Designing Lightweight Feature Descriptor Networks with Depthwise Separable Convolution

Yeo Ree Wang and Atsunori Kanemura

LeapMind Inc., Tokyo 150-0044, Japan
{yeo,atsu-kan}@leapmind.io

Abstract. Extracting feature points and their descriptors from images is one of the fundamental techniques in computer vision with many applications such as geometric fitting and camera calibration, and for this task several deep learning models have been proposed. However, existing feature descriptor networks have been developed with the intention of improving the accuracy, and consideration for practical networks that can run on embedded devices has somewhat been deferred. Therefore, the objective of this study is to devise light feature descriptor networks. To this end, we employ lightweight convolution operations that have been developed for image classification networks (e.g. SqueezeNet and MobileNet) for the purpose of replacing the normal convolution operators in the state-of-the-art feature descriptor network, RF-Net. Experimental results show that the model size of the detector can be reduced by up to 80% compared to that of the original size with only a 11% degradation at worst performance in our final lightweight detector model for image matching tasks. Our study indicates that the modern convolution techniques originally proposed for small image classification models can be effectively extended to designing tiny models for the feature descriptor extraction and matching portions in deep local feature learning networks.

Keywords: Neural networks, lightweight feature descriptor networks, depthwise separable convolution, deep learning

1 Introduction

Detecting salient feature points and their corresponding feature descriptors is a classical and useful technique in computer vision [4], and recent studies [5, 8] have proposed high-performing feature descriptors based on deep neural networks. Although the methodology of designing practical networks that can run on embedded devices is one of the active research areas in deep learning [2, 3, 6, 7], the existing feature detection networks incur prohibitively large computational load.

The goal of feature detection is to robustly locate points of interest between similar images. The de facto standard feature descriptor before deep learning was SIFT [4] and its variants, which were designed by "hand-crafting" a feature detector and a feature descriptor. Once keypoints are extracted from an image, only the positions of the pixels are known. In order to compare the keypoints, feature descriptors, or in other words, the area or patch of pixels surrounding each detected keypoint, are compared against all of the corresponding patches from another image. The feature descriptors are simply

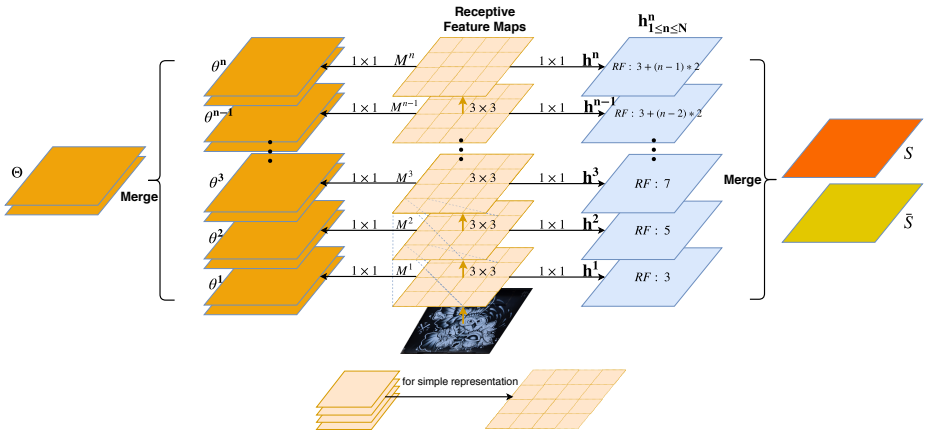

Fig. 1. Visual representation of RF-Net detector (Adopted from [8]).

vectors that can be compared using metrics such as the Euclidean distance. A matching pair of keypoints would therefore be defined by the minimum Euclidean distance between two feature descriptor vectors.

Deep learning methods for feature detection and description have recently been proposed. One such example is the state-of-the-art LF-Net, which is "trainable end-to-end and does not require using a hand-crafted detector to generate training data" [5]. LF-Net employs a novel architecture split into two divisions: the keypoint detector/descriptor extractor and the learning of the global image and descriptor patches. However, a few technical drawbacks of LF-Net are that the datasets it uses are quite huge and the model requires camera pose information. A simpler, improved version of LF-Net called RF-Net [8] also achieved state-of-the-art results. The technical improvements of RF-Net are that it offers more receptive fields of an image (Fig. 1), which assists in more effective keypoint detection, and a more stable descriptor training approach [8]. The quality of life changes are that it is a simpler model as it does not have to account for 3D data (i.e. camera orientation and pose) and trains on lighter datasets.

The RF-Net detector is the first step in the network to determine keypoints. Once the detector has found enough keypoints, descriptors, or patches of pixels around each keypoint are then collected and run through the second portion of RF-Net, which is the training network. The detector is where our focus was placed in order to create a lightweight detector model.

In this paper, we investigated the adaptability of RF-Net into a model with less computational load. Specifically, we employ lightweight convolutional layers such as depthwise separable layers to RF-Net's keypoint detection model. We found that decreasing the computational load by 80% of that of the original detector's computational load had only a minimal reduction (at most 11%) on the matching scores of RF-Net. However, the number of detected keypoints decreased depending on the amount of replaced convolutional layers with depthwise separable layers. In addition, when evaluating our final lightweight model on an actual machine with respect to GPU inference

runtime percentage breakdown, we found that there were slightly higher percentages of overall pre-processing times but shorter keypoint matching and post-processing percentages in the context of the total runtime when compared to that of [8]'s pretrained model. If the principles employed by state-of-the-art image classification networks such as SqueezeNet [3] and MobileNet [2] are also applicable to feature descriptor networks, then there is potential in making feature descriptor networks available on older machines or edge devices.

2 Making RF-Net detector lightweight

2.1 Basic convolutional layer architecture

A standard convolutional layer is fed feature maps of height H, width W, and input channels C_{in}, of which the outputs are then batch normalized, and subsequently passed into one of a variety of possible activation layers. Assuming that the convolutional layer uses a square kernel of size $K \times K$ and that the output channels of the feature maps are C_{out}, the computational cost of a standard convolutional layer is:

$$K^2 C_{\text{in}} C_{\text{out}} HW. \tag{1}$$

2.2 Making RF-Net detector lightweight

Our method to reduce the theoretical computational load of the RF-Net detector was done through the replacement of the default convolutional layers with depthwise separable layers. The default RF-Net detector uses ten *concerned receptive fields* to generate feature maps in order to detect points of interests (Fig. 1). The output of each subsequent convolutional layer is convolved with a kernel size of 3×3 with a stride of 1, which increases the receptive field size from 3, 5, 7, . . . , 21.

Since the kernel size is not 1×1 for most of the layers shown in the middle column (receptive feature maps) of Fig. 1, this makes the RF-Net detector a good candidate for application of depthwise separable convolution proposed by MobileNet [2]. This portion of RF-Net is where we centered our experiments on. When a convolutional layer is replaced by a depthwise separable layer, it is decomposed into a depthwise layer and a pointwise layer.

In a normal convolutional layer, each area screened by the kernel outputs a value, which then is combined with other input channels to become part of a new feature map. A basic RGB image often has its output channels reduced to 3 from 1. Multiple kernels are then stacked together to output multi-channel feature maps. In short, an image is transformed C_{out} times.

2.3 Estimating improvements

In a depthwise layer, an image is only transformed once as each kernel iterates only over one channel. As a result, the input channels, C_{in}, and output channels, C_{out}, remain the same. Afterwards, a normal convolutional layer with a kernel size of 1×1 is then applied

on the outputs from the depthwise layer. In the same vein as a normal convolutional layer, the number of output channels is determined by the amount of kernels used.

While it may seem that basic convolutional and depthwise separable layers process feature maps in the same way, the depthwise separable layer saves immensely due to the way the transformation of the feature maps is only handled once:

$$K^2 C_{\text{in}} HW. \tag{2}$$

As represented above, there is a factor less of C_{out} that needs to be considered for the depthwise layer. As for the pointwise layer where the kernel size is 1×1, and thus negligible in the computation, the pointwise layer can be represented by:

$$C_{\text{in}} C_{\text{out}} HW. \tag{3}$$

Once summed together, the theoretical computational cost of a depthwise separable layer comes out to:

$$K^2 C_{\text{in}} HW + C_{\text{in}} C_{\text{out}} HW = C_{\text{in}} HW(K^2 + C_{\text{out}}). \tag{4}$$

When the basic convolutional layer is compared to the depthwise separable layer:

$$\frac{K^2 C_{\text{in}} HW}{C_{\text{in}} HW(K^2 + C_{\text{out}})} = \frac{K^2}{(K^2 + C_{\text{out}})}. \tag{5}$$

Thus, the depthwise separable convolutional layer is a factor of $K^2/(K^2 + C_{\text{out}})$ smaller compared to that of the basic convolutional layer.

3 Experimental results

3.1 Configurations

Datasets. We used the same datasets as the RF-Net study [8]: the HPatches [1] and EF [9] datasets. HPatches includes 51 sequences, each of which has one reference image and five images with varying photometric changes. RF-Net [8] chose to split this dataset further into viewpoint (30 pair images) and illumination (285 pair images) categories for testing purposes. The EF dataset consists of five different base images with each base image having corresponding images in various viewpoint and illumination changes.

The initial convolutional layer. After applying depthwise separable convolution to all the layers in the RF-Net detector, we found that it was most effective to not replace the initial convolutional layer (data not shown). By not doing so, the validation and matching scores of the trained models were found to be higher compared to the models in which the initial convolutional layer was replaced. The reason for this may be that the kernel size of the initial convolutional layer was already 1×1, so making this layer depthwise separable may have traded too much feature map information for too little reduction in computational load.

Table 1. Nine variants of lightweght RF-Net architectures.

#Layers replaced (Label)	Replaced Layers
1	self.conv2
2	self.conv2 ~ self.conv3
3	self.conv2 ~ self.conv4
4	self.conv2 ~ self.conv5
5	self.conv2 ~ self.conv6
6	self.conv2 ~ self.conv7
7	self.conv2 ~ self.conv8
8	self.conv2 ~ self.conv9
9	self.conv2 ~ self.conv10

Comparison. We compare the performance of our lightweight RF-Net with those reported in [8] and also those of the pretrained model (downloaded from https://github.com/Xylon-Sean/rfnet). We examined nine variants of lightweight RF-Net architectures shown in Table 1.

3.2 Performance evaluation of the lightweight RF-Net detector

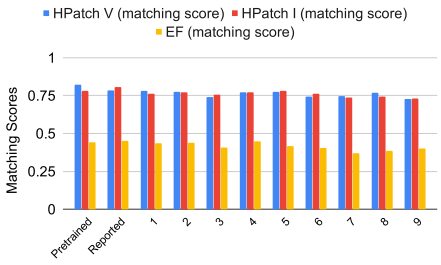

Fig. 2. Model evaluation for our various lightweight RF-Net detector showing the number of layers replaced vs. matching scores.

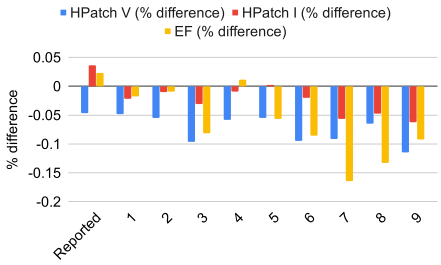

Fig. 3. Percentage difference in matching score when comparing [8]'s pretrained model with our various lightweight RF-Net detector models.

As shown in Fig. 2, when all other layers besides the initial convolutional layer in RF-Net's detector are replaced with lightweight depthwise separable layers, the most single significant drop in matching score (Fig. 3) is by about 18% in EF dataset's matching score when the first six layers after the initial convolutional layer are replaced. However, as shown in Figs. 4 and 5 for our final lightweight model, the worst performance is around 11% for the HPatch viewpoint dataset.

In Fig. 6, when the computational load of the RF-Net detector is slowly reduced by incrementally replacing normal convolutional layer with depthwise separable convolutional layers, the matching scores across all datasets do not vary significantly.

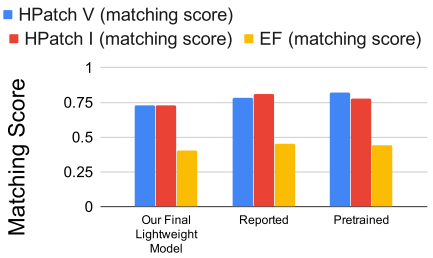

Fig. 4. Model evaluation for our final lightweight RF-Net detector vs. matching scores.

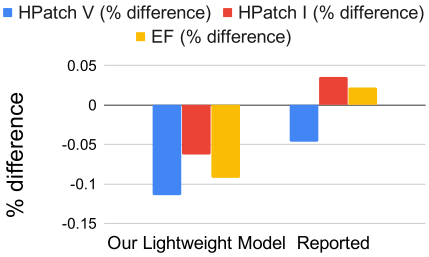

Fig. 5. Percentage difference in matching score when comparing [8]'s pretrained model with our final lightweight RF-Net detector model and [8]'s reported model.

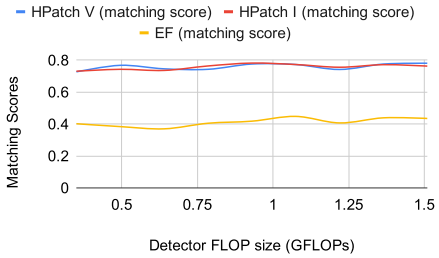

Fig. 6. Lightweight RF-Detector model performance in terms of computational load vs. matching scores. Dimensions of test image are 240×320.

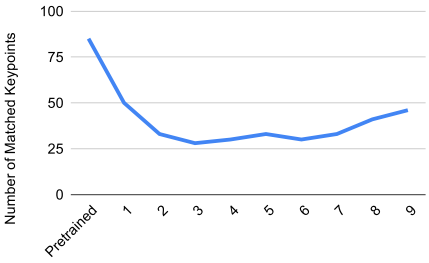

Fig. 7. Matched keypoints vs. number of convolutional layers replaced for author provided sample image (Fig. 8).

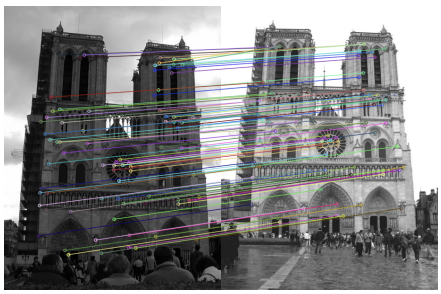

Fig. 8. Visual representation (using a sample image provided by [8]) of matched keypoints in [8]'s pretrained model.

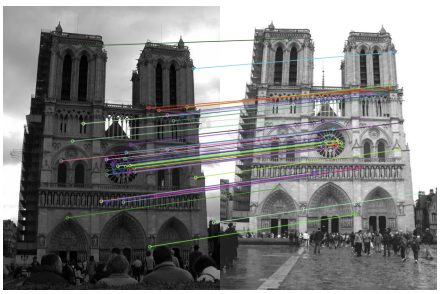

Fig. 9. Visual representation (using a sample image provided by [8]) of matched keypoints in our final lightweight model.

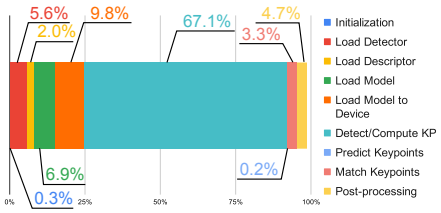

Fig. 10. Percentage breakdown of inference run times (ms) for [8]'s pretrained model. Dimensions of test image are 480×640. GPU used was a single NVIDIA GeForce GTX 1080 Ti.

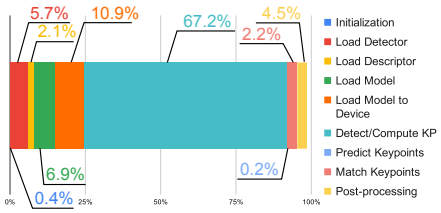

Fig. 11. Percentage breakdown of inference run times (ms) for our final lightweight model. Dimensions of test image are 480×640. GPU used was a single NVIDIA GeForce GTX 1080 Ti.

3.3 Computational load and number of matched keypoints

To measure the amount of keypoints matched, we used the default image that the authors of RF-Net used as an example. As seen in Fig. 7, the number of matched keypoints drops significantly after only one convolutional layer is replaced. After about six convolutional layers are replaced with depthwise separable layers, the number of matched keypoints seems to increase compared to the number of the previous layers. Comparisons between visual representation in Figs. 8 and 9 show roughly how many and which keypoints are lost from making the RF-Net detector lightweight.

3.4 Practical runtime efficiency

To empirically determine the effect of our lightweight model on an actual computer, we evaluated the GPU runtimes during inference. Each significant code block within the inference script was placed between two CUDA events, which were then synchronized with the CUDA stream in order to record accurate runtimes. Twenty trials each for [8]'s pretrained model and our own final lightweight model were conducted, and the lowest and highest two total GPU runtimes were removed from the mean. The visual representations in Figs. 10 and 11 show that there were no significant differences between the code blocks within the inference script. However, interesting points that appeared were that our final lightweight model had higher initialization (by 0.1%), detector loading (by 0.1%), descriptor loading (by 0.1%), model loading to GPU times (by 0.9%), and keypoint detection/computation (by 0.1%) in the percentage breakdown when compared with the breakdown of [8]'s pretrained model. However, our final lightweight model had lower breakdown portions for keypoint matching (by 1%) and post-processing (by 0.1%).

4 Conclusions

In this work, we applied the technique of depthwise separable convolution to RF-Net, a state-of-the-art feature descriptor network. We heavily reduced the computational load of the feature detector portion of the architecture with a negligible change in terms of relative matching scores. In addition, the proposed changes to the detector model of RF-Net revealed that there was only a minor decrease in overall matching score performance

when comparing our final lightweight detector model with that of [8]'s pretrained model. While there were small fluctuations in relative matching scores during the process of making convolutional layers in the RF-Net detector lightweight, we found that a more significant trade-off was the decrease in number of matched keypoints.

Further experiments investigating the practical runtime efficiency of both [8]'s pretrained model and our own lightweight model revealed that there was no significant change in the percentage breakdown of the inference runtimes. While there were higher percentages of pre-processing and keypoint detection/computation in the overall scope of the inference runtimes in our final lightweight model compared to that of [8]'s pretrained model, there were also lower percentages in the keypoint matching and post-processing blocks. From the data gleaned in the subsequent experiments, it appears that there is still a gap between theoretical manipulation and pragmatic application. Deep-learning frameworks may not implement depthwise separable layers as efficiently now, and the runtime benchmarking of CUDA events might also not be the best metric for evaluation of inference speed.

References

1. V. Balntas, Lenc K., A. Vedaldi, and K. Mikolajczyk. HPatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
2. A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv:1704.04861*, 2017.
3. F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. *arXiv:1602.07360*, 2016.
4. D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.
5. Y. Ono, E. Trulls, P. Fua, and K. M. Yi. LF-Net: Learning local features from images. In *Advances in Neural Information Processing Systems (NIPS)*, 2018.
6. M. Sandler, G. Chu, and L.-C. Chen. Searching for MobileNetV3. In *International Conference on Computer Vision (ICCV)*, 2019.
7. M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
8. X. Shen, C. Wang, X. Li, Z. Yu, J. Li, C. Wen, M. Cheng, and Z. He. RF-Net: An end-to-end image matching network based on receptive field. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
9. C. L. Zitnick and K. Ramnath. Edge foci interest points. In *International Conference on Computer Vision (ICCV)*, 2011.