SpotPatch: Parameter-Efficient Transfer Learning for Mobile Object Detection

Keren Ye*[1][0000-0002-7349-7762], Adriana Kovashka[1][0000-0003-1901-9660], Mark Sandler[2][0000-0003-0352-6051], Menglong Zhu[2][0000-0003-4796-6235], Andrew Howard[2], and Marco Fornoni[2][0000-0001-5538-8012]

[1] University of Pittsburgh, Pittsburgh PA 15260, USA
[2] Google Research, Paris 75009, France

1 Introduction

As mobile hardware technology advances, on-device computation is becoming more and more affordable. On one hand, efficient backbones like MobileNets optimize feature-extraction costs by decomposing convolutions into more efficient operations. On the other hand, one-stage detection approaches like SSD provide mobile-friendly detection heads. As a result, detection models are now massively being moved from server-side to on-device. While this constitutes great progress, it also brings new challenges. Specifically, multiple isolated models are often downloaded to perform related tasks, like detecting faces, products, bar-codes, etc. This rises the questions: Can we represent a diverse set of detection tasks as a small set of "patches" applied to a common model? If the common model also needs to be updated, can we represent the update as a small patch too? To answer the above questions, we studied two experimental scenarios:

1. Adapting a mobile object detector to solve a new task.
2. Updating an existing model, whenever additional training data is available.

To learn the patches, we propose an approach simultaneously optimizing for both accuracy and footprint: (1) for each layer, we compactly represent the patch with scaled 1-bit weight residuals; (2) we employ a gating mechanism to adaptively patch only the most-important layers, reusing the original weights for the remaining ones. We evaluate on object detection tasks, using a setting similar to [9], which we refer to as "*Detection Decathlon*". We also showcase our method's ability to efficiently update a detector whenever new data becomes available. To the best of our knowledge this is the first systematic study of parameter-efficient transfer learning techniques on object detection tasks.

Related Work: [9] proposed a Visual Decathlon benchmark for adapting image classifiers. [1, 8] proposed to "patch" MobileNets by fine-tuning only batch-normalization and depthwise-convolution layers. [6, 7] used binary masks and simple linear transformations to obtain the target kernels. Their work is similar to quantization approaches [4, 5] but using 1-bit representation. [2] built a dynamic routing network to choose for each layer, between fine-tuning it and reusing the pre-trained weights. Our approach differs from the above in that we

* Work partially done during an internship at Google

combine scaled binary mask residuals, with a loss function explicitly minimizing the number of patched layers. This allows our model to adaptively minimize the patch footprint in a task-dependent fashion.

2 Approach

We assume a deep neural network $\mathcal{M}$ of depth N is composed of a set of layers $\boldsymbol{\theta} = \{\mathbf{W}_1, \ldots, \mathbf{W}_N\}$. To adapt $\mathcal{M}$ to solve a new task, we seek a task-specific $\boldsymbol{\theta}' = \{\mathbf{W}'_1, \ldots, \mathbf{W}'_N\}$ that optimizes the loss on the target dataset. In addition, since we do not want to fully re-learn $\boldsymbol{\theta}'$, we look for a transformation with minimum cost (in terms of bit-size) to convert the original $\boldsymbol{\theta}$ to $\boldsymbol{\theta}'$. Assume the transformation function can be expressed as $\boldsymbol{\theta}' = f(\boldsymbol{\theta}, \boldsymbol{\gamma})$ where $\boldsymbol{\gamma}$ is an additional set of parameters for the new task, and f is a function that combines the original parameters $\boldsymbol{\theta}$ with the parameter "patch" $\boldsymbol{\gamma}$. Our goal is to reduce the bit-size of $\boldsymbol{\gamma}$. In other words, we want the "footprint" of $\boldsymbol{\gamma}$ to be small.

Task-specific weight transform. Given the weights $\boldsymbol{\theta} = \{\mathbf{W}_1, \ldots, \mathbf{W}_N\}$ shared across tasks, the task-specific weight trasformation adds weight residuals $\omega_i \mathbf{S}_i$ to them: $\mathbf{W}'_i = \mathbf{W}_i + \omega_i \mathbf{S}_i$, where $\mathbf{S}_i$ is a binary mask of the same shape as $\mathbf{W}_i$ with values in $\{-1, +1\}$, and ω_i is a scaling factor. To learn these scaled and zero-centered residuals, we use similar techniques as [6, 7], which use real-valued mask variables to achieve differentiable gradients during training. To deploy the model, only the masks $\mathbf{S}_i$ and the per-layer scalar ω_i are used. The incremental footprint of the model is thus $\{\omega_i, \mathbf{S}_i | \; i \in \{1, \ldots, N\}\}$, or roughly 1-bit per model weight, with a negligible additional cost for the per-layer scalar ω_i

Spot patching. Since the difficulty of adapting a detector depends on the target task, we design a gating mechanism to adjust the model patch complexity in a task-dependent fashion: $\mathbf{W}'_i = \mathbf{W}_i + g_i(\omega_i \mathbf{S}_i)$. Simply speaking, we add a *gate* g_i for each network layer. The layer uses the original pre-trained weights if the gate value is 0, and weight transform otherwise (Fig. 1). The benefit of gating with g_i is that it allows to search for a task-specific subset of layers to patch, rather than patching all layers. Compared to patching the whole network, it reduces the patch footprint to $\boldsymbol{\gamma} = \{\omega_i, \mathbf{S}_i | \; i \in \{1, \ldots, N\} \text{ and } g_i = 1\}$. To learn g_i we use similar differentiable binarization techniques as for learning $\mathbf{S}_i$. Furthermore, to force the number of patched layers to be small, we add to the training loss a term minimizing the number of patched layers $\sum_{i=1}^{N} g_i$.

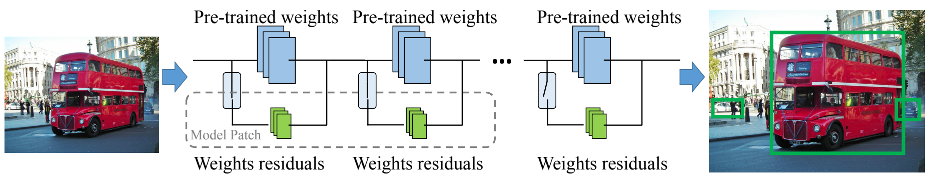

Fig. 1: Our method simultaneously reduces: (1) the bit-size of the weights residuals and (2) the number of patched layers.

Besides task-specific weight transform and spot patching, we also train task-specific Batch Normalization (BN), which does not significantly increase the footprint. To learn the task-specific patch $\boldsymbol{\gamma}$, we minimize the following loss: $L(\boldsymbol{\gamma}) = L_{det}(\boldsymbol{\gamma}) + \lambda_{sps} L_{sps}(\boldsymbol{\gamma}) + \lambda_{adp} L_{adp}(\boldsymbol{\gamma})$, where: $L_{det}(\boldsymbol{\gamma})$ is the detection loss optimizing the class confidence scores and box coordinates. $L_{sps}(\boldsymbol{\gamma}) = \sum_{i=1}^{N} g_i$ is the sparsity-inducing loss, pushing the number of patched layers to be small. Finally, $L_{adp}(\boldsymbol{\gamma}) = \sum_{i=1}^{N} \|\omega_i\|_2^2$ is a domain-adaptation loss forcing the scaling factors ω_i to be small, and thus $\boldsymbol{\theta}'$ to be similar to $\boldsymbol{\theta}$.

3 Experiments

We consider the two experimental scenarios proposed in the introduction. For the baselines, we compare our method with the following transfer learning methods, which we reproduced in the detection setting:

- Fine-Tuning [3] fine-tunes the whole network.
- Tower Patch [3] fine-tunes only the parameters in the detection head.
- Bn Patch [8], Dw Patch [8], and Bn+Dw Patch [1, 8] learn task-specific BatchNorm, Depthwise, or BatchNorm + Depthwise layers.
- Piggyback [6] and WeightTrans [7] learn task-specific 1-bit masks and simple linear transformations to obtain the task-related kernels.

Detection Decathlon. We adapt a model trained on OpenImages V4 to nine additional vertical-specific detection tasks: Caltech-UCSD Birds, Stanford Cars, COCO, Stanford Dogs, WiderFace, Kitti, Oxford-IIIT Pet, Retail Product Checkout (RPC), and Pascal VOC. We compare models on the basis of how well they solve all the problems (mAP@0.5), and how small is their relative footprint. Similarly to [9], we use a decathlon-inspired scoring function to evaluate the performance of each method. Scores are normalized so that the Fine-Tuning method reaches 2,500 points. We also report the Score/Footprint ratio [7], which practically measures the performance achieved for every Mb of footprint. Tab. 1 shows detection decathlon results. Our approach provides the best tradeoff by being parameter-efficient and yet accurate, as measured by the Score/Footprint ratio. This is achieved by learning patches with a task-adaptive footprint, resulting on average in a 24% footprint gain with respect to WeightTrans.

Model updating. We pre-trained detection models on 10%, 20%, 40%, and 80% of the COCO data. These models achieved 20.9%, 24.2%, 31.4%, and 35.7% mAP@0.5 on the COCO17 validation set, respectively. Then, we applied different patching approaches to update these imprecise models, using 100% of the COCO data. We then compared mAP@0.5 of the patched models, as well as the resulting patch footprint. Tab. 2 shows the results. At 10% training data, we achieve comparable mAP as WeightTrans (32.0% v.s. 32.7%) at a comparable footprint (5.04% v.s. 5.15%). However, when more data is available, the patch footprint generated by our approach is smaller than WeightTrans (2.08% v.s. 5.15%), while accuracy remains comparable (35.8% v.s. 35.9%). Our method can effectively adapt the patch footprint to the amount of new data to be learned by the patch, while maintaining high accuracy.

Table 1: Detection Decathlon. We show footprint, per-dataset mAP, average mAP, score, and score/footprint for each method, best method in **bold**, second-best underline. High score, low footprint, and high score/footprint ratio are good.

Method	Foot-print	Bird	Car	COCO	Dog	Face	Kitti	Pet	RPC	VOC	Avg mAP	Score	Score/ Foot-print
Fine-Tuning	9.00	40.8	90.4	39.7	68.5	35.6	71.9	90.9	99.5	68.3	67.3	2500	278
Tower Patch	0.35	10.0	25.5	31.4	21.2	29.1	49.7	66.1	87.6	70.6	43.5	827	2362
Bn Patch	**0.19**	22.6	71.6	30.2	47.8	26.0	50.7	80.6	92.0	**71.1**	54.7	910	4789
Dw Patch	0.34	22.6	69.2	30.7	43.6	26.4	52.1	80.0	92.6	70.8	54.2	898	2642
Bn+Dw Patch	0.50	27.3	80.9	31.0	52.7	28.0	53.1	83.3	95.7	70.6	58.1	1012	2023
Piggyback	0.30	32.2	87.5	32.4	60.8	28.6	57.4	87.7	97.0	66.0	61.1	1353	4509
WeightTrans	0.46	**36.6**	**90.3**	**37.2**	**66.6**	**30.6**	**65.3**	**90.5**	98.7	70.7	**65.2**	**1987**	4319
Ours	0.35	35.8	89.8	36.6	63.3	30.1	64.0	90.3	**98.9**	70.6	64.4	1858	**5310**

Table 2: Model updating. Only one footprint number is shown for the baseline methods since they can only generate a constant-sized model patch.

Method	Footprint (%)				mAP (%)			
	10%	20%	40%	80%	10%	20%	40%	80%
Fine-Tuning	100.0				37.6	37.6	38.2	38.5
Tower Patch	3.85				24.5	26.7	32.4	35.8
Bn Patch	**2.08**				26.2	28.1	33.0	35.8
Dw Patch	3.76				25.9	27.8	33.1	**36.0**
Bn+Dw Patch	5.59				26.8	28.7	33.4	35.9
Piggyback	3.32				26.4	28.3	32.0	35.3
WeightTrans	5.15				**32.7**	**32.4**	**34.8**	35.9
Ours	5.04	4.98	4.21	**2.08**	32.0	31.4	34.2	35.8

References

1. Guo, Y., Li, Y., Feris, R.S., Wang, L., Rosing, T.S.: Depthwise convolution is all you need for learning multiple visual domains. In: AAAI (2019)
2. Guo, Y., Shi, H., Kumar, A., Grauman, K., Rosing, T., Feris, R.: Spottune: Transfer learning through adaptive fine-tuning. In: CVPR (2019)
3. Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., Murphy, K.: Speed/accuracy trade-offs for modern convolutional object detectors. In: CVPR (2017)
4. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y.: Quantized neural networks: Training neural networks with low precision weights and activations. JMLR (2017)
5. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., Kalenichenko, D.: Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: CVPR (2018)
6. Mallya, A., Davis, D., Lazebnik, S.: Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In: ECCV (2018)
7. Mancini, M., Ricci, E., Caputo, B., Rota Bulò, S.: Adding new tasks to a single network with weight transformations using binary masks. In: ECCV Workshop on Transferring and Adapting Source Knowledge in Computer Vision (2018)
8. Mudrakarta, P.K., Sandler, M., Zhmoginov, A., Howard, A.: K for the price of 1. parameter efficient multi-task and transfer learning. In: ICLR (2019)
9. Rebuffi, S.A., Bilen, H., Vedaldi, A.: Learning multiple visual domains with residual adapters. In: NeurIPS (2017)