An Edge Computing Visual System for Vegetable Categorization

Chang Liu, Xizhe Wang, Jing Ni, Yu Cao, Benyuan Liu
Department of Computer Science
University of Massachusetts Lowell, MA, USA
{chang_liu, xizhe_wang, jing_ni}@student.uml.edu
{ycao, bliu}@cs.uml.edu

***Abstract*—In self-service supermarket and retail industry, efforts to reduce customer wait time using automatic grocery item identification are challenged by low recognition accuracy, long response time and substantial requirement for equipment. In this paper, we propose a novel edge computing system named EdgeVegfru for vegetable and fruit image classification. While existing work on Vegfru dataset shows excellent performance, few of them have been deployed in real-world applications. We adopt an edge computing paradigm, design, implement and evaluate the whole system on the Android devices. The proposed deep learning model and quantization algorithm reduce the model size and inference time significantly. Our system has shown outstanding accuracy within limited time and computation resources, compared with other machine learning methods(such as Support Vector Machine(SVM), Random Forest(RF)), thus providing the potential path for automatic recognition and pricing in self-service retail stores.**

***Index Terms*—convolutional neural network, image classification, deep learning, edge computing**

I. Introduction

Self-service Technologies (SSTs) are those interfaces that allow customers to experience service without directly dealing with the service employees [1]. Emerging technologies in artificial intelligence and computer vision are adopted in self-service facilities to save labor cost and improve the check-out efficiency. However, previous advances in technology is still lagging behind under the strict accuracy and response time constraints. In the retail industry, especially the grocery stores and food markets, the need for high-accurate, low response-time intelligent system is becoming a critical factor to improve customer satisfaction and store profits. In this paper, we focus on developing efficient and accurate algorithm and system for automatic recognition of vegetables and fruits for grocery stores. Our motivation arises from the urgent need of store owners. In real world scenarios, even though many grocery items have been tagged with price, the semi-intervention of store employee is still essential, especially for vegetables and fruits, due to their various packaging and pricing methods. Another factor that delays the process is the strict requirement for grocery item label to align with the electronic scanner during checking out. Most of the customers and newly-hired service employees find it hard to align the price label with electronic scanner. By using the automatic recognition of grocery items, most of the tagging and scanning work can be saved, thus improving the overall efficiency of checking out and reduce human labor cost.

According to the survey [2], the in-store technologies are changing over the years, including evolving transformations, such as handheld scanners, the use of smartphone, artificial intelligence and geofencing technology heralded by Amazon Go [3]. In modern automated retail store like Amazon Go, customers are able to purchase products without being checked out by a cashier or using a self-checkout station [4]. The just-walk-out technology behind such facility involves computer vision, deep learning algorithms and sensor fusion. Under such advanced technology, the system depends heavily on the deployment of multiple sensors, especially various types of cameras. By using images and videos captured by the camera, the deep learning algorithm can accurately predict the product types and quantities. Among all such technologies, convolutional neural networks (CNN) have become the state-of-art method in various computer vision tasks, especially in the field of image classification. Various CNN architectures [5]–[7] have been proposed and shown outstanding performance in public datasets, including ImageNet [8], Microsoft Coco [9] and OpenImage [10]. The success of such CNN-based algorithm and applications relies on the rapid advancement in computing hardware and well-annotated datasets.

Existing datasets mainly consist of general objects and few of them focus on vegetables and fruits. As the urgent need for automatic classification of vegetable and fruit in retail industry increases, researchers have made great efforts to build large-scale, well-annotated fruit datasets. Vegfru [11] is one of the most recent large-scale domain-specific datasets for fruits and vegetables. However, algorithm design and application are still lagging behind the development of dataset.In this paper, we adopt the edge computing paradigm to realize fast and accurate on-device image classification on mobile platforms. Our system is composed of two parts: 1) training deep learning models on the server using a compressed neural network; 2) developing and deploying an application running on mobile devices for fast and accurate inference. We develop an Android app to demonstrate the on-device CNN-based image classification on mobile devices, as depicted in Fig.1. To the best of our knowledge, this is the first attempt to deploy a large-scale CNN network for fruit and vegetable recognition on mobile devices. The main contributions of our paper include:

1) We study and evaluate different CNN architectures and training parameters to get higher accuracy compared with other methods; 2) We use model compression and quantization to optimize the model for mobile and embedded devices; 3) We design and develop an Android app for model inference and image classification, and evaluate the response time, memory consumption and CPU usage on the device. Our extensive experiment results show that our system can achieve excellent recognition accuracy with limited computation resources in a short time frame. Our application has shown great potential in the retail industry and supermarkets.

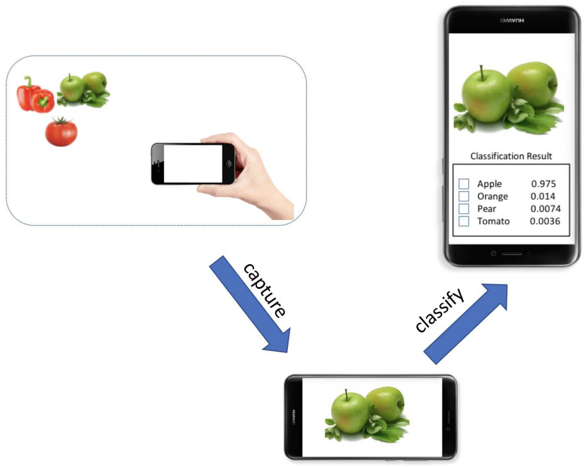

Fig. 1. System demo for classifying vegetable and fruit using mobile device.

II. Background and Related work

There has been rising interest in using CNN-based model to improve the accuracy and reduce the response time for mobile vision applications. Existing work on mobile vision can be divided into three categories: model-based vision application [12], device-based mobile system [13] and edge-based computing paradigms [14].

A. Model-based Application

Recent years have witnessed the explosive advances in deep learning models. With the development of hardware, it's now possible to run CNN models on the mobile devices and lightweight embedded devices [15], [16]. The efforts in this field can be divided into two categories. The first category is to devise novel network architecture that exploit computation and memory efficient operation. Howard et al. [12], [17] proposed MobileNet that utilize the depth-wise and point-wise convolution to build lightweight CNNs and reduce inference time. ShuffleNet [18] used point-wise group convolution and channel shuffle, to greatly reduce computation cost while maintaining accuracy. DenseNet [19] proposed another structure that connects each layer to every other layer in a feed-forward fashion which substantially reduce the number of parameters with high accuracy. The second category is to use compression and quantization techniques to compress CNN models to reduce its resource demands. Jacob et al. [20] proposed a quantization scheme that allows inference to be carried out using integer-only arithmetic, which can be implemented more efficiently than floating point inference on integer-only hardware. Raghuraman et al. [21] presented techniques of quantizing the convolutional neural networks for inference with integer weights and activations. DeepCompression [22] also proposed a three stage processing pipeline: pruning, trained quantization and Huffman coding to reduce the storage and memory constraints.

B. Device-based Mobile System

There has been a significant amount of research going on and plenty of them focus on studying the system performance of deep learning system running on the mobile devices. Previous work on system research focus on the overall evaluation metrics when conducting the recognition tasks, which include accuracy, inference time, response time and power consumption [23]. The studied system can be divided into two categories. The first category focus on studying the combination of cloud-based server and mobile devices [24], [25]. Researchers developed the system using mobile and cloud server together. Chen et al. [24] proposed to use mobile-edge and cloud services for system integration and studied the influence of computation offloading for multiple-users. Kang et al. [26] designed a lightweight scheduler to automatically partition DNN computation between mobile devices and data centers at the granularity of neural network layers to reduce latency and power consumption. The second category focus on developing system without cloud intervention. Keiji and Austin et al. developed a system [13], [27] for food recognition using CNN architecture and running the inference on-device separately. Latifi et al. [28] designed a system called CNNDroid for running CNN models on Android devices with GPU-accelerated execution. The device-based mobile approaches studied the system performance in real-world scenarios and provide complete evaluation and guideline for deployment.

C. Edge-based Computing Paradigms

Edge computing [29] usually refers to the enabling technology that allows computation to be performed at the edge of the network. The "edge" devices can be any computing and network resources along the path between data sources and cloud data center. The data source can be any sensing devices like smartphone, smartwatch, PDA and tablet that collect sensor data like images, audio and video. Cloud data center is equipped with powerful servers that can perform complex computation and data processing. By utilizing the computation ability of edge devices, we can address the critical issues of response time requirement, battery life constraint, bandwidth cost saving, as well as data safety and privacy. The major challenge [25], [30] in applying deep learning algorithms and building visual categorization system is to devise a high-performance mechanism that utilizes the edge computing power for accurate recognition within limited response time and computation resources.

Most of the current research for automatic categorization of vegetable and fruit images are focused on two aspects. The first aspect is to build a large scale, well-annotated high-quality dataset [31]–[33], providing the foundation for designing well-structured deep learning models and neural networks. The second aspect is to design a highly efficient algorithm that utilize the previous dataset and achieve the state-of-art precision or speed. Recent years have witnessed an increase of available datasets. Hou et al. [11] have collected and annotated the largest fine-grained vegetable and fruit dataset VegFru that contains 292 categories. Other efforts in developing such dataset include the Fruits-360 [34], DeepFruit [35], Food-101 [36], UEC-256 [37], but they only contain a small number of categories and images. With the development of well-annotated dataset, the next big challenge for developing accurate algorithms for categorization is to address the inter-class and intra-class differences within the dataset. Recent progress in convolutional neural network [5]–[7], [38] has shown significantly better performance than hand-engineered features (shape, color and texture) [31], [39].

In this paper, we adopt the structure and computing paradigm of edge computing and develop our edge-computing based system based on convolutional neural network. Our efforts are divided into three parts. We first train our customized neural network based on MobileNet [12], [17] which could reduce the computation and inference time. Then we compress and quantize the trained model using tensorflow for mobile deployment. Furthermore, we implement the model inference and develop an Android application to classify the image and video in real time.

III. Proposed Approach

A. *Convolutional Neural Network*

Convolutional Neural Network is widely used in computer vision tasks, such as image classification [40], object detection, and visual question answering. A convolutional neural network is usually composed of convolutional layers, pooling layers, and fully-connected layers. Each layer is connected to the previous layer by predefined, fix-sized kernels. The weights or parameters within each layer are shared to reduce computation complexity. By using carefully designed network architecture, the CNN model learns the parameters from a large-scale dataset to represent the global and local features in the images without using hand-crafted features. Every model has various types of layers and activation function as well as different number of layers and connections that can exhibit strong hidden feature representation ability than human-engineered features. More details of the network structure can be found in [5], [41], [42].

Most of the previous CNN models are designed for desktop computers or servers with reasonable CPU, memory and GPU support. In this paper, we explore various CNN models suitable for mobile devices. While general CNN models such as AlexNet [5], GoogLeNet [7], ResNet [38] show good results for the large benchmark dataset ImageNet, for domain-specific fine-grained dataset VegFru, due to the minor difference between different classes, the models fail to achieve the best performance. Another drawback of such general models is that they contain a large number of parameters, making it unsuitable to be deployed directly on the mobile devices due to the long inference time and large memory footprint. To tackle such problems, we dive deep into the neural network structure and propose several training strategies to achieve the best result with limited resources.

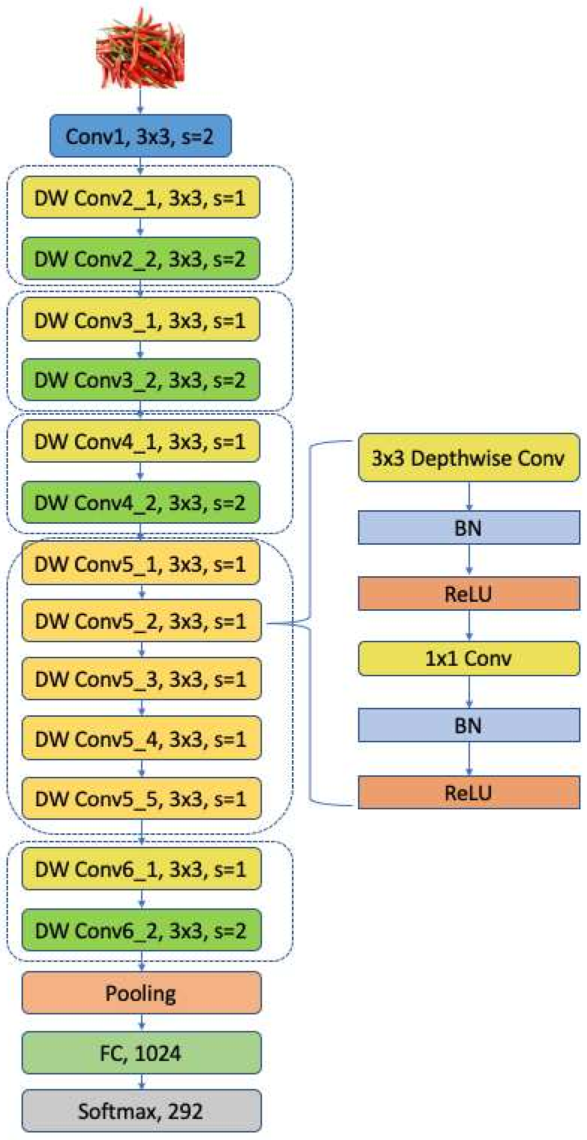

Fig. 2. Proposed method using MobileNet.

MobileNet. MobileNet is a light-weight convolutional neural network that uses depth-wise separable convolutions to build network specifically for mobile and embedded devices. Normally it contains 6 convolutional blocks $conv_1$, $conv_2$, ..., $conv_6$, as shown in Fig.2. Each block contains some 3x3 kernels with a stride size of 1 or 2. $conv_1$ is a convolutional layer with 3x3 kernel and the stride size is 1. For $conv_2$, $conv_3$, $conv_4$ and $conv_6$, they contain repeated units such as two depthwise (DW) convolution layer with a 3x3 kernel and the stride size is 1 and 2 separately. $conv_5$ contains 5 layers of depthwise convolution. The whole network is formed by appending a fully-connected layer and softmax layer to the convolution structure, producing a classification score for each pre-defined class in the dataset. The last softmax layer has a fixed size that equals to the class number. Here we use 292 for the whole Vegfru dataset. A depth-wise separable convolution is a form of factorized convolutions that factorize a standard convolution into a depth-wise convolution and a 1x1 convolution. As illustrated in Fig.2, each unit in the dotted rectangle has similar structure as the right module, which contains a 3x3 depthwise convolution layer and 1x1 pointwise convolutional layer, with one batch normalization (BN) layer and rectified linear unit (ReLU) appended after each convolutional layer in the module. By using this convolution to replace standard convolution, a new network structure is formed as MobileNet.

Transfer Learning. Transfer learning [43], [44] is a popular method in computer vision that helps to build accurate models in a time-saving way. As shown in Fig.3, transfer learning aims to apply the learned knowledge from one domain to another different but related domain. Instead of starting the learning process from scratch, it starts the learning process from previous pretrained model, which usually requires large number of images to learn powerful features, thus reducing the learning time and requirement for large-scale domain-specific dataset. A pretrained CNN model on ImageNet has been widely used for transfer learning, either by using pretrained network as a feature extractor or using it to finetune the whole network. During finetuning, the model can still learn powerful weights to represent image features without too much training data. The reason behind such a training strategy is that the CNN model gains general representation ability from pretrained model on natural images. After finetuning, the model adjusts the parameters to represent the unique features in the target dataset. Experimental results show that transfer learning is very effective on major public datasets [45]. When finetuning a model, there are usually several kinds of strategies to achieve the best result. The first strategy is to train the entire model that updates all the parameters in every layer. This usually requires a large number of training images. Another strategy is to train some layers and leave the others unchanged, or freeze the whole convolutional base and only update the fully-connected layers. This latter strategy is suitable for large dataset with small amount of model parameters. In our experiment, we replace the last softmax classifier layer with our predefined layer and then explore all possible strategies. Our model is pretrained from ImageNet dataset and finetuned on our Vegfru dataset to get the best accuracy using those training strategies to achieve the best result.

B. Model Quantization

Mobile quantization [21], [22] refers to the technique that allows for the reduced precision representation of weights and

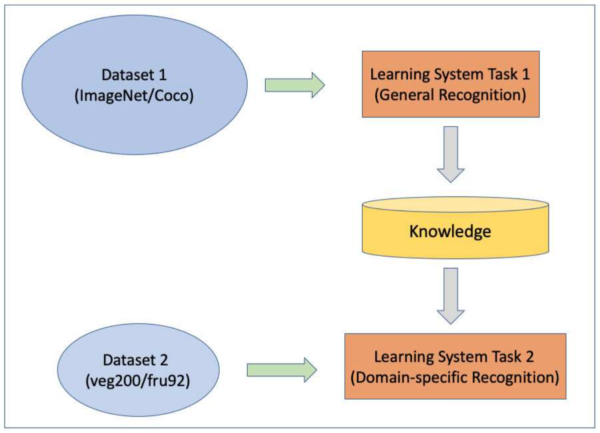

Fig. 3. Transfer learning - training on small domain-specific dataset from pretrained model.

activations for both storage and computation. As a result of quantization, memory access for reading and storing intermediate activations are largely reduced. As shown in Fig.4.

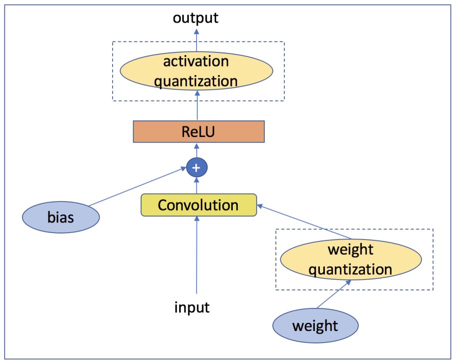

Fig. 4. Model quantization.

To provide practical support for general models, we use Tensorflow [46] to implement our algorithm and neural network. Furthermore, we conduct post-training quantization that quantizes that weights and activations. The post-training quantization quantizes weights to 8-bits of precision from floating point, thus reducing the model size and providing up to 3x speed up with little degradation in model accuracy. As shown in Fig.4, the dotted rectangle depicts the added module for weights and activation quantization. After the quantization, the computation and in-memory storage are significantly reduced. Here we use Tensorflow-lite (TF-Lite) to conduct such a conversion. More specifically, TF-Lite provides built-in models and quantization tools called TOCO to convert trained tensorflow model to a compressed format **.tflite*. Such quantization and on-device inference will use the 8-bit instead of floating point operation on the original model. Our experiment result shows that such a conversion improves the speed significantly.

IV. Experiments

We conduct extensive experiments on the public Vegfru image dataset and follow the standard evaluation process. The model accuracy, memory consumption and response time are compared for several different models.

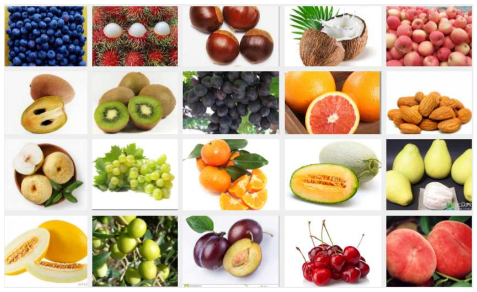
Fig. 5. Fru92 dataset images.

Fig. 6. Veg200 dataset images.

A. Dataset Details

Due to the limitation of publicly available vegetable and fruit datasets, we only conduct our experiment on Vegfru [11] dataset, as it is the first large-scale, well-annotated dataset. This dataset covers vegetables and fruits with 25 upper-level categories and 292 sub-level categories, accounting for more than 160,000 images in total. For vegetables, it has 15 sup-classes with 200 sub-classes and every category has a maximum of 1,807 images and a minimum of 202 images. For fruits, it contains 10 sup-classes and 92 sub-classes and every category has a maximum of 1,615 images and a minimum of 202 images. There are 91,117 images for vegetables and 69,614 images for fruits in total. Fig. 5 and Fig. 6 show the thumbnails for images from the fruit and vegetable category separately. Table I illustrates the characteristics of the dataset, with the number of images for each sub-class varying from 200 to 2,000. Here we also name the vegetables as Veg200 and Fru92 for the fruits. In the following section, we will evaluate the accuracy of our model on these datasets separately.

TABLE I
DATA DISTRIBUTION IN VEGFRU DATASET.

	#Sup	**#Sub**	**#Min**	**#Max**
Vegetables	15	200	202	1807
Fruits	10	92	202	1615

B. Hyperparameter Tuning

We finetune our model based on the pretrained MobileNet model from ImageNet. For the last layer of the MobileNet, we place a softmax layer to get the confidence score for each category. Each score ranges from 0 to 1, representing the probability of classifying each class correctly. When training our MobileNet model, dropout and ReLU are used to address the overfitting and convergence issues. We use Tensorflow and TF-Slim to implement our neural network architecture. The model is trained using stochastic gradient descent (SGD) with different combinations of parameters.

For MobileNet, we choose two hyper-parameters, width multiplier and resolution multiplier. Width multiplier α is defined to thin a network uniformly at each layer. α ranges from 0 to 1 with a typical value of 0.25, 0.5, 0.75 and 1. $\alpha = 1$ is the baseline MobileNet model and smaller α defines thinner models. The resolution multiplier ρ is used to reduce image size. In practice, we usually set the input size implicitly to be 128, 160, 192, 224 for different ρ values. By experiments, we find that using larger image resolution will contribute to the increase of accuracy without introducing too much latency. We use 224x224 as the input image size for training the neural network.

C. Implementation

Our experiment contains two parts: the server side and client side. In the first stage, we train our models in various settings with different parameters and choose the best performer as the final selected model. We train all the models on multiple NVIDIA Tesla K80 GPUs using Tensorflow. During training, the input image is randomly cropped from the original image and resized to a fixed input size with scale. We train all networks using the RMSProp optimizer with a momentum of 0.9, and the batch size of 32. The initial learning rate is 0.045, with exponential decay of 0.98 per epoch. We use batch normalization after every layer, and the standard weight decay is set to be 0.00004.

In the second stage, we convert and quantize the trained-well tensorflow model to TF-Lite format to reduce the model size and shorten inference time. Then we build an Android application that follows the standard testing procedure. An image is first preprocessed before being fed into the neural network. The original image is center cropped and resized to the target input size 224x224. Then we remove the mean value for each pixel afterward. The model inference is implemented in Java on the Android devices using *libtensorflow* and other built-in Android libraries for image preprocessing.

D. Model Evaluation

We classify all the classes for the whole datasets (VegFru, Veg200, Fru92). For each dataset, we evaluate the performance on the super-classes and their sub-classes. To make a fair comparison, we strictly follow the same data splits as in VegFru. For each subclass, the first 100 images are selected as train set, the following 50 images as val set, and the rest as test set. As released in the original dataset, the image lists are used to construct the tfrecords for training our models. All models are evaluated with the *test* set. The results are shown in Table II. Note that due to the missing splits and image lists (Veg200's super-class and Fru92's super-class), the result on such two sets is not available in our experiment.

Table II shows that our proposed MobileNet model can achieve the state-of-art accuracy in all datasets. The classification accuracy on super-class is usually better than on sub-class due to the more significant difference in super-class and the minor intra-class difference in sub-class. This single model is trained in one-stage without any model ensemble, which makes the inference faster and more suitable for mobile devices.

TABLE II
BASELINES ON VEGFRU FOR CNN MODELS. THE TYPICAL CAFFENET, VGGNET, GOOGLENET ARE CHOSEN AS BASELINES WHILE MOBILENET ARE COMPARED WITH THE PREVIOUS STATE-OF-ART RESULTS.

Dataset	Category	CaffeNet	VGGNet	GoogLeNet	MobileNet
Veg200	15 sup-classes	74.92%	83.81%	83.50%	-
	200 sub-classes	67.21%	78.5%	80.17%	**82.26%**
Fru92	10 sup-classes	79.86%	86.81%	87.54%	-
	92 sub-classes	71.60%	79.80%	81.79%	**83.43%**
VegFru292	25 sup-classes	72.87%	82.45%	82.52%	**82.72%**
	292 sub-classes	66.40%	77.12%	79.22%	**81.72%**

Table III shows that our proposed MobileNet model can achieve the state-of-art accuracy in all datasets over other feature-based machine learning methods. Here we use PHOW features as our feature descriptors. As shown in [47], PHOW features are a variant of dense SIFT descriptors extracted from multiple scales. Using the same splits of training and testing dataset as for the CNN-based method, we first build the feature vocabulary, then compute the spatial histograms and generate the bins for feature encoding, and finally generate the feature maps. In the last step, an SVM and random forest classifier are trained based on the feature maps. We train the SVM classifier by using VLfeat's MATLAB imeplementation[1]. Stochastic Gradient Descent (SGD) and Stochastic Dual Coordinate Ascent (SDCA [48]) solver are chosen separately to reach fast convergence and optimize the training process. For random forest, we try different configurations of parameters and find that a maximum number of 100 trees and a maximum depth of 9 generate the best result. Experimental results show that our proposed CNN model achieves best classification accuracy and outperforms these traditional feature based classifiers by a large margin.

E. System Evaluation

In the second stage, we evaluate our system on mobile devices. As shown in Table IV, our model is first converted to a frozen graph file to store the graph definition of the neural network structure. We remove the nodes that are not called during inference, shrink expressions that are constant into single nodes and optimize away some multiplication

[1]http://www.vlfeat.org/overview/svm.html#tut.svm

TABLE III
BASELINE ON VEGFRU FOR OTHER CLASSIFIERS. THE TYPICAL SVM, RANDOM FOREST (RF) ARE CHOSEN AS BASELINES WHILE MOBILENET ARE COMPARED WITH THE PREVIOUS STATE-OF-ART RESULTS.

	Veg200	Fru92	Vegfru292
PHOW+SVM (sgd, [47])	24.54%	26.41%	18.45%
PHOW+SVM (sdca, [48])	28.81%	30.33%	26.06%
PHOW+RF	40.73%	43.21%	38.01%
MobileNet(proposed)	**82.26%**	**83.43%**	**81.72%**

operations during batch normalization by pre-multiplying the weights for convolutions. Then we use *TF-Converter* to convert the frozen graph to TF-Lite format for reducing storage and inference. The result shows that the model size decreases by more than 60% from the original tensorflow model. We also evaluate our MobileNet and Inception-v3 model derived from the GoogLeNet on a UnisCom MZ96-Plus tablet equipped with Intel Z23735F Quadro cores. As shown in Table V, the inference time, storage and memory consumption are significantly reduced compared with other baseline models without sacrificing the classification accuracy. For simple baseline models like AlexNet and VGGNet, our model can outperform them with much less computation resources.

TABLE IV
MODEL SIZE ON VEGFRU. THE TRAINED MODEL IS CONVERTED TO FROZEN GRAPH AND QUANTIZED TO TF-LITE FORMAT.

	Checkpoint	Frozen graph	TF lite
Veg200	29MB	9.8MB	9.5MB
Fru92	27MB	9.2MB	8.9MB
Vegfru292	31MB	11MB	9.9MB

TABLE V
SYSTEM EVALUATION ON MOBILENET AND BASELINE MODEL FOR TIME, STORAGE, MEMORY CONSUMPTION AND CPU UTILIZATION ON MOBILE DEVICES.

Model	Time	Storage	Memory	CPU
Inception-v3(lit)	3291ms	84MB	129MB	41%
Inception-v3(lit-quant)	2890ms	22MB	109MB	33%
MobileNet(lit)	573ms	11MB	74MB	31%
MobileNet(lit-quant)	462ms	9.9MB	67MB	27%

The proposed approach and system has three characteristics: First, the system can provide potential deployment solution for building automatic vegetable and fruit recognition solution in retail store. Second, the algorithm and method presented here utilize memory and computation efficient CNN models for mobile devices that can be applicable to other embedded devices and intelligent systems. Third, the system provides stable service, fast response and high accuracy.

F. Application study

To verify the effectiveness of our proposed system in real application scenarios, we collaborate with supermarkets for deployment. Even though our model improves the accuracy significantly, there is some gap between machine-based vision recognition and human recognition. To overcome such gap, we dive deep into the specific problem and dataset. There are several factors that causes the deterioration of accuracy: 1) the minor difference within the same super class. For example, in the original VegFru292 dataset, there are multiple kinds of mushrooms that belong to different categories. However, in real world scenarios, there are only limited kinds of mushrooms available in specific areas and supermarkets; 2) the random noises in the image dataset, especially for the misclassified label, text occlusion and other objects in the foreground and background; 3) the various angle, environment and exposure in the images crawled from the internet.

To address the aforementioned issues, we explore two kinds of efforts. Firstly, we ask our collaborator in China to collect a new dataset with their smartphone and tablet. Such images are taken in real grocery stores and only contain limited kinds of categories. We use such dataset to train and evaluate our model's accuracy. Such efforts will help to reduce noise and reduce the number of categories, thus improving the overall accuracy, as shown in Table VI. Our dataset is named GroCN20 which contains 20 categories of very common vegetables in grocery stores in China. There are 13,192 images in total, with 80% of them used for training and the rest used as the test set. Our experiment result shows that our model can achieve a very high accuracy with nearly no false classification. Such model and dataset development will help to boost the training accuracy and speed up the deployment in real world application settings.

Second, to study the feasibility in the United States, we visit various grocery stores in the US and choose the corresponding vegetable and fruit categories. Since VegFru292 contains many categories that only appear in specific countries and areas, we manually choose 26 categories of fruits and 46 categories of vegetables that appear in the supermarket Market Basket [2] from the original VegFru dataset. Most of the fruits and vegetables in Market Basket are covered in our new dataset. We name these two datasets as FruMB26 and VegMB46 separately and combine them as VegFruMB72. We use the same approach and strategy to train and evaluate our model. The new dataset still use the 80/20 splits as the training and testing set. As shown in Table VI, our model shows excellent accuracy boost on this new dataset and provides a promising path for future deployment in real grocery stores and supermarkets.

TABLE VI
EVALUATION ON CUSTOMIZED GROCERY DATASET USING PROPOSED APPROACH.

	# TrainSet	# TestSet	# Total	Accuracy
GroCN20	10,554	2,638	13,192	**99.30%**
FruMB26	16,424	4,106	20,530	**97.05%**
VegMB46	24,586	6,146	30,732	**96.30%**
VegFruMB72	41,010	10,252	51,262	**96.52%**

[2]http://www.mygrocerychecklist.com

V. Conclusions and Future work

We designed and implemented an edge computing visual system to classify vegetables and fruits on the domain-specific VegFru dataset. Our work is the first research attempt using CNN in mobile devices for a large vegetable dataset. Based on the experiment result, our system has shown outstanding performance in image categorization with limited memory consumption in a short time frame. The next step is to collaborate with engineering researchers to deploy such a system in retail stores and measure its performance in realistic scenarios. We will also use more labeled region-based information for accurate localization to further improve accuracy and handle extreme cases in real-world settings. Our system has provided a promising path for automatic item categorization in self-service supermarkets.

References

[1] M. L. Meuter, A. L. Ostrom, R. I. Roundtree, and M. J. Bitner, "Self-service technologies: understanding customer satisfaction with technology-based service encounters," *Journal of marketing*, vol. 64, no. 3, pp. 50–64, 2000.

[2] S. Bulmer, J. Elms, and S. Moore, "Exploring the adoption of self-service checkouts and the associated social obligations of shopping practices," *Journal of Retailing and Consumer Services*, vol. 42, pp. 107–116, 2018.

[3] D. Grewal, A. L. Roggeveen, and J. Nordfält, "The future of retailing," *Journal of Retailing*, vol. 93, no. 1, pp. 1–6, 2017.

[4] N. Wingfield, "Amazon moves to cut checkout line, promoting a grab-and-go experience," *The New York Times*, 2016.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[7] C. Szegedy, W. Liu, Y. Jia, Sermanet *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 248–255.

[9] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[10] I. Krasin, T. Duerig, N. Alldrin, A. Veit, S. Abu-El-Haija, S. Belongie, D. Cai, Z. Feng, V. Ferrari, V. Gomes *et al.*, "Openimages: A public dataset for large-scale multi-label and multi-class image classification," *Dataset available from https://github. com/openimages*, vol. 2, no. 6, p. 7, 2016.

[11] S. Hou, Y. Feng, and Z. Wang, "Vegfru: A domain-specific dataset for fine-grained visual categorization," in *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 2017, pp. 541–549.

[12] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2018, pp. 4510–4520.

[13] A. Meyers, N. Johnston, V. Rathod, A. Korattikara, A. Gorban, N. Silberman, S. Guadarrama, G. Papandreou, J. Huang, and K. P. Murphy, "Im2calories: towards an automated mobile vision food diary," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1233–1241.

[14] U. Drolia, K. Guo, J. Tan, R. Gandhi, and P. Narasimhan, "Cachier: Edge-caching for recognition applications," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 276–286.

[15] C. Liu, Y. Cao, M. Alcantara, B. Liu, M. Brunette, J. Peinado, and W. Curioso, "Tx-cnn: Detecting tuberculosis in chest x-ray images using convolutional neural network," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 2314–2318.

[16] M. F. Alcantara, Y. Cao, C. Liu, B. Liu, M. Brunette, N. Zhang, T. Sun, P. Zhang, Q. Chen, Y. Li *et al.*, "Improving tuberculosis diagnostics using deep learning and mobile health technologies among resource-poor communities in peru," *Smart Health*, vol. 1, pp. 66–76, 2017.

[17] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[18] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6848–6856.

[19] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[20] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704–2713.

[21] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," *arXiv preprint arXiv:1806.08342*, 2018.

[22] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[23] N. D. Lane and P. Georgiev, "Can deep learning revolutionize mobile sensing?" in *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*. ACM, 2015, pp. 117–122.

[24] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2015.

[25] C. Liu, Y. Cao, Y. Luo, G. Chen, V. Vokkarane, M. Yunsheng, S. Chen, and P. Hou, "A new deep learning-based food recognition system for dietary assessment on an edge computing service infrastructure," *IEEE Transactions on Services Computing*, vol. 11, no. 2, pp. 249–261, 2018.

[26] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *ACM SIGARCH Computer Architecture News*, vol. 45, no. 1. ACM, 2017, pp. 615–629.

[27] K. Yanai, R. Tanno, and K. Okamoto, "Efficient mobile implementation of a cnn-based object recognition system," in *Proceedings of the 24th ACM international conference on Multimedia*. ACM, 2016, pp. 362–366.

[28] S. S. Latifi Oskouei, H. Golestani, M. Hashemi, and S. Ghiasi, "Cnndroid: Gpu-accelerated execution of trained deep convolutional neural networks on android," in *Proceedings of the 24th ACM international conference on Multimedia*. ACM, 2016, pp. 1201–1205.

[29] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[30] C. Liu, Y. Cao, Y. Luo, G. Chen, V. Vokkarane, and Y. Ma, "Deepfood: Deep learning-based food image recognition for computer-aided dietary assessment," in *International Conference on Smart Homes and Health Telematics*. Springer, 2016, pp. 37–48.

[31] S. R. Dubey and A. S. Jalal, "Fruit and vegetable recognition by fusing colour and texture features of the image using machine learning," *International Journal of Applied Pattern Recognition*, vol. 2, no. 2, pp. 160–181, 2015.

[32] K. Hameed, D. Chai, and A. Rassau, "A comprehensive review of fruit and vegetable classification techniques," *Image and Vision Computing*, vol. 80, pp. 24–44, 2018.

[33] Y. Sakai, T. Oda, M. Ikeda, and L. Barolli, "A vegetable category recognition system using deep neural network," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2016 10th International Conference on*. IEEE, 2016, pp. 189–192.

[34] H. Mureşan and M. Oltean, "Fruit recognition from images using deep learning," *Acta Universitatis Sapientiae, Informatica*, vol. 10, no. 1, pp. 26–42, 2018.

[35] I. Sa, Z. Ge, F. Dayoub, B. Upcroft, T. Perez, and C. McCool, "Deepfruits: A fruit detection system using deep neural networks," *Sensors*, vol. 16, no. 8, p. 1222, 2016.

[36] L. Bossard, M. Guillaumin, and L. Van Gool, "Food-101–mining discriminative components with random forests," in *European Conference on Computer Vision*. Springer, 2014, pp. 446–461.

[37] Y. Kawano and K. Yanai, "Automatic expansion of a food image dataset leveraging existing categories with domain adaptation," in *Proc. of ECCV Workshop on Transferring and Adapting Source Knowledge in Computer Vision (TASK-CV)*, 2014.

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[39] H. M. Zawbaa, M. Abbass, M. Hazman, and A. E. Hassenian, "Automatic fruit image recognition system based on shape and color features," in *International Conference on Advanced Machine Learning Technologies and Applications*. Springer, 2014, pp. 278–290.

[40] Y. Cao, C. Liu, B. Liu, M. J. Brunette, N. Zhang, T. Sun, P. Zhang, J. Peinado, E. S. Garavito, L. L. Garcia *et al.*, "Improving tuberculosis diagnostics using deep learning and mobile health technologies among resource-poor and marginalized communities," in *2016 IEEE First International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*. IEEE, 2016, pp. 274–281.

[41] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.

[42] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[43] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural computation*, vol. 29, no. 9, pp. 2352–2449, 2017.

[44] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[45] C. S. A. H. S. B. Yin Cui, Yang Song, "Large scale fine-grained categorization and domain-specific transfer learning," in *CVPR*, 2018.

[46] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[47] A. Bosch, A. Zisserman, and X. Munoz, "Image classification using random forests and ferns," in *2007 IEEE 11th international conference on computer vision*. Ieee, 2007, pp. 1–8.

[48] S. Shalev-Shwartz and T. Zhang, "Stochastic dual coordinate ascent methods for regularized loss minimization," *Journal of Machine Learning Research*, vol. 14, no. Feb, pp. 567–599, 2013.