Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

A privacy protection approach in edge-computing based on maximized dnn partition strategy with energy saving

Guo Chaopeng, Lin Zhengqing and Song Jie*

Abstract

With the development of deep neural network (DNN) techniques, applications of DNNs show state-of-art performance. In the cloud edge collaborative mode, edge devices upload the raw data, such as texts, images, and videos, to the cloud for processing. Then, the cloud returns prediction or classification results. Although edge devices take advantage of the powerful performance of DNN, there are also colossal privacy protection risks. DNN partition strategy can effectively solve the privacy problems by offload part of the DNN model to the edge, in which the encoded features are transmitted rather than original data. We explore the relationship between privacy and the intermedia result of the DNN. The more parts offloaded to the edge, the more abstract features we can have, indicating more conducive to privacy protection. We propose a privacy protection approach based on a maximum DNN partition strategy. Besides, a mix-precision quantization approach is adopted to reduce the energy use of edge devices. The experiments show that our method manages to increase at most 20% model privacy in various DNN architecture. Through the energy-aware mixed-precision quantization approach, the model's energy consumption is reduced by at most 5x comparing to the typical edge-cloud solution.

Keywords Privacy protection, Edge-intelligent, DNN partition, Mixed-precision quantization, Energy optimization

Introduction

As the widespread deep learning, deep neural networks (DNNs) have achieved state-of-the-art performance in various research fields, such as image classification [1], object detection [2] and natural language processing [3] etc. With the improvement of the accuracy requirements for the inference results, the topology of DNN tends to be more and more complex, evolving from chain topology to directed acyclic graph (DAG) topology [4], which leads to tremendous computational resources that correspondingly necessitate substantial energy consumption.

In Internet of Things (IoT) environment, there are mainly three deployment models of DNNs shown in Fig. 1, which takes the classical LeNet5 [5] model as an example. In Fig. 1a, all inference steps of DNNs are deployed in the cloud. Edge devices upload raw data, such as images, texts, and videos, to the cloud, and the cloud returns the inference result for later use. This model provides high stability and availability for applications since cloud servers are well-protected in cloud data centers and are easier to control and protect. Also, edge devices can save considerable energy and computation resources for other tasks. However, it brings substantial privacy risks since the raw data is sent to the cloud, which might be hacked by a third party or used by the cloud services provider. In Fig. 1b, we adopt the edge-computing concept and try to offload the computation to edge devices as much as possible, namely all inference steps of DNNs are deployed in the edge, and then edge devices upload the inference result to the cloud. Under this model, the edge has the highest privacy level regarding DNN usage

*Correspondence:
Song Jie
songjie@mail.neu.edu.cn
Software College, Northeastern University, Shenyang, China

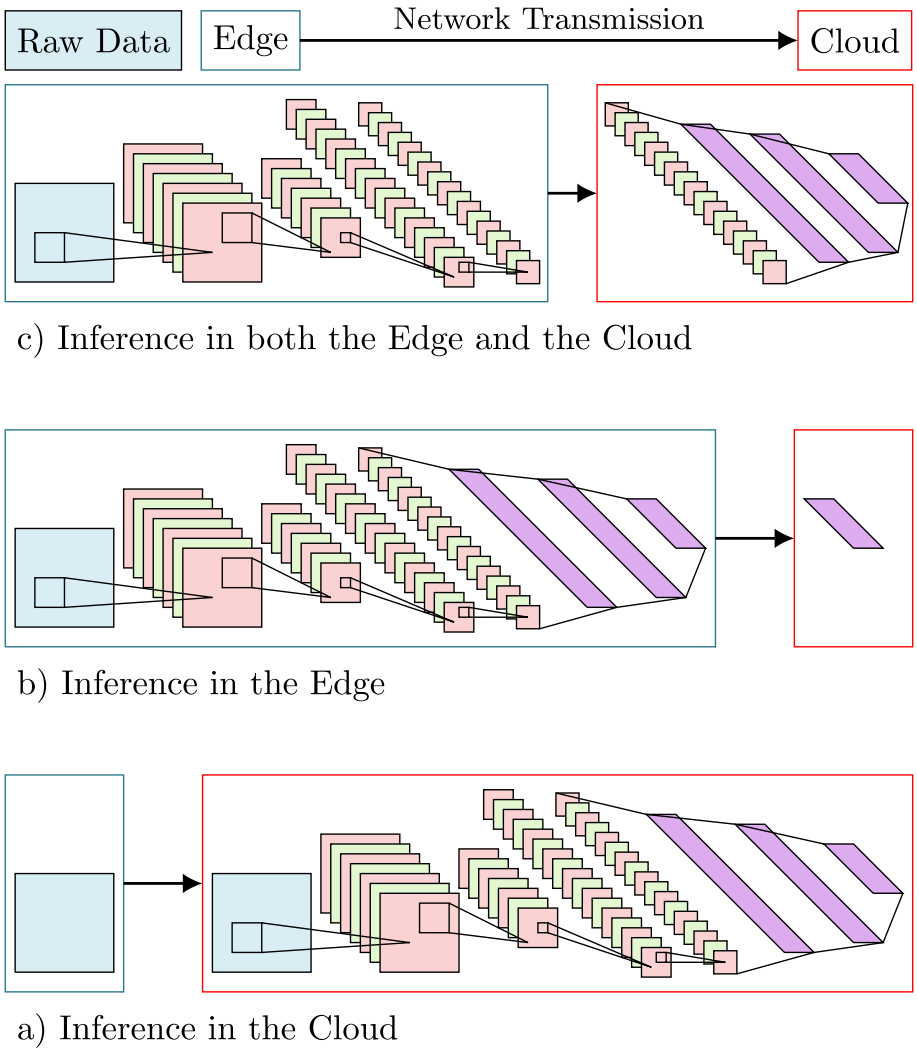

Fig. 1 Different deployment models for DNNs in IoT environment taking LeNet5 as an example

since the edge holds the raw data and the whole inference process. However, this method offloads all DNN to edge devices which could be harmful to application developers since malicious users can hack their models for other uses [6]. Moreover, edge devices have to pay for enormous computational resources and energy. The cost might be too huge to implement the model since edge devices are power-limited [7].

Both the academic and the industry utilize the hybrid model (Fig. 1c) to balance privacy and energy use. They introduced "DNN partition" approaches to split the DNN model and deploy part of the DNN model in the edge and another part in the cloud. After receiving the raw data, the edge executes the inference steps using a part DNN model and obtains the intermediate result (IR), namely feature maps. Then, the IR is sent to the cloud, finishing the rest of the inference steps. Since the edge only executes part of the inference steps, it saves computational resources and energy. Meanwhile, users' privacy is protected since the cloud only receives the IR, and it is not easy to reconstruct the original data from the IR directly [7].

DNN partition approach is privacy-related and energy-related. At first, previous works have proven that sending IRs to the cloud is beneficial for privacy protection compared with sending original data [7]. During the DNN inference process, along with the abstraction from each network layer, IRs are more distinguishable from the original input since each layer extracts task-relevant information and abandons the rest. Secondly part of the DNN model still consumes significant computational resources and energy on edge devices. Numerous

embedded devices, especially battery-powered or solar-powered devices, are power-limited, indicating that we cannot deploy too many DNN layers on the device. Therefore, how to utilize the DNN partition strategy to achieve maximum privacy protection within the edge device's energy budget becomes a critical problem.

To face the high energy demand challenge, many related kinds of research on energy optimization for DNN have emerged in academia and industry to reduce the energy use of DNNs in aspects of the hardware layer, the runtime environment, and the software layer. For example, the software-level optimization approaches include pruning [8], quantization [9], tensor decomposition [10], knowledge distillation [11], etc.

Quantification is an effective method to reduce the size and energy consumption of DNN by quantizing the weight and activation to lower precision. The most widely used quantization method is unified quantization, in which each layer of the neural network is quantified to the same number of bits. Unified quantization is a simple but effective quantization method that can significantly reduce the energy consumption of neural networks. However, with the emergence of more edge devices, unified quantization may not meet the energy consumption requirements of these devices. Since the nature and structure of each layer of the neural network are very different, the use of mixed precision quantization for different layers can get a higher energy consumption compression rate [12], in which different layers and different activations may have different quantization levels.

This work introduces an energy-aware mix-precision approach to DNN partition, named maximized DNN partition strategy, to maximize the user's privacy protection level within the edge device's power budget. Our intuition is shown in Fig. 2. The maximized DNN partition strategy not only splits the given DNN model but also quantizes the edge model partition to further reduce energy use. Our work's fundamental problem is finding a DNN partition-related mix-precision quantization

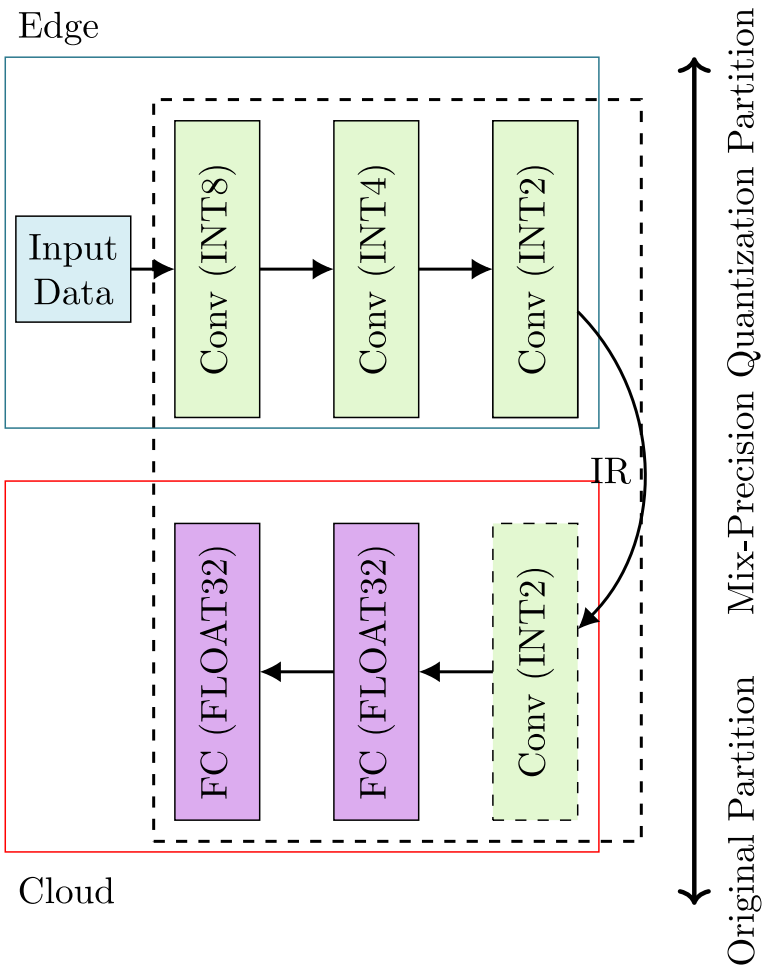

Fig. 2 Example of Mixed-Precision Quantization Model in DNN Partition Approach for Privacy Protection and Energy Save

strategy in which only prepositive layers are quantized and the energy consumption of the edge device is minimized.

We make the following contributions in this work:

- We analyze the relationship between privacy and IRs on DAG model architectures and propose block-wise partition method.
- We investigate the influence of the mix-precision quantization on privacy protection under the DNN partition strategy.
- We propose an energy-aware mix-precision quantization method to minimize the energy consumption of the DNN model for inference.
- We propose a maximized DNN partition strategy to maximize the user's privacy protection level.

The paper is organized as follows: Section Related works gives a brief literature review. Section Privacy analysis thoroughly investigates the influence factors of privacy. Section Maximized DNN partition modeling abstracts our problem with its constraints and objective as a searching problem. Section Searching algorithm describes a heuristic-based searching algorithm to find the maximum DNN partition strategy. Section Experiment gives a series of test cases to verify our algorithm and result. Section Conclusion and further works summarizes our work and points out further research directions.

Related works

This section mainly focuses on three aspects: DNN Partition, Deep Learning Privacy, and Neural Network Quantization. Firstly, we introduce the related approaches. Then, we give a brief discussion on the similarities and differences between this work and previous works.

DNN partition

In the IoT environment, lots of work have been done to propose various DNN partition strategies. Generally, these works can be categorized into two kinds according to their partition mechanism.

DNN models can be split into several edge devices to form a distributed DNN inference. MoDNN [13] is a locally distributed mobile computing system for DNN applications, which partition trained DNN model onto several mobile devices to accelerate DNN computations by alleviating device-level computing cost and memory usage. DeepThings [14], a framework for adaptively distributed execution of CNN-based inference applications on edge clusters, employs a scalable Fused Tile Partitioning of convolution layers to minimize memory footprint while exposing parallelism. Miao et al. [4] focused on the DNN models with the directed acyclic graph topology. They split the DNN inference task into several small jobs and assigned them to different edge devices through a load-balancing algorithm to accelerate the inference speed.

The DNN model is split between the edge and the server, which is widely adopted. The challenge is to have an appropriate partition point to achieve the specific objective. Neurosurgeon [15], a lightweight scheduler to automatically partition DNN models between mobiles and clouds via model analysis to reduce the computation latency and the energy use on the mobile side and maximize the throughput on the cloud side. Ko et al. [16] proposed a DNN partition strategy for resource-constrained IoT platforms. They coupled the DNN partition with a feature encoding mechanism to significantly improve the energy efficiency and throughput of the edge device. Edgent [17] adaptively partitions DNN computation between device and edge and utilizes right-sizing technique to accelerate DNN inference through early-exit at a proper intermedia DNN layer. Li et al. [18] split DNN models between edge devices and edge servers considering the task latency. To solve the excessive queueing delay in the edge servers, they introduced an early-exit mechanism to accelerate DNN inference while achieving high accuracy.

Our work follows the style that splits DNN model between the edge devices and the servers. Unlike the previous literature, we pay more attention to privacy and energy use of edge devices.

Deep learning privacy

In today's edge-cloud DNN inference mode, there are potential risks in user privacy. For example, user data can be abused by a malicious cloud provider for other usages, or the third-party attacker may hijack the data sent to the cloud through the network. Recently, with the development of IoT and the wide-spear of edge devices, more and more researches have begun to focus on privacy under the edge-cloud scenario. The main challenge is how to quantify the privacy risk and how to defend against privacy leakage.

Osia et al. [19] proposed a hybrid deep learning architecture to extract the necessary features from the input and send them to the cloud while casting out other irrelevant features which have the potential risk of being used to extract user privacy information. Tang et al. [20] proposed a framework to decide whether to upload the data to the cloud for higher performance or compute natively for privacy protection based on the data content. However, this method requires knowledge of what part of the information is sensitive and may cause potential privacy leakage. These can be impractical in actual

circumstances that users may face privacy risks from various perspectives.

Gu et al. [7] systematically analyzed the privacy risks in the edge-computing environment for image classification applications using DNN models. Based on their discovery, they proposed a Ternary Model Partitioning mechanism to mitigate the identified information leakages. Shi et al. [21] based on Gu's work, further study the privacy protection by reserving layers on the edge device based on the trade-off between DNN performance and energy consumption. They utilized Kullback-Leibler (KL) divergence to measure the privacy between different IRs and they have proven that more privacy edges can get if more DNN parts are offloaded to the edge. Based on their findings, they utilized DNN partition to balance the DNN performance in the edge device and the user's privacy protection level.

This paper utilizes the DNN partition mechanism to project users' privacy in the edge-computing environment. Our work is quite close to Shi's work [21]. However, we make two more contributions. Firstly, Shi's work only focused on the chain topology DNN models, like AlextNet. We further study the privacy problem under DNN partition strategy on the DAG topology DNN models. Secondly, we introduce an energy-aware mix-precision quantization approach further to reduce the energy use in the edge device. Along with the mix-precision quantization approach, we can reduce the energy consumption of the inference process in the edge device leading to that we can deploy more layers on the device. However, how to find the mix-precision-related split point and keep the model's accuracy become new challenges.

Neural network quantization

As today's DNN becoming complex than ever before, the enormous computation resources they consume have primarily limited applications on edge devices. With the IoT approaching, there is a raising need for DNN compression techniques. Neural network quantization effectively shrinks the model size and reduces energy consumption by quantizing original weights and activations to lower precision without changing model architecture. The challenge of DNN quantization is the trade-off between model accuracy and model performance. Uniform 8-bit quantization has been able to reduce the model sizes by a factor of 4 with insignificant accuracy loss [22]. Moreover, as the structure and importance vary among DNN layers, mixed-precision is adopted to further compress the model and save energy.

Want et al. [12] proposed HAQ, a framework that utilizes reinforcement learning to automatically search for mixed-precision quantization strategies to optimize the latency and energy consumption of neural networks through a hardware simulator. Yuan et al. [23] proposed EvoQ, a mixed-precision strategy based on an evolutionary algorithm to optimize the model bit allocation with limited data. To improve search efficiency, EvoQ analyzes the quantization sensitivity of each layer to boost search efficiency. Rusci et al. [24] propose a memory-driven neural network quantization method to achieve optimal accuracy within the memory constraint.

In our work, we plan to utilize the mix-precision approach to optimize the energy consumption of the edge device. Minimize energy is our primary objective within the mix-precision approach. Besides, we combined the mix-precision with the DNN partition mechanism for privacy protection.

Privacy analysis

In this section, we propose our privacy measurement approach. Then, we fully investigate the influence of the DNN partition strategy and the mix-precision quantization strategy on privacy.

Privacy measurement

The edge device's privacy problems come from three aspects: input content leakages via projected IRs, input content leakages via input reconstruction, and input attribution leakages via model interpretation [7]. In our work, we mainly focus on the first type, namely input content leakages via projected IRs, in which IRs computed out of shallow layers still bestow low-level photographic information of the original inputs. Figure 3 shows an example, making a comparison between IRs from the first five convolution layers of VGG11 with the input image. In order to visualize IRs, the IRs with different sizes and different numbers of channels are processed in the following steps: (1) We take each channel in an IR as a grayscale image and resize them as the same size as the original input image; (2) we calculate the distances between all grayscale images with the original input image; (3) The grayscale image with the minimum distance is selected as our output. From Fig. 3, alone with the inference process, the IR become more and more abstract and hardly be recognized compared with the original input.

To systematically investigate the privacy gain from each IR, we define our privacy measurement function as Eq. 1. In the Equation, x stands for the original input, IR stands for one of intermedia result from the inference process. $\widehat{IR}[i]$ stands for the IR image from the i-th channel after resizing, where $i \in [1, d(IR)]$ and $d(IR)$ denotes the number of channel of the IR. dist$(\cdot)$ denotes the function that calculates the distance between two images. It can be implemented by commonly used image comparison methods such as MSE, SSIM [25], or KL divergence of classification results(KL) [7].

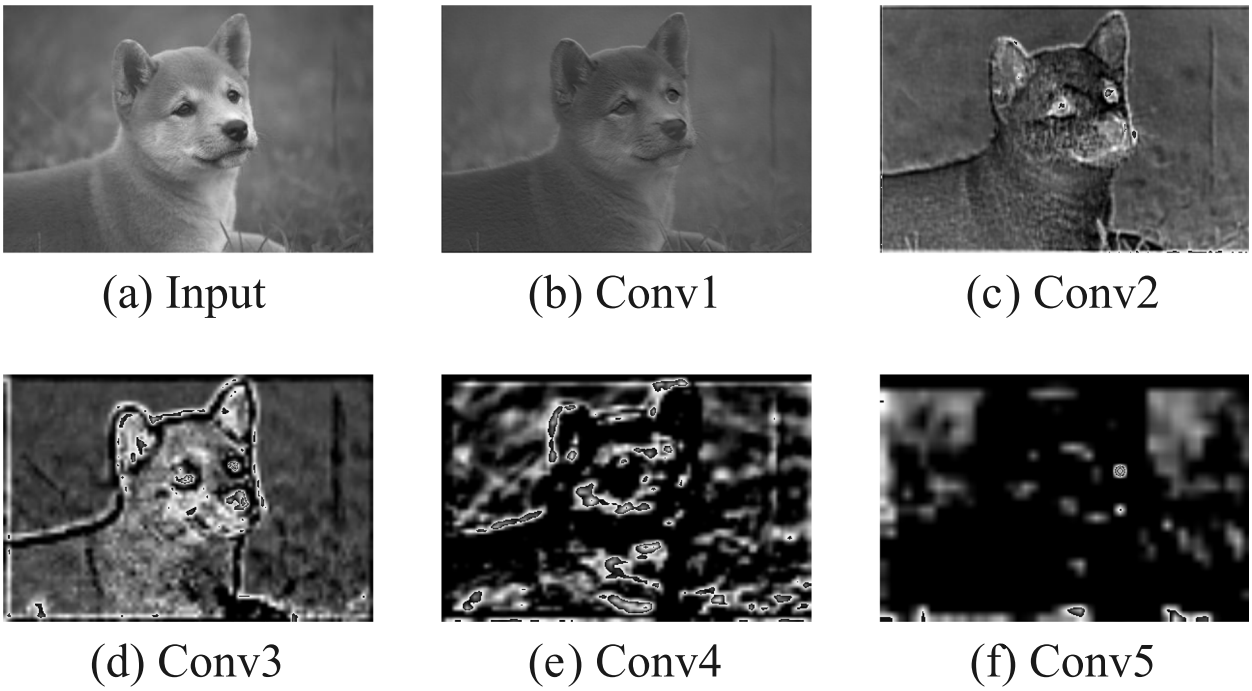

Fig. 3 Comparison between the Input Image with Different IRs

$$P(x, IR) = \min_i \text{dist}(x, \widehat{IR[i]}) \tag{1}$$

To choose the distance function, we evaluate the privacy of each layer's IR in VGG11 with the CIFAR10 dataset using MSE, SSIM, and KL. As shown in Fig. 4, after the value has been normalized, the three measurements all demonstrate a similar trend. We adopt the MSE method, shown as Eq. 2 in the following experiment considering it is the simplest way to compute and it can boost our searching efficiency. In Eq. 2, $\mathbf{x}$ and $\mathbf{y}$ present two images respectively. m and n stand for the width and the height of the picture.

$$\text{MSE}(\mathbf{x}, \mathbf{y}) = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (\mathbf{x}[i][j] - \mathbf{y}[i][j])^2 \tag{2}$$

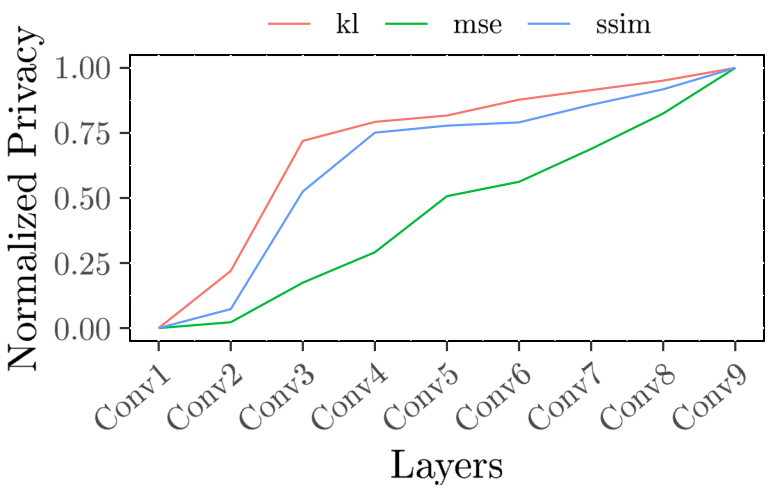

Fig. 4 Privacy changes with Different IRs in VGG11

Privacy with DNN partition

Intuitively, in a traditional chain network structure, the activation becomes more abstract after processing layers and less similar to the original image, thus less risky for privacy leakage. Our experiment mentioned in Fig. 4 also prove this intuition. It is shown from the Fig. 4 that under different measurement methods, there is a regular pattern between IRs and privacy that is as the layer grows, the bigger the privacy value P becomes, indicating we offload more layers to the edge device can better protect privacy. This conclusion is also consistent with the result in [21].

Compared with the chain topology DNNs, the DAG topology DNNs, such as ResNet [26], DenseNet [27] and MobileNets [28, 29]. Same with the chain DNNs, we try to find out the relationship between the IRs and privacy. We experiment on a typical DAG network, ResNet18, with the CIFAR100 dataset. The result is shown in Fig. 5. Different from Fig. 4, the relationship between IRs and privacy is unclear. Like the first two layers, the privacy is much higher along with the layer processing. However, after the Conv5 layer, the privacy fluctuates up and down without a certain pattern.

Unlike the chain DNNs, DAG DNNs are generally composed of a series of "basic block"s. Each basic block combines series layers, and within each basic block, the combination between layers is usually a complex as a non-chained and parallel model. For example, Fig. 6 shows the basic block model from DenseNet, in which the last layer is connected with all previous layers within the same block. Therefore, some layers within a DAG DNN receive all information from the previous layers that cause the fluctuated privacy.

Despite the complex structure within the basic block, the structure between blocks is still simple as a chain structure in DAG DNNs. Therefore, our experiment, shown in Fig. 7, indicates that if we view a DAG DNN as a chain network that consists of a series of basic blocks. Then the relationship between its block-wisely IRs and P is consistent with the above regular pattern.

So we argue that when dealing with DAG DNN, the partition point should be set block-wisely rather than layer-wisely for two reasons:

- It is difficult and inefficient. The output of the layer is often required to be combined with other layers' output in the same block to produce the final output of

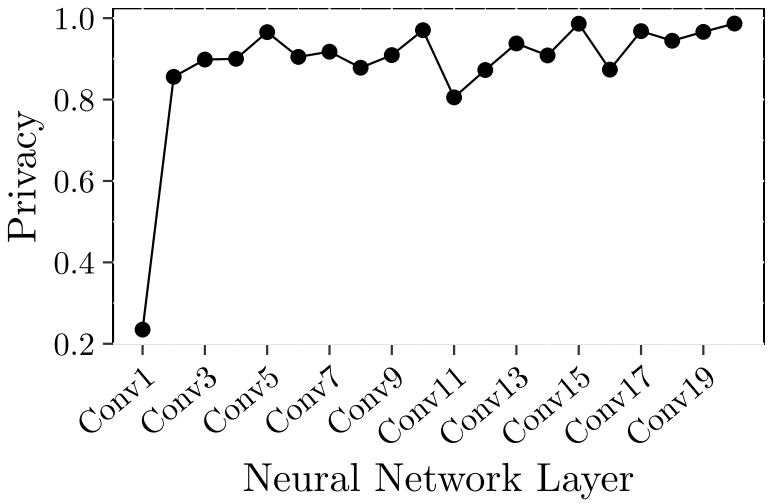

Fig. 5 Privacy changes in Resnet18

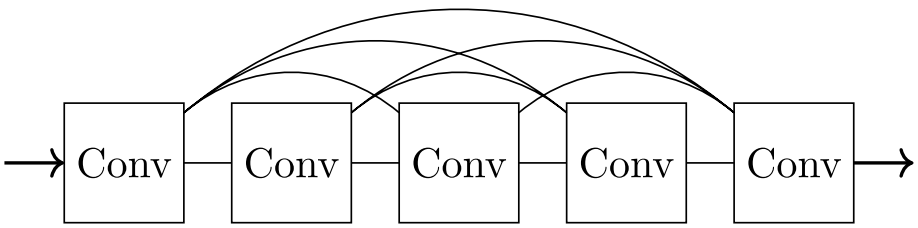

Fig. 6 Basic Block Example from DenseNet

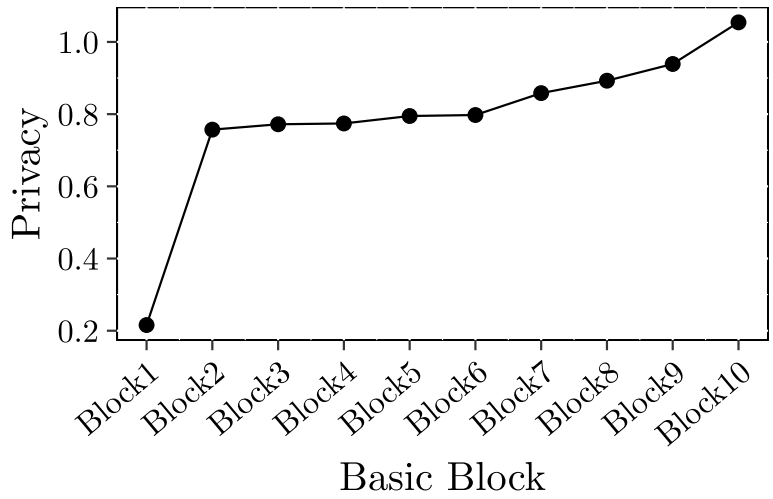

Fig. 7 Privacy changes in Resnet18 with Block-Wise Perspective

the block. Meanwhile, the combination is often too complex to be partitioned, since it can cause huge network transmission cost by sending related IRs.

- Due to the complexity and variety of basic block structure, there haven't been a clear pattern between partition point and privacy to rely on when partitioning between layers within basic block.

Privacy with precision quantization

Since quantization is a method to compress the DNN model by lowering the precision of model weights and activations, it also impacts privacy. Intuitively, the lower precision the weight and activation have, the harder compression and model privacy should be better protected. We randomly pick an intermediate result from VGG11, and quantize it to 2, 4, 8-bit precision, respectively. Later, we calculate the P value. As shown in Fig. 8, the result that consists of our intuition that quantization has a beneficial impact on privacy.

Maximized DNN partition modeling

This section first models the DNN inference process and DNN partition strategy under the mix-precision quantization. Then, we construct our optimization objective and constraints to form a searching problem. At last, we give the measurements of the devices' energy consumption. The notations in this section are summarized in Table 1.

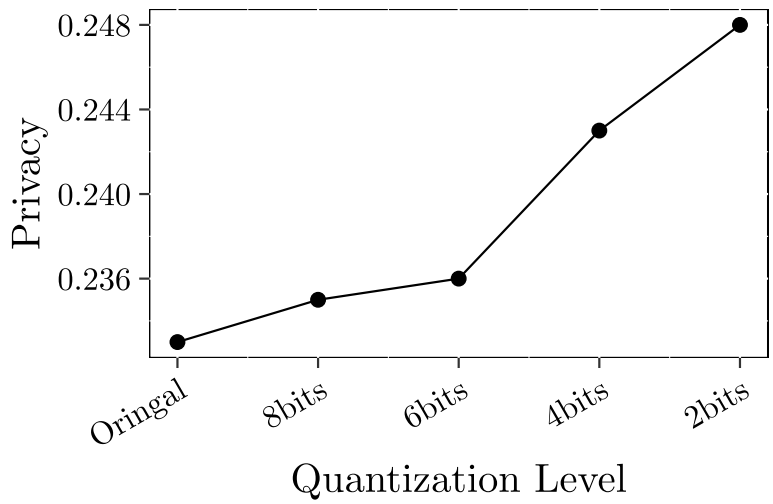

Fig. 8 Privacy changes with the Quantizer

Table 1 Notations

Notation	Description
$\mathbf{M}$	DNN model
$\mathbf{M_{edge}}$	edge part of DNN model
$\mathbf{M_{cloud}}$	cloud part of DNN model
L_i	the ith layer of model
w_i	weight of the ith layer
$\widehat{w_i}$	quantized weight of the ith layer
a_i	activation of the ith layer
$\widehat{a_i}$	quantized activation of the ith layer
Q_b	quantization function with b bit precision
π_i	quantization policy of the ith layer
Π	quantization policy of DNN model
B_i	the ith Block of model
P	privacy evaluation function
p	partition point of DNN model
E	energy measurement of quantized model
E_L	inference energy consumption of one layer
E_T	energy consumption of network transmission
ϵ	quantization loss evaluation function
E_b	energy consumption constraint
ε_b	quantization loss constraint

Without loss of generality, we use computer vision DNN models as our example to construct the modeling. In a typical computer vision scenario, an image is sent as an input to the classifier powered by a convolution neural network (CNN). Later, the classifier returns the prediction value as an output. The process is defined as inference. More specifically, the inference can be considered as the input processed through a series of neural network layers or blocks, which are transformation functions. CNN is a composite of these functions(layers/blocks) that maps the raw input image to prediction output.

Given an input x and a chain CNN with n layers $\mathbf{M} = \{L_1, L_2, ..., L_n\}$, the input is firstly processed by the first layer and produces an activation (also known as IR - intermediate result) $a_1 = L_1(x; w_1)$, where w_1 represent the weights this layer learned from training. Then the activation is the input to the next layer. As for the activation of the ith layer, it can be represented as $a_i = L_i(a_{i-1}; w_i)$, where $i > 1$. If we composite of i to $i-1$ layers, then we have $a_i = L_{1\to i}(x; w_{1\to i}) = L_i L_{i-1} ... L_1(x)$, where $w_{1\to i}$ represents the set of weights from the first layer to the ith layer $\{w_1, w_2, .., w_i\}$.

In a cloud-edge scenario, DNN inference is split and deployed separately in the edge and the cloud. Given a DNN with n layers and a partition point p where $p \in (1, n)$. The $1 \to p$th layers are deployed in the edge and the edge partition model is defined as $\mathbf{M}_{edge} = \{L_1, L_2, ...L_p\}$. $\mathbf{M}_{edge}$ produces a IR as $a_p = L_{1\to p}(x; w_{1\to p})$, which is sent to the cloud though network. The $p+1 \to n$ layers are deplyed in the cloud and the cloud partition model is defined as $\mathbf{M}_{cloud} = \{L_{p+1}, .., L_n\}$. $\mathbf{M}_{cloud}$ produces the final classification result $y = L_{p+1\to n}(a_p; w_{p+1\to n})$ and return it back to the edge.

Due to the constraints of computation capability and energy budget of the edge device, the neural network quantization method can be applied to $\mathbf{M}_{edge}$ for energy-saving and performance-boost. Given a full-precision tensor t, we quantize it to b bit precision. The result can be presented as $\widehat{t}^b = Q_b(t)$ where $Q_b(\cdot)$ represents the quantizer for b bit precision. Given the ith layer of the DNN model L_i, we quantize its weights and activation to b_w and b_a bit precision respectively. Let the input IR be a_{i-1}, the inference process can be presented as Eq. 3

$$\widehat{a}_i = L_i^{\pi_i}(\widehat{a}_{i-1}; \widehat{w}_i) = L_i(Q_{b_a}(a_{i-1}); Q_{b_w}(w_i)) \tag{3}$$

In Eq. 3, $\pi_i = (b_w, b_a)$ is a quantization policy of this layer. The quantized edge model can be represented as $Q(\mathbf{M}_{edge}, ; \Pi) = \{L_1^{\pi_1}, L_2^{\pi_2}, ...L_p^{\pi_n}\}$, where $\Pi = \{\pi_1 ... \pi_p\}$ denotes the collection of quantization policies. Note that in mixed-precision quantization, each π_i may have different value.

Previously we focus on traditional chain DNN. As for DAG DNN that consists of basic blocks B, we first project it into a chain structure $M = \{B_1, B_2, ..., B_n\}$, and, as discussed in Section Privacy with DNN Partition, rather than partition between layers, we make the partition strategy block-wisely. We should also note that even within the same block, the structure, importance to accuracy, and parameter size of each layer are different, so we still make the quantization policy for each layer of DAG DNN.

We aim to find an appropriate partition point and a quantization policy Π to maximize user privacy P while ensuring edge device's energy consumption E and whole model quantization loss ϵ within constraints. The problem is presented in Eq. 4.

$$\begin{aligned} &\textbf{maximize} \quad P(\mathbf{M}, \Pi, p) \\ &\textbf{subjectto} \quad E(\mathbf{M}_{edge}, \Pi) < E_b \\ &\qquad\qquad\ \ \epsilon(\mathbf{M}, \Pi) < \varepsilon_b \end{aligned} \tag{4}$$

In Eq. 4, $P((\mathbf{M}, \Pi, p)$ represents the model's privacy after the partition from p point with the $\mathbf{M}_{edge}$ quantized with policy Π. $E((\mathbf{M}_{edge}, \Pi)$ denotes the energy consumption of the quantized $\mathbf{M}_{edge}$, and $\epsilon(\mathbf{M}_{edge}, \Pi)$ represents the quantization loss. E_b and ε_b denote the energy constraint and loss constraint respectively. We discuss quantizer, privacy evaluation, energy evaluation, and accuracy evaluation in the following sections.

Quantizer

We adopt the linear quantizer, which is more hardware-friendly [22] and quantizes both activation and weight in the edge model, enabling the edge model to do inference using only integer arithmetic which is more efficient for common hardware [30].

Given a weight tensor w, we first truncate it to range $[-c, c]$, and then we quantize it linearly. The quantization process can be described as Eq. 5.

$$\widehat{w}^b = Q_b(w) = \text{round}(\text{clamp}(w, c)/s) \times s \tag{5}$$

In Eq. 5 $\text{clamp}(\cdot, c)$ denotes the truncate function, which truncates range of the $\cdot$ to $[-c, c]$. s stands for the scaling factor. This work adopts the symmetric and uniform quantization method. Given a weight tensor, we use the max absolute value of a given tensor T as the range of the values and map it uniformly to b bit range. Therefore, $c = max(abs(T))$ and s is obtained according to Eq. 6. As for the activation(IR), we apply the similar method but truncate the value to $[0, c]$ rather than $[-c, c]$ since the activation are non-negative.

$$s = \frac{c}{2^{b-1} - 1} \tag{6}$$

Energy measurement

In this work, we intend to maximize the number of layers deployed in the edge to protect privacy while considering the energy use. The energy consumption of the $\mathbf{M}_{edge}$ can be modeled as the sum of each layer and the network transmission cost. Thus given an edge model and quantized it with policy Π, its energy consumption can be represented as Eq. 7, in which $E_L(L_i^{\pi_i})$ denotes the inference energy consumption of layer $L_i^{\pi_i}$, and E_T denotes the network cost to transfer the IR $\widehat{a}_p$ to cloud.

$$E(\mathbf{M}_{edge}, \Pi) = \sum_{i=1}^{p} E_L(L_i^{\pi_i}) + E_T(\widehat{a}_p) \tag{7}$$

We use an analytical method proposed by [31] to measure the inference energy consumption. This allow us to obtain model inference energy without any hardware setup, which has great impact on research efficiency. This method divide the inference energy into two parts: the memory access energy and the multiply-accumulate (MAC) operation energy.

$$E_L = N_{MAC} \times E_{mem|k} + N_{mem} \times E_{MAC|k} \tag{8}$$

Equation 8 gives method to obtain E_L, where N_{MAC} and N_{mem} represent the number of the MAC operations and the memory access respectively. $E_{mem|k}$ and $E_{MAC|k}$ represent the energy consumption of one MAC operation and one memory access on k-bit precision data. Given a layer L, N_{MAC} and N_{mem} can be obtained from the layer's hyper-parametersThe number of memory access N_{mem} is calculated as the sum of input size and parameter size for each layer. For example, in a depthwise convolutional layer, each input channel is convolved with its own set of filters(if size $\frac{out\_channels}{in\_channels}$) and there will be $in\_channels \times out\_channels/in\_channels$ channels total, so the parameter size is $K^2 \times out_{channels}$ and the N_{mem} is $W \times H \times I + K^2 \times O/I \times I$. As for the M_{mac}, because each output channel is calculated through one input channel and a kernel with size $K^2 \times O/I$, so the number of MAC operations of each channel is suppose to be $K^2 \times M^2 \times O/I$, and total N_{mac} of O output channels is $K^2 \times M^2 \times O^2/I$. Typically, for a depthwise convolutional layer, the output channel is same as the input channel, thus above can be simplified to $N_{mac} = K^2 \times M^2 \times O$.

As for network transmission, we assume the network environment is stable. Given a chunk of data d with the size of $M(d)$ and the energy consumption of sending every unit of data by device E_{tran}, the network transmission cost can be represented as Eq. 9.

$$E_T(d) = M(d) \times E_{tran} \tag{9}$$

Due to the limited battery capacity on edge devices, we set the energy constraint on edge model energy consumption as $E_b = \alpha B$, where B denotes the battery capacity and α is the fraction of battery energy that can be used for DNN inference, namely the energy budget.

Quantization loss evaluation

Quantization can bring benefits like reducing model size, accelerating inference and saving energy, but it costs model accuracy [22]. In general, 8-bit quantization only leads to insignificant accuracy loss, but lower precision quantization need finetune to help model recovering from quantization loss. Thus during the search stage, we first finetune the model for one epoch on training set, then evaluate its quantization loss. We think this one turn finetune is necessary for model evaluation because the serve loss introduced by very-low bit quantization will make post training quantization models' accuracy too low to assess. One turn finetune will help a great deal for accuracy recovery without introducing too much cost like fully finetuning, and we notice that this method has been adopt in many NAS-like works [12]. Once the model is quantized and finetuned for one epoch on training set, we evaluate the quantization loss through model accuracy loss directly shown as Eq. 10, where $accur_{origin}$ denotes the model top1 classification accuracy, $accur_{quant}$ denotes the quantized model accuracy after one epoch

finetuning. In order to ensure the quantization loss, we set different loss bound ε_b according to different model architecture and original model accuracy.

$$\epsilon(\mathbf{M}, \Pi) = accur_{origin} - accur_{quant} \tag{10}$$

Searching algorithm

We construct our problem as a searching problem in Eq. 4. We adopt an evolution algorithm to search for the partition point and the quantization policy. Given a DNN model with n layers, we encode the partition point along with the quantization policy into a $n+1$ dimension vector $V = (p, \Pi_n)$, where p stands for the partition point, Π_n denotes a quantization policy with the length of n. After the strategy vector is given, the original model will be split from the p-th layer, and the edge model will be quantized according to the first p elements in Π_n while the $p+1 \rightarrow n$ elements will be masked. Let the quantization bit range be R_w and R_a for weights and activations, then the search space is $O(p \times len(R_w)^p \times len(R_a)^p)$.

We use a classic genetic algorithm to explore the search space automatically. The procedure is summarized in Algorithm 1. A population with N strategy vectors is randomly initialized and kept throughout the search. Each strategy vector is evaluated by a fitness function. In each iteration, S samples are randomly selected from the population. The strategy with the highest fitness in the samples is selected from the population as the parent, and an offspring is generated by mutation operation on the parent with mutating probability *prob*. The strategy with the worst fitness in the samples is excluded from the population, and the offspring is added into the population.

As it is mentioned in Eq. 4, we use the privacy P as the fitness to evaluate each strategy vector. In order to meet the energy and loss constraint, we also measure the energy and the quantization loss for each strategy. To those strategies that do not meet the constraint, we punish their fitness to ensure the final search result satisfies the constraint. The fitness function is defined as Eq. 11.

$$\begin{aligned} F(V, E_b, \epsilon_b) = P(\mathbf{M}, p, \Pi) \\ \times \text{punish}(\lambda_1, E(\mathbf{M}, p, \Pi), E_b) \\ \times \text{punish}(\lambda_2, \epsilon(\mathbf{M}, \Pi), \varepsilon_b) \end{aligned} \tag{11}$$

In Eq. 11, $\text{punish}(\lambda, v, b)$ represents the punishment function, where λ represents the punishment coefficient. We use a exponential punishment function as shown in Eq. 12 where $\lambda \in (0, 1)$ and smaller λ represents more strict punishment. v and b denotes the value and bound respectively. When v is less than b, the function will return 1 with no punishment. When v is greater than b, as shown in Fig. 9, the punishment value decreases gradually to zero as v grows. Comparing to exclude the strategy directly, our method is more suitable for ensuring v to be within b while preserving population diversity.

$$\text{punish}(\lambda, v, b) = \begin{cases} \lambda^{\frac{v}{b}-1} & v > b \\ 1 & \text{otherwise} \end{cases} \tag{12}$$

In Algorithm 1, there will be totally $N+T$ individuals generated and assessed for fitness(N individuals at initiation and T during the search), then the complexity of Algorithm 1. can be represented in big O annotation as $O(N+T)$. However, in experiment, the time cost can largely depend on the original model architecture with the same N and T setup. This is because , as mentioned in Quantization loss evaluation section, we finetune each individual for one turn before assessing their accuracy and this process is the main time cost of Algorithm 1

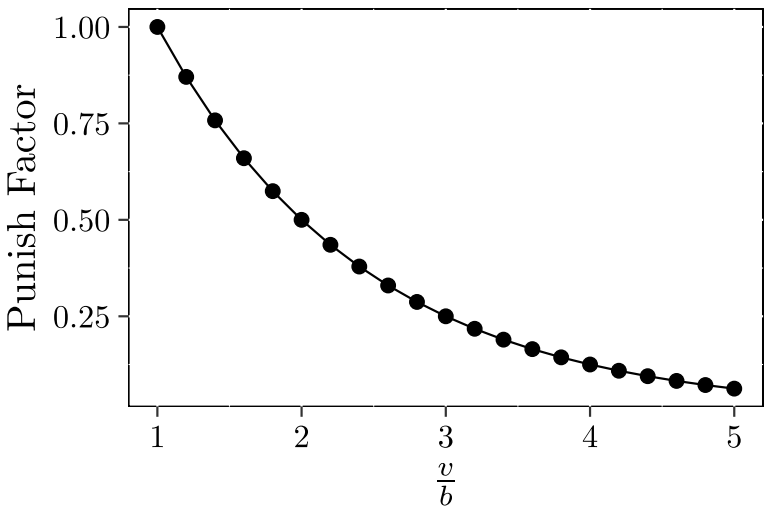

Fig. 9 Punishment Function Example

Table 2 Search Time of 1000 iterations

Model	Search Time
MobileNet	2h
ResNet18	3.4h
ResNet50	12h

in our experiment. Therefore, with the same setup, big deep model architectures on which the one turn finetune process can take long time to finish, will have significant longer search time than those small and compact models, as presented in Table 2.

Require: pre-trained model $\mathbf{M}$, population size N, sample size S, max iteration epochs T, mutation probability $prob$, weight bit range R_w, activation bit range R_a, energy bound E_b, quantization loss bound ϵ_b

population $\leftarrow$ { }
while len(population) $< P$ **do**
  $V(p, \Pi_n) \leftarrow RANDOM\_POLICY(len(\mathbf{M}), R_a, R_w)$
  $Fitness \leftarrow F(V, E_b, \epsilon_b)$
  add V into population
end while
for $i = 1$ to T **do**
  sample $\leftarrow$ { }
  while len(sample)$< S$ **do**
    sample$\leftarrow RANDOM_SAMPLE$(population)
  end while
  $V_{best} \leftarrow$ policy with highest fitness in sample
  $V_{worst} \leftarrow$ policy with lowest fitness in sample
  $V_{child} \leftarrow MUTATE(V_{best}, prob)$
  $Fitness \leftarrow F(V_{child}, E_b, \epsilon_b)$
  add V_{child} into population
end for

Ensure: Quantization policy with highest-fitness Π_{best}

Algorithm 1 Evolution Search

Experiment

We conduct extensive experiments to verify the effectiveness of our method on improving model privacy and reducing model energy consumption.

Environment

Our experiments are performed on the CIFAR100 dataset with Resnet18/50 and MobileNet. We perform per-channel and per-layer quantization on weights and activations. In this work, we only explore the mixed quantization policy on weights $R_w = \{2, 4, 6, 8\}$ and uniform quantized the activation to 8-bit $R_a = \{8\}$. In addition, in order to prevent dramatic accuracy loss, we left the first layer and the full-collection layer unquantized as mentioned in [12]. Inference energy $E_{mem|k}$ and $E_{MAC|k}$ are set and shown in Table 3 according to [31] based on 45nm CMOS devices. Transmission energy E_{tran} is set to 100pJ/b with 1Mbps bandwidth. Energy bound E_b is set to 0.35J which is half of the energy consumption used in [21] with full-precision models. Quantization loss bound ϵ_b is set according to different model architecture to ensure final model accuracy loss less than 3%. In evolution search, the population size $P = 20$, sample size $S = 10$, mutation probability for partition point $prob_p = 0.5$, and for quantization policy $prob_\Pi = 0.3$, punishment coefficient $\lambda_1 = \lambda_2 = 0.1$, and max iterations $T = 1000$. During quantized model finetuning, SGD is used as the optimizer with the learning rate and momentum set to 1e-3 and 0.9 respectively. All search process are performed on a NIVIDIA A1000 GPU with 20G memory.

Table 3 Energy Consumption Estimation Parameters

Operation	Energy(pJ)
k bit Memory access ($E_{Mem|k}$)	2.5k
32 bit MULT INT	3.1
32 bit ADD INT	0.1
k bit MAC INT ($E_{MAC|k}$)	((3.1*k)/32 +0.1)

Methodology

We first evaluate the fitness change in evolution process to prove the convergence of the search algorithm. Then we systematically compare our approach to full-precision, and uniform quantized models in partition point, privacy, and energy. We first evaluate three methods under the same energy bound. Then we make a further comparison by using the same partition point. We prove the advantage of our method using mixed-precision quantization, which can effectively reduce model energy consumption, enable more layer offload to the edge device and preserve privacy. We also make a deep analysis of the AI decided partition point and quantization strategy. Finally, we compare our method to a typical edge-cloud solution.

Experiment results

Table 2 shows the search time of 1000 iterations on MobileNet, ResNet18 and ResNet50. We can see that MobileNet take about 6x time shorter than ResNet50.

Figure 10 shows the evolution process of ResNet18/50 and MobileNet. The privacy value increase rapidly at early iterations and is kept steady with only slight changes. In all three model architecture, the optimization process can be accomplished within 600 iterations. We observe, in general, that aggressive changes are lead by p change and slight changes are lead by Π_n mutation. For example, we can notice in ResNet18 there is a jump on privacy at around the 550th iteration. Before this iteration, the population is dominated by individuals that have

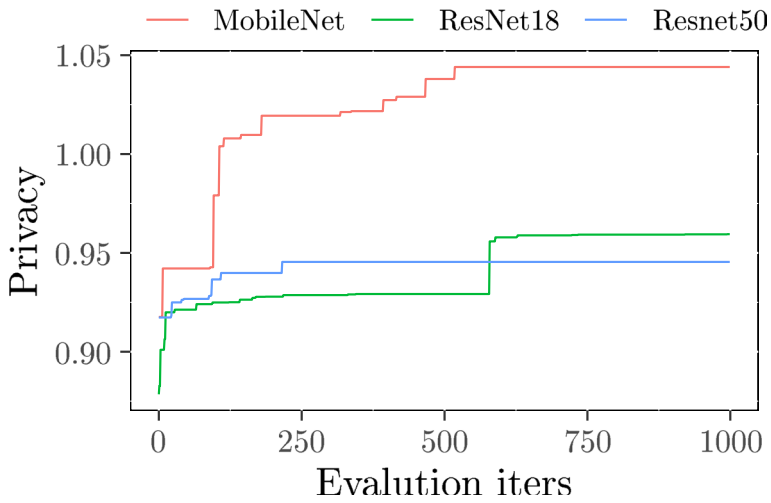

Fig. 10 The search process of ResNet18/50 and MobileNet

Table 4 Edge offloads and Privacy Under E_b

Model	Quantization	Offloads		Privacy	Accuracy (top1)
		#B	*#L*		
ResNet18	full-precision	1	2	0.8045	0.7679
	8-bit Uniq	5	12	0.8877	0.7679
	ours	**7**	**17**	**0.9595**	0.7302
ResNet50	full-precision	1	4	0.8493	0.7812
	8-bit Uniq	4	14	0.8542	0.7807
	ours	**6**	**20**	**0.9454**	0.7642
MobileNet	full-precision	13	24	0.9826	0.6516
	8-bit Uniq	14	26	1.0427	0.6462
	ours	14	26	**1.1102**	0.6207

$p = 6$ and the best privacy increases only sightly due to Π_n changes towards lower precision. This situation lasts about 500 iterations(from 50th to 500th iteration) until a individual with $p = 7$ is generated and while its Π_n is efficient enough to meet the energy bound, therefore, leads a jump on best population privacy.

We compare our method with full-precision and 8-bit uniform quantized model on model accuracy, partition point, and privacy under the same energy bound E_b. In the experiment, we offload as much layers to edge model as the energy can afford for full-precision and uniform quantized model. The result is shown in Table 4. In ResNet18, our method manages to offload more blocks in edge device under the same energy consumption from 1 block(2 conv layers), 5 blocks(12 layers) to 7 blocks(17 layers) increasing privacy by about 20% and 8% with accuracy loss less than 4%. In ResNet50, we observe a similar increase with insignificant model accuracy loss(less than 2%). As for MobileNet which is a compact network designed for mobile devices, our method and 8-bit uniform quantization are both able to offload all conv layers to edge device under energy constraint E_b. However, we still observe a 6% increase in model privacy and 20% reduction on energy of our method, as shown in Fig. 11, with less than 3% accuracy loss.

In order to further demonstrate our advantage that leads by mixed precision quantization, we set the partition point uniformly to the same as our method comparing privacy and energy in ResNet18/50. As shown in Fig. 11, under the same offload blocks, our method reduce the model energy by about 40% and 30% respectively comparing to 8-bit uniform quantized model.

We visualize the quantization strategy of ResNet18 in Fig. 12. We can observe that, in ResNet18, the model is split from the 7th block. The size of its IR is only half the size of the previous block and 1/8 of the first block. Also, we can notice that, within the basic block, extra layers in block3, block5, and block7 are assigned with the lowest bit number(3rd layer in block3, block5, and block7). Intuitively, this is because these layers are used for adjusting channel numbers and have only a few parameters (kernel size 1x1), and are not sensitive to model accuracy.

Table 5 presents the result comparing our method to a typical edge-cloud solution: edge model uniform quantized to 8-bit [22] and model split between convolutional layers and full connection layers [21]. The results show that, in ResNet18/50, our method reduce the energy consumption by about 2x and 5x with trading off about 5% and 1% privacy.

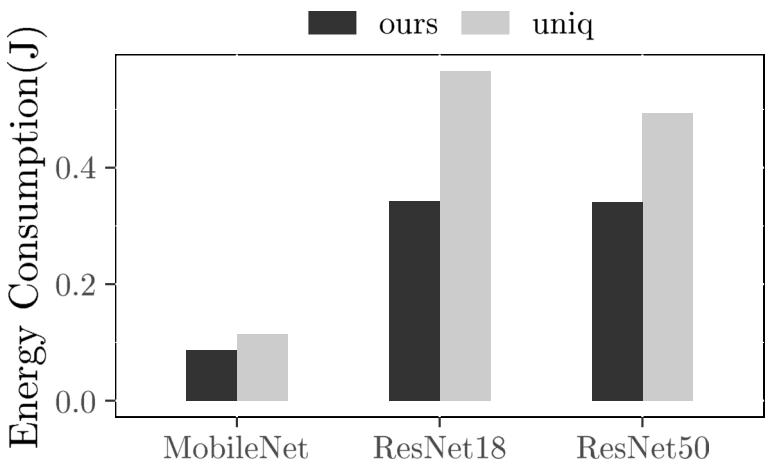

Fig. 11 Energy Under Same Partition Point

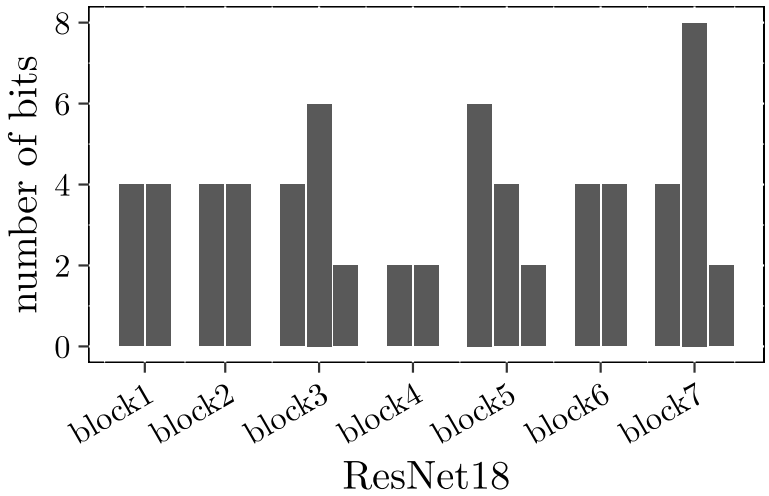

Fig. 12 Partition point and Quantization strategy

Table 5 Comparison between conv/fc partition

Model	Strategy	Offloads		Energy	Privacy
		#B	*#L*		
ResNet18	8-bit conv/fc	8	20	0.7269	1.0068
	ours	7	18	**0.3427**	0.9595
ResNet50	8-bit conv/fc	16	53	1.6776	0.9583
	ours	6	21	**0.3402**	0.9454

Conclusion and further works

In this paper, we propose a privacy protection approach based on a maximum DNN partition strategy. We first explore the relationship between privacy and the number of offload layers in both chain and DAG structure DNN. We observe that privacy increasing with offload layers in chain structure and no clear relationship in DAG structure. In the DAG structure, we further discover that privacy increases with offload blocks and proposes block-wise partition. Secondly, we combine mixed-precision quantization with DNN partition and study its impact on privacy. Last but not least, we adopt an Evolution Search algorithm to search the design space automatically. Experiment results show our method can increase at most 20% privacy in various DNN architectures and reduce energy by at most 5x comparing to the typical edge-cloud solution with only insignificant accuracy loss. For the further works, we point two interesting directions. 1. In the modeling section, we barely consider the stable network connections, and our energy model for network energy is roughly simple, since the transmitted data amount is quite limited since the quantization method. However, unstable network connection might

introduce huge delay and extra energy consumption. It could be investigate deeply in the further work. 2. To make keep simplicity and effective of the work, we level the inference latency out of consideration. However it might cause bad user experience and violation of quality of service. We could introduce the latency measurement in our further work.

Acknowledgements
Not applicable.

Authors' contributions
All authors have participated in conception and design, or analysis and interpretation of this paper. The author(s) read and approved the final manuscript.

Funding
This paper is supported by the National Natural Science Foundation of China under Grant No. 62162050 and the Fundamental Research Funds for the Central Universities No. N2017005.

Availability of data and materials
Not applicable.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 16 October 2021 Accepted: 8 February 2023
Published online: 03 March 2023

References

1. Kumar N, Kaur N, Gupta D (2019) Major Convolutional Neural Networks in Image Classification: A Survey. In: Dutta M, Krishna CR, Kumar R, Kalra M (eds) International Conference on IoT Inclusive Life, vol 116. Springer. NITTTR Chandigarh, India, pp 243–258
2. Liu L, Ouyang W, Wang X, Fieguth P, Chen J, Liu X, Pietikäinen M (2020) Deep Learning for Generic Object Detection: A Survey. Int J Comput Vis 128(2):261–318
3. Otter DW, Medina JR, Kalita JK (2021) A Survey of the Usages of Deep Learning for Natural Language Processing. IEEE Trans Neural Netw Learn Syst 32(2):604–624
4. Miao W, Zeng Z, Wei L, Li S, Jiang C, Zhang Z (2020) Adaptive DNN Partition in Edge Computing Environments. IEEE International Conference on Parallel and Distributed Systems. IEEE, Hong Kong, China, pp 685–690
5. Lecun Y, Bottou L (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
6. Goldstein BF, Patil VC, Ferreira VC, Nery AS, França FMG, Kundu S (2021) Preventing dnn model ip theft via hardware obfuscation. IEEE J Emerg Sel Top Circ Syst 11(2):267–277. https://doi.org/10.1109/JETCAS.2021.3076151
7. Gu Z, Huang H, Zhang J, Su D, Jamjoom H, Lamba A, Pendarakis D, Molloy I (2020) Confidential Inference via Ternary Model Partitioning. arXiv preprint arXiv:1807.00969
8. Blalock DW, Ortiz JJG, Frankle J, Guttag JV (2020) What is the state of neural network pruning? In: Dhillon IS, Papailiopoulos DS, Sze V (eds) Proceedings of Machine Learning and Systems, PMLR, Austin, Texas, pp 129–146
9. Tonellotto N, Gotta A, Nardini FM, Gadler D, Silvestri F (2021) Neural network quantization in federated learning at the edge. Inf Sci 575:417–436
10. Wang X, Che M, Wei Y (2020) Tensor neural network models for tensor singular value decompositions. Comput Optim Appl 75(3):753–777
11. Gou J, Yu B, Maybank SJ, Tao D (2021) Knowledge Distillation: A Survey. Int J Comput Vis 129(6):1789–1819
12. Wang K, Liu Z, Lin Y, Lin J, Han S (2019) HAQ: Hardware-Aware Automated Quantization With Mixed Precision. Conference on Computer Vision and Pattern Recognition. IEEE, Long Beach, California, pp 8604–8612
13. Mao J, Chen X, Nixon KW, Krieger C, Chen Y (2017) MoDNN: Local distributed mobile computing system for Deep Neural Network. In: Design, Automation Test in Europe Conference Exhibition (DATE), IEEE, Lausanne, Switzerland, pp 1396–1401
14. Zhao Z, Barijough KM, Gerstlauer A (2018) DeepThings: Distributed Adaptive Deep Learning Inference on Resource-Constrained IoT Edge Clusters. IEEE Trans Comput-Aided Des Integr Circ Syst 37(11):2348–2359
15. Kang Y, Hauswald J, Gao C, Rovinski A, Mudge T, Mars J, Tang L (2017) Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge. In: Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems, ACM, Xi'an China, pp 615–629
16. Ko JH, Na T, Amir MF, Mukhopadhyay S (2018) Edge-Host Partitioning of Deep Neural Networks with Feature Space Encoding for Resource-Constrained Internet-of-Things Platforms. In: IEEE International Conference on Advanced Video and Signal Based Surveillance, IEEE, pp 1–6
17. Li J, Wang X (2018) Research on the Application of Blockchain in the Traceability System of Agricultural Products. In: IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference, IEEE, Xi'an, pp 2637–2640
18. Li C, Xu H, Xu Y, Wang Z, Huang L (2021) DNN Inference Acceleration with Partitioning and Early Exiting in Edge Computing. In: Liu Z, Wu F, Das SK (eds) International Conference on Wireless Algorithms, Systems, and Applications vol 12937. Springer, Nanjing, pp 465–478
19. Osia SA, Shahin Shamsabadi A, Sajadmanesh S, Taheri A, Katevas K, Rabiee HR, Lane ND, Haddadi H (2020) A Hybrid Deep Learning Architecture for Privacy-Preserving Mobile Analytics. IEEE Internet Things J 7(5):4505–4518
20. Tang Y, Wang Y, Li H, Li X (2021) To cloud or not to cloud: An on-line scheduler for dynamic privacy-protection of deep learning workload on edge devices. CCF Trans High Perform Comput 3(1):85–100
21. Shi C, Chen L, Shen C, Song L, Xu J (2019) Privacy-Aware Edge Computing Based on Adaptive DNN Partitioning. IEEE Global Communications Conference. IEEE, Waikoloa, Hawaii, pp 1–6
22. Krishnamoorthi R (2018) Quantizing deep convolutional networks for efficient inference: A whitepaper. arXiv preprint arXiv:1806.08342
23. Yuan Y, Chen C, Hu X, Peng S (2020) EvoQ: Mixed Precision Quantization of DNNs via Sensitivity Guided Evolutionary Search. 2020 International Joint Conference on Neural Networks (IJCNN). IEEE, Glasgow, pp 1–8
24. Rusci M, Capotondi A, Benini L (2020) Memory-Driven Mixed Low Precision Quantization for Enabling Deep Network Inference on Microcontrollers. In: Machine Learning and Systems 2 (MLSys 2020), Austin, TX, USA, p 1-10
25. Wang Z, Bovik A, Sheikh H, Simoncelli E (2004) Image Quality Assessment: From Error Visibility to Structural Similarity. IEEE Trans Image Process 13(4):600–612
26. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. IEEE Conference on Computer Vision and Pattern Recognition. IEEE, Las Vegas, Nevada, pp 770–778
27. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely Connected Convolutional Networks. IEEE Conference on Computer Vision and Pattern Recognition. IEEE, Honolulu, pp 2261–2269
28. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017) MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. CoRR abs/1704.04861:1–9
29. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC (2018) MobileNetV2: Inverted Residuals and Linear Bottlenecks. 2018 IEEE/CVF

Conference on Computer Vision and Pattern Recognition. IEEE, Salt Lake City, UT, pp 4510–4520
30. Jacob B, Kligys S, Chen B, Zhu M, Tang M, Howard A, Adam H, Kalenichenko D (2018) Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. IEEE, Salt Lake City, pp 2704–2713
31. Panda P (2020) QUANOS: Adversarial noise sensitivity driven hybrid quantization of neural networks. In: Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design, ACM, Boston Massachusetts, pp 187–192

Publisher's Note