Gemini: a Real-time Video Analytics System with Dual Computing Resource Control

Rui Lu*, Chuang Hu†, Dan Wang*, Jin Zhang§
*The Hong Kong Polytechnic University {csrlu, csdwang}@comp.polyu.edu.hk
†Wuhan University hchuchuang@gmail.com *Corresponding Author*
§Southern University of Science and Technology zhangj4@sustech.edu.cn

***Abstract*—Edge-side real-time video analytics systems recognize spatial or temporal events (e.g., vehicle counting) in a video stream. To meet the delay requirement, existing systems in smart edge cameras conduct video preprocessing to filter out unnecessary frames and model inference using appropriately selected neural network (NN) models. Video preprocessing is instruction-intensive computing (IIC) and executed by the CPU of the edge camera, and model inference is data-intensive computing (DIC) and executed by the GPU of the edge camera.**

In this paper, we show that the analytics accuracy of existing systems can largely vary in fields. The root cause is that video analytics applications have different *contents*, which result in *dynamic* IIC and DIC workloads. Unfortunately, intelligent cameras in fields have *fixed* CPU and GPU resources and cannot effectively adapt to workload dynamics. We develop Gemini, a new real-time video analytics system enhanced by a dual-image FPGA. The newly developed dual-image FPGAs can be pre-configured with two FPGA images with a key advantage of negligible image switching time. We thus pre-configure one CPU image and one GPU image and elastically multiplex the dual CPU-GPU resources in the *time* dimension. The Gemini system design requires both hardware and software revisions. We overcame a challenge that the application development on different dual-image FPGAs is hardware-dependent. We develop a new abstraction of hardware functions to make the Gemini system hardware-agnostic. It is also a challenge to adapt to the dynamic workloads and optimize video analytics accuracy. We develop a bandit learning approach to capture content dynamics and conduct dual computing resource control. We implement Gemini and show that Gemini can improve the analytics accuracy to 90.35%. We further evaluate Gemini by a case study where we use Gemini to support an intrusion detection application, and Gemini shows consistent high analytics accuracy.

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 (e.g., vehicle counting) 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] [3]. The hardware used for edge-side video analytics systems are smart cameras such as AWS DeepLens [4], with a CPU and a GPU. Typical edge-side video analytics systems include Microsoft Rocket [5], Amazon Rekognition [6], Canon Milestone [7], 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.

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 (vehicle counting) as an example (details in §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 86% 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, Xilinx Artix-7, etc., can pre-store two or more images in the FPGA image flash and switch them with negligible switching time. 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

2022 IEEE/ACM 7th Symposium on Edge Computing (SEC) | 978-1-6654-8611-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/SEC54971.2022.00020

978-1-6654-8611-8/22/$31.00 ©2022 IEEE
DOI 10.1109/SEC54971.2022.00020

switching address to enable more than two images stored on FPGA by simply modifying the source codes. Thus, we can pre-configure CPU images and 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 three 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, i.e., a video analytics application developed on one type of FPGA cannot be portable to a different type of FPGA. To solve this problem, 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; leading to one processor 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 the other. This achieves high-throughput microprocessor-FPGA communications.

Third, 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 is particularly suitable since the workloads depend on the video analytics application, not on the action choices of the computing 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 a case study where we use our prototype to support an intrusion detection application deployed in a laboratory for more than 8 hours. 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 (§II-B) can greatly vary in fields. We investigate the root causes of such a phenomenon.
- We develop Gemini ($\S III - \S IV$), 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 video analytics application workload dynamics and optimize the accuracy with dual computing resource control.
- We implement Gemini ($\S VI$) and evaluate Gemini with real-world video traces. We present a case study ($\S VII$), showing the end-to-end operations of Gemini in field.

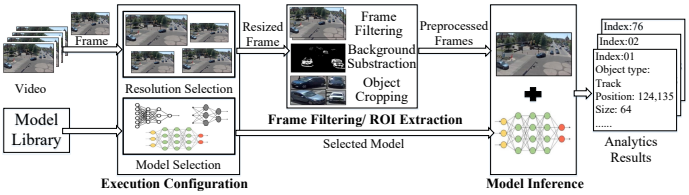


Fig. 1: The real-time video analytics pipeline.

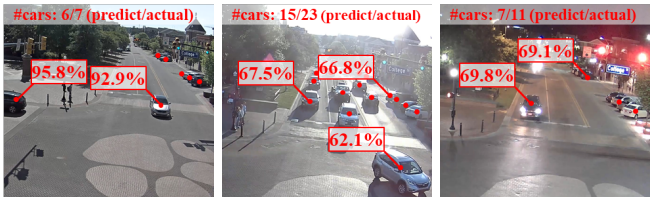
II. Motivation And Approach

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 [5]. One application atop the Rocket system is Microsoft Vision Zero [9], where NN models are pre-trained for vehicle counting of the City of Bellevue.

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 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) from this frame through background subtraction and object cropping technologies [13]. 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 apporaches, however they are heavy-weight and, to



(a) *Dawn Time* (b) *Rush Hours* (c) *Midnight*

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

TABLE I: Measurement results

Parameters & Utilization	*Dawn Time*	*Rush Hours*	*Midnight*
#vehicles per frame	7	23	11
Filtering Rate	82%	27%	64%
Resolution	540p	480p	1080p
Model	RetinaNet [16]	TinyYOLO [14]	SSD300 [15]
CPU Utilization Rate	62%	48%	100%
GPU Utilization Rate	99%	97%	76%

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[1]. The rocket system runs the built-in filtering algorithm [9], and the configuration algorithm [10] to select video resolutions from {144, 280, 540, 720, 1080}p and the NN model from pre-trained models of TinyYOLO [14], SSD300 [15], RetinaNet [16], and YOLOv3 [17]. For the video dataset, we use the video captured by the traffic cameras in the city of Bellevue [18] 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 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.

[1]Azure Stack Edge is currently not available in our country; we thus use AWS DeepLens, which is comparable with Azure Stack Edge.

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%

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 resource 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 different images to be stored in the user flash memory (UFM) and support fast switching with negligible switching time (~9ms for Intel Max10). 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 a dual-image FPGA. We configure an Intel Max10 FPGA to support Vision Zero. We divide the FPGA time into one-second periods, and we 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-$x\%$-$y\%$, indicating different

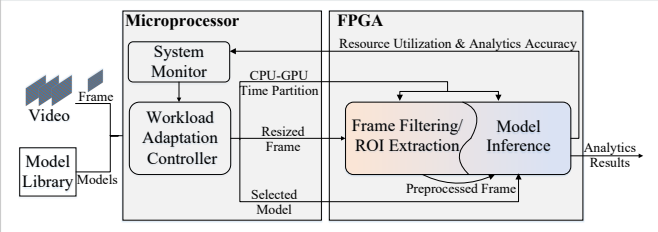


Fig. 3: The Gemini System.

CPU/GPU resources. 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 camera[2] 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 three 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 wastes.

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: 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 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 3: 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 predicts the workloads of IIC tasks and DIC tasks; and leverages 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, actions) 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.

[2]The 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.

TABLE III: Analysis of four representative video analytics applications and their required hardware support.

Function Abstraction	*Video Analytics Applications*		*Reducto [12]*			*FFS-VA [19]*			*FCN-rLSTM [20]*				*Faster-RCNN [21]*		
	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 module SWITCH_IMG(input clk, input [67
      :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

```
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 xdbr(.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

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

Fig. 6: Reducto filtering codes using Gemini hardware functions

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 ($SLAVE\_IMAGE$); Line 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 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. Line 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. Line 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. Line 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 agnostics to the dual-image FPGA specifics.

To develop a proper hardware function abstraction, we carefully analyze four representative video analytics applications, Reducto [12], FFS-VA [19], FCN-rLSTM [20], Faster-RCNN [21]. 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.

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 (Line 2-3), uploads the CPU image that implements the filtering function to the user flash memory (UFM) of FPGA (Line 4), switches to the CPU image (Line 5), transfers the frames to the FPGA (Line 6-7), executes the IIC CPU 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 64MHz [22], and the clock frequencies of Intel Max10 and Xilinx Artix-7 are 300MHz and 500MHz, respectively. The clock idling waste is non-trivial, i.e., as much as four-fifths of the FPGA clock cycles can be wasted for STM32f1 and Max10.

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 to the other processor to read the data. After finishing reading, the processor will be released for other computing tasks.

Discussion on the design choice. Our design on 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 [23], frequency virtualization [24], and CPU multiplexing [25]. 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., DRAM, the AXI bus, and PCI Express. With high-capacity communication physical lines, these interfaces can solve the capacity mismatch. However, hardware revision would require a new customized FPGA design.

C. A Bandit Learning Approach For Elastic Dual Computing Resource Control

1) Problem Formulation: We consider a video analytics application 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 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:

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

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:

$$t_i^C + t_i^G \leq \frac{1}{f} \tag{2}$$

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

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) (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 $A(v_i, m_i)$.[3]

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 of video analytics tasks.

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. For example, in a slot machine, the bandit learns the statistic outcomes (i.e., the expected differences) of the arms by pulling the arms.

[3]An edge camera can have additional fixed CPU and GPU resources. In such cases, we can develop a problem $\mathbf{P}_2$ 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.

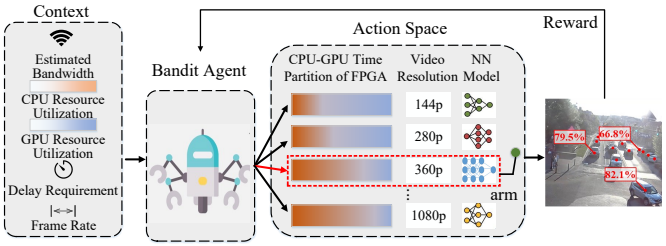


Fig. 7: The workflow of the bandit algorithm.

In a nutshell, 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 [26]. 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 [27].

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. 7), a bandit agent needs to make a sequence of decisions. At each time $t \in \{1, 2, \ldots, T\}$, the agent observers different *context* u_t and different *arms* a_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 configrurations 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 ten 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 $\boldsymbol{x}_{t,a} \in \mathbb{R}^d$, and can be represented as

$$\boldsymbol{E}[r_{t,a}|\boldsymbol{x}_{t,a}] = \boldsymbol{x}_{t,a}^\top \boldsymbol{\theta}_t^*. \tag{4}$$

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 $\boldsymbol{D}_t \in \mathbb{R}^{m \times d}$ and $\boldsymbol{c}_t \in \mathbb{R}^m$ 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

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

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

$$\boldsymbol{A}_t = \boldsymbol{D}_t^\top \boldsymbol{D}_t + \boldsymbol{I}_d \tag{6}$$

Following the proof in [28], we know that the we know that for any $\delta > 0$, $\alpha = 1 + \sqrt{ln(2/\delta)/2}$, with probability at least $1 - \delta$ we have

$$|\boldsymbol{x}_{t,a}^\top \hat{\boldsymbol{\theta}}_a - \boldsymbol{x}_{t,a}^\top \boldsymbol{\theta}_a^*| \leq \alpha \sqrt{\boldsymbol{x}_{t,a}^\top \boldsymbol{A}_t^{-1} \boldsymbol{x}_{t,a}}. \tag{7}$$

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

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

where $\boldsymbol{A}_t = \boldsymbol{D}_t^\top \boldsymbol{D}_t + \boldsymbol{I}_d$. Then, the arm selection rule at round t can be written as

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

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

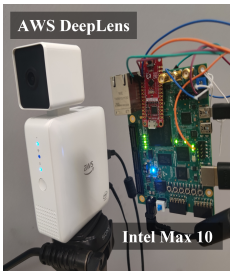


Fig. 8: Gemini prototype.

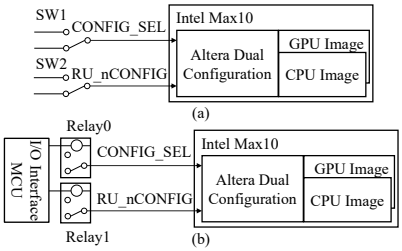


Fig. 9: Enhancement to FPGA Image Switching.

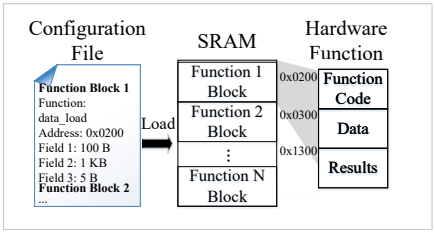


Fig. 10: The FPGA Adapter.

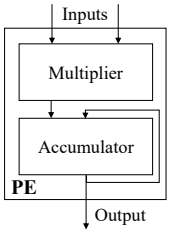


Fig. 11: SGPE in GPU Image.

Algorithm 1: Bandit-based Computing Resource Control

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

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

V. Implementation

We implement a Gemini prototype as shown in Fig. 8. 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, off-the-shelf Intel Max10 only provides manual switching.[4] 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 §IV-A to access the FPGA computing resources, and 3) the BCRC algorithm in Gemini needs to choose different NN models in runtime to maximize accuracy. However, FPGA requires the pre-storage of a GPU image. To allow multiple NN models to be used in runtime, we design a simple and generic processing element (SGPE) that can be utilized by different types of NN processing models.

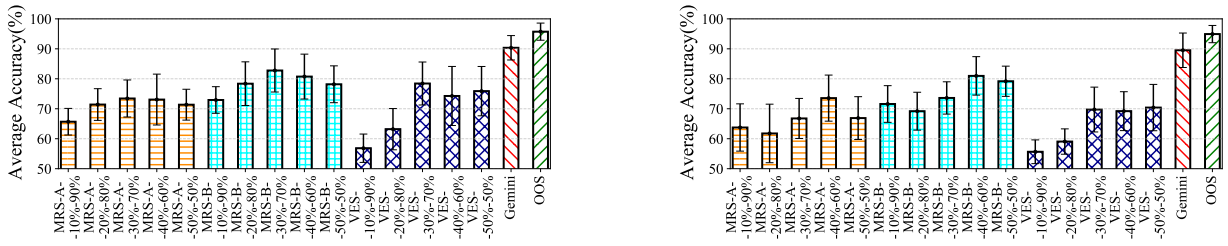
Enhancement to FPGA Image Switching. To complete a video analytics task (i.e., frame processing tens to hundreds ms), we need both filtering (CPU) and model inference (GPU), and thus a switch is needed. Intrinsically, the switch is not triggered by context changes, but by executing tasks. Context change 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. 9(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. We conduct a hardware redesign to replace the hand switch $SW1$ and $SW2$ by relays shown in Fig. 9(b). We remove the switch $SW1$ and $SW2$ and solider 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.

FPGA Adapter. We show the workflow of the adapter developed for Intel Max10 in Fig. 10. 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 control program monitors whether there is a hardware function call. Once the program detects a hardware function call, it searches the SRAM to find the corresponding function, sends the parameters to the parameter field, 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 utilized by different types of NN computations, similar to [29]. We show our SGPE in Fig. 11: 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 through controlling the number of SGPEs.

We employ Reducto [12] as the filtering algorithm and implement a CPU image to support it. For the system resource monitor module, we use mpstat [30] to record the system states. We implement the Gemini modules in Python with 2K+ lines of code. We develop the Gemini system in GitHub, and we plan to publicly release the codes as open sources.

[4]Xilinx Artix-7 does not have this problem.



(a) Application VC

(b) Application VPD

Fig. 12: The average analytics accuracy comparison.

TABLE IV: The Application Specifications.

Applications	*VC*	*VPD*
IIC Functions	SDD ROI Extraction [13]	Reduction Filtering [12]
DIC Functions	Model Inference	Model Inference
Resolutions (p)	144,280,540,720,1080	280,540,720,1080,2160
NN Models	TinyYOLO [14], SSD300 [15], RetinaNet [16], YOLOv3 [17]	MobileNet [31], R-FCN [32], YOLOv3 [17]
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? (§VI-B)
- To what extent can Gemini reduce the development effort for video analytics applications? (§VI-C)
- How do the internal factors, e.g., the computing resource control algorithm, affect the performance? (§VI-D)

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 IV.

• *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 [18] that contains 24 hours video streams of 38GB.

• *Vehicles and Pedestrians Detection (VPD)* paints the bounding box of the Vehicles and Pedestrians in the video. We use the Auburn dataset [33] that contains 24 hours traffic video stream 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 wile some other systems are not.

• *Microsoft Rocket System (MRS)* [5] is the status quo real-time video analytics system. It detects the available computing resources of a camera and uses the built-in resolution selection and model selection algorithms to balance the analytics accuracy and delay. We apply two various MRS with different algorithms based on two existed works [5], [34]. We annotate the system from [34] as *MRS-A* that lower down the computation cost and the one from [5] as *MRS-B* that is focusing on reducing the latency and improving the accuracy.

• *VideoEdge System (VES)* [35] 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 stream 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.

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.

Evaluation Metrics. We use three metrics to evaluate the performance of Gemini and the baselines.

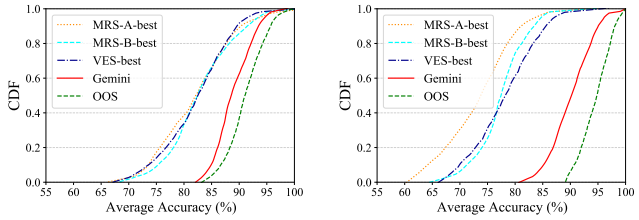
• *Analytics Accuracy* is the first priority. For the VC application, the analytics accuracy is computed as $1 - \frac{|r-g|}{g}$, where r is the number of vehicles in the analytics result, and g is the ground truth. For the VPD application, the analytics accuracy is computed as $\frac{v}{w}$, where 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.

• *Latency Miss Rate* quantifies the percent of the video analytics tasks that do not meet the latency requirement set by a video analytics application.

• *Hardware Utilization* indicates how effectively the computing resource has been used in video analytics systems.

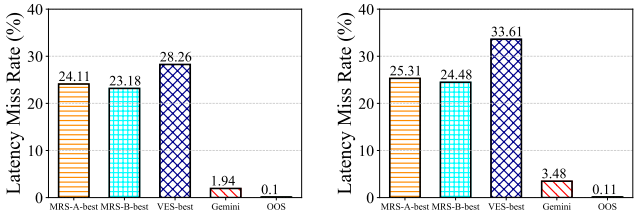
B. Overall Performance

1) Improvement on Analytics Accuracy: Fig. 12 shows the average analytics accuracy of Gemini, MRS-A, MRS-B and VES. For the application VC as shown in Fig. 12(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



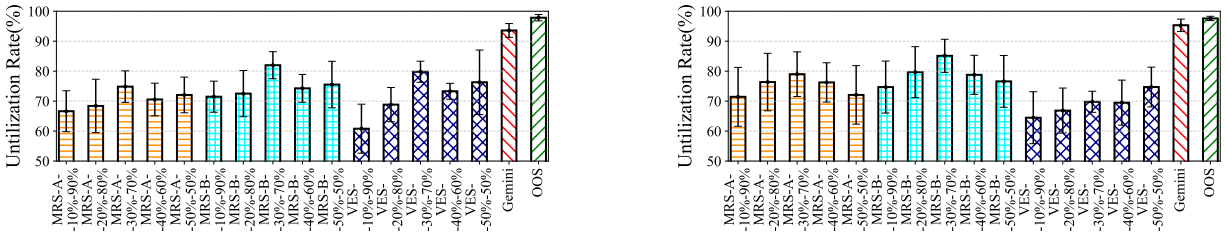
(a) Application VC (b) Application VPD

Fig. 13: The CDF of analytics accuracy comparison.



(a) Application VC (b) Application VPD

Fig. 14: The latency miss rate comparison.



(a) Application VC

(b) Application VPD

Fig. 15: The hardware utilization comparison.

MRS-B has an about 7% higher average analytics accuracy than MRS-A. The value becomes even lower for VPD, as shown in Fig. 12(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. 13 shows the 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. For application VC as shown in Fig. 13(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 in VPD as shown in Fig. 13(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. 14 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%.

TABLE V: Development effort w/o Hardware Function Abstraction (HFA) among different applications.

APP.	*HFA 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%)
VPD	×	32.5k	34.9k	67.4k
	✓	14.9k(-54.5%)	14.9k(-57.3%)	14.9k(-77.9%)

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. 15 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 reduces the idling computing resources. The high hardware utilization explains why Gemini is able to achieve a high accuracy compared to those using fixed CPU/GPU resources.

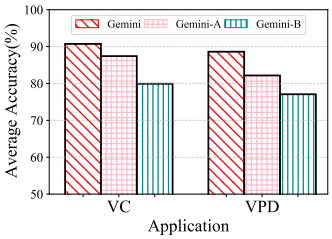


Fig. 16: Accuracy comparison for Component Analysis.

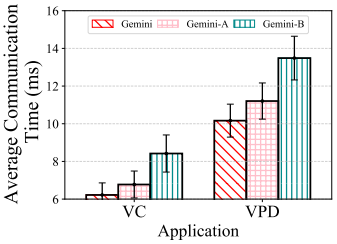


Fig. 17: Transmission time comparison for Component Analysis.

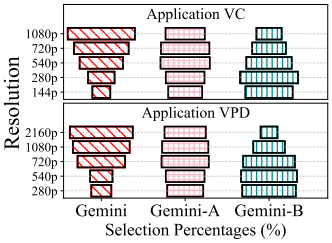


Fig. 18: The resolution selection distributions comparison for Component Analysis.

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 provided by Gemini.

Table V 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 of Gemini can reduce the development effort significantly for all applications. For example, With hardware function abstraction, the total lines of code are reduced by 41.2% and 54.5% 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 V). 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

In this section, we explore Gemini's internal components better to understand their contributions to the performance of the system. We implemented two breakdown versions of Gemini to take a closer look at the contribution of each component: 1) **Gemini-A** has the asynchronous data transfer mechanism but does not enable bandit-resource computing resource control algorithm. We choose the best CPU/GPU resources configuration with the highest accuracy, and 2) **Gemini-B** has the bandit-resource computing resource control algorithm but does not enable asynchronous data transfer.

Fig. 16-18 show the comparison results on application VC and VPD that the accuracy gains brought by both Gemini-A and Gemini-B are significant. For example, in application VC, Fig. 17 shows the transmission time of Gemini, Gemini-A, and Gemini-B. Gemini reduces about 24% of time than

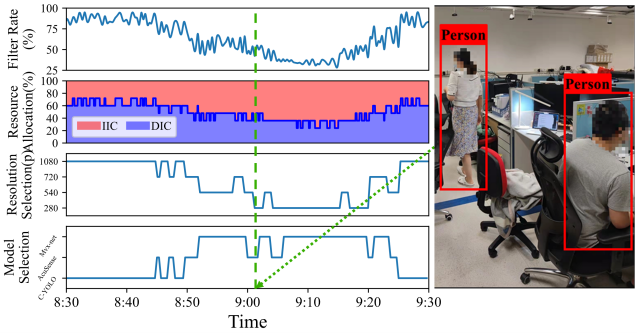


Fig. 19: The end-to-end operations of Gemini in the field. (The photo has been informed and approved by the people in it and erased private information.)

Gemini-B. And it leads to the fact that Gemini can infer higher resolution of input images than Gemini-A and Gemini-B as shown in Fig. 18.32.04% of image resolutions Gemini infers are 1080p, 1.71, and 2.66 times more than the other two. In Fig. 16, we observe the average analytics accuracy of Gemini, Gemini-A, and Gemini-B are 90.73%, 87.42%, and 79.93%, respectively. It means 3.31% of accuracy reduction will be incurred by disabling elastic computing resource control, and 10.80% accuracy reduction will be incurred by disabling asynchronous data transfer mechanism. We observe a similar accuracy reduction in VPD. These results indicate the importance of both asynchronous data transfer mechanisms and elastic computing resource control. Gemini is able to combine the advantages to achieve better performance than using only one of them.

VII. Case Study

We present a case study where we use our Gemini prototype (Fig. 8) 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 [36]. It has three pre-trained models, Mvx-net [37], AcuSense [38], and Complete-YOLO [39]. We ran the application supported by Gemini for over 8 hours. The content change in this scenario is in minute-level. As shown in Fig. 17, where content changed at 8:52am as compared to 8:55am, i.e., a one-person content changes to a three-person content.

Table VI shows the average analytics accuracy, hardware utilization, and the latency miss rate of three periods on a

TABLE VI: Average analytics accuracy, hardware utilization rate and latency miss rate of three periods on a day.

Time	*Average Accuracy*	*Hardware Utilization Rate*	**Latency Miss Rate**
6:30am–9:30am	92.4%	96.4%	0.002%
9:30am–12:00pm	97.9%	97.7%	0%
12:00pm–2:30pm	93.3%	98.9%	0%

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 0.002% of video analytics tasks violate the latency requirement between 8:30 am and 9:30 am. It illustrates that Gemini can provide high service quality.

Fig. 19 shows the end-to-end operations in the field of Gemini between 8:30 am to 9:30 am. In Fig. 19, 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. 19, a burst of persons appears in the video, the frame 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, changes the 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) [31]. 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.

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., AWStream [2] and Chameleon [10]. Edge-cloud video analytics systems were developed, e.g., DNN Surgery [3], EdgeBox [40], 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 [41], real-time requirements [42], as well as privacy concerns [43], edge-side real-time video analytics systems were developed. Microsoft Rocket [5] performs on-camera video analytics through dynamically adapting parameters and frame filtering techniques. VideoEdge [35] is a fully distributed framework to partition the video analytics pipeline across cameras and the edge cluster. Distream [44] performs workload allocation across cameras and edge clusters based on workload dynamics. 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 past years, we see a flourish of dedicated AI processors. Commercial products emerge such as Google TPU [45], NVIDIA SCNN [46], etc. These processors specifically focus on DIC workloads. FPGA can perform customized hardware acceleration [47]. There are studies using FPGA for video analytics acceleration, such as recognition [48] and classification [48]. There is a history to develop reprogramable FPGAs. Early FPGAs were based on static memory and cannot be reprogrammed. New generations of silicon have led to the SRAM-based FPGAs with reprogrammabililty. 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: Google developed small-scale DNNs for mobile platforms [49], MCDNN [50] generates alternative DNN models, etc. 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 [13] 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 a case study. We believe that Gemini can be extended into an edge-cloud video analytics system, or a collaborative video analytics sys-

tem, where its effective control on dual computing resources can improve the performance of these systems as well.

ACKNOWLEDGEMENT

Dan Wang's work is supported by National Key R&D Program of China under Grant No. 2020YFE0200500, GRF 15210119, 15209220, 15200321, ITF-ITSP ITS/070/19FP, CRF C5026- 18G, C5018-20G, PolyU 1-ZVPZ.

REFERENCES

[1] T. Xu, L. M. Botelho, and F. X. Lin, "Vstore: A data store for analytics on large videos," in *Proc. of EuroSys'19*, Dresden, Germany, Mar. 2019.

[2] B. Zhang, X. Jin, S. Ratnasamy, J. Wawrzynek, and E. A. Lee, "Awstream: Adaptive wide-area streaming analytics," in *Proc. of ACM SIGCOMM'18*, Budapest Hungary, Aug. 2018.

[3] C. Hu, W. Bao, D. Wang, and F. Liu, "Dynamic adaptive dnn surgery for inference acceleration on the edge," in *Proc. of IEEE INFOCOM'19*, Paris, France, Apr. 2019.

[4] Amazon, "Aws deeplens-deep learning enabled video camera for developers," 2019. [Online]. Available: https://aws.amazon.com/deeplens

[5] Microsoft, "Microsoft rocket for live video analytics," 2017. [Online]. Available: https://www.microsoft.com/en-us/research/project/live-video-analytics/

[6] "Machine Learning Image and Video Analysis-Amazon Rekognition," 2021. [Online]. Available: https://aws.amazon.com/rekognition/

[7] "Canon's surveillance solution - Crowd People Counter for Milestone XProtect," 2020. [Online]. Available: https://www.milestonesys.com/marketplace/canon-inc/crowd-people-counter/

[8] Z. Zhao, Z. Jiang, N. Ling, X. Shuai, and G. Xing, "Ecrt: an edge computing system for real-time image-based object tracking," in *Proc. of ACM SenSys'18*, 2018.

[9] F. Loewenherz, V. Bahl, and Y. Wang, "Video analytics towards vision zero," *ITE Journal*, 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. of ACM SIGCOMM'18*, Budapest Hungary, Aug. 2018.

[11] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "Deepdecision: A mobile deep learning framework for edge video analytics," in *Proc. of IEEE INFOCOM'18*, Honolulu, HI, USA, Apr. 2018.

[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. of ACM SIGCOMM'20*, Virtual Event, Aug. 2020.

[13] 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 Intelligent systems*, vol. 31, no. 3, 2016.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. of IEEE CVPR'16*, San Juan, PR, USA, June 2016.

[15] W. Liu, D. Anguelov, D. Erhan *et al.*, "Ssd: Single shot multibox detector," in *Proc. of ECCV'16*, Amsterdam, Netherlands, Oct. 2016.

[16] T.-Y. Lin, P. Goyal *et al.*, "Focal loss for dense object detection," in *Proc. of IEEE ICCV'17*, Venice, Italy, Oct. 2017.

[17] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv:1804.02767*, 2018.

[18] R. Poddar, G. Ananthanarayanan, S. Setty, S. Volos, and R. A. Popa, "Visor: Privacy-preserving video analytics as a cloud service," in *Proc. of USENIX Security'20*, Virtual Event, Aug. 2020.

[19] 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. of ICPP'18*, Eugene, OR, USA, Aug. 2018.

[20] 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. of IEEE ICCV'17*, Venice, Italy, Oct. 2017.

[21] 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. of IEEE CVPR'18*, Salt Lake City,UT,USA, June 2018.

[22] STMicroelectronics, "STM32F1 Series." [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32f1-series.html

[23] S. Ibrahim, T.-D. Phan, A. Carpen-Amarie *et al.*, "Governing energy consumption in hadoop through cpu frequency scaling: An analysis," *Future Generation Computer Systems*, vol. 54, pp. 219–232, 2016.

[24] W. Kanda, Y. Yumura, Y. Kinebuchi, K. Makijima, and T. Nakajima, "Spumone: Lightweight cpu virtualization layer for embedded systems," in *Proc. of IEEE EUC'08*, vol. 1, Shanghai, China, Dec. 2008.

[25] A. Dhakal, S. G. Kulkarni, and K. Ramakrishnan, "Machine learning at the edge: efficient utilization of limited cpu/gpu resources by multiplexing," in *Proc. of IEEE ICNP'20*, Madrid, Spain, Oct. 2020.

[26] W. Wu, J. Yang, and C. Shen, "Stochastic linear contextual bandits with diverse contexts," in *Proc. of AISTATS'20*, Palermo, Italy, Aug. 2020.

[27] R. Mathonat, D. Nurbakova, J.-F. Boulicaut, and M. Kaytoue, "Seqscout: Using a bandit model to discover interesting subgroups in labeled sequences," in *Proc. of IEEE DSAA'19*, DC, USA, 2019.

[28] T. J. Walsh, I. Szita *et al.*, "Exploring compact reinforcement-learning representations with linear regression," *arXiv:1205.2606*, 2012.

[29] H. Cho, P. Oh *et al.*, "Fa3c: Fpga-accelerated deep reinforcement learning," in *Proc. of ASPLOS'19*, Providence, RI, USA, Apr. 2019.

[30] B. Gregg, "Linux performance," *[Online]. http://www. brendangregg. com/linuxperf. html*, 2018.

[31] Z. Qin, Z. Zhang, X. Chen, C. Wang, and Y. Peng, "Fd-mobilenet: Improved mobilenet with a fast downsampling strategy," in *Proc. of IEEE ICIP'18*, Taipei, Taiwan, Sept. 2018.

[32] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," *arXiv:1605.06409*, 2016.

[33] C. of Auburn AL, "City of auburn toomer's corner webcam2," Jan 2019. [Online]. Available: https://www.youtube.com/watch?v=hMYIc5ZPJL4

[34] A. Kewalramani, "Live Video Analytics with Microsoft Rocket for reducing edge compute costs," 2020. [Online]. Available: https://www.microsoft.com/en-us/research/publication/live-video-analytics-with-microsoft-rocket-for-reducing-edge-compute-costs/

[35] C.-C. Hung, G. Ananthanarayanan, P. Bodik *et al.*, "Videoedge: Processing camera streams using hierarchical clusters," in *Proc. of IEEE/ACM SEC'18*, Bellevue, WA, USA, Oct. 2018.

[36] "Hikvision Intrusion Detectors," 2020. [Online]. Available: https://www.hikvision.com/products/Alarm-Products/Hikvision-Intrusion-Detector/

[37] V. A. Sindagi *et al.*, "Mvx-net: Multimodal voxelnet for 3d object detection," in *Proc. of IEEE ICRA'19*, Montreal, Canada, May 2019.

[38] "AcuSense Technology," 2020. [Online]. Available: www.hikvision.com/hk/products/IP-Products/Network-Cameras/acusense-products/

[39] M. Simony, S. Milzy, K. Amendey, and H.-M. Gross, "Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds," in *Proc. of ECCV'18*, Munich, Germany, Sept. 2018.

[40] B. Luo, S. Tan, Z. Yu, and W. Shi, "Edgebox: Live edge video analytics for near real-time event detection," in *Proc. of IEEE/ACM SEC'18*, Bellevue, WA, USA, Oct. 2018.

[41] Z. Xu, F. Yu, Z. Qin, C. Liu, and X. Chen, "Directx: Dynamic resource-aware cnn reconfiguration framework for real-time mobile applications," *IEEE TCAD'20*, vol. 40, no. 2, pp. 246–259, 2020.

[42] J. Lee, B. Varghese, R. Woods, and H. Vandierendonck, "Tod: Transprecise object detection to maximise real-time accuracy on the edge," in *Proc. of IEEE ICFEC'21)*, 2021.

[43] W. Du, A. Li, P. Zhou, B. Niu, and D. Wu, "Privacyeye: A privacy-preserving and computationally efficient deep learning-based mobile video analytics system," *IEEE TMC'21*, 2021.

[44] X. Zeng, B. Fang, H. Shen, and M. Zhang, "Distream: scaling live video analytics with workload-adaptive distributed edge intelligence," in *Proc. of ACM SenSys'20*, Virtual Event, Nov. 2020.

[45] J. Dean, "Recent advances in artificial intelligence via machine learning and the implications for computer system design," in *Proc. of IEEE HCS'17*, Cupertino, CA, USA, Aug. 2017.

[46] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli *et al.*, "Scnn: An accelerator for compressed-sparse convolutional neural networks," *ACM SIGARCH Computer Architecture News*, vol. 45, no. 2, pp. 27–40, 2017.

[47] S. Biookaghazadeh, P. K. Ravi, and M. Zhao, "Toward multi-fpga acceleration of the neural networks," *ACM JETC'21*, pp. 1–23, 2021.

[48] S. Han, J. Kang, H. Mao, Y. Hu, X. Li *et al.*, "Ese: Efficient speech recognition engine with sparse lstm on fpga," in *Proc. of ACM/SIGDA FPGA'17*, Monterey, CA,USA, Feb. 2017.

[49] A. Gordon, E. Eban, O. Nachum *et al.*, "Morphnet: Fast & simple resource-constrained structure learning of deep networks," in *Proc. of IEEE CVPR'18*, Salt Lake City,UT,USA, June 2018.

[50] S. Han, H. Shen, M. Philipose *et al.*, "Mcdnn: An approximation-based execution framework for deep stream processing under resource constraints," in *Proc. of ACM MobiSys'16*, Singapore, June 2016.