Evaluating Edge Processing Requirements in Next Generation IoT Network Architectures

Brooks Olney, Shakil Mahmud, Robert Karam

Evaluating Edge Processing Requirements in Next Generation IoT Network Architectures

Brooks Olney, Shakil Mahmud, and Robert Karam

University of South Florida, Tampa FL 33620, USA

Abstract. The Internet of Things (IoT) is a massively growing field with billions of devices in the field serving a multitude of purposes. Traditionally, IoT architectures consist of "edge" sensor nodes which are used purely for data collection, actuators for intermediary connectivity, gateways for transmitting data, and cloud servers for processing. However, as application requirements change, IoT architectures must evolve. In the next generation of IoT, traditional architectures may not hold due to power limitations, privacy concerns, or network reliability in certain environments. In particular, Artificial Intelligence (AI) and Machine Learning (ML) applications – especially image/video processing – have become increasingly widespread, and are famously data and compute intensive applications. In this paper, we give an overview and describe the shortcomings of traditional IoT architectures, and outline key scenarios where local processing may be preferred over transferring data to the cloud. We then describe and evaluate metrics which designers can use to assess the needs of their IoT platform. This framework will be beneficial to IoT device manufacturers and network architects for improving usability and reliability, while protecting privacy in the next generation of IoT.

Keywords: internet of things (IoT), artificial intelligence, machine learning, image processing, low power

1 Introduction

The Internet of Things (IoT) has become a pillar of the integrated circuit (IC) market in recent years, with an estimated 212 billion devices predicted to be deployed globally by the end of 2020 [11]. Today, more devices are connected to the internet than ever before. Traditional IoT architectures consist of sensor nodes which are used for data collection, and/or provide some basic means of remote control [33]. For example, smart home components such as smart thermostats, smart fridges, and smart TVs allow consumers to control them using apps on their mobile devices, or perform self regulation based on a list of settings configured by the user. One reason that the market for IoT devices has experienced tremendous growth is because of the ability to provide convenience to consumers in their own home. Another application space seeing a surge in IoT usage is in health care, where devices may provide monitoring services to patients in hospitals, as well as assisted living services at home [4]. In this architecture, devices

typically send data through gateways to the cloud for processing. However, as the nature of these workloads changes, e.g. becoming increasingly data intensive, the cost of data transfer may eventually exceed that of processing the data locally. Power constraints, concerns for data privacy in cloud services, or network reliability issues may ultimately lead to more reliance on *edge computing*, where edge nodes are responsible not only for data collection, but some data preprocessing as well. For example, many individuals use their smart phones to capture pictures and video which they then share to social media platforms. In this case, the image or video is rather large due to the improvement of camera quality in smart phones, incurring large overhead in network bandwidth requirements. Such devices typically have the ability to adjust and compress the image or video data before transferring to the cloud, reducing bandwidth overhead at the cost of requiring additional local processing power. However, this is just one example of a need for sophisticated edge computing. Other examples include self-driving automobiles, which may generate almost one Gigabyte of data every second [22]. In the case for self-driving automobiles, suitable compute platforms are implemented within the vehicles as cloud processing cannot be relied on due to latency. Local processing in real-time is required because the on-board computer must make decisions as quickly as possible to avoid collision with other vehicles or pedestrians. So, Artificial Intelligence (AI) and Machine Learning (ML) applications like visual processing in autonomous vehicles make up a large percentage of current compute and data requirements in the cloud.

There are many different sub-fields within AI and ML, with one of the most popular being deep learning or Deep Neural Networks (DNNs) and Convolutional Neural Networks (CNNs) [23], [15]. The concept of deep learning involves the use of a backpropagation algorithm that fine tunes the machine's model based on the error between the values computed by the model and the true target value in the training dataset. With repeated backpropagation during training, the model is able to update the weights in each layer to be more accurate to predict or classify items in real-time. Because of this, DNNs have had tremendous breakthroughs in processing and classifying images, video, speech and audio [3], [34]. Recurrent Neural Networks (RNNs) have also gained popularity in problem spaces with sequential data inputs [7]. In general, the fields of AI and ML have hinted at promising, robust solutions for diverse problem spaces; however, they are also famously demanding on compute and memory resources, both in training of the model and in prediction and classification during feedforward operation [21].

General purpose processors (GPPs) such as central processing units (CPUs) are not preferred for processing of ML workloads as they are more optimized for solving sequential problems, whereas neural networks are highly parallel in nature. The fundamental building block in such networks is the multiply-and-accumulate operation (MAC). Each MAC consists of three memory reads - to read the filter weight, filter map activation, and partial sum. Because of this, memory access becomes the largest bottleneck for processing these networks. For example, feedforward in the AlexNet architecture consists of 724 million MACs, which requires nearly 3 billion DRAM accesses [29]. Thus, cloud processing of

these workloads typically leverages the parallel architecture and abundance of on-chip dynamic random-access memory (DRAM) in graphics processing units (GPUs). However, GPUs are very power hungry, and often consume extra power through static power requirements of the given GPU. Because of this, GPUs are only seen in edge devices like self-driving automobiles. An example of this is Nvidia's DRIVE PX Pegasus program which is capable of level 5 (i.e. no human intervention required at all) autonomous driving [18]. For autonomous vehicles, the extra power overhead *may* be acceptable considering the design requirements. However, the same may not be said for edge devices which are required to operate with low power, and/or in environments with minimal internet connectivity. In recent years, Field Programmable Gate Arrays (FPGAs) have been explored as an alternative to conventional ML workload accelerators due their power efficiency, which is measured as the number of Giga-operations per second per Watt, GOP/s/W [25] [14]. The in-field reconfigurability of FPGAs makes them an attractive option for designers implementing deep learning models, as the models themselves have variance in terms of layers, filter size, optimizers, activation functions, etc. Because FPGAs have the advantage in terms of maximizing performance per Watt of power consumption, and can be configured for customized acceleration of ML workloads, they may be especially suited to assess the needs of edge computing in IoT.

Because IoT devices generate data in such large quantities, and deep learning is highly efficient in analyzing complex data, deep learning is becoming an attractive technology for IoT applications [16]. However, ML workloads (including deep learning) are very compute and data intensive tasks. Thus, the typical architecture where data processing is offloaded to the cloud may not be amenable to ML applications. There are several reasons why a designer may wish to perform such computation at the edge. One such reason is that there may be limited or unreliable network connectivity. Many IoT applications require redundancy and minimal latency. So, offloading tasks to the cloud may not be able to meet those design requirements. Besides the fact that transmitting images in real-time may not be efficient in terms of power and bandwidth, transmitting images over the network for cloud processing presents some security and privacy risks as well. Additionally, privacy may be a critical design requirement for the device, and *no* data should be shared with the cloud. Finally, the power consumed to transmit the images to the cloud may exceed the processing power to perform analysis locally. In environments where there is limited or unreliable network connectivity - and potentially in an untrusted environment - local processing may be a *requirement*. In cases where the power consumption of data transmission exceeds local processing power, it may be beneficial to trade off local processing with cloud processing. For example, high level or partial processing of images to determine "important" events may be done prior to transmission in order to minimize power consumption. To the best of our knowledge, there is no comprehensive framework for designers to determine whether local or cloud processing (or some combination of the two) is more appropriate for the application they are targeting.

The main contributions of this paper are as follows:

1. We summarize the current state of the art for IoT architectures and common use cases, as well as relevant image preprocessing in edge computing workloads
2. We describe a framework for balancing design requirements with capabilities of edge nodes for compute and data intensive applications using different optimization metrics
3. We evaluate the framework in real-world situations using current standards for IoT processing and transmission hardware in typical network environments

In summary, this paper establishes a comprehensive framework for IoT device and network design as application requirements evolve towards more data and compute intensive workloads, in the context of ever-increasing privacy concerns.

The rest of the paper is structured as follows: Section 2 discusses common IoT architectures present today, and describes some image processing techniques that may be employed by edge nodes. Section 3 presents our framework for determining when it may be more appropriate for designers to opt for local processing over cloud processing, and provides metrics for each individual case. Example case studies are provided to illustrate how the framework can be used to support using a particular edge compute fabric over alternatives. Finally, we conclude in Section 5.

2 Background

In this section, we provide a background on common trends in IoT architectures and give context for shifting processing power to the edge. We also elaborate on various image processing techniques used to minimize processing or network transmission overhead with AI and ML applications.

2.1 IoT Architectures

Generic Architecture: There is currently no general agreement on a "standard" IoT architecture. In recent years, researches have proposed various types of architectures. Conventional IoT architecture usually consists of the 3 layers: Perception Layer, Network Layer, and Application Layer. Besides these, the Middleware layer and Business layer were introduced in [13]. Figure 1 shows all of the five layers which are briefly described below:

Perception layer refers to the "layer of sensing", consisting of sensors which are used to identify relevant subjects and gather pertinent information, and actuators, which respond to electrical inputs and perform some physical action. This physical layer may contain various types of sensors and/or actuators which are chosen based on the applications requirement. The data collected in this layer could be information about the environment, physical objects, or programmable

parameters. A few of the technologies in this layers include Radio Frequency Identification (RFID) tags, wearable sensing devices, barcodes, and smart sensors.

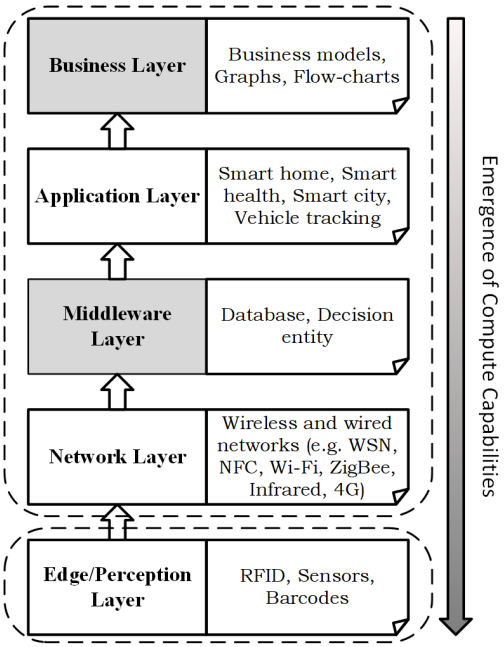

Fig. 1. The five-layered IoT architecture showing the different technologies or applications belonging to each layer, as well as move towards the edge for integrated compute capabilities.

Network layer connects the Perception layer and the Application layer. It also connects network devices, smart devices, and any service in the network. This layer is responsible for transmitting information gathered by the sensors from the perception layer to the internet, using, for example, Wireless sensor networks (WSN), Near Field Communication (NFC), and Bluetooth.

Middleware layer is also known as a processing layer which receives the information sent from the Network layer. In this layer, the received data gets processed, analyzed, and stored in the database. Based on the information processing and computational results, this layer can make decisions automatically. Additionally, it manages various type of services and has a connection to the database [13].

Application layer receives the processed information from the Middleware layer and based on the user demand, it manages service delivery globally. Smart home, smart city, smart health, vehicle tracking and smart farming are some applications which are implemented by IoT defined in this layer [13].

Business layer gets the data from the application layer and based on the information it generates charts, diagrams, business models, and profit models. This layer serves as a controller managing the application-specific services of the entire IoT system and is also responsible for the end users privacy.

Cloud Computing, Edge Computing and Gateway Architecture

According to National Institute of Standards and Technology (NIST), "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [17]. Resources in distributed environments are well managed by cloud computing, realizing greater performance. Instead of having a large storage and computing devices, cloud computing offers global access to the data. Highly manageable, schedulable and scalable virtual servers, networks, computational power, and storage are provided by the cloud computing [1]. A few essential characteristics of cloud computing are extensive, multiple service providers, dynamic resource sharing, high reliability, network access from all over the place, flexibility in service management, adaptability, and cost-effective [35] [36].

Based on the type of the provided service, there can be three principal models of cloud computing which are briefly described below:

1. **Infrastructure as a Service (IaaS):**
   IaaS provider provides infrastructural resources such as Virtual Machines (VMs), data center space, network equipment or servers as a service on-demand. With IaaS clients are benefited with great flexibility of managing their resources.

2. **Software as a Service (SaaS):**
   User applications are commonly mentioned as Software as a Service. In SaaS, a finished product is provided to the end-users which are managed by the service provider. Human resource management system and enterprise resource planning generally use SaaS [36].

3. **Platform as a Service (PaaS):**
   PaaS is another type of SaaS, where service is provided as a platform to develop and manage applications. PaaS provides a relaxed environment to the developers by not to think about software maintenance, patching, resource management, or any additional factors. The developers only need to concentrate on developing their programs and provide that to the end-users.

In recent years, vast amounts of data have been produced at the edge of the network, initiating another type of system architecture, *edge computing*. The basic idea behind edge computing is when the data is produced by the devices, it is processed closer to its origin rather than being offloaded to the cloud.

Edge computing has been evolving and growing rapidly to lower latency and to improve the efficiency when data is processed locally. As the cloud possesses significant computing power compared to the edge devices, this has been effective for compute, but not data-intensive, workloads. However, with a rapid increase of data at the edge devices introduce the delay of data transportation to the cloud as the *bandwidth bottlenecks* [26]. Autonomous vehicles exemplify this local computing requirement.

IoT Gateway is the point of connection between multiple smart devices and the cloud. It serves as the entry or exit point of the data to or from the cloud. A smart home is a good example of IoT gateway functionality. Inside the home, it creates a local network of all the smart devices connected which can communicate internally. Generally, a gateway will support one or more wireless communication standards in addition to WiFi, enabling it to transfer data to and from edge devices from different manufacturers.

In general, all smart devices are connected to the internet and synchronize with each other by exchanging information seamlessly. However, there may arise a state when - for a very short time - there is no requirement of data transportation to the cloud as that data is not required at the given moment. In certain situations, either the IoT Gateway needs to be intelligent enough to interpret the environment and stop sending the data to the cloud. Or, the smart devices must stop producing data to have efficient usage of the network resources. Such a scenario introduces the "Smart Gateway" which performs some additional data processing and decides when and what type of data must be uploaded to the cloud [1] [2]. In the context of AI/ML, certain preprocessing techniques performed locally to compress or filter the data and minimize transmission overhead may serve similar purposes to a Smart Gateway.

2.2 Image Processing

In AI and ML applications, image processing techniques are used prior to inference or training of a neural network. Some goals of such techniques are to preprocess the images in order to prepare them for analysis through a DNN or other network, compress them for storage or network transmission, and filtering out certain images which do not contain pertinent information. Multiple techniques in image processing are typically used together to achieve these goals. In particular, they are necessary for optimizing ML workloads on IoT edge devices which operate under power and connectivity constraints.

preprocessing consists of operations that are performed on images at the lowest level of abstraction - that is, input data to preprocessing algorithms is the original, unfiltered image. The main goals of preprocessing techniques differ between the application. For images which are to be fed through a DNN, the main goals are to improve the quality of the image data, remove distortions that may skew the model, and enhance latent features which are important for further pro-

cessing by a DNN. preprocessing of images when training a model have similar goals in mind, but may also apply certain transformations to each image.

With image compression - especially prior to inference in a DNN - it is important to compress the images in such a way that minimizes the loss in quality and feature distinction, while maximizing the amount of storage space saved. Compressing the image too much can cause the model to miss features and misclassify certain events because of it. Thus, image and video compression algorithms have been an important contribution in image processing research [8] [6] [37]. Some widely used image/video compression algorithms are:

1. **JPEG** (Joint Photographic Experts Group) [30] is used for single-frame image compression, and is the most widely known and used standards for such tasks. It is typically performed with up to a 16:1 compression ratio without a visible degradation in quality or detail.
2. **PNG** (Portable Network Graphics) is a lossless, portable compression algorithm that was designed to work very well for web-applications.
3. **H.264/MPEG-4** (Motion Picture Experts Group) is used for video compression for very low bitrates with aims for applications that target "conversational" and "non-conversational" applications [32]. Basically an approach that can be applied to streaming/broadcast and storage applications.

Filtering in the context of edge computing devices can be viewed as the process of filtering out images based on their content. For example, consider an edge sensor serving as a security camera which uses facial detection processed in the cloud to identify subjects of the camera feed. The device should not continually stream the video footage to the cloud for processing, as this is a clear waste of power to transmit over the network. Thus, the device should filter out images using some method of motion detection to tell whenever a person has entered the field-of-view (FOV) of the lens. This way, the device will only transmit when absolutely necessary, and minimize the power consumed to transmit, as well as the impact on bandwidth and the cloud.

Visual surveillance (including motion detection) has been a highly influential research topic in computer vision. Intelligent surveillance systems are much more practical than traditional passive surveillance which requires a human operator to constantly view security tapes [31]. Motion detection is incredibly important for edge devices which capture video feeds. Relating back to the security camera; in addition to filtering out images (video feed) that do not contain a person of interest, filtering can further minimize the amount of data transmission required by determining the frames with the *highest quality* capture of the subject of interest. For instance, assume that a suspect remains within the FOV of a security camera for 2 minutes. But, for most of the time the suspect has their back turned, and only reveals their face for a few seconds of that time. An advanced filtering technique could analyze this segment of data for facial features, and isolate the frames at which the subject is the most identifiable, and then transmit those frames to the cloud for processing. Assuming a camera which captures at 1080p (1920x1080 pixel array) with 30 frames per second

(FPS), and 8-bit color depth, the total size of the 2 minute segment would be $1920 \times 1080 \times 8 \times 30 \times 120 = 179,159,040,000$ bits. If we consider the codec for compression being H.264, and a bitrate for video quality of 8 Mbit/sec, then the size of the video would be around 120MB. However, most of the data in the video is unsuitable for facial detection. In this case, the only suitable frames may be within a few seconds of the entire footage. This means that the amount of data required to transmit could be reduced by up to 94% (considering 1 second of usable data). Thus, filtering can be a powerful tool to minimize overhead in edge devices, but requires significant computing resources to achieve such a reduction in data complexity.

Analysis of images is done using some form of DNN or CNN, depending on the application. The process of inputting an image into a neural network is referred to as an *inference* or feedforward operation. This may be done either locally, or in the cloud. Analysis will be done after preprocessing and filtering, or in real-time when filtering is not used. preprocessing is important to ensure that the images have the correct structure for the input layer to the network, and filtering is important to minimize the dynamic power consumption when running inference on a neural network locally, or for transmitting to the cloud for processing. In some cases, it may be beneficial to do local inference to conserve power and limit network bandwidth overhead. We quantify this in more detail in the next section.

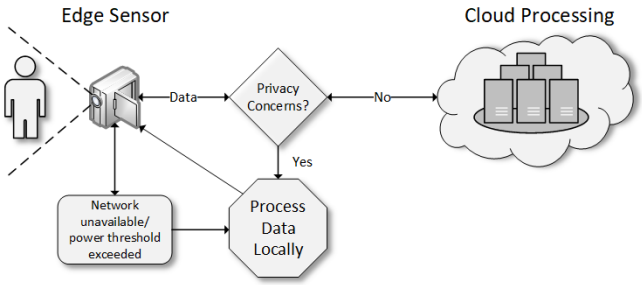

Fig. 2. Flow Diagram of proposed framework. The device can do some basic checks to determine if latency and power thresholds are met, and if there are security/privacy concerns before choosing between local or cloud processing.

In general, the image processing techniques we have described are very useful for edge devices. Whether certain methods like compression and filtering are done locally before transmitting to the cloud, or as part of the local processing paradigm, they can alleviate the overhead of storage and transfer from large amounts of data generation. Such techniques are important design considerations and can be integrated within our proposed framework for trading off between local and cloud processing.

3 Edge Node Design and Analysis Framework

In this section, we describe the design and analysis framework for edge nodes handling data and compute intensive tasks. We outline three specific scenarios where more robust edge computing capabilities will be beneficial, providing an in-depth description for each particular case where designers may opt for local processing, or some combination of local and cloud processing for their IoT platform. Finally, we offer sample case studies which quantify the requirements in each scenario using state-of-the-art wireless network hardware for data transfer to the cloud, as well as efficient FPGA implementations of common neural network models from literature. Figure 2 provides a basic flow diagram of the proposed framework.

3.1 Limited Network Connectivity

In Section 2, we have described the generic architecture of IoT consisting of 5 layers. Of the 5-layers, the perception/edge layer is the lowest physical layer and consists of different types of sensors, which are the key elements of that layer. To quantify when designers may opt for edge computing over cloud computing in limited or poor network connectivity, we first need to understand the flow from the sensor to the actuator. Sensors sense the data, gather information, and transmit to the gateway. The gateway then sends the data to the cloud and receives a response after the data has been processed. After that, the actuator functions according to the response received from the cloud. Efficiency of this entire process depends on bandwidth, latency, and throughput.

Bandwidth is defined as the maximum amount of data that can be sent through a wired or wireless transmission link from one node to another in a given amount of time. It is the data transfer rate across a network and expressed in bits per second (bps). *Latency* is the time required for a data packet transmitting from its source node to the destination node. One of the common ways of measuring latency is calculating the *Round Trip Time (RTT)*, which is the duration a single packet requires to travel from the source node, reach the destination node and back to the source node. Latency is an accumulation of all the delays involved in the transmission. Specifically, the type of transmission medium, and the time required for propagation, routing, switching, or storage are several factors which affect network latency. We will use the term latency and round-trip-time interchangeably. Another important term in measuring network performance is *throughput*, which is the maximum amount of data that successfully travels through the transmission link, and is expressed in bits per second (bps). The maximum throughput of a system can be calculated by Eqn. 1. The Transmission Control Protocol (TCP) is a connection-oriented protocol where both the sender and the receiver keep track of the transmitted data so that any non-received data can be re-transmitted to maintain stable data transportation. TCP window size, expressed in bytes, is the maximum possible data sent before receiving an acknowledgment (ACK) message which can be computed by Eqn. 2. When the network is unstable, it is better to maintain a small TCP window to

refrain from re-transmitting the data, but with a reliable network, the TCP window size can be considerably large. Another feature of TCP is the sliding window protocol, where the devices can dynamically scale the TCP window based on the available connectivity.

$$\text{Max. Throughput} = \frac{\text{TCP Window Size}}{\text{RTT}} \tag{1}$$

$$\text{TCP Window Size} = \text{Bandwidth (BW)} \times \text{RTT} \tag{2}$$

$$\text{RTT or Latency} = \frac{\text{TCP Window Size}}{\text{Bandwidth (BW)}} \tag{3}$$

Edge nodes with compute capability have the opportunity to maintain system functionality even if the network connection is lost. If the network is good, processing can be done in the cloud as before, and the additional compute resources may be turned off to minimize static power dissipation [12]. We propose a dynamic model where the edge nodes automatically decide between performing local computation or cloud-based computation depending on the latency and available bandwidth. We are considering the TCP window size as constant, because dynamic scaling of TCP window is an expensive operation which will not be cost effective for edge devices. We suggest the designers should take into account the packet loss during transmission when determining the TCP window size. The edge nodes can calculate the total RTT or latency using Eqn. 3. After that, it compares the calculated latency with the acceptable latency that has been defined earlier by the system designer and makes the decision of either local processing or cloud-based processing.

For example, consider an edge node with an available bandwidth of 100 Mbps, a constant TCP window size of 64 KB, and the acceptable round-trip-time or latency is 50 ms. Using Eqn. 3, the edge node calculates the latency as, (64000×8) bits/100000000 bps = 5.12 ms. Comparing the calculated latency with the acceptable latency, the edge node decides to process the data at the cloud as the computed latency is less than the acceptable latency.

Now, we consider the same edge node operating at limited network connectivity. For example, with a bandwidth of 1 Mbps, but with the same TCP window size of 64 KB and acceptable latency of 50 ms. Similar to the earlier calculation, the edge node computes the latency as 512 ms which is much higher than the acceptable latency of 50 ms. Under certain circumstances, processing the data at the cloud is not feasible as it will incur a very high delay. Our approach may suggest that the edge node should process the data locally rather than in the cloud.

3.2 Privacy and Security Concerns

Privacy and security are significant concerns in cloud computing. Designers and end users alike should be wary about the many privacy and security risks associated with IoT devices, such as having credentials compromised, breaches of

private data, violation of data sharing agreements, phishing, and other trust issues. A numerical value quantifying the risk associated with sharing different data to the cloud could enable a data gathering edge node to opt for local processing. Specifics of such a metric would necessarily be determined at design time, considering the environment, data, use cases, and other variables. Here, we describe some of the privacy and security concerns relating to edge-gathered data, which may motivate designers to integrate additional compute capabilities into edge nodes.

General Security Threats: Cloud computing suffers from a number of security threats. Multiple scenarios are briefly described below:

- *Botnets and other malware* represent one of the major threats against the security and reliability of cloud computing. It contains many connected devices, storage, and memory. Botnet can be used to execute a distributed denial-of-service (DDoS) attack and steal the organization's data. The attackers execute the attack by controlling a huge number of automated bots which are malware operated [28].
- *Multitenancy in Cloud Servers* When the same physical machine shares resources like a virtual machine (VM), data, or hard disk drive, this may enable unanticipated side-channels between different running processes, especially if there is no separation between data and processes on a virtual level [9]. This may result in data leakage between different applications.

Integrity and Confidentiality of Personal and Private Information: Integrity means the assets can only be modified after an approved agreement from the owner, while confidentiality refers to ensuring private information is only accessible to the appropriate and trusted parties.Examples of Personal Identifiable Information (PII) include name, address or location data, biometric data, and images, among others [19]. Moreover, information related to religion, health, race, sexual orientation, or finance are considered highly sensitive. Preventing unauthorized access and tampering of private and sensitive data is one of the largest challenges for cloud service providers, and one of the greatest concerns for enterprise customers and consumers alike. Providing an option for local data processing obviates the need for trusting third parties with sensitive data.

Military and Government Applications: Military applications may generate a large amount of sensitive data, from creating a cohesive fighting force, to locating an enemy, or observing a soldiers current state, especially from health sensor networks. These data must be secured from any sort of leakage as the contents are highly confidential. Edge computing is preferred for the Internet of Military/Battlefield Things (IoMT and IoBT) because of the sensitive nature of the data produced by IoMT and IoBT devices [5]. More generally, government regulations in non-military contexts still have strict regulations regarding the storage and sharing of certain data. For example, types of PII, how long those data can be saved, or to whom those data an be shared. Either due to dishonest

cloud providers or vulnerabilities in the cloud, it can result in transferring the data to an outside country, which is completely against the privacy law. Hence, military and government contexts for IoT applications would greatly benefit from additional edge processing capabilities.

Whether due to an end-user feeling uncomfortable sharing edge-gathered data with cloud services, or strict rules exist regulating the types of data that can be shared, the option to process at the edge rather than in the cloud for the sake of privacy and security is becoming increasingly important. If the edge nodes do not possess the capability of processing data locally, then the end-users have no choice, and little control on how data are used in the cloud.

3.3 Transmission Power Exceeds Local Processing Power

Given the application of the edge device, it may be possible that the power consumed to transmit data to the cloud for processing exceeds that of local processing on the device itself. In this case, a designer may decide to implement some protocol to trade off processing between local and cloud platforms, or do all processing locally. We propose a dynamic model which can be used by the device to assess it's own requirements and circumstances in real-time, and autonomously make the decision to transmit to the cloud, or process its data locally. To quantify the metrics to decide based on the power consumption, we must consider state-of-the-art WiFi components, as well as an example neural network implementation in FPGA to compare the overhead of local processing to offloading to the cloud. There are three factors that play into assessing the power requirements of cloud processing; namely, the size of the payload, the maximum transmission bitrate, and the power consumed by the transmission circuitry during operation.

To determine payload size, there are three main considerations; the size of the image, the number of frames required to transmit for the target application, and the compression and filtering algorithms used. The size of the image will vary based on the quality of the camera in the device, as well as the compression ratio. The number of frames is dependent on the application, and if any form of filtering is done prior to transmission. In our example with the security camera; with proper filtering techniques, the number of frames is dependent on the duration which the subject's face is distinguishable in the FOV. On the other hand, the application could only require a single frame, or a constant feed. So, we denote the size of the image S as $width \times height \times bit\text{-}depth = S$, the number of required frames as F, the energy consumed by compressing the video as C, and energy from filtering as M. For network transmission in edge devices, the relevant attributes of the transmission circuitry are the power consumed during operation P, and the maximum throughput T. The total power consumption of each is then calculated by multiplying S and F together to get the total amount of data to transmit, then dividing by the throughput of the wireless standard, and finally multiplying by the rated power consumption of the same standard. However, this does not complete the picture. As we mentioned, security and

privacy are also concerns. So, we have to consider that the data is encrypted before transmission. For data encryption, we denote E as the power consumed by encrypting a byte of data. The total energy expended to transmit a payload of size $S \times F$ is computed by adding the power consumed by encrypting the data, and the power consumed by transmitting. This is represented by Eqn. 4 in terms of Joules (J).

$$P_{transmit} = (C \times F) + (M \times F) + (E \times S \times F) + \frac{S \times F \times P}{T} \tag{4}$$

The power consumed to transmit to the cloud is shown to be directly proportional to the image size and the amount of frames to transmit (total payload size), as the power and throughput of the transmitting circuitry will remain static. These values will changed based on the native quality of the camera used, whether or not there is any preprocessing or compression prior to transmission, any filtering using motion detection, etc. We compare these values to an example state-of-the-art implementation on FPGA. For this purpose, we refer to the work in [10], a low-power CNN accelerator implemented on a Zynq XC7Z045 platform. The authors experimented with CNN-based optimization methods to improve the performance and power efficiency of the VGG16 network [27] in comparison to other FPGA implementations on the same platform. They reported two separate configurations; one with 16-bit fixed precision, and one with 8-bit. We base our estimations on the 8-bit precision implementation. For this implementation, they reported the performance at 84.3 GOP/s, and power consumption at 3.5W - yielding an energy efficiency of 24.1 GOP/s/W. Additionally, their implementation of VGG16 requires 30.76 GOP per image. Using these values, we can estimate the average power consumption during operation per image as $(30.76/84.3) \times 3.5 = 1.28$J/img. For image compression, we consider two different types, 4L H.264 and 4L HEVC, with energy consumption values of 461 and 606 mJ/frame, and average compression ratio of 10.49:1 and 12.5:1, respectively [16]. For image filtering, we consider the power consumed by a Texas Instruments TIDA-01398 PIR motion sensor as 1.57 mW. We also consider two of the most widely used wireless standards in IoT. Namely, Bluetooth Low Energy (BLE) and ZigBee. Each are typically used in some combination as they operate fairly complimentary to each other. Each is around the same transmission speed with 270 kbps in BLE and 250 kbps in Zigbee. BLE is effective at shorter ranges with 10mW transmit power, whereas Zigbee is effective at a much greater distance but with a higher transmit power of 100mW [20]. For E, we use the value of power consumption in AES at around 4.2 μJ/byte as measured by [24]. Figure 3 shows graphs of the energy consumed to transmit or process a certain number frames (segment of a video). Estimations are made for native resolution at 1080p (a)(c), and 720p(b)(d). For comparison, (a-b) use 4L H.264 compression with 8-bit color depth, and (c-d) use 4L HEVC with 12-bit color depth.

For the higher quality resolutions with 1080p (Fig 3(a,c)), the larger file size contributes to a larger power consumption to transmit to the cloud. So, in these situations it may be more energy efficient to do processing locally. With lower resolutions (Fig 3(b,d)), the energy to transmit is lower, so offloading to the

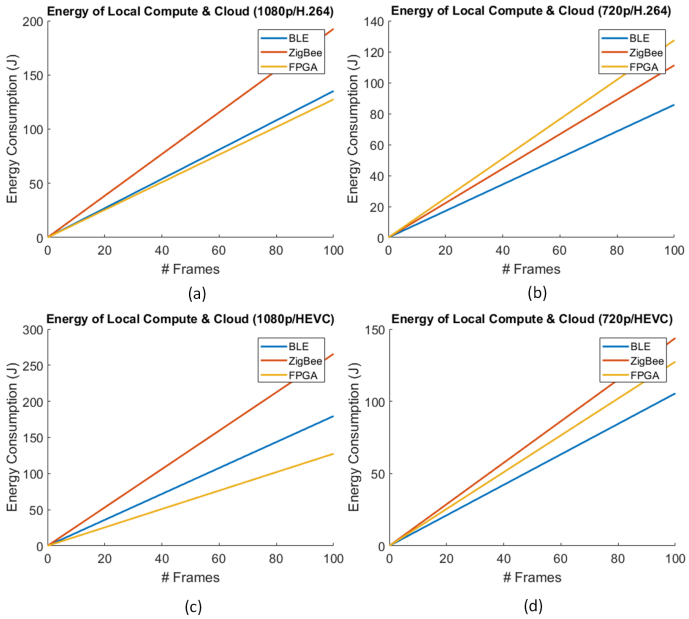

Fig. 3. Comparison of power consumed from local and cloud processing.

cloud may be the best option. While 4L HEVC has a very large compression ratio of 12.5:1, it is much more computationally complex than other compression standards, contributing to a much higher energy consumption. An important characteristic to note is the power consumed by using advanced filtering techniques. In these examples, we considered a static power draw from a motion detector that may be used during the duration of the transfer. However, a more sophisticated filtering system would require a frame-by-frame analysis where some frames are excluded due to their content. The energy consumed during such a task is hard to estimate, so we elect to use a static value in this example. To arrive at the optimal solution, one must factor in the energy cost of the frame filtering techniques used, as well as the number of actual frames subject for processing. For example, considering the 2 minute security camera example from earlier, we assume that processing such a video locally would be more energy efficient than transmitting to the cloud. However, it could be possible that a filtering technique reduces the amount of frames with pertinent data to 1 second of video, resulting in a much smaller data payload. In that case, transmitting to the cloud may be more efficient, considering the energy consumed to filter and encrypt the video data. Thus, there is a wide range of design space exploration for sophisticated filtering techniques in image processing.

4 Preliminary Results

To provide a comparison between local processing and cloud processing, we conducted a simple image processing experiment using a Raspberry Pi Zero – a

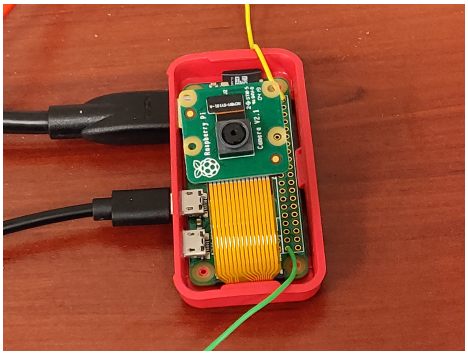

Fig. 4. Experimental setup of image processing application on Raspberry Pi Zero board.

cheap off-the-shelf device suitable for IoT applications. In this experiment, we utilized OpenCV to perform simple face detection algorithm that can detect multiple faces simultaneously. We also set up a Lambda server on Amazon Web Services (AWS) which accepts the video feed from the Raspberry Pi, runs the same face detection code, and sends the resulting images back to the device. We ran these scripts separately and measured the power consumption during operation in both scenarios. The experimental setup is shown in Figure 4.

Preliminary results are shown in Table 1. In this experiment, processing the data in the cloud was shown to consume less power than processing the data locally. So, if our proposed framework were used in this scenario, the device would elect to process the data in the cloud. However, there are many factors to consider, such as the application used (face detection, face recognition, object detection, etc.), and the underlying hardware and the implementation of the algorithm(s).

Table 1. Preliminary results for local processing and cloud processing.

Experimental Procedure	**Resolution**	**Framerate**	**Throughput**	**Power**
Face Detection (without faces)	320x240	20 FPS	4.61 MBps	1.4524 W
Face Detection (with faces)	320x240	20 FPS	4.61 MBps	1.5118 W
Cloud Processing	320x240	15 FPS	3.46 MBps	1.3714 W

5 Conclusion

In this paper, we have provided a background on different IoT architectures, AI and ML workloads, and staple image processing techniques useful for reducing IoT edge network bandwidth requirements. Edge computing is rapidly becoming a critical design requirement as deep learning workloads consume a disproportionate amount of compute, memory, and network resources. To assess this demand, we have created a framework that considers network capabilities, privacy requirements, and power constraints, enabling designers to more accurately

assess whether or not – and the extent to which – local processing capabilities should be considered in edge device design. We envision the integration of this framework within edge nodes, enabling them to independently determine the optimal utilization of computing, memory, and network resources when necessary. This represents a shift from traditional architectures towards a smarter and more scalable IoT.

References

1. Aazam, M., Huh, E.N.: Fog computing and smart gateway based communication for cloud of things. In: 2014 International Conference on Future Internet of Things and Cloud, IEEE (2014) 464–470
2. Aazam, M., Khan, I., Alsaffar, A.A., Huh, E.N.: Cloud of things: Integrating internet of things and cloud computing and the issues involved. In: Proceedings of 2014 11th International Bhurban Conference on Applied Sciences & Technology (IBCAST) Islamabad, Pakistan, 14th-18th January, 2014, IEEE (2014) 414–419
3. Abdel-Hamid, O., Mohamed, A., Jiang, H., Deng, L., Penn, G., Yu, D.: Convolutional neural networks for speech recognition. IEEE/ACM Transactions on Audio, Speech, and Language Processing **22**(10) (Oct 2014) 1533–1545
4. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M.: Internet of things: A survey on enabling technologies, protocols, and applications. IEEE Communications Surveys Tutorials **17**(4) (Fourthquarter 2015) 2347–2376
5. Castiglione, A., Choo, K.K.R., Nappi, M., Ricciardi, S.: Context aware ubiquitous biometrics in edge of military things. IEEE Cloud Computing **4**(6) (2017) 16–20
6. Christopoulos, C., Skodras, A., Ebrahimi, T.: The jpeg2000 still image coding system: an overview. IEEE Transactions on Consumer Electronics **46**(4) (Nov 2000) 1103–1127
7. Connor, J.T., Martin, R.D., Atlas, L.E.: Recurrent neural networks and robust time series prediction. IEEE Trans. on Neural Networks **5**(2) (March 1994) 240–254
8. Delp, E., Mitchell, O.: Image compression using block truncation coding. IEEE Trans. on Communications **27**(9) (Sep. 1979) 1335–1342
9. Dillon, T., Wu, C., Chang, E.: Cloud computing: issues and challenges. In: 2010 24th IEEE international conference on advanced information networking and applications, Ieee (2010) 27–33
10. Guo, K., Sui, L., Qiu, J., Yu, J., Wang, J., Yao, S., Han, S., Wang, Y., Yang, H.: Angel-eye: A complete design flow for mapping cnn onto embedded fpga. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems **37**(1) (Jan 2018) 35–47
11. J. Gantz, D.R.: The digital universe in 2020: Big data bigger digital shadows and biggest growth in the far east. IDC iView: IDC Anal. Future (2007) (December 2012) 1–16
12. Kao, J.T., Chandrakasan, A.P.: Dual-threshold voltage techniques for low-power digital circuits. IEEE Journal of Solid-state circuits **35**(7) (2000) 1009–1018
13. Khan, R., Khan, S.U., Zaheer, R., Khan, S.: Future internet: the internet of things architecture, possible applications and key challenges. In: 2012 10th international conference on frontiers of information technology, IEEE (2012) 257–260
14. Lacey, G., W. Taylor, G., Areibi, S.: Deep learning on fpgas: Past, present, and future. (02 2016)

15. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521** (May 2015) 436
16. Li, H., Ota, K., Dong, M.: Learning iot in edge: Deep learning for the internet of things with edge computing. IEEE Network **32**(1) (Jan 2018) 96–101
17. Mell, P., Grance, T., et al.: The nist definition of cloud computing. (2011)
18. NVIDIA: Artificial intelligence computer designed to drive autonomous cars (2017)
19. Pearson, S.: Taking account of privacy when designing cloud computing services. In: 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, IEEE (2009) 44–52
20. Ray, B.: A bluetooth & zigbee comparison for iot applications. https://www.link-labs.com/blog/bluetooth-zigbee-comparison (October 2015)
21. Ray, T.: Ai is changing the entire nature of compute — zdnet. https://www.zdnet.com/article/ai-is-changing-the-entire-nature-of-compute/
22. Rijmenam, M.: Self-driving cars will create 2 petabytes of data, what are the big data opportunities for the car industry? (December 2016)
23. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088) (October 1986) 533–536
24. Salama, D.D., Abd elkader, H., Hadhoud, M.M.: Performance evaluation of symmetric encryption algorithms on power consumption for wireless devices. International Journal of Computer Theory and Engineering (01 2009) 343–351
25. Shawahna, A., Sait, S.M., El-Maleh, A.: Fpga-based accelerators of deep learning networks for learning and classification: A review. IEEE Access **7** (2019) 7823–7859
26. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: Vision and challenges. IEEE Internet of Things Journal **3**(5) (Oct 2016) 637–646
27. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2014)
28. Somani, G., Gaur, M.S., Sanghi, D., Conti, M., Buyya, R.: Ddos attacks in cloud computing: Issues, taxonomy, and future directions. Computer Communications **107** (2017) 30–48
29. Sze, V., Chen, Y., Yang, T., Emer, J.S.: Efficient processing of deep neural networks: A tutorial and survey. Proc. IEEE **105**(12) (Dec 2017) 2295–2329
30. Wallace, G.K.: The jpeg still picture compression standard. IEEE Trans. on Consumer Electronics **38**(1) (Feb 1992) xviii–xxxiv
31. Weiming Hu, Tieniu Tan, Liang Wang, Maybank, S.: A survey on visual surveillance of object motion and behaviors. IEEE Trans. on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **34**(3) (Aug 2004) 334–352
32. Wiegand, T., Sullivan, G.J., Bjontegaard, G., Luthra, A.: Overview of the h.264/avc video coding standard. IEEE Transactions on Circuits and Systems for Video Technology **13**(7) (July 2003) 560–576
33. Xu, L.D., He, W., Li, S.: Internet of things in industries: A survey. IEEE Transactions on Industrial Informatics **10**(4) (Nov 2014) 2233–2243
34. Yi, D., Lei, Z., Liao, S., Li, S.Z.: Deep metric learning for person re-identification. In: 2014 22nd International Conference on Pattern Recognition. (Aug 2014) 34–39
35. Zhang, Q., Cheng, L., Boutaba, R.: Cloud computing: state-of-the-art and research challenges. Journal of internet services and applications **1**(1) (2010) 7–18
36. Zhang, S., Zhang, S., Chen, X., Huo, X.: Cloud computing research and development trend. In: 2010 Second international conference on future networks, Ieee (2010) 93–97
37. Zhou Wang, Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing **13**(4) (April 2004) 600–612