Volume 5, Issue 1, March 01, 2020

TC-CPS Newsletter

Technical Articles

Summary of Activities

Call for Contributions

IEEE TC-CPS Cyber-Physical Systems

Multi-Modal Attack Detection for Cyber-Physical Additive Manufacturing

Shih-Yuan Yu, Arnav Vaibhav Malawade, Mohammad Abdullah Al Faruque

Department of Electrical Engineering and Computer Science, University of California, Irvine

1 Introduction

Cyber-Physical Additive Manufacturing (AM) constructs a physical 3D object layer-by-layer according to its digital representation and has been vastly applied to fast prototyping and the manufacturing of functional end-products across fields. The computerization of traditional production processes propels these technological advancements; however, this also introduces new vulnerabilities, necessitating the study of cyberattacks on these systems [12]. The AM Sabotage Attack is one kind of kinetic cyberattack that originates from the cyber domain and can eventually lead to physical damage, injury, or even death [2]. By introducing inconspicuous yet damaging alterations in any specific process of the AM digital process chain, the attackers can compromise the structural integrity of a manufactured component in a manner that is invisible to a human observer. If the manufactured objects are critical for their system, those attacks can even compromise the whole system's structural integrity and pose a severe safety risk to its users. For example, an inconspicuous void (less than 1 mm in dimension) placed in the 3D design of a tensile test specimen can reduce its yield load by 14% [11]. However, security studies primarily focus on securing digital assets [10], overlooking the fact that AM systems are CPSs.

The AM system, or the printer, is comprised of a set of connected hardware components, and thus can unintentionally produce analog emissions during the operation of printing through different physical side-channels such as acoustics, electromagnetic radiation, vibration, and power. In AM systems, the information flow in the cyber domain has at least one corresponding control signal sent to the physical domain. This signal flow, in turn, actuates the physical processes accordingly, resulting in side-channel emissions that have a high degree of mutual information with the digital control signals. This property allows our group to a series of research on utilizing the correlation between the two domains and validating that physical domain signals match their cyber domain counterparts [4, 9, 1, 6, 7, 3, 8, 5]. Sabotaging the structural integrity of 3D objects requires the attacker to make subtle variations to one or more of the sub-processes in the AM process chain, resulting in a change in the printer's control parameters and a corresponding change to its analog emissions. In this case, monitoring the operation of the targeted AM systems can be the most direct defense [4]. To detect such modifications, our group proposes an attack detection system that continuously monitors and analyzes the different side-channel information leaked during the operation of AM systems, allowing us to identify unusual analog emissions resulting from potential sabotage attacks.

2 The AM Process Chain

The manufacturing of 3D objects in AM requires a chain of cyber-physical processes [12]. We primarily describe the process chain for the Fused Deposition Model (FDM) based AM. As shown in Figure 1, the process chain begins in the cyber domain with an idea for a 3D object. The first item on the workflow is to substantiate the design specifications using Computer-Aided Design (CAD) tools. Next, the generated CAD model gets converted into the STereoLithography (STL) format that uses a series of triangles to model the surface geometry. In the Computer-Aided Manufacturing (CAM) process, the slicing algorithms convert the STL files into a layer-by-layer description

file which instructs the printer on how to create the 3D object. The description file uses a Numerical Control Programming Language called G/M-code. The G-codes describe the motion and flow settings of the printer, determining the speed of the nozzle along each axis and the amount of material to deposit at each printing step. As an example, *G1 F2100 X5 Y6 Z1.2 E2.1* represents a single line of G-code for controlling the movement of the nozzle, where G1 means coordinated linear motion, F defines travel feed rate (speed) which is measured in mm/min, and distance and extrusion are measured in mm. On the other hand, the M-codes control the machine settings such as temperature, coolant. Lastly, the printer's firmware interprets each received G/M-code instruction into the corresponding control signals to actuate the printer hardware and, in turn, print the object.

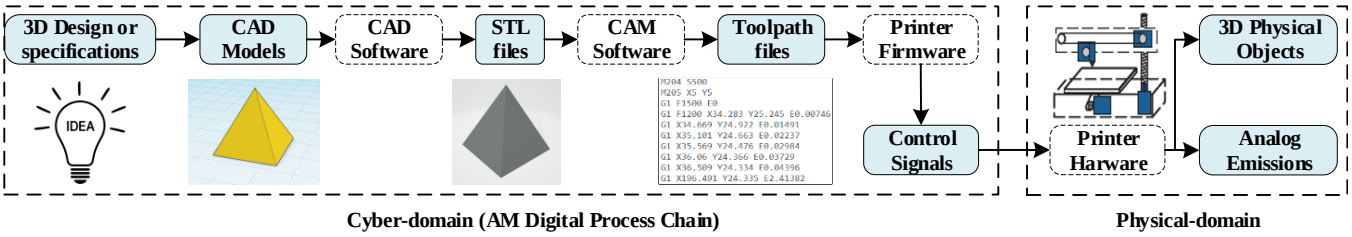


Figure 1: The AM process chain.

In the physical domain, the FDM-based printer uses thermoplastic filament materials to form 3D physical objects. The printer uses motors and belts to control the heated nozzle to melt the materials, depositing the extrusion onto the surface of the build plate at a precisely controlled rate. An array of stepper motors enables a fine-grained degree of control, each controlling movement along a single axis. These components act as a significant source of information leakage because of their analog emissions during printer operation. The analog emissions come from various side-channels of the printer (vibration, acoustic, magnetic, and power side-channels), which are streams of information outside of the primary data path containing information which can be correlated with the data in the primary path. In this case, the primary data path includes the control signals sent to the printer. The stepper motors are a major source of vibration and acoustic emissions due to the fluctuating radial force and torque ripples working on the stator core of the stepper motor. The varying electric field in the stepper motors also leaks data to the magnetic side channel. The power side-channel reveals the primary consumers of power in a 3D printer, such as the heating elements, stepper motors, fans, bed heater, and the internal control circuitry. In short, through these side-channels, the unintentional information leakage about the primary data path enables the inference of control signal values from side-channel information.

3 Sabotage Attack Detection for AM Systems

As for the threat model, the attackers can exploit multiple attack surfaces on the AM process chain (Figure 1) to sabotage manufactured parts. From the cyber domain, the adversaries can modify the intermediate forms of the 3D object by altering the integrity of the tools, firmware, or computers in the process chain, eventually resulting in the modification of the printer's control signals and the sabotage of the printed object. Cyber-security techniques can help mitigate risks further upstream in the digital process chain; however, AM systems often have network connectivity or physical access ports that can enable an adversary to exploit firmware vulnerabilities and compromise the system. To this end, we focus our on detecting adversarial attacks to the printer firmware that modify the control parameters directly. With this in mind, we assume that the G-codes sent to the AM system have not been previously tampered with and are representative of the correct 3D object model. In this case, the movement of the AM machine's internal components during a sabotage attack will not match the movements described by the instructions given in the G-code file, and thus, will result in different side-channel emissions. By utilizing these side-channel emissions with the G-codes sent to the printer, we can infer the values of the control signals and then determine if the printer firmware has been hacked.

The architecture of our proposed Sabotage Attack Detection System is described in Figure 2. To infer the control signals, the **System State Estimation Model** learns the relationship between the various side-channel emissions and the various control signals using supervised machine learning approaches. The dataset for training consists of the analog emissions from the **Benchmark Printer** along with their corresponding control signals, which are parsed from the G-codes. Once the mapping between analog emissions and control signals has been learned, our system first infers each control signal by continuously observing the analog emissions from the **Monitoring Printer**. The **Attack Detection Algorithm** then compares the unmodified control signals to the inferred signals to determine if a sabotage attack has occurred.

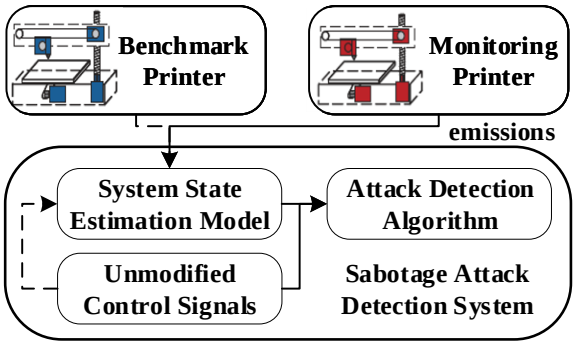


Figure 2: System Architecture.

4 Results

Our experimental setup is shown in Figure 3, consisting of an Ultimaker 3 3D-printer, four microphones, three vibration sensors (accelerometers), three magnetic sensors (magnetometers), and one current sensor. We use timestamps to match each G-code to its corresponding side-channel emissions recorded from the sensors. To evaluate the performance of our proposed detection system, we combined data from several different 3D-prints to produce a 20-gigabyte dataset containing 60,959 rows with 18,276 features per row. In the following sections, the selected results for state estimation and attack detection are shown (see [13] for more results).

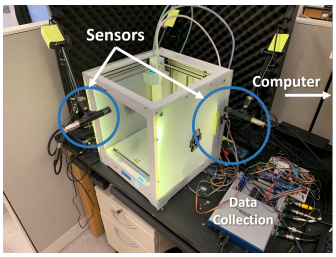


Figure 3: Experimental setup

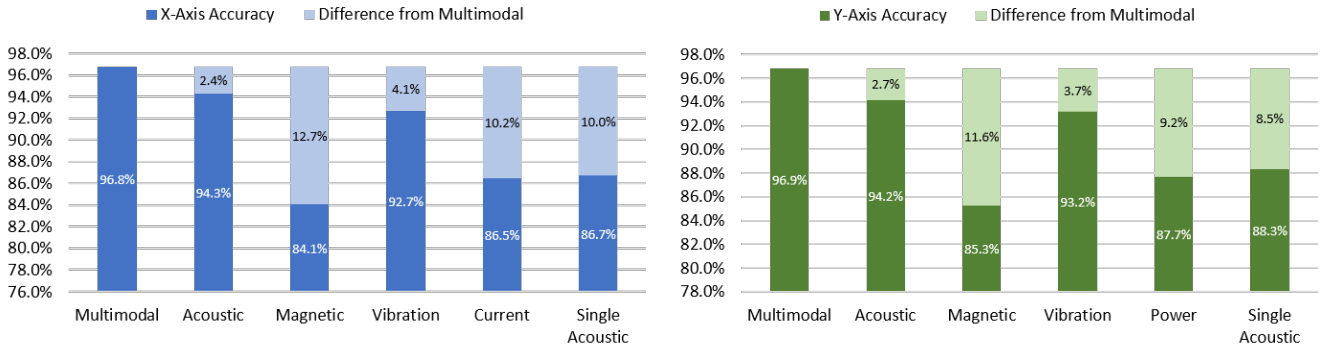
4.1 AM System State Estimation

The ability of our system to detect sabotage attacks depends on how precisely our system models the relationship between side-channel information and the corresponding machine control parameters. To improve our system's performance, we use multiple modalities of side-channel data for control signal estimation. To evaluate our multi-modal approach in comparison to single side-channel (unimodal) methods, we assessed the movement-axis prediction (A_x and A_y) performance of our classifiers using data from each modality. The results of the highest accuracy classifiers in each modality in comparison to our multi-modal approach and the single acoustic sensor approach presented in [4] are shown in Figure 4. As shown in the figure, our multi-modal technique outperforms the uni-modal methods as well as the technique from [4]. Notably, the acoustic and vibration modalities show the highest uni-modal accuracy and reach within 5% of the accuracy of the multi-modal technique. Overall, these results demonstrate that our multi-modal technique results in performance improvement over uni-modal methods.

4.2 AM Sabotage Attack Detection

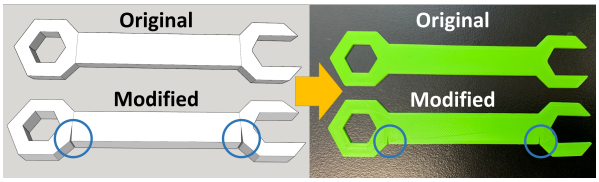
To evaluate our attack detection performance, we generate synthetic attacks by adding adversarial modifications to G-Codes. Then, we pass the unmodified sensor data and the modified G-code files as inputs to our attack detection system to evaluate its ability to detect mismatches. Our overall row-level accuracy is **99.17%** for movement-axis prediction. For axis-velocity prediction our average accuracy is **96.64%**. Our overall attack detection accuracy for all control parameters is **98.15%**. Although these attacks are synthetic, they provide empirical benchmarks to demonstrate the attack detection system's capabilities.

To validate these results in a real scenario, we tested two real-world sabotage attacks: a gear and a wrench. Since gears and wrenches are generally placed in high-stress scenarios, it is plausible that a kinetic cyberattack could compromise their structural integrity and significantly increase the chance of equipment failure. Moreover, these

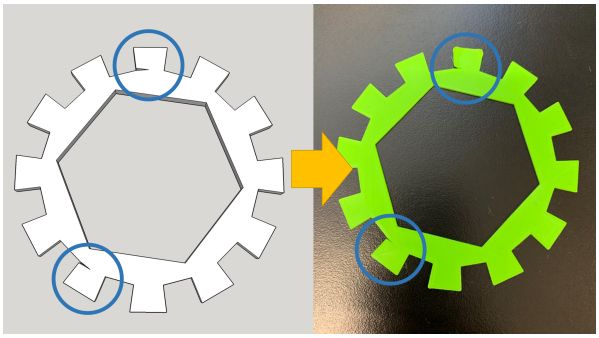


(a) (b)

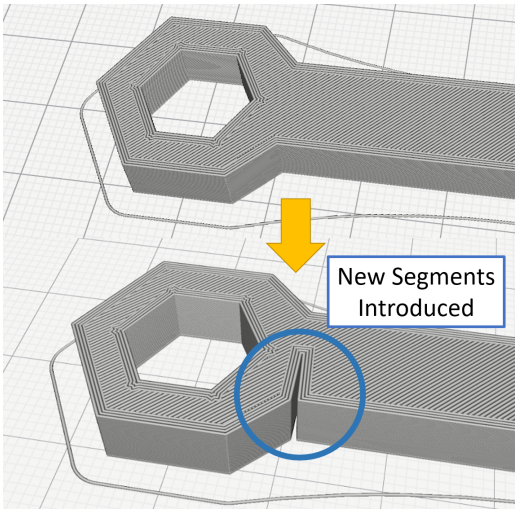
Figure 4: The comparison of multimodal approach and unimodal approaches in axis movement prediction.



(a) Original and modified wrench.



(b) Original and modified gear.



(c) Adversarial modifications in details.

Figure 5: The adversarially modified objects used for real-world testing.

parts serve as analogues to other load-bearing 3D printed parts to demonstrate that our detection system can identify kinetic cyberattacks on these types of components. The original and attacked wrenches are shown in Figure 5(a), and the adversarial modifications to the gear are shown in Figure 5(b). Although these specific modifications are visible to the human eye in the final print, this was done primarily for demonstrative purposes. These modifications can easily be concealed in internal layers of the print without modifying the outward appearance of the object, enabling it to pass visual quality inspections done by both machines and employees. For both test cases, the detection system was able to detect the attack successfully. These tests demonstrate that our methodology can be applied to real-world, practical AM tasks.

5 Conclusion

AM systems are core to the future of manufacturing and Industry 4.0. Since these systems are advancing in terms of complexity and connectivity as well as producing parts for increasingly critical and taxing applications, the security of these systems is a significant concern. Since AM systems are CPSs, they present a broad and diverse attack surface, making attack detection and prevention a daunting task. Our proposed sabotage attack detection system has

demonstrated the ability to detect various attacks by correlating multiple forms of physical-domain emissions of the AM system with cyber-domain information. We have achieved an overall detection accuracy of 98.15% on synthetic benchmarks and demonstrated that our system could detect two real-world scenarios of sabotage attacks. Although our solution addresses a significant security risk in AM systems, more work is needed to address the broad scope of potential AM vulnerabilities.

References

[1] M. A. Al Faruque, S. R. Chhetri, A. Canedo, and J. Wan. Forensics of thermal side-channel in additive manufacturing systems. *University of California, Irvine*, 2016.

[2] S. D. Applegate. The dawn of kinetic cyber. In *2013 5th international conference on cyber conflict (CYCON 2013)*, pages 1–15. IEEE, 2013.

[3] S. R. Chhetri and M. A. Al Faruque. Side channels of cyber-physical systems: Case study in additive manufacturing. *IEEE Design & Test*, 34(4):18–25, 2017.

[4] S. R. Chhetri, A. Canedo, and M. A. A. Faruque. Kcad: kinetic cyber-attack detection method for cyber-physical additive manufacturing systems. In *Proceedings of the 35th international conference on Computer-Aided Design*, page 74. ACM, 2016.

[5] S. R. Chhetri, S. Faezi, and M. A. Al Faruque. Information leakage-aware computer-aided cyber-physical manufacturing. *IEEE Transactions on Information Forensics and Security*, 13(9):2333–2344, 2018.

[6] S. R. Chhetri, S. Faezi, and M. A. A. Faruque. Fix the leak!: an information leakage aware secured cyber-physical manufacturing system. In *Proceedings of the Conference on Design, Automation & Test in Europe*, pages 1412–1417. European Design and Automation Association, 2017.

[7] S. R. Chhetri, N. Rashid, S. Faezi, and M. A. Al Faruque. Security trends and advances in manufacturing systems in the era of industry 4.0. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1039–1046. IEEE, 2017.

[8] S. Faezi, S. R. Chhetri, A. V. Malawade, J. C. Chaput, W. H. Grover, P. Brisk, and M. A. Al Faruque. Oligo-snoop: A non-invasive side channel attack against dna synthesis machines. In *NDSS*, 2019.

[9] A. Faruque, M. Abdullah, S. R. Chhetri, A. Canedo, and J. Wan. Acoustic side-channel attacks on additive manufacturing systems. In *Proceedings of the 7th International Conference on Cyber-Physical Systems*, page 19. IEEE Press, 2016.

[10] M. Holland, J. Stjepandić, and C. Nigischer. Intellectual property protection of 3d print supply chain with blockchain technology. In *2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, pages 1–8. IEEE, 2018.

[11] L. Sturm, C. Williams, J. Camelio, J. White, and R. Parker. Cyber-physical vunerabilities in additive manufacturing systems. *Context*, 7(2014):8, 2014.

[12] M. Yampolskiy, W. E. King, J. Gatlin, S. Belikovetsky, A. Brown, A. Skjellum, and Y. Elovici. Security of additive manufacturing: Attack taxonomy and survey. *Additive Manufacturing*, 21:431–457, 2018.

[13] S. Yu, A. V. Malawade, S. R. Chhetri, and M. A. Al Faruque. Sabotage attack detection for additive manufacturing systems. *IEEE Access*, 8:27218–27231, 2020.

Standardization and Diversity in Industrial Control Systems: Fallacies and Pitfalls from a Cybersecurity Standpoint

Dimitrios Tychalas*, Esha Sarkar*, Prashant Rajput*, Hadjer Benkraouda**, and Michail Maniatakos**
*NYU Tandon School of Engineering, Brooklyn, USA
**New York University Abu Dhabi, Abu Dhabi, UAE

1 Introduction

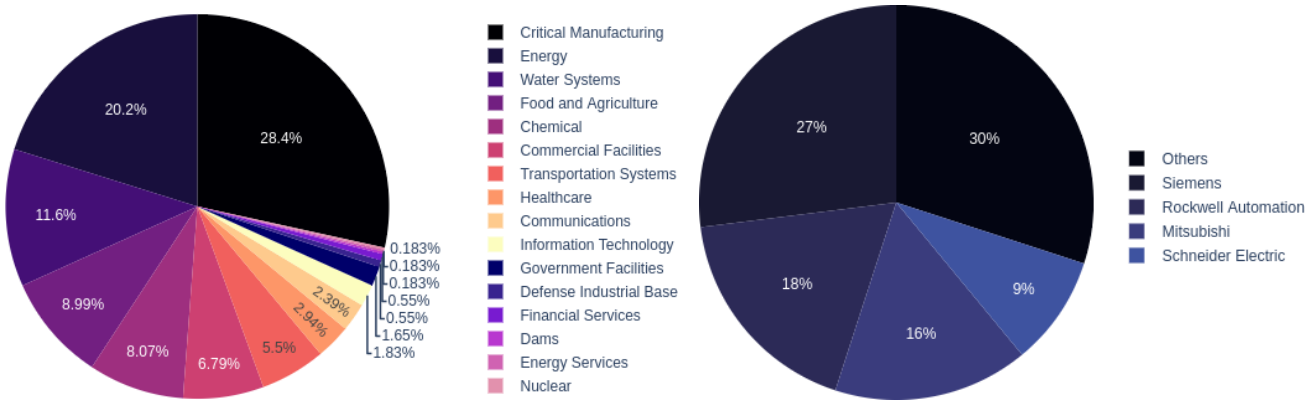
Industrial Control Systems (ICS) have become a staple of contemporary society, being aggressively integrated in the increasingly automated world of Industry 4.0. From simple temperature control to highly complex machinery mediation, ICS are specialized systems that regulate industrial processes. Given the variety of processes and industrial sectors, ICS must be widely diverse in terms of size and complexity in order to cater to the constant introduction of novel automated features in modern industries. This diversity is further expanded by the large number of associated vendors and the lack of standardization in many aspects of ICS function. From a cybersecurity perspective, high diversity rapidly raises the complexity of defense strategies for these devices, be it a dynamic or static system evaluation, malware analysis or machine learning-based approaches. In this article we will discuss the challenges, diversity, and lack of standardization impose on proactive and reactive defense methodologies applicable to ICS, elaborating on dynamic security assessment techniques, malware analysis, emulation, simulation, and machine-learning based detection.

2 Motivation

One of the principal challenges in securing ICS devices, such as Programmable Logic Controllers (PLC), is the diversity and lack of standardization among these devices. This diversity can be attributed to PLC vendors, with each vendor providing specialized hardware and customized software. In 2015, more than 70% of the PLC hardware market was controlled by four vendors namely Siemens, Rockwell Automation, Schneider Electric, and Mitsubishi [1]. Figure 1(b) illustrates the breakdown of the global PLC hardware market share of leading vendors. Each of these vendors equips its hardware with proprietary software, a tactic that tightly paired the PLC software and hardware markets [2]. This trend incurs a prohibitive time cost for reverse engineering and understanding the distinctive hardware and software configurations of each PLC. On one hand this has substantially raised the difficulty for researchers and engineers in performing comprehensive security analyses and developing general security solutions for PLC from different vendors. On the other hand, attackers also face the same challenges: the prohibitive time cost of finding vulnerabilities that affect many PLC with distinctive hardware and software, unintentionally increases the security of PLC.

In addition, the industries themselves and the variety of the standardized sectors, as defined by the US Department of Homeland Security [3], further complicate security application efforts on the industrial processes. Variety, coupled with the popularity of some sectors based on reported cybersecurity incidents [4] as illustrated in Figure 1(a), create an imbalance in the invested research effort. This translates to decreased availability of resources as well as a lack of in-depth understanding of the infrastructures for some of the less popular industrial sectors.

The lack of standardization among the various PLC vendors combined with the diversity of the industrial processes has lead to a security model that relies on obscurity and diversity rather than robust security mechanisms. Indeed, this phenomenon can be readily observed in contemporary methodologies that are routinely deployed to strengthen the cybersecurity posture of these environments. Whether acting proactively, trying to prevent potential vulnerabilities through security assessments, or reactively, opting for uncovering threats that exist in an ICS, researchers and engineers have to contest with the lack of standardization and sheer diversity.



(a) ICS cybersecurity incidents by sector [4]

(b) PLC vendor market share [1].

Figure 1: Statistics on ICS market and cybersecurity incidents.

3 Proactive Security

Proactive action, in the context of ICS cybersecurity, revolves around evaluating a target industrial device in order to uncover vulnerabilities that could potentially compromise the device itself or the connected industrial process. Techniques in this field can be direct, such as performing dynamic security assessment and malware analysis or indirect, e.g. emulating the target system or simulating the hosted industrial process.

3.1 Dynamic Security Evaluation

Security assessment covers a multitude of aspects in an ICS, spanning from application to system level, and from local I/O to network connections. Hardware/Software configuration diversity impedes a potential unified approach that could cover a large portion of currently deployed devices. This leads to vendor-specific research efforts which have led to similar exploitable conditions existing in devices from different, the uncovering of which happened years apart [5] [6]. Besides network evaluation, where most efforts for standardization focus on, system or application evaluation is lagging behind with considerable threats being uncovered on a monthly basis. Generic security assessment methodologies, popular to desktop and server grade computers, are fuzzing and symbolic execution. Fuzzing is deployed to extensively test corner cases in order to uncover potential threats, while symbolic execution is used to thoroughly analyze complex programs. Both methodologies are based upon comprehensive standards on program structure, where mature tools have been constantly evolving and proven to be very efficient in uncovering underlying vulnerabilities [7] [8]. For ICS however, the lack of standards in system software structure and applications format, inhibit the use of such powerful and established security assessment techniques. This fact can be readily observed in recent literature where a considerable amount of research effort is spent on attempting to target different systems hosting different applications each time [9] [10].

3.2 Malware Analysis

PLC have been studied extensively in literature as a pert of an ICS environment for performing malware analysis [11] [12] [13]. These devices are used to monitor sensor reading for implementing decisions based on a custom program to control actuators. They are predominantly based on the ARM architecture which creates additional difficulties for performing their security assessment. This stems from the seeming domination of publicly available datasets for malware by the x86 architecture based ELF format (Executable and Linkable Format) and Android-ARM based APK (Android Application Package) format. There is a strong need for an ARM ELF based malware dataset, such that performance of malware detection techniques for such PLC can be compared and evaluated. It has also prompted some researchers to port malware from x86 platforms to ARM or to create their own synthetic malware for evaluation of proposed techniques. Moreover, due to a lack of sufficient information on the system-

level details of the PLC, it is often hard to evaluate malware that are tied to a specific kernel version. To tackle this, researchers have replicated their test environment on comparable embedded devices and crated their own malware samples. Finally, PLC manufacturers such as WAGO are moving towards Linux-based environment which has facilitated porting malware to their PLC and their security assessment.

3.3 Emulation

Emulation has been established as a potent means of replicating the function of a system without the requirement of a physical device. In addition to a lower cost, emulation offers great scalability in any type of evaluation, given that multiple emulated instances can be deployed simultaneously. An emulated system consists of multiple subsystems, be it the main processor, memory configuration, and peripheral devices, in addition to the various integrated communication protocols such as SPI and I^2C. This variable hardware configuration in turn supports a similarly diverse amount of system software hosting the control functions of typical ICS. At the same time, vendors still cling to in-house closed-source developed monolithic firmware in comparison to highly moderated open-source modular software, such as Embedded Linux or Nucleus OS. When attempting to emulate a collection of devices, the engineer or researcher must acquire hardware configurations, which in most cases are withheld by the vendors, as well as obtain the system software driving each device, which is typically unique for each vendor requiring a case-by-case study [14]. This predicament is evident in the amount of recent emulation related research efforts where the aforementioned diversity pushes for individual studies [15] [16] instead of an effort for a more unified approach.

3.4 Simulation

Studies of cyberattacks on simulations of ICS processes are not representative of an actual ICS infrastructure. To elaborate, ICS simulations focus on process-level and neglect the mechanical intricacies involved. This inevitably results in analyses limited to process-aware inferences. Consider Damn Vulnerable Chemical Process (DVCP), implemented as a Matlab Simulink based simulation of Vinyl Acetate Process. This model allows researchers to study the changes in production of chemicals as a result of cyberattacks [17]. On the other hand, Secure Water Treatment (SWaT) testbed is a scaled-down implementation of an actual Reverse Osmosis based water treatment plant. It provides monitoring facility of the underlying process by using SCADA and uses mechanical components utilized in large-scale treatment plants [18]. This enables researchers to evaluate process-aware attacks as well as its associated mechanical impacts on SWaT. Such in-depth analysis is more representative of the impacts of cyberattacks on actual critical infrastructure. Building a scaled-down testbed is not plausible for complex critical infrastructures such as Multi-Stage Flash desalination plants with multiple stages of operation. To address this, researchers have started using simulation to study the impact of cyberattacks on process-level alterations along with its mechanical impact. Researchers in [19] performed a process-aware attack study on a Multi-Stage Flash (MSF) desalination plant Simulink model and exported the pressure values obtained from the process simulation to ANSYS for studying its effects on the mechanical components, in this case: pipes. Such a simulation based approach enveloping both process and mechanical focused analysis can help encourage in-depth analysis of cyberattacks on critical infrastructures.

4 Reactive Security

Reactive action for securing ICS correlates to anomaly detection, on a process or network level. In recent years, machine learning has become an invaluable tool for developing anomaly detection systems, such as detecting intentional misbehavior in network packets or process irregularities, offering flexibility as well as future-proofing. In both network or process based detection, a single rule cannot fit all situations. Therefore, it is crucial to have standardized datasets targeting each of the 16 ICS sectors. Currently, the available open-sourced datasets only focus on few sectors like water and waste-water treatment, chemical, and energy sectors. In the next subsection we discuss the two levels of anomaly detection and the available datasets for them.

4.1 Anomaly detection using network packets

The datasets reflecting misbehavior at the network level, like KDD99 [20], are targeted for advancement of generic Intrusion Detection Systems (IDS). Due to lack of ICS-based network datasets, even ICS-specific IDS use these datasets for evaluation of their detection algorithms [21]. Since common IT-based IDS cannot recognize the network dynamics of an ICS process, network packets of an ICS protocol (like MODBUS), from an actual (or simulated) ICS process is necessary for evaluation. A network-based ICS dataset, using a small testbed of physical devices and two Human Machine Interfaces (HMIs) hosting a water storage system and a gas pipeline system [22], was open-sourced in 2014. A further elaborate plant-based ICS dataset [23] was created to enhance the design of anomaly detection algorithms. The dataset includes MODBUS function codes, values, description along with source and destination IP. The time stamps help in detecting precise patterns for genuine, faulty, and intentionally malicious behavior. Since the dataset was built using a hardware-in-the-loop testbed, it captures the intricacies of communication between PLC, SCADA, Historian and engineering workstations. But the lack of standardization of these datasets impede the ICS-specific IDS research. Other ICS sectors have not been extensively explored and thus, it is difficult to design a network-based or host-based IDS for all ICS sectors.

4.2 Anomaly detection using process-aware data

For process-aware attack detection, there has been research on chemical plants [24], water treatment plants [23], energy sector [25], and gas pipeline [26]. However, there is a lack of standardization amongst the type of attacks presented in the datasets. Moreover, the attack vectors aim to achieve different attack goals, mostly causing a *conspicuous* change in a physical quantity, lacking stealth. Across the datasets presented in ICS literature, there is no standardization of attack types or goals.

In this context there is a need for classification of datasets aiming for advancement of different aspects of ICS research. Attack payloads which aim at causing extremely dangerous manipulations must be detected before the full-blown launch of the attack and should specifically aim at Intrusion Prevention research. The attack payloads that have smaller attack goals like minimal monetary gains but focus on stealth of the payload, may be used to advance Intrusion Detection research. The process-aware attack datasets can also be used to evaluate the mitigation algorithms for system recovery [27]. The process-aware datasets capture the complexity of ICS ecosystem comprehensively but lack of standardized datasets result in creation of local testbeds and evaluating algorithms on them without any evaluation on their quality.

The open-source ICS datasets are extracted either from simulations or hardware-in-the-loop testbeds which focus on specific physical quantities. Difficulty in standardizing datasets lies in the difficulty of standardizing the industrial processes themselves. As mentioned before, even different vendors implement the same process in different ways. Moreover, there are other facets of ICS data which remain unexplored: There is no standardized dataset on visual HMI data or PLC source codes and binaries. This limits the approaches taken to build attack detection, prevention, and mitigation mechanisms for ICS.

5 Conclusion

In this article we presented several challenges the sheer diversity in ICS devices and processes introduce to a potential structured and comprehensive security assessment. Lack of standardization and closed-source approaches will continue to inhibit security research for industrial settings, with singular case studies which have small impact while consuming significant resources. A firm step must be taken by the major actors in ICS development towards either adapting tried and effective solutions with the open source mindset for wide moderation or introduce new standards that will streamline security assessment and timely vulnerability discovery.

References

[1] Technavio, “Global plc hardware market 2016-2020,” Technavio Information Technology Research, 2015.

[2] Technavio, "Global plc software market 2016-2020," Technavio Information Technology Research, 2015.

[3] D. of Homeland Security, "Critical infrastructure sectors," https://www.dhs.gov/cisa/critical-infrastructure-sectors, [Online].

[4] I. CERT, "Threat landscape for industrial automation systems. h2 2018," https://ics-cert.kaspersky.com/reports/2019/03/27/threat-landscape-for-industrial-automation-systems-h2-2018/, 2018.

[5] S. H. Houmb, "More exploits: the great plc hack," https://www.controldesign.com/articles/2018/more-exploits-the-great-plc-hack/, 2018.

[6] S. A. Milinković and L. R. Lazić, "Industrial plc security issues," in *2012 20th Telecommunications Forum (TELFOR)*. IEEE, 2012, pp. 1536–1539.

[7] M. Zalewski, "American fuzzy lop," 2014.

[8] M. Sutton, A. Greene, and P. Amini, *Fuzzing: brute force vulnerability discovery*. Pearson Education, 2007.

[9] M. Tenekedzhiev, "Fuzzing the s7 network protocol methodology for security evaluation of industrial control systems through fuzz testing programmable logical controller operating with the siemens s7 network protocol," Master's thesis, 2017.

[10] F. Tacliad, T. D. Nguyen, and M. Gondree, "Dos exploitation of allen-bradley's legacy protocol through fuzz testing," in *Proceedings of the 3rd Annual Industrial Control System Security Workshop*, 2017, pp. 24–31.

[11] S. Zonouz, J. Rrushi, and S. McLaughlin, "Detecting industrial control malware using automated plc code analytics," *IEEE Security & Privacy*, vol. 12, no. 6, pp. 40–47, 2014.

[12] G. Sandaruwan, P. Ranaweera, and V. A. Oleshchuk, "Plc security and critical infrastructure protection," in *2013 IEEE 8th International Conference on Industrial and Information Systems*. IEEE, 2013, pp. 81–85.

[13] S. E. McLaughlin, "On dynamic malware payloads aimed at programmable logic controllers." in *HotSec*, 2011.

[14] D. Tychalas and M. Maniatakos, "Iffset: In-field fuzzing of industrial control systems using system emulation," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018.

[15] G. Palavicini Jr, J. Bryan, E. Sheets, M. Kline, and J. San Miguel, "Towards firmware analysis of industrial internet of things (iiot)," *Special Session on Innovative CyberSecurity and Privacy for Internet of Things: Strategies, Technologies, and Implementations*, 2017.

[16] L. McMinn and J. Butts, "A firmware verification tool for programmable logic controllers," in *International Conference on Critical Infrastructure Protection*. Springer, 2012, pp. 59–69.

[17] M. Krotofil and J. Larsen, "Rocking the pocket book: Hacking chemical plants," in *DefCon Conference, DEFCON*, 2015.

[18] A. P. Mathur and N. O. Tippenhauer, "SWaT: a water treatment testbed for research and training on ICS security," in *International Workshop on Cyber-physical Systems for Smart Water Networks*, 2016, pp. 31–36.

[19] P. H. N. Rajput, P. Rajput, M. Sazos, and M. Maniatakos, "Process-aware cyberattacks for thermal desalination plants," in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, 2019, pp. 441–452.

[20] S. D. Bay, D. Kibler, M. J. Pazzani, and P. Smyth, "The uci kdd archive of large data sets for data mining research and experimentation," *SIGKDD Explor. Newsl.*, vol. 2, no. 2, p. 81–85, Dec. 2000. [Online]. Available: https://doi.org/10.1145/380995.381030

[21] Y. Zhang, L. Wang, W. Sun, R. C. Green II, and M. Alam, "Distributed intrusion detection system in a multi-layer network architecture of smart grids," *IEEE Transactions on Smart Grid*, vol. 2, no. 4, pp. 796–808, 2011.

[22] T. Morris and W. Gao, "Industrial control system traffic data sets for intrusion detection research," in *International Conference on Critical Infrastructure Protection*. Springer, 2014, pp. 65–78.

[23] J. Goh, S. Adepu, K. N. Junejo, and A. Mathur, "A dataset to support research in the design of secure water treatment systems," in *Critical Information Infrastructures Security*, G. Havarneanu, R. Setola, H. Nassopoulos, and S. Wolthusen, Eds. Cham: Springer International Publishing, 2017, pp. 88–99.

[24] A. Keliris, H. Salehghaffari, B. Cairl, P. Krishnamurthy, M. Maniatakos, and F. Khorrami, "Machine learning-based defense against process-aware attacks on industrial control systems," in *2016 IEEE International Test Conference (ITC)*, 2016.

[25] R. C. B. Hink, J. M. Beaver, M. A. Buckner, T. Morris, U. Adhikari, and S. Pan, "Machine learning for power system disturbance and cyber-attack discrimination," *2014 7th International Symposium on Resilient Control Systems (ISRCS)*, pp. 1–8, 2014.

[26] T. H. Morris, Z. Thornton, and I. Turnipseed, "Industrial control system simulation and data logging for intrusion detection system research," *7th Annual Southeastern Cyber Security Summit*, 2015.

[27] S. A. Zonouz, H. Khurana, W. H. Sanders, and T. M. Yardley, "Rre: A game-theoretic intrusion response and recovery engine," in *2009 IEEE/IFIP International Conference on Dependable Systems Networks*, June 2009, pp. 439–448.

Hardware/Software Co-Exploration of Neural Architecture Search for Resource-Constrained Cyber-Physical Systems

Weiwen Jiang, University of Notre Dame

Abstract

As the edge devices become more powerful, it brings new opportunities to bring the machine learning (ML) to the Cyber-Physical Systems (CPS). In ML algorithms, the neural architecture is the key component, and therefore, a fundamental problem is: what kind of neural architecture fits CPS? Recently, several Neural Architecture Search (NAS) frameworks have been developed to automatically search neural architectures with the highest accuracy. However, most of them aim at accuracy only and do not take into consideration the hardware that will be used to implement the architecture. This will potentially lead to excessive latencies beyond specifications, rendering the resulting architectures useless. To solve this problem, this work brings the hardware/software co-design philosophy into the search loop, and we simultaneously explore the neural architecture space and CPS system design space. As such, the best neural architecture and CPS system design pair can be identified.

1 Introduction

The neural architecture search (NAS) has achieved great success to liberate human labor in the design of neural architectures for various tasks including image classification and language modeling [1]. Although the research on the automatic prediction of neural network architectures can trace back to the 1980s [2], after deep neural networks have achieved great success in AI domains, there have been growing interests in generating good neural architectures for the interested dataset recently. With the fact that the architectures are growing deeper, the search space expands exponentially, leading to more difficulties in exploring the search space. In existing work, there are two mainstreams of architecture search: (1) employing reinforcement learning [1, 3, 4], (2) applying evolutionary algorithms [5, 6, 7]. The basic idea is to iteratively update hyperparameters to generate better "child networks" in terms of accuracy.

Most of the existing NAS frameworks explore the architecture search space only, without considering the hardware design freedom available in many applications. In addition to neural architecture design, those hardware platforms should also be optimized, otherwise the system will finally yield to the sub-optimal solutions.

Interestingly, the hardware design space is tightly coupled with the architecture search space, i.e., the best neural architecture depends on the hardware (hardware-aware NAS), and the best hardware depends on the neural architecture. It is therefore best to jointly explore both spaces to push forward the Pareto frontier between hardware efficiency and test accuracy for better design tradeoffs. Most recently, targeting the power edge devices, such as FPGAs and ASICs, the hardware-aware NAS [8, 9, 10, 11, 12, 13, 14] has been proposed to take into consideration the hardware efficiency in addition to accuracy; while they target on more powerful devices like Field Programmable Gate Array; which cannot be directly applied for CPS systems, where ultra low power is required.

2 HW/SW Co-Exploration Framework

This section will present the proposed framework. We will use the NAS discussed in [1] as the backbone framework and CPS systems as the hardware platform to demonstrate our concept. It can be integrated with any existing NAS techniques [1, 15, 16, 17].

Figure 1 shows the HW/SW co-exploration framework. The framework contains a RNN based controller and two levels of explorations. Unlike that in [1], the controller has multiple RNN cells instead of one. More specifically, each layer in a child network has a corresponding RNN cell. During the exploration, cells will be reorganized to support different optimization goals.

Fast Exploration. In the first level, namely Fast Exploration (FE), the objective is to minimize the overall power consumption FE takes three types of inputs: (1) a set of available sensors, (2) hyperparameters of a child network, (3) a power consumption requirement. It will generate a new child network, whose power consumption at inference phase can meet power constraint.

we apply reinforcement learning to update the parameters in those N RNNs, and use these RNNs to predict the hyperparameters of child networks. In each iteration, we will predict T child networks, which can be viewed as a list

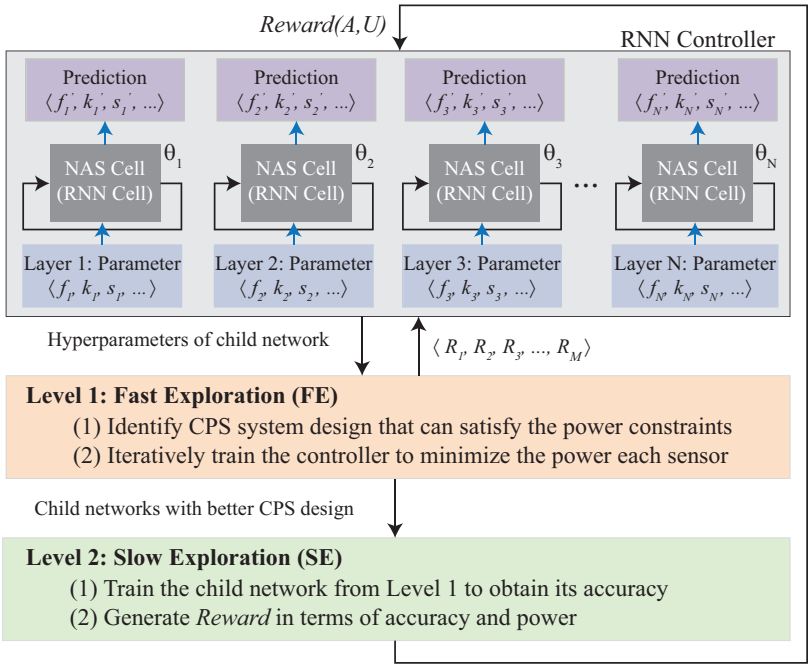


Figure 1: An overview of HW/SW co-exploration framework: the fast exploration level prunes child networks with inferior power consumption; the slow exploration level updates controller using power consumption and accuracy obtained by training child networks.

of actions $a_{1:T}$. Correspondingly, notation $a^i_{1:T}$ represents the hyperparameters of the i^{th} pipeline stage in these child networks. For each child network predicted by the controller, we can obtain the utilization of the i^{th} pipeline stage (corresponding to one FPGA) using BLAST, denoted as U_i. Then, for RNN i, we utilize U_i to generate a reward R_i to update its parameters θ_i. The reward R_i can be calculated using the following formula.

$$R_i = \begin{cases} P_i & P_i \leq 1 \\ 1 - P_i & 1 < P_i \leq 2 \\ -1 & P_i > 2 \end{cases} \tag{1}$$

where $P_i > 1$ indicates that the required throughput cannot be satisfied, and we give the We will train the child network on the held-out validate set, and therefore the exploration speed is much slower than that of the first one.negative reward. For each RNN, our objective is to maximize the expected reward for actions from time 1 to T, represented by $J(\theta_i) = E_{Pr(a^i_{1:T};\theta_i)}[R_i]$. Since the reward is non-differentiable, we apply the policy of gradient method to update θ_i. Specifically, the method of REINFORCE rule [18] has been employed as in [1, 15].

Slow Exploration. After obtaining a child network meeting the power specification through the fast exploration level, we now move to the second level. In this level, we aim to update the controller RNN to generate new child networks with higher accuracy and power efficiency. We call it Slow Exploration (SE).

SE takes the generated child network, the CPS system design from FE as the inputs. The child network is first trained to obtain accuracy A. Then, the average power efficiency P of the child network under the partition and assignment will be calculated. Finally, we compute the reward to update the controller using the following formula.

$$Reward(A,P) = \beta \times A + (1-\beta) \times P \tag{2}$$

where β is an adjustment parameter, which reflects the bias on test accuracy and power consumption. The value of β ranges from 0 to 1. After that, we update the controller using the reward by applying the policy gradient reinforcement learning, which is the same as that in FE level.

3 Conclusion

While deep neural network has demonstrated great potential to be implemented in CPS systems, the restricted power consumption requirement in still hinders the applications of neural networks in CPS systems. To overcome this problem, we proposed the co-exploration framework to open up the CPS design freedom in neural architecture search. This is driven by the trend that the CPS platform can be fully customized for restricted power constraints. This work presents a fast-slow tow-step hardware/software co-exploration framework. In the fast exploration phase, the controller will predict neural architectures for hardware optimization without the consideration of accuracy; while the accuracy is optimized in the slow exploration. Such a co-exploration framework can guarantee that the identified neural network on the CPS system can satisfy the design specifications (e.g., power consumption), meanwhile, maximizing the architecture accuracy.

References

[1] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *Proc. of ICLR*, 2017.

[2] J David Schaffer et al. Combinations of genetic algorithms and neural networks: A survey of the state of the art. In *Proc. of International Workshop on COGANN*, pages 1–37. IEEE, 1992.

[3] Barret Zoph et al. Learning transferable architectures for scalable image recognition. In *Proc. of CVPR*, pages 8697–8710, 2018.

[4] Bowen Baker et al. Designing neural network architectures using reinforcement learning. *arXiv:1611.02167*, 2016.

[5] Esteban Real et al. Large-scale evolution of image classifiers. *arXiv:1703.01041*, 2017.

[6] Lingxi Xie and Alan Yuille. Genetic cnn. In *Proc. of ICCV*, pages 1388–1397. IEEE, 2017.

[7] Ye-Hoon Kim et al. Nemo: Neuro-evolution with multiobjective optimization of deep neural network for speed and accuracy. In *Proc. of ICML 2017 AutoML Workshop*, 2017.

[8] Weiwen Jiang et al. Accuracy vs. efficiency: Achieving both through fpga-implementation aware neural architecture search. In *Proc. of DAC*, Las Vegas, USA, 2019. ACM.

[9] Qing Lu et al. On neural architecture search for resource-constrained hardware platforms. *arXiv preprint arXiv:1911.00105*, 2019.

[10] Xinyi Zhang et al. When neural architecture search meets hardware implementation: from hardware awareness to co-design. In *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 25–30. IEEE, 2019.

[11] Song Bian et al. Nass: Optimizing secure inference via neural architecture search. *arXiv preprint arXiv:2001.11854*, 2020.

[12] Lei Yang, Zheyu Yan, Meng Li, Hyoukjun Kwon, Liangzhen Lai, Tushar Krishna, Vikas Chandra, Weiwen Jiang, and Yiyu Shi. Co-exploration of neural architectures and heterogeneous asic accelerator designs targeting multiple tasks. In *ACM/EDAC/IEEE Design Automation Conference (DAC)*, July 2020.

[13] Weiwen Jiang et al. Hardware/software co-exploration of neural architectures. *arXiv:1907.04650*, 2019.

[14] Lei Yang, Weiwen Jiang, Weichen Liu, Edwin Sha, Yiyu Shi, and Jingtong Hu. Co-exploring neural architecture and network-on-chip design for real-time artificial intelligence. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2020.

[15] Han Cai et al. Proxylessnas: Direct neural architecture search on target task and hardware. In *Proc. of ICLR*, 2019.

[16] Hanxiao Liu et al. Darts: Differentiable architecture search. In *Proc. of ICLR*, 2019.

[17] Gabriel Bender et al. Understanding and simplifying one-shot architecture search. In *Proc. of ICML*, pages 549–558, 2018.

[18] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Power-Driven DNN Dataflow Optimization on FPGA

Qi Sun[1], Tinghuan Chen[1], Jin Miao[2], Bei Yu[1]

[1]CSE Department, The Chinese University of Hong Kong
[2]Cadence Design Systems

1 Introduction

Deep Neural Networks (DNNs) have been proven to achieve unprecedented success on a wide range of AI relevant applications, such as image classification, object detection, and design for manufacturing [1, 2, 3]. FPGA has been chosen as a promising hardware platform to deploy DNN model due to the reconfigurability [6, 7]. When deploying a DNN application onto an FPGA board, in an end-to-end automation flow, two pivotal steps are involved: architectural design and dataflow optimization. For architectural design, a typical DNN accelerator architecture is shown in Figure 1(a). The processing unit (PU) is responsible for the computations, which is composed of several processing engine (PE) arrays. A multiplier-accumulator (MAC) can be implemented by various processing engines (PEs), such as arithmetic multiplication array or systolic array [8, 9, 10]. For dataflow optimization, an analytical model, per the given architecture, is often developed to capture and measure various design configurations and their attainable performances by metrics like latency, power consumption, data transfer size, and on-chip resource consumption, etc. [11, 10, 12, 13, 14, 15, 8]. The optimal design configuration is iterated and achieved through above explorations [13, 14, 10, 8].

For fast prototyping, the deployment of DNN onto FPGA is facilitated by high-level synthesis (HLS) tools which emancipate designers from the complicated hardware description languages (like Verilog or VHDL) crafting. In addition, HLS makes it possible for efficient DNN design modeling and configurations [16], unlocking rapid design space explorations for high-performance designs [17].

Previous literatures on FPGA based DNN accelerators have primarily focused solely on the latency reduction [10, 12, 13, 15]. In latency-driven designs, hardware resources are activated without restrictions for best performances. Apparently, such latency-driven designs often end up with sub-optimality in power consumption and are excluded from applications demanding low-power.

In the traditional ASIC or FPGA hardware design domain, power-driven design methodologies have been well studied. Chen *et al.* proposed a low-power HLS methodology for general application FPGA designs without significant loss of performance [18]. Besides, optimization on linear algebra core was presented to achieve high-performance while reducing power consumption [19]. Nevertheless, the power efficiency explorations and optimizations on DNN deployed onto FPGA have yet been sufficiently studied.

In this article, we propose a framework to optimize the power efficiency of DNN dataflow on FPGA while minimizing the impact of latency. We first propose power and latency models that are built upon different dataflow configurations. Then a power-driven dataflow formulation is proposed, which enables a hierarchical exploration strategy on the dataflow configurations, leading to globally optimal power efficiency with insignificant latency loss.

We also deploy two DNN models on FPGA to verify the effectiveness of our proposed framework. In this architecture, neighboring layers are fused to reduce the volume of data transferred between on-chip buffer and off-chip memory. Systolic array is adopted as the computation core to balance data access and computation. Overall experimental results have demonstrated the power improvement of up to 31% with latency degradation of no worse than 6.5%.

The rest of this article is organized as follows. Section 2 provides preliminaries including concrete analysis from DNN dataflow to systolic array and fused layer. Section 3 introduces our power formulation. Section 4 presents a dataflow configuration estimation and exploration process using our proposed power-driven formulation. Section 5 demonstrates the experiments and results. The final conclusion and further discussions in Section 6.

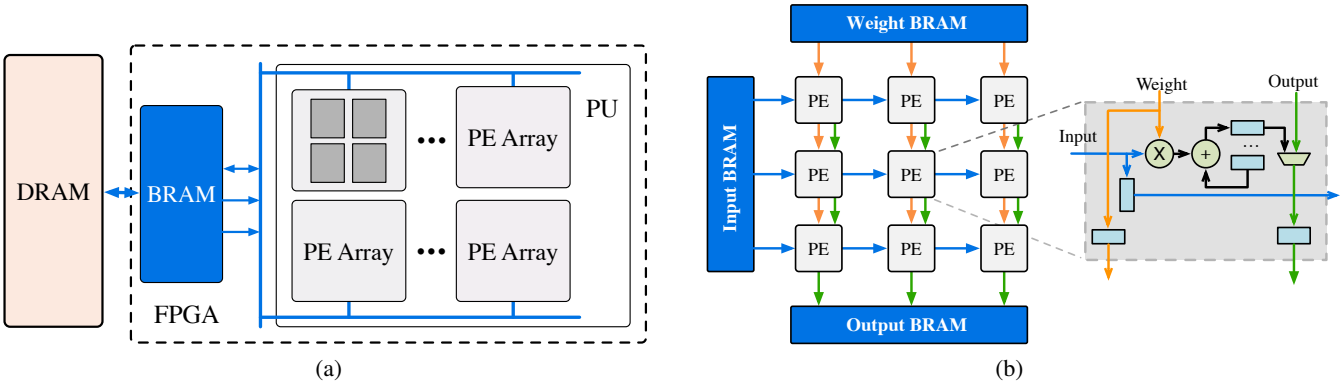


Figure 1: (a) DNN FPGA accelerator architecture; (b) Systolic array architecture and processing engine structure. Data are fed into array from the top and the left.

2 Preliminaries

A typical DNN model contains several operational layers, including convolution, pooling and ReLU, where convolutional layers dominate both storage and computation in current popular DNN structures [1, 20]. Many efforts have been made to help solve this problem in the model level [4, 5]. In the hardware level, the limited on-chip memory resources would therefore cause frequent data transfer between on-chip buffer and off-chip memory. In addition, pipelining and parallelism, depending on the size of on-chip buffer and the processing unit (PU) allocated by on-chip logic resources (LUTs) and DSPs, are often applied on convolutional layers. Convolutional layers therefore have been the focal of success in design and optimization of hardware accelerators.

Recall in Figure 1(a), we show a typical DNN FPGA accelerator architecture. The accelerator is composed of off-chip memory (usually by DRAM) and an FPGA accelerator chip. On the aspect of hierarchical memory, it can be decomposed into three parts: off-chip DRAM, on-chip global buffer (BRAM) and local buffer (register). Accessing unit data at different levels implies different energy consumption [21]. We load data from off-chip DRAMs to global buffers. Then the data are passed from global buffers to local buffers (in PE array) and do the computations. Afterwards, outputs are stored to DRAMs.

Systolic array is used as the processing engine array, as shown in Figure 1(b), which becomes more and more popular [10, 22, 23] in DNN FPGA and ASIC designs recently. Each PE in systolic array connects to its neighbors while the boundary PEs also connects to the global buffers (BRAM). In each cycle, the input feature and kernel weight are passed to its neighbors which are then stored in local buffers in PE simultaneously. The boundary PEs receive data passed down from the global buffers. The input feature and kernel weight are multiplied first, then passed through accumulation operation with partial sum. Note that for computations in each PE, the input features are fed from the left to right, whereas the weights are fed from the top to bottom. By contrast, results of all PEs flow from top to bottom, into global buffers. Obviously, the size of systolic array also impacts the power consumption when passing data and conducting computations.

The design of DNN Dataflow refers to the selection procedure of configurations on data storage and data access pattern on FPGA. Typical techniques include loop tiling, unrolling, data reuse and layer fusion. We detail each of them as below.

Loop tiling partitions a loop into smaller blocks. For example, Figure 2(a) is a 6-level for-loops of a convolutional layer. The outermost loop with step size 1 is decomposed into two sub-loops and the size of inner sub-loop is *OC*, as shown in Figure 2(b). The decomposed sub-loops are responsible for transferring data between off-chip memory and on-chip global buffer. The sub-loop boundary is called the loop tiling factor, which reflects the size of global buffers. Data loaded in the outer loop flow into the local buffers (in computation engines) in the inner loops.

To facilitate parallelism, we unroll the convolutional group function as shown in Figure 2(c). Originally, only a single convolution function is called for *IC* times. During computations, different data segments flow into the same PU sequentially. Once we set the unrolling factor as *IC*, the original loop will be replaced with *IC* parallel

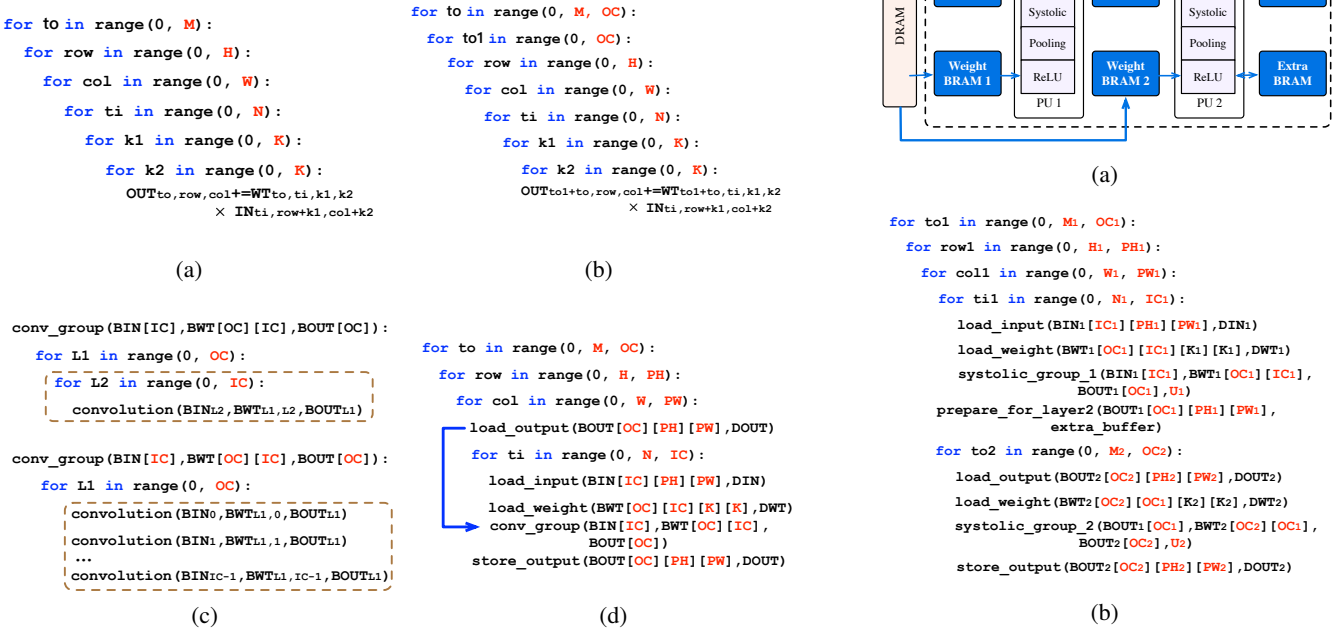


Figure 2: Pseudo-code of dataflow optimization. (a) A 6-loops convolutional layer. (b) Loop Tiling. (c) Loop Unrolling. (d) Output Data Reuse.

Figure 3: Fusing two convolutional layers: (a) hardware architecture using systolic array; (b) pseudo-codes.

convolutional functions. In this case, these data will be passed into *IC* different PUs in parallel. Loop unrolling also relies on loop tiling factors because we always unroll the sub-loops after tiling the loop. Traditional works usually unroll and parallel as many tasks as possible. This kind of choices imply challenges on power since lots of on-chip logics are instantiated and activated. In order to improve power performance, the unrolling and tiling factors need to be carefully configured.

Data reuse strategy is also beneficial. Figure 2(d) is an example of the output data reuse strategy. The output data are loaded from DRAM to BRAM once and reused in the inner loop several times. The blue line in Figure 2(d) shows the reuse situation. Similarly, if we reuse inputs, we only need to load inputs once. Different strategies may affect the total length of dataflow, hence the communication power consumption.

A naive DNN accelerator design runs layers one by one on FPGA. Due to the sequential nature, the output features of the previous layer have to be first stored in off-chip memory before starting the next layer computation. The drawback is obvious that we have to transfer the same data between FPGA and DRAM repeatedly. But for layer fusion, once we get parts of the output data of the first layer, they will be passed to compute engines of the second layer as inputs and start the computations immediately. We can fuse convolution layers with the neighboring pooling layers or ReLU layers [24, 23]. A more effective strategy is to fuse several neighboring convolution layers together with pooling and ReLU layers, which further reduces the size of data transfer [25]. Some extra data processing is necessary to handle the overlapping data resulted from layer fusion, at the cost of more memory consumption. Using layer fusion limits the choice of data reusing strategy. If we reuse input features of the second layer, only when we finish all the computations of one block in the first layer, can we start the computations of the corresponding block in the second layer. In this process, computations in the second layer are blocked, which is not helpful for the system pipelining. Typically, as shown in Figure 3(b), we reuse outputs in the first layer, which means we must reuse inputs in the second layer.

Table 1: List of Parameters

Name	Definition
OC_i	# of output channels per feature block in fused layer i
IC_i	# of input channels per feature block in fused layer i
PH_i	Feature height of feature block in fused layer i
PW_i	Feature width of feature block in fused layer i
Th_i	row # of PEs in one systolic array in fused layer i
Tw_i	column # of PEs in one systolic array in fused layer i
U_i	# of instantiated systolic arrays in fused layer i

3 Power Minimization

In this section, we discuss the details of our proposed power optimization framework. As energy is defined as the integral of power over the latency of an underline task, after introducing parameters and notations, we will first derive energy and latency models respectively. We will further show the overall power minimization formulation to be explored and optimized.

3.1 Notations

Denote the number of layers in a DNN model as Z. For a convolutional layer i, the number of input channels is N_i, number of output channels is M_i, kernel size is K_i, kernel stride is S_i, feature height is H_i, and feature width is W_i. Here we assume the output feature size is equal to the input feature size, for the expression simplicity.

Generally, we summarize the parameters in Table 1. Some have been used in Figure 2(b) and Figure 2(c). We can decompose the input convolutional channels into blocks, with IC_i channels in each block. Output channels are decomposed into blocks and each has OC_i output channels. Features in each channel are also decomposed into smaller blocks, with height PH_i and width PW_i. The corresponding block stride is PS_i, which can be determined according to the block size and kernel stride. The pseudo-code of fusing two convolutional layers using systolic array is in Figure 3(b). The output channels of the first layer are the input channels of the second layer. Therefore, we have $M_1 = N_2$ and $OC_1 = IC_2$. Also, the number of input blocks of the second layer is equal to the number of blocks in the first layer.

Data used in systolic array should be in the matrix format while they are often stored as tensors in DRAM. The process of transforming data from the tensors to matrices is conducted on board to avoid unnecessarily repeated DRAM access and extra power load. Denote the height and width of systolic array for fused layer i as Th_i and width Tw_i, the number of instantiated systolic arrays in a processing unit as U_i. For any feature tensors, they should be flattened as matrices, with height $(\frac{PH_i-K_i}{PS_i}+1)\times(\frac{PW_i-K_i}{PS_i}+1)$. For weights, they are flattened as matrices with width OC_i. The depth of the transformed matrix is D_i, as shown in Equation (3)[26].

$$D_i = K_i^2 \times IC_i. \tag{3}$$

3.2 Modeling the Energy

Without loss of generality, the total energy can be composed of two parts, data transfer and computation.

As mentioned before, there are three-levels of hierarchical memory in DNN accelerator system. We measure the data transfer energy by calculating the data size transferring between all these levels. The energy of accessing one unit data at each level is regarded as a constant. Compared with on-chip buffers, accessing data from DRAM consumes much more energy. In Figure 3(b), the *load* and *store* functions read and store data between DRAM and FPGA. As shown in Figure 3(a), both two layers need to load weights from off-chip memory to on-chip buffer. The first layer needs to load input features from DRAM. Meanwhile, we transfer output features, or partial sums of the second layer between off-chip memory and on-chip buffer several times since we reuse input features here. Other

sources of energy consumptions include extra buffers used to prepare data and handle data overlaps for the second fused layer, as well as passing data between global buffer and systolic array, etc.

For simplicity, we group the data transfer energy in systolic array as part of the computation energy due to the computations within systolic array. The data transfer energy of layer i can be simplified as in Equation (4). Here $\{\alpha_1, \alpha_2, \ldots, \alpha_7\}$ are model-specific constants that can be predetermined once the DNN model is given. For example, if reusing outputs, α_1 is the multiplication of feature size and channel number, and $(\alpha_1 \ / \ OC_i)$ refers to the input data size we load. Similarly, $(\alpha_7 \ / \ IC_i)$ refers to how many output data we load, if we reuse inputs. $(\alpha_2 \ / \ PH_i \times PW_i)$ stands for weights energy. In addition, we have four terms $(\alpha_5 \ / \ PH_i)$, $(\alpha_6 \ / \ PW_i)$, $\alpha_3 PH_i$ and $\alpha_4 PW_i$ for fused data preparations.

$$ED_i = \frac{\alpha_1}{OC_i} + \frac{\alpha_2}{PH_i \times PW_i} + \alpha_3 PH_i + \alpha_4 PW_i + \frac{\alpha_5}{PH_i} + \frac{\alpha_6}{PW_i} + \frac{\alpha_7}{IC_i}. \tag{4}$$

As to the computation energy, naively, we may assume the total computation energy is a constant. This is because from the perspective of DNN model description, the total number of computations (accumulation and multiplication) are definite. However, for hardware deployments, the computation energy is tightly proportional to the sizes of computation engines. As mentioned earlier, the size of computation engine (or systolic array) is roughly a product of the height of input feature block times the width of weight block. Input feature block with size $(IC_i \times PH_i \times PW_i)$ is decomposed into $(\lceil (\frac{PH_i - K_i}{PS_i} + 1) \times (\frac{PW_i - K_i}{PS_i} + 1) \ / \ Th_i \rceil)$ sub-matrices. Each has size $(D_i \times Th_i)$, where D_i is the depth as shown in Equation (3). Similarly, for weights with size $(OC_i \times K_i^2 \times IC_i)$, they are decomposed into $(\lceil OC_i \ / \ Tw_i \rceil)$ sub-matrices. Each has size $(D_i \times Tw_i)$. The ceiling operations here are to tackle with the boundary situations.

In each computation cycle, all the $(Th_i \times Tw_i)$ PEs are in active status and consume energy. For each pair of inputs and weights sub-matrices, $(D_i + Th_i + Tw_i - 2)$ cycles are needed to finish the computations. Energy consumed to finish the computation of one feature block and one corresponding weight block using systolic array is denoted as eb, in Equation (5), where ec is the energy consumed by each PE in each cycle, including inside data transfer energy.

$$eb = \left\lceil \frac{(\frac{PH_i - K_i}{PS_i} + 1) \times (\frac{PW_i - K_i}{PS_i} + 1)}{Th_i} \right\rceil \times \left\lceil \frac{OC_i}{Tw_i} \right\rceil \times (Th_i \times Tw_i) \times (D_i + Th_i + Tw_i - 2) \times ec. \tag{5}$$

The total energy consumed by layer i is the summation of all blocks, as in Equation (6).

$$EC_i = \lceil N_i / IC_i \rceil \times \lceil H_i / PH_i \rceil \times \lceil W_i / PW_i \rceil \times \lceil M_i / OC_i \rceil \times eb. \tag{6}$$

Here we assume that the energy consumption have nothing to do with the number of systolic arrays (parallelism), *i.e.* U_i. The reason is that no matter how many systolic arrays we have instantiated, we only call it $(\lceil (\frac{PH_i - K_i}{PS_i} + 1) \times (\frac{PW_i - K_i}{PS_i} + 1) \ / \ Th_i \rceil \times \lceil OC_i \ / \ Tw_i \rceil)$ times in computations. By contrast, the size of each individual systolic array is proportional to energy consumption.

The overall model energy consumption E_{total} is the summation of computation and data transfer energy of all layers, as shown in Equation (7).

$$E_{total} = \sum_{i=1}^{Z} (ED_i + EC_i). \tag{7}$$

3.3 Modeling the Latency

Intuitively, the latency consists of computation latency and data transfer latency. The characteristic of using systolic array is that, in each PE, we pass data to neighboring PEs per cycle and finish the computation simultaneously. Therefore, part of data transfer latency has overlapped with the computation latency. Total data transfer latency is determined by the size of data we transfer. Although some transfer overlaps could occur per engineering implementations, we can still sum up all the data to be transferred to estimate the latency. The on-chip data transfer latency should be divided by the degree of system parallelism since the data are passed into the unrolled computation engines simultaneously.

$$\begin{aligned} \min \ & \frac{E_{total}}{L_{total}}, \\ \text{s.t.} \ & Buffer_{used} \leq Buffer_{total}, \\ & DSP_{used} \leq DSP_{total}, \\ & L_{total} \leq L_{upper}. \end{aligned} \tag{12}$$

Except for the latency regarding to systolic arrays, the data transfer latency in layer i can be formulated as in Equation (8), where $\{\beta_1, \beta_2, \ldots, \beta_7\}$ are model-specific constants. The explanations to these terms are similar to Equation (4). For example, $(\beta_1 \ / \ OC_i)$ is for input data transfer latency.

Here we highlight the latency of passing data into systolic arrays. For input feature block with size $(IC_i \times PH_i \times PW_i)$, and weight block with size $(OC_i \times IC_i \times K_i^2)$, we have computation latency in Equation (9).

$$LD_i = \frac{\beta_1}{OC_i} + \frac{\beta_2}{PH_i \times PW_i} + \beta_3 PH_i + \beta_4 PW_i + \frac{\beta_5}{PH_i} + \frac{\beta_6}{PW_i} + \frac{\beta_7}{IC_i}. \tag{8}$$

$$lc = \left\lceil \frac{(\frac{PH_i - K_i}{PS_i} + 1) \times (\frac{PW_i - K_i}{PS_i} + 1)}{Th_i} \right\rceil \times \left\lceil \frac{OC_i}{Tw_i} \right\rceil \times (D_i + Th_i + Tw_i - 2), \tag{9}$$

where D_i is the depth of transformed matrix as in Equation (3). The total latency consumed by layer i is the summation of all blocks, as in Equation (10).

$$LC_i = \lceil \lceil N_i \ / \ IC_i \rceil \times \lceil H_i \ / \ PH_i \rceil \ / \ U_i \rceil \times \lceil W_i \ / \ PW_i \rceil \times \lceil M_i \ / \ OC_i \rceil \times lc. \tag{10}$$

Since we have U_i instantiated parallel systolic arrays in total for fused layer i, the overall latency should be divided by U_i. The ceiling operations here are to tackle with the boundary situations, too.

The total model latency L_{total} can be formulated as Equation (11), which is summation of latencies of all layers.

$$L_{total} = \sum_{i=1}^{Z} (LD_i + LC_i). \tag{11}$$

3.4 Power Minimization Formulation

The overall optimization objective is to minimize the ratio of energy consumption over system latency, constrained by resources, as shown in Formula (12). This formulation is also constrained by latency upper bound L_{upper} since we cannot allow latency to be infinitely large. $Buffer_{total}$ and DSP_{total} represent the total buffers and DSPs available on FPGA. $Buffer_{used}$ and DSP_{used} denote the memory and computation resources used.

Equation (11) and Equation (7) are both the summation of all layers. However, for a given DNN model, the resources used on chip only need to handle one fused layer group, i.e., we run the fused groups one by one sequentially.

Let $Buffer^i_{global}$ represent the size of global buffers used by fused layer i, which includes the following terms. Denote the unit size of buffers used to store these data as C_{GLB}, which is determined by data precision. The size of input features buffers needed by one block in layer i is $IC_i \times PH_i \times PW_i \times C_{GLB}$. Similarly, the sizes of output features and weights buffers are $OC_i \times PH_{i+1} \times PW_{i+1} \times C_{GLB}$ and $OC_i \times IC_i \times K_i^2 \times C_{GLB}$ respectively. For the second fused layer, as shown in Figure 3(a), extra buffers are needed to store the overlapping features. The horizontal overlapping data need buffers with size $IC_{i+1} \times W_{i+1} \times PS_{i+1} \times C_{GLB}$, and the vertical data have size $IC_{i+1} \times PH_{i+1} \times PS_{i+1} \times C_{GLB}$.

The local buffers $Buffer^i_{local}$ of fused layer i are needed in each PE to store the intermediate data. Therefore, local buffer size is equal to $U_i \times Th_i \times Tw_i \times C_{PE}$, where C_{PE} is a constant representing size of local buffers per PE.

The number of DSPs used in each PE in systolic array can be assumed as a constant, denoted as C_{DSP}, which is also determined by data precision. The total number of DSPs used by one layer is equal to $U_i \times Th_i \times Tw_i \times C_{DSP}$.

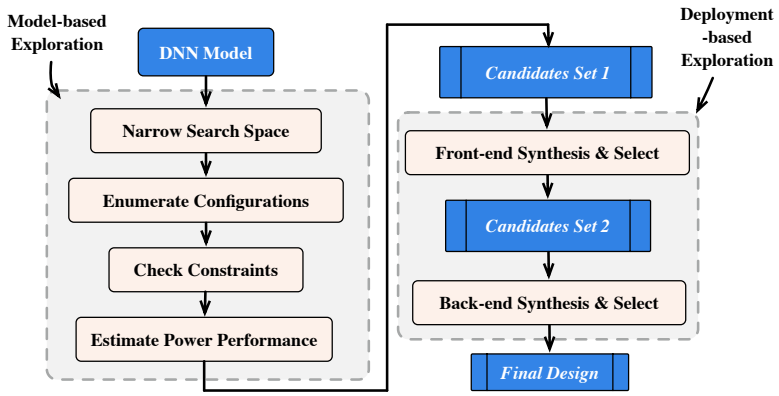


Figure 4: The proposed exploration flow, where the input is DNN model, the output is final dataflow configuration.

For a fused layer group with L layers, we have $Buffer_{used}$ and DSP_{used} in Equations (13) and (14).

$$Buffer_{used} = \sum_{i=1}^{L} \left(Buffer^{i}_{local} + Buffer^{i}_{global} \right), \tag{13}$$

$$DSP_{used} = \sum_{i=1}^{L} \left(U_i \times Th_i \times Tw_i \times C_{DSP} \right). \tag{14}$$

We use the proposed model as the metric of measuring power performance of dataflow configurations. Any configurations violating the constraints will be excluded.

4 Dataflow Exploration

In this section, we discuss how to explore power optimal dataflow configurations based on our proposed formulation. The Formula (12) is non-convex and non-linear. Besides, all the parameters should be integers according to their hardware meanings. It is nearly impossible to enumerate all configurations in the search space and run synthesis for them all. We therefore propose hierarchical exploration which can be decomposed into two steps: **model-based exploration** and **deployment-based exploration**. The overall solution exploration flow is in Figure 4.

In **model-based exploration**, we estimate the power performance of dataflow configurations using our power model. For a given DNN model, firstly, we narrow the search space by adding more practical and empirical constraints. This step is reasonable since some specific domain knowledge cannot be expressed explicitly in the objective function. For example, the on-chip buffer size is usually the power of two. Therefore, the search space for $\{OC_i, PH_i, PW_i, IC_i\}$ is narrowed at a large scale. Further, to help solve the latency constraints in Equation (12), we can constrain the parameters in $\{U_i, Th_i, Tw_i\}$ to be greater than an acceptable lower bound. Layer fusion strategy is highly related to the DNN model structures. The fused layers should share similar structures due to hardware regularity especially in FPGA. For example, for VGG16, we split the model into several groups at pooling layers. Then all layers in one group are fused, including convolutional layers, ReLU layers, and pooling layers. For AlexNet, convolutional layers sharing 3×3 kernels are fused. For layers with 11×11 and 5×5 kernels, they are not fused since the different kernel sizes may lead to unstructured hardware designs which consume lots of logic resources to do the logical controls. For the layers which are not fused, we can follow the empirical ideas as proposed in [27]. After narrowing the exploration space with such, all possible legal configurations can be enumerated. Then we check the constraints and discard the configurations violating any constraint. A set of candidates with lower power values out of the remaining configurations will survive.

In **deployment-based exploration**, we further verify the configurations selected in previous step, removing those incompatible to FPGA hardware platform. Thus an even smaller set of candidates are generated after front-end synthesis. Eventually, those candidates passing all above pruning procedures are fed into the next step to run the

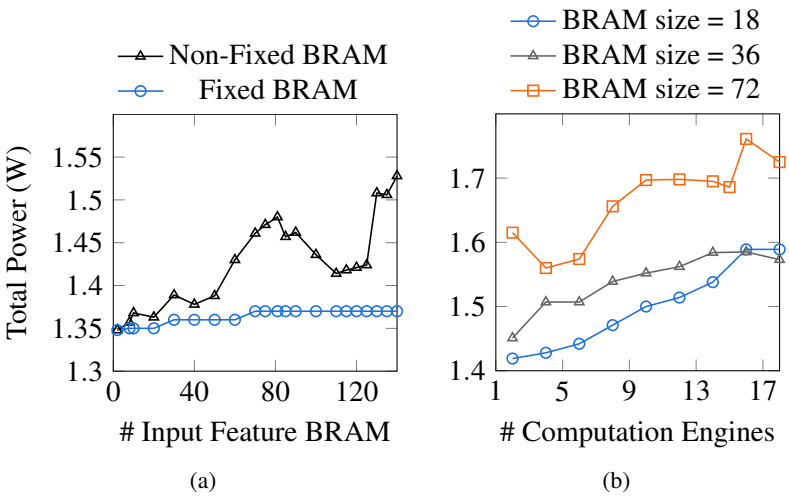


Figure 5: (a) If the number of BRAMs is not fixed, as the number increases, more power is consumed. By fixing the BRAM but read in same amounts of data, the power consumption is stable; (b) An example of deploying 2×2 convolutions on PYNQ-Z1. As the number of computation engines increases, the power increases. With more BRAMs used, the power performance also decreases.

back-end synthesis to get the actual power-performance metrics. The final optimal configurations is the one with the best back-end power performance.

5 Experimental Results

All experiments are conducted on Xilinx Zynq UltraScale+ MPSoC ZCU102 or PYNQ Z1 [28]. The design and deployment flow are via Xilinx Vivado HLS 2018.2. Some fundamental convolution examples, together with AlexNet and VGG16 are evaluated. The L_{upper} is set as 108% of the baseline latency for AlexNet and VGG16.

Firstly, the influences of instantiating different numbers of BRAMs (global buffers) and computation engines are shown in Figure 5. As in Figure 5(a), we only load and store data between DRAM and FPGA, with different buffer sizes and data sizes. The x-axis represents the number of BRAMs. If we enlarge the BRAM capacity, *i.e.* instantiate more BRAMs, the system power will increase significantly. If the BRAM size is fixed and we load in the same data with the non-fixed version, data transfer wouldn't have negative impacts on power performance. Another example in Figure 5(b) shows the situation of finishing a 2×2 convolution task at various parallel levels. The convolution task is decomposed into several smaller parallel sub-tasks. One parallel sub-task is assigned with one computation engine. The difficulties of partitioning the origin problem into sub-tasks lead to some fluctuations. It is remarkable that the power consumption trend increases as the number of computation engines increases. Considering the situations of using same number of computation engines with different BRAM sizes, obviously, more BRAMs consume more power. Inspired by this experiment, it is reasonable to instantiate less BRAMs to save power without loss of parallelism.

Secondly, two well known DNN models, AlexNet [1] and VGG16 [20] are applied to evaluate the power fidelity of our design flow. We use the 16-bit fixed-point for feature maps and 8-bit fixed-point for weights. A set of initial designs are selected randomly to show the model fidelity, shown in Figure 6(a) and Figure 6(b). The x-axis represents the index. To show the order fidelity, all the results are normalized to 1. The power performance estimated by our model have similar orders compared with the back-end synthesis power results. This experiment has shown a clear proof on high fidelity of our power model.

Crucially, a reasonable trade-off between power and latency is also often desired, *i.e.* improvements on power should come with only minimal or acceptable latency degradation. The initial designs in Figure 6(a) and Figure 6(b) are used here as the baselines. Some parameters of these designs are fixed and others are free to vary in the search

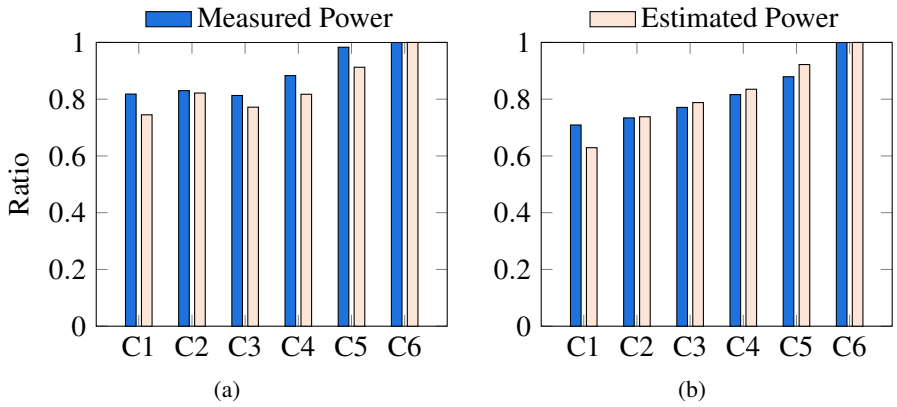


Figure 6: Model fidelity analysis: (a) AlexNet; (b) VGG16.

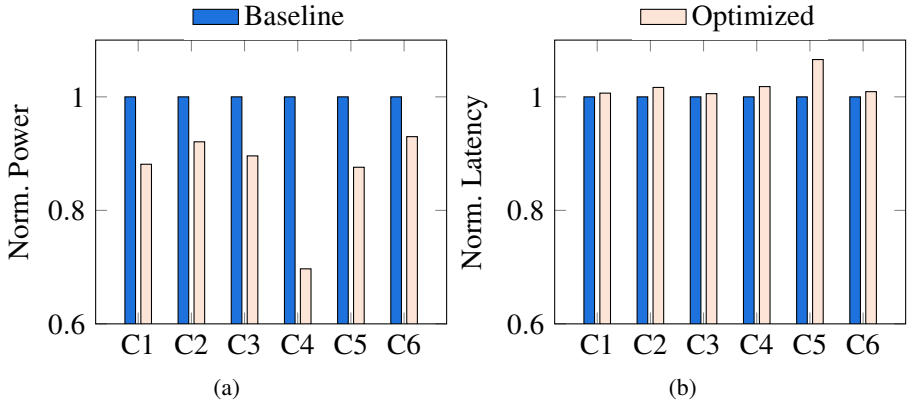


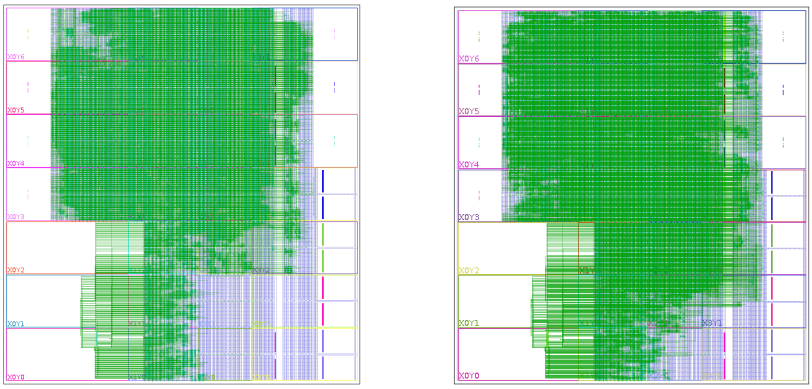
Figure 7: The results of a set of configurations on: (a) power; (b) latency.

Table 2: Power Performance and Utilization

DNN Model	Power (W)	DSP	BRAM	FF	LUT
AlexNet	4.042	1195	192	82255	126326
VGG16	5.333	1179	618	61034	134706

space, to get the corresponding model-optimized configurations. As shown in Figure 7(a) and Figure 7(b), the baseline designs are represented as 1 and the optimized configurations are represented with ratios to the baselines. Most designs can achieve more than 10% power benefits with the best even up to 31%. As to the latency loss, most are less than 2% while only one is about 6.5%. It therefore, with the help of the proposed model, becomes intriguing and effective to trade small latency for much better power efficiency.

Finally, we show the placement and routing results of VGG16 and AlexNet in Figure 7 on ZCU102 to further consolidate our designs. As shown in Figure 8, not all hardware resources are used in our optimized design and routed wires are not very congested. As a result, power consumption can be reduced. The corresponding performance and utilization reports are in Table 2. We consume similar number of DSPs for computations in two models. The model structure of VGG16 is more friendly to layer fusion, since the model can be divided directly at pooling layers. Each group has two or three convolutional layers and more BRAMs are consumed here. Compared with VGG16, the model structure of AlexNet is more complex therefore the layer fusion is limited. That explains why less BRAMs are consumed for AlexNet.



(a) (b)

Figure 8: The schematics of our designs: (a) AlexNet; (b) VGG16.

6 Conclusion and Discussions

In this article, we have proposed a power-driven dataflow optimization flow, which can estimate the dataflow configuration power and guide the design efficiently. The results show that better power performance can be obtained at a low cost of latency. There are still several critical challenges existing in dataflow optimization of DNN accelerator, and we expand a little bit in the rest of the section.

Design Space Exploration. In Figure 4, likes most existing efforts [10, 13, 15, 8], although the tremendously large design space can be effectively reduced by domain knowledge, resource constraints and empirical rules, and hence combinations of design parameters can be almost exhaustively enumerated to determine the optimal solution with the help of simulations [13, 10], the quality of design space exploration still heavily depends on the configuration performance measurement model and the time of simulation. Recently, machine-learning-based design space exploration methodologies have been proposed to achieve good design parameters of circuits without much more simulations [29, 30, 31], where a set of representative design configurations are selected for simulation so that the predicted performance curve can be accurately fitted with less simulations. The optimal design configuration can be therefore well determined by the trained model. We believe the machine-learning-based design space exploration will be a good option to fast and effectively design DNN accelerators.

Timing Closure. Some low-power DNN FPGA accelerators often result in high device utilization, which potentially imposes difficulty on timing closure [32]. even though systolic array with the local communication and regular layout have been used as the computation core for the best performance. Recently, many arts were proposed to generate better quality of result by tuning design parameters in logic synthesis and physical synthesis [33, 34, 35]. In addition, FPGA placement and routing methods were customized for different scenarios [36, 37, 38, 39]. However, performance benefits by either tuning these back-end design parameters or customizing FPGA placement and routing methods are more limited. Note that compared to RTL language, HLS is easier to revise. If revising HLS code, tuning back-end design parameters and customizing FPGA placement/routing methods are considered in a uniform framework, it would bring more benefits for timing closure.

7 Acknowledgment

This work is partially supported by Tencent Technology, Cadence Design Systems, and The Research Grants Council of Hong Kong SAR (No. CUHK24209017). We also acknowledge the fruitful discussions in experiment with Cong Hao from UIUC, Jianli Chen from Fudan University, Meng Zhang and Zhenyu Zhu from Southeast University.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Conference on Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105.

[2] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440.

[3] R. Chen, W. Zhong, H. Yang, H. Geng, X. Zeng, and B. Yu, "Faster region-based hotspot detection," in *ACM/IEEE Design Automation Conference (DAC)*, 2019, p. 146.

[4] Y. Ma, R. Chen, W. Li, F. Shang, W. Yu, M. Cho and B. Yu, "A Unified Approximation Framework for Non-Linear Deep Neural Networks," in *IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 2019.

[5] Z. Zhang, J. Li, W. Shao, Z. Peng, R. Zhang, X. Wang and P. Luo, "Differentiable Learning-to-Group Channels via Groupable Convolutional Neural Networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 3542–3551, 2019.

[6] Q. Zhang, M. Zhang, T. Chen, Z. Sun, Y. Ma, and B. Yu, "Recent advances in convolutional neural network acceleration," *Neurocomputing*, vol. 323, pp. 37–51, 2019.

[7] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[8] C. Zhang, G. Sun, Z. Fang, P. Zhou, P. Pan, and J. Cong, "Caffeine: Towards uniformed representation and acceleration for deep convolutional neural networks," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2016, pp. 1–8.

[9] X. Wei, Y. Liang, X. Li, C. H. Yu, P. Zhang, and J. Cong, "TGPA: tile-grained pipeline architecture for low latency CNN inference," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2018, pp. 1–8.

[10] X. Wei, C. H. Yu, P. Zhang, Y. Chen, Y. Wang, H. Hu, Y. Liang, and J. Cong, "Automated systolic array architecture synthesis for high throughput CNN inference on FPGAs," in *ACM/IEEE Design Automation Conference (DAC)*, 2017, pp. 29:1–29:6.

[11] L. Lu and Y. Liang, "SpWA: an efficient sparse winograd convolutional neural networks accelerator on FPGAs," in *ACM/IEEE Design Automation Conference (DAC)*, 2018, pp. 1–6.

[12] X. Zhang, J. Wang, C. Zhu, Y. Lin, J. Xiong, W.-m. Hwu, and D. Chen, "DNNBuilder: an automated tool for building high-performance DNN hardware accelerators for FPGAs," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2018, pp. 56:1–56:8.

[13] Y. Ma, Y. Cao, S. Vrudhula, and J.-s. Seo, "Optimizing loop operation and dataflow in FPGA acceleration of deep convolutional neural networks," in *ACM International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2017, pp. 45–54.

[14] Y. Ma, Y. Cao, S. Vrudhula, and J.-s. Seo, "Performance modeling for CNN inference accelerators on FPGA," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2019.

[15] M. Motamedi, P. Gysel, V. Akella, and S. Ghiasi, "Design space exploration of FPGA-based deep convolutional neural networks," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2016, pp. 575–580.

[16] J. Mu, W. Zhang, H. Liang, and S. Sinha, "A collaborative framework for FPGA-based CNN design modeling and optimization," in *IEEE International Conference on Field Programmable Logic and Applications (FPL)*, 2018, pp. 139–1397.

[17] J. Zhao, L. Feng, S. Sinha, W. Zhang, Y. Liang, and B. He, "COMBA: A comprehensive model-based analysis framework for high level synthesis of real applications," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2017, pp. 430–437.

[18] D. Chen, J. Cong, and Y. Fan, "Low-power high-level synthesis for FPGA architectures," in *IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, 2003, pp. 134–139.

[19] A. Pedram, A. Gerstlauer, and R. A. Van De Geijn, "A high-performance, low-power linear algebra core," in *IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 2011, pp. 35–42.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *International Conference on Learning Representations (ICLR)*, 2015.

[21] Y.-H. Chen, J. Emer, and V. Sze, "Using dataflow to optimize energy efficiency of deep neural network accelerators," *IEEE Micro*, vol. 37, no. 3, pp. 12–21, 2017.

[22] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *IEEE/ACM International Symposium on Computer Architecture (ISCA)*, 2017, pp. 1–12.

[23] D. J. Moss, S. Krishnan, E. Nurvitadhi, P. Ratuszniak, C. Johnson, J. Sim, A. Mishra, D. Marr, S. Subhaschandra, and P. H. Leong, "A customizable matrix multiplication framework for the Intel HARPv2 Xeon+FPGA platform: A deep learning case study," in *ACM International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2018, pp. 107–116.

[24] T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Yan, H. Shen, M. Cowan, L. Wang, Y. Hu, L. Ceze *et al.*, "TVM: An automated end-to-end optimizing compiler for deep learning," in *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2018, pp. 578–594.

[25] M. Alwani, H. Chen, M. Ferdman, and P. Milder, "Fused-layer CNN accelerators," in *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2016, pp. 22:1–22:12.

[26] Y. Ma, Y. Cao, S. Vrudhula, and J.-s. Seo, "Optimizing the convolution operation to accelerate deep neural networks on FPGA," in *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, 2018, pp. 1354–1367.

[27] J. Li, G. Yan, W. Lu, S. Jiang, S. Gong, J. Wu, and X. Li, "SmartShuttle: Optimizing off-chip memory accesses for deep learning accelerators," in *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*, 2018, pp. 343–348.

[28] "Xilinx Inc." http://www.xilinx.com.

[29] W. Lyu, F. Yang, C. Yan, D. Zhou, and X. Zeng, "Batch bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design," in *International Conference on Machine Learning (ICML)*, 2018, pp. 3312–3320.

[30] Y. Ma, S. Roy, J. Miao, J. Chen, and B. Yu, "Cross-layer optimization for high speed adders: A pareto driven machine learning approach," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2018.

[31] S. Zhang, W. Lyu, F. Yang, C. Yan, D. Zhou, X. Zeng, and X. Hu, "An efficient multi-fidelity bayesian optimization approach for analog circuit synthesis," in *ACM/IEEE Design Automation Conference (DAC)*, 2019, p. 64.

[32] "UltraFast Design Methodology Guide for the Vivado Design Suite," https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_1/ug949-vivado-design-methodology.pdf.

[33] Q. Yanghua, H. Ng, and N. Kapre, "Boosting convergence of timing closure using feature selection in a learning-driven approach," in *IEEE International Conference on Field Programmable Logic and Applications (FPL)*, 2016, pp. 1–9.

[34] E. Ustun, S. Xiang, J. Gui, C. Yu, and Z. Zhang, "Lamda: Learning-assisted multi-stage autotuning for FPGA design closure," 2019, pp. 74–77.

[35] Q. Yanghua, C. Adaikkala Raj, H. Ng, K. Teo, and N. Kapre, "Case for design-specific machine learning in timing closure of FPGA designs," in *ACM International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2016, pp. 169–172.

[36] P. Maidee, C. Ababei, and K. Bazargan, "Timing-driven partitioning-based placement for island style FPGAs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 24, no. 3, pp. 395–406, 2005.

[37] C.-W. Pui, G. Chen, W.-K. Chow, K.-C. Lam, J. Kuang, P. Tu, H. Zhang, E. F. Y. Young, and B. Yu, "RippleFPGA: A routability-driven placement for large-scale heterogeneous FPGAs," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2016, pp. 67:1–67:8.

[38] C.-W. Pui, G. Chen, Y. Ma, E. F. Young, and B. Yu, "Clock-aware ultrascale FPGA placement with machine learning routability prediction," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2017, pp. 929–936.

[39] G. Chen, C.-W. Pui, W.-K. Chow, K.-C. Lam, J. Kuang, E. F. Young, and B. Yu, "RippleFPGA: Routability-driven simultaneous packing and placement for modern FPGAs," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 37, no. 10, pp. 2022–2035, 2017.

Depth Images Inpainting Using Edge-guided GAN

Kunhua Liu, Yuting Xie and Long Chen

School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510275, China

1 Introduction

Depth images play an important role in robotic, 3D reconstruction, autonomous driving, etc. However, depth sensors such as those used in Microsoft Kinect and Intel RealSense still produce depth images with missing data. In some fields, such as those using high-dimension maps for autonomous driving (including RGB images and depth images), objects not belonging to these maps (people, cars, etc.) should be removed. After removing objects from the depth image, the corresponding areas will be blank (i.e. missing data). Therefore, to accomplish some 3D tasks, depth images with missing data should be repaired.

Depth image inpainting approaches can be divided into two groups: image-guided depth image inpainting [1, 2, 3] and single-depth image inpainting [4, 5, 6]. Image-guided depth image inpainting repairs depth images through information on color images and previous or next frames. Without this information, these approaches are useless. Meanwhile, it is challengeable for single-depth image inpainting approoches to repair images without any information from other frames. Currently, only a few studies [4, 5, 6] have tackled this issue by using and improving depth low-rank components in depth images. There still exists the weakness that current single-depth image inpainting methods only repair depth images with sparse missing data rather than small or big holes (Fig. 1).

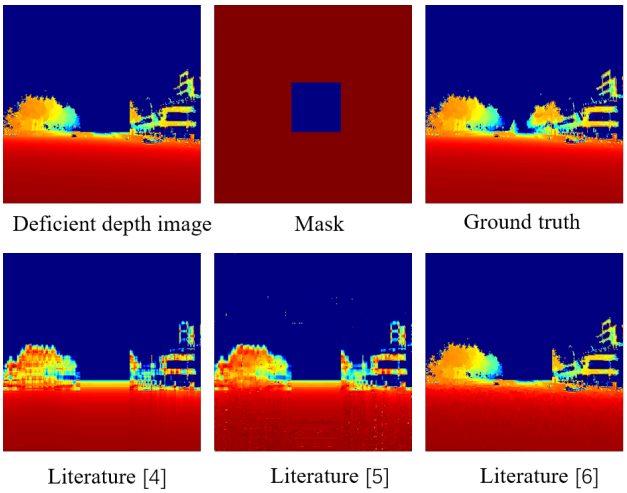


Figure 1: Results of current single-depth image inpainting approaches. We present literature results [4, 5, 6] with the ApolloScape dataset [7]. The size of the input is 256×256, and the size of the mask is 32×32.

GANs [8], as proposed by Goodfellow in 2014, have been widely studied in the field of image processing (classification [9], data augmentation [10], image-to-image translation [11, 12], high-resolution image synthesis and semantic manipulation [13], fine-grained text to image generation [14], image inpainting [15, 16], and NLP [17, 18, 19]) and have achieved state-of-the-art results. However, to our knowledge, there is still no GAN-based approach for depth image inpainting. The reasons are as belows.

On the one hand, the depth image records the distance between different objects, lacking of texture informations. Due to this characteristic, some researchers have expressed concerns about that whether convolutional neural networks (CNNs) can extract depth image features well. On the other hand, there are no public depth image datasets for CNN-based approaches to train. For the first reason, Han et al. [20] successfully employed DQN [21] and CNN to

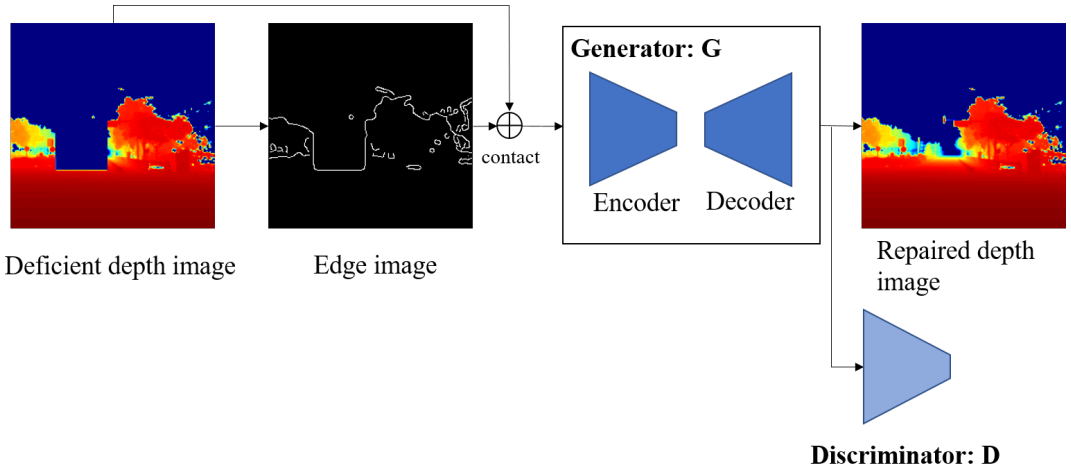


Figure 2: The pipline of our edge-guided GAN. We first obtained the edge image of the deficient depth image and then combined the results into 2 channels of data. These data are inputs to the GAN for depth image inpainting, and the output is the repaired depth image.

realize depth image inpainting, which verified that CNNs can extract features of depth images. For the second reason, the Baidu company released the ApolloScape dataset in 2018 which contains 43592 depth ground truth images. These images are sufficient to explore the GAN-based approach for depth image inpainting.

2 Edge-guided GAN

We provide a GAN-based depth image inpainting approach (edge-guided GAN) to repair deficient depth images. The architecture of our edge-based GAN is shown in Fig. 2. Edge-guided GAN is inspired by EdgeConncect[22], which is designed for color images inpainting and its architecture includes two parts: the Image-to-Edges part and the Edges-to-Image part. The Image-to-Edges part hallucinates edges of the missing region, and the Edges-to-Image part completes the edges to color image. Similar to EdgeConncect[22], edge-guided GAN also includes two parts. The first part produces a deficient depth image's edge image and combines the results into 2 channels of data. Unlike the Image-to-Edges part of EdgeConncect[22], which hallucinates edges of the missing region, the edge image presents the edge information of deficient depth image which guides inpainting. The second part of edge-guided GAN is like the Edges-to-Image part, which takes 2 channels of data as input and repairs this input by the designed GAN architecture.

The designed GAN is compose of a generator (G) and a discriminator (D). The purpose of generator (G) is to make the simulated sampling distribution close to the real data distribution to cheat discriminator (D). The purpose of discriminator (D) is to improve the discriminant ability by learning and trying to identify the sample distribution generated by generator (G). Therefore, in the actual training process, generator (G) and discriminator (D) are alternately trained. At this point, generator (G) has enough generating ability to simulate the real sample distribution to the maximum extent. The optimization objective of edge-guided GAN is to minimize the generator loss and maximize the discriminator loss:

$$\min_G \max_D L_G = \min_G (\alpha \max_D L_{adv} + \beta L_{gen}). \tag{15}$$

Here, L_{adv} and L_{gen} are discriminator loss and generator loss, respectively. α and β are parameters of L_{adv} and L_{gen}.

3 Conclusions

Generative adversarial network (GAN)-based approaches have been widely researched for color image inpainting and have achieved state-of-the-art results. However, there is still no GAN-based approach for depth image inpainting. This paper provides a GAN for depth image inpainting that contains two parts. The first part produces a deficient depth image's edge image and combines the results into 2 channels of data. The second part takes these data as input and repairs this input by the designed GAN architecture. This work not only can be used for depth image inpainting, but also can be used for objects removal if we extract objects as masks.

4 Acknowledgment

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1305002 and in part by the National Natural Science Foundation of China under Grant 61773414.

References

[1] Y. Zhang and T. Funkhouser, "Deep depth completion of a single rgb-d image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 175–185.

[2] H.-T. Zhang, J. Yu, and Z.-F. Wang, "Probability contour guided depth map inpainting and superresolution using non-local total generalized variation," *Multimedia Tools and Applications*, vol. 77, no. 7, pp. 9003–9020, 2018.

[3] A. K. Thabet, J. Lahoud, D. Asmar, and B. Ghanem, "3d aware correction and completion of depth maps in piecewise planar scenes," in *Asian Conference on Computer Vision*. Springer, 2014, pp. 226–241.

[4] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational mathematics*, vol. 9, no. 6, p. 717, 2009.

[5] F. Shi, J. Cheng, L. Wang, P.-T. Yap, and D. Shen, "Low-rank total variation for image super-resolution," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2013, pp. 155–162.

[6] H. Xue, S. Zhang, and D. Cai, "Depth image inpainting: Improving low rank matrix completion with low gradient regularization," *IEEE Transactions on Image Processing*, vol. 26, no. 9, pp. 4311–4320, 2017.

[7] X. Huang, X. Cheng, Q. Geng, B. Cao, D. Zhou, P. Wang, Y. Lin, and R. Yang, "The apolloscape dataset for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 954–960.

[8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[9] X. Zhu, Y. Liu, and Z. Qin, "Data augmentation in classification using gan," *arXiv preprint arXiv:1711.00648*, 2017.

[10] X. Han, L. Zhang, K. Zhou, and X. Wang, "Progan: Protein solubility generative adversarial nets for data augmentation in dnn framework," *Computers & Chemical Engineering*, vol. 131, p. 106533, 2019.

[11] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[12] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[13] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8798–8807.

[14] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He, "Attngan: Fine-grained text to image generation with attentional generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1316–1324.

[15] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2536–2544.

[16] N. Wang, J. Li, L. Zhang, and B. Du, "Musical: multi-scale image contextual attention learning for inpainting," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 2019, pp. 3748–3754.

[17] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[18] J. Li, W. Monroe, T. Shi, S. Jean, A. Ritter, and D. Jurafsky, "Adversarial learning for neural dialogue generation," *arXiv preprint arXiv:1701.06547*, 2017.

[19] Z. Yang, W. Chen, F. Wang, and B. Xu, "Improving neural machine translation with conditional sequence generative adversarial nets," *arXiv preprint arXiv:1703.04887*, 2017.

[20] X. Han, Z. Zhang, D. Du, M. Yang, J. Yu, P. Pan, X. Yang, L. Liu, Z. Xiong, and S. Cui, "Deep reinforcement learning of volume-guided progressive view inpainting for 3d point scene completion from a single depth image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 234–243.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[22] K. Nazeri, E. Ng, T. Joseph, F. Z. Qureshi, and M. Ebrahimi, "Edgeconnect: Generative image inpainting with adversarial edge learning," *arXiv preprint arXiv:1901.00212*, 2019.

Technical Activities

1 Conferences and Workshops

- IEEE International Conference on Industrial Cyber-Physical Systems (ICPS 2020)
- IEEE International Conference on Cyber Physical and Social Computing (CPSCom 2020)
- Design Automation for CPS and IoT (DESTION) 2020

2 Special Issues in Academic Journals

- IEEE Transactions on Industrial Informatics special issue on Cloud-Edge Computing for Cyber-Physical Systems and Internet-of-Things (Submission deadline: Mar. 15, 2020)

3 EiC Call For Nomination

- Call for Nominations Editor-In-Chief of ACM Transactions on Design Automation of Electronic Systems

Call for Contributions

Newsletter of Technical Committee on Cyber-Physical Systems (IEEE Systems Council)

The newsletter of Technical Committee on Cyber-Physical Systems (TC-CPS) aims to provide timely updates on technologies, educations and opportunities in the field of cyber-physical systems (CPS). The letter will be published twice a year: one issue in February and the other issue in October. We are soliciting contributions to the newsletter. Topics of interest include (but are not limited to):

- Embedded system design for CPS
- Real-time system design and scheduling for CPS
- Distributed computing and control for CPS
- Resilient and robust system design for CPS
- Security issues for CPS
- Formal methods for modeling and verification of CPS
- Emerging applications such as automotive system, smart energy system, internet of things, biomedical device, etc.

Please directly contact the editors and/or associate editors by email to submit your contributions.

Submission Deadline:

All contributions must be submitted by **Jul. 1st, 2020** in order to be included in the February issue of the newsletter.

Editor:

- Bei Yu, Chinese University of Hong Kong, Hong Kong byu@cse.cuhk.edu.hk

Associate Editors:

- Xianghui Cao, Southeast University, China xhcao@seu.edu.cn
- Long Chen, Sun Yat-Sen University, China chenl46@mail.sysu.edu.cn
- Wuling Huang, Chinese Academy of Science wuling.huang@ia.ac.cn
- Yier Jin, University of Florida, USA yier.jin@ece.ufl.edu
- Abhishek Murthy, Philips Lighting Research, USA abhishek.murthy@philips.com
- Rajiv Ranjan, Newcastle University, United Kingdom raj.ranjan@ncl.ac.uk
- Muhammad Shafique, Vienna University of Technology, Austria mshafique@ecs.tuwien.ac.at
- Yiyu Shi, University of Notre Dame, USA yshi4@nd.edu
- Ming-Chang Yang, Chinese University of Hong Kong, Hong Kong mcyang@cse.cuhk.edu.hk