ThumbNet: One Thumbnail Image Contains All You Need for Recognition

Chen Zhao  
chen.zhao@kaust.edu.sa  
King Abdullah University of Science and Technology (KAUST), Saudi Arabia

Bernard Ghanem  
bernard.ghanem@kaust.edu.sa  
King Abdullah University of Science and Technology (KAUST), Saudi Arabia

ABSTRACT

Although deep convolutional neural networks (CNNs) have achieved great success in computer vision tasks, its real-world application is still impeded by its voracious demand of computational resources. Current works mostly seek to compress the network by reducing its parameters or parameter-incurred computation, neglecting the influence of the input image on the system complexity. Based on the fact that input images of a CNN contain substantial redundancy, in this paper, we propose a unified framework, dubbed as ThumbNet, to simultaneously accelerate and compress CNN models by enabling them to infer on one thumbnail image. We propose three effective strategies to train ThumbNet. In doing so, ThumbNet learns an inference network that performs equally well on small images as the original-input network on large images. With ThumbNet, not only do we obtain the thumbnail-input inference network that can drastically reduce computation and memory requirements, but also we obtain an image downscaler that can generate thumbnail images for generic classification tasks. Extensive experiments show the effectiveness of ThumbNet, and demonstrate that the thumbnail-input inference network learned by ThumbNet can adequately retain the accuracy of the original-input network even when the input images are downscaled 16 times.

KEYWORDS

Image recognition, neural networks, network acceleration, autoencoder, knowledge distillation

ACM Reference Format:
Chen Zhao and Bernard Ghanem. 2020. ThumbNet: One Thumbnail Image Contains All You Need for Recognition. In Proceedings of the 28th ACM International Conference on Multimedia (MM ’20), October 12–16, 2020, Seattle, WA, USA. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394171.3413937

1 INTRODUCTION

Recent years have witnessed not only the growing performance of deep convolutional neural networks (CNNs) [13, 35, 38, 39, 47, 50, 51], but also their expanding computation and memory costs [3]. Though the intensive computation and gigantic resource requirements are somewhat tolerable in the training phase thanks to the powerful hardware accelerators (e.g. GPUs), when deployed in real-world systems, a deep model can easily exceed the computing limit of hardware devices. Mobile phones and tablets, which have constrained power supply and computational capability, are almost intractable to run deep networks in real-time. A cloud service system, which needs to respond to thousands of users, has an even more stringent requirement of computing latency and memory. Therefore, it is of practical significance to accelerate and compress CNNs for test-time deployment.

Before delving into the question of how to speed up deep networks, let us first analyze what dominates the computational complexity of a CNN. We calculate the total time complexity of convolutional layers as follows [12]:

\[ O\left( \sum_{l=1}^{d} n_{l-1} \cdot s_l^2 \cdot n_l \cdot m_l^2 \right) \],

(1)

where \( l \) is the index of a layer and \( d \) is the total number of layers; \( n_{l-1} \) is the number of input channels and \( n_l \) is the number of filters in the \( l \)-th layer; \( s_l^2 \) is the spatial size of the filters and \( m_l^2 \) is the spatial size of the output feature map.

Decreasing any factor in Eq. (1) can lead to reduction of total computation. One way is to sparsify network parameters by filter pruning [11, 23], which by defining some mechanism to prioritize the parameters, sets unimportant ones to zero. However, some researchers claim that these methods usually require sparse BLAS
libraries or even specialized hardware. Hereby, they propose to prune filters as a whole [25, 28]. Another method is to decrease the number of filters by low rank factorization [8, 20, 22, 44]. If a more dramatic change in network structure is required, knowledge distillation [16, 34] can do the trick. It generates a new network, which can be narrower (with fewer filters) or shallower (with fewer layers), by transferring hidden information from the original network. Moreover, there are other approaches to lower the convolution overload by means of fast convolution techniques (e.g. FFT [30] and the Winograd algorithm [21]), and quantization [11, 37] and binarization [6, 33].

All the above methods attempt to accelerate or compress neural networks from the viewpoint of network parameters, thereby neglecting the significant role that the spatial size of feature maps are playing in the overall complexity. According to Eq. (1), the required computation is diminished as the spatial size of feature maps decreases. Moreover, the memory required to accommodate those feature maps at run-time will also be reduced. Given a CNN architecture, we can simply decrease the spatial size of all feature maps by reducing the size of the input image.

In this paper, we propose to use a thumbnail image, i.e., an image of lower spatial resolution than its original-size counterpart, as test-time network input to accelerate and compress CNNs of any architecture and of any depth and width. This thumbnail-input network can dramatically reduce computation as well as memory requirements, as shown in Fig. 1.

Contributions. (1) We propose an orthogonal mechanism to accelerate a deep network compared to conventional methods: from the novel perspective of enabling the network to infer on one single downscaled image efficiently and effectively. To this end, we propose a unified framework called ThumbNet to train a thumbnail-input network that can tremendously reduce computation and memory consumption while maintaining the accuracy of the original-input network. (2) We present a supervised image downscaler that generates a thumbnail image with good discriminative properties and a natural chromatic look. This downscaler is reliably trained by exploiting supervised image downscaling, distillation-boosted supervision, and feature-mapping regularization. The ThumbNet generated images can replace their original-size counterparts and be stored for other classification-related tasks, reducing resource requirements in the long run. (3) The proposed ThumbNet effectively preserves network accuracy at speedup ratios of up to 4× (Imagenet) and 16× (Places) on various networks, surpassing other network acceleration/compression methods by significant margins.

2 RELATED WORK
In this section, we give an overview of the related works to our proposed ThumbNet in the literature.

2.1 Knowledge Distillation
Knowledge Distillation (KD) [16] was introduced as a model compression framework, which aims to reduce the computational complexity of a deep neural network by transferring knowledge from its original architecture (teacher) to a smaller one (student). The student is penalized according to the discrepancy between the softened versions of the teacher’s and student’s output logits\(^1\). It claims that this teacher-student paradigm easily transfers the generalization capability of the teacher network to the student network in that the student not only learns the characteristics of the correct labels but can also benefit from the invisible finer structure in the wrong labels. There are some extensions to this work, e.g., using intermediate representations as hints to train a thin-and-deep student network [34], applying it in object detection models [4], and using it to enhance network resilience to adversarial samples [32]. These works mostly focus on learning a new network architecture. In our paper, we utilize the idea of KD to train the same network architecture with thumbnail images as input.

2.2 Auto-Encoder
An auto-encoder [17] is an unsupervised neural network, which learns a data representation of reduced dimensions by minimizing the difference between input and output. It consists of two parts, an encoder which maps the input to a latent feature, and a decoder which reconstructs the input from the latent feature. Early auto-encoders are mostly composed of fully-connected layers [2, 17]. These days, with the popularity of CNNs, some researchers propose to incorporate convolution to an auto-encoder and design a convolutional auto-encoder [29, 36], which utilizes convolution / pooling to downscale the image in the encoder and utilizes deconvolution [41] / unpooling in the decoder to restore the original image size. Though the downscaled images from the encoder are effective for reconstructing the original images, they do not perform well for classification due to lack of supervision. In our work, instead of using convolutional auto-encoder as a downscaler, we incorporate it into ThumbNet as unsupervised pre-training to regularize the classification task.

2.3 Downscaled Image Representation
Representing an image with a downscaled size is an effective way to reduce computational complexity. One emerging method is to apply compressive sensing [42, 43, 45–47, 49, 51] when acquiring an image to sample only a small number of measurements. But these measurements are not in the format of 2D grid and cannot be directly processed by a CNN. Alternatively, the work WIDIC [48] proposes to downscale an image into a smaller image in the wavelet domain, so as to increase its coding efficiency without compromising reconstruction quality. The proposed ThumbNet also preserves the image grid when downscaling but in a learnable manner. It focuses on the classification task and aims to increase network efficiency instead. For the task of classification, we find one recent work [5] (denoted here as LWAЕ), which also attempts to accelerate a neural network by using small images as input. It decomposes the original input image into two low-resolution sub-images, one with low frequency which is fed into a standard classification network, and one with high frequency which is fused with features from the low-frequency channel by a lightweight network to obtain the classification results. Compared to LWAЕ, our ThumbNet is able to achieve higher network accuracy with\(^1\)In this paper, we use ‘logits’ to refer to the output of a neural network before the softmax activation function in the end.
one single thumbnail-image, resulting in fewer requirements for computation, memory and storage.

3 PROPOSED THUMBNET

3.1 Network Architecture

We illustrate the architecture of ThumbNet in Fig. 2. The well-normalization and the rectified linear unit (ReLU). In the first layer, not be destroyed or misaligned. That is to say, if pixel values in natural images follow a distribution, then the generated small image should remain visually pleasant and all the nested operations in our downscaler as \( \mathcal{E} \) operations and high-dimensional feature projection. By denoting only adaptively trains the filters, but also incorporates non-linear scaling ratio. Compared to bicubic, this learnable downscaler not channel. The stride of each layer depends on the required down-layer there are exactly 3 output channels to restore the image color.

3.2 Details of Network Design

There are three main techniques in the proposed ThumbNet: 1) supervised image downscaling, 2) distillation-boosted supervision, and 3) feature-mapping regularization.

3.2.1 Supervised Image Downscaling

Traditional image downsampling methods (e.g., bilinear and bicubic [24]) do not consider the discriminative capability of the downscaled images, which as a consequence lose critical information for classification. We instead exploit CNNs to adaptively extract discriminative information from the original images to tailor to the classification goal.

For the sake of computational efficiency and simplicity, our supervised image downscaler \( E \) merely comprises two convolutional layers, each with a \( 5 \times 5 \) convolutional operation followed by batch normalization and the rectified linear unit (ReLU). In the first layer, there are more output channels than input channels to empower the network to learn more intermediate features, and in the second layer there are exactly 3 output channels to restore the image color channel. The stride of each layer depends on the required down-scaling ratio. Compared to bicubic, this learnable downscaler not only adaptively trains the filters, but also incorporates non-linear operations and high-dimensional feature projection. By denoting all the nested operations in our downscaler as \( E \), we obtain a small image \( y \) from the original image \( x \) via the following:

\[
y = E(x; W_E).
\]

A significant consideration in designing this downscaler is that the generated small image should remain visually pleasant and recognizable, e.g., the information in the color channels should not be destroyed or misaligned. That is to say, if pixel values in natural images follow a distribution, then the generated small image should follow the same distribution with similar moments. Hereby, we propose a moment-matching (MM) loss as follows:

\[
\mathcal{L}_{MM}(W_E) = \frac{1}{3} \| \mu(x) - \mu(y) \|_2^2 + \lambda \frac{1}{3} \| \sigma(x) - \sigma(y) \|_2^2,
\]

where \( \mu(\cdot) \) and \( \sigma(\cdot) \) compute the first and second moments respectively of the image pixel values in each color channel, and \( \lambda \) is a tunable parameter that balances the two moments. This MM loss encourages that the mean and variance (loosely approximating the distribution) in each color channel of the downscaled image stay close to the mean and variance of the original image. This has been used in other application as well in the literature, such as deep generative model learning [26] and style transfer [27].

Please note that this downscaler is not trained independently with merely the MM loss, but incorporated into the whole architecture of ThumbNet and trained together with other components and losses. This includes the classification loss, which provides supervision to the image downsampling process, guiding it to generate a small image that is discriminative for accurate classification. Hence, we name \( E \) a supervised image downscaler. Once trained, this downscaler can be used for generating small images, which are not only used as input of the inference network, but also for other classification-related tasks as well.

3.2.2 Distillation-Boosted Supervision

A straightforward way to train the network is to minimize the classification (CL) loss defined as follows:

\[
\mathcal{L}_{CL}(W_E, W_S) = \mathcal{H}(b, S(y; W_S)),
\]

where \( y \) is calculated as in Eq. (2), \( b \) indicates the ground-truth labels, \( S(\cdot) \) denotes all the nested functions in the inference network \( S \), and \( \mathcal{H} \) refers to cross entropy. This loss seeks to match the predicted label with the ground-truth label and is a typical cost function for supervised learning. However, only using this loss cannot exploit the information embedded in the well-trained model \( T \). To address this issue, we propose to distill the learned knowledge in \( T \) and transfer it to the inference network. Therefore, apart from the CL loss, we also enforce the computed probabilities of each class in the inference network to match those in the well-trained network.

Let \( S_0(\cdot) \) and \( T_0(\cdot) \) denote the deep nested functions before softmax in the inference network and the well-trained network, respectively. Then, their logits are calculated as:

\[
as_S = S_0(y; W_S), \quad a_T = T_0(x).
\]

Following [16], we define the knowledge-distillation (KD) loss as the cross entropy between the two softened probabilities:

\[
\mathcal{L}_{KD}(W_E, W_S) = \mathcal{H} \left( \text{softmax} \left( \frac{a_S}{\tau} \right), \text{softmax} \left( \frac{a_T}{\tau} \right) \right),
\]

where \( \tau \) is the temperature to soften the class probabilities and is usually greater than 1. With the aid of this KD loss, the supervised training process of ThumbNet can benefit from the well-trained model to learn finer discriminative structures, so we call it distillation-boosted supervision.

3.2.3 Feature-Mapping Regularization

It is widely observed that unsupervised pre-training, e.g., using an auto-encoder, can help with supervised learning tasks as a form of regularization [9]. Inspired by this, we design the feature-mapping (FM) regularization to pre-train ThumbNet.
Figure 2: ThumbNet Architecture. Green: well-trained network $T$; red: inference network $S$; blue: downscaler $E$. Blocks represent feature maps, solid arrows contain network operations such as convolution, the rectified linear unit (ReLU) [31], pooling, batch normalization [19], etc. The numbers below each feature map are their spatial resolutions. Dots represent the four losses, which are moment-matching (MM) loss in brown, feature-mapping (FM) loss in yellow, knowledge-distillation (KD) loss in cyan and classification (CL) loss in purple.

**Algorithm 1 Training Strategy of ThumbNet**

The well trained parameters $W_T$ of the original network are provided as input to the algorithm. All the trainable parameters are initialized as random values which are denoted by $W^0_E, W^0_S, W^0_D$.

1: **Input:** $W_T, W^0_S, W^0_E, W^0_D$

2: $W^*_E, W^*_S, W^*_D \leftarrow \arg \min_{W^0_E, W^0_S, W^0_D} \mathcal{L}_{MM}(W_E) + \alpha \mathcal{L}_{FM}(W_E, W_S^l, W_D) + \frac{1}{2} \theta R_{E,S,D}$

3: $W^*_E, W^*_S \leftarrow \arg \min_{W_E, W_S} \mathcal{L}_{CL}(W_E, W_S) + \beta \mathcal{L}_{KD}(W_E, W_S) + \frac{1}{2} \theta R_{E,S}$

4: **Output:** $W^*_E, W^*_S$

In Fig. 2, ThumbNet is partitioned into two segments by the FM loss (the yellow dot). In order to give a clearer sense of the rationale, we re-illustrate the left segment of ThumbNet along with the FM loss from a different point of view in Fig. 3 (note that the MM loss is left out and the deconvolution in the FM loss is unrolled for the sake of clarity). We can see that it is analogous to an auto-encoder, which is trained by minimizing the difference between the pre-processed input and the output.

In the decoder, each deconvolutional layer has a stride size 2 and the number of layers is determined by the downsampling ratio. The decoder does not change the number of channels but upscales the feature map in $S$ to match the spatial size of the corresponding feature map in $T$. We compute their mean square error as the FM loss:

$$
\mathcal{L}_{FM}(W_E, W^l, W_D) = \frac{1}{2N} \left\| \mathcal{T}_l(x) - \mathcal{D}(S_l(y; W^l); W_D) \right\|_2^2,
$$

where $N$ is the product of all dimensions of the intermediate feature map in $T$. $\mathcal{T}_l$ represents the nested functions in the left segment of the original network, $S_l$ represents the nested functions in the left segment of the inference network, $\mathcal{D}$ represents operations in the deconvolutional layers, and $W^l_l$ and $W^l_D$ refer to the parameters in the left segment of $S$ and in the deconvolutional layers, respectively.

Figure 3: Feature-mapping regularization. This re-illustrates the left segment of ThumbNet along with the FM loss, which is essentially an auto-encoder. The layers in the red dashed box constitute the encoder, the layers in the yellow dashed box constitute the decoder, and the feature map in the middle is learned latent representation. The shaded area can be viewed as a pre-processing for the input. The yellow solid arrow represents deconvolutional layers and the yellow box is the upscaled feature map.

### 3.3 Training Details

We summarize the training strategy of ThumbNet in Algorithm 1. In Algorithm 1, the $R$ terms refer to $l_2$ regularization of the corresponding parameters, and $\theta, \alpha, \beta$ are the tradeoff weights. In Line 2, we perform an unsupervised pre-training by minimizing the
MM loss and the FM loss, and obtain the parameters $W^*_E$ and $W^*_S$. In Line 3, we perform the distillation-boosted supervised learning by minimizing the KD loss and CL loss, using the values $W^*_E$ and $W^*_S$ as initialization for the corresponding parameters. In this step, we train the whole ThumbNet end-to-end to learn the optimal values for the parameters $W^*_E$ and $W^*_S$. Note that the parameters $W^*_E$ and $W^*_S$ are already pre-trained via Line 2, so they are only finetuned with a small learning rate. In contrast, we use a relatively large learning rate for the untrained parameters $W_S$, in the right segment. We set the hyper-parameters in ThumbNet as follows. We use a starting learning rate 0.1, and divide it by 10 when the loss plateaus. We use a momentum 0.9 for the optimizer and the weight decay $\theta$ is 0.0001. The parameters $\alpha$ and $\beta$ in Algorithm 1 are 1.0 and 0.5, respectively; $\tau$ in Eq. (6) is 2; $\lambda$ in Eq. (3) is 0.1. For finetuning the pre-trained parameters $W_E$ and $W_S$, the learning rate is set to 0.01 times that of the other parameters, meaning their learning rate starts from 0.001 and is decreased by 10 in the same fashion.

4 EXPERIMENTS

In this section, we demonstrate the performance of the inference network $S$ obtained via ThumbNet in terms of classification accuracy and resource requirements. We also provide experiments to verify that the learned downscaler has generic applicability to other classification-related tasks as well.

4.1 Ablation Study

In order to study the effectiveness of the proposed techniques in ThumbNet, we ablate each technique individually and test the performance of the reduced version of ThumbNet. We provide experimental results on the object recognition and scene recognition tasks using different backbone networks (e.g., Resnet-50, VGG-11).

Table 1: Configuration of different methods. (a) is the baseline, (b) does not use any of the three techniques, (c)-(e) contain one or two, and (f) consists of all of them.

| Methods               | SD | KD | FM |
|-----------------------|----|----|----|
| (a) Original / Direct | ✓  | ✓  | ✓  |
| (b) Bicubic downscaler| ×  | ✓  | ×  |
| (c) Supervised downscaler| ✓  | ✓  | ×  |
| (d) Bicubic + distillation| ✓  | ✓  | ✓  |
| (e) Supervised + distillation| ✓  | ✓  | ✓  |
| (f) ThumbNet          | ✓  | ✓  | ✓  |

4.1.1 Baseline and Comparative Methods. We implement six variations of ThumbNet for training a backbone network, which are introduced in the following.

(a) Original / Direct. ‘Original’ refers to the network model trained on the original-size images. The testing performance of the model on an original-size image is the upperbound baseline for all the comparative methods. For networks like Resnet, which use global average pooling instead of fully-connected layers at the end, the ‘Original’ model can be directly used to test on a small image without altering the network structure. We refer to the case of directly using the ‘Original’ model for inference on small images without re-training as ‘Direct’, which is the lowerbound baseline for all the methods.

(b) Bicubic downscaler. This trains the network from scratch on images that are downscaled with the bicubic method.

(c) Supervised downscaler. This trains the network from scratch on small images that are downscaled with the supervised downscaler in ThumbNet. The downscaler and the network are trained jointly end-to-end based on the MM loss and the CL loss.

(d) Bicubic + distillation. This trains the network on bicubic-downscaled small images with the aid of distillation from the ‘Original’ model. The network is trained based on the KD loss and the CL loss.

(e) Supervised + distillation. This trains the network on supervised-downscaled small images with the aid of distillation from the ‘Original’ model. The supervised downscaler and the network are trained jointly end-to-end based on the MM loss, the KD loss and the CL loss.

(f) ThumbNet. This is the full configuration of our proposed method, which is trained on supervised-downscaled small images with the aid of distillation from the ‘Original’ models as well as feature mapping regularization. It is trained based on the four losses as described in Section 3.3.

In Table 1, we demonstrate the configuration of each method with respect to the three techniques used in ThumbNet: supervised image downscaling (SD), knowledge-distillation boosted supervision (KD) and feature mapping regularization (FM). The hyper-parameters in all the methods are the same, as specified in Section 3.3.

Table 2: Error rates (%) for scene recognition on Places36. ThumbNet retains the accuracy of the original-input network when downsampling the image 16 times.

| Methods               | Resnet-18 | VGG-11 |
|-----------------------|-----------|--------|
| (a) Original          | 21.11     | 3.28   |
| (b) Direct            | 66.94     | 33.97  |
| (b) Bicubic downscaler| 32.08     | 7.83   | 27.83  | 10.17 |
| (c) Supervised downscaler| 25.94 | 5.31   | 24.28  | 6.58  |
| (d) Bicubic + distillation| 26.00 | 4.47   | 22.69  | 4.92  |
| (e) Supervised + distillation| 24.33 | 3.94   | 21.31  | 3.94  |
| (f) ThumbNet          | 22.78     | 3.69   | 21.58  | 3.72  |

4.1.2 Object Recognition on ImageNet Dataset. We evaluate the performance of our proposed ThumbNet on the task of object recognition with the benchmark dataset ILSVRC 2012 [7], which consists of over one million training images drawn from 1000 categories. Besides using the full dataset (referred to as ImagenetFull), we also form a smaller new dataset by randomly selecting 100 categories from ILSVRC 2012 and refer to it as Imagenet100 (the categories in Imagenet100 are given in the supplementary material). With Imagenet100, we can efficiently compare all the methods on a variety of backbone networks. In addition, by partitioning ILSVRC 2012 into two parts (Imagenet100 and the rest 900 categories referred to as Imagenet900), we can also evaluate the downsampler on unseen data categories (see Section 4.3 for details). For backbone networks,
we consider various architectures (ResNet [13] and VGG [35]) and various depths (from 11 layers to 50 layers).

In Table 3, we demonstrate the performance of all the methods with four different backbone networks in terms of top-1 and top-5 error rates on the validation data. The input image size of ‘Original’ is 224 × 224 and the input image sizes of the other methods are 112 × 112, meaning that in this experiment, the image downsampling ratio is 4 : 1. Thus, compared to ‘Original’, our ThumbNet only uses 1/4 computation and memory, which will be detailed in Section 4.1.4, and it also preserves the accuracy of the original models. ‘Direct’ is a baseline of inference on small images, compared to which our ThumbNet improves by large margins for all the different backbones. Moreover, by comparing different pairs of methods, we can obviously observe the contribution of each technique. By comparing (c) to (b) or comparing (e) to (d), we can see the benefits of supervised image downsampling. By comparing (e) to (c), we can see the benefits of distillation-boosted supervision. By comparing (f) to (e), we can see that the benefits of feature-mapping regularization.

(f) ThumbNet always shows the lowest error rates.

4.1.3 Scene Recognition on Places Dataset. We also apply our proposed ThumbNet to the task of scene recognition using the benchmark dataset Places365-Standard [52]. This dataset consists of 1.8 million training images from 365 scene categories. We randomly select 36 categories from Places365-Standard as our new dataset Places36 (the chosen categories are given in the supplementary material).

In Table 2, we report the error rates on the Places36 validation dataset using two backbone networks Resnet-18 and VGG-11. The input image size of ‘Original’ is 224 × 224 and the input image sizes of the other methods are 56 × 56, meaning that in this experiment, the image downsampling ratio is 16 : 1. Thus, compared to ‘Original’, our ThumbNet only uses 1/16 computation and memory, which will be detailed in Section 4.1.4. In terms of recognition accuracy, ThumbNet nearly preserves the accuracy of the original models, where the top-5 accuracy drops by only 0.41% for Resnet-18 and by only 0.11% for VGG-11. Compared to ‘Direct’, our ThumbNet improves significantly, by 44.16% for top-1 accuracy and 30.28% for top-5 accuracy.

4.1.4 Resource Consumption. To evaluate the test-time resource consumption of the networks, we measure their number of FLoat point OPerations (FLOPs) and memory consumption of their feature maps. Fig. 4 plots the number of FLOPs required by each method to classify one image for object recognition on Imagenet100 and scene recognition on Places36 with the two backbone networks Resnet-18 and VGG-11. For the task of object recognition, in which the images are downscaled 4 times, the small-input networks use only 1/4 FLOPs compared to the original models. For the task of scene recognition, in which the images are downscaled 16 times, the small-input networks use only 1/16 FLOPs compared to the original models.

Similar to the reduction of computation, the memory consumption of the 1/4-input models is about 1/4 of the original models, and the memory consumption of the 1/16-input models is about 1/16 of the original models.

4.2 Comparison to State-of-the-Art Methods

4.2.1 Comparison with LWAE. We compare our ThumbNet to LWAE [5] in terms of classification accuracy and resource requirements. Considering that the trained models of LWAE provided by the authors use different backbone networks on different datasets from ours, we re-implement LWAE in Tensorflow [1] (the same as our ThumbNet) with the same backbone networks on the same datasets as ours for fair comparison. We follow the instructions in the paper for setting the hyper-parameters and training LWAE. To evaluate both methods, we test their inference accuracy in terms of top-1 and top-5 error rates, and their inference efficiency in terms of number of FLOPs, number of network parameters, memory consumption of feature maps, and storage requirements for input images.

Table 4 demonstrates the results of testing a batch of 224 × 224 color images using the two methods as well as the benchmark networks for the four tasks: object recognition on Imagenet100 with VGG-11 and Resnet-18, scene recognition on Places36 with VGG-11 and Resnet-18. For the object recognition tasks, the images are downscaled by 4 in LWAE and ThumbNet; and for the scene recognition tasks, the images are downscaled by 16 in LWAE and ThumbNet. The batch size in these experiments is set to 32.

Seen from Table 4, ThumbNet has an obvious advantage over LWAE in terms of efficiency owing to its one single input image and one single network for inference. ThumbNet only computes one network, whereas LWAE has to compute an extra branch for fusing the high-frequency image. Therefore, when downsampling an image by the same ratio, ThumbNet uses fewer FLOPs, has fewer network parameters, requires less memory for the intermediate feature maps, and stores only one small image (as compared to two images for LWAE). Regarding classification accuracy, ThumbNet has lower top-1 and top-5 errors in the first two tasks. For the third task, ThumbNet has an obviously lower top-5 error and roughly the same top-1 error as LWAE. For the fourth task, LWAE is slightly better than ThumbNet at the cost of higher computational complexity and memory usage.
Table 3: Error rates (%) for object recognition on Imagenet. By comparing (c) to (b) or comparing (e) to (d), we can see the benefits of supervised image downscaling. By comparing (e) to (c), we can see the benefits of distillation-boosted supervision. By comparing (f) to (e), we can see that the benefits of feature-mapping regularization. (f) ThumbNet always shows the lowest error rates.

| Tasks          | Methods               | Error rates | FLOPs  | Parameters | Feature Memory | Image Storage |
|----------------|-----------------------|-------------|--------|------------|----------------|---------------|
|                |                       | Top-1       | Top-5  | # (B)      | ↓ rate         | Size (MB)     | ↓ rate        |
| Imagenet100    | VGG-11                | 13.64       | 4.36   | 15.06      | 4.56           | 12.72         | 3.50          |
|                | Resnet-34             | 17.54       | 4.98   | 15.06      | 4.56           | 12.72         | 3.50          |
|                | Resnet-50             | 17.54       | 4.98   | 34.38      | 14.32          | 29.42         | 11.80         |
|                |                       | 21.53       | 7.06   | 18.44      | 6.16           | 16.84         | 5.66          |
| Places36       | Resnet-18             | 17.54       | 4.98   | 15.06      | 4.56           | 12.72         | 3.50          |
|                |                       | 17.53       | 4.98   | 34.38      | 14.32          | 29.42         | 11.80         |
|                |                       | 21.53       | 7.06   | 18.44      | 6.16           | 16.84         | 5.66          |
|                |                       | 21.53       | 7.06   | 34.38      | 14.32          | 29.42         | 11.80         |

Table 4: Accuracy and efficiency comparison with state-of-the-art. B: billion; M: million; MB: MegaBytes. Regarding efficiency, ThumbNet has an obvious advantage over LWAE in terms of all metrics and for all tasks, i.e., ThumbNet requires fewer FLOPs, fewer network parameters, less memory feature maps, and less storage for images. Regarding classification accuracy, ThumbNet has lower top-1 and top-5 errors in the first two tasks. For the third task, ThumbNet has an obviously lower top-5 error and roughly the same top-1 error as LWAE. For the fourth task, LWAE is slightly better than ThumbNet at the cost of higher computational complexity and memory usage.

Table 5: ThumbNet compared with representative network compression methods on Imagenet

| Tasks          | Methods | Top-1 error | Top-5 error | FLOPs |
|----------------|---------|-------------|-------------|-------|
|                |         | Err. %       | Err. %       |        |
|                |         | ↑           | ↑           | # (B) |
| ImagenetFull   | Resnet-50 | 21.67       | 11.22       | 15.13 |
|                | SSS [18]  | 21.06       | 11.22       | 15.13 |
|                | SFP [14]  | 21.06       | 11.22       | 15.13 |
|                | CP [15]   | 21.06       | 11.22       | 15.13 |
|                | LWAE [5]  | 8.25        | 4.62        | 22.78 |
|                | ThumbNet  | 8.25        | 4.62        | 22.78 |

4.2.2 Comparison with representative network acceleration / compression methods. Table 5 compares ThumbNet to representative network acceleration/compression methods in recent literature on ImagenetFull/Resnet-50. The results of the comparative methods and their respective baselines are directly taken from the published papers. We can see that the methods SSS [18], SFP [14] and CP [15] can only speed up by up to 2 times, which is much lower than ThumbNet. The methods ThiNet [28], DCP [53] and Slimmable [40] have similar speedup ratios (still lower than) to ThumbNet, but they have obviously higher increase in error rates. Compared to these methods, ThumbNet has obviously lower increase in error rates with even higher acceleration ratios. Moreover, since ThumbNet is orthogonal to conventional acceleration/compression methods, it can be used in conjunction with them for further speedup.

4.3 Evaluation of the Supervised Downscaler

4.3.1 Visual Comparison of Downscaled Images. In Fig. 5, we show examples of the generated thumbnail images by our ThumbNet downscalers compared to other different methods for the task of object recognition on Imagenet100 with Resnet-18. We can see that the images of ‘Supervised downscaler’, ‘Supervised + distillation’, ThumbNet, and ‘W/o MM’ (ThumbNet without the MM loss) have noticeable edges compared to the ‘Bicubic’ one. This is because these downscalers are trained in a supervised way with the classification loss being considered. The edge information in the
downscaled images is helpful for making discriminative decisions. Also, note that the thumbnail image of ThumbNet is natural in color owing to the MM loss, whereas the image of 'W/o MM' is obviously yellowish because the color channels are more easily messed up when the MM loss is not utilized. The top-1 error rates of ThumbNet and 'W/o MM' are 17.32% and 17.90%, respectively, and their top-5 error rates are 4.98% and 5.26%, respectively. This shows that adding the MM loss does not deteriorate the network accuracy but produces more pleasant small images, which contain generic information that also benefits other tasks.

4.3.2 Does the Supervised Downscaler Generalize? In order to verify that the ThumbNet downscaler is also useful apart from being used in the specific inference network T, we also consider three different scenarios for applying our learned downscaler. Suppose that the downscaler is trained on the dataset A with the backbone network F. We have a new dataset A\text{new} and a different network F\text{new}. The three scenarios for applying the well-trained downscaler are as follows: (1) downscaling A to generate small images to train F\text{new}; (2) downscaling A\text{new} to generate small images to train F; and (3) downscaling A\text{new} to generate small images to train F\text{new}.

Table 6 reports the results of these scenarios using the downscaler learned from the object recognition task with Resnet-18, which downscales an image by 4. In this case, the dataset A is Imagenet100; the network F is Resnet-18. We use Imagenet900 and Caltech256 [10] as the new dataset A\text{new}, and VGG-11 as the new network F\text{new}. The first two columns correspond to Scenario (1); the third and fourth columns correspond to Scenario (2); the last two columns correspond to Scenario (3), where we use the downscaler to generate thumbnail images for the new dataset Caltech256, and use these thumbnail images to train a different network VGG-11.

Table 6: Performance of the ThumbNet downscaler on different networks and datasets.

| Networks/Datasets | VGG-11/Imagenet100 | Resnet-18/Imagenet900 | VGG-11/Caltech256 |
|-------------------|--------------------|-----------------------|-------------------|
|                   | Top-1 | Top-5 | Top-1 | Top-5 | Top-1 | Top-5 |
| Original          | 13.64 | 4.36  | 28.80 | 10.17 | 30.68 | 18.56 |
| Bicubic           | 18.20 | 6.60  | 35.52 | 14.56 | 32.80 | 19.63 |
| ThumbNet          | 16.74 | 6.42  | 33.54 | 13.22 | 32.04 | 18.56 |

The first row shows the performance of the respective networks trained on the original-size images, while the second row shows their performance when trained on the small images downsized with bicubic interpolation. The third row shows the performance of the networks trained on the small images downscaled by our ThumbNet downscaler. We can see that the third row obviously outperforms the second row in all scenarios, indicating that our supervised downscaler tends to generalize to other datasets and other network architectures. In fact, it is very promising to see that in Scenario (3) when both the dataset and the network are new to the downscaler, it can still bring about significant gains compared to the bicubic naive downscaler, leading to the same top-5 error rate as the original network.

5 CONCLUSIONS

In this paper, we propose a unified framework ThumbNet to tackle the problem of accelerating run-time deep convolutional network from a novel perspective: downscaling the input image. Based on the fact that reducing the input image size lowers the computation and memory costs of a CNN, we seek to obtain a network that can retain original accuracy when applied on one thumbnail image. Experimental results show that, with our ThumbNet, we are able to learn a network that dramatically reduces resource requirements without compromising classification accuracy. Moreover, we have a supervised downscaler as a side product, which can be utilized for generic classification purposes, generalizing to datasets and network architectures that it was not exposed to in training. This work can be used in conjunction with other network acceleration/compression methods for further speed up without incurring additional overheads.

ACKNOWLEDGMENTS

This publication is based upon work supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research (OSR) under Award No. OSR-CRG2017-3405.

REFERENCES

[1] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. TensorFlow: A System for Large-Scale Machine Learning. In OSDI, Vol. 16. 265–283.

[2] Hervé Bourlard and Yves Kamp. 1988. Auto-association by multilayer perceptrons and singular value decomposition. Biological cybernetics 59, 4-5 (1988), 291–294.

[3] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. 2016. An analysis of deep neural network models for practical applications. arXiv preprint arXiv:1605.07678 (2016).

[4] Guobin Chen, Wengun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. 2017. Learning Efficient Object Detection Models with Knowledge Distillation. In Advances in Neural Information Processing Systems. 742–751.

[5] Tianshui Chen, Liang Lin, Wangmeng Zuo, Xiaonan Luo, and Lei Zhang. 2018. Learning a Wavelet-like Auto-Encoder to Accelerate Deep Neural Networks. In AAAI.
doi: 10.1109/MM.2020.3015200

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In CVPR09.

[2] Emily L. Denton, Viorica Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. 2014. Exploiting linear structure within convolutional networks for efficient evaluation. In Advances in neural information processing systems. 1269–1277.

[3] Damiu Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? Journal of Machine Learning Research 11, Feb (2010), 625–660.

[4] Gregory Griffin, Alex Holub, and Pietro Perona. 2007. Caltech-256 object category dataset. (2007).

[5] Song Han, Huizi Mao, and William J. Dally. 2016. Deep Compression: Compressing Deep Neural Networks with Pruning, Training Quantization and Huffman Coding. International Conference on Learning Representations ICLR abs/1510.00149 (2016).

[6] Kaiming He and Jian Sun. 2015. Convolutional neural networks at constrained time cost. In Proceedings of the IEEE conference on computer vision and pattern recognition. 770–778.

[7] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. 2018. Soft Filter Pruning for Accelerating Deep Convolutional Neural Networks. arXiv abs/1808.06866 (2018).

[8] Yihui He, Xiayu Zhang, and Jian Sun. 2017. Channel Pruning for Accelerating Very Deep Neural Networks. 2017 IEEE International Conference on Computer Vision (ICCV) (2017), 1398–1406.

[9] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015).

[10] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. Science 313, 5786 (2006), 504–507.

[11] Zehao Huang and Nanyan Wang. 2018. Data-Driven Sparse Structure Selection for Deep Neural Networks. In ECCV.

[12] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep neural network training by reducing internal covariate shift. In International conference on machine learning. 448–456.

[13] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. 2014. Speeding up convolutional neural networks with low rank expansions. arXiv preprint arXiv:1405.8866 (2014).

[14] Andrew Lavin and Scott Gray. 2016. Fast algorithms for convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 4013–4021.

[15] Vadim Lebedev, Yaroslav Ganin, Maximakh Khushba, Ivan Oseledets, and Victor Lempitsky. 2014. Speeding up convolutional neural networks using fine-tuned cp-decomposition. arXiv preprint arXiv:1412.6553 (2014).

[16] Yann LeCun, John S Denker, and Sara A Solla. 1990. Optimal brain damage. In Proceedings of the 1990 Workshop on Neural Information Processing Systems. 598–605.

[17] Thomas Martin Lehmann, Claudia Gonner, and Klaus Spitzer. 1999. Survey: Enhancing medical image quality with neural networks. International Journal of Imaging Systems and Technology 10, 6 (1999), 393–400.

[18] Andrew Lavin and Scott Gray. 2016. Fast algorithms for convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 4013–4021.

[19] Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. 2010. Deconvolutional networks. In Computer Vision and Pattern Recognition Conference (CVPR). 2010 IEEE Conference on. IEEE, 2528–2535.

[20] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. 2016. Discrimination-aware Channel Pruning. arXiv preprint arXiv:1610.01036 (2016).

[21] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. 2017. ThiNet: A Filter Level Pruning Framework for Deep Convolutional Neural Networks. arXiv preprint arXiv:1701.04949 (2017).

[22] Jiashu Xu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. 2019. Shmmable Neural Networks. In International Conference on Learning Representations. https://openreview.net/forum?id=B1gMCsAqY7

[23] Chen Zhao, Jian Zhang, Ronggang Wang, and Wen Gao. 2014. Image compressive sensing recovery using adaptively learned sparsifying basis via l0 minimization. Signal Process. 103 (2014), 114–126.

[24] Jian Zhang, Debin Zhao, Chen Zhao, Ruijun Xiong, Siwei Ma, and Wen Gao. 2012. Image Compressive Sensing Recovery via Collaborative Sparsity. IEEE Journal on Emerging and Selected Topics in Circuits and Systems 2 (2012), 380–391.

[25] Xinxi Zhang, Jianhua Zou, Kaiming He, and Jian Sun. 2016. Accelerating very deep convolutional networks for classification and detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 38, 10 (2016), 1943–1955.

[26] Chen Zhao, Siwei Ma, and Wen Gao. 2014. Image compressive-sensing recovery using structured laplacian sparsity in dct domain and multi-hypothesis prediction. 2014 IEEE International Conference on Multimedia and Expo (ICME) (2014), 16.

[27] Chen Zhao, Siwei Ma, Jian Zhang, Ruijun Xiong, and Wen Gao. 2017. Video Compressive Sensing Reconstruction via Reweighted Residual Sparsity. IEEE Transactions on Circuits and Systems for Video Technology 27 (2017), 1182–1195.

[28] Chen Zhao, Ronggang Wang, and Wen Gao. 2018. BoostNet: A Structured Deep Recursive Network to Boost Image Deblocking. In 2018 IEEE Visual Communications and Image Processing (VCIP). IEEE, 1–4.

[29] Chen Zhao, Jian Zhang, Siwei Ma, and Wen Gao. 2016. Nonconvex lp Nuclear Norm based ADMM Framework for Compressed Sensing. 2016 Data Compression Conference (DCC) (2016), 161–170.

[30] Chen Zhao, Jian Zhang, Ronggang Wang, and Wen Gao. 2018. CNN-Reguarized ADMM framework for compressive-sensed image reconstruction. IEEE Access 6 (2018), 76838–76853.

[31] Bolei Zhou, Agata Lapedriza, Adriana Khosla, Aude Oliva, and Antonio Torralba. 2017. Places: A 10 million image database for Scene Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 1718–1727.

[32] Yanghao Li, Nanyan Wang, Juying Liu, and Xiaodi Hou. 2017. Demystifying neural style transfer. arXiv preprint arXiv:1701.01036 (2017).

[33] Jian-Hao Luo, Jiaxin Wu, and Weiyao Lin. 2017. ThuNet: A Filter Level Pruning Method for Deep Neural Network Compression. 2017 IEEE International Conference on Computer Vision (ICCV) (2017), 5068–5076.

[34] Han Zhao, Jian Zhang, Ronggang Wang, and Wen Gao. 2018. BoostNet: A Structured Deep Recursive Network to Boost Image Deblocking. In 2018 IEEE Visual Communications and Image Processing (VCIP). IEEE, 1–4.

[35] Chen Zhao, Jian Zhang, Siwei Ma, and Wen Gao. 2013. Wavelet inpainting driven image compression via collaborative sparsity at low bit rates. In 2013 IEEE International Conference on Image Processing. IEEE, 1685–1689.

[36] Chen Zhao, Jian Zhang, Siwei Ma, and Wen Gao. 2016. Convolutional Neural Network meets CNN: compressive-sensed image reconstruction. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.

[37] Anna Wang, Xinyi Wang, and Wentao Gao. 2014. Nonconvex Lp Nuclear Norm based ADMM Framework for Compressed Sensing. IEEE Transactions on Image Processing 23, 12 (2014), 525–542.

[38] Chen Zhao, Jian Zhang, Siwei Ma, and Wen Gao. 2013. Wavelet inpainting driven image compression via collaborative sparsity at low bit rates. In 2013 IEEE International Conference on Image Processing. IEEE, 1685–1689.

[39] Chen Zhao, Jian Zhang, Siwei Ma, and Wen Gao. 2016. Nonconvex lp Nuclear Norm based ADMM Framework for Compressed Sensing. 2016 Data Compression Conference (DCC) (2016), 161–170.

[40] Chen Zhao, Jian Zhang, Ronggang Wang, and Wen Gao. 2018. CNN-Reguarized ADMM framework for compressive-sensed image reconstruction. IEEE Access 6 (2018), 76838–76853.

[41] Bolei Zhou, Agata Lapedriza, Adriana Khosla, Aude Oliva, and Antonio Torralba. 2017. Places: A 10 million image database for Scene Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 1718–1727.

[42] Zhuangwei Zhong, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Gou, Qinyao Wu, Junzhou Huang, and Jun-Hui Zhu. 2018. Discrimination-aware Channel Pruning for Deep Neural Networks. In NeurIPS.