Lightweight Convolution Neural Networks for Mobile Edge Computing in Transportation Cyber Physical Systems

Junhao Zhou, Macau University of Science and Technology, Macau SAR
Hong-Ning Dai, Macau University of Science and Technology, Macau SAR
Hao Wang, Norwegian University of Science and Technology

Cloud computing extends Transportation Cyber-Physical Systems (T-CPS) with provision of enhanced computing and storage capability via offloading computing tasks to remote cloud servers. However, cloud computing cannot fulfill the requirements such as low latency and context awareness in T-CPS. The appearance of Mobile Edge Computing (MEC) can overcome the limitations of cloud computing via offloading the computing tasks at edge servers in approximation to users consequently reducing the latency and improving the context awareness. Although MEC has the potential in improving T-CPS, it is incapable of processing computational-intensive tasks such as deep learning algorithms due to the intrinsic storage and computing-capability constraints. Therefore, we design and develop a lightweight deep learning model to support MEC applications in T-CPS. In particular, we put forth a stacked convolutional neural network (CNN) consisting of factorization convolutional layers alternating with compression layers (namely lightweight CNN-FC). Extensive experimental results show that our proposed lightweight CNN-FC can greatly decrease the number of unnecessary parameters thereby reducing the model size while maintaining the high accuracy in contrast to conventional CNN models. In addition, we also evaluate the performance of our proposed model via conducting experiments at a realistic MEC platform. Specifically, experimental results at this MEC platform show that our model can maintain the high accuracy while preserving the portable model size.

Additional Key Words and Phrases: Convolutional neural network, Model Compression, Factorization, Mobile Edge Computing, Cyber Physical Systems, Jetson TX2 Module

1. INTRODUCTION

In recent years, we have witnessed the proliferation of various physical objects connected in a wireless/wired manner to form the Internet of Things (IoT). IoT enables the interactions between the physical environment and computing platforms consequently constructing cyber-physical systems (CPS) [Wang et al. 2018]. During the CPS interaction, learning from massive IoT data is a critical step to extract valuable information so as to make intelligent decisions. The recent advances in machine learning and deep learning bring opportunities in extracting valuable information from massive IoT data. However, many machine learning (especially for deep learning) methods have stringent requirements on computing devices while most of IoT devices do not fulfill these requirements due to the storage and computing limitations. The appearance of cloud computing can overcome the limitations of IoT devices via offloading computing tasks to remote cloud servers [Wang et al. 2019].

Despite the strength in data storage and computing capability, cloud computing cannot fulfill the growing application requirements such as low latency and context awareness. Recently, Mobile Edge Computing (MEC) [Mao et al. 2017], as a complement for cloud computing can potentially overcome the limitations of cloud computing by offloading tasks at edge servers deployed at base stations (BSs), access points (APs) and gateways in approximation to users [Wang et al. 2017b; Wang et al. 2017a]. Take the traffic-sign recognition in Transportation Cyber-Physical Systems (T-CPS) or intelligent transportation system (ITS) [Deka et al. 2018] as an example. Traffic-sign recognition plays an important role in developing T-CPS [Luo et al. 2017]. User requests initiated from mobile devices can be redirected to a nearest edge server (mounted at onboard device in the car) instead of obtaining the results from a remote cloud sever. In this manner, the latency can be greatly reduced and the context awareness can be also improved.

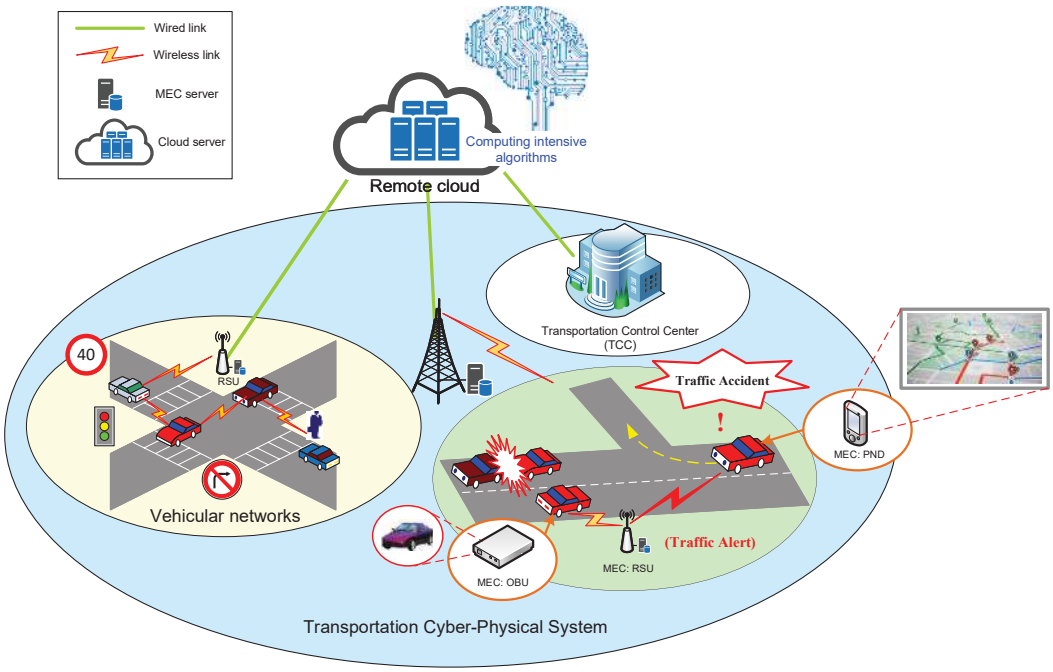


Fig. 1: MEC-Cloud Architecture for T-CPS

1.1. Architecture of MEC-Cloud Architecture for T-CPS

Fig. 1 shows an MEC-Cloud architecture to support T-CPS. This architecture consists of three elements related to MEC deployments: 1) **Road Side Units (RSU) with MEC.** In T-CPS, RSU is an infrastructure node co-located with BSs or APs along the roadside. In the MEC-Cloud architecture for T-CPS, MEC servers can be deployed at RSUs to support the applications of detecting and tracking vehicles, relaying traffic information (such as traffic signs and traffic lights) sent by vehicles. 2) **Portable Navigation Devices (PNDs) or On Board Units (OBUs) with MEC.** A PND is a portable electronic product which combines a positioning capability (acquired by Global Positioning System) and navigation functions. An OBU is a device installed at a vehicle. MEC servers can be deployed at both PNDs and OBUs to collect location information, route structures and traffic flow data consequently offering assistance to drivers. 3) **Transportation Control Center (TCC) with MEC.** The TCC is a supporting system for both PND, OBU and RSU. During the communication process, information is collected and transferred over the TCC. Without the resource limitations, the TCC can provide comprehensive support to make an optimal decision and apply the optimized strategies to the T-CPS.

In such MEC-Cloud architecture for T-CPS, MEC plays a crucial role in offering low-latency and context-aware services to RSUs, vehicles (especially for PNDs and OBUs) and TCC. Several recent studies investigate MEC technologies in T-CPS. For example, [Yang et al. 2017; Contreras-Castillo et al. 2017] show that MEC is an efficient technology to support Internet of Vehicles (IoV). Kaiwartya et al. [Kaiwartya et al. 2016] have proposed a comprehensive IoV five-layer architecture based on the **cloud**, **connections** and **clients**. The work of [Showering 2016] proposed a navigation system based on mobile computing devices.

Nowadays, more and more applications depend on MEC technology, including secure IoT service [Wu et al. 2019], detection of hidden data attacks in sensor-cloud system [Zhang et al. 2018], resource management in smart city systems [Wang et al. 2019]. In particular, vehicles may communicate with other vehicles through the network in real-time when they connect to the distributed edge devices. Therefore, MEC is an efficient technology to support IoV, since the MEC server can be installed in various places at the network edge. Ref. [Datta et al. 2017] proposed an architecture of IoT that considers additional enablers such as edge and cloud platforms, smart-phones and powerful OBUs to support T-CPS.

1.2. Limitations of deep learning models in MEC of T-CPS

However, MEC servers are still incapable of computational-intensive tasks, such as deep learning algorithms due to the storage and computing constraints. For example, deep convolutional neural network (CNN) models as one of the most typical deep learning schemes, show the advantages in learning complicated and hierarchical features of massive image data [Krizhevsky et al. 2012]. The work of [Cirean et al. 2012] proposed a Multi-column deep neural network (MCDNN) structure, which has superior performance than other machine learning models in German Traffic Sign Recognition Benchmark (GTSRB) [Namor et al. 2011]. Meanwhile, other deep CNN models such as AlexNet [Krizhevsky et al. 2012], VGG [Simonyan and Zisserman 2014], GoogLeNet [Szegedy et al. 2015], ResNet [He et al. 2016] also demonstrate the outstanding performance in image classification. Nevertheless, deep CNN models usually contain multiple layers with a large number of parameters. As a result, CNN models typically have a large model size. Moreover, they also require using strong processing devices (e.g., Graphics Processing Units) to train the models. In addition, the large size of CNN models also results in the huge communication overhead in distributed CNN model-training [Iandola et al. 2016]. Therefore, these drawbacks hinder the wide deployment of CNN models in mobile and portable devices in T-CPS, e.g., RSUs, PNDs or OBUs.

1.3. Contributions

In this paper, we design and develop a lightweight CNN model to support MEC applications in T-CPS. Our model has the advantages including much smaller model size than that of conventional CNN models while maintaining high accuracy in traffic-sign and vehicle classification. The main research contributions of this paper are summarized as follows.

— We put forth a stacked convolutional structure consisting of factorization convolutional layers alternating with compression layers. In particular, the factorization convolution converts the conventional convolution into a depthwise convolution and a pointwise convolution consequently reducing the number of unnecessary parameters. Moreover, we further improve the efficiency of the activation function and reduce the redundant parameters by using Concatenated Rectified Linear Units (CReLU) for compression layers. We name the proposed model as lightweight CNN-FC.
— We conduct extensive experiments based on realistic datasets include GTSRB dataset as well as a vehicle dataset (namely VCifar-100). We evaluate the performance of the proposed Lightweight CNN-FC model with the comparison of other representative CNN models including MCDNN [Cirean et al. 2012] model, VGG-16 [Simonyan and Zisserman 2014] and AlexNet [Krizhevsky et al. 2012]. Our model outperforms the conventional models in terms of higher classification accuracy and smaller model size. For example, our model has the model size of 4.9 MB in contrast to 118.8 MB of VGG-16 while maintaining high accuracy (i.e., above 98.9% in GTSRB dataset).

— We also evaluate the performance of our proposed lightweight model in a realistic MEC platform. In particular, the MEC platform is mounted with a Jetson TX2 chipset released by NVIDIA, which is a low-power embedded system with the support of lightweight deep learning schemes. Experimental results also show that our model can obtain high accuracy in the mobile MEC platform.

The remainder of this paper is organized as follows: Section 2 describes related work in this paper. Section 3 presents our model structure. Sections 4 - 6 give the details of the main methods used in our model. Experiment results are presented in Section 7. We conclude and discuss the future work in Section 8.

2. RELATED WORK

We categorize the studies into two categories: 1) traffic sign recognition and vehicle recognition in T-CPS and 2) lightweight approaches.

2.1. Traffic Sign Recognition and Vehicle Recognition in T-CPS

Nowadays, more and more T-CPS applications are applied in real traffic condition. In particular, T-CPS applications mainly focus on traffic sign recognition and they can detect objects of traffic signs [Luo et al. 2017]. Traditional methods for traffic sign recognition are mainly based on various machine learning algorithms including support-vector-machine (SVM) classifiers with local image permutation interval descriptor (LIPID) [Tian et al. 2014] and sparse reprentations [Lu et al. 2012]. It is shown in [Hoferlin and Zimmermann 2009] and [Nguwi and Kouzani 2008] that Multilayer perceptron (MLP) performs high accuracy and achieves low false positive rates during identifying the characters in speed limit signs in [Bargeton et al. 2008]. The work of [Sochor et al. 2016] proposed a scheme that CNN can improve fine-grained vehicle recognition by extracting 3D information of input data sets from video data. In [Zhang et al. 2017], an MEC-based model in a vehicular network was developed to support the computation off-loading process thereby preserving the service continuity in a mobile environment.

Recently, CNN approaches have shown excellent performance in various computer vision (CV) applications such as traffic sign recognition. The work of [Cirean et al. 2012] depicts Multi-column deep neural network (MCDNN) structure and implemented the classification experiments for traffic signs on GTSRB dataset. MCDNN consists of multi standard CNNs architecture, gathers and integrates the results from each CNN. It is shown in [Sermanet and LeCun 2011] that the local and global features can be used for traffic sign recognization consequently improving the performance. The recent advances of CNNs have further promoted the proliferation of CV and T-CPS applications. For example, deeper (more layers) CNN models have been proposed including Alexnet [Krizhevsky et al. 2012] with 8 layers and VGG [Simonyan and Zisserman 2014] with 19 layers and GoogLeNet [Szegedy et al. 2015] with 22 layers. The deeper CNN models can achieve higher accuracy than shallower models [Simonyan and Zisserman 2014].

However, deep CNN models also result in a number of challenges, especially for the mobile applications at IoT nodes and MEC devices. Due to the storage and computing limitations, mobile devices cannot afford the intensive computing taks and the bulky model size of deep CNN models. Therefore, it is necessary to design portable deep learning models to support T-CPS applications.

2.2. Lightweight Approaches

In order to reduce parameters while preserving accuracy in convolutional neural network, a common method is to take an existing CNN model and to simplify its structure.

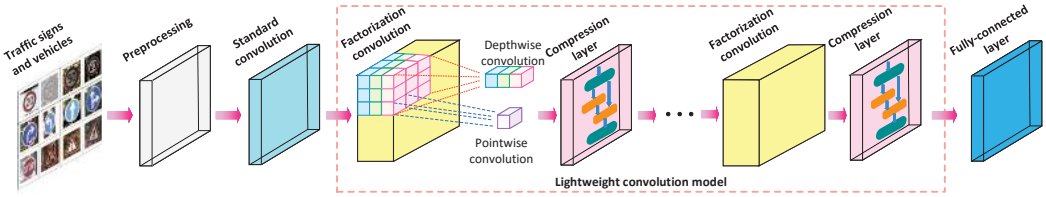


Fig. 2: Lightweight CNN-FC Model consists of factorization convolution layers and compression layers.

A straightforward way is to apply singular value decomposition (SVD) to a pretrained CNN model by [Denton et al. 2014]. Meanwhile, network pruning [Han et al. 2015b] was proposed to replace the parameters with zero to form a sparse matrix via the threshold. Recently, [Han et al. 2015a] creates an approach called Deep Compression combining Network Pruning with quantization (to 8 bits or less) and huffman encoding. In addition, BinaryConnect [Courbariaux et al. 2015], BinaryNet [Courbariaux and Bengio 2016] and XNORNetworks [Rastegari et al. 2016] are effective in quantization CNNs. Furthermore, recently researches accelerate the CNNs via using simpler filters, such as MobileNet [Howard et al. 2017] and SqueezeNet [Iandola et al. 2016]. MobileNet structure uses depth-wise seprable convolutions to construct a lightweight deep neural network. Moreover, MobileNet can be used for a wide range of applications effectively, such as object detection, face attribute extraction and large scale geo-localization. SqueezeNet is a small CNN architecture and and can achieve the equivalent accuracy level to AlexNet. SqueezeNet compresses a neural network with about 50 fewer parameters by replacing 3×3 convolution with 1×1 convolution than conventional CNN.

3. OVERVIEW OF ARCHITECTURE

In this paper, we present a Lightweight CNN-FC model, which consists of two major components: 1) factorization convolutional layer and 2) compression layer. Fig. 2 shows a layout of this model, in which several factorization convolutional layers alternate with compression layers to form a stacked structure. We then briefly describe the working procedure of the Lightweight CNN-FC model.

(1) **Image preprocessing.** The recent study [Buda et al. 2018] shows that the class-imbalance problem in input data sets is detrimental to CNN models. Meanwhile, the traffic-sign data sets such as GTSRB dataset often contain blur, distorted and blemished images, consequently affecting the performance of CNN models. Therefore, we adopt data oversampling and augmentation methods [Chawla 2009] to solve the class-imbalance problem and noisy data.

(2) **Standard convolution.** We choose a standard convolution structure to process the traffic-sign images. The standard convolution structure consists of several convolutional layers, pooling layers and a fully-connected layer. The convolutional input is an $m \times m \times r$ image, where m denotes the height (and also the width) of image and r denotes the number of channels (e.g., $r = 3$ in RGB model because of red, green and blue channels). Meanwhile, we choose b filters, each of which has a size of $n \times n \times q$ in the convolutional layer, where n is typically smaller than the dimension of the input image and q is equal to the number of channels.

(3) **Factorization convolutional layer.** It is a key component in our Lightweight CNN-FC model. In this layer, a conventional convolution is decomposed into a depthwise convolution and a pointwise convolution. Moreover, we also optimize the convolution stride to reduce the computing cost. The number of factorization con-

volutional layers is denoted by α. Details about this structure will be given in Section 5.

(4) **Compression layer.** Another key component in our Lightweight CNN-FC model is the compression layer. We employ a Concatenated Rectified Linear Unit (CReLU) proposed in [Shang et al. 2016] to design a compression convolution filter that can significantly reduce the number of unused parameters. Details about this structure will be introduced in Section 6.

(5) **Fully-connected layer.** We next employ a fully-connected layer which consists of a number of neurons to extract the main features of traffic signs. The calculation procedure is similar to that in the standard convolution layer. In particular, we denote the number of neurons by β, which is tuneable in our experiments.

(6) **Optimizer.** The loss function plays a critical role in prediction accuracy of CNN models. In this paper, we also employ different optimizers to investigate the impacts of them on the loss function. In this experiment, we select 4 optimizers to evaluate the performance of them. The selected optimizers are Stochastic Gradient Desce (SGD), Adagrad, RmsProp and Adam. We present the performance comparison of different optimizers in Section 7.3.4.

(7) **Evaluation on MEC platform.** Finally, we also deploy our proposed lightweight model to the mobile device to support MEC. In particular, we adopt Jetson TX2 formally released by NVIDIA to conduct the experiments. The extensive experiments demonstrate that our proposed lightweight model can provide a practical solution to T-CPS application with high prediction accuracy while maintaining a portable model size.

4. IMAGE PREPROCESSING

First, we need to perform image preprocessing for massive traffic signs and vehicle data sets. In general, raw data sets often have the imbalance class problem [Buda et al. 2018] since the origin data is always raw and crude during experiments or productions. This problem is due to the fact that the number of labels in a class is much larger than that of another class. For example, the number of incurable-disease labels is much smaller than that of normal-disease labels. As a result, the learning model of incurable-disease group causes under-fitting, compared with that of normal-disease group [Wang et al. 2014]. Therefore, it is confirmed that the class imbalance problem causes serious damage in classification mission. It will result in slow convergence in training phase, meanwhile, weaken the generalization ability in test set. In order to solve this problem, the widely used approach is oversampling [Jo and Japkowicz 2004].

In this section, we solve the class imbalance problem by adopting oversampling along with data augmentation. In particular, oversampling directly optimizes the number of samples in each class and averages the class distribution. This approach can solve the class-imbalance problem of datasets. Fig. 3 illustrates the number of samples before (in blue) and after oversampling (in orange) in GTSRB dataset. It is shown in Fig. 3 that the number of samples per class becomes even after sampling.

It is worth mentioning that the effect of data augmentation is essential for solving the class imbalance problem in image preprocessing. In particular, we focus on some specific categories; the numbers of samples in these specific categories are much smaller than those of normal categories. However, just repeating the number of classes simply will lead to overfitting [Chawla et al. 2002]. In order to avoid overfitting, we employ data augmentation when oversampling. Data augmentation is an effective method since it can obtain a number of images with different effects. In data augmentation, we employ 4 different data augmentation methods as follows:

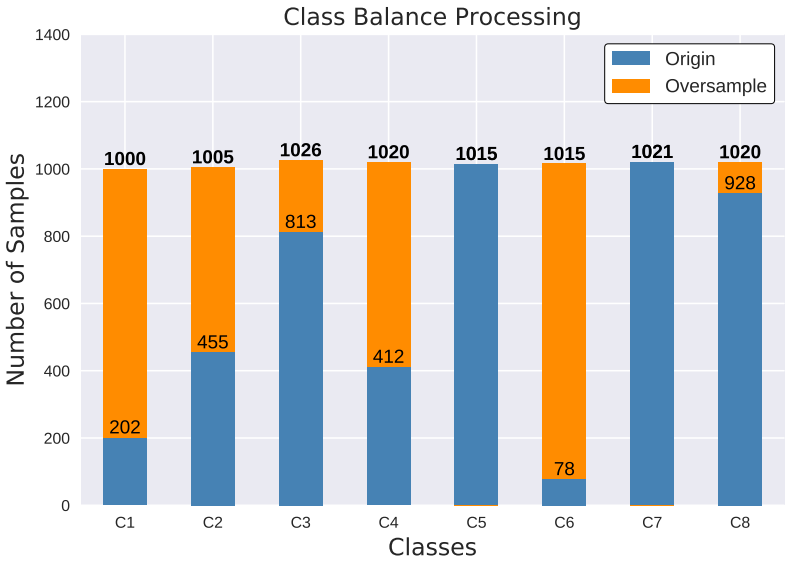


Fig. 3: Effect of oversampling. Origin data and oversampling data are represented in blue and in orange, respectively. For example, the number of samples in Class 4 is 412 before oversampling becomes 1,020 thereby balancing the number of samples in each category.

(1) **Color Augmentation**. Color enhancement includes adjusting color saturation, brightness and contrast;
(2) **PCA Jittering** [Krizhevsky et al. 2012]. The feature value and feature vector can be obtained via calculating the mean and the standard deviation from RGB channels;
(3) **Gaussian Augmentation**. Images are processed with the addition of Gaussian noise;
(4) **Rotation Augmentation** [Szegedy et al. 2015]. Images are rotated within the designated degrees (chosen within 0 to 10 degrees).

Fig. 4 shows the effectiveness of different data augmentation methods.

Fig. 4: From left to right: Origin picture; Color Augmentation processing; PCA Jittering processing; Gaussian Augmentation processing; Rotation Augmentation processing.

As a result, we expand the number of GTSRB dataset after oversampling along with data augmentation. Meanwhile, the major features of each image in the expanded

dataset are still well preserved as before. Therefore, the expanded dataset is beneficial to our model (to be illustrated in Section 7).

5. FACTORIZATION CONVOLUTION

In this section, we describe the structure of factorization convolutional layer. Factorization convolutional layer can reduce the computing cost effectively, thus it can reduce the consumption of computing resources at MEC devices. Fig. 5 depicts the working procedure of the factorization convolutional layer. Unlike the standard convolutional layer, a factorization convolutional layer is decomposed into a depthwise convolution and a pointwise convolution. The depthwise convolution essentially factorizes the standard convolution into M depthwise convolutional filters, each with a size of $D_K \cdot D_K$. The pointwise convolution combines two outputs of depthwise convolution through N pointwise convolutional filters, each with a size of 1×1.

5.1. Computational cost

We next calculate the computational cost of the factorization convolution and evaluate the cost reduction in contrast to the standard convolution operation. The computation costs of depthwise convolutions and pointwise convolutions are denoted by C_{dc} and C_{pc}, respectively. Following the similar steps in MobileNets [Howard et al. 2017], we have C_{dc} as the following equation.

$$C_{dc} = D_K \cdot D_K \cdot M \cdot D_F \cdot D_F, \tag{1}$$

where $D_K \cdot D_K$ represents the convolution kernel size, the output feature map size is $D_F \cdot D_F$ and M is the number of input channels.

Meanwhile, the computation cost of pointwise convolution is calculated by

$$C_{pc} = M \cdot N \cdot D_F \cdot D_F, \tag{2}$$

where N is the number of output channels.

We denote the total cost of factorization convolutional operation by C_f, which consists of two components C_{dc} and C_{pc}. In other words, we have

$$C_f = C_{dc} + C_{pc} = D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F. \tag{3}$$

5.2. Cost reduction

By contrast, we denote the computational cost of the standard convolution operation by C_{std}, which can be calculated by the following equation,

$$C_{std} = D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F. \tag{4}$$

Compared with the standard convolution operation, the factorization convolutional operation can significantly save the computational cost. In particular, we denote the cost-reduction gain of factorization convolutional operation to standard convolution operation by G_r, which is given by the following equation,

$$G_r = \frac{C_f}{C_{std}} = \frac{(D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F)}{(D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F)} = \frac{1}{N} + \frac{1}{D_K^2}. \tag{5}$$

It is shown in Eq.(5) that the cost reduction gain only depends on the number of output channels and the convolution kernel size while it is independent of input size.

6. COMPRESSION LAYER

In our Lightweight CNN-FC model, we also construct compression layer in order to reduce the model size. Therefore, the model can be installed at a small storage

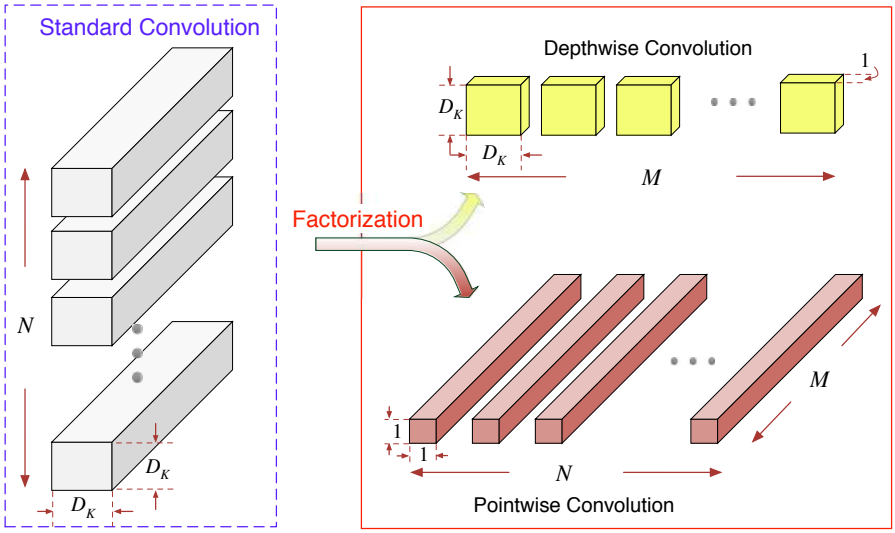


Fig. 5: Factorization Convolution Layout

platform like MEC device. In this layer, it is necessary to utilize a nonlinear activation function to represent and map features after convolution operations. Compared with conventional activation functions such as Sigmoid and Tanh, Rectified Linear Units (ReLU) can effectively mitigate the problem of gradient disappearance [Krizhevsky et al. 2012]. ReLU is defined as:

$$y = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{6}$$

where x denotes input value, and y denotes output of activation function. When $x \leq 0$, $y = 0$; otherwise, $y = x$. Using ReLU activation function can converge the network more quickly. Since ReLU is still not saturate, it can resist the problem of gradient disappearance. Moreover, the computation of ReLU is efficient via using a simple thresholding.

However, the recent study of [Shang et al. 2016] shows that sophisticated CNN models like AlexNet taking ReLU as activation functions may have redundant filters, consequently resulting in the extra and unnecessary computational cost. Concatenated ReLU (CReLU) activation function can overcome the drawbacks of ReLU with a simple but effective modification.

Fig. 6 shows the structure of a compression layer with CReLU activation function. In particular, a negation operation and a concatenation operation are conducted before invoking the ReLU activation function in contrast to the conventional ReLU activation function. Moreover, CReLU can also help to reduce the redundant filters. Specifically, we can compress the CNN model via halving the number of the filters by using CReLU compression layer.

We next analyze the relationship between CReLU and ReLU. We denote a signal by x. The non-negative operation on x can be represented by

$$[x]_+ = \max(x, 0), \tag{7}$$

where $[x]_+$ denotes the non-negative value of x. For example, if $x = 3$, then $[x = 3]_+ = \max(3, 0) = 3$. If $x = -3$, then $[x = -3]_+ = \max(-3, 0) = 0$. The term $[x]_+$ essentially

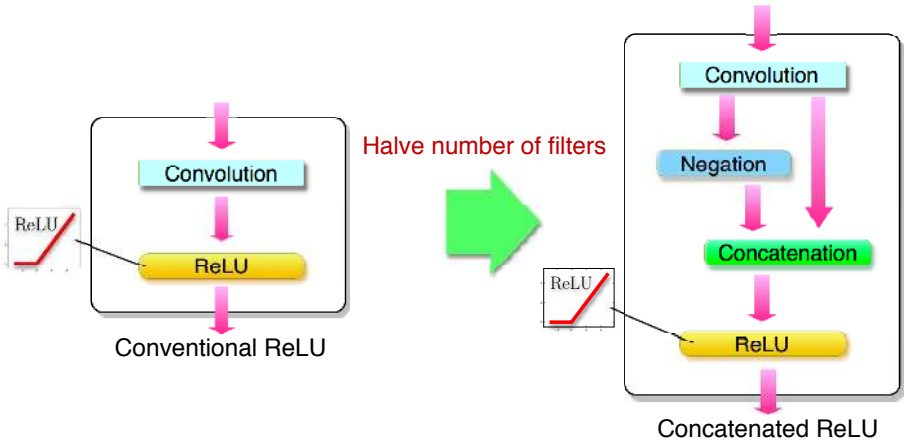


Fig. 6: Concept of CReLU

represents the ReLU activation function, i.e., $\mathsf{ReLU} = [x]_+$. The activation function $\mathsf{CReLU}(x)$ is defined as the following equation,

$$\mathsf{CReLU}(x) = ([x]_+, [-x]_+). \tag{8}$$

Combining Eq.(7) with Eq.(8), the following equation can represent the relation between CReLU and ReLU:

$$\mathsf{CReLU}(x) = [\mathsf{ReLU}(x), \mathsf{ReLU}(-x)]. \tag{9}$$

7. EXPERIMENT

In this section, we conduct the experiments to evaluate the performance of the proposed Lightweight CNN-FC model. We first describe the experimental settings in Section 7.1. We then evaluate the performance of the proposed Lightweight CNN-FC model by comparing with conventional CNN models in Section 7.2. Moreover, we also evaluate the impacts of parameters on the performance of the proposed Lightweight CNN-FC model in Section 7.3.

7.1. Experimental Settings

7.1.1. Experimental Environment. We perform the experiments on two platforms: a PC and an MEC platform (i.e., Jetson TX2 module). Table I gives the detailed configurations of both the PC and the MEC platform. The software framework in our experiments is Keras 2.0 (i.e., Tensorflow as backend) running on Ubuntu 16.04; this setting is the same as for both PC and Jetson TX2 Module.

It is worth mentioning that Jetson TX2 module is mainly designed for MEC with small size and low power consumption. Fig. 7 gives the description of Jetson TX2 module. In particular, this embedded platform features an integrated 256-core NVIDIA Pascal GPU, a hex-core ARMv8 CPU, and 8GB of LPDDR4 memory.

7.1.2. Dataset description. We conduct our experiments on two datasets mainly for T-CPS applications. The first dataset is GTSRB dataset, which has been widely used in evaluating classification algorithms in traffic sign recognition. GTSRB dataset contains more than 50,000 traffic sign images, which have been categorized into 40 classes. We select three major categories: *Speed-limit signs*, *Direction signs* and *Attention signs*. Fig. 8 shows some selected examples from each of the datasets. In addition, we also

Table I: Experimental Environment

PC	
Processor	Intel Core i7-7700HQ
Memory (RAM)	16 GB
Graphics	NVIDIA GTX 1050
Operating Systems	Ubuntu 16.04
Jetson TX2 Module	
Processor	ARM Cortex-A57 + NVIDIA Denver2
Memory (RAM)	8 GB
Graphics	NVIDIA 256-core Pascal
Operating Systems	Ubuntu 16.04

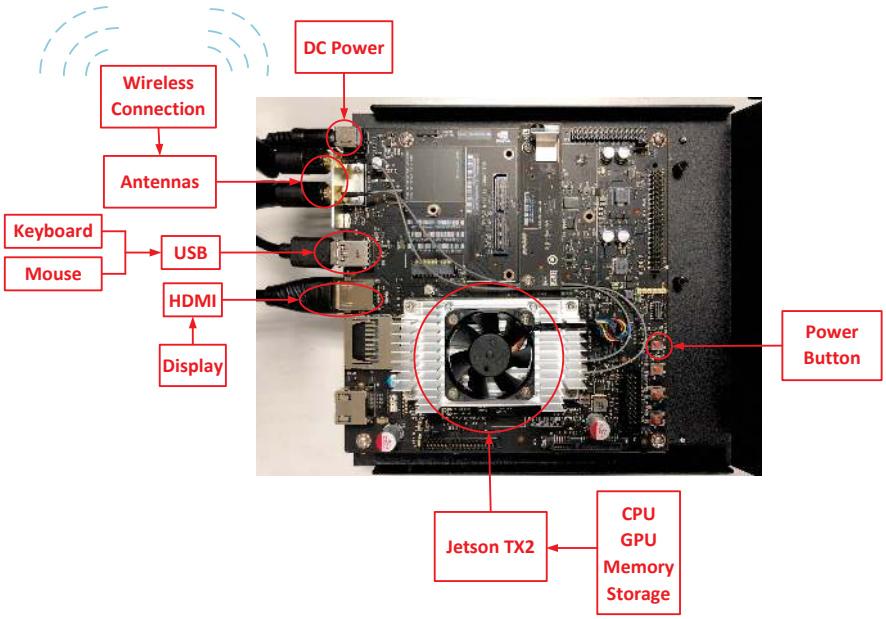


Fig. 7: Detailed specifications of Jetson TX2 Module

need to solve the class-imbalance problem since the number of traffic signs images in each category has significant difference. Therefore, we first preprocess the dataset via the aforementioned oversampling and data augmentation. To simplify our discussion, we name the dataset containing Speed-limit signs as GTSRB-1, the dataset containing Direction signs as GTSRB-2, the dataset containing Attention signs as GTSRB-3 and the dataset containing all the three categories of traffic signs as GTSRB-T (GTSRB Total). After oversampling and augmentation, the exact number of traffic signs in each category is shown in Fig. 8. We can observe that the class-imbalance problem is solved since the number of traffic signs in each category becomes even.

Furthermore, we also construct a vehicle dataset consisting of vehicle images via extracting vehicle images from Cifar-100 dataset, which is a well-established object

Speed-limit signs (GTSRB-1)								
	1000	1005	1025	1020	1015	1015	1020	1020
Direction signs (GTSRB-2)								
	1005	1010	1005	1020	1010	1005	1000	1000
Attention signs (GTSRB-3)								
	1000	1005	1010	1000	1005	1015	10020	1005

Fig. 8: Examples from GTSRB Dataset

recognition dataset that contains 100 classes and each class has 600 images collected by [Krizhevsky 2009]. We name this dataset as VCifar-100 dataset, which contains 5 classes: bicycles, buses, motorcycles, pickup trucks, trains. In addition, VCifar-100 dataset has no class-imbalance problem since the number of samples in each category is identical (i.e., 500). Fig. 9 shows several examples selected from each class of the VCifar-100 dataset.

bicycle	bus	motorcycle	pickup truck	train
				

Fig. 9: Examples from VCifar-100 Dataset

7.1.3. Comparison algorithms. We evaluate the performance of the proposed Lightweight CNN-FC model with other conventional CNN models as described as follows.

MCDNN [Cirean et al. 2012] is a multi-layer CNN model used for GTSRB dataset and performed excellent (won the final phrase in the benchmark of German traffic sign recognition with even better accuracy than human recognition in 2011). This model consists of 6 layers (i.e., 2 convolutional layers, 2 pooling layers and 2 fully-connected layers) .

AlexNet was proposed and developed by Krizhevsky, Sutskever and Hinton [Krizhevsky et al. 2012]. It consists of totally 8 layers: 5 convolutional layers and 3 fully-connected layers. The activation function is ReLU.

VGG-16 was proposed and developed by Simonyan and Zisserman [Simonyan and Zisserman 2014]. This model significantly increases the number of layers in CNN architectures to 16 layers (the 19-layer version is named as VGG-19). It consists of 13 convolutional layers and 3 fully-connected layers.

Factorization-Net is a CNN model with a single factorization convolutional layer. It can be regarded as a special case of our proposed Lightweight CNN-FC model without compression layers.

Table II: Accuracy on GTSRB and VCifar-100 datasets. GTSRB-1: Speed limit signs; GTSRB-2: Direction signs; GTSRB-3: Attention signs; GTSRB-T: GTSRB Total.

Models	Model Size	# of Parameters	Accuracy (GTSRB-1)	Accuracy (GTSRB-2)	Accuracy (GTSRB-3)	Accuracy (GTSRB-T)	Accuracy (VCifar-100)
AlexNet	30.2 MB	3,889,835	95.02%	96.60%	95.53%	96.31%	95.31%
VGG-16	118.8 MB	15,291,499	96.52%	97.29%	97.70%	98.60%	94.99%
MCDNN	19.7 MB	2,466,507	97.95%	97.79%	97.21%	98.50%	95.91%
Factorization Net	6.1 MB	754,373	96.75%	95.61%	94.95%	97.71%	95.11%
Lightweight CNN-FC (PC)	4.9 MB	602,475	98.43%	98.61%	97.91%	98.96%	96.98%
Lightweight CNN-FC (Jetson TX2)			97.14%	97.72%	97.79%	98.15%	97.16%

7.1.4. Performance metrics. We conduct the experiments by considering two performance metrics: *classification accuracy* and *model size*. In particular, the classification accuracy is defined as the ratio of the number of correct classifications to the total number of classifications. To evaluate the model size, we mainly consider the total number of parameters of the trained models and the file size of the trained models (in terms of MB).

7.2. Experimental results

Table II presents the performance comparison of our proposed Lightweight CNN-FC model with other conventional CNN models. It is worth noting that the experiments were conducted on five datasets: GTSRB-1, GTSRB-2, GTSRB-3, GTSRB-T and VCifar-100. In the experiments, we choose the number of the factorization convolutional layers to be $\alpha = 4$ and the number of of neurons in the fully connected layer to be $\beta = 256$. Factorization-Net has the same number of the factorization convolutional layers as our model.

Accuracy. It is shown in Table II that Lightweight CNN-FC model outperforms other existing models in all the five datasets (GTSRB-1, GTSRB-2, GTSRB-3, GTSRB-T and VCifar-100). For example, the accuracy of Lightweight CNN-FC model in GTSRB-T is 98.96%, which is the highest among all the models even though MCDNN and VGG-16 achieve the close accuracy values to our model. Furthermore, we ran Lightweight CNN-FC model on the mobile MEC device (Jetson TX2) and obtain the accuracy on GTSRB-T; the accuracy on TX2 is 98.15%, very close to that of PC. We can observe that accuracy of our model on MEC device is similar to that on PC with different datasets even though PC platform has greater computation capability than MEC device (see Table I). Therefore, our lightweight structure is feasible after factorization and compression procedure. The performance improvement of the proposed Lightweight CNN-FC model may attribute to the excellent characteristics of Lightweight CNN-FC model such as reducing the unnecessary and redundant parameters.

Table III: Comparison of computational cost

Convolution type	Computational cost
Standard convolution	2,359,296
Factorization convolution	**335,872**

Model size. Model size is an important parameter to evaluate the portability of CNN models in MEC deployment. On one hand, the model can be easily loaded on MEC devices when its model size is small enough. On the other hand, a portable CNN model is also beneficial in distributing training in T-CPS. Table II also gives the comparison on the model size between the proposed Lightweight CNN-FC model and other conventional models. It is shown in Table II that Lightweight CNN-FC model has much smaller model size than those of other models. For example, Lightweight CNN-FC model has the file size of 4.9 MB with 602,475 parameters, which is about 6.2$\times$ smaller than that of AlexNet and 24.25$\times$ smaller than that of VGG-16 with high classification accuracy. Lightweight CNN-FC model has even 4.02 $\times$ smaller model size than that of the shallow structure MCDNN model. In summary, after analyzing both the accuracy and the model size, our Lightweight-FC model is advantageous to be deployed at MEC equipment.

7.3. Impacts of parameters

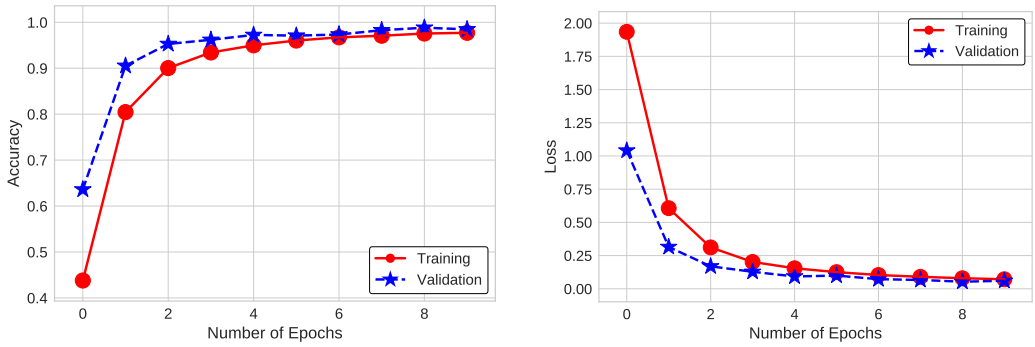
We then investigate the impacts of various parameters on the performance of Lightweight CNN-FC model.

7.3.1. Effect of factorization convolution. In CNN, the computational cost is an important factor reflecting the efficiency of algorithms. We first evaluate the computational cost of the factorization convolution of the proposed Lightweight CNN-FC model with in contrast to that of a standard convolution. In particular, the computational cost of the factorization convolution and that of the standard convolution can be calculated by Eq.(3) and Eq.(4), respectively, as given in Section 5. Specifically, we set parameters $D_K = 3$, $M = 32$, $N = 32$, $D_F = 16$. Table III summarizes the computational costs of the factorization convolution and the standard convolution.

It is shown in Table III that the computational cost of the factorization convolution is much smaller than that of the standard convolution (i.e., it is 7$\times$ cost reduction for factorization convolution). As a result, the model size of the trained model with the factorization convolution can also be greatly reduced. Take Table II as an example. The model with factorization convolution only (i.e., Factorization-Net) has the model size of 6.1 MB, which is much smaller than that of MCDNN (with the standard convolution).

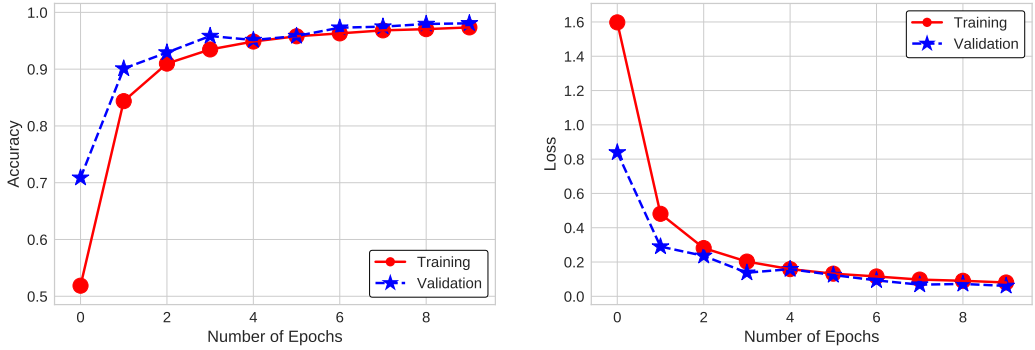
Meanwhile, the factorization convolution does not significantly affect the classification accuracy. Fig. 10 shows the accuracy and loss values of the proposed Lightweight CNN-FC model with factorization layer only (i.e., Factorization-Net). As shown in Fig. 10, both the accuracy and loss values of the model converge after 10 iterations. Moreover, the accuracy of training set is 97.71%, which is just slightly lower than the accuracy of MCDNN model (i.e., 98.50%). We next show that the higher accuracy can be achieved with the combination of compression layer.

7.3.2. Effect of compression layer. We next evaluate the impact of compression layer in our proposed Lightweight CNN-FC model. We add compression layers with CReLU activation function. As shown in Table II, the introduction of compression layers reduces the model size. For example, Lightweight CNN-FC model has the model size of 4.9 MB, which is smaller than that of Factorization-Net (with size of 6.1 MB).



(a) Accuracy (b) Loss

Fig. 10: From left to right: Accuracy and loss of Lightweight CNN-FC model with factorization convolution only running on GTSRB-T. Left: Accuracy of training set is 97.71% and accuracy of validation set is 98.46% after 10 iterations (i.e., converges). Right: Loss of training set is 0.0717 and that of validation set is 0.0603 after 10 iterations.



(a) Accuracy (b) Loss

Fig. 11: From left to right: Accuracy and loss of Lightweight CNN-FC model with both factorization and compression layers running on GTSRB-T dataset. Left: Accuracy of training set is 98.07% and accuracy of validation set is 98.96% after 10 iterations (i.e., converges). Right: Loss of training set is 0.0609 and that of validation set is 0.0603 after 10 iterations.

On the other hand, the compression layers can further improve the classification accuracy. Fig. 11 shows the accuracy and the loss values after the compression layers are added. Compared with the model with factorization layer only (as shown in Fig. 10), adding compression layer can further improve the classification accuracy.

7.3.3. Effect of number of factorization convolutional layers. We then investigate the impact of the number of factorization convolutional layers on the performance. We vary the number of factorization convolutional layers from 1 to 4 (we denote the number of factorization convolutional layers by α). It is worth mentioning that we also need to supplement compression layers between factorization convolutional layers (as shown Fig. 2). The experiments were also conducted on data GTSRB-T and VCifar-100 only.

Table IV: Evaluation with different numbers of factorization convolutional layers

No. of factorization layers	Accuracy (GTSRB-T)	Accuracy (VCifar-100)	Model Size	No. of parameters
$\alpha = 1$	95.25%	87.62%	3.38 MB	434,635
$\alpha = 2$	97.09%	90.06%	3.5 MB	433,221
$\alpha = 3$	97.16%	91.11%	4.3 MB	522,309
$\alpha = 4$	98.96%	96.98%	4.9 MB	602,475

Table V: Evaluation of different optimizers on Lightweight CNN-FC ($\alpha = 1$)

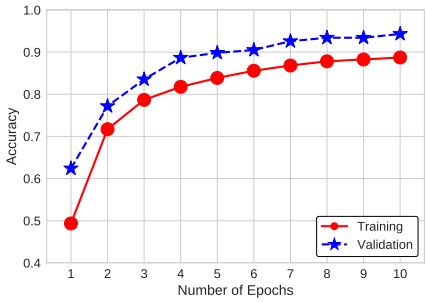
Optimizers	Model Size	Accuracy (GTSRB-T)	# of Parameters
Adagrad	3.38 MB	88.71%	434,635
Adam	5.03 MB	87.73%	434,635
RMSprop	3.38 MB	84.81%	434,635
SGD	**3.38 MB**	**95.08%**	**434,635**

Table IV presents the results. It is shown in Table IV that increasing the number of factorization convolutional layers results in the increment of the classification accuracy while the model size is also increased. Note that the increment of the model size is quite insignificant (e.g., enlarges from 3.5MB to 4.9MB when α increases from 2 to 4). This result implies that the proposed CNN is quite portable and may be used for mobile applications.

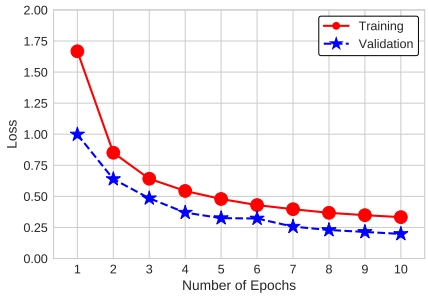
7.3.4. Effect of optimizers. In our lightweight structure, a suitable optimizer can improve the performance during model training and parameter updating. Without loss of generality, our lightweight convolution model uses a single factorization convolutional layer (i.e., $\alpha = 1$) and a compression layer. We then adopt four optimizers to evaluate the performance: Adagrad, Adam, RMSprop, SGD.

Adagrad [Duchi et al. 2011]. Adagrad is a gradient-based optimization algorithm suitable for dealing with sparse data. In our experiment, we adopt the learning rate as the parameter of Adagrad optimizer. Figs. 12 (a) and (b) show the accuracy and the loss of Adagrad, respectively. The experiment results show that the convergence of the loss is pretty slow (see Fig. 12(b)). For example, when the accuracy is only 88.71%, the loss still maintains at 0.3328. Therefore, Adagrad may not be a suitable optimizer in our model.

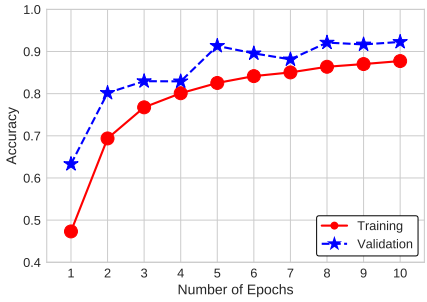
Adam [Kingma and Ba 2014]. Adaptive Moment Estimation (Adam) is an optimized method that computes the adaptive learning rate for each parameter. In the second set of experiments, we evaluate impact of Adam on Lightweight CNN-FC. Figs. 12 (c) and (d) show the accuracy and the loss of Adam. We can find that Adam still results in slower convergence (e.g., when the accuracy of training set is 87.73%, the loss of training set is 0.3610).



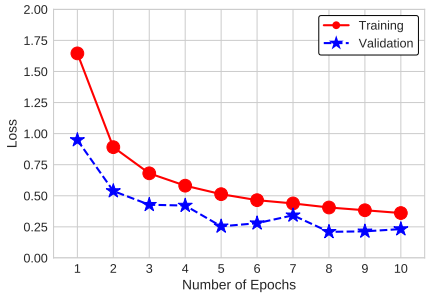
(a) Accuracy of Adagrad



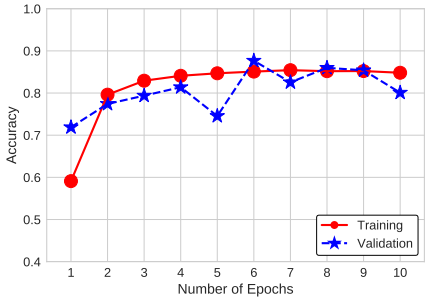
(b) Loss of Adagrad



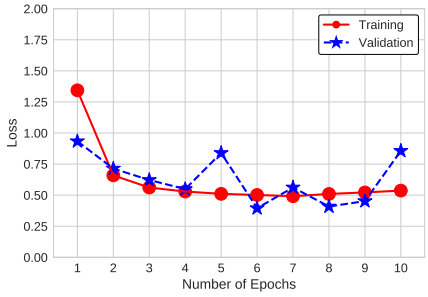
(c) Accuracy of Adam



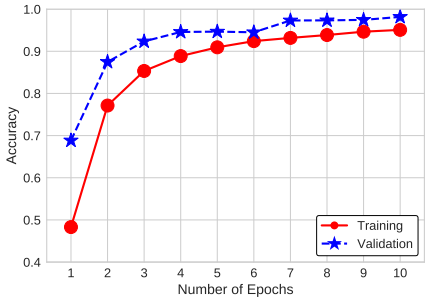
(d) Loss of Adam



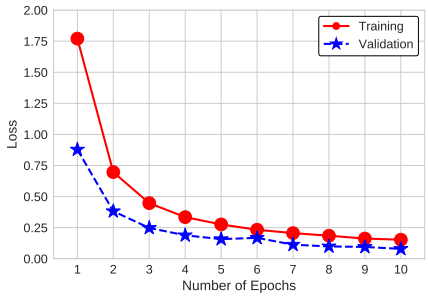
(e) Accuracy of RMSprop



(f) Loss of RMSprop



(g) Accuracy of SGD



(h) Loss of SGD

Fig. 12: Evaluation of Optimizers

Table VI: Evaluation on number of neurons in the fully-connected layer on GTSRB-T)

β	AlexNet	VGG-16	MCDNN	Factorization Net	Lightweight CNN-FC
$\beta = 64$	78.29%	89.34%	96.17%	82.66%	95.97%
$\beta = 128$	93.51%	96.82%	97.85%	96.54%	97.04%
$\beta = 256$	96.31%	98.60%	98.50%	97.11%	98.96%

Table VII: Evaluation on number of neurons in the fully-connected layer on VCifar-100

β	AlexNet	VGG-16	MCDNN	Factorization Net	Lightweight CNN-FC
$\beta = 64$	91.15%	84.78%	91.55%	92.91%	95.67%
$\beta = 128$	93.59%	91.15%	94.71%	93.67%	95.71%
$\beta = 256$	95.31%	94.99%	95.91%	95.11%	96.98%

RMSprop [Tieleman and Hinton 2012]. RMSprop is an adaptive learning rate method proposed by Geoff Hinton. It mainly contributes to resolving radically diminishing learning rates of Adagrad optimizer. Figs. 12 (e) and (f) show that the accuracy and the loss fluctuate radically in training phase. Furthermore, the model cannot converge after 10 epochs.

SGD [Bottou 2010]. SGD is an efficient optimizer since it can eliminate the redundancy of computations for large datasets by performing one update at a time. Figs. 12 (g) and (h) show the accuracy and the loss of the SGD optimizer. We observe from the results that SGD can effectively achieve the convergence after 10 epochs.

Table V also compares the model size, the accuracy, the number of parameters of all the four optimizers. Compared with Adagrad, RMSprop and Adam, we can draw the conclusion from Table V that SGD is the best optimizer in our lightweight CNN-FC model because SGD can achieve the highest accuracy (i.e., 95.08%) and the minimum model size (i.e., 3.38 MB).

7.3.5. Effect of Number of Neurons in Fully-Connected Layer. We also investigate the impact of the number of neurons in the fully-connected layer. Similarly, we conduct the experiments on dataset GTSRB-T and VCifar-100 only. In particular, we denote the number of neurons in the fully-connected layer by β. We vary the values of β from 64 to 256. Meanwhile, we also compare the lightweight CNN-FC with other conventional models when other parameters are fixed.

It is shown in Table VI and Table VII that the proposed Lightweight CNN-FC outperforms other conventional models in terms of the highest accuracy when the number of neurons in the fully-connected layer varies from 64 to 256. Moreover, the highest accuracy is achieved when $\beta = 256$.

8. CONCLUSION

In this paper, we put forth a lightweight convolutional neural network used for MEC in T-CPS. In particular, this model contains a stacked structure, in which several factorization convolutional layers alternate with compression layers. Our model has the merits including the small model size while maintaining high classification accuracy. For example, the proposed Lightweight CNN-FC model with 4 factorization convolu-

tional layers has model size of 4.9 MB, which are much smaller than other conventional CNN models. Meanwhile, the accuracy of the proposed model also outperforms other models. This is mainly because the optimized design on convolution layers and compression layers, consequently removing the redundant parameters. Finally, we also evaluate the performance of the proposed lightweight CNN-FC models by conducting experiments on a realistic MEC platform.

ELECTRONIC APPENDIX

ACKNOWLEDGMENTS

The work is supported by Macao Science and Technology Development Fund under Grant No. 0026/2018/A1, National Natural Science Foundation of China (NFSC) under Grant No. 61672170, NSFC-Guangdong Joint Fund under Grant No. U1401251, the Science and Technology Planning Project of Guangdong Province under Grant No. 2015B090923004 and No. 2017A050501035, Science and Technology Program of Guangzhou under Grant No. 201807010058.

REFERENCES

A. Bargeton, F. Moutarde, F. Nashashibi, and B. Bradai. 2008. Improving pan-European speed-limit signs recognition with a new "global number segmentation" before digit recognition. In *2008 IEEE Intelligent Vehicles Symposium*. 349–354. DOI: http://dx.doi.org/10.1109/IVS.2008.4621168

Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*. Springer, 177–186.

Mateusz Buda, Atsuto Maki, and Maciej A. Mazurowski. 2018. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks* 106 (2018), 249–259.

Nitesh V Chawla. 2009. Data mining for imbalanced datasets: An overview. In *Data mining and knowledge discovery handbook*. Springer, 875–886.

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research* 16, 1 (2002), 321–357.

D Cirean, U Meier, J Masci, and J Schmidhuber. 2012. Multi-column deep neural network for traffic sign classification. *Neural Networks the Official Journal of the International Neural Network Society* 32, 1 (2012), 333–338.

Juan Contreras-Castillo, Sherali Zeadally, and Juan Antonio Guerrero Ibáñez. 2017. A seven-layered model architecture for Internet of Vehicles. *Journal of Information and Telecommunication* 1, 1 (2017), 4–22.

Matthieu Courbariaux and Yoshua Bengio. 2016. BinaryNet: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1. *CoRR* abs/1602.02830 (2016). http://arxiv.org/abs/1602.02830

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. 2015. BinaryConnect: Training Deep Neural Networks with binary weights during propagations. *CoRR* abs/1511.00363 (2015). http://arxiv.org/abs/1511.00363

S. K. Datta, J. Haerri, C. Bonnet, and R. Ferreira Da Costa. 2017. Vehicles as Connected Resources: Opportunities and Challenges for the Future. *IEEE Vehicular Technology Magazine* 12, 2 (June 2017), 26–35. DOI: http://dx.doi.org/10.1109/MVT.2017.2670859

Lipika Deka, Sakib M. Khan, Mashrur Chowdhury, and Nick Ayres. 2018. 1 - Transportation Cyber-Physical System and its importance for future mobility. In *Transportation Cyber-Physical Systems*, Lipika Deka and Mashrur Chowdhury (Eds.). Elsevier, 1 – 20. DOI: http://dx.doi.org/https://doi.org/10.1016/B978-0-12-814295-0.00001-0

Emily Denton, Wojciech Zaremba, Joan Bruna, Yann Lecun, and Rob Fergus. 2014. Exploiting linear structure within convolutional networks for efficient evaluation. In *International Conference on Neural Information Processing Systems*. 1269–1277.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.

Song Han, Huizi Mao, and William J. Dally. 2015a. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *Fiber* 56, 4 (2015), 3–7.

Song Han, Jeff Pool, John Tran, and William J. Dally. 2015b. Learning both Weights and Connections for Efficient Neural Networks. *CoRR* abs/1506.02626 (2015). http://arxiv.org/abs/1506.02626

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

Benjamin Hoferlin and Klaus Zimmermann. 2009. Towards reliable traffic sign recognition. In *2009 IEEE Intelligent Vehicles Symposium*. IEEE, 324–329.

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. (2017). http://arxiv.org/abs/1704.04861

Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and $<$ 0.5MB model size. (2016). http://arxiv.org/abs/1602.07360 cite arxiv:1602.07360Comment: In ICLR Format.

Taeho Jo and Nathalie Japkowicz. 2004. Class imbalances versus small disjuncts. *Acm Sigkdd Explorations Newsletter* 6, 1 (2004), 40–49.

Omprakash Kaiwartya, Abdul Hanan Abdullah, Yue Cao, Ayman Altameem, Mukesh Prasad, Chin-Teng Lin, and Xiulei Liu. 2016. Internet of vehicles: Motivation, layered architecture, network model, challenges, and future aspects. *IEEE Access* 4 (2016), 5356–5373.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Alex Krizhevsky. 2009. *Learning multiple layers of features from tiny images*. Technical Report.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet classification with deep convolutional neural networks. *Communications of the Acm* 60, 2 (2012), 2012.

Ke Lu, Zhengming Ding, and Sam Ge. 2012. Sparse-Representation-Based Graph Embedding for Traffic Sign Recognition. *IEEE Trans. Intelligent Transportation Systems* 13, 4 (2012), 1515–1524.

Hengliang Luo, Yi Yang, Bei Tong, Fuchao Wu, and Bin Fan. 2017. Traffic Sign Recognition Using a Multi-Task Convolutional Neural Network. *IEEE Transactions on Intelligent Transportation Systems* PP, 99 (2017), 1–12.

Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief. 2017. A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Communications Surveys Tutorials* 19, 4 (Fourthquarter 2017), 2322–2358. DOI: http://dx.doi.org/10.1109/COMST.2017.2745201

Angela F. Danil De Namor, Mohammad Shehab, Rasha Khalife, and Ismail Abbas. 2011. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *International Joint Conference on Neural Networks*. 1453–1460.

Yok-Yen Nguwi and Abbas Z Kouzani. 2008. Detection and classification of road signs in natural environments. *Neural computing and applications* 17, 3 (2008), 265–289.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. 2016. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. *CoRR* abs/1603.05279 (2016). http://arxiv.org/abs/1603.05279

Pierre Sermanet and Yann LeCun. 2011. Traffic sign recognition with multi-scale convolutional networks. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*. IEEE, 2809–2813.

Wenling Shang, Kihyuk Sohn, Diogo Almeida, and Honglak Lee. 2016. Understanding and Improving Convolutional Neural Networks via Concatenated Rectified Linear Units.. In *ICML (JMLR Workshop and Conference Proceedings)*, Maria-Florina Balcan and Kilian Q. Weinberger (Eds.), Vol. 48. JMLR.org, 2217–2225.

Paul Edward Showering. 2016. Navigation system configured to integrate motion sensing device inputs. (Aug. 2 2016). US Patent 9,405,011.

Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR* abs/1409.1556 (2014). http://arxiv.org/abs/1409.1556

Jakub Sochor, Adam Herout, and Jiri Havel. 2016. BoxCars: 3D Boxes as CNN Input for Improved Fine-Grained Vehicle Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.

Tian Tian, Ishwar Sethi, and Nilesh Patel. 2014. Traffic sign recognition using a novel permutation-based local image feature. In *Neural Networks (IJCNN), 2014 International Joint Conference on*. IEEE, 947–954.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4, 2 (2012), 26–31.

Kung Jeng Wang, Bunjira Makond, Kun Huang Chen, and Kung Min Wang. 2014. A hybrid classifier combining SMOTE with PSO to estimate 5-year survivability of breast cancer patients. *Applied Soft Computing Journal* 20, 7 (2014), 15–24.

Tian Wang, Yuzhu Liang, Weijia Jia, Muhammad Arif, Anfeng Liu, and Mande Xie. 2019. Coupling resource management based on fog computing in smart city systems. *Journal of Network and Computer Applications* 135 (2019), 11 – 19. DOI:http://dx.doi.org/https://doi.org/10.1016/j.jnca.2019.02.021

Tian Wang, Wenhua Wang, Anfeng Liu, Shaobin Cai, and Jiannong Cao. 2018. Improve the Localization Dependability for Cyber-Physical Applications. *ACM Trans. Cyber-Phys. Syst.* 3, 1, Article 6 (Sept. 2018), 21 pages.

Tian Wang, Jiandian Zeng, Yongxuan Lai, Yiqiao Cai, Hui Tian, Yonghong Chen, and Baowei Wang. 2017b. Data collection from WSNs to the cloud based on mobile Fog elements. *Future Generation Computer Systems* (2017). DOI:http://dx.doi.org/https://doi.org/10.1016/j.future.2017.07.031

T. Wang, G. Zhang, A. Liu, M. Z. A. Bhuiyan, and Q. Jin. 2019. A Secure IoT Service Architecture with an Efficient Balance Dynamics Based on Cloud and Edge Computing. *IEEE Internet of Things Journal* early access (2019), 1–13. DOI:http://dx.doi.org/10.1109/JIOT.2018.2870288

Xiaokang Wang, Laurence T Yang, Xia Xie, Jirong Jin, and M Jamal Deen. 2017a. A cloud-edge computing framework for cyber-physical-social services. *IEEE Communications Magazine* 55, 11 (2017), 80–85.

YouKe Wu, Haiyang Huang, Qun Wu, Anfeng Liu, and Tian Wang. 2019. A risk defense method based on microscopic state prediction with partial information observations in social networks. *J. Parallel and Distrib. Comput.* 131 (2019), 189 – 199. DOI:http://dx.doi.org/https://doi.org/10.1016/j.jpdc.2019.04.007

Fangchun Yang, Jinglin Li, Tao Lei, and Shangguang Wang. 2017. Architecture and key technologies for Internet of Vehicles: a survey. *Journal of Communications and Information Networks* 2, 2 (2017), 1–17.

Guangxue Zhang, Tian Wang, Guojun Wang, Anfeng Liu, and Weijia Jia. 2018. Detection of hidden data attacks combined fog computing and trust evaluation method in sensor-loud system. *Concurrency and Computation: Practice and Experience* (12 2018), e5109. DOI:http://dx.doi.org/10.1002/cpe.5109

Ke Zhang, Yuming Mao, Supeng Leng, Yejun He, and Yan Zhang. 2017. Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading. *IEEE Vehicular Technology Magazine* 12, 2 (2017), 36–44.