Supplementary Material for "ViP: Virtual Pooling for Accelerating CNN-based Image Classification and Object Detection"

Zhuo Chen Jiyuan Zhang Ruizhou Ding
Carnegie Mellon University
zhuoc1, jiyuanz, rding@andrew.cmu.edu

Diana Marculescu
University of Texas at Austin & Carnegie Mellon University
dianam@utexas.edu & dianam@cmu.edu

Figure 1 further illustrates how ViP method can be applied to applications, such as a face detector in a mobile camera system. In mobile phone cameras, a face detector can not only show where the faces are, but also help camera auto-focus for taking better pictures with people in sharp imaging. Therefore, in this case lower accuracy can be tolerated as long as some of the faces, if not all, are detected. At the same time, there is also a speed requirement for camera to focus as fast as possible. For such applications, we can generate a set of models using the ViP method. We start from a pre-defined or pre-trained model, *e.g.*, Faster-RCNN, and a pre-collected dataset for human faces. Then we perform Per-layer Sensitivity Analysis (PSA) to determine the sensitivity of each conv layer in the network. Based on the sensitivity, we can perform either grouped or per-layer finetuning to generate new models with higher speedup until we obtain the model that best satisfies the requirement on speedup-accuracy trade-off.

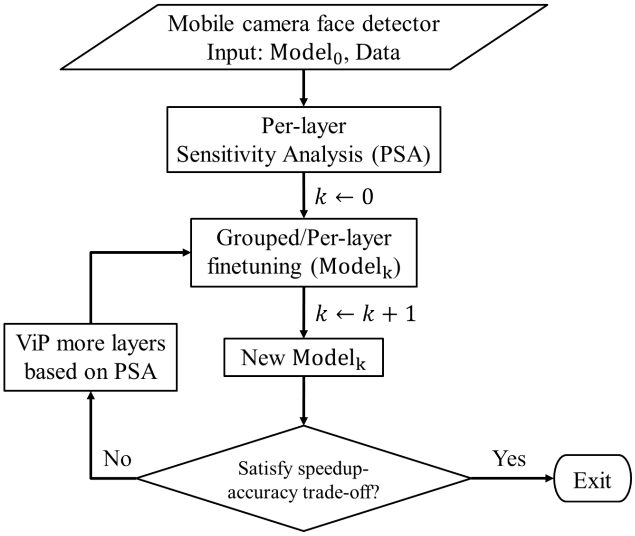

Figure 1. An example of applying ViP to the mobile phone camera face detector. ViP progressively generates new models with higher speedup until we obtain the model that best satisfies the requirement on speedup-accuracy trade-off.

We also tried out our ViP method on the All-CNN network (Network All-CNN-C from [2]) which consists of nine convolution layers without a fully-connected layer and delivers high accuracy on CIFAR-10 dataset. With three rounds of ViP insertion and fine-tuning, we obtain three models with different speedup-accuracy trade-offs, as illustrated in Figure 2. We are able to achieve **1.77x** speedup on the Titan X GPU and up to **3.03x** speedup on the mobile CPU, with desktop CPU and mobile GPU in between, while the top-1 accuracy drop is within 4%. Also, with less than 1% accuracy degradation, we obtain a **1.36x** speedup on Titan X GPU and **1.54x** speedup on mobile CPU.

We port both Caffe and our custom ViP layer to Jetson TX1. We use the on-board sensor to measure the power consumption of CNNs with and without ViP technique, and obtain the energy consumption by multiplying power by CNN latency. We can see that, in terms of power, All-CNN has almost the same power on both CPU and GPU across all different ViP configurations. In terms of energy consumption, All-CNN achieves up to **46%** and **70%** energy reduction on mobile GPU and CPU, respectively.

To speedup the network, the upsampling overhead must be small so it does not offset the latency decrease resulting from the larger-stride convolution. ViP indeed works as an upsampling method and is thus similar to transposed convolution (Deconv) in terms of functionality. However, ViP is very fast with low runtime overhead (on average 4.7% of the network) and hence can achieve high speedup for the

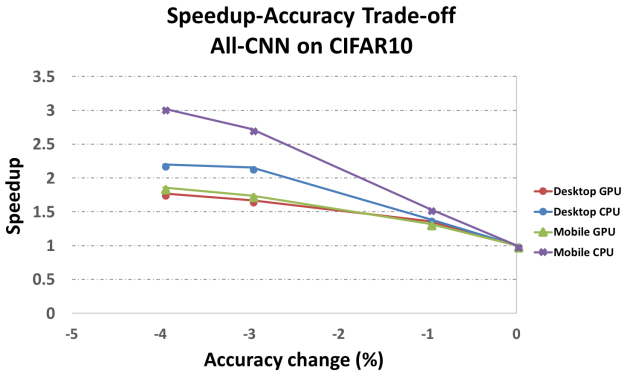

Figure 2. Speedup-Accuracy trade-off obtained by applying ViP on All-CNN model with CIFAR-10 dataset.

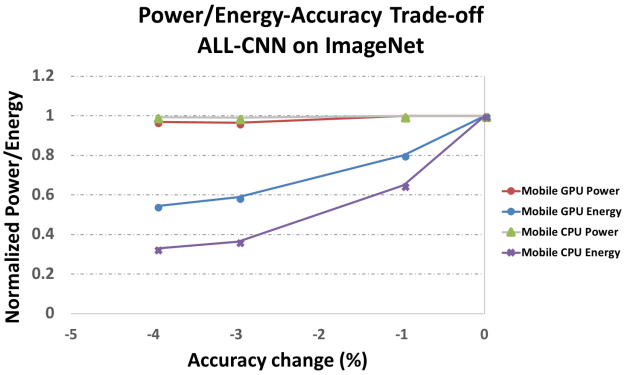

Figure 3. Power/Energy-Accuracy trade-off obtained by applying ViP on All-CNN model with CIFAR-10 dataset.

overall network. FCN [1] uses Deconv to upsample, and its major drawback is its expensive computation. For example, consider a conv layer of input size $H * W * C$ ($H = W = 224$, $C = 3$) and $N = 64$ filters of size $M*M*C$ ($M = 3$). The MAC count for linear interpolation (after a stride 2 conv) is $(H/2 * W/2) * 2 * C * 2 + (H/2 * W/2) * C * 4 = 301,056$ (last three cases in Eq.5). The MAC count of Deconv (after a stride 2 conv) is $H * W * M * M * N * N = 1,849,688,064$ (We have two Ns since the number of output channels after Deconv when used as upsampling should be the same as its input, which is $N = 64$ after stride 2 conv). We note that ViPs computation is only 0.016% of transposed convolution.

To apply ViP on Faster-RCNN, we analyze the sensitivity of each layers to ViP and the results are shown in Fig.4. Notice that, other than the conv layers from VGG16, we also have one layer from the region proposal network, and this layer turns out to be among the least sensitive layers that we need to insert ViP in early finetuning rounds. The mAP degradation from per-layer ViP ranges from $-0.23 \sim -0.27$, however, the recovered mAP degradation after finetuning is only -0.024. Besides, we again observe that layers immediately followed by pooling are the most robust to ViP operation, as already discussed in the section of image classication.

Proof for Proposition 1 is presented here.

Proof. We prove this bound by first bounding the error of the ViP layer $\mathcal{O}^{(l_s)}$, and then bounding the error accumulated to higher layers.

For the ViP layer $\mathcal{O}^{(l_s)}$, $\forall c, h, w$, the ViP error of $\mathcal{O}^{(l_s)}_{c,h,w}$ can be bounded:

$$\begin{aligned}
&|\mathcal{O}^{(l_s)ViP}_{c,h,w} - \mathcal{O}^{(l_s)Orig}_{c,h,w}| \\
=&|\frac{1}{K} \sum_{\substack{k=1 \\ (h_k,w_k)\in\mathcal{N}_{h,w}}}^{K} \mathcal{O}^{(l_s)ViP}_{c,h_k,w_k} - \mathcal{O}^{(l_s)Orig}_{c,h,w}| \\
\leq&\frac{1}{K} \sum_{\substack{k=1 \\ (h_k,w_k)\in\mathcal{N}_{h,w}}}^{K} |\mathcal{O}^{(l_s)ViP}_{c,h_k,w_k} - \mathcal{O}^{(l_s)Orig}_{c,h,w}| \\
\leq&L * d_{max} = \sqrt{2}L
\end{aligned} \tag{1}$$

where $\mathcal{N}_{h,w}$ is the set of neighbor locations of (h, w) that are averaged to compute the ViP value for pixel (h, w), and d_{max} is the maximum l_2-norm distance of the location (c, h, w) and a neighbor location (c, h_k, w_k), which is $\sqrt{2}$ in ViP.

Then, we bound the error accumulated from layer $l-1$ to l, *i.e.*, $|\mathcal{O}^{(l)ViP}_{c',h,w} - \mathcal{O}^{(l)Orig}_{c',h,w}|$. Due to Eq. (1),

$$\begin{aligned}
\mathcal{O}^{(l)ViP}_{c',h,w} &= (\sum_{c=1}^{C^{(l)}} \sum_{m,n=-\lfloor M^{(l)}/2 \rfloor}^{\lfloor M^{(l)}/2 \rfloor} \mathcal{O}^{(l-1)ViP}_{c,s\cdot h-m,s\cdot w-n} * \mathcal{W}^{(l)}_{c',c,m,n})_+ \\
\mathcal{O}^{(l)Orig}_{c',h,w} &= (\sum_{c=1}^{C^{(l)}} \sum_{m,n=-\lfloor M^{(l)}/2 \rfloor}^{\lfloor M^{(l)}/2 \rfloor} \mathcal{O}^{(l-1)Orig}_{c,s\cdot h-m,s\cdot w-n} * \mathcal{W}^{(l)}_{c',c,m,n})_+
\end{aligned} \tag{2}$$

where $(.)_+$ is the ReLU activation function. Due to the fact that $\forall x, y \in R, |(x)_+ - (y)_+| \leq |x - y|$, we have:

$$\begin{aligned}
&|\mathcal{O}^{(l)ViP}_{c',h,w} - \mathcal{O}^{(l)Orig}_{c',h,w}| \\
\leq&|\sum_{c=1}^{C^{(l)}} \sum_{m,n=-\lfloor M^{(l)}/2 \rfloor}^{\lfloor M^{(l)}/2 \rfloor} (\mathcal{O}^{(l-1)ViP}_{c,s\cdot h-m,s\cdot w-n} - \mathcal{O}^{(l-1)Orig}_{c,s\cdot h-m,s\cdot w-n}) \mathcal{W}^{(l)}_{c',c,m,n}|
\end{aligned} \tag{3}$$

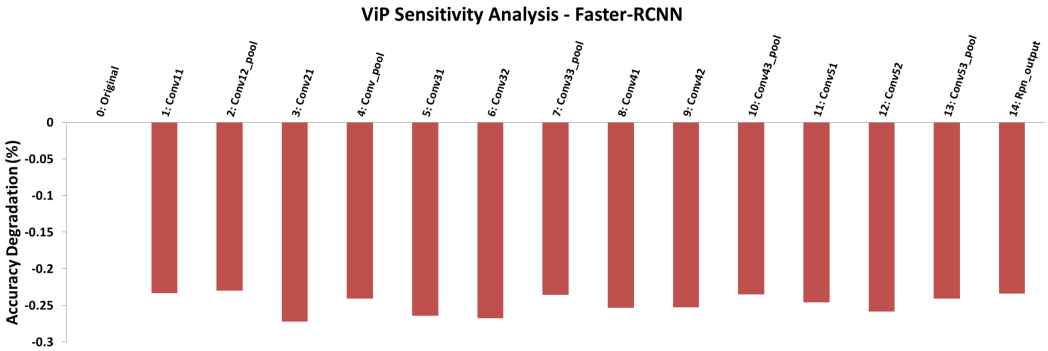

Figure 4. ViP sensitivity analysis of faster-rcnn with VGG16 backbone under PASCAL VOC 2007 dataset. For each of the conv layers, we insert ViP immediately after it, and evaluate the accuracy without finetuning. The sensitivity is measured as the accuracy degradation.

Using Cauchy-Schwarz inequality, we have:

$$
\begin{aligned}
&|\mathcal{O}^{(l)ViP}_{c',h,w} - \mathcal{O}^{(l)Orig}_{c',h,w}| \\
\leq &\sqrt{\sum_{c=1}^{C^{(l)}} \sum_{m,n=-\lfloor M^{(l)}/2 \rfloor}^{\lfloor M^{(l)}/2 \rfloor} (\mathcal{O}^{(l-1)ViP}_{c,s\cdot h-m,s\cdot w-n} - \mathcal{O}^{(l-1)Orig}_{c,s\cdot h-m,s\cdot w-n})^2} \\
&* \sqrt{\sum_{c=1}^{C^{(l)}} \sum_{m,n=-\lfloor M^{(l)}/2 \rfloor}^{\lfloor M^{(l)}/2 \rfloor} \mathcal{W}^{(l)}_{c',c,m,n}} \\
\leq &\sqrt{C^{(l)}} M^{(l)} B^{(l)} \max_{c',h,w} |\mathcal{O}^{(l-1)ViP}_{c,h,w} - \mathcal{O}^{(l-1)Orig}_{c,h,w}|
\end{aligned}
\tag{4}
$$

Therefore, accumulating the error from the ViP layer l_s to layer l_e, we have:

$$
|\mathcal{O}^{(l_e)ViP}_{c',h,w} - \mathcal{O}^{(l_e)Orig}_{c',h,w}| \leq \sqrt{2} L \prod_{l=l_s+1}^{l_e} \sqrt{C^{(l)}} M^{(l)} B^{(l)}, \tag{5}
$$

Thus, the l_2-norm of the output error is bounded by:

$$
\begin{aligned}
&||\mathcal{O}^{(l_e)ViP} - \mathcal{O}^{(l_e)Orig}||_2 \\
\leq &\sqrt{2} L \sqrt{C'^{(l_e)} H^{(l_e)} W^{(l_e)}} \prod_{l=l_s+1}^{l_e} \sqrt{C^{(l)}} M^{(l)} B^{(l)}.
\end{aligned}
\tag{6}
$$

□

References

[1] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[2] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.