF 15209220, 15200321, 15201322, ITF-ITSP ITS/070/19FP, ITF ITS/056/22MX, and CRF C5018-20G, and in part by the ITC via project “Smart Railway Technology and Applications” under Grant K-BBY1. Recommended for acceptance by T. Chantem. (Corresponding authors: Dazhao Cheng; Qing Li.)

Chuang Hu, Qianlong Sang, Huanghuang Liang, and Dazhao Cheng are with the School of Computer Science, Wuhan University, Wuhan 430000, China (e-mail: handc@whu.edu.cn; qlsang@whu.edu.cn; hh-liang@whu.edu.cn; dcheng@whu.edu.cn).

Rui Lu and Dan Wang are with the Hong Kong Polytechnic University, Hong Kong (e-mail: csrlu@comp.polyu.edu.hk; csdwang@comp.polyu.edu.hk).

Jin Zhang is with the Southern University of Science and Technology, Shenzhen 518000, China (e-mail: zhangj4@sustech.edu.cn).

Qing Li is with the Peng Cheng Laboratory, Shenzhen 518000, China (e-mail: liq@pcl.ac.cn).

JunKun Peng is with the Tsinghua University, Beijing 100190, China (e-mail: pj20@mails.tsinghua.edu.cn).

Digital Object Identifier 10.1109/TC.2023.3301136

I. INTRODUCTION

Video analytics systems nowadays support many applications such as video surveillance, vehicle counting, traffic control, self-driving, and others. These systems feed video frames into a pre-trained neural network (NN) model and conduct model inference. In this paper, we study *edge-side real-time video analytics systems*, where videos are generated in edge-side smart cameras and the video analytics are conducted in the edge for real-time response and/or privacy protection. There are orthogonal research directions where video analytics is conducted on pre-stored videos [1], or the real-time videos are sent to the cloud for cloud or edge-cloud analytics [2]. The hardware used for edge-side video analytics systems are smart cameras such as AWS DeepLens [3], with a CPU and a GPU. Typical edge-side video analytics systems include Microsoft Rocket [4], Amazon Rekognition [5], and others.

A real-time video analytics system needs to achieve high video analytics accuracy while satisfying delay requirements. To face limited edge-side resources, existing systems have an execution pipeline to preprocess video frames to filter out unnecessary frames or to extract only the Region of Interest (ROI) in a frame. When sending the preprocessed frame to model inference, existing systems will select appropriate NN models that best balance the model inference accuracy and delay. In this execution pipeline, the computing workloads of video preprocessing, which involve a large number of searching, sorting, matching operations, are *instruction-intensive computing (IIC)* and are executed in the CPU of the edge camera. The computing workloads of model inference, which involve simple operations but on a large amount of data, are *data-intensive computing (DIC)*, and the DIC workloads are executed in the GPU of the edge camera [6].

When using real-time edge-side video analytics systems in fields, we observe that the analytics accuracy of the systems can greatly vary. We take Microsoft Rocket [7] (vehicle counting) as an example (details in Section II.B). The analytics accuracy at dawn time is 85.7%, and it drops to 65.2% at rush hours. We observe that at dawn time with fewer vehicles, 82% of frames can be filtered and only 18% of frames are fed to model inference. When it comes to rush hours, only 27% of frames can be filtered and 73% of frames are fed to model inference. The video applications have *contents*, e.g., Dawn Time and Rush Hours, and different contents can result in *dynamic* IIC and DIC

workloads that most of these content changes are in minute-level. Unfortunately, the CPU/GPU resources are *fixed* in field cameras. This limits the potential to adapt to the workloads and optimize the video analytics accuracy, as we often see that one of the CPU/GPU has reached its maximum capacity, yet the resource utilization of the other is still low.

In this paper, we propose Gemini, a new real-time video analytics system enhanced by a dual-image FPGA. An image in FPGA is a bit file to configure every Logic Unit to the target functions. The newly developed dual-image FPGA, e.g., Intel Max10, can pre-store two or more images in the FPGA image flash and fast switch between them. The dual-image FPGA has a key advantage over current FPGAs on the reconfiguration time, which can take minutes. Moreover, we can alternatively choose the image switching address to enable more than two images stored on FPGA by simply modifying the source codes. Thus, we can pre-configure CPU and GPU images and switch them in runtime. As a result, we can have elastic CPU-GPU computing resources by multiplexing the dual computing resources in the time dimension.

To develop a new edge-side real-time video analytics system with the benefits of dual-image FPGAs, we need both hardware and software revisions. We face four unique challenges.

First, dual-image FPGAs have many variants developed by different vendors, e.g., Intel Max10, Xilinx Artix-7. The programming development of FPGAs is hardware-dependent. We analyze a set of video analytics applications. We abstract the commonly used FPGA functions and develop a new logic view of hardware functions. Different dual-image FPGAs can register into the Gemini system through adapters/drivers to support these functions. Thus, Gemini decouples video analytics applications from specific FPGA hardware, allowing the applications to be hardware-agnostic.

Second, there are great communications between the smart camera microprocessor and the FPGA for video data. Yet the performance of the two processors can mismatch, leaving one idle during data transmission. We develop an asynchronous data transfer mechanism, where the two processors write/read data into a shared memory asynchronously without blocking each other. This achieves high-throughput microprocessor-FPGA communications.

Third, considering the complex process of data communication between FPGA and edge devices, incorporating applications into Gemini brings a great burden to programmers. Moreover, images with different functions differ in interfaces, which requires a lot of work to load them into the FPGA and complete the adaptation of images. We design a middleware using stub-skeleton structure that creates a remote procedure call (RPC) style interaction scheme and eliminates the complexity of image and application adaptation.

Fourth, it is a challenge to adapt to the contents of a video analytics application and its dynamic IIC and DIC workloads, to optimize the video analytics accuracy given the dual computing resources. We observe that it is difficult to explicitly model the application workload dynamics, and then optimize the dual computing resources. We thus seek a learning-based approach. We develop a bandit-based algorithm: bandit learning

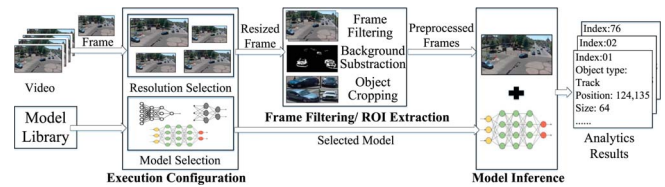


Fig. 1. The real-time video analytics pipeline.

is particularly suitable since the workloads depend on the video analytics application, not on the action choices of resource control algorithm.

We implement a Gemini prototype. We evaluate Gemini using real video trace-based experiments on two representative video analytics applications. We show that Gemini significantly outperforms existing video analytics systems. We develop two case studies where we use our prototype to support an intrusion detection application deployed in a laboratory for more than 8 hours and a disaster monitoring application deployed in the industrial system for two months. This study shows the end-to-end operations of Gemini in field and its consistent high accuracy.

In summary, the contributions of this paper are:

- We show through a measurement study that the accuracy of existing real-time video analytics systems (Section II.B) can greatly vary in fields. We investigate the root causes of such a phenomenon.
- We develop Gemini (Sections III and IV), a new real-time video analytics system enhanced by a dual-image FPGA. We present a set of hardware and software designs. Gemini can adapt to application workload dynamics and optimize the accuracy with dual computing resource control.
- We implement Gemini (Section VI) and evaluate Gemini with real-world video traces. We present two case studies (Section VII), showing the end-to-end operations of Gemini in field.

II. MOTIVATION AND APPROACH

A. Background on Real-Time Video Analytics

Real-time video analytics systems perform analytics on pre-trained neural network (NN) models to recognize spatial or temporal events (e.g., vehicle counting, object tracking [8]) in a video stream with latency requirements. An example is Microsoft Rocket [4]. One application atop the Rocket system is Microsoft Vision Zero [9], where NN models are pre-trained for the City of Bellevue vehicle counting.

To meet the delay requirements, e.g., the Vision Zero application typically requires processing 25 frames per second (fps), a real-time video analytics system preprocesses a frame and selects an appropriate NN model for model inference of this frame to optimize the analytics accuracy and latency. Fig. 1 depicts the execution pipeline. First, there is an execution configuration module, which includes a video resolution selection sub-module to resize the resolution of this frame by resolution selection algorithms [10], and an NN model selection sub-module to select an NN model that best balances the analytics

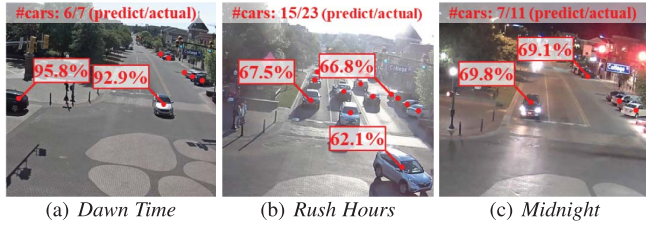


Fig. 2. The analytics results of Vision Zero application.

accuracy and delay by model selection algorithms [11]. Second, this frame will be sent to a frame filtering/ROI extraction module to filter out this frame if it does not contain relevant information by filtering technologies [12], and/or to extract Region of Interest (ROI) through background subtraction and object cropping technologies. Finally, this preprocessed frame and the selected NN model will be sent to a model inference module to execute analytics and output results.

In the fields, the hardware of a real-time video analytics system commonly consists of a smart camera enhanced with an edge device equipped with a CPU and a GPU. For example, Microsoft Vision Zero used a Webcam camera, with Azure Stack Edge with Intel Xeon CPU and NVIDIA T4 Tensor Core GPU. The microprocessor in the camera can conduct the lightweight execution configuration module and resize the frame into the appropriate resolution. This frame is then sent to the CPU to execute frame filtering/ROI extraction, which is computation-intensive since it requires a large amount of searching, sorting, matching operations when comparing frames. The computation is instruction-intensive computing (IIC) suitable for a CPU to process. There are some NN-based ROI approaches, however they are heavy-weight and, to the best of our knowledge, less used in the edge. Finally, this frame is sent to the GPU for model inference, which is again computational intensive. The computation is data-intensive computing (DIC) suitable for a GPU to process, so we consider all the NN-based methods are DIC tasks and others are IIC tasks.

B. Motivation

We conduct a measurement study on real-time video analytics systems to show that the analytics accuracy can significantly vary in fields and investigate the root causes.

Measurement Setup. We measure the Microsoft Rocket system with the Vision Zero application on an AWS DeepLens camera with Intel Atom CPU and Intel Gen9 GPU. The rocket system runs the built-in filtering algorithm [9], and the configuration algorithm [10] to select video resolutions from 144, 280, 540, 720, 1080p and the NN model from pre-trained models of TinyYOLO, SSD300, RetinaNet, and YOLOv3. For the video dataset, we use the video captured by the traffic cameras in the city of Bellevue [13] that contains 24 hours video streams of 38GB in a 25fps video frame rate.

Accuracy Dynamics and Causes. Fig. 2 shows the analytics results at Dawn Time, Rush Hours and Midnight. We observe that the accuracy varies: at Dawn Time Fig. 2(a), the accuracy is

TABLE I
MEASUREMENT RESULTS

Parameters & Utilization	Dawn Time	Rush Hours	Midnight
Number of Vehicles per Frame	7	23	11
Filtering Rate	82%	27%	64%
Resolution	540p	480p	1080p
Model	RetinaNet	TinyYOLO	SSD300
CPU Utilization Rate	62%	48%	100%
GPU Utilization Rate	99%	97%	76%

85.7%; at Rush Hours Fig. 2(b), the accuracy reduces to 65.2%; and at Midnight, the accuracy is 63.6%.

We investigate the root causes of the accuracy dynamics. Table I shows the number of vehicles, the frame filtering rate, the selected video resolution, the selected NN model, and the CPU/GPU utilization rate. We observe that the number of vehicles varies in different periods of a day. It leads to various frame filtering rates, the selected video resolutions and NN models. For example, when the number of vehicles was 7, 23, and 11 per frame, the frame filtering rates were 82%, 27%, and 64%, respectively. Intuitively, a greater number of vehicles in a video stream will increase the differences between two adjacent frames, leading to fewer frames to be filtered. At Dawn Time, the optimal video resolution and NN model selected were 540p and RetinaNet. Here, the CPU and GPU utilization rates were 62% and 99%. At Rush Hours, the frame filtering rate dropped to 27%. It means that a greater number of frames (73%) were fed to the GPU, and the GPU workloads increased. To meet the delay requirement, the video resolution decreased to 480p, and a small TinyYOLO model was used, leading to the low accuracy. Here, the GPU utilization was 97%, and the CPU utilization was only 48% since a low resolution reduces the filtering computation workloads on the CPU. At Midnight, the frame filtering rate was 64%. The video resolution and NN model were 1080p and SSD300. Here, the CPU and GPU utilization rates were 100% and 76%; this is the optimal configuration yet the GPU utilization is only moderate.

These measurements show that applications have *contents*, e.g., Dawn Time, Rush Hours, and Midnight; and different contents can result in *dynamic* IIC and DIC workloads. Unfortunately, the CPU/GPU resources are *fixed* in field cameras. This limits the potential to adapt to the workloads and optimize video analytics accuracy, as we can see that one of the CPU/GPU has reached its maximum capacity, yet the utilization of the other is still low.

C. Dual-Image FPGA and Potential Approach

An FPGA consists of an array of reconfigurable logic blocks and can be reconfigured to different customized functions. The FPGA reconfiguration can take minutes to complete, making it difficult to be used in runtime. Recent FPGA developments lead to a brand new dual-image FPGA, which allows two or more different images to be stored in the user flash memory and support fast switching counted by FPGA cycles. For example, Intel Max10 requires about $4.5e5$ cycles to switch to another image in ~ 800 ns under 472.5 MHz system clock frequency. Xilinx Artix-7 can also reach ~ 1 ms switching time under maximal frequency 464 MHz. Higher maximal frequency tolerance FPGAs have smaller switching time, e.g., Intel Stratix 10 and

TABLE II
THE ANALYTICS ACCURACY OF THREE CONTENTS UNDER
DIFFERENT CPU AND GPU RESOURCES

Resource	Dawn Time	Rush Hours	Midnight
CG-10%-90%	96.9%	69.2%	68.5%
CG-20%-80%	95.5%	84.3%	75.0%
CG-30%-70%	93.1%	64.0%	69.7%
CG-40%-60%	82.6%	66.1%	89.2%
CG-50%-50%	78.4%	61.7%	75.9%

Xilinx Zynq UltraScale are 250 ns and 120 ns, respectively (see Section V for details). With a dual-image FPGA, we can pre-store a CPU image to support IIC workloads and a GPU image to support DIC workloads. We can then make elastic CPU/GPU resources possible by multiplexing the two images in the *time* dimension, i.e., by adjusting the FPGA time allocated to the CPU image and the GPU image.

We now study the potential of dual-image FPGAs. We configure an Intel Max10 FPGA to support Vision Zero. We divide the FPGA time into one-second periods, and implement an FPGA time allocation strategy, where in each period, $x\%$ and $y\%$ of the FPGA time are allocated to the CPU image and GPU image; denoted as CG- $x\%$ - $y\%$. Table II shows the analytics accuracy of Dawn Time, Rush Hours and Midnight under different CG- $x\%$ - $y\%$. We observe that there are choices for high accuracy (bold red values in Table II) in each content.

Intuitively, we can develop a new real-time video analytics system enhanced by a dual-image FPGA where the system can optimize the analytics accuracy by resource allocation through x, y , as well as system configuration on video resolutions and NN models to adapt to the contents.

III. DESIGN OVERVIEW

A. The Gemini System

We now present Gemini, a new real-time video analytics system enhanced by a dual-image FPGA. The system hardware consists of a smart camera¹ and a dual-image FPGA.

Gemini (see Fig. 3) has a **System Monitor** to monitor the current and historical system states on resources and analytics accuracy. The core of Gemini is a **Workload Adaptation Controller** which takes the dynamics of analytics accuracy (which can reflect potential content changes) and system states to compute the optimal resource configuration to process this frame, i.e., the CPU-GPU time partition of the FPGA, the video resolution and the NN model. This video frame is resized to the appropriate resolution and then sent to the FPGA, which will first switch to the CPU image to execute **Frame Filtering/ROI Extraction**. The FPGA will then switch to the GPU image and take the preprocessed frame and the selected NN model to execute **Model Inference**.

B. Challenges and Key Design Choices

Gemini introduces both hardware and software revisions of an edge-side real-time video analytics system. We thus face four

¹The smart camera is general. In this paper, we assume a basic camera, e.g., a Raspberry Pi Zero camera with an ARM11 microprocessor. It can also be an upscale camera with extra fixed CPU and GPU resources. We can take these resources as constant factors into optimization; and all our results hold.

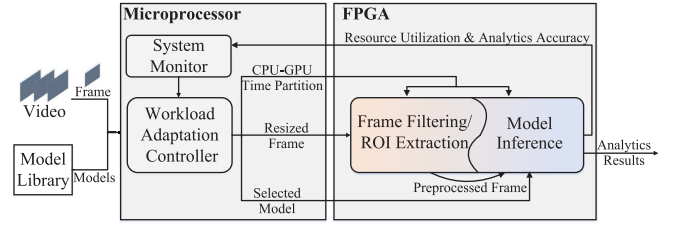


Fig. 3. The Gemini system.

unique challenges. From the FPGA hardware to the workload adaptation controller, they are:

Challenge 1: Dual-image FPGAs have different types, e.g., Intel Max10, Xilinx Artix-7, and the programming development is hardware-dependent, making it difficult for a video analytics application to be portable across different FPGAs.

Design 1: A new abstraction of hardware functions to make the Gemini system FPGA hardware-agnostic. Specifically, we analyze a set of existing representative video analytics applications. We abstract the commonly used FPGA functions and develop a new logic view of hardware functions. Different dual-image FPGAs can register into the Gemini system through adapters/drivers to enable the new FPGA functions. In this way, video analytics programming development is decoupled from hardware programming development, and a video analytics application can be portable across the Gemini system enhanced by different dual-image FPGAs.

Challenge 2: There are large data communications between the microprocessor and FPGA; yet the performance of the two processors is mismatched. One processor will be idle during data transmission, leading to significant resource waste.

Design 2: An asynchronous data transfer mechanism for microprocessor-FPGA communication. We design an asynchronous data transfer mechanism where the camera microprocessor and the FPGA write/read data into a shared memory asynchronously. This can unblock the processors from undertaking other computing workloads.

Challenge 3: The interface and data communication between processor and FPGA puts a lot of burden on the programmers, especially when using High-Level Synthesis (HLS) languages and tools. At the same time, the Dual image FPGA needs to be well adapted to different types of images, which makes applications and images incorporated in Gemini require a lot of work.

Design 3: A stub-skeleton design that creates a RPC style interaction scheme. We develop a middleware including stub-skeleton design and middleware kernel to process data communication between FPGA and edge device. It can reduce the complexity of using various accelerators and the workload of incorporating video analytics applications into Gemini for programmers.

Challenge 4: Video analytics applications have different contents, leading to dynamic IIC and DIC tasks and workloads. It is a challenge to optimize the video analytics accuracy given the dual computing resources of a dual-image FPGA to adapt to the contents and the dynamic IIC and DIC workloads. Note that both IIC and DIC tasks can be run on CPU or GPU if the

```

1 module SWITCH_IMG(input clk, input [63
  :0] in, output [15:0] res, output
  BOOT_SEL, output CFG_SEL);
2 assign SLAVE_IMAGE = IMAGE0;
3 wire [15:0] RD_ADDR;
4 assign [3:0] WR_ADDR = [35:32] in;
5 assign [31:0] DATA = [67:36] in;
6 altera_dual_boot adb(.clk(clk), .img(
  SLAVE_IMAGE), .data(DATA), .write
  (WR_ADDR), .read(RD_ADDR), .
  boot_sel(BOOT_SEL), .cfg_sel(
  CFG_SEL));
7 assign res = read;
8 endmodule

```

Fig. 4. FPGA image switching codes of Intel Max10.

system utilization is low. However, running IIC tasks on GPU is much less efficient. One of the design goals is not to run IIC tasks on the GPU mode or DIC tasks on the CPU mode.

Design 4: A bandit learning approach for elastic dual computing resource control. We consider it difficult to explicitly model the contents and the workload dynamics to allocate resources to optimize the analytics accuracy. We thus seek a learning-based approach to predict the workloads of IIC tasks and DIC tasks; and leverage the elastic resources of the dual-image FPGA to support these tasks. Our problem is intrinsically a control optimization problem, and the solution falls into a reinforcement learning (RL) algorithm. We argue that *Bandit Learning* is a suitable class of learning algorithms as compared to other regular RL algorithms. This is because RL algorithms learn the state-action pairs, i.e., the states should be affected by the control actions. In our problem, the workloads rely on the contents (e.g., rush hours) but are not affected by the resource allocation actions (e.g., FPGA time partition). However, a great challenge of applying bandit learning to our application is materializing the components (e.g., arms, agents, rewards) of bandit learning.

IV. GEMINI DESIGN

A. A Decoupled Design to Make the System FPGA Hardware-Agnostic

Dual-image FPGAs have many variants, e.g., Intel Max10, Xilinx Artix-7, etc. Due to their differences in low-level specifications, e.g., the number of pins and their instruction sets, programming on different FPGAs is tightly coupled to each type of FPGA.

An example of FPGA-dependent programming: We take the implementation of an image switching function as an example. Image switching is to program the specified image into the FPGA logic units for executing the functions implemented in the image.

Fig. 4 shows the Verilog code of image switching in Intel Max10. Line 2 loads the image file *IMAGE0* to the image area. Max10 stores the images in fixed memory areas, named master/slave image areas. Here, the image is stored into the slave image area (*SLAVE_IMAGE*); Lines 3–5 initiate the write/read parameters. The setups here are the read width is set to 16 bits, the write width is set to 4 bits, the maximum data width allowed to read/write an image in FPGA is set to

```

1 module SWITCH_IMG(input clk, input [63
  :0] in, output [31:0] res);
2 assign IMG0_ADDR = [15:0] in;
3 assign IMG0_SIZE = 42230;
4 wire [31:0] RD_ADDR;
5 assign [15:0] WR_ADDR = [31:16] in;
6 assign [31:0] DATA = [63:32] in;
7 xilinx_rcv_read xdb(.clk(clk), .
  data(DATA), .read(RD_ADDR));
8 xilinx_rcv_write xdbw(.clk(clk), .
  write(WR_ADDR), .addr(IMG0_ADDR),
  .size(IMG0_SIZE));
9 assign res = RD_ADDR;
10 endmodule

```

Fig. 5. FPGA image switching codes of Xilinx Artix-7.

32 bits. Line 6 reads the image from the slave image area and writes the image into the FPGA logic unit supported by a module (*altera_dual_boot*) from Intel. Max10 completes this procedure by sending signals to pre-defined output pins. Specifically, it sends signal 1 to pin *BOOT_SEL*, signal 0 to pin *CFG_SEL*.

Fig. 5 shows the Verilog code of switching to image *IMAGE0* in Xilinx Artix-7. Lines 2–3 assign the storage memory address and the image size (42230 bytes) of *IMAGE0*. This differs from Intel Max10 since Intel Max10 stores the image in a fixed image area, while Xilinx Artix-7 stores images in the dynamically allocated memory area. Lines 4–6 initiate the write/read parameters to set up the read width and the write width. Compared to Intel Max10, the values of read and write width are different due to the difference in the width of the data line in these two types of FPGAs. Lines 7–8 read the image from memory and write the image into the FPGA logic units. Here, Xilinx Artix-7 completes this procedure with a read module (*xilinx_dual_boot_read*) and a write module (*xilinx_dual_boot_write*) from Xilinx.

Hardware function abstraction: This example is a simple illustration that FPGA programming is hardware-dependent, yet video analytics applications only need the computing resources of the FPGAs. To solve this problem, we develop a new logic abstraction of hardware functions. This can decouple the video analytics application development and the FPGA hardware development, allowing applications to be agnostic to the FPGA specifics.

To develop a proper hardware function abstraction, we carefully analyze four representative video analytics applications, Reducto [12], FFS-VA [14], FCN-rLSTM [15], Faster-RCNN [16]. We examine the common hardware functions needed to support these applications. Table III shows a summary. We find that we can categorize the computing functions of the four video analytics applications into: 1) basic controls, 2) filtering, 3) interference, 4) background subtraction, and 5) cropping bounding box. Their requirements on hardware functions can be categorized into: 1) Hardware Setup functions that control the FPGA states such as On/Off/Sleep mode; 2) Image Management functions that manage the images of the FPGA, and 3) Data Processing that handles the data in the FPGA. We develop the detailed hardware functions in each category, and we show their descriptions in Table III.

TABLE III
ANALYSIS OF FOUR REPRESENTATIVE VIDEO ANALYTICS APPLICATIONS AND THEIR REQUIRED HARDWARE SUPPORT

Function Abstraction	Video Analytics Applications		Reducto [12]			FFS-VA [14]			FCN-rLSTM [15]			Faster-RCNN [16]			
	Function Name	Description	BC	Fil	Inf	BC	Fil	Inf	BC	BS	CBB	Inf	BC	CBB	Inf
Hardware Setup	FPGA_ON()	Turns on or awake FPGA.	✓			✓			✓				✓		
	FPGA_OFF()	Turns off FPGA.	✓			✓			✓				✓		
	FPGA_INIT()	Initializes FPGA.	✓			✓			✓				✓		
	FPGA_SLEEP()	Switches FPGA to sleep mode.	✓			✓			✓				✓		
Image Management	IMG_UPLOAD_CPU()	Uploads a CPU image into FPGA UFM.		✓			✓			✓	✓			✓	
	IMG_UPLOAD_GPU()	Uploads a GPU image into FPGA UFM.			✓		✓					✓			✓
	IMG_SWITCH_CPU()	Switches to CPU image on FPGA UFM.	✓				✓			✓	✓			✓	
	IMG_SWITCH_GPU()	Switches to GPU image on FPGA UFM.			✓		✓					✓			✓
Data Processing	DATA_WR()	Writes in data onto external memory.		✓	✓		✓	✓		✓	✓	✓		✓	✓
	DATA_RD()	Reads in data onto external memory.		✓	✓		✓	✓		✓	✓	✓		✓	✓
	IIC_TASK_PROCESS()	Executes IIC task on external memory.	✓				✓			✓	✓			✓	
	DIC_TASK_PROCESS()	Executes DIC task on external memory.			✓		✓	✓				✓			✓

BC: Basic Controls. Fil: Filtering. Inf: Inference. BS: Background Subtraction. CBB: Cropping Bounding Box.

```

1 def Reducto_Filter_Acc(img,
2   frame1, frame2):
3   if FPGA_STATUS!=ON:
4     FPGA_ON()
5   IMG_UPLOAD_CPU(img)
6   IMG_SWITCH_CPU()
7   DATA_ADDR, size=DATA_WR(frame1)
8   _, size=DATA_WR(frame2)
9   RES_ADDR=IIC_TASK_PROCESS(
10    DATA_ADDR, size)
11   result=DATA_RD(RES_ADDR)
12   FPGA_SLEEP()
13   return result

```

Fig. 6. Reducto filtering codes using Gemini hardware functions.

To illustrate how the abstraction can help application development, we use the implementation of the filtering function of Reducto as an example, where we can directly write Python codes, see Fig. 6. The filtering function computes the differences of two frames and filters out the one if the value of differences is less than a predefined threshold. Specifically, Reducto filtering turns the FPGA on (Lines 2–3), uploads the CPU image that implements the filtering function to the user flash memory of FPGA (Line 4), switches to the CPU image (Line 5), transfers the frames to the FPGA (Lines 6–7), executes the IIC to filter frame (Line 8), returns the filtering results (Line 9) and turns the FPGA off (Line 10).

B. An Asynchronous Data Transfer Mechanism for Microprocessor-FPGA Communication

An asynchronous data transfer mechanism. In a video analytics application, data need to be transmitted between the host microprocessor and the FPGA. Specifically, each constructs a sender thread and a receiver thread to establish a communication connection. The sender thread consumes clock cycles to send data, and the receiver thread consumes clock cycles to receive data. The sender/receiver threads will block the opposite processor until transmission completion to ensure communication correctness.

This mechanism performs poorly if the clock frequencies of the microprocessor and the FPGA mismatch: the fast processor idles. For example, a typical microprocessor, e.g., STM32f1 has a usual clock frequency of 64 MHz [17], and the clock frequencies of Intel Max10 and Xilinx Artix-7 are 473.5 MHz and 464 MHz, respectively. The clock idling waste is non-trivial,

i.e., as much as four-fifths of the FPGA clock cycles can be wasted.

We design a new data transfer mechanism where the microprocessor and the FPGA transfer data by writing/reading an external shared memory in an asynchronous manner. Specifically, after one processor writes the data into this shared memory, an interrupt will be triggered for the other processor to read the data. After finishing reading, the processor will be released for other computing tasks.

To support such a mechanism, we design an asynchronous data transfer controller, making sure that it supports the specific microprocessor and FPGA interface requirements such as bus width, data transfer rate, and addressing capabilities. We re-organize the priority level to prevent conflicts between the FPGAs and microprocessors. We configure the controller's arbitration scheme and adjust the priority of competing requests to minimize contention and reduce latency. We minimize data dependency between the microprocessor and FPGA. It allows both devices to perform their operations independently and concurrently, reducing the need for frequent synchronization and improving overall performance.

Discussion on the design choice. Our design of the asynchronous data transfer mechanism for microprocessor-FPGA communication is principally pragmatic. Our rationale is to eliminate resource waste in the microprocessor-FPGA communication without incurring large overheads. There are other methods to handle the frequency differences between two processors, e.g., frequency scaling, frequency virtualization and CPU multiplexing [18]. These software-based methods work for high-capacity CPUs, but are heavy for low-capacity MCUs. There are also hardware-based methods. For example, we can add parallel interfaces to FPGA, e.g., the AXI bus and PCI Express. With high-capacity communication physical lines, these interfaces can solve the capacity mismatch but would require a new customized FPGA design.

C. Middleware Between Edge Device and FPGA

Along with hardware function abstraction, we use stub-skeleton design that creates a RPC style interaction scheme to call these functions and simplify procedure of switching different images. This RPC method has the advantage of simplicity and practicality, so many heterogeneous systems would benefit

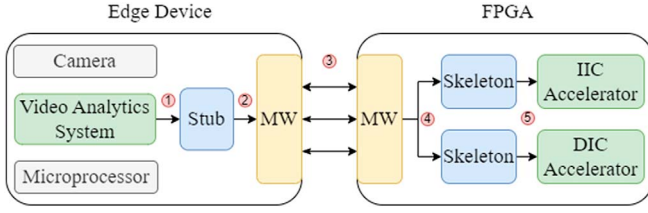


Fig. 7. Middleware workflow.

from this transparent data communication and control of the hardware IP-cores [19].

Fig. 7 shows the middleware workflow: 1) *Video Analytic System* calls the function with parameter provided by the *Stub*, and 2) the *Stub* packs the data to middleware. 3) The *Middleware kernel* (MW in Fig. 7) deployed on the edge device transmits the data to the FPGA through the asynchronous shared memory mentioned above. 4) Then the *Middleware kernel* in FPGA transmits the data to the corresponding *Skeleton* of the accelerator. 5) Data is unpacked by the *Skeleton* and passed to the accelerator. Finally, results come back to the original path after completing the task.

1) *Stub and Skeleton*: Stub and Skeleton have two main functions. The first function is to pack data into a specific data structure for communication. Stub is responsible for data packing, sending it to the middleware kernel, and waiting for the result to return. While Skeleton is responsible for unpacking data and sending it to the accelerator, returning the result after the accelerator finish task. The second function is to provide a better and clean abstraction both for FPGA image and video analytics applications. Stub provides a clean abstraction for the upper video analytic applications, it encapsulates the hardware-level function and provides the function interfaces required by each application, such as the filter function `Reducto_Filterer_Acc` shown in Fig. 6. When different video analytic systems use their own algorithms to analyze the video, they simply use the functions provided by the stub. Skeleton provides an abstraction between middleware and accelerators, which can implement different data processing patterns, such as one-shot computations that produce a single result from a single input, or stream processing of a sequence of data.

2) *Middleware Kernel*: Middleware kernel connects stub and skeleton, and its core is the asynchronous communication mechanism based on the shared memory mentioned above. Middleware kernel writes data such as function parameters or return values to a specific area, and notifies stub and skeleton through interrupt signals. Moreover, the kernel is also responsible for switching the skeleton of the accelerator. When the video analytic system calls the functions provided by stub which use other accelerators than the one loaded in FPGA, the kernel performs all necessary steps to switch the FPGA image. It loads the accelerator and skeleton which correspond to the called stub function onto the FPGA. Furthermore, the middleware kernel needs to manage the aforementioned FPGA function abstraction like `FPGA_ON`. When a function call is encountered, the kernel will look up the address corresponding to the function and transfer the data to the corresponding memory address, which greatly hides the complexities of using FPGA.

D. A Bandit Learning Approach for Elastic Dual Computing Resource Control

1) *Problem Formulation*: We consider a video analytics stream consisting of consecutive frames with a frame rate of f . For each frame, we need to select the resolution and the NN model as well as the CPU-GPU time partition of the FPGA, so that the delay constraint is satisfied, the system completes processing a frame before the next frame comes, and the analytics accuracy is maximized. Formally, Let $v_i \in \mathcal{V}$ be the resolution variable and $m_i \in \mathcal{M}$ be the NN model variable for frame i , where \mathcal{V} and \mathcal{M} are the set of resolutions and NN models. Let $T_i^{IIC}(v_i)$ denote the IIC processing time of frame i given v_i . Let $T_i^{DIC}(v_i, m_i)$ denote the DIC processing time of frame i given v_i and m_i . Let D be the delay constraint. We have:

$$T_i^{IIC}(v_i) + T_i^{DIC}(v_i, m_i) \leq D \quad (1)$$

Let t_i^C be the FPGA times allocated to CPU image for supporting IIC workload of frame i . Let t_i^G be the FPGA times allocated to GPU image for supporting DIC workload of frame i . t_i^C and t_i^G are decision variables to be optimized. We have:

$$t_i^C + t_i^G \leq \frac{1}{f} \quad (2)$$

$$T_i^{IIC}(v_i) \leq t_i^C \wedge T_i^{DIC}(v_i, m_i) \leq t_i^G \quad (3)$$

Let $A(v_i, m_i)$ be the analytics accuracy of frame i . Our objective is to maximize $A(v_i, m_i)$.

The Dual Computing Resource Control Problem: given the video frame i , the frame rate f , the set of video resolutions \mathcal{V} , the set of pre-trained models \mathcal{M} , and the delay requirement D , subject to delay constraint (1) and FPGA time constraints (2) and (3), determine the FPGA times allocated to CPU image t_i^C and GPU image t_i^G , the resolution v_i and the model m_i for frame i , to maximize the analytics accuracy $A(v_i, m_i)$.²

2) *Problem Analysis*: Video analytics applications have contents, and different contents result in dynamic IIC and DIC workloads. For example, the number of frames fed to the GPU depends on how many frames are left unfiltered, which further depends on whether the frames contain relevant information of the video analytics task. It is difficult to explicitly model the content and the workload dynamics.

We thus seek a learning-based approach. Our problem falls into an optimization control problem. There are two major categories of learning algorithms, reinforcement learning (RL) and bandit learning (which can also be categorized into RL, yet we emphasize its differences from RL). At a high level, RL is commonly used for a control problem where the control actions have a direct impact on future states, and RL learns the state-action interactions. Bandit learning is widely used for a control problem where the emphasis is to learn the statistical outcomes of the adjustment strategies.

²An edge camera can have additional fixed CPU and GPU resources. In such cases, we can develop a problem P_2 by replacing eq. (3) with $T_i^{IIC}(v_i) \leq t_i^C + t_i^{C'} \wedge T_i^{DIC}(v_i, m_i) \leq t_i^G + t_i^{G'}$, where $t_i^{C'}$ and $t_i^{G'}$ be the additional CPU time and GPU time to process frame i . It is easy to verify that P_2 is equivalent to the Dual Computing Resource Control Problem.

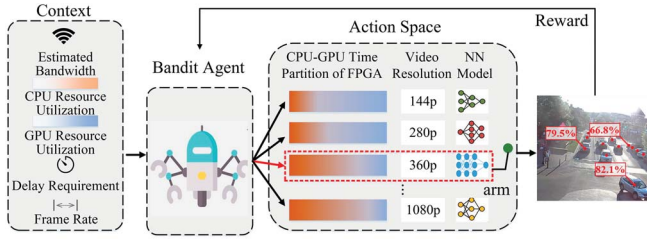


Fig. 8. The workflow of the bandit algorithm.

Bandit learning repeatedly observes a context, chooses an action, and observes the reward for the selected action. Bandit learning is suitable for applications where the contexts change smoothly [20]. This matches video analytics applications well, e.g., in a vehicle counting application, the traffic intensity changes smoothly as compared to the video analytics task rate [12]. Bandit learning requires fewer data and less computational power, and it is widely used in a large number of applications.

3) *The Bandit-Based Computing Resource Control Algorithm*: To exploit the bandit learning to solve the dual computing resource control problem, we first present the problem into the contextual multi-bandit framework, and then design a *Bandit-based Computing Resource Control (BCRC)* algorithm. In a contextual multi-bandit problem (Fig. 8), a bandit agent needs to make a sequence of decisions. At each time $t \in \{1, 2, \dots, T\}$, the agent observes different $contextu_t$ and different $armsa_t$. It then chooses an action, i.e., which arm a_t to pull. A reward r_{t,a_t} will be given based on u_t and a_t , with the rewards of other arms unknown. Let \mathcal{A} denote the arm set. Let $\mathbf{x}_{t,a} \in \mathbb{R}^d$ denote the feature vector capturing all the available side information, including selected features of the current context u_t and the arm a_t .

In our problem, a context u_t has four dimensions: the CPU resource utilization rate h_t^C , the GPU resource utilization rate h_t^G , the delay requirement d_t and the frame rate f_t . The agent can choose an action a_t at each time t , which has three features: the CPU-GPU time partition of the FPGA, the video resolution v_t , and the NN model m_t .

The system parameter configurations in BCRC are (1) the CPU-GPU time partition of the FPGA, (2) the video resolution, and (3) the NN models. The NN models are discrete in nature. We discretize the video resolution, e.g., in our case study in Section VII, it is 144, 280, 540, 720, 1080, and the FPGA time that nine predefined discrete levels for CPU:GPU-10:90, 20:80, 30:70, 40:60, 50:50, 60:40, 70:30, 80:20, 90:10. This is the action space of our bandit algorithm. Such discretization is pragmatic in general, e.g., we can discretize in finer granularity, yet it achieved good performance in practice.

We define the reward r_t as the observed analytics accuracy after the action. Since we cannot directly get the analytics results and ground truth once it goes online, we apply the uncertainty of the analytics results instead, which shows how uncertain they are in their predictions. The lower uncertainty indicates a higher probability of correct analytics.

One may want to simply make a decision to maximize the reward. Without sufficient exploration, however, this exploitation-based strategy can result in an arbitrarily bad outcome: our

agent can be trapped in a local optimum and continuously select sub-optimal arms without exploring better solutions. Therefore, we must carefully balance exploration and exploitation. We design a bandit-based computing resource control algorithm to achieve a good balance between exploration and exploitation and has a good efficiency with low complexity. The idea is to choose the arm with the highest upper confidence bound (UCB) instead of choosing the arm with the highest mean reward. Specifically, we model the expected reward of an arm a has a linear relationship with its feature vector $\mathbf{x}_{t,a} \in \mathbb{R}^d$, and can be represented as

$$E[r_{t,a}|\mathbf{x}_{t,a}] = \mathbf{x}_{t,a}^\top \boldsymbol{\theta}_t^* \quad (4)$$

where $\boldsymbol{\theta}_t^* \in \mathbb{R}^d$, $\|\boldsymbol{\theta}_t^*\| \leq 1$, is a weight vector representing the accuracy model parameter to be learned online.

We use a LMMSE estimator to estimate $\boldsymbol{\theta}_t$. Specifically, let $\mathbf{D}_t \in \mathbb{R}^{m \times d}$ and $\mathbf{c}_t \in \mathbb{R}^m$ are the input samples of the feature matrix and the corresponding reward vector at the round t , respectively, where m is the number of samples and d is the feature dimension. By applying the Bayesian Gauss-Markov Theorem, the estimated coefficient $\boldsymbol{\theta}_t$ at round t is derived as

$$\hat{\boldsymbol{\theta}}_t = (\mathbf{D}_t^\top \mathbf{D}_t + \mathbf{I}_d)^{-1} \mathbf{D}_t^\top \mathbf{c}_t, \quad (5)$$

where \mathbf{I}_d is a d -dimension identity matrix. The covariance of the estimation error \mathbf{A}_t above is

$$\mathbf{A}_t = \mathbf{D}_t^\top \mathbf{D}_t + \mathbf{I}_d \quad (6)$$

Following the proof in [21], we know that for any $\delta > 0$, $\alpha = 1 + \sqrt{\ln(2/\delta)/2}$, with probability at least $1 - \delta$ we have

$$|\mathbf{x}_{t,a}^\top \hat{\boldsymbol{\theta}}_t - \mathbf{x}_{t,a}^\top \boldsymbol{\theta}_t^*| \leq \alpha \sqrt{\mathbf{x}_{t,a}^\top \mathbf{A}_t^{-1} \mathbf{x}_{t,a}} \quad (7)$$

where α is a hyperparameter to balance exploration and exploitation: the larger α , the more emphasis on exploration, and vice versa. Accordingly, the UCB of the estimated reward for selecting arm a at round t can be computed as

$$s_{t,a} = \hat{\boldsymbol{\theta}}_t^\top \mathbf{x}_{t,a} + \alpha \sqrt{\mathbf{x}_{t,a}^\top \mathbf{A}_t^{-1} \mathbf{x}_{t,a}} \quad (8)$$

where $\mathbf{A}_t = \mathbf{D}_t^\top \mathbf{D}_t + \mathbf{I}_d$. Then, the arm selection rule at t is

$$a_t = \arg \max_{a \in \mathcal{A}} s_{t,a} \quad (9)$$

Finally, we describe the flow of BCRC in Algorithm 1. We first initialize all UCB of all arms (Lines 1–2). At time t , we compute the estimated coefficient $\hat{\boldsymbol{\theta}}_t$ and the covariance of the estimation error \mathbf{A}_t according to Eqs. (5) and (6) respectively based on the observed current context and the features of all arms (Lines 4–7). Then we compute the UCB of all arms (Lines 8–9) and output the most appropriate arm a_t with the highest UCB (Line 10).

V. IMPLEMENTATION

We implement a Gemini prototype as shown in Fig. 9. Here, we use an AWS DeepLens as the camera and an Intel Max10 as the dual-image FPGA. We overcome a set of challenges and present three necessary enhancements: 1) the FPGA images should switch automatically by instructions. However,

Algorithm 1: Bandit-based Computing Resource Control

Input: \mathcal{A}
Output: At iteration t , output arm a_t for context u_t

```

1 for all  $a \in \mathcal{A}$  do
2   Initialize  $s_{t,a} = 0$ ;
3 for  $t = 1, 2, 3, \dots, T$  do
4   Observe current context  $u_t$ ;
5   Observe features of all arm  $a \in \mathcal{A}$ :  $\mathbf{x}_{t,a} \in \mathbb{R}^d$ ;
6   Compute  $\hat{\theta}_t$  according to Eq.(5);
7   Compute  $\mathbf{A}_t$  according to Eq.(6);
8   for all  $a \in \mathcal{A}$  do
9      $s_{t,a} = \mathbf{x}_{t,a}^\top \hat{\theta}_t + \alpha \sqrt{\mathbf{x}_{t,a}^\top \mathbf{A}_t^{-1} \mathbf{x}_{t,a}}$ ;
10  Select arm  $a_t$  with the highest UCB, i.e.,
     $a_t = \arg \max_{a \in \mathcal{A}} s_{t,a}$ ;
    
```

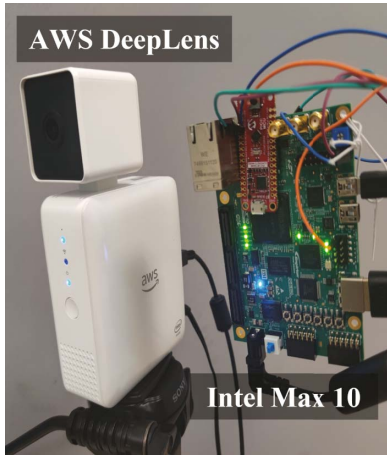


Fig. 9. Gemini prototype.

off-the-shelf Intel Max10 only provides manual switching. We conduct a hardware redesign to facilitate instruction-triggered image switching and enable multi-image functions on dual-image FPGA by modifying Altera Dual Configuration; 2) we develop an example adapter for Intel Max10 so that it can be registered into the Gemini system of a host edge device. The applications can then use the hardware functions in Section IV-A to access the FPGA computing resources, and 3) the BCRC algorithm in Gemini requires the pre-storage of CPU and GPU images. To allow multiple functions to be used in runtime, we design a simple and generic processing element (that can be utilized by different types of NN processing models and image implementation for generic instructions).

Enhancement to FPGA Image Switching. To complete a video analytics task, we need both filtering (CPU) and model inference (GPU), and thus a switch is needed. Intrinsically, the switch is not triggered by context changes, but by executing tasks. Context change will trigger our bandit algorithm to set parameters, e.g., video resolution, NN model choice. FPGA switch is between images, not the configurations, e.g., there is no change in the GPU image, only spending more time on the GPU image. In Max10, there are two control points, $SW1$ and $SW2$, see Fig. 10(a). $SW1$ is used to select the (next) image by the $CONFIG_SEL$ pin, and $SW2$ is used to trigger reconfiguration by the $RU_nCONFIG$ pin. To complete an image switch operation, $SW1$ and $SW2$ should be triggered manually.

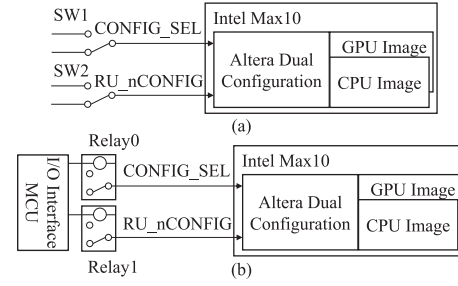


Fig. 10. Image switching enhancement.

 TABLE IV
 IMAGE SWITCHING TIME OF DIFFERENT FPGAS

FPGA	Intel Max 10	Intel Stratix 10	Xilinx Artix-7	Xilinx Zynq UltraScale
Maximal Clock Frequency	472.5 MHz	1,066 MHz	464 MHz	1,334 MHz
# Switching Cycle	4.5e5	2.5e5	4.6e5	1.6e5
Minimal Switching Time	~800 ns	~250 ns	~1 ms	~120 ns

We conduct a hardware redesign to replace the hand switch $SW1$ and $SW2$ by relays shown in Fig. 10(b). We remove the switch $SW1$ and $SW2$ and solder a relay at the original place of $SW1$ and $SW2$. When switching to the CPU image stored in the flash area one, Gemini switches $CONFIG_SEL$ pin by Relay0, then triggers the reconfiguration by pulling the $RU_nCONFIG$ pins down by Relay1, and vice versa. And we reprogrammed the Altera Dual Configuration module to alternatively change the allocation address of images to enable multiple images functions.

Although the image switching period counted by clock cycles, is fixed in different FPGA types, which is hard to optimize, we can further reduce it by increasing the FPGA operating frequency. Therefore, in Gemini, we maintain the operating frequency of Max10 at the highest level, 472.5 MHz. Compared with the 9 ms switching time refer to the Intel official reference frequency, 50 MHz, the switching time is reduced to about 10% of the original, about 800 ns. We also analyze many popular FPGAs switching time as shown in Table IV, and find that under maximum frequency, the switching time is able to satisfy common video analytics tasks.

FPGA Adapter. We show the workflow of the adapter developed for Intel Max10 in Fig. 11. The adapter consists of an FPGA configuration file and a control program. The FPGA configuration file records the function name, function code, parameter field, and the result field of each hardware function in the form of a function block. This file is loaded to the SRAM of the camera. The middleware monitors whether there is a hardware function call. Once the middleware detects a hardware function call, it searches the SRAM to find the corresponding function, sends the parameters, and finally collects the results from the result field.

GPU Image Implementation for Generic NN Computation. For NN processing models, different characters of NN layers vary in the FPGA implementation, depending on the computation type and the number of values that need to be accumulated (we call it accumulation frequency). We design a simple and generic processing element (SGPE) that can be

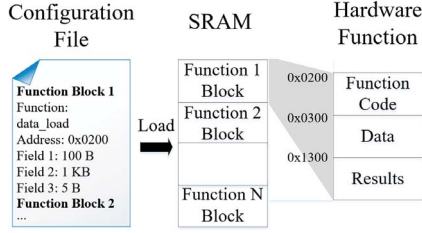


Fig. 11. The FPGA adapter.

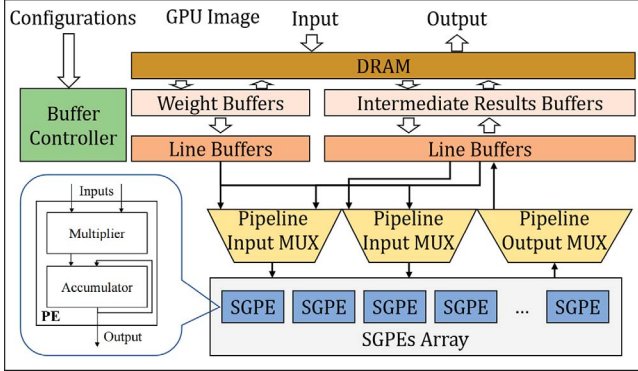


Fig. 12. A GPU image implementation for neural network model inference.

utilized by different types of NN computations. We show our SGPE in Fig. 12: each SGPE consists of a multiplier and an accumulator. The SGPE can complete the essential operation in all types of NN computation. In this way, we can implement one type of NN computation by combining SGPEs, and control the accumulation frequency by controlling the number of SGPEs. The GPU image implementation is shown in Fig. 12. The NN model's weights and intermediate results are routinely stowed away in line buffers and regularly swapped with a dynamic random-access memory (DRAM). Pipeline multiplexers (MUXs) govern the data flow exchange between the SGPEs and line buffers. The NN's structure is regulated by the buffer controller, while the weight buffers maintain the NN parameters during operation. The input and output results are held in DRAM and subsequently exchanged with the intermediate results buffers.

CPU Image Implementation for Generic Instruction Intensive Computation. Data flow in each instruction ties closed in the pipeline for the IIC workload in Frame Filtering/ ROI Extraction. Consequently, to decrease the total execution time, we implement a buffer-instruction-core architecture to preload data into buffers via Data Controller to guarantee that the instruction cores (IC) constantly run without idling, as illustrated in Fig. 13. Note that the instruction core contains an Instruction-Intensive algorithm that can handle a region of pixels and yield some intermediate results for the next instruction core. For instance, we employ Reducto [12] as the filtering algorithm and implement a CPU image to support it. The function of instruction cores here is to calculate the feature values in pixels. For the system resource monitor module, we use mpstat to record the system states. We implement the Gemini modules in Python with 2K+ lines of code.

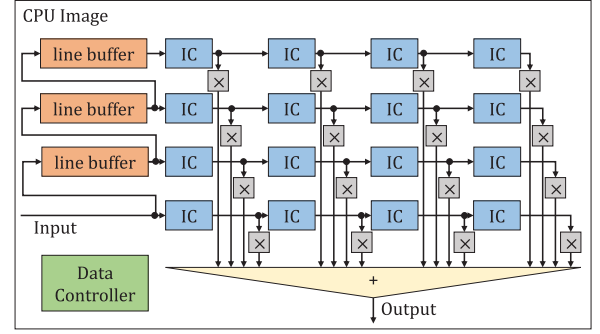


Fig. 13. A frame filtering sample of the CPU image implementation.

TABLE V
THE APPLICATION SPECIFICATIONS

Applications	VC	VPD
IIC Functions	SDD ROI Extraction [22]	Reduction Filtering [12]
DIC Functions	Model Inference	Model Inference
Resolutions (p)	144,280,540,720,1080	280,540,720,1080,2160
NN Models	TinyYOLO, SSD300, RetinaNet, YOLOv3	MobileNet, R-FCN, YOLOv3
Delay Requirement	40 ms	70 ms
Frame Rate	25 fps	30 fps

VI. EVALUATION

In this section, we evaluate the performance of Gemini with the aim to answer the following questions:

- How does Gemini compare to existing video analytics systems using the fixed CPU/GPU resources? (Section VI-B)
- To what extent can Gemini reduce the development effort for video analytics applications? (Section VI-C)
- How do the internal factors, e.g., the computing resource control algorithm, affect the performance? (Section VI-D)
- Why use dual-image FPGA for video analytics acceleration compared to traditional CPU-GPU systems? (Section VI-E)

A. Methodology

Testbed. We evaluate the Gemini prototype equipped with an AWS DeepLens camera and an Intel Max10 FPGA. The camera has an Intel Atom CPU with 8GB memory running Ubuntu OS-16.04 LTS. Intel Max10 FPGA has 50,000 configurable logic blocks and 4GB DDR3 memory.

Applications. We use two representative applications to evaluate Gemini. The specifications are shown in Table V.

- *Vehicle Counting (VC)* counts the number of vehicles in video footage. For the video dataset, we use the video captured by the traffic cameras in the city of Bellevue [13] that contains 24 hours of video of 38GB.

- *Vehicles and Pedestrians Detection (VPD)* paints the bounding box of vehicles and pedestrians in the video. We use the Auburn dataset [23] that contains 24 hours traffic video of 54GB.

Baselines. We compare Gemini against two state-of-the-art edge-side video analytics systems and an offline optimal scheme serving as the performance upper bound. They are open-source and have been widely used as benchmarks. We run all the baselines on the DeepLens-Max10 testbed. DeepLens

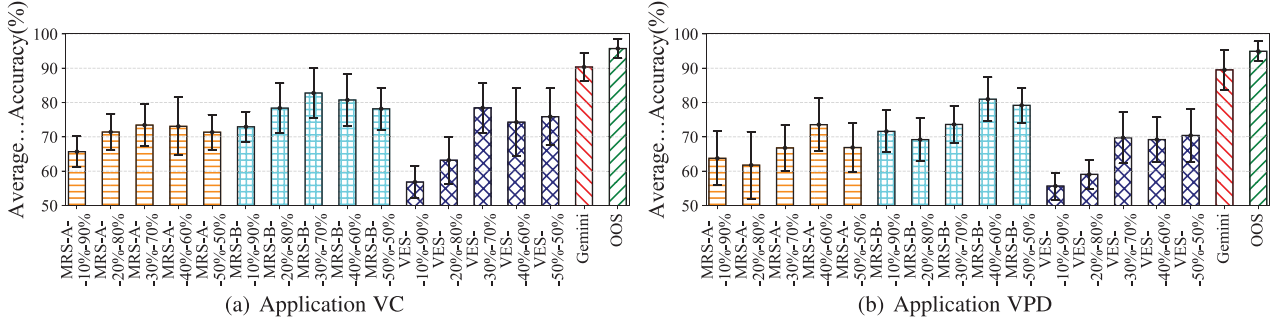


Fig. 14. The average analytics accuracy comparison.

responses for transmitting the frame to Max10. We use FPGA to emulate CPU and GPU resources. Let s - $x\%$ - $y\%$ denote $x\%$ and $y\%$ FPGA time allocated as CPU and GPU resources for system s . Let s - $best$ represent the best-fixed CPU and GPU resources allocation for the system s .

- *Microsoft Rocket System (MRS)* [4] is the status quo real-time video analytics system. It detects the available computing resources and uses the built-in resolution selection and model selection algorithms to balance the accuracy and delay. We apply two various MRS with different algorithms based on two existing works [4], [24]. We annotate the system from [24] as *MRS-A* that lowers the computation cost and the one from [4] as *MRS-B* that is focusing on reducing the latency and improving the accuracy.

- *VideoEdge System (VES)* [25] is a real-time video analytics that collects the available resources of multiple cameras and then selects the video resolution and models to maximize the analytics accuracy. For a fair comparison, we implement a variant VideoEdge that only collects the available resource of a camera and processes video streams locally.

- *Offline Optimal Scheme (OOS)* is computed using dynamic programming with complete workload information. It outputs the optimal computing resource allocation strategy, the resolution, and the NN model. The offline optimal serves as an upper bound on the accuracy of an omniscient policy with complete knowledge of the future IIC and DIC workload.

Evaluation Metrics. We use three metrics to evaluate Gemini and the baselines: 1) *Analytics Accuracy* is the first priority. For the VC application, the analytics accuracy is computed as $1 - \frac{|r-g|}{g}$, where r is the number of vehicles in the analytics result, and g is the ground truth. For the VPD application, the analytics accuracy is computed as $\frac{v}{w}$, where v is the overlapping area between the localization box of the analytics result and the correct localization box, w is the area of the correct localization box; 2) *Latency Miss Rate* quantifies the percent of the video analytics tasks that do not meet the latency requirement set by a video analytics application; and 3) *Hardware Utilization* indicates how effectively the computing resource has been used in video analytics systems.

B. Overall Performance

1) *Improvement on Analytics Accuracy:* Fig. 14 shows the average analytics accuracy of Gemini, MRS-A, MRS-B and

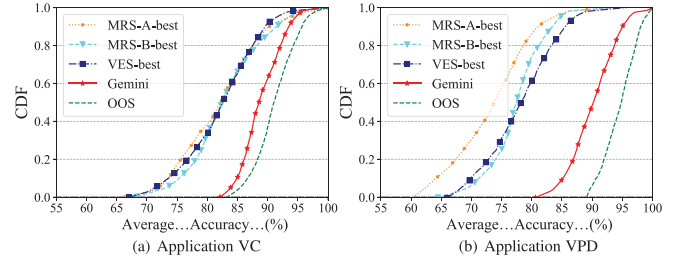


Fig. 15. The CDF of analytics accuracy comparison.

VES. For VC as shown in Fig. 14(a), we can see that MRS-A and VES achieve only 59.36%–81.92% of the accuracy of the OOS scheme. The average analytics accuracy of MRS-A is lower than 74% under all fixed CPU and GPU resources, which are qualitatively similar to VES. The MRS-B has about 7% higher average analytics accuracy than MRS-A. The value becomes even lower for VPD, as shown in Fig. 14(b), where MRS-A, MRS-B and VES achieve 58.63%–77.48% accuracy. It reveals that systems based on fixed CPU/GPU resources are far from satisfactory.

We can observe that Gemini outperforms MRS-A, MRS-B and VES over both applications. More specifically, Gemini outperforms MRS-A with an improvement in average analytics accuracy of 16.92%–24.67% and MRS-B with 7.58%–17.42%. The gap widens to 15.97%–27.76% for VES. Both experiments show that the performance gap of Gemini within 4.38%–5.40% of OOS scheme across both applications. Recall that the performance of OOS cannot be achieved in practice because complete knowledge of future workloads is required. It reveals that little room exists for video analytics systems without future knowledge to improve over Gemini in these scenarios.

Fig. 15 shows the Cumulative Distribution Function (CDF) of the average accuracy of Gemini, MRS-A, MRS-B and VES with the best-fixed CPU/GPU resources (i.e., MRS-A-*best*, MRS-B-*best*), and OOS. CDF is a function that describes the distribution of the accuracy taking on a value less than or equal to a given value. Here, it is used to visualize and analyze the distribution of our system and other's prediction accuracy across the dataset. For the application VC as shown in Fig. 15(a), only 19.32%, 20.32% and 19.21% of the accuracy of MRS-A-*best*, MRS-B-*best* and VES-*best* can reach the accuracy 85%, while 89.78% of the accuracy of Gemini can reach the accuracy 85%. These values of MRS-A, MRS-B and VES become even smaller

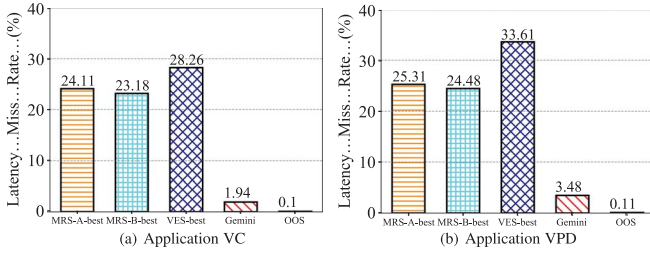


Fig. 16. The latency miss rate comparison.

in VPD as shown in Fig. 15(b), with only 11.52% of MRS-A-best, 12.06% of MRS-B-best and 12.88% of VES-best. However, Gemini can maintain 85.25% accuracy exceeding 85%. It illustrates Gemini output high accuracy results consistently.

2) *Reduction on Latency Miss Rate*: Fig. 16 compares the latency miss rates of Gemini, MRS-A-best, MRS-B-best, VES-best, and OOS. As shown, Gemini and OOS have similar low latency miss rates. Gemini outperforms MRS-best and VES-best by a large margin. Specifically, Gemini achieves a near-zero latency miss rate (1.94% in VC and 3.48% in VPD), while MRS-A-best (24.11% in VC and 25.31% in VPD), MRS-B-best (23.18% in VC and 24.48% in VPD), and VES-best (28.26% in VC and 33.61% in VPD) are much higher. Compared with MRS-A, MRS-B and VES, Gemini has a latency miss rate reduction from 85.78% to 93.14%.

Gemini can achieve better performance because MRS and VES model the workloads of video analytics tasks through one-time measurement, which leads to significant deviation in practice deploying, while Gemini learns the workloads through statistics with a much higher accuracy.

3) *Improvement on Hardware Utilization*: Fig. 17 shows the hardware utilization rate of different systems. We can see that MRS-A, MRS-B and VES suffer low hardware utilization, while Gemini can achieve a high hardware utilization under all scenarios. Specifically, the hardware utilization rate of Gemini reaches up to 93.58% in VC and 95.32% in VPD, and outperforms MRS-A, MRS-B and VES, with an improvement of 11.60% to 32.78% in VC, and 10.2% to 30.85% in VPD.

Please note that a higher hardware utilization means more computing resources are used to accelerate video analytics. Systems using fixed CPU and GPU resources result in a large amount of idling resources. Gemini can transform fixed CPU/GPU resources to elastic CPU/GPU resources, thus reducing the idling computing resources. The high hardware utilization explains why Gemini is able to achieve a high accuracy compared to baselines.

C. Application Development Effort

In this section, we investigate to what extent Gemini can simplify application development. We develop the application VC and VPD on two dual-image FPGAs, the Intel Max10 and the Xilinx Artix-7. We compare the development effort in terms of program length when programming with/without the hardware function abstraction and middleware provided by Gemini.

TABLE VI
DEVELOPMENT EFFORT W/O HARDWARE FUNCTION ABSTRACTION (HFA)
AND MIDDLEWARE (MW) AMONG DIFFERENT APPLICATIONS

APP	HFA Enable	MW Enable	# Code lines(Reduction Rate)		
			Max10	Artix-7	Max10+Artix-7
VC	×	×	21.1k	26.4k	47.5k
	✓	×	12.4k(-41.2%)	12.4k(-53.3%)	12.4k(-73.9%)
	×	✓	9.6k(-54.5%)	9.6k(-63.7%)	9.6k(-79.8%)
	✓	✓	7.9K(-62.6%)	7.9K(-70.0%)	7.9K(-83.4%)
VPD	×	×	32.5k	34.9k	67.4k
	✓	×	14.9k(-54.5%)	14.9k(-57.3%)	14.9k(-77.9%)
	×	✓	11.2k(-65.5%)	11.2k(-68.0%)	11.2k(-83.4%)
	✓	✓	8.6k(-73.5%)	8.6k(-75.3%)	8.6k(-87.2%)

Table VI shows the development effort of two applications on two different dual-image FPGAs. There are two key takeaways from these results. First, we find that the hardware function abstraction and middleware of Gemini can reduce the development effort significantly for all applications. For example, With hardware function abstraction and middleware, the total lines of code are reduced by 62.6% and 73.5% for VC and VPD on Max10.

Second, we observe that Gemini provides portability for application development. For example, with hardware function abstraction, the program length of VC deployed on Max10 and Artix-7 is 12.4k lines. If the developer develops the application on both Max10 and Artix-7, the program length is still 12.4k lines (Max10+Artix-7 in Table VI). It is because hardware function abstraction enables the developer to reuse the code without modification for FPGA specifications.

Third, in compile, the compile-time increases slightly on Intel Quartus Compiler, e.g., with and without abstraction differs about 0.21%. It is because the hardware function abstraction does not change the functional logic of FPGA. Thus, in runtime, no overhead is added to the execution time.

D. Component Analysis Study

We also explore Gemini's internal components better to understand their contributions to the performance of the system. We implemented three breakdown versions of Gemini to take a closer look at the contribution of each component: 1) **Gemini-A** has the asynchronous data transfer mechanism and middleware but does not enable the bandit-based computing resource control algorithm. We choose the best CPU/GPU resources configuration with the highest accuracy, 2) **Gemini-B** has the bandit-based computing resource control algorithm and middleware but does not enable asynchronous data transfer, and 3) **Gemini-C** has the bandit-based computing resource control algorithm and asynchronous data transfer but does not enable middleware. Figs. 18–20 show the comparison results that the accuracy gains, transmission time reduction and resolution selection distribution, brought by Gemini-A, Gemini-B, and Gemini-C are significant.

Gemini-A, as depicted in Fig. 18, we note a 3.31% decrease in accuracy relative to Gemini when applied to VC. This can be attributed to the implementation of elastic computing resource control in Gemini, which enhances its resource allocation capabilities, allowing it to determine optimal suitable configurations. For instance, the high-resolution image selection rate,

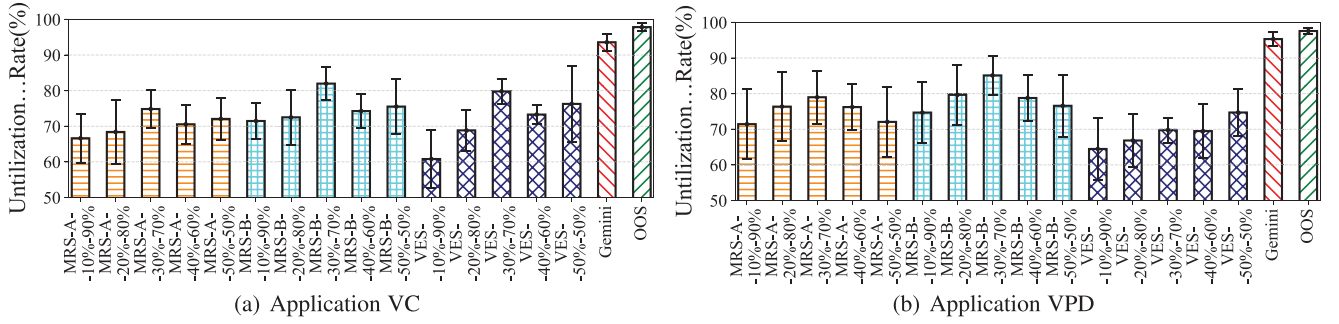


Fig. 17. The hardware utilization comparison.

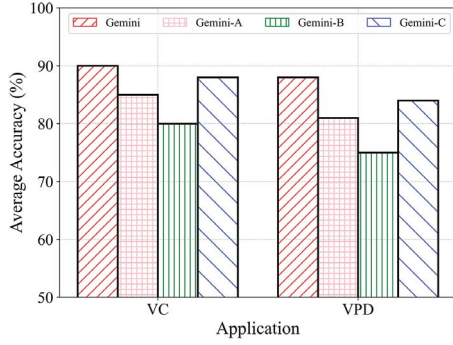


Fig. 18. Accuracy comparison for Component Analysis.

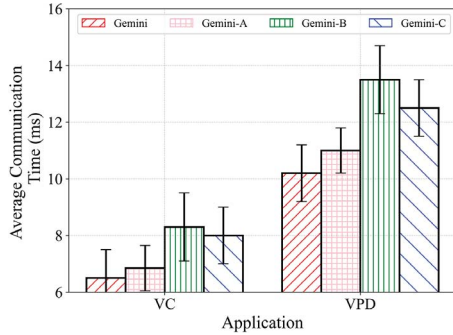


Fig. 19. Transmission time comparison for Component Analysis.

i.e., 1080p, increases from 21% to 35%. Regarding Gemini-B, Fig. 18 illustrates a 10.80% reduction in accuracy compared to Gemini in the context of VC. This outcome arises from the 24% reduction in transmission time for Gemini as opposed to Gemini-B, as demonstrated in Fig. 19, due to the deactivation of the asynchronous data transfer mechanism. Consequently, the high-resolution image selection rate, i.e., 1080p, increases from 12% to 35%. For Gemini-C, we observe a 2.01% decline in accuracy as a result of disabling middleware when contrasted with Gemini in VC, as shown in Fig. 18. This is a consequence of the 24% decrease in transmission time for Gemini in comparison to Gemini-C, as evidenced in Fig. 19. The saved time contributes to an increased high-resolution image selection rate, i.e., 1080p, increases from 17% to 35%.

We observe a similar accuracy reduction in VPD. These results indicate the importance of asynchronous data transfer

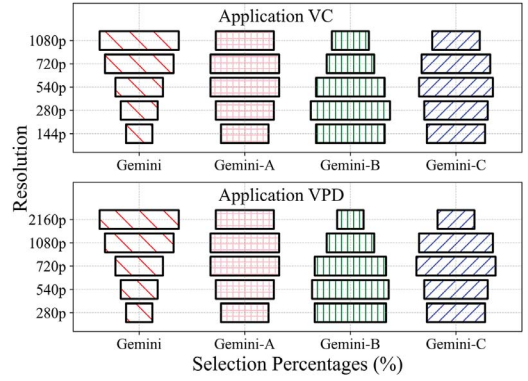


Fig. 20. The resolution selection distributions comparison for Component Analysis.

TABLE VII
VIDEO ANALYTICS PERFORMANCE OF INTEL MAX10, BANANA PI M2+, ORANGE PI 3 LTS AND RASPBERRY PI 4

Components	Asynchronous Data Transfer	Middleware	Bandit-based Resource Control
<i>Gemini-A</i>	✓	✓	×
<i>Gemini-B</i>	×	✓	✓
<i>Gemini-C</i>	✓	×	✓
<i>Gemini</i>	✓	✓	✓

mechanisms, elastic computing resource control and middleware. Gemini is able to combine the advantages to achieve better performance than using only one of them.

E. Between Dual-Image FPGA and CPU-GPU Systems

In this section, we compare the performance of traditional CPU-GPU systems with Gemini running on dual-image FPGA to highlight the advantages of dual-image FPGA for video analytics acceleration. We utilize three single-board computers, namely the Banana Pi M2+ [26], the Orange Pi 3 LTS [27], and the Raspberry Pi 4 [28], all equipped with both CPU and GPU.

As shown in Table VII, when executing the application VC, we observe that the average accuracy of the traditional CPU-GPU solution with the same price as Intel Max10 is lower than that of the dual-image FPGA, and the latency miss rate is higher. This is because the IIC task and DIC task in the application VC change dynamically, and the fixed CPU and GPU resources on Banana Pi M2+ and Orange Pi 3 cannot adapt to the changes, resulting in performance degradation. At the same time, the utilization rate of Banana Pi M2+ and Orange Pi 3 is also lower

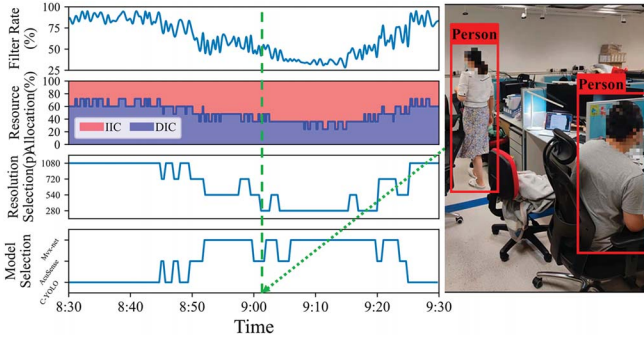


Fig. 21. The end-to-end operations of Gemini in the field. (The photo has been informed and approved by the people in it.)

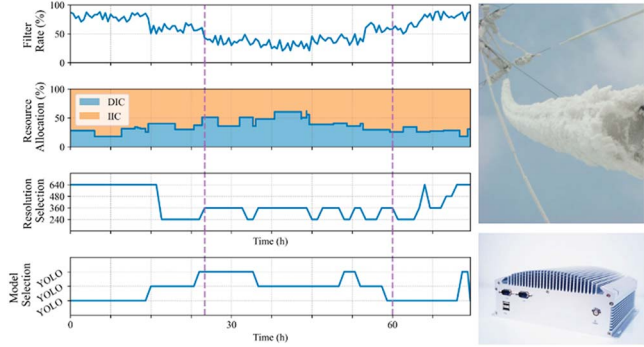


Fig. 22. The end-to-end operations of Gemini when monitoring ice disaster.

than that of Intel Max10, because Gemini can dynamically allocate resources on the dual-image FPGA according to changes in workloads to improve FPGA utilization. Furthermore, the Raspberry Pi 4 exhibits better average accuracy and latency miss rates than the Intel Max10, due to its improved hardware capabilities. However, its hardware utilization rate is significantly lower than that of the Intel Max10, resulting in wastage of hardware resources. Considering that the power consumption of FPGA is lower than that of CPU and GPU, using a dual-image FPGA to accelerate video analytics could offer both cost and efficiency advantages.

VII. CASE STUDIES

In this section, we apply Gemini to support an intrusion detection application and disaster monitor in an industrial system.

A. Case Study I: Intrusion Detection

We present a case study where we use our Gemini prototype (Fig. 9) to support an intrusion detection application that has been deployed in a computer laboratory in our department. The application applies a Hikvision intrusion detection application indoor [29]. It has three pre-trained models, Mvx-net, AcuSense [30], and Complete-YOLO. We ran the application supported by Gemini for over 8 hours.

Table VIII shows the average analytics accuracy, hardware utilization, and the latency miss rate of three periods on a day. The accuracy is holding at high accuracy (above 92.4%). The computing resource can be fully utilized as the hardware utilization ranges from 96.4% to 98.9%. We also observe that only

TABLE VIII
AVERAGE ANALYTICS ACCURACY, HARDWARE UTILIZATION RATE AND LATENCY MISS RATE OF THREE PERIODS ON A DAY

Hardware	Price (\$)	Average Accuracy	Latency Miss Rate	Hardware Utilization Rate
Banana Pi M2+	45	88.6%	12.3%	80.6%
Orange Pi 3 LTS	48	90.7%	8.12%	75.4%
Raspberry Pi 4	197	98.4%	0.81%	54.8%
Intel Max 10	50	96.1%	1.02%	98.7%

0.002% of video analytics tasks violate the latency requirement. It illustrates that Gemini can provide high service quality.

Fig. 21 shows the end-to-end operations of Gemini between 8:30 am to 9:30 am. In Fig. 21, the top graph shows the frame filtering rate over a period. The bottom three graphs show the amount of allocated CPU/GPU resources, the selected resolution, and the employed model over a period of time, respectively. We can see that Gemini adjusts computing resource allocation, video resolution, and model in runtime successfully. For example, at the time 9:05 am (green dash line in Fig. 21), a burst of persons appears in the video, the filter rate drops to 27%, and more frames are fed to the GPU. Gemini detects the increasing workloads on GPU, thus allocates more resources for GPU, and downsizes resolution to 280p, and model to Mvx-net to keep high accuracy.

We further estimate the Gemini overhead by computing the number of floating-point operations (FLOPs) of Gemini in this case. We find that Gemini has the computation of 25.63 MFLOPs, which is only 18.3% of MobileNet (140 MFLOPs). It takes only 1.7 ms to compute the dual computing resource allocation strategy, which occupied 2% of the total CPU time of the camera. In short, we believe that Gemini can successfully be deployed on laptops, mobile phones, or even on low-capacity cameras.

B. Case Study II: Disaster Monitor

We also apply our Gemini prototype to support edgebox deployed in an industrial system. The industrial system uses Jiangxing EdgeBox [31] to collect data from various sensors and execute video analytics about the frost and icing of the power grids.

The right top of Fig. 22 is the image collected by the camera connected with EdgeBox when the ice disaster occurred, and the right bottom is the prototype of EdgeBox. The subgraph on the left side of Fig. 22 shows the operation of Gemini in the ice disaster detection of the power grid in the industrial system. During the ice disaster monitoring period, Gemini can dynamically adjust computing resource allocation, video resolution and model. When the ice disaster has not yet appeared, the image of the power grid always remains unchanged for a long time. At this time, the filter rate is very high and the load is mainly on the CPU running the filtering algorithm. Due to the light GPU load, Gemini will select a higher resolution and more complex models to improve accuracy. In the range of the purple dash line in the figure, as the temperature drops, the power grid gradually freezes. At this time, the useful images of video analytics are increased, and the filter rate drops to 30%. Due to the increased load on the GPU, Gemini allocates more time to the GPU, while

reducing the resolution and changing the model to ensure high accuracy of the analysis results.

We test the overhead of Gemini in the industrial scenario. Unlike the previous case, which focuses on FLOPs, we care whether Gemini can successfully complete the task under the given delay requirements and frame rate. We found that the probability of Gemini failing to meet the requirements is less than 0.1%, which meets the needs of industrial control systems. Meanwhile, due to Gemini's low computing overhead, we believe that Gemini can be successfully used in various industries' scenarios.

VIII. RELATED WORK

In the research literature, Gemini falls into an *edge-side real-time video analytics system* that leverages a newly developed *accelerator*, i.e., the dual-image FPGA, with end-to-end *video analytics optimization*.

Edge-side video analytics systems: Early video analytics systems were developed in the cloud environment; some handle pre-stored videos [1] and some handle real-time videos, e.g., AStream [32] and Chameleon [10]. Edge-cloud video analytics systems were developed, e.g., DNN Surgery [2], VaBUS [33], where the workloads are partitioned between the edge device and the cloud. These systems optimized the analytics workloads through spatial-temporal contextual filtering, workload partitioning between the edge devices and the cloud, etc., under bottlenecks such as network delays, throughput variance.

With the increase of the edge-side computing power, real-time requirements and privacy concerns, edge-side real-time video analytics systems were developed. Microsoft Rocket [4] performs on-camera video analytics through dynamically adapting parameters and frame filtering techniques. VideoEdge [25] is a fully distributed framework to partition the video analytics pipeline across cameras and the edge cluster. BALB [34] exploits the spatial-temporal data correlations in multi-view video streams and performs target-to-camera assignments.

Existing edge-side video analytics systems run on fixed CPU/GPU resources and cannot effectively adapt to dynamic IIC/DIC workloads. Gemini leverages a newly developed dual-image FPGA to provide elastic dual computing resources and we present a full set of hardware and software designs.

Accelerators for video analytics: In the field of accelerated video analytics, there are a large number of researches using the common hardware CPU or GPU on the edge side for acceleration, but it has long been a trend to use FPGAs or ASICs to accelerate video analytics. And a large number of studies have shown that FPGAs and ASICs have more advantages in accelerating video analytics than CPUs or GPUs [35].

In the past years, we see a flourish of dedicated AI processors. Commercial products emerge such as Google TPU [36]. These processors specifically focus on DIC workloads. FPGA can also perform customized hardware acceleration [37]. There are studies using FPGA for video analytics acceleration, such as recognition and classification [38]. There is a history to develop

reprogrammable FPGAs [39]. Early FPGAs were based on static memory and cannot be reprogrammed. New generations of silicon have led to the SRAM-based FPGAs with reprogrammability. Then the programmable FPGAs can support partial reconfiguration, where a part of the FPGA can be reprogrammed while another part of the FPGA is being used. However, the reconfiguration normally takes minutes. The most recent FPGA development with external non-volatile memory allows dual-image storage and supports runtime reconfiguration through fast image switching. Gemini leverages such an advance to support elastic workloads.

Video analytics optimization techniques: There are optimization techniques to reduce the NN model size and thus decrease the model inference (DIC) workloads, e.g., model compression, model quantization. Small models have also been developed: DDSL [40] developed a dynamically distillability-and-sparsability learning framework for model compression. DECORE [41] used a reinforcement learning-based approach to automate the network compression process. PIT [42] performed light-weight Neural Architecture Search (NAS) to generate size-optimized and hardware-friendly temporal convolutional networks. There are also optimization techniques to reduce the number of frames fed into model inference, e.g., Reducto [12] has a lightweight filtering algorithm to filter out irrelevant frames. A region of targeted objects [22] was extracted based on common-feature analysis. These techniques incur IIC workloads, which need to be taken into consideration in resource-constrained edge devices. Gemini is an end-to-end system that leverages these techniques as components.

IX. CONCLUSION

In this paper, We developed Gemini, a new real-time video analytics system enhanced by a dual-image FPGA. Gemini can provide elastic computing resources in CPU and GPU in runtime. With its workload adaptation controller running a bandit-based algorithm, Gemini can adapt to the workload dynamics of video analytics applications in field, and substantially improve the video analytics accuracy. We presented a Gemini prototype implementation and evaluated Gemini through real-world video trace experiments and two case studies. We believe that Gemini can be extended into an edge-cloud video analytics system, or a collaborative video analytics system, where its effective control on dual computing resources can improve the performance of these systems as well.

REFERENCES

- [1] T. Xu, L. M. Botelho, and F. X. Lin, "VStore: A data store for analytics on large videos," in *Proc. 14th EuroSys Conf. (EuroSys)*, Dresden, Germany, Mar. 2019, pp. 1–17.
- [2] C. Hu, W. Bao, D. Wang, and F. Liu, "Dynamic adaptive DNN surgery for inference acceleration on the edge," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Paris, France, Apr. 2019, pp. 1423–1431.
- [3] Amazon, "AWS deeplens-deep learning enabled video camera for developers," 2019. Accessed: Aug. 9, 2023. [Online]. Available: <https://aws.amazon.com/deeplens>
- [4] Microsoft, "Microsoft rocket for live video analytics," 2017. Accessed: Aug. 9, 2023. [Online]. Available: <https://www.microsoft.com/en-us/research/project/live-video-analytics/>

- [5] "Machine learning image and video analysis-Amazon rekognition," 2021. Accessed: Aug. 9, 2023. [Online]. Available: <https://aws.amazon.com/rekognition/>
- [6] S. Deepika and V. Arunachalam, "Analysis & design of convolution operator for high speed and high accuracy convolutional neural network-based inference engines," *IEEE Trans. Comput.*, vol. 71, no. 2, pp. 390–396, Feb. 2022.
- [7] L. Izzouzi and A. Steed, "Integrating Rocketbox Avatars with the Ubiquitous Social VR platform," in *Proc. IEEE Conf. Virtual Reality 3D User Interfaces Abstr. Workshops (VRW)*, 2022, pp. 69–70.
- [8] M. Hanyao, Y. Jin, Z. Qian, S. Zhang, and S. Lu, "Edge-assisted online on-device object detection for real-time video analytics," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2021, pp. 1–10.
- [9] F. Loewenherz, V. Bahl, and Y. Wang, "Video analytics towards vision zero," *ITE J.*, vol. 87, no. 3, p. 25, 2017.
- [10] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica, "Chameleon: Scalable adaptation of video analytics," in *Proc. ACM SIGCOMM*, Budapest Hungary, Aug. 2018, pp. 253–266.
- [11] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "DeepDecision: A mobile deep learning framework for edge video analytics," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Honolulu, HI, USA, Apr. 2018, pp. 1421–1429.
- [12] Y. Li, A. Padmanabhan, P. Zhao, Y. Wang, G. H. Xu, and R. Netravali, "Reducto: On-camera filtering for resource-efficient real-time video analytics," in *Proc. ACM SIGCOMM*, Virtual Event, Aug. 2020, pp. 359–376.
- [13] R. Poddar, G. Ananthanarayanan, S. Setty, S. Volos, and R. A. Popa, "Visor: Privacy-preserving video analytics as a cloud service," in *Proc. USENIX Secur.*, Virtual Event, Aug. 2020, pp. 1039–1056.
- [14] C. Zhang, Q. Cao, H. Jiang, W. Zhang, J. Li, and J. Yao, "FFS-VA: A fast filtering system for large-scale video analytics," in *Proc. ICPP*, Eugene, OR, USA, Aug. 2018, pp. 1–10.
- [15] S. Zhang, G. Wu, J. P. Costeira, and J. M. Moura, "FCN-rLSTM: Deep spatio-temporal neural networks for vehicle counting in city cameras," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Venice, Italy, Oct. 2017, pp. 3687–3696.
- [16] Y. Chen, W. Li, C. Sakaridis, D. Dai, and L. Van Gool, "Domain adaptive faster R-CNN for object detection in the wild," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Salt Lake City, UT, USA, June 2018, pp. 3339–3348.
- [17] STMicroelectronics, "STM32F1 series." Accessed: Aug. 9, 2023. [Online]. Available: <https://www.st.com/en/microcontrollers-microprocessors/stm32f1-series.html>
- [18] A. Dhakal, S. G. Kulkarni, and K. Ramakrishnan, "Machine learning at the edge: Efficient utilization of limited CPU/GPU resources by multiplexing," in *Proc. IEEE Int. Conf. Netw. Protocols (ICNP)*, Madrid, Spain, Oct. 2020, pp. 1–6.
- [19] E. H. D'Hollander, B. Chevalier, and K. De Bosschere, "Calling hardware procedures in a reconfigurable accelerator using RPC-FPGA," in *Proc. Int. Conf. Field Programmable Technol. (ICFPT)*, 2017, pp. 271–274.
- [20] W. Wu, J. Yang, and C. Shen, "Stochastic linear contextual bandits with diverse contexts," in *Proc. AISTATS*, Palermo, Italy, Aug. 2020, pp. 2392–2401.
- [21] T. J. Walsh, I. Szita, C. Diuk, and M. L. Littman, "Exploring compact reinforcement-learning representations with linear regression," 2012, *arXiv:1205.2606*.
- [22] H. Kuang, L. Chen, F. Gu, J. Chen, L. Chan, and H. Yan, "Combining region-of-interest extraction and image enhancement for nighttime vehicle detection," *IEEE Intell. systems*, vol. 31, no. 3, pp. 57–65, May/Jun. 2016.
- [23] C. of Auburn AL, "City of Auburn Toomer's Corner webcam2," Jan 2019. Accessed: Jun. 8, 2022. [Online]. Available: <https://www.youtube.com/watch?v=hMYIc5ZPJL4>
- [24] A. Kewalramani, "Live video analytics with Microsoft Rocket for reducing edge compute costs," 2020. Accessed: Jan. 12, 2022. [Online]. Available: shorturl.at/cNUXY
- [25] C.-C. Hung et al., "VideoEdge: Processing camera streams using hierarchical clusters," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Bellevue, WA, USA, Oct. 2018, pp. 115–131.
- [26] "Banana Pi M2+," 2023. Accessed: Aug. 9, 2023. [Online]. Available: https://wiki.banana-pi.org/Banana_Pi_BPI-M2%2B
- [27] "Orange Pi 3 LTS," 2023. Accessed: Feb. 3, 2023. [Online]. Available: <http://www.orangepi.org/html/hardWare/computerAndMicrocontrollers/details/orange-pi-3-LTS.html>
- [28] "Raspberry Pi 4 Model B," 2023. Accessed: Aug. 9, 2023. [Online]. Available: <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/>
- [29] "Hikvision Intrusion Detectors," 2020. Accessed: May. 23, 2021. [Online]. Available: <https://www.hikvision.com/products/Alarm-Products/Hikvision-Intrusion-Detector>
- [30] "AcuSense Technology," 2020. Accessed: Aug. 9, 2023. [Online]. Available: www.hikvision.com/hk/products/IP-Products/Network-Cameras/acusense-products/
- [31] "JiangXing Intelligence Inc.," 2019. Accessed: Aug. 9, 2023. [Online]. Available: <http://www.jiangxingai.com>
- [32] B. Zhang, X. Jin, S. Ratnasamy, J. Wawrzyniak, and E. A. Lee, "AWStream: Adaptive wide-area streaming analytics," in *Proc. ACM SIGCOMM*, Budapest Hungary, Aug. 2018, pp. 236–252.
- [33] H. Wang et al., "VaBUS: Edge-cloud real-time video analytics via background understanding and subtraction," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 1, pp. 90–106, Jan. 2023.
- [34] S. Liu et al., "Multi-view scheduling of onboard live video analytics to minimize frame processing latency," in *Proc. IEEE 42nd Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2022, pp. 503–514.
- [35] X. Zhang, Y. Ma, J. Xiong, W.-M. W. Hwu, V. Kindratenko, and D. Chen, "Exploring HW/SW co-design for video analysis on CPU-FPGA heterogeneous systems," *IEEE Trans. Computer-Aided Des. Integr. Circuits Syst.*, vol. 41, no. 6, pp. 1606–1619, Jun. 2022.
- [36] J. Dean, "Recent advances in artificial intelligence via machine learning and the implications for computer system design," in *Proc. IEEE HCS*, Cupertino, CA, USA, Aug. 2017, pp. 1–116.
- [37] S. Biookaghazadeh, P. K. Ravi, and M. Zhao, "Toward multi-FPGA acceleration of the neural networks," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 17, no. 2, pp. 1–23, 2021.
- [38] H. Cho, J. Lee, and J. Lee, "FARNN: FPGA-GPU hybrid acceleration platform for recurrent neural networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 7, pp. 1725–1738, Jul. 2022.
- [39] A. Rodriguez, A. Otero, M. Platzner, and E. de la Torre, "Exploiting hardware-based data-parallel and multithreading models for smart edge computing in reconfigurable FPGAs," *IEEE Trans. Comput.*, vol. 71, no. 11, pp. 2903–2914, Nov. 2021.
- [40] Y. Liu, J. Cao, B. Li, W. Hu, and S. Maybank, "Learning to explore distillability and sparsability: A joint framework for model compression," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3378–3395, Mar. 2023.
- [41] M. Alwani, Y. Wang, and V. Madhavan, "DECORE: Deep compression with reinforcement learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2022, pp. 12339–12349.
- [42] M. Rissotto et al., "Pruning in time (PIT): A lightweight network architecture optimizer for temporal convolutional networks," in *Proc. 58th ACM/IEEE Des. Autom. Conf. (DAC)*, 2021, pp. 1015–1020.



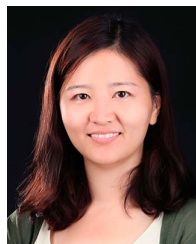
Chuang Hu received the B.S. and M.S. degrees in computer science from Wuhan University, in 2013 and 2016, respectively, and the Ph.D. degree from the Hong Kong Polytechnic University, in 2019. He is an Associate Researcher with the School of Computer Science, Wuhan University. His research interests include edge learning, federated learning/analytics, and distributed computing.



Rui Lu received the B.S. degree from the Department of Computing Science and Engineering, Southern University of Science and Technology, in 2019. He is currently working toward the Ph.D. degree with the Hong Kong Polytechnic University. His research interests include edge computing and edge learning.



Qianlong Sang received the B.S. degree in cyber science and engineering from Wuhan University, in 2022. He is currently pursuing the Ph.D. degree in computer science with the Wuhan University. His research interests include edge computing and big data systems.



Jin Zhang (Member, IEEE) received the Ph.D. degree in computer science from the Hong Kong University of Science and Technology, in 2009. She is an Associate Professor with the Department of Computer Science and Engineering, Southern University of Science and Technology. Her research interests include wireless sensing and mobile computing, wearable computing and mobile healthcare, mobile crowdsensing, and blockchain.



Huanghuang Liang received the B.S. and M.S. degrees in automation engineering from the Anhui University of Technology, in 2016, and the University of Electronic Science and Technology of China, in 2019, respectively. He is currently working toward the Ph.D. degree in computer science with Wuhan University. His research interests include cloud computing and big data systems.



Qing Li received the Ph.D. degree from Tsinghua University. He is currently an Associate Researcher with the Peng Cheng Laboratory, Shenzhen, China. His research interests include software defined networking, network function virtualization, in-network caching/computing, edge computing, and video delivery.



Dan Wang (Senior Member, IEEE) received the Ph.D. degree from Simon Fraser University. His research falls in general computer networking and systems, where he published in *ACM SIGCOMM*, *ACM SIGMETRICS*, and the *IEEE INFOCOM*, and many others. He is the Steering Committee Chair of *IEEE/ACM IWQoS*. His research interests include network architecture and QoS, smart building, and Industry 4.0.



Junkun Peng received the B.S. degree in information management and information systems from Shanghai University, Shanghai, China, in 2015. He is currently working toward the Master's degree in computer technology with Tsinghua University. His research interests include in-network caching/computing, real-time video transmission/analysis, and smart drone/robot swarm.



Dazhao Cheng (Senior Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from the Hefei University of Technology, in 2006, and the University of Science and Technology of China, in 2009, respectively. He received the Ph.D. degree from the University of Colorado, Colorado Springs, in 2016. He is currently a Professor with the School of Computer Science, Wuhan University. His research interests include big data and cloud computing.