SSBNet: Improving Visual Recognition Efficiency by Adaptive Sampling

Ho Man Kwan and Shenghui Song

The Hong Kong University of Science and Technology
hmkwan@connect.ust.hk eeshsong@ust.hk

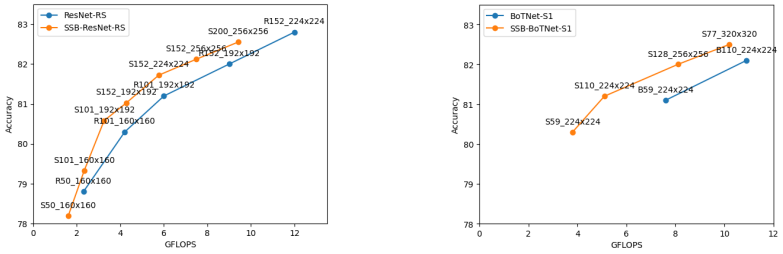
Abstract. Downsampling is widely adopted to achieve a good trade-off between accuracy and latency for visual recognition. Unfortunately, the commonly used pooling layers are not learned, and thus cannot preserve important information. As another dimension reduction method, adaptive sampling weights and processes regions that are relevant to the task, and is thus able to better preserve useful information. However, the use of adaptive sampling has been limited to certain layers. In this paper, we show that using adaptive sampling in the building blocks of a deep neural network can improve its efficiency. In particular, we propose SSBNet which is built by inserting sampling layers repeatedly into existing networks like ResNet. Experiment results show that the proposed SSBNet can achieve competitive image classification and object detection performance on ImageNet and COCO datasets. For example, the SSB-ResNet-RS-200 achieved 82.6% accuracy on ImageNet dataset, which is 0.6% higher than the baseline ResNet-RS-152 with a similar complexity. Visualization shows the advantage of SSBNet in allowing different layers to focus on different positions, and ablation studies further validate the advantage of adaptive sampling over uniform methods.

Keywords: Convolutional neural networks, Image recognition, Network architecture, Adaptive sampling, Attention mechanism

1 Introduction

Deep learning models such as convolutional neural networks (CNNs) [7,13,22,24] and Transformers [5] have made unprecedented successes in computer vision. However, achieving efficient inference with stringent latency constraints in real-world applications is very challenging. To obtain a good trade-off between accuracy and latency, downsampling is normally used to reduce the number of operations. Most existing CNNs [7, 13, 22, 24] perform downsampling between stages, coupled with the increase in channel dimension to balance the representation power and computational cost. Typical downsampling operations include strided average/max pooling and convolutions [7, 13, 14, 18, 22, 24], which are uniformly applied in the spatial dimension.

Besides uniform sampling, there are also non-uniform or adaptive approaches [4,11,12,20,26,32], with which different transformations including zooming, shifting, and deforming can be utilized to selectively focus on the important regions



(a) (b)

Fig. 1: Comparison between the SSB-ResNet-RS and ResNet-RS [2], SSB-BoTNet-S1 and BoTNet-S1 [23] on ImageNet [21] dataset. The proposed SSB-ResNet-RS/SSB-BoTNet-S1 outperforms ResNet-RS and BoTNet-S1 in terms of accuracy to FLOPS ratio. Results of ResNet-RS are from the original paper, where the results of BoTNet-S1 are from reimplementation. See section 4.1.

during downsampling. However, the use of adaptive sampling has been limited to certain layers and its application in backbone networks has not been well investigated. Backbone networks are usually pre-trained in some large scale datasets, which are agnostic to the end task like object detection [31]. The challenge for applying adaptive sampling in backbone networks lies in the possible information loss. In particular, later layers cannot access pixels that were skipped by earlier layers, which is quite possible due to the stacking of sampling layers.

Another approach to improve efficiency is to reduce the number of channels. In ResNet [7], the bottleneck layers reduce the channel dimension by using a 1×1 convolution and then perform a costly 3×3 convolution on the low dimensional features to reduce computational complexity. After that, another 1×1 convolution is utilized to restore the dimension and match the shortcut connection. It is noteworthy that residual networks can preserve informative features after the bottleneck operations because the shortcut connection allows signal to bypass the bottleneck.

The bottleneck structure with shortcut connection can be utilized to enable adaptive sampling on backbone networks. In this paper, we propose Saliency Sampling Bottleneck Network (SSBNet), which applies saliency sampler [20, 32] in a bottleneck structure to reduce the spatial dimension before costly operations. An inverse operation is then used to restore the feature maps to match the spatial structure of the input that passes through the shortcut connection. Like other bottleneck structures, computationally expensive operations like convolution are applied in a very compact space to save computations. There are two major advantages for applying adaptive sampling over the bottleneck structure. First, by zooming into important regions, SSBNet can better extract features than uniform downsampling. More importantly, with the shortcut connection, each intermediate layer with adaptive sampling can focus on different regions of the feature maps in a very deep network, without loss of information.

In the experiments, we built SSBNets by inserting lightweight convolutional layers and samplers into existing networks to estimate the saliency map and perform down/upsampling. The results in Figure 1 show that SSBNet can achieve better accuracy/FLOPS ratio than ResNet-RS [2] and BoTNet-S1 [23]. For example, with only 4.4% more FLOPS, the SSB-ResNet-RS-200 with input size of 256×256 achieved 82.6% accuracy on ImageNet dataset [21], which is 0.6% higher than the baseline ResNet-RS-152 with input size of 192×192. The SSB-BoTNet-S1-77 with input size of 320×320 obtained 82.5% accuracy, which is 0.4% higher than BoTNet-S1-110 with input size of 224×224, and required 6% less computation. The contributions of this paper include:

- We investigate the use of adaptive sampling in the building blocks of a deep neural network. By applying adaptive sampling in the bottleneck structure, we propose SSBNet which can be utilized as a backbone network and trained in an end-to-end manner. Note that existing networks only utilized adaptive sampling in specific tasks, where a pre-trained backbone is required for feature extraction.
- We show that the proposed SSBNet can achieve better image classification and object detection performance than the baseline models, and visualize its capability in adaptively sampling different locations at different layers.
- Experiment results and ablation studies validate the advantage of adaptive sampling over uniform sampling. The result in this paper may lead to a new direction of research on network architecture.

2 Related Works

In the following, we explain the connection between the proposed SSBNet and existing works, and highlight the innovation.

Attention Mechanisms. Different types of attention mechanisms have been explored in computer vision tasks. One category of work utilizes attention mechanism to predict a softmask that scales the feature maps. Squeeze-and-Excitation [9] utilizes global context to refine the channel dimension. CBAM [29] uses attention mask to emphasize the important spatial positions.

Besides improving feature maps, another direction of research applies attention as a stand-alone layer that can extract features and act as a replacement for the convolutional layer. Stand-alone self-attention [19] replaces the spatial convolutional layer to efficiently increase the receptive field. Vision Transformer [5] adapts the Transformer [28] structure and takes non-overlapped patches as individual tokens, instead of a map representation that is normally used in vision tasks.

The proposed SSBNet follows the first approach and inserts attention layers to improve efficiency. However, instead of utilizing attention to scale features, SSBNet performs weighted downsampling by the attention map to save computations.

Adaptive Sampling. There are some works [4, 11, 12, 20, 26, 32] that perform adaptive geometric sampling on the images or feature maps rather than scaling the features, as is done by attention mechanisms. Spatial transformer network [11] uses localization network to predict transformation parameters and performs geometric transformation on the image or feature maps. Saliency sampler [20] applies saliency map estimator to compute the attention map and distorts the input based on this map. Trilinear attention sampling network (TASN) [32] applies trilinear attention to compute the attention map and uses the map to perform sampling in a less distorted way. The sampling mechanism of the proposed SSBNet is inspired by TASN, but with two major differences: 1) SSBNet can be used as a backbone and trained end-to-end, but TASN requires a pre-trained backbone; 2) SSBNet performs different sampling at different layers to extract useful features, where TASN only performs sampling once on the image input.

There are only very few works that apply adaptive sampling in the backbone network, which is the core feature extractor for computer vision tasks. One of the exceptions is the Deformable convolutional neural network (DCNN) [4]. DCNN computes the sampling offset to deform the sampling grid of convolutions and RoI poolings, which provides significant performance improvement for object detection and semantic segmentation tasks. Different from DCNN which deforms the convolutions and RoI poolings, SSBNet samples the feature map into a lower dimension to improve efficiency.

In summary, the proposed SSBNet utilizes adaptive sampling in most of its building blocks and allows different sampling at different layers, where most existing works only perform adaptive sampling several times. SSBNet can be used as a backbone network for different tasks like classification and object detection.

Dimension Reduction. Dimension reduction is commonly used in different architectures. Reducing spatial dimensions can save a large amount of computation and increase the effective receptive field of the convolution operations. For example, many CNNs reduce the spatial dimension when they increase the number of channels [7, 13, 22, 24].

There are networks that temporarily reduce the channel dimension. Inception [24] applies 1×1 convolutions to reduce the channel dimension and lower the cost of the following 3×3 and 5×5 convolutions. ResNet [7] has a bottleneck layer design, which reduces the number of channels before the 3×3 convolution and restores the channel dimension afterwards.

There are also applications of the bottleneck structure in the spatial dimension. Spatial bottleneck [18] replaces spatial convolution by a pair of strided convolution and deconvolution to reduce the sampling rate and achieve speed-up. HBONet [14] utilizes depthwise convolution and bilinear sampling to perform down/upsampling, where the costly operations are applied in between.

The proposed SSBNet has a similar structure as HBONet, but utilizes adaptive sampling instead of strided convolution or pooling for downsampling. Fur-

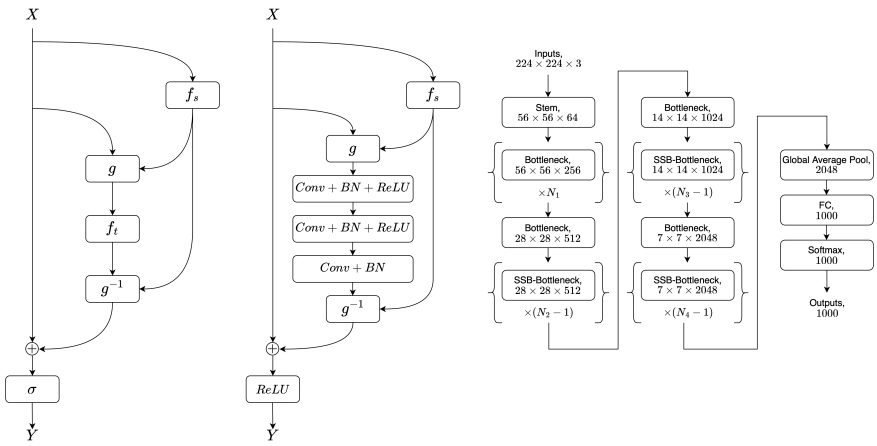


Fig. 2: Left: The structure of SSB layer. Middle: The instantiation of SSB layer built from the bottleneck layer of ResNet [7]. Right: SSB-ResNet, where N_1, N_2, N_3, N_4 follow the configurations of ResNet [7].

thermore, the adaptive sampling can perform spatial transformation like zooming, which could better preserve useful information for feature extraction.

3 Methodology

In this section, we first introduce the SSBNet and then present the details of the building block for SSBNet, i.e. the SSB layer.

3.1 Saliency Sampling Bottleneck Networks

Since the focus of this work is to apply adaptive sampling to improve network efficiency, we modify existing networks to reduce the searching space. To build SSBNet, we insert samplers to the building blocks of the original model such as ResNet [7]. To this end, we only need to determine the sampling size and the position to insert the sampler. In the experiments, we follow the standard approach that shares configuration in a group of building blocks, i.e. same sampling size in one group. Without native implementation of the sampler, some earliest groups that have high spatial dimension will significantly slow down the training. So we skip those earliest groups.

We also skip the first block of each group, i.e. the block that reduces the spatial dimension and increases the number of channels, due to the fact that they usually utilize shortcut connection with strided pooling or/and 1×1 convolution for downsampling [2, 7, 8], or do not contain shortcut connection [25]. Thus, adding samplers to the first block of each group could lead to loss of information. For example, SSB-ResNet is shown in Figure 2 (right).

3.2 Saliency Sampling Bottleneck Layer

The SSB layer is the main building block of the SSBNet and is constructed by wrapping a set of layers with the samplers. The SSB layer has a similar structure as the bottleneck layer from ResNet [7], with two branches, i.e., the shortcut branch and the residual branch. The function of the two branches are similar to those in ResNet. Specifically, the shortcut branch passes the signal to the higher level layer and the residual branch performs operations to extract features. The difference is that the residual branch in the SSB layer adaptively samples features in the spatial dimension, but the bottleneck layer of ResNet reduces the channel dimension, and both of them perform the most costly operations in the reduced space. Figure 2 (left) shows the structure of the SSB layer. Next, we introduce the key operations of the SSB layer.

Saliency Map: Given the input feature map $X \in \mathbb{R}^{H_{in} \times W_{in} \times D}$, where H_{in}, W_{in}, D denote the height, the weight and the number of channels, respectively, we first compute the saliency map by $S = f_s(X)$, where S has dimensions of $H_{in} \times W_{in}$. There are many possible choices of f_s. In this paper, we use a 1×1 convolutional layer with one filter, a batch normalization layer [10] and the sigmoid activation, followed by a reshape operation which change the map of $H_{in} \times W_{in} \times 1$ to a 2D matrix of $H_{in} \times W_{in}$. The whole process has negligible overhead in the number of operations and parameters. To stabilize the training, we always initialize the scaling weight γ in the batch normalization layer to zero, such that the network performs uniform sampling at the beginning.

Sampling Output Computation: For a target sampling size of $H_r \times W_r$, we compute the sampling output $X^r = g(X, S)$ with $X^r \in \mathbb{R}^{H_r \times W_r \times D}$. Our approach is close to TASN [32]. Specifically, we apply inverse transform to convert the saliency map into the weights of sampling, where features having higher scores in the saliency map will be sampled with a larger weight into the output feature maps. Unlike TASN, our implementation does not involve bilinear sampling [11]. Instead, we directly compute the sampling weights between the input and output pixels.

To compute X^r with a saliency map $S \in \mathbb{R}^{H_{in} \times W_{in}}$, we first obtain the elements of the saliency vectors $S^y \in \mathbb{R}^{H_{in}}$ and $S^x \in \mathbb{R}^{W_{in}}$ as

$$S^y_j = \frac{\sum_{w=1}^{W_{in}} S_{j,w}}{\sum_{h=1}^{H_{in}} \sum_{w=1}^{W_{in}} S_{h,w}} \quad \forall 1 \leq j \leq H_{in} \tag{1}$$

and

$$S^x_i = \frac{\sum_{h=1}^{H_{in}} S_{h,i}}{\sum_{h=1}^{H_{in}} \sum_{w=1}^{W_{in}} S_{h,w}} \quad \forall 1 \leq i \leq W_{in}. \tag{2}$$

Note that both S^y and S^x are normalized.

We also compute uniform vectors, $U^y \in \mathbb{R}^{H_r}$ and $U^x \in \mathbb{R}^{W_r}$, where

$$U^y_j = \frac{1}{H_r} \quad \forall 1 \leq j \leq H_r \tag{3}$$

$$U_i^x = \frac{1}{W_r} \quad \forall 1 \le i \le W_r. \tag{4}$$

Then, we calculate the cumulative sums C^{S^y}, C^{S^x}, C^{U^y} and C^{U^x}. For example, in the y-axis, we first compute

$$C_j^{S^y} = \sum_{h=1}^{j-1} S_h^y \quad \forall 1 \le j \le H_{in} + 1 \tag{5}$$

$$C_j^{U^y} = \sum_{h=1}^{j-1} U_h^y \quad \forall 1 \le j \le H_r + 1 \tag{6}$$

and then the sampling weights can be determined as

$$\begin{gathered} G_{i,j}^y = max(min(C_{j+1}^{S^y}, C_{i+1}^{U^y}) - max(C_j^{S^y}, C_i^{U^y}), 0) \\ \forall 1 \le j \le H_{in}, 1 \le i \le H_r. \end{gathered} \tag{7}$$

The weight matrix in the x-axis, G^x, can be computed similarly. Weight matrices G^y and G^x have dimensions of $H_r \times H_{in}$ and $W_r \times W_{in}$, respectively.

Finally, we can compute the sampling output X^r by

$$\begin{gathered} X_{i,j,d}^r = \sum_{h=1}^{H_{in}} \sum_{w=1}^{W_{in}} H_r W_r G_{i,h}^y G_{j,w}^x X_{h,w,d} \\ \forall 1 \le i \le H_r, 1 \le j \le W_r, 1 \le d \le D. \end{gathered} \tag{8}$$

We applied scaling with a factor of $H_r W_r$, such that the average value of the output map is independent of the sampling size.

Feature Extraction: After computing the sampled feature maps X^r, we can extract features by costly operations like convolutions with $Y^r = f_t(X^r)$. When building SSBNet from existing networks with shortcut connections [7], we use the original residual branch as f_t. Figure 2 (middle) shows the SSB layer built from the (channel) bottleneck layer of ResNet.

Inverse Sampling: After the feature extraction stage, an inverse sampling is applied to restore the spatial dimension. For that purpose, we apply the same sampling method, except that the transposed weight matrices, i.e. $(G^y)^T$ and $(G^x)^T$, are utilized. Together with the shortcut connection and the activation function σ, the final output of the SSB layer can be expressed as

$$\begin{gathered} Y_{i,j,d} = \sigma(X_{i,j,d} + \sum_{h=1}^{H_r} \sum_{w=1}^{W_r} H_{in} W_{in} (G^y)_{i,h}^T (G^x)_{j,w}^T Y_{h,w,d}^r) \\ \forall 1 \le i \le H_{in}, 1 \le j \le W_{in}, 1 \le d \le D. \end{gathered} \tag{9}$$

Instead of using bilinear sampling [11], we compute the weights between the input and output pixels, and directly use the weighted sum as the output value.

This approach can simplify the calculation, as it does not involve calculation of the coordinates. Furthermore, bilinear sampling may skip some pixels due to the possible non-uniform downsampling, but the proposed method takes all input pixels into account.

Note that the sampling function can be simply implemented by two batch matrix multiplications, which gives a complexity of $O(H_r H_{in} W_{in} D + H_r W_r W_{in} D)$ (when computed in y-axis first). The complexity is higher than bilinear sampling that has a complexity of $O(H_r W_r D)$. However, the weight matrices G^y and G^x contain at most $H_{in} + H_r$ and $W_{in} + W_r$ non-zero elements [1] , respectively. If the sampling sizes scaled linearly with the input sizes, the complexity of the sampling function can be reduced to $O(H_r W_r D)$. Thus, an optimized implementation which considers the sparsity of the matrices could significantly reduce the complexity and latency, and allow SSBNet to scale well with high dimension input.

4 Experiments

In this section, we first train SSBNet and the baseline models for image classification tasks on the ImageNet dataset [21], and then fine-tune the models to the object detection and instance segmentation tasks on the COCO dataset [16]. After that, we report the inference performance of SSBNet. All experiments were conducted with TensorFlow 2.6 [1] and Model Garden [30], and ran on TPU v2-8/v3-8 with bfloat16, except for Section 4.3. For ease of presentation, we denote the configurations for the last L groups of SSBNets by (M_1, ...,M_L). Here, M_l indicates that the sampling size of the last $(L - l + 1)$-th group is $M_l \times M_l$.

4.1 Image Classification

For image classification, we trained SSBNets and the baseline models on the ImageNet [21] dataset, which contains 1.28M training and 50k validation samples. We built SSBNets based on ResNet-D [8], ResNet-RS [2], EfficientNet [25] and BoTNet-S1 [23], and compare their performance with the original models. Due to limited resources, we only trained some variants of ResNet-RS and found that the results are close to the original work [2]. Thus, we will report other results directly from the original paper. For EfficientNet and BotNet-S1, we were not able to reproduce the same results from the papers [23, 25]. For fair comparison, we trained and reported all variants that have similar complexity as SSB-EfficientNet and SSB-BoTNet-S1.

Note that in this paper, we focus on the theoretical improvement regarding the accuracy-FLOPS trade-off. In Section 4.3, we will compare the inference time between SSBNet and the baselines, which shows that real speedup is achievable.

[1] Consider a weight matrix G^y with dimensions $H_r \times H_{in}$. If the (i, j)-th element of the weight matrix is non-zero, the next non-zero index will be $(i+\Delta i, j+\Delta j)$, where Δi, Δj are non-negative integers with either $i + \Delta i > i$ or $j + \Delta j > j$. As a result, there are at most $H_{in} + H_r$ non-zero elements. The same is true for G^x.

Table 1: ImageNet results of (SSB-)ResNet-D and (SSB-)ResNet-RS

Model	Input size	Params	FLOPS	Top-1(%)
R-50	224 × 224	25.6M	4.3G	78.1
S-50	224 × 224	25.6M	3.0G	78.1
R-101	224 × 224	44.6M	8.0G	79.5
S-101	224 × 224	44.6M	4.3G	78.9
R-152	224 × 224	60.2M	11.8G	80.1
S-152	224 × 224	60.3M	5.6G	79.2

R: ResNet-D [8] S: SSB-ResNet-D

Model	Input size	Params	FLOPS	Top-1(%)
R-50	224 × 224	25.6M	4.3G	78.2
S-50	224 × 224	25.6M	3.0G	78.2
R-101	224 × 224	44.6M	8.0G	80.0
S-101	224 × 224	44.6M	4.3G	79.5
R-152	224 × 224	60.2M	11.8G	80.6
S-152	224 × 224	60.3M	5.6G	80.1

R: ResNet-D [8] + RandAugment [3]
S: SSB-ResNet-D + RandAugment [3]

Model	Input size	Params	FLOPS	Top-1(%)
R-50	160 × 160	35.7M	2.3G	78.8
S-50	160 × 160	35.7M	1.6G	78.2
R-101	160 × 160	63.6M	4.2G	80.3*
S-101	160 × 160	63.6M	2.3G	79.3
R-101	192 × 192	63.6M	6.0G	81.3
S-101	192 × 192	63.6M	3.2G	80.6
R-152	192 × 192	86.6M	9.0G	82.0*
S-152	192 × 192	86.7M	4.3G	81.0
R-152	224 × 224	86.6M	12.0G	82.5
S-152	224 × 224	86.7M	5.8G	81.7
R-152	256 × 256	86.6M	15.5G	83.0*
S-152	256 × 256	86.7M	7.5G	82.1
R-200	256 × 256	93.2M	20.0G	83.4*
S-200	256 × 256	93.3M	9.4G	82.6

R: ResNet-RS [2] S: SSB-ResNet-RS
*: from the original paper

Comparison with ResNet-D. For ResNet-D [8] and SSB-ResNet-D, we trained three scales with the configuration of ResNet50/101/152 [7]. We followed the training and testing settings of [8] with batch size of 1024 and input size of 224×224. The sampling sizes of SSB-ResNet-D are $(16, 8, 4)$. The results are shown in the top-left table of Table 1, which clearly demonstrate the advantage of SSB-ResNet-D. For example, SSB-ResNet-D-50 achieved similar accuracy as ResNet-D-50, where the FLOPS is reduced by 30%. SSB-ResNet-D-101 has the same FLOPS as ResNet-D-50, but achieved 0.8% higher performance. The deepest SSB-ResNet-D-152 also performed only 0.3% worse than ResNet-D-101, with 30% less FLOPS.

In addition, we conducted experiments with RandAugment [3] as data augmentation. The number of transformations and the magnitude were 2 and 5, respectively. It can be observed from the bottom-left table of Table 1 that SSB-ResNet-D-101 outperformed ResNet-D-50 by 1.3% accuracy with the same FLOPS, and SSB-ResNet-D-152 achieved similar accuracy as ResNet-D-101 but saved 30% operations. This indicates that SSBNets can benefit more from stronger regularization.

Comparison with ResNet-RS. We also conducted experiments with ResNet-RS [2] by followed the same training settings in the original paper. We trained (SSB-)ResNet-RS-50/101/152/200, with different input sizes of $160 \times 160/192 \times 192/224 \times 224/256 \times 256$, and the sampling sizes are scaled to (12, 6, 3)/(14, 7, 3)/(16, 8, 4)/(18, 9, 5), respectively. The performance comparison between SSB-ResNet-RS and ResNet-RS is shown Table 1(right). The SSB-ResNet-RS achieved competitive results. For example, the SSB-ResNet-RS-200 with input size of 256×256 achieved 0.6% higher accuracy than ResNet-RS-152 with input

Table 2: ImageNet results of (SSB-)EfficientNet and (SSB-)BoTNet-S1

Model	Input size	Params	FLOPS	Top-1(%)
E-B0	224 × 224	5.3M	0.4G	76.4
S-B0	224 × 224	5.3M	0.3G	75.4
E-B1	240 × 240	7.9M	0.7G	78.5
S-B1	240 × 240	7.9M	0.5G	77.4
E-B2	260 × 260	9.2M	1.0G	79.6
S-B2	260 × 260	9.2M	0.7G	78.6
E-B3	300 × 300	12.3M	1.8G	81.0
S-B3	300 × 300	12.3M	1.3G	80.1

E: EfficientNet [25] S: SSB-EfficientNet

Model	Input size	Params	FLOPS	Top-1(%)
B-59	224 × 224	30.5M	7.3G	81.1
S-59	224 × 224	30.5M	3.8G	80.3
B-110	224 × 224	51.7M	10.9G	82.1
S-110	224 × 224	51.8M	5.1G	81.2
B-128	256 × 256	69.1M	19.3G	82.9
S-128	256 × 256	69.1M	8.1G	82.0
B-77	320 × 320	47.9M	23.3G	–
S-77	320 × 320	47.9M	10.2G	82.5

B: BoTNet-S1 [23] S: SSB-BoTNet-S1

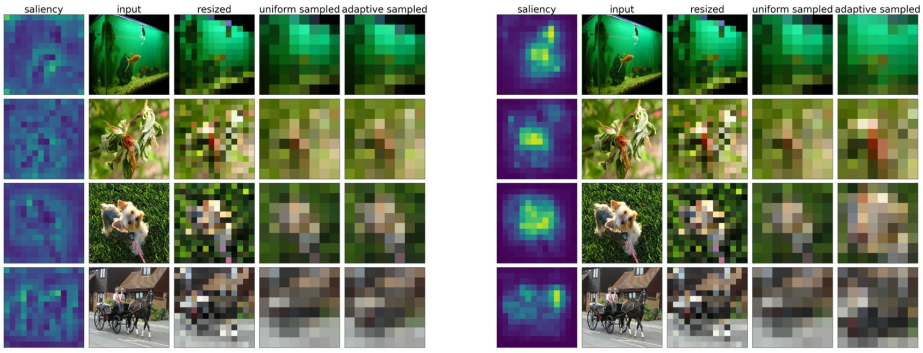
size of 192 × 192, where the FLOPS is only 4.4% higher. Figure 1a compares SSB-ResNet-RS with ResNet-RS with less than 10 GFLOPS and shows that SSB-ResNet-RS achieved better accuracy to FLOPS ratio in different scales.

Comparison with EfficientNet. For EifficientNet [25], we followed the original setting, except that we used a batch size of 1024. Due to the use of 5 × 5 convolutions, we applied larger sampling size for EfficientNet. Specifically, we used (20, 10, 10, 5, 5)/(22, 11, 11, 5, 5)/(24, 12 ,12 ,6, 6) and (26, 13 ,13 ,7, 7) for SSB-EfficientNet-B0/1/2/3, respectively.

However, adaptive sampling did not improve EfficientNet in our experiments as shown in Table 2(left). For example, SSB-EfficientNet-B2 has less number of operations than EfficientNet-B2, but nearly the same number of operations and accuracy as EfficientNet-B1. This may be due to the fact that EfficientNet is designed by neural architecture search, thus the network configurations and training parameters do not transfer well to SSB-EfficientNet. We also note that the speed-up in EfficientNet is limited when compared with ResNet. This is because EfficientNet has more groups of layers and we didn't replace the first layer in each group, due to the reason discussed in Section 3.1.

Comparison with BoTNet-S1. The recent development of Visual Transformers [5] has gained attention in the research community. However, training Transformers is challenging. For example, larger dataset or additional augmentation is required [5, 27]. To evaluate the compatibility of adaptive sampling with self-attention layer, we built SSBNet from BoTNet-S1 [23], which is a hybrid network composed of both convolutional and self-attention layers. It inherited the techniques from modern CNNs, including the choice of normalization layers, optimizers, and training setting.

Results in Table 2(right) show that SSB-BoTNet-S1 achieved better accuracy to FLOPS trade-off. For example, SSB-BoTNet-S1-110 performed similar as BoTNet-S1-59, but with 30% less FLOPS; SSB-BoTNet-S1-77 achieved 0.4% higher accuracy than BoTNet-S1-110, but 6% less FLOPS. The results also suggest that adaptive sampling does not only improve CNNs, but also hybrid



(a) Example 1 (b) Example 2

Fig. 3: Examples of the saliency map and sampling output. (a) and (b) are the samples from different layers of the SSB-ResNet-RS-152.

Table 3: COCO-2017 [16] results of SSB-ResNet-D/ResNet-D [8] with FPN [15] and Mask R-CNN [6]

	Model	AP_{box}	AP_{mask}		Model	AP_{box}	AP_{mask}
12 Epochs	ResNet-D-50	40.14	35.72	36 Epochs	ResNet-D-50	42.50	37.56
	SSB-ResNet-D-50	40.67	35.98		SSB-ResNet-D-50	42.89	37.84
24 Epochs	ResNet-D-50	41.95	37.17				
	SSB-ResNet-D-50	42.36	37.41				

networks that utilize self-attention. Figure 1b shows the comparison between SSB-BoTNet-S1 and BoTNet-S1 that have less than 11 GFLOPS, where SSB-BoTNet-S1 achieved better accuracy to FLOPS ratio.

Visualization. The outputs of two sampled layers of SSBNet are shown in Figure 3. While the high dimension features are hard to visualize, we first resized the original images to the same size as the sampler input, and then applied sampling on the resized images for visualization. Figure 3a shows that the first reported layer samples from the whole image, where the background is weighted heavier; Figure 3b shows that the second reported layer is able to zoom into smaller regions when performing downsampling, which can better preserve the discriminative features in these regions. The results also suggest that different layers of the SSBNet zoom into different regions, which justifies the use of adaptive sampling in multiple layers.

4.2 Object Detection and Instance Segmentation

We also evaluated the performance of SSBNet for object detection and instance segmentation tasks on the COCO-2017 dataset [16] which contains 118K images in the training set and 5K images for validation. For that purpose, we used

Table 4: Latency comparison between (SSB-)ResNet-RS and (SSB-)BoTNet-S1

Model	Input size	Params	FLOPS	Latency(ms)
R-50	160 × 160	35.7M	2.3G	130/163
S-50	160 × 160	35.7M	1.6G	117/138
R-101	192 × 192	63.6M	6.0G	298/381
S-101	192 × 192	63.6M	3.2G	235/267
R-152	224 × 224	86.6M	12.0G	565/726
S-152	224 × 224	86.7M	5.8G	427/472
R-200	256 × 256	93.2M	20.0G	988/1244
S-200	256 × 256	93.2M	9.4G	744/822

R: ResNet-RS [2] S: SSB-ResNet-RS
First number: Latency on V100 GPU
Second number: Latency on 3090 GPU

Model	Input size	Params	FLOPS	Latency(ms)
B-59	224 × 224	30.5M	7.3G	404/469
S-59	224 × 224	30.5M	3.8G	291/301
B-110	224 × 224	51.7M	10.9G	559/674
S-110	224 × 224	51.8M	5.1G	386/406
B-128	256 × 256	69.1M	19.3G	925/1127
S-128	256 × 256	69.1M	8.1G	612/639
B-77	320 × 320	47.9M	23.3G	1234/1487
S-77	320 × 320	47.9M	10.2G	773/784

B: BoTNet-S1 [23] S: SSB-BoTNet-S1
First number: Latency on V100 GPU
Second number: Latency on 3090 GPU

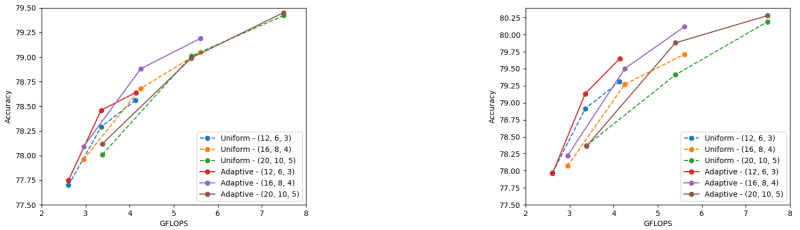
the pre-trained ResNet-D-50 [8] and SSB-ResNet-D-50 with FPN [15] as the backbone, and applied Mask R-CNN [6] for object detection and segmentation. The same was applied to the baseline models for comparison purposes. We used the default setting of Mask R-CNN in Model Garden [30], with input size of 1024×1024, batch size of 64 and a learning rate of 0.01. Horizontal flipping and Scale jitter with a range between 0.8 and 1.25 were applied. For SSB-ResNet-D, we used the sampling sizes of (72, 36, 18) for the last 3 groups. The results of training with 12/24/36 epochs are reported.

The results in Table 3 show that SSBNet is transferable to new tasks with high performance. While the pre-trained ResNet-D-50 and SSB-ResNet-D-50 have similar accuracy on ImageNet [21], the SSB-ResNet-D-50 performs slightly better than ResNet-D-50 on COCO-2017, with less operations. This may be due to two reasons: 1.) there are paddings to the images, but SSBNet can zoom into the non-padding regions such that no computation is wasted, 2.) the images from COCO dataset have higher resolution than those from ImageNet.

4.3 Latency Comparison between SSBNet and the Original Networks

To explore the actual speed-up by adaptive sampling, we implemented the sampling function in TensorFlow 2.6 [1] and CUDA [17], and performed comparison between ResNet-RS [2], SSB-ResNet-RS, BoTNet-S1 [23], and SSB-BoTNet-S1. Experiments were conducted on V100 and 3090 GPU with float16. We report the results from two GPUs as we noticed the difference in performance. Specifically, V100 is commonly used in the literatures, but our implementation performs better in 3090, which is possibly due to the degree of optimization. The results are reported by the batch latency with size of 1024.

The results in Table 4 show that actual speed-up is achievable with adaptive sampling. For example, the latency of SSB-ResNet-RS-200 is reduced by up to 34% when compared with ResNet-RS-200, where ideally the latency can be reduced by 53%. The latency of SSB-BoTNet-S1-77 is 47% lower than BoTNet-S1-77, which is close to the theoretical improvement, i.e. 56%. We would like to



(a) Normal (b) With RandAugment [3]

Fig. 4: Comparison between uniform and adaptive sampling. In each line, the three points denote SSB-ResNet-D-50/101/152, respectively.

highlight that we only did limited implementation optimization over SSBNet. We expect further improvement with better optimization.

5 Ablation study

In this section, we provide results of additional experiments to justify the use of adaptive sampling in deep neural networks.

5.1 Comparison between Uniform and Adaptive Sampling

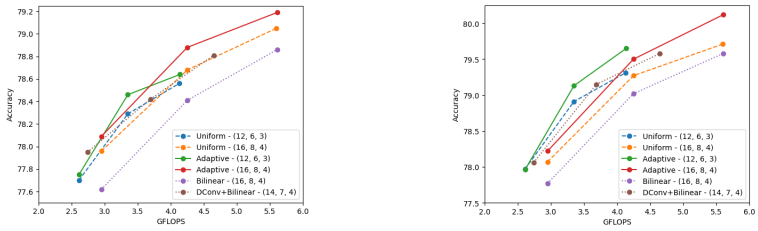
To validate the effectiveness of adaptive sampling utilized in SSBNet, we compared the performance of two SSB-ResNet-D networks, which applied adaptive and uniform sampling, respectively. For a fair comparison, the two networks used the same sampling mechanism (Equation 8).

The results in Figure 4a show that, in general, the networks with adaptive sampling outperform the networks that applied uniform sampling. We observed the largest difference from SSB-ResNet101-D, where the adaptive model achieved 0.2% higher accuracy. In Figure 4b, results with RandAugment [3] are shown, where models with adaptive sampling obtain larger improvement than those with uniform sampling. For example, at sampling sizes (16, 8, 4), the SSB-ResNet152-D with adaptive sampling achieved 0.4% higher accuracy.

The results suggest that adaptive sampling is a better choice for downsampling in SSBNet. In addition, the results show that the SSB-ResNet-D with sampling sizes of (16, 8, 4) achieved a good trade-off between accuracy and FLOPS at different depths. Thus, we used this configuration as default.

5.2 Comparison with Other Sampling Methods

To compare different down/upsampling mechanisms, we conducted experiments with different sampling methods that can be applied in the bottleneck structure: 1) the proposed adaptive sampling; 2) the uniform sampling used in Section 5.1; 3) the uniform sampling with bilinear interpolation; and 4) the depthwise



(a) Normal (b) With RandAugment [3]

Fig. 5: Comparison between adaptive sampling and other sampling methods. In each line, the three points denote SSB-ResNet50/101/152-D, respectively.

convolution for downsampling with bilinear sampling for upsampling [14]. For bilinear sampling, we used sampling sizes of $(16, 8, 4)$ as it is the common choice in our paper. For 4), the kernel size and stride of depthwise convolutions are 5 and 2 respectively, with sampling sizes of $(14, 7, 4)$. For a fair comparison, we included the results of adaptive sampling with sizes $(16, 8, 4)$ and $(12, 6, 3)$, such that the cost of 4) falls between them.

Figure 5a shows the results with basic training setting and Figure 5b shows the results with RandAugment [3]. In both settings, adaptive sampling outperformed other methods, especially when the model is deeper and additional data augmentation is used. Surprisingly, although method 2) is also a uniform sampling method, it outperformed the widely utilized method 3).

6 Conclusion

In this paper, we proposed a novel architecture to apply adaptive sampling in the main building block of deep neural networks. The proposed SSBNet outperformed other benchmarks in both image classification and object detection tasks. Different from most existing works that applied adaptive sampling for specific tasks [4, 11, 12, 20, 26, 32] and performed very few sampling operations, the proposed structure can work as a backbone network and be transferred to different tasks. Visualization illustrated SSBNet's capability in sampling different regions at different layers and ablation studies demonstrated that adaptive sampling is more efficient than uniform sampling.

The results in this paper suggest that adaptive sampling is a promising mechanism in deep neural networks. We expect that designing the network with adaptive sampling from scratch and fine-tuning the training process may provide further performance improvement.

Acknowledgement

This work was supported by the Cloud TPUs from Google's TPU Research Cloud (TRC) and the HKUST-WeBank Joint Lab under Grant WEB19EG01-L.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jozefowicz, R., Jia, Y., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Schuster, M., Monga, R., Moore, S., Murray, D., Olah, C., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow, Large-scale machine learning on heterogeneous systems (11 2015). https://doi.org/10.5281/zenodo.4724125 8, 12
2. Bello, I., Fedus, W., Du, X., Cubuk, E.D., Srinivas, A., Lin, T.Y., Shlens, J., Zoph, B.: Revisiting resnets: Improved training and scaling strategies. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems. vol. 34, pp. 22614–22627. Curran Associates, Inc. (2021) 2, 3, 5, 8, 9, 12
3. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.: Randaugment: Practical automated data augmentation with a reduced search space. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 18613–18624. Curran Associates, Inc. (2020) 9, 13, 14
4. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (Oct 2017) 1, 4, 14
5. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2021) 1, 3, 10
6. He, K., Gkioxari, G., Dollar, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (Oct 2017) 11, 12
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016) 1, 2, 4, 5, 6, 7, 9
8. He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., Li, M.: Bag of tricks for image classification with convolutional neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019) 5, 8, 9, 11, 12
9. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018) 3
10. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning. pp. 448–456. PMLR (2015) 6
11. Jaderberg, M., Simonyan, K., Zisserman, A., kavukcuoglu, k.: Spatial transformer networks. In: Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 28. Curran Associates, Inc. (2015) 1, 4, 6, 7, 14
12. Jin, C., Tanno, R., Mertzanidou, T., Panagiotaki, E., Alexander, D.C.: Learning to downsample for segmentation of ultra-high resolution images. In: International Conference on Learning Representations (2022) 1, 4, 14
13. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C., Bottou, L., Weinberger,

K. (eds.) Advances in Neural Information Processing Systems. vol. 25. Curran Associates, Inc. (2012) 1, 4
14. Li, D., Zhou, A., Yao, A.: Hbonet: Harmonious bottleneck on two orthogonal dimensions. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (October 2019) 1, 4, 14
15. Lin, T.Y., Dollar, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (July 2017) 11, 12
16. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 740–755 (2014) 8, 11
17. Nickolls, J., Buck, I., Garland, M., Skadron, K.: Scalable parallel programming with cuda: Is cuda the parallel programming model that application developers have been waiting for? Queue **6**(2), 40–53 (2008) 12
18. Peng, J., Xie, L., Zhang, Z., Tan, T., Wang, J.: Accelerating deep neural networks with spatial bottleneck modules. arXiv preprint arXiv:1809.02601 (2018) 1, 4
19. Ramachandran, P., Parmar, N., Vaswani, A., Bello, I., Levskaya, A., Shlens, J.: Stand-alone self-attention in vision models. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc. (2019) 3
20. Recasens, A., Kellnhofer, P., Stent, S., Matusik, W., Torralba, A.: Learning to zoom: a saliency-based sampling layer for neural networks. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 51–66 (2018) 1, 2, 4, 14
21. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision (IJCV) **115**(3), 211–252 (2015). https://doi.org/10.1007/s11263-015-0816-y 2, 3, 8
22. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (2015) 1, 4
23. Srinivas, A., Lin, T.Y., Parmar, N., Shlens, J., Abbeel, P., Vaswani, A.: Bottleneck transformers for visual recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 16519–16529 (June 2021) 2, 3, 8, 10, 12
24. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2015) 1, 4
25. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning. pp. 6105–6114. PMLR (2019) 5, 8, 10
26. Thavamani, C., Li, M., Cebron, N., Ramanan, D.: Fovea: Foveated image magnification for autonomous navigation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 15539–15548 (October 2021) 1, 4, 14
27. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: International Conference on Machine Learning. pp. 10347–10357. PMLR (2021) 10
28. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio,

S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017) 3
29. Woo, S., Park, J., Lee, J.Y., Kweon, I.S.: Cbam: Convolutional block attention module. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 3–19 (2018) 3
30. Yu, H., Chen, C., Du, X., Li, Y., Rashwan, A., Hou, L., Jin, P., Yang, F., Liu, F., Kim, J., Li, J.: TensorFlow Model Garden. https://github.com/tensorflow/models (2020) 8, 12
31. Zhao, Z.Q., Zheng, P., Xu, S.t., Wu, X.: Object detection with deep learning: A review. IEEE transactions on neural networks and learning systems **30**(11), 3212–3232 (2019) 2
32. Zheng, H., Fu, J., Zha, Z.J., Luo, J.: Looking for the devil in the details: Learning trilinear attention sampling network for fine-grained image recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019) 1, 2, 4, 6, 14