Appendix: Spatially Adaptive Inference with Stochastic Feature Sampling and Interpolation

Zhenda Xie[1,2†⋆], Zheng Zhang[2⋆‡], Xizhou Zhu[2,3⋆†], Gao Huang[4], and Stephen Lin[2]

[1] Tsinghua University
xzd18@mails.tsinghua.edu.cn
[2] Microsoft Research Asia
{zhez,stevelin}@microsoft.com
[3] University of Science and Technology of China
ezra0408@mail.ustc.edu.cn
[4] Tsinghua University, Department of Automation
gaohuang@tsinghua.edu.cn

1 Experimental Settings

Object detection All models are trained on the 118k images of the COCO 2017 [7] train set, and evaluated on the 5k images of the COCO 2017 validation set. The standard mean average precision (mAP) is used to measure accuracy. The baseline model is based on Faster R-CNN with Feature Pyramid Network (FPN) [6] and deformable convolution [3]. ImageNet [4] pre-trained ResNet-101 [5] is chosen as the default backbone model. Following [12], all the 3×3 convolutions from the conv3 to conv5 stages are replaced by deformable convolutions.

The implementation and hyper-parameters are based on mmdetection [1]. Anchors with 5 scales and 3 aspect ratios are used. 2k and 1k region proposals are generated with a non-maximum suppression threshold of 0.7 at training and inference, respectively. The network is trained for 12 epochs on 8 GPUs with 1 image per GPU. In SGD training, the learning rate is initialized to 0.01 and is divided by 10 at the 8th and 11th epochs. The weight decay and the momentum parameters are set to 0.0001 and 0.9, respectively.

Semantic segmentation All models are trained on the 2975 finely annotated images of the Cityscapes [2] train set and evaluated on the 500 images of the validation set. Accuracy is measured by the standard mean IoU. The baseline model is ResNet-50 based dilated FCN [8] with deformable convolution layers [3]. Following [12], we use deformable convolutions to replace all 3×3 convolutions from the conv3 to conv5 stages; the strides of the conv4 and conv5 stages are set to 1, and the dilations of all 3×3 convolutions in the conv4 and conv5 stages are set to 2 and 4, respectively.

The implementation and hyper-parameters are based on the open-source implementation of TorchCV [11]. The networks are trained on 4 GPUs with 2

⋆ Equal contribution. †This work is done when Zhenda Xie and Xizhou Zhu are interns at Microsoft Research Asia. ‡Corresponding author.

Table 1. The numerical results of Fig. 4 (a) in the main paper. Experiments are conducted on object detection the with COCO2017 validation set

Model	mAP	GFLOPs
ResNet-50(1000 px)	41.2	149.2
Uniform Sampling(1000 px)	43.4	289.5
Uniform Sampling(800 px)	42.3	184.1
Uniform Sampling(600 px)	40.4	100.2
Uniform Sampling(500 px)	38.5	70.0
Determ Gumble-Softmax(0.02)	43.1	184.9
Determ Gumble-Softmax(0.05)	42.8	150.7
Determ Gumble-Softmax(0.1)	42.0	121.5
Determ Gumble-Softmax(0.2)	40.7	94.3
ReLU(0.02)	43.4	252.8
ReLU(0.05)	43.0	202.0
ReLU(0.1)	42.7	181.9
ReLU(0.2)	41.2	135.6
Ours(0.02)	43.3	160.4
Ours(0.05)	42.8	122.8
Ours(0.1)	42.0	96.6
Ours(0.2)	40.7	73.3

images per GPU for 60k iterations. The SGD optimizer with the poly learning rate policy is employed. The initial learning rate is 0.01 and the decay exponent is 0.9. The weight decay and momentum are set to 0.0001 and 0.9, respectively. In the training stage, the data is augmented with random scaling (from 0.5 to 2.0), random cropping and random horizontal flipping. Synchronized Batch Normalization [9] with learnable weights is placed after every newly added layer.

2 Numerical Results

Object detection Numerical results of Fig. 4 (a) in the main paper are shown in Table 1. For the ResNet-50 model, we resize the sides of input images to 1000. For uniform sampling, we resize the shorter sides of input images to {1000, 800, 600, 500} to generate results under different FLOPs. For our method and the two deterministic methods (deterministic Gumbel-Softmax and ReLU), we obtain results by adopting different sparse loss weights, i.e. {0.2, 0.1, 0.05, 0.02}.
Semantic segmentation The numerical results of Fig. 4 (b) in the main paper are shown in Table 2. For uniform sampling, we resize the shorter sides of input images to {1024, 896, 736, 512} to obtain results under different FLOPs. For our method and deterministic Gumbel-Softmax, we generate results by adopting different sparse loss weights, i.e. {0.3, 0.2, 0.1, 0.05}.

3 Inference Stability

Our stochastic Gumbel-Softmax sampling method has randomness in the inference phase, so the results may differ with each test. In this section, we

Table 2. Numerical results of Fig. 4 (b) in the main paper. Experiments are conducted on semantic segmentation with the Cityscapes validation set

Model	mean IoU	GFLOPs
Uniform Sampling(1024 px)	80.82	920.6
Uniform Sampling(896 px)	80.67	704.8
Uniform Sampling(736 px)	79.37	475.6
Uniform Sampling(512 px)	77.24	230.1
Determ Gumble-Softmax(0.05)	80.56	463.4
Determ Gumble-Softmax(0.1)	79.90	373.2
Determ Gumble-Softmax(0.2)	78.90	334.4
Determ Gumble-Softmax(0.3)	78.37	311.1
Ours(0.05)	80.60	373.2
Ours(0.1)	80.27	328.8
Ours(0.2)	79.83	275.4
Ours(0.3)	79.59	252.9

Table 3. Evaluation of inference stability on object detection (COCO2017 validation)

Loss Weight	Grid Prior	mAP mean / std	GFLOPs mean / std
0.02	✓	43.28/0.02	160.35/0.02
0.05	✓	42.81/0.02	122.79/0.03
0.1	✓	41.98/0.02	96.64/0.03
0.2	✓	40.69/0.03	73.25/0.02
0.02		42.61/0.11	156.99/0.02
0.05		41.25/0.05	113.47/0.02
0.1		38.59/0.09	85.87/0.02
0.2		32.63/0.08	65.03/0.03

evaluate the inference stability of our model (ResNet-101) on object detection (COCO2017 validation). We evaluate multiple models, which were trained by different loss weights. For each model, we test five times and report the mean and standard deviation of the mAP and FLOPs. The results are shown in Table 3. For all models, the variance of mAP and FLOPs is small, indicating that our method has strong stability in the inference phase. We speculate that this stability may arise from the grid prior. Therefore, we further evaluate the testing stability without the grid prior and found that the variance of mAP over multiple tests increased from about 0.02 to 0.09.

4 Compatibility with Pruning Method

High computational costs are alleviated in our work by exploring spatial redundancy, an approach that differs from other techniques such as model pruning. For further verification that our method is compatible with pruned models, we

Table 4. Studying the compatibility with pruning method. We applied the global unstructured pruning method on baseline model and our model (sparse loss weight of 0.02) with various pruning ratios.

Prune Ratio	Baseline			Ours(0.02)		
	params(M)	GFLOPs	mAP	params(M)	GFLOPs	mAP
0.00	47.25	289.5	43.4	47.80	160.4	43.3
0.50	26.05	166.4	42.8	26.60	96.8	42.7
0.55	23.93	155.5	42.1	24.48	91.0	42.2
0.60	21.81	144.5	40.9	22.36	85.0	41.1
0.66	19.69	133.3	38.4	20.24	78.8	38.8
0.79	17.57	122.0	32.7	18.12	72.5	34.0
0.75	15.45	110.3	20.3	16.0	65.7	23.4

applied the global unstructured pruning method provided in the official PyTorch implementation (`torch.nn.utils.prune.global_unstructured`) on the backbone network, but exclude the offset convolution layer used in deformable convolution and the mask prediction layer of our method. A comparison between the baseline model and our model on the COCO object detection benchmark is shown in Table 4. Since our method introduces additional parameters to predict the sampling positions, the number of parameters is slightly larger than that of the baseline model. The results show that at various pruning levels, the mAP performance of our model is comparable to that of the baseline model. Moreover, it is seen that our method consistently uses less FLOPs than the baseline model under similar mAP. These results indicate that our method is complementary to and compatible with the pruning method.

5 Analysis on Final Temperature

Intuitively, the decay factor should be related to the training dynamics, rather than to the tasks. Ideally, the decay factor should not be too large or too small. If it is too small, the temperature will quickly drop to near zero, and then the mask will degenerate to binary, resulting in no gradient and insufficient training. On the other hand, if the factor is too large, the temperature will still be high at the end of training, and the mask is not binary in this case, which will result in inconsistency between inference and training. In principle, we want the temperature to be close to 0 at the end of training.

In practice, we tried different decay factors on COCO, so that the final temperature at the end of training is 0.03, 0.01 or 0.005. We found the difference in performance to be small (less than 0.2 mAP and 3 GFLOPs for different models), showing that the final temperature has little effect on the results within a reasonable interval. In addition, we did not tune the final temperature on other tasks, but directly adopted 0.01 and found it to work well. This may partly indicate that the decay factor is insensitive to different tasks.

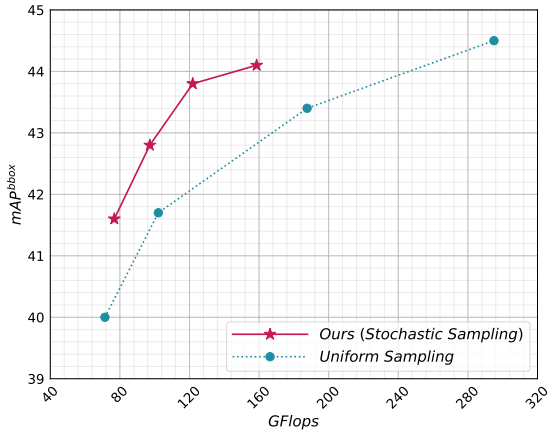


Fig. 1. Comparison of our method and uniform sampling on object detection (COCO2017 validation). ResNeXt is chosen as the backbone model. Curve of "Uniform Sampling" is drawn from various resolutions. Curve of our method is drawn from various sparse loss weights (with shorter side of 1000 pixels).

Table 5. Numerical results of Fig. 1. ResNeXt is chosen as the backbone model. Experiments are conducted on object detection with the COCO2017 validation set

Model	mAP	GFLOPs
Uniform Sampling(1000 px)	44.5	295.0
Uniform Sampling(800 px)	43.4	187.6
Uniform Sampling(600 px)	41.7	102.1
Uniform Sampling(500 px)	40.0	71.4
Ours(0.02)	44.1	158.6
Ours(0.05)	43.8	121.9
Ours(0.1)	42.8	97.2
Ours(0.2)	41.6	76.8

6 Performance on ResNeXt

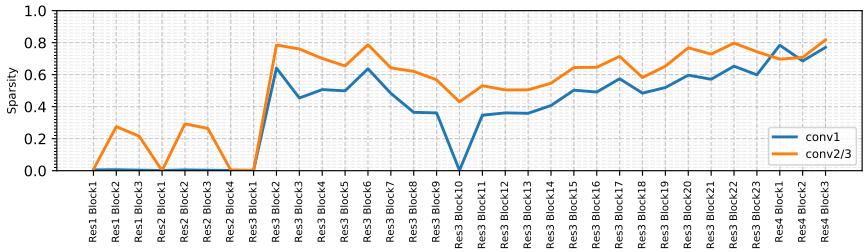
To examine the compatibility of our method with other network architectures, we integrate it with ResNeXt [10] and evaluate the performance on object detection. ImageNet pre-trained ResNeXt-101($32\times4d$) with deformable convolution is chosen as the backbone model. We use the same hyper-parameters and training setting as in Sec. 4.1 of the main paper. For uniform sampling models, we resize the shorter sides of input images to {1000, 800, 600, 500}. For our method, we use different loss weights {0.2, 0.1, 0.05, 0.02} to draw the speed-accuracy curve. The results are shown in Fig. 1 and the numerical results are shown in Table 5. Our method is found to outperform uniform sampling by a large margin.

7 Sparsity of different blocks

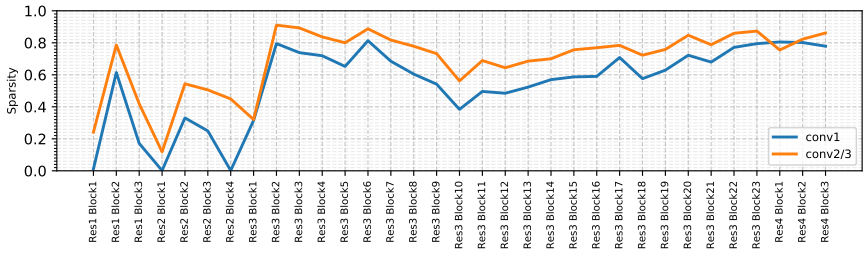
We evaluate the average sparsity of different blocks on the COCO2017 validation set for models trained under different loss weights. Results are shown in Fig. 2. For all the models, we observed that deeper blocks are more sparse, and this phenomenon is more pronounced in models trained with small loss weights. For example, in the model trained with a loss weight of 0.02, the sparsity of conv1 is close to 0 from Res1-Block1 to Res2-Block4. One possible reason is that shallow blocks primarily compute local visual features (such as edges and textures), so their spatial redundancy is small, while deeper blocks are more likely to capture semantic features which are more redundant. Another observation is that the sparsity of conv2 and conv3 are always greater than the sparsity of conv1. The reason is unclear but we suspect that this phenomenon may be related to the receptive field of operators.

References

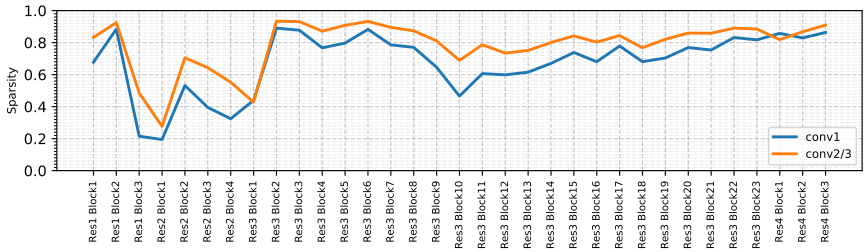
1. Chen, K., Pang, J., Wang, J., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Shi, J., Ouyang, W., Loy, C.C., Lin, D.: mmdetection. https://github.com/open-mmlab/mmdetection (2018)
2. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: CVPR (2016)
3. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: CVPR (2017)
4. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009)
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
6. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR (2017)
7. 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: ECCV (2014)
8. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR (2015)
9. Peng, C., Xiao, T., Li, Z., Jiang, Y., Zhang, X., Jia, K., Yu, G., Sun, J.: Megdet: A large mini-batch object detector. In: CVPR (2018)
10. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1492–1500 (2017)
11. You, A., Li, X., Zhu, Z., Tong, Y.: Torchcv: A pytorch-based framework for deep learning in computer vision. https://github.com/donnyyou/torchcv (2019)
12. Zhu, X., Hu, H., Lin, S., Dai, J.: Deformable convnets v2: More deformable, better results. In: CVPR (2019)



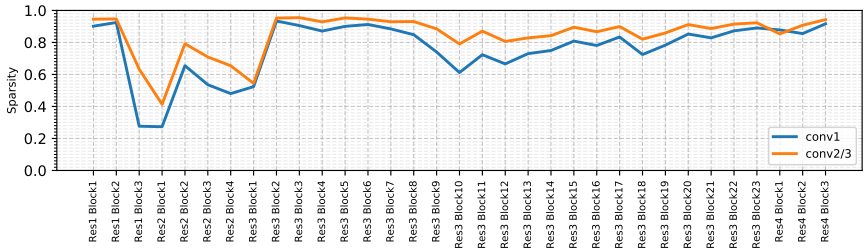
(a) Model trained on loss weight 0.02



(b) Model trained on loss weight 0.05



(c) Model trained on loss weight 0.10



(d) Model trained on loss weight 0.20

Fig. 2. Sparsity of different residual blocks. The evaluated models are trained with different loss weights.