MobileDeepPill: A Small-Footprint Mobile Deep Learning System for Recognizing Unconstrained Pill Images

Xiao Zeng, Kai Cao, Mi Zhang
Michigan State University

ABSTRACT

Correct identification of prescription pills based on their visual appearance is a key step required to assure patient safety and facilitate more effective patient care. With the availability of high-quality cameras and computational power on smartphones, it is possible and helpful to identify unknown prescription pills using smartphones. Towards this goal, in 2016, the U.S. National Library of Medicine (NLM) of the National Institutes of Health (NIH) announced a nationwide competition, calling for the creation of a mobile vision system that can recognize pills automatically from a mobile phone picture under unconstrained real-world settings. In this paper, we present the design and evaluation of such mobile pill image recognition system called *MobileDeepPill*. The development of MobileDeepPill involves three key innovations: a triplet loss function which attains invariances to real-world noisiness that deteriorates the quality of pill images taken by mobile phones; a multi-CNNs model that collectively captures the shape, color and imprints characteristics of the pills; and a Knowledge Distillation-based deep model compression framework that significantly reduces the size of the multi-CNNs model without deteriorating its recognition performance. Our deep learning-based pill image recognition algorithm wins the First Prize (champion) of the NIH NLM Pill Image Recognition Challenge. Given its promising performance, we believe MobileDeepPill helps NIH tackle a critical problem with significant societal impact and will benefit millions of healthcare personnel and the general public.

CCS Concepts

•Human-centered computing → Ubiquitous and mobile computing; •Computing methodologies → Visual content-based indexing and retrieval;

Keywords

Mobile Deep Learning Systems; Unconstrained Pill Image Recognition; Deep Neural Network Model Compression; Mobile Health

MobiSys'17, June 19–23, 2017, Niagara Falls, NY, USA.

DOI: http://dx.doi.org/10.1145/3081333.3081336

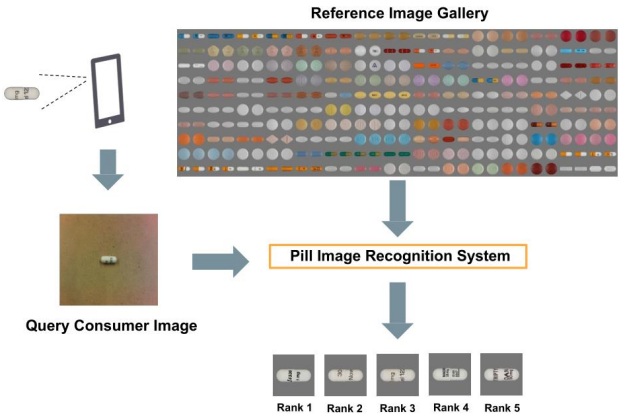


Figure 1: Illustration of the unconstrained pill image recognition problem.

1. INTRODUCTION

Smartphones with built-in high-quality cameras are ubiquitous today. These cameras coupled with advanced computer vision algorithms are turning smartphones into powerful tools for people to retrieve information from the physical world. In recent years, a number of such mobile vision systems have become applicable for daily-life tasks and have demonstrated impressive performance in unconstrained real-world settings. For example, Google Translate app can accurately translate texts on books, road signs, and menus from one language into another [2]. As another example, Amazon shopping app is capable of recognizing objects appeared in the scene and labeling them with associated product information [1].

In 2016, the U.S. National Library of Medicine (NLM) of the National Institutes of Health (NIH) announced a nationwide competition, calling for the creation of a mobile vision system that can recognize pills automatically from a mobile phone picture under unconstrained real-world settings [4]. This competition is motivated by the imperative need for a mobile pill recognition tool that can provide a simple way to recognize mystery pills, prevent unnecessary medication errors, and most importantly, save people's lives in urgent cases. Figure 1 illustrates this unconstrained pill image recognition problem. As shown, a user takes an image of a pill using his or her mobile phone. The pill image recognition system takes the image as input, compares it against thousands of high-quality pill images in the database, and returns a list of most similar pills to the user. The images taken by mobile phones in the unconstrained real-world settings are called *consumer images*. The images stored in the database are called *reference images*, which are taken by higher-resolution cameras in the controlled laboratory

setting. However, developing such pill image recognition system is challenging. This is because in mobile unconstrained scenarios, the quality of photos taken by mobile phones can be easily deteriorated by a variety of factors such as illumination, shading, background, and phone orientation. Moreover, the number of types of prescription pills manufactured by pharmaceutical companies is large. Existing solutions are not capable of handling such large number of types and thus fail when the scale is increased considerably.

Over the past few years, deep learning has become the dominant approach in machine learning due to its impressive capability of handling large-scale learning problems [20]. Deep learning-based approaches have achieved state-of-the-art performance in solving a variety of computer vision problems such as object recognition [18], face recognition [27], video classification [17], and scene understanding [28]. However, deep learning-based approaches are very expensive in terms of computation, memory, and power consumption. As such, given the resource constraints of mobile devices, most of the existing deep learning-based mobile vision systems offload the workload to the cloud [9, 12]. Unfortunately, cloud offloading becomes impossible when the Internet connection is lost. It also suffers from unpredictable end-to-end network delay, which can be affected by many factors such as distance between mobile device and cloud, network bandwidth and data quantity. More importantly, the medication an individual takes contains very sensitive and private information about his or her medical record. Uploading pill images onto the cloud constitutes a great danger to an individual's privacy [6].

In this paper, we present the design, implementation, and evaluation of *MobileDeepPill*, a small-footprint mobile deep learning system that achieves state-of-the-art performance on recognizing unconstrained pill images without cloud offloading. The development of MobileDeepPill includes three key innovations. First, MobileDeepPill employs a triplet loss function [24] to attain invariances to real-world noisiness that deteriorates the quality of pill images taken by mobile phones. Second, MobileDeepPill involves a novel pill image recognition algorithm based on deep Convolutional Neural Network (CNN) [18]. Protected by patent laws, each manufactured prescribed pill is uniquely characterized by the combination of its shape, color and imprints. MobileDeepPill achieves state-of-the-art pill image recognition performance by utilizing a multi-CNNs architecture that collectively captures the shape, color and imprints characteristics of the pills. Third, MobileDeepPill incorporates a novel deep model compression technique based on Knowledge Distillation [15]. The model compression technique significantly reduces the size of the multi-CNNs model without deteriorating its recognition performance by designing a much smaller model and training it using the knowledge extracted from the multi-CNNs model. As a result, MobileDeepPill is able to perform accurate pill image recognition on commodity smartphones efficiently.

The innovations involved in MobileDeepPill are *generic* which can be applied to the development of a wide range of mobile sensing systems powered by on-device deep learning algorithms. First, data collected by mobile sensing systems is likely to be noisy. The triplet loss function can be leveraged to build mobile sensing systems that are resilient to noisiness in unconstrained mobile conditions. Second, the proposed multi-CNNs architecture can be used to build mobile sensing systems that perform inferences based on fusing knowledge extracted from multiple sensing modalities. Finally, the proposed Knowledge Distillation-based model compression framework allows practitioners who want to adopt large off-the-shelf deep convolutional models to generate small-footprint models for resource-limited mobile sensing systems without deteriorating its recognition performance.

Summary of Experimental Results: We have conducted a rich set of experiments to examine both the recognition and system performance of MobileDeepPill. To examine the recognition performance, we have evaluated MobileDeepPill on the official NIH NLM Pill Image Recognition Challenge dataset under two evaluation schemes [4]. To examine the system performance, we have implemented MobileDeepPill on three platforms with different computing power: 1) a desktop installed with an Intel i7-5930k CPU and a Nvidia GTX 1080 GPU (cloud CPU and GPU), 2) a Nvidia Jetson TX1 mobile development board equipped with a Nvidia Tegra X1 GPU (high-end mobile GPU), and 3) a Samsung Galaxy S7 edge smartphone (mobile CPU). Our results show that:

- MobileDeepPill achieves state-of-the-art unconstrained pill image recognition performance. Specifically, in the one-side pill recognition scheme, MobileDeepPill achieves an average 52.7% Top-1 accuracy and 81.7% Top-5 accuracy. In the two-side pill recognition scheme, MobileDeepPill achieves an average 73.7% Top-1 accuracy and 95.6% Top-5 accuracy.
- Our Knowledge Distillation-based deep model compression technique significantly reduces the number of parameters (by 86.8%) and floating-point operations per second (FLOPS) (by 62.8%) of the multi-CNNs model without deteriorating the recognition performance. As a result, MobileDeepPill only requires 34MB runtime memory to run the multi-CNNs model and is able to perform low-power (7665mJ), near real-time (1.58s) pill image recognition on commodity smartphones without cloud offloading. With the support of high-end mobile GPU, the runtime performance of MobileDeepPill is significantly improved (270ms) for real-time usage.

Summary of Contributions: In this work, we introduce the first mobile vision system that achieves real-world applicable performance for recognizing unconstrained pill images. *Our deep learning-based pill image recognition algorithm achieves the best recognition performance among all the contestants and wins the First Prize (champion) of the NIH NLM Pill Image Recognition Challenge*. Moreover, our Knowledge Distillation-based deep model compression technique is *complementary* to existing deep model compression techniques developed for mobile devices [19, 7], thus representing a unique contribution.

The need for a mobile pill recognition tool is more important today than ever before. Given its great performance, we believe MobileDeepPill helps NIH tackle a critical problem with significant societal impact and will benefit millions of healthcare personnel and the general public.

2. CHALLENGES AND OUR SOLUTIONS

Accurately recognizing unconstrained pill images on mobile devices without cloud offloading presents a number of challenges. In this section, we describe these challenges followed by explaining how MobileDeepPill addresses these challenges.

Differences between Reference and Consumer Images: The goal of unconstrained pill image recognition is to match the consumer image of a pill with its corresponding reference image. Figure 2 shows five pairs of consumer images and reference images of the same pills. As shown, depending on how people take the pictures using their phones, and due to the deterioration caused by a variety of real-world noisiness such as shading, blur, illumination and background, consumer image and reference image of the same pill look very different. As a consequence, the recognition accuracy will suffer if we use the model trained on reference images to recognize pills appeared in consumer images. To address this challenge,

Figure 2: Illustration of differences between reference and consumer images of the same pills under five different scenarios. For each scenario, the image on the left is the consumer image; and the image on the right is the reference image of the same pill.

we employ a *triplet loss function* [24] to learn features that attain invariances to the noisiness illustrated in Figure 2 using CNNs. As such, MobileDeepPill can accurately match consumer images and reference images of the same pills even if they look different.

Unreliable Pill Characteristics: Although each prescribed pill can be identified by its unique combination of color, shape and imprints, the characteristics of a pill may appear differently in consumer images caused by real-world noisiness. For example, as shown in Figure 2 (b), a white pill turns into a yellow pill when the illumination in the environment is yellow. To address this challenge, we propose a *multi-CNNs architecture*. Specifically, we generate the gray and gradient images of the original color pill images to train three independent CNNs. CNNs trained on gray and gradient images not only alleviate the dependency on color information which can be deteriorated in unconstrained conditions, but also attain enhanced shape and imprint information which is helpful in distinguishing different pills. By combining the outputs of these three CNNs, MobileDeepPill becomes more resilient to the unreliability of pill characteristics.

Lack of Training Samples: A large volume of training images that contain significant variations is needed to ensure the robustness of our pill image recognition system in mobile settings. Unfortunately, collecting a large volume of diverse consumer images for all types of pills is very challenging. To address this challenge, we perform a number of *data augmentation* techniques to generate new augmented images. The data augmentation techniques generate variations that mimic the variations occurred in mobile settings. With the large amount of newly generated augmented images, CNNs are effectively trained. As a result, MobileDeepPill becomes more robust to the diverse variations in mobile settings.

Resource Constraints of Mobile Devices: To best protect the privacy of the user, pill images need to be completely processed inside mobile devices without cloud offloading. However, CNNs that achieve state-of-the-art accuracy in computer vision tasks usually consume hundreds of MB of memory and billions of FLOPS [18, 27, 25, 14]. Although mobile devices today are equipped with powerful computing resources, they are not designed for efficiently running such computation and memory intensive models. To address this challenge, we propose a novel deep neural network model compression technique based on *Knowledge Distillation* [15], which significantly reduces the memory footprint and FLOPS of CNNs without deteriorating the recognition accuracy. As such, MobileDeepPill can be efficiently running on today's commodity mobile phones while still achieving state-of-the-art recognition accuracy.

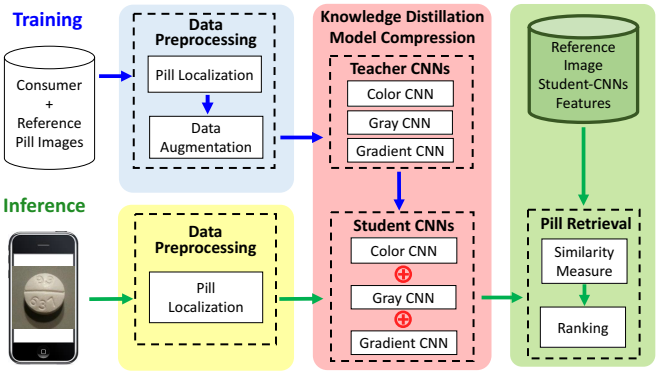


Figure 3: The system architecture of MobileDeepPill. Blue arrows represent flow in training stage. Green arrows represent flow in inference stage.

To the best of our knowledge, MobileDeepPill is the first mobile system that addresses these challenges and achieves real-world applicable performance.

3. SYSTEM OVERVIEW

The system architecture of MobileDeepPill is illustrated in Figure 3. As shown, the training stage and the inference stage contain different system components.

In the training stage, MobileDeepPill first localizes and segments the pill in every consumer and reference image. It then performs data augmentation on the segmented consumer and reference images to generate new augmented images to increase the number of training samples. For every original and augmented image in the training set, MobileDeepPill generates the gray and gradient images of the original color pill images to train three independent CNNs: *Color CNN*, *Gray CNN*, and *Gradient CNN*. Finally, these three CNNs are imported as *Teacher CNNs* into the *Knowledge Distillation Model Compression* framework to derive three small-footprint *Student CNNs*.

In the inference stage, given a consumer image taken by the smartphone, MobileDeepPill first localizes and segments the pill in the consumer image. The segmented consumer image is then fed into the *Student CNNs* to extract CNN features. These features will be compared with the CNN features of all the reference images. Finally, the ranking based on the similarity between the consumer image and all the reference images in the CNN feature space is generated in descending order. Based on the ranking, the reference images of the top N pill candidates are returned to the user.

In the next two sections, we describe the design of MobileDeepPill in details.

4. MULTI-CNNS BASED PILL IMAGE PROCESSING

4.1 Data Preprocessing

4.1.1 Pill Localization

The backgrounds of pill images do not contain useful information to help identify pills, and in real-world settings, can become a source of noises that will affect recognition accuracy. Therefore,

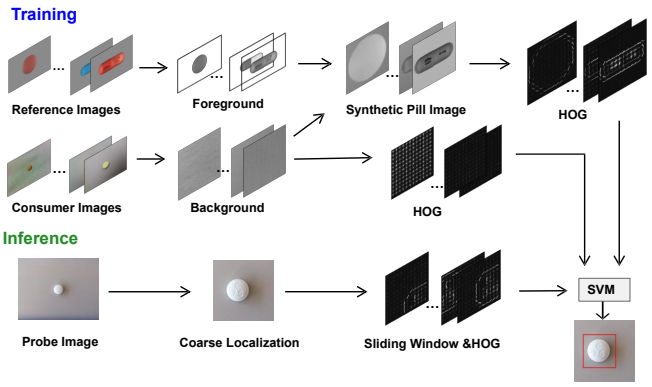


Figure 4: Illustration of pill localization approach for consumer images.

it is necessary to localize and segment the pills in the images in the first place. Considering the differences between reference images and consumer images, we propose two different localization schemes for reference images and consumer images, respectively.

Pill Localization for Reference Images: Since reference images were taken by high-resolution professional cameras in controlled settings, the backgrounds of reference images are uniform and only contain plain texture. As such, we use a simple gradient detection method to localize and segment pills in reference images. Specifically, the gradient detection method finds the image gradients that exceed a given threshold. These gradients strongly imply the location of the pill in a reference image. Using morphologically image operations, the largest connected region, which is also the region of the pill in the reference image is obtained.

Pill Localization for Consumer Images: Directly applying the gradient detection method to consumer images does not achieve good performance because consumer images contain diverse backgrounds. To address this problem, we adopt a learning-based approach to train a pill detector using Support Vector Machine (SVM). Figure 4 provides a graphical overview of our pill localization approach for consumer images. Specifically, we leverage the localization results of reference images and use them as positive localization samples. We manually extract backgrounds of consumer images and use them as negative localization samples. We then use these positive and negative samples as training data and compute the histogram of oriented gradients (HOG) of each sample as features to train the SVM-based pill detector. We localize and segment the pill in a consumer image by first using the gradient detection method to obtain the coarse location of the pill in the image. A sliding window is then used to crop image patches inside the coarse region. These patches are then fed into the trained SVM. The patch with the highest probability is considered as the pill location.

4.1.2 Data Augmentation

We have applied three data augmentation techniques on the segmented pill images. Specifically, we have applied Gaussian filtering to simulate blur images; we have zoomed the images by a random factor in the range of (0.8, 1.2) to ensure our system is robust to pill size variation; and finally, we have applied random translation (in the range of (-5, 5) pixels) and rotation (in the range of (-6, 6) degrees) to the segmented pill images to ensure our system is less sensitive to localization errors.

4.2 Feature Learning Using Triplet Loss and Deep Convolutional Neural Networks

4.2.1 Triplet Loss

The goal of unconstrained pill image recognition is to match the consumer image of a pill with its corresponding reference image. However, as illustrated in Figure 2, due to the deterioration caused in unconstrained conditions, consumer images and reference images of the same pills may look very different. To this end, we employ a triplet loss function (i.e., triplet loss) to address this problem. Triplet loss has been successfully used for addressing variations in expression, pose, and illumination in the problem of unconstrained face recognition [24]. In the context of unconstrained pill recognition, we use triplet loss to guide the training of CNNs to create a new feature space such that in this feature space, the distances between consumer and reference images of the same pills are small, whereas the distances between consumer and reference images of different pills are large.

Specifically, triplet loss takes a set of three images (i.e., triplet) as its input: an *anchor image* I^a, a *positive image* I^p, and a *negative image* I^n. Among them, the *anchor image* and the *positive image* are from the same pill, while the *negative image* is from a different pill. The objective of triplet loss is to find an embedding $f(I)$, from an image I into a d-dimensional feature space $\mathbb{R}^d$, such that in the feature space, the anchor image is closer to all positive images, *independent of the deterioration caused by real-world noisiness*, than to any negative image. This objective can be translated into the following mathematical formulation:

$$\|f(I^a) - f(I^p)\|_2^2 + \alpha < \|f(I^a) - f(I^n)\|_2^2 \quad (1)$$

and the corresponding triplet loss function is formulated as:

$$\mathcal{L} = \max(\|f(I^a) - f(I^p)\|_2^2 - \|f(I^a) - f(I^n)\|_2^2 + \alpha, 0) \quad (2)$$

where α is a margin that controls the distance between the positive pair (I^a, I^p) and negative pair (I^a, I^n).

Figure 5 illustrates the effect of triplet loss. As shown, in the original image space, due to real-world noisiness, the *anchor image* I^a is closer to the *negative image* I^n compared to the *positive image* I^p. This leads to the consequence where the *anchor image* I^a is misclassified to the same pill as the *negative image* I^n. By

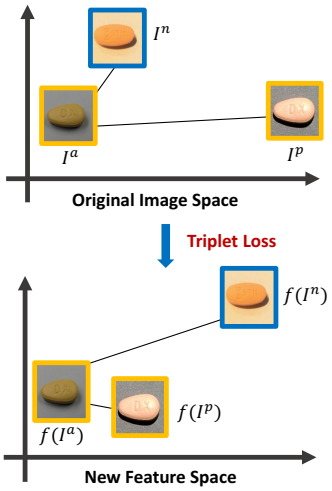


Figure 5: Illustration of the effect of triplet loss. The colors of bounding boxes indicate the classes of the pills.

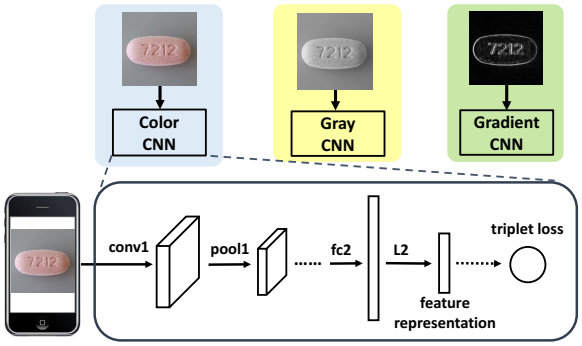


Figure 6: Illustration of the multi-CNNs architecture of MobileDeepPill.

employing triplet loss, the three pill images I^a, I^p, and I^n are projected into a new feature space where the *anchor image* I^a is closer to the *positive image* I^p compared to the *negative image* I^n. As a result, the *anchor image* I^a is correctly classified to the same pill as the *positive image* I^p.

4.2.2 Triplet Generation

One critical issue of employing triplet loss is how to generate the correct triplets. This is because triplets that are generated using random combination method do not contain much useful information. As a result, these randomly selected triplets do not contribute to the CNN training and thus lead to slow convergence [24].

To address this problem, we have designed an online scheme that generates informative triplets from intermediate results during the training process. The key idea is to generate triplets based on current pill ranking. Specifically, for an image in the training dataset, we treat it as I^a and randomly select I^p from the same pill class. We leverage the current ranking of I^a and choose its neighboring pill class in the ranking as I^n. The rationale behind this strategy is that we want the CNN to learn features that are distinctive among different pill classes, especially among very similar pills. As an example, assume we have I^a and its corresponding reference image ranks $10th$. We randomly select one pill that ranks close to $10th$ as the negative image I^n. As we have observed in our experiments, our online triplet generation scheme successfully generates informative triplets that contribute to improving the CNN model as well as fast convergence during the training process.

4.2.3 Multi-CNNs Architecture

Figure 6 shows the multi-CNNs architecture of our MobileDeepPill system. As illustrated, the multi-CNNs architecture consists of three independent CNNs: *Color CNN*, *Gray CNN*, and *Gradient CNN*. These three CNNs play different roles and extract complementary features that collectively capture the shape, color and imprints characteristics of the pills. Specifically, *Color CNN* is responsible for extracting color information of the pills; and *Gray CNN* and *Gradient CNN* are responsible for extracting shape and imprints information of the pills.

We design the network architecture of the three CNNs based on AlexNet [18]. All the three CNNs have the same network architecture except the first convolutional layer. This is because for *Color CNN*, the input is the RGB color images and thus the number of input channels is three; whereas for *Gray CNN* and *Gradient CNN*, the input is the gray and gradient images respectively and thus the number of input channels is one. Table 1 provides the details of the network architecture of *Color CNN*. As shown, the network consists of five convolutional layers (*conv1* to *conv5*), three pooling layers (*pool1*, *pool2*, and *pool5*), and two fully connected layers (*fc1* and *fc2*). Finally, on top of the *fc2* layer, a L2 normalization layer is added to output the feature representation $f(I)$.

Layer	Size In	Size Out	Kernel
conv1	$227 \times 227 \times 3$	$55 \times 55 \times 96$	$11 \times 11, 4$
pool1	$55 \times 55 \times 96$	$27 \times 27 \times 96$	$3 \times 3, 2$
conv2	$27 \times 27 \times 96$	$27 \times 27 \times 256$	$5 \times 5, 1$
pool2	$27 \times 27 \times 256$	$13 \times 13 \times 256$	$3 \times 3, 2$
conv3	$13 \times 13 \times 256$	$13 \times 13 \times 384$	$3 \times 3, 1$
conv4	$13 \times 13 \times 384$	$13 \times 13 \times 384$	$3 \times 3, 1$
conv5	$13 \times 13 \times 384$	$13 \times 13 \times 256$	$3 \times 3, 1$
pool5	$13 \times 13 \times 256$	$6 \times 6 \times 256$	$3 \times 3, 2$
fc1	$6 \times 6 \times 256$	1024	
fc2	1024	128	
L2	128	128	

Table 1: The network architecture of *Color CNN*. The parameters in the Size In and Size Out columns follow the format of $height \times width \times \#kernels$. The parameters in the Kernel column follow the format of $height \times width, stride$. The network architecture of *Gray CNN* and *Gradient CNN* are the same except the Size In of *conv1* is $227 \times 227 \times 1$.

4.3 Pill Retrieval

Given a query consumer image taken by a smartphone, the objective of pill retrieval is to compare the query consumer image to the reference images of all pill classes and then output a similarity ranking in a descending order. MobileDeepPill achieves this by measuring the similarity between consumer and reference images in the learned CNN feature space. Specifically, we use the cosine distance as the metric to calculate a similarity score:

$$S = f(I_q)^T f(I_{ref}) \tag{3}$$

where I_q is the query consumer image and I_{ref} is the reference image that I_q is compared to. Since we have three independent CNNs, three similarity scores S_{color}, S_{gray} and $S_{gradient}$ are calculated respectively. Finally, since color, shape and imprints information is complementary for identifying pills, a final similarity score is calculated as the unweighted sum of S_{color}, S_{gray} and $S_{gradient}$.

$$S_{final} = S_{color} + S_{gray} + S_{gradient} \tag{4}$$

5. KNOWLEDGE DISTILLATION-BASED MODEL COMPRESSION

Although our multi-CNNs model is able to extract complementary features that collectively capture the color, shape, and imprints characteristics of pills, it contains about 40 million parameters and consumes over 3.2 billion FLOPS (see Table 3 for breakdown details). The large number of parameters makes the model memory intensive; and the high FLOPS makes the model computation intensive. Although mobile devices today are equipped with powerful computing resources, they are not designed for efficiently running such memory and computation intensive model.

To address this problem, MobileDeepPill adopts a novel deep neural network model compression technique based on Knowledge Distillation [15]. It compresses the large multi-CNNs model into a small network with much smaller numbers of parameters and FLOPS, and can be efficiently running on mobile devices without deteriorating the recognition accuracy. The Knowledge Distillation model compression framework involves two networks: a *student network* which is the original large network to be compressed; and a *student network* which is the desired small network to be designed. By training the student network to imitate the outputs of

the teacher network, the knowledge of the teacher network is transferred to the student network. Although there are a few existing model compression techniques developed for mobile devices, they focus on compressing pretrained large networks [19, 7]. In contrast, by following a number of optimization strategies, our compression technique focuses on designing and training a new network architecture with a much smaller footprint.

In this section, we first provide the background knowledge of Knowledge Distillation. We then describe the strategies that guide the design of the student network. By following these strategies, we describe the architecture of the newly designed student network, and compare it with the original network (i.e., teacher network). Finally, we describe how to use the teacher network to train the student network.

5.1 A Primer on Knowledge Distillation

Knowledge Distillation was first introduced by Hinton *et al.* as a framework for model compression [15]. It aims to train a small network (i.e., *student network*) using the outputs of the original large network (i.e., *teacher network*). By doing it, the teacher network transfers its knowledge learned from data to the student network.

In [15], this framework is applied to the object recognition problem where the outputs of both teacher and student networks are softmax outputs $P_t = softmax(\mathbf{a}_t^L)$ and $P_s = softmax(\mathbf{a}_s^L)$:

$$P_t^\tau = softmax(\frac{\mathbf{a}_t^L}{\tau}) \tag{5}$$

$$P_s^\tau = softmax(\frac{\mathbf{a}_s^L}{\tau}) \tag{6}$$

where $\mathbf{a}^L$ is the activation of the last layer L and $\tau > 1$ is the relaxation to soften the outputs of both teacher and student networks. Based on the softmax outputs, the student network is trained to optimize the following loss function:

$$\mathcal{L} = \mathcal{H}(\mathbf{y}, P_s) + \lambda \mathcal{H}(P_t^\tau, P_s^\tau) \tag{7}$$

where $\mathcal{H}$ is the cross-entropy, $\mathbf{y}$ are true labels, and λ controls how much knowledge should be transferred from the teacher network.

In our work, since our goal is to compress the large multi-CNNs model without deteriorating its recognition performance, we use the following simplified loss function instead to only enforce the student network to imitate the outputs of the teacher network.

$$\mathcal{L} = \left\| (\mathbf{a}_t^L - \mathbf{a}_s^L) \right\|_2^2 \tag{8}$$

5.2 Student Network Design Strategies

The overarching goal of the student network design is to minimize the computational cost (i.e., the number of FLOPS) and memory footprint (i.e., the numbers of model parameters) while maintaining the recognition performance. As such, the design of the student network involves the examination of the trade-offs between recognition performance and a number of factors of the deep neural network architecture including depth, width, filter numbers, and filter sizes. However, this is a very complicated problem due to the complexity of the deep neural network architecture. To simplify the problem, we adopt the *layer replacement* scheme proposed in [13]. The core of the *layer replacement* scheme is that, at each time, one or a few layers are replaced by some other layers that preserve the recognition performance, without changing the other layers. Based on this scheme, we use the teacher network as the starting point, and progressively modify the teacher network architecture by examining the trade-offs between the recognition performance and the network architecture factors through a series of controlled experiments. This process not only leads to a small-footprint student network that maintains the recognition performance as its teacher network, but also helps formulate a number of strategies that can guide model compression.

In the following, we describe these strategies one by one. It is worthwhile to note that many similar strategies have been validated by other researchers in building compressed models for other tasks such as object recognition [13, 16]. Therefore, we expect our student network design strategies are helpful to building small-footprint deep neural network models on resource limited platforms like mobile phones and wearables for many other applications.

Strategy 1: Late downsampling. Late downsampling refers to maintaining large activation maps (i.e., the outputs of convolutional layers) at early layers and downsampling activation maps at later layers. It has been shown that late downsampling not only helps reduce the number of model parameters but also leads to higher recognition performance by keeping activation maps at early layers large to preserve information contained in images [16]. As such, when designing the student network, we propose to perform late downsampling to enhance recognition performance while reducing the number of model parameters.

Strategy 2: Use 1×1 filters for channel reduction. The concept of 1×1 filter was first introduced in [22]. It has been shown in GoogLeNet [26] that the 1×1 filter can significantly reduce the channel dimensions without sacrificing recognition accuracy. As such, when designing the student network, we propose to insert a new convolutional layer that uses 1×1 filter before the first fully-connected layer. By doing so, the number of parameters of the first fully-connected layer can be significantly reduced.

Strategy 3: Split one convolutional layer into multiple convolutional layers with filter number trade-off. Increasing network depth by adding extra layers has been demonstrated to be the key to improving accuracy [25, 26, 13]. However, adding extra layers also increases the number of model parameters. As such, when designing the student network, we propose to split one convolutional layer into two sub convolutional layers to increase depth. We also propose to adjust the filter number of the two sub convolutional layers such that the number of parameters is reduced or remains roughly the same. By doing this, the student network becomes deeper without increasing the number of model parameters.

Strategy 4: Split one convolutional layer into multiple convolutional layers with filter size trade-off. The receptive field of a convolutional layer determines how much information is preserved in that layer and thus plays a key role on recognition performance. It has been shown that the receptive field of one convolutional layer with large filter size (e.g., 5×5, 7×7) is equivalent to stacking multiple 3×3 convolutional layers [26]. This finding shows a way to increase the network depth by trading off filter size. As such, when designing the student network, we propose to split one convolutional layer that uses larger filter size into multiple 3×3 convolutional layers. By doing this, the student network becomes deeper without loosing the receptive field.

Strategy 5: Reduce filter number without splitting. Keeping increasing the network depth does not necessarily lead to accuracy improvement. This is because deeper networks are more difficult to train due to the degradation problem [14]. As such, when designing the student network, we propose to reduce the number of filters when increasing the depth does not help to improve the recognition accuracy. By doing this, it reduces the numbers of model parameters and FLOPS.

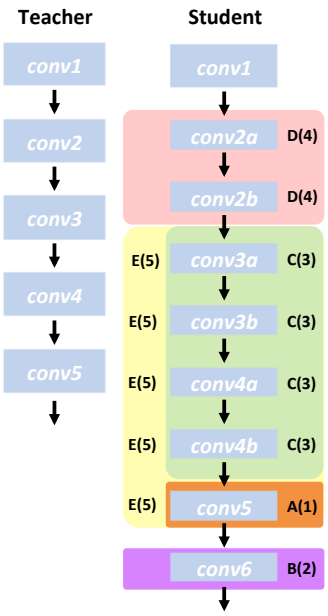


Figure 7: Illustration of the student network design process with the annotated strategy numbers and model symbols. Syntax: model symbol (strategy number).

5.3 Student Network Architecture

By following the above design strategies step by step, we gradually transform the teacher network into the student network. In this section, we go through the models generated at each step. The model to be generated at the next step is built on top of the current model. To help illustrate the whole process, Figure 7 is plotted with the annotated strategy numbers and model symbols. In addition, to illustrate the progress we have made at each step towards building a small-footprint student network, Table 2 lists the numbers of model parameters and FLOPS for each model.

Model A: We have followed **Strategy 1** to derive this model. Specifically, we perform late downsampling at *conv5* by changing its stride from 1 to 2. As a result, the number of parameters in *fc1* is reduced by 75%, dropping from $1024 \times 6 \times 6 \times 256$ to $1024 \times 3 \times 3 \times 256$.

Model B: We have followed **Strategy 2** to derive this model. Specifically, we add a new layer *conv6* comprised of 64 1×1 filters after *conv5*. As a result, the network depth increases one; the channels of activation maps after *conv6* are reduced to 64; and the number of parameters in *fc1* is reduced by 75%, dropping from $1024 \times 3 \times 3 \times 256$ to $1024 \times 3 \times 3 \times 64$.

Model C: We have followed **Strategy 3** to derive this model. Specifically, we split *conv3* and *conv4* into two sub convolutional layers *conv3a*, *conv3b* and *conv4a*, *conv4b*, respectively. As a result, the network depth increases two; and the filter number is reduced to 256.

Model D: We have followed **Strategy 4** to derive this model. Specifically, we split *conv2* into two sub convolutional layers *conv2a* and *conv2b*. As a result, the network depth increases one; and the filter size is reduced to 3×3.

Model E (Student Network): We have followed **Strategy 5** to derive this model. As a result, the numbers of filters in *conv3a*, *conv3b*, *conv4a*, *conv4b*, and *conv5* are further reduced to 128.

Model	T	A	B	C	D	E
# params (m)	13.32	6.24	4.49	4.34	3.69	1.76
FLOPS (m)	1093	984	973	985	676	407

Table 2: The numbers of parameters and FLOPS for each model during the student network design process (T = teacher network, m = million).

Table 3 and Table 4 show the details of the architecture of the teacher network and the student network of *Color CNN*, respectively. For *Gray CNN* and *Gradient CNN*, the architecture is the same as *Color CNN* except for the first layer where the number of parameters is 12k and the number of FLOPS is 37.4m. As shown in the tables, the generated student network only has 1.76m parameters and 407m FLOPS. Compared to the original teacher network, the corresponding compression ratio is 7.6 (13.32m / 1.76m) for the number of parameters, and 2.7 (1093m / 407m) for FLOPS.

Layer	Size Out	Kernel	# params	FLOPS
conv1	$55 \times 55 \times 96$	$11 \times 11, 4$	35k	112.2m
pool1	$27 \times 27 \times 96$	$3 \times 3, 2$		
conv2	$27 \times 27 \times 256$	$5 \times 5, 1$	615k	447.9m
pool2	$13 \times 13 \times 256$	$3 \times 3, 2$		
conv3	$13 \times 13 \times 384$	$3 \times 3, 1$	885k	149.5m
conv4	$13 \times 13 \times 384$	$3 \times 3, 1$	1330k	224.3m
conv5	$13 \times 13 \times 256$	$3 \times 3, 1$	885k	149.5m
pool5	$6 \times 6 \times 256$	$3 \times 3, 2$		
fc1	1024		9440k	9.4m
fc2	128		131k	0.1m
Total			13.32m	1093m

Table 3: The architecture of the *Color CNN* teacher network (Model T).

Layer	Size Out	Kernel	# params	FLOPS
conv1	$55 \times 55 \times 96$	$11 \times 11, 4$	35k	112.2m
pool1	$27 \times 27 \times 96$	$3 \times 3, 2$		
conv2a	$27 \times 27 \times 128$	$3 \times 3, 1$	111k	80.6m
conv2b	$27 \times 27 \times 128$	$3 \times 3, 1$	147k	107.5m
pool2	$13 \times 13 \times 128$	$3 \times 3, 2$		
conv3a	$13 \times 13 \times 128$	$3 \times 3, 1$	148k	24.9m
conv3b	$13 \times 13 \times 128$	$3 \times 3, 1$	148k	24.9m
conv4a	$13 \times 13 \times 128$	$3 \times 3, 1$	148k	24.9m
conv4b	$13 \times 13 \times 128$	$3 \times 3, 1$	148k	24.9m
conv5	$7 \times 7 \times 128$	$3 \times 3, 2$	148k	6.2m
conv6	$7 \times 7 \times 64$	$1 \times 1, 1$	8k	0.4m
pool6	$3 \times 3 \times 64$	$3 \times 3, 2$		
fc1	1024		591k	0.6m
fc2	128		131k	0.1m
Total			1.76m	407m

Table 4: The architecture of the *Color CNN* student network (Model E).

5.4 Student Network Training

In our proposed system, the teacher networks are our original color, gray and gradient CNNs described in section 4.2.3. Given the newly designed student network architecture in section 5.3, we apply the Knowledge Distillation framework described in section 5.1 to train the student network.

The training process consists of two stages. In the first stage, we let the student network learn by itself by training the student network using triplet loss. In the second stage, we let the pretrained teacher network train and transfer knowledge to the student network using Knowledge Distillation. By doing this, the student network learns the low-level features better and achieves much better generalization performance.

6. EVALUATION

6.1 Experimental Setup

6.1.1 Dataset

We conduct our experiments using the official NIH NLM Pill Image Recognition Challenge dataset [4]. The dataset contains 1000 distinct pills and has two categories of pill images:

Reference Images: This category contains 2000 reference images of the 1000 pills, with one image for the front side of the pill and one image for the back side of the pill. These images were taken under the same controlled laboratory settings using a high-resolution camera directly above the front and back sides of the pills.

Consumer Images: This category contains 5000 consumer images of the 1000 pills. These images were taken by digital cameras in mobile phones in real-world settings. For each of the 1000 pills, there are five consumer images taken under different conditions with variations in camera angle, illumination, and background.

6.1.2 Evaluation Metrics

To evaluate the pill recognition performance of MobileDeepPill, we use the Mean Average Precision (MAP) and Top-K accuracy as the evaluation metrics.

Mean Average Precision (MAP): MAP is the evaluation metric used in the official NIH NLM Pill Image Recognition Challenge. It is defined as:

$$MAP = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{N_i}\sum_{j=1}^{N_i}\frac{j}{MT(i,j)}\right) \tag{9}$$

where N denotes the number of query images used for evaluation, N_i is the number of genuine images which is always 2 because there are two sides for each pill, and $MT(i,j)$ are the rankings of two genuine images.

Top-K Accuracy: In practice, users are usually only interested in the top K pill candidates (e.g., K equal to 1, 5) retrieved by the system. Thus we employ Top-K accuracy as another metric to assess the recognition performance, which is defined as the ratio of the number of query images whose true mates are within the top K candidates retrieved by the system to the number of query images.

Besides the pill recognition performance, we also evaluate the system performance of MobileDeepPill including runtime, runtime memory, and energy consumption.

6.1.3 Evaluation Schemes and Protocol

Evaluation Schemes: We evaluate the recognition performance under the following two schemes:

- **One-Side Pill Recognition:** In this scheme, the system takes one consumer image, either front side or back side of a pill, as the input. The goal is to retrieve the reference image of either side of the same pill.
- **Two-Side Pill Recognition:** In this scheme, the system takes two consumer images, both front and back side of the same pill, as the input. The goal is to retrieve the reference images of both sides of the same pill.

Evaluation Protocol: For each evaluation scheme, we use 5-fold cross validation as the evaluation protocol. Specifically, we evenly split the 1000 pills into 5 folds of mutually exclusive pill sets, with the front side and back side of the same pill in the same fold. As such, each fold contains 200 distinct pills with 400 reference images (two reference images for each pill) and their corresponding 1000 consumer images. During training, we use all the images in four folds. During testing, we use the consumer images in the remaining fold as query images and all the reference images in 5 folds as gallery images.

6.2 Pill Image Recognition Performance

6.2.1 Superiority of Multi-CNNs over Single-CNN

Table 5 lists the recognition performance of the Single-CNN model (i.e., *Color CNN*) and the Multi-CNNs model in terms of MAP and Top-K accuracy (K = 1, 5) under both one-side and two-side pill recognition schemes. As shown, in both schemes, the Multi-CNNs model outperforms the Single-CNN model by a large margin across all three metrics. Specifically, in the one-side pill recognition scheme, MAP score increases 0.14 (from 0.242 to 0.382); Top-1 accuracy increases 27.1% (from 26.0% to 53.1%); and Top-5 accuracy increases 29.9% (from 53.2% to 83.1%). In the two-side pill recognition scheme, MAP score increases 0.27 (from 0.567 to 0.837); Top-1 accuracy increases 31% (from 43.1% to 74.1%); and Top-5 accuracy increases 22.9% (from 73.5% to 96.4%). This result demonstrates the significant superiority of the Multi-CNNs model over the Single-CNN model. This is because for the Single-CNN model (i.e., *Color CNN*), its recognition performance suffers when the color of the pill is changed in unconstrained conditions. In contrast, even if the color of the pill is changed, the Multi-CNNs model can still count on the shape and imprints information extracted from the *Gray CNN* and *Gradient CNN* to retrieve the correct pill. It is also worthwhile to note that both Single-CNN and Multi-CNNs achieve much better recognition performance in the two-side scheme than in the one-side scheme. This result indicates that providing both front and end side images of the query pill can significantly enhance the recognition performance.

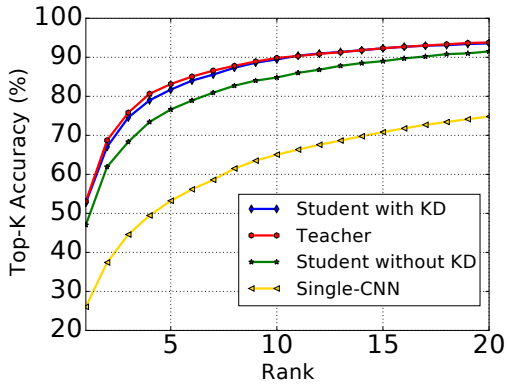
6.2.2 Capability of the Student Network

Table 5 also compares the recognition performance between the teacher network and the student network. Two types of student network are considered: 1) *student network with KD*, which is the compressed Multi-CNNs model trained with Knowledge Distillation; and 2) *student network without KD*, which is the compressed Multi-CNNs model trained without Knowledge Distillation. As shown, for *student network with KD*, even if the student network has much fewer parameters, with the knowledge passed from the teacher network, it achieves almost equivalent recognition performance as the teacher network. For example, in the two-side pill recognition scheme, *student network with KD* achieves 95.6% Top-5 accuracy, which is only 0.8% less than the teacher network. This result indicates that the student network, though small in size, is able to absorb the knowledge passed from the teacher network, which demonstrates the effectiveness of our Knowledge Distillation-based model compression technique. In contrast, for *student network without KD*, by learning from the training dataset by itself with no knowledge passed from the teacher network, it has a significant performance drop across all three metrics in both schemes compared to the teacher network.

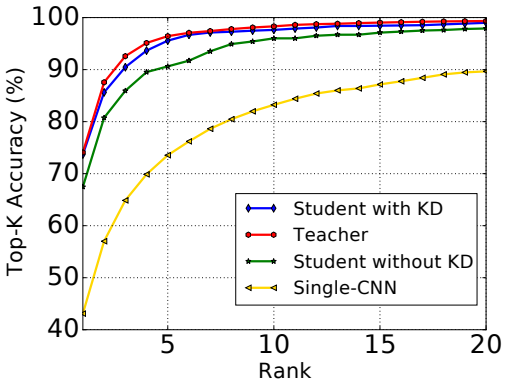
Finally, Figure 8 plots the Cumulative Match Characteristic (CMC) curves for both one-side and two-side pill recognition schemes to illustrate the trend of the Top-K accuracy across different models. As shown, in both one-side and two-side schemes, *student network with KD* is closely aligned with the teacher network from Top-1 to Top-20 accuracy. It also consistently outperforms *student network without KD* and the Single-CNN model from Top-1 to Top-20 accuracy. These results demonstrate once again the effectiveness of our Knowledge Distillation-based model compression technique.

	One-Side Pill Recognition			Two-Side Pill Recognition		
Model	**MAP**	**Top-1**	**Top-5**	**MAP**	**Top-1**	**Top-5**
Single-CNN	0.242 ± 0.008	$26.0 \pm 1.2\%$	53.2 ± 1.9	0.567 ± 0.007	$43.1 \pm 1.2\%$	$73.5 \pm 1.0\%$
Multi-CNNs (Teacher Network)	$\mathbf{0.382 \pm 0.008}$	$\mathbf{53.1 \pm 1.0\%}$	$\mathbf{83.1 \pm 0.9\%}$	$\mathbf{0.837 \pm 0.007}$	$\mathbf{74.1 \pm 0.8\%}$	$\mathbf{96.4 \pm 0.8\%}$
Student Network with KD	0.375 ± 0.009	$52.7 \pm 1.1\%$	$81.7 \pm 1.1\%$	0.829 ± 0.007	$73.7 \pm 1.0\%$	$95.6 \pm 0.5\%$
Student Network without KD	0.353 ± 0.008	$47.1 \pm 1.0\%$	$76.6 \pm 1.1\%$	0.779 ± 0.007	$67.5 \pm 0.9\%$	$90.6 \pm 0.7\%$

Table 5: Summary of unconstrained pill image recognition results (KD = Knowledge Distillation).



(a) The CMC curve of the One-Side Pill Recognition scheme.



(b) The CMC curve of the Two-Side Pill Recognition scheme.

Figure 8: The Cumulative Match Characteristic (CMC) curves of two evaluation schemes. The horizontal axis is the rank K; and the vertical axis is the corresponding Top-K accuracy.

6.3 System Performance

To evaluate the system performance of MobileDeepPill, we have implemented MobileDeepPill end-to-end on three hardware platforms. Our goal is to profile the system performance of MobileDeepPill across platforms with different computing power. Specifically, we use a desktop installed with an Intel i7-5930k CPU and a Nvidia GTX 1080 GPU to simulate a cloud server; we use a Nvidia Jetson TX1 mobile development board that contains a Nvidia Tegra X1 GPU to simulate the next-generation smartphone with built-in high-end mobile GPU; and we use the Samsung Galaxy S7 edge as the commodity smartphone platform and run MobileDeepPill on its CPU. Figure 9 shows the three hardware platforms. The specs of those three hardware platforms are summarized in Table 6.

To provide a comprehensive evaluation, we have evaluated the system performance of all the six models listed in Table 2 (i.e., Model T, A, B, C, D, and E) in terms of runtime performance, runtime memory performance, and energy consumption. In the following, we report their system performance respectively.

(a) Desktop with Intel i7-5930k CPU and Nvidia GTX 1080 GPU

(b) Nvidia Jetson TX1

(c) Samsung Galaxy S7 edge

Figure 9: Hardware platforms.

Platform	CPU		RAM	GPU		
	Cores	Speed		Cores	GFLOPS	Speed
Desktop	6	3.7 GHz	32 GB	2560	8228	1.67 GHz
Jetson TX1	4	1.9GHz	4GB	256	512	1 GHz
Galaxy S7 edge	8	2.3 GHz	4 GB	-	-	-

Table 6: The specs of the three hardware platforms.

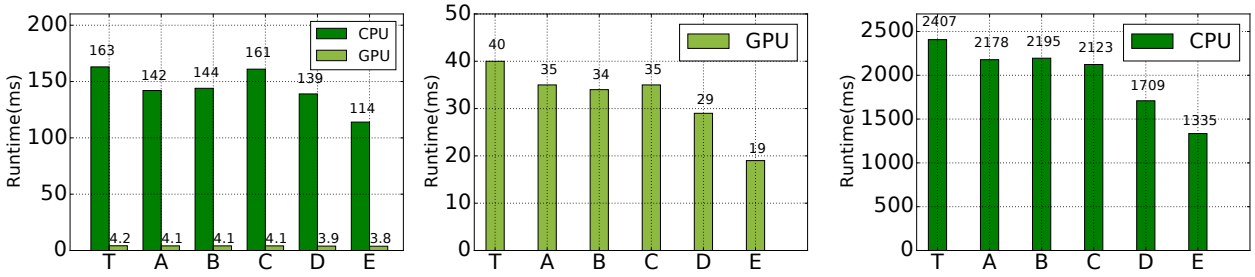
6.3.1 *Runtime Performance*

To evaluate the runtime performance of MobileDeepPill, we measure the average processing time consumed at each computing stage during inference. An inference carried out by MobileDeepPill consists of four computing stages: 1) coarse-grained localization, 2) fine-grained localization, 3) CNN feature extraction, and 4) pill retrieval. We run 2000 inferences and report the average runtime at each stage. Specifically, since the computations at stages 1), 2) and 4) only involve CPU, we first examine the runtime performance of these three stages. Table 7 lists the CPU runtime of these three stages on the three hardware platforms. As shown, among the three hardware platforms, the Galaxy S7 edge has the worst runtime performance. Among the three computing stages, fine-grained localization has the worst runtime performance. This is because it involves extracting HOG features which is computation intensive.

Platform	Coarse-Grained Localization	Fine-Grained Localization	Pill Retrieval
Desktop	0.2	6.1	1.2
Jetson TX1	0.5	39.8	2.7
Galaxy S7 edge	47.2	165.1	39.4

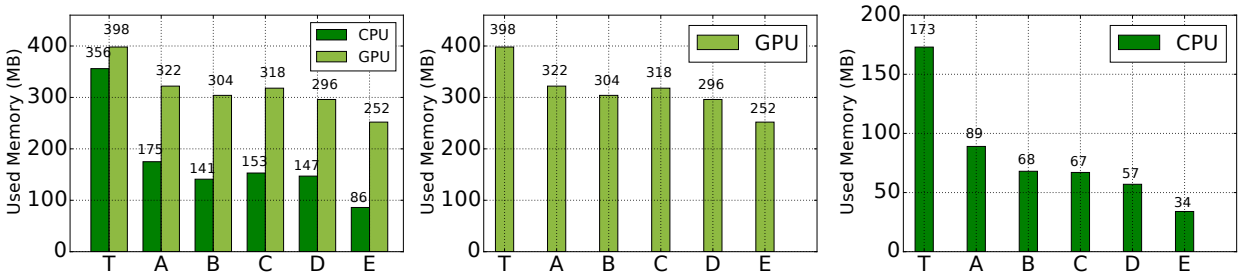
Table 7: CPU runtime of coarse-grained localization, fine-grained localization, and pill retrieval computing stages during inference (unit: ms).

Next we evaluate the runtime performance of stage 3) of all the six CNN models listed in Table 2 on the three platforms. Figure 10 illustrates the results and we have three observations. First, the CNN runtime on GPU is significantly lower than the CNN runtime on CPU across all six CNN models, demonstrating the significant superiority of GPU over CPU for running CNN models. For each CNN model, GPU achieves at least 30X speedup over CPU on the desktop platform; and at least 58X speedup on the mobile platforms (i.e., TX1 and S7 edge). Second, our model compression technique effectively reduces CNN runtime on both desktop and mobile platforms. Specifically, on the desktop platform, the student network (i.e., Model E) achieves 1.43X (163ms / 114ms) speedup on CPU



(a) CNN runtime performance on Desktop **(b)** CNN runtime performance on TX1 **(c)** CNN runtime performance on S7 edge

Figure 10: CNN runtime performance on three platforms (T = teacher network, E = student network).



(a) CNN runtime memory usage on Desktop **(b)** CNN runtime memory usage on TX1 **(c)** CNN runtime memory usage on S7 edge

Figure 11: CNN runtime memory usage on three platforms (T = teacher network, E = student network).

and 1.1X (4.2ms / 3.8ms) speedup on GPU over the teacher network (i.e., Model T); on the mobile platforms, the student network achieves 1.8X (2407ms / 1335ms) speedup on CPU and 2.1X (40ms / 19ms) speedup on GPU over the teacher network. Finally, the reduction in runtime is not as significant as the reduction in FLOPS listed in Table 2. This is due to the parallel operations in the CNN inference implementations. As an example, on the desktop platform, the GPU runtime barely changes as the CNN model size decreases. This is because the operations on the GTX 1080 GPU are highly parallel.

6.3.2 *Runtime Memory Usage*

We evaluate the runtime memory usage of all six CNN models on the three platforms. To accurately measure the CNN runtime memory usage, for Galaxy S7 edge, garbage collection operation is invoked each time before we measure the runtime memory usage. For the desktop and Jetson TX1, we subtract the memory usage before the CNN model is loaded and only report the memory usage that is allocated to the CNN model only. Figure 11 illustrates the results. As shown, our model compression technique effectively reduces the CNN runtime memory usage on both desktop and mobile platforms. In particular, the student network only requires 34MB runtime memory to run on the commodity Samsung Galaxy S7 edge smartphone, achieving a reduction of 139MB runtime memory compared to its teacher network (173MB).

6.3.3 *Energy Consumption*

To evaluate the energy consumption of MobileDeepPill, we measure the average energy consumption at each computing stage during inference on two mobile platforms: Galaxy S7 edge and Jetson TX1. We use the Monsoon power monitor [3] to measure the power consumption of Galaxy S7 edge and the PWRcheck power analyzer [5] to measure the power consumption of Jetson TX1. We run 2000 inferences and report the average energy consumption

Stage	Power (mW)	Time (ms)	Energy (mJ)
Sleep	39.7	-	-
Screen-On	567.6	-	-
App-On	1180.9	-	-
Coarse-Grained Localization	2895.1	47.2	137
Fine-Grained Localization	2473.5	165.1	408
Teacher Network (CPU)	5546.2	2407	13350
Student Network (CPU)	5243.8	1335	7000
Pill Retrieval	3042.7	39.4	120
Total (Teacher)	-	-	14015
Total (Student)	-	-	7665

Table 8: Energy consumption on Samsung Galaxy S7 edge smartphone.

Stage	Power (mW)	Time (ms)	Energy (mJ)
Idle	3564.9	-	-
Coarse-Grained Localization	5711.3	0.5	3
Fine-Grained Localization	5849.5	39.8	233
Teacher Network (GPU)	11763.4	40	471
Student Network (GPU)	9378.7	19	178
Pill Retrieval	6533.1	2.7	18
Total (Teacher)	-	-	725
Total (Student)	-	-	432

Table 9: Energy consumption on Nvidia Jetson TX1.

at each stage. Table 8 and Table 9 show the energy consumption of Galaxy S7 edge and Jetson TX1 at each computing stage, respectively. As shown, the most power hungry stage is the CNN feature extraction. This is expected because CNN contains much more computing operations than any other stage. More interestingly, although the difference in power consumption between the teacher network and the student network is not large, due to the significant runtime reduction, the student network consumes much less energy than the teacher network, reducing energy consumption by 47.6% (from 13350mJ to 7000mJ) on Galaxy S7 edge and by 62.2% (from 471mJ to 178mJ) on Jetson TX1.

Platform	Model	CPU	GPU
Galaxy S7 edge	Teacher Network	3560	-
	Student Network	6509	-
Jetson TX1	Teacher Network	-	68822
	Student Network	-	115500

Table 10: Estimated numbers of inferences with a 3600mAh battery.

Finally, we report the estimated number of inferences that can be performed with a fully charged 3600mAh battery, assuming an ideal discharge curve. As listed in Table 10, when the teacher network is used, S7 edge can process a total number of 3560 inferences. In comparison, when the student network is used, the total number of inferences can be processed is increased to 6509.

7. DISCUSSION AND FUTURE WORK

Impact on Pill Image Recognition Technology: MobileDeepPill represents the state-of-the-art mobile vision system for pill image recognition in unconstrained real-world settings. It demonstrates the superiority of deep learning-based approaches which learn features automatically over traditional approaches which use hand-crafted features designed based on domain knowledge. Given its promising performance and the prevalence of mobile phones, we envision that MobileDeepPill can become a widely used tool that provides people with a simple and convenient way to identify mystery pills and fetch pill information when needed.

Generality of MobileDeepPill: Although MobileDeepPill is designed for solving unconstrained pill image recognition problem using mobile phones, many techniques involved in MobileDeepPill can be generalized to the development of other mobile sensing systems with on-device deep learning algorithms. First, the quality of the sensor data collected by mobile sensing systems can be easily deteriorated by a variety of noises in unconstrained mobile conditions. The employment of triplet loss can enhance the resilience of mobile sensing systems to those noises. Second, modern mobile sensing systems normally utilize more than one sensing modalities to capture information from users and their surrounding environment. The multi-CNNs architecture can be applied to building mobile sensing systems that need to combine information from multiple sensor data sources. Finally, the general trend on developing new deep neural networks has been to make the networks larger and more complicated to maximize the recognition performance without considering model size and computational complexity. The Knowledge Distillation-based model compression technique can be applied to any large deep convolutional models to reduce their numbers of parameters and FLOPS so that the compressed models can efficiently run on the resource-limited mobile sensing systems.

Opportunities for Improvement: There are opportunities to make MobileDeepPill better. As an example, MobileDeepPill currently does not support recognizing multiple pills appeared in one image. To solve this issue, as the first step, algorithms that can accurately detect and localize multiple pills need to be developed. After the pills are localized and segmented, the orientation of each individual pill must be determined, and each pill must be oriented correctly before sending to the pill recognition system for identification. We leave it as our future work.

8. RELATED WORK

Our work is related to two research areas: 1) pill image recognition; and 2) deep neural network model compression. In this section, we provide a brief review of the most relevant work within each area.

Pill Image Recognition: Due to the lack of large openly available datasets, there are only a few works on pill image recognition, and most of them focus on pill images taken in well-controlled conditions. Lee *et al.* [21] developed a system called Pill-ID to recognize illicit pills using Hu moment and Grid intensity, where the illicit pills are of high quality and manually cropped. Caban *et al.* [8] proposed a pill recognition scheme using shape, imprint and color features. However, they used augmented images which were generated by random perturbation of the reference images to evaluate the system. All of these methods used hand-crafted features designed based on domain knowledge for recognition. In contrast, MobileDeepPill is the first work on developing deep learning-based approach to automatically learn the features for recognizing unconstrained pill images.

Deep Neural Network Model Compression: Model compression for deep neural networks has attracted significant attention in recent years due to the imperative demand on running deep learning models on resource limited platforms. In general, deep neural network model compression techniques can be grouped into two categories. The first category is focused on compressing pretrained large networks. For example, Han *et al.* [11] proposed a network pruning method that identifies important connections in the deep neural network and removes all the unimportant connections whose weights are lower than a threshold. Rastegari *et al.* [23] developed Binary-Weight-Networks and XNOR-Networks that compress networks by approximating standard convolutional operations using binary operations. Denton *et al.* [10] exploited the redundancy present within the convolutional filters to derive approximations using Singular Vector Decomposition (SVD) that significantly reduce the computation and memory footprint. The second category is focused on designing and training small networks directly. For example, Iandola *et al.* [16] proposed a squeeze layer that only has 1×1 filters to re-design the network architecture. The generated smart network achieves AlexNet-level accuracy with 50x fewer parameters. MobileDeepPill follows the idea of the second category. It uses a number of strategies to design a new network architecture with a much smaller footprint and learns the parameters of the smart network via the knowledge distillation framework [15].

9. CONCLUSION

In this paper, we presented the design, implementation and evaluation of MobileDeepPill, a small-footprint mobile vision system for recognizing unconstrained pill images. Based on a novel multi-CNNs architecture, MobileDeepPill achieves state-of-the-art pill image recognition performance with 52.7% Top-1 accuracy and 81.7% Top-5 accuracy in one-side pill recognition scheme as well as 73.7% Top-1 accuracy and 95.6% Top-5 accuracy in two-side pill recognition scheme. By leveraging a novel Knowledge Distillation-based deep model compression framework, MobileDeepPill only requires 34MB runtime memory to run the multi-CNNs model and is able to perform low-power, near real-time pill image recognition on commodity smartphones without cloud offloading. With the support of high-end mobile GPU, the runtime performance is significantly improved from 1.58s to 270ms.

10. ACKNOWLEDGEMENT

We would like to thank the National Library of Medicine (NLM) of the National Institutes of Health (NIH) for organizing the Pill Image Recognition Challenge and providing the pill image dataset. We are also grateful to the anonymous MobiSys reviewers for their valuable reviews and insightful comments. This research was partially funded by NSF awards #1565604 and #1617627.

11. REFERENCES

[1] Amazon Shopping Mobile App. https://itunes.apple.com/us/app/amazon-app-shop-scan-compare/id297606951?mt=8.

[2] Google Translate. https://play.google.com/store/apps/details?id=com.google.android.apps.translate&hl=en.

[3] Monsoon Power Monitor. https://www.msoon.com/LabEquipment/PowerMonitor/.

[4] NIH NLM Pill Image Recognition Challenge. https://pir.nlm.nih.gov/challenge/.

[5] PWRcheck DC power analyzer. http://www.westmountainradio.com/product_info.php?products_id=pwrcheck.

[6] Your Prescriptions and Your Privacy. https://www.privacyrights.org/consumer-guides/your-prescriptions-and-your-privacy-california-medical-privacy-series.

[7] S. Bhattacharya and N. D. Lane. Sparsification and separation of deep learning layers for constrained resource inference on wearables. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, pages 176–189. ACM, 2016.

[8] J. J. Caban, A. Rosebrock, and T. S. Yoo. Automatic identification of prescription drugs using shape distribution models. In *2012 19th IEEE International Conference on Image Processing*, pages 1005–1008. IEEE, 2012.

[9] T. Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan. Glimpse: Continuous, real-time object recognition on mobile devices. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 155–168. ACM, 2015.

[10] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems*, pages 1269–1277, 2014.

[11] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, pages 1135–1143, 2015.

[12] S. Han, H. Shen, M. Philipose, S. Agarwal, A. Wolman, and A. Krishnamurthy. Mcdnn: An approximation-based execution framework for deep stream processing under resource constraints. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '16, pages 123–136, New York, NY, USA, 2016. ACM.

[13] K. He and J. Sun. Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5353–5360, 2015.

[14] 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*, pages 770–778, 2016.

[15] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[16] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 1mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[17] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

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

[19] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, L. Jiao, L. Qendro, and F. Kawsar. Deepx: A software accelerator for low-power deep learning inference on mobile devices. In *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 1–12. IEEE, 2016.

[20] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[21] Y.-B. Lee, U. Park, A. K. Jain, and S.-W. Lee. Pill-id: Matching and retrieval of drug pill images. *Pattern Recognition Letters*, 33(7):904–910, 2012.

[22] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[23] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. *arXiv preprint arXiv:1603.05279*, 2016.

[24] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

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

[26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[27] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014.

[28] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014.