Alleviating Representational Shift for Continual Fine-tuning

Appendix

A. Proofs

A.1. Proof of Proposition 1

Proof. Without loss of generality, we suppose the Conv layer does not have a bias parameter. We denote its kernel size as K, and its weight parameter as $\boldsymbol{W} \in \mathbb{R}^{C\times C'\times K\times K}$. When padding size is $K-1$, the input $\boldsymbol{a} \in \mathbb{R}^{B\times C\times H\times W}$ is zero-padded to $\boldsymbol{a}' \in \mathbb{R}^{B\times C\times (H+2K-2)\times (W+2K-2)}$. The output of this Conv layer is $f_{Conv}(\boldsymbol{a}) \in \mathbb{R}^{B\times C'\times H'\times W'}$. We have:

$$
\begin{aligned}
&AvgPool(f_{Conv}(\boldsymbol{a}))\\
=&\frac{1}{BH'W'}\sum_{b=1}^{B}\sum_{h=1}^{H'}\sum_{w=1}^{W'} f_{Conv}(\boldsymbol{a})_{b,:,h,w}\\
=&\frac{1}{BH'W'}\sum_{b=1}^{B}\sum_{h=1}^{H+K-1}\sum_{w=1}^{W+K-1}\\
&\qquad\sum_{c=1}^{C}\sum_{i=1}^{K}\sum_{j=1}^{K}\boldsymbol{W}_{c,:,i,j}\boldsymbol{a}'_{b,c,h+i-1,w+j-1}\\
=&\frac{1}{BH'W'}\sum_{c=1}^{C}\sum_{i=1}^{K}\sum_{j=1}^{K}\left(\boldsymbol{W}_{c,:,i,j}\right.\\
&\qquad\left.\sum_{b=1}^{B}\sum_{h=1}^{H+K-1}\sum_{w=1}^{W+K-1}\boldsymbol{a}'_{b,c,h+i-1,w+j-1}\right)\\
=&\frac{1}{BH'W'}\sum_{c=1}^{C}\sum_{i=1}^{K}\sum_{j=1}^{K}\left(\boldsymbol{W}_{c,:,i,j}\sum_{b=1}^{B}\sum_{h=1}^{H}\sum_{w=1}^{W}\boldsymbol{a}_{b,c,h,w}\right)\\
=&\frac{1}{BH'W'}\sum_{c=1}^{C}\sum_{i=1}^{K}\sum_{j=1}^{K}\left(\boldsymbol{W}_{c,:,i,j}\right.\\
&\qquad\left.\sum_{b=1}^{B}\sum_{h=1}^{H}\sum_{w=1}^{W}AvgPool_{DP}(\boldsymbol{a})_{b,c,h,w}\right)\\
=&\frac{1}{BH'W'}\sum_{c=1}^{C}\sum_{i=1}^{K}\sum_{j=1}^{K}\left(\boldsymbol{W}_{c,:,i,j}\right.\\
&\qquad\left.\sum_{b=1}^{B}\sum_{h=1}^{H+K-1}\sum_{w=1}^{W+K-1}AvgPool_{DP}(\boldsymbol{a})'_{b,c,h+i-1,w+j-1}\right)\\
=&\frac{1}{BH'W'}\sum_{b=1}^{B}\sum_{h=1}^{H'}\sum_{w=1}^{W'}f_{Conv}(AvgPool_{DP}(\boldsymbol{a}))_{b,:,h,w}\\
=&AvgPool(f_{Conv}(AvgPool_{DP}(\boldsymbol{a})))
\end{aligned}
$$

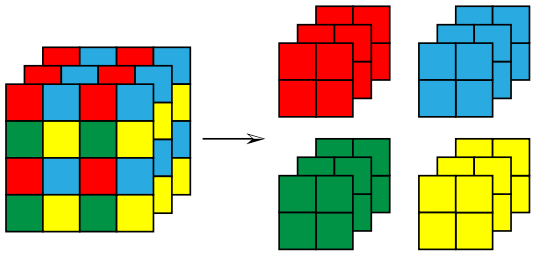

Figure 1

If $stride \neq 1$, we suppose $stride = 2$ as an instance. We can transform the convolution with $stride = 2$ into 4 convolutions with $stride = 1$, by breaking down both $\boldsymbol{a}$ and $\boldsymbol{W}$ as in Fig. 1. Then the proposition still holds for each of the 4 convolutions. Meanwhile, we need to store 4 pre-convolution means with respect to the 4 parts of $\boldsymbol{a}$. Similarly, if $stride = m$, we just need to transform the convolution into m^2 convolutions with $stride = 1$, and store m^2 pre-convolution means.

As for the cases where $padding \neq K-1$, we hypothesize the pixels at the edges of images are less informative background, which can be ignored securely. So the proposition still holds with negligible error.

A.2. Proof of Proposition 2

Proof. This proposition is an immediate deduction of Theorem 3.2 in [2] [1] by simply regarding inputs of task

[1] We notice that the theorem IDs are different between the openreview camera-ready version and arxiv version. We are in line with the former.

Dataset	CIFAR100	CUB200	Caltech101	Flowers102
#classes	100	200	101	102
#tasks	20	10	10	10
#classes in each task	5	20	10	10
max #images of a class in training set	500	30	640	206
min #images of a class in training set	500	29	25	32
#images in training set	50000	5994	6941	6551
#images in test set	10000	5794	1736	1638

Table 1. Main statistics of datasets.

$\mathcal{T}_t$ as in-distribution data, and inputs of task $\mathcal{T}_{t'}$ as out-of-distribution data. A slight difference is that we assume the "out-of-distribution data" are in the deterministic worst case, so we do not have a second moment term $\sqrt{\sigma_{min}(\Sigma)}$ which is derived from the stochasticity of the out-of-distribution data in Lemma A.7 of [2].

As for multi-head setting, we have

$$\begin{aligned}
&\sqrt{\mathcal{L}_{t'}(\boldsymbol{B}_t^*, \boldsymbol{v}_{t'}^*)} = \sqrt{\max_{\|\boldsymbol{x}\|\le 1} \left(\boldsymbol{v}_{t'}^{*\top}\boldsymbol{B}_t^*\boldsymbol{x} - \boldsymbol{v}_{t'}^{*\top}\boldsymbol{B}_{t'}^*\boldsymbol{x}\right)^2} \\
&= \left\|\boldsymbol{v}_{t'}^{*\top}\boldsymbol{B}_t^* - \boldsymbol{v}_{t'}^{*\top}\boldsymbol{B}_{t'}^*\right\|_2 \\
&\ge \left\|\boldsymbol{v}_t^{*\top}\boldsymbol{B}_t^* - \boldsymbol{v}_{t'}^{*\top}\boldsymbol{B}_{t'}^*\right\|_2 - \left\|\boldsymbol{v}_t^{*\top}\boldsymbol{B}_t^* - \boldsymbol{v}_{t'}^{*\top}\boldsymbol{B}_t^*\right\|_2 \\
&= \sqrt{\mathcal{L}_{t'}(\boldsymbol{B}_t^*, \boldsymbol{v}_t^*)} - \epsilon_{mh}
\end{aligned}$$

in which $\epsilon_{mh} = \left\|\boldsymbol{v}_t^{*\top}\boldsymbol{B}_t^* - \boldsymbol{v}_{t'}^{*\top}\boldsymbol{B}_t^*\right\|_2$ is the divergence between the two sub-networks corresponding to the two heads. We can regard ϵ_{mh} as a relaxation provided by multi-head setting.

A.3. Proof of Proposition 3

Proof. For all task $\mathcal{T}_{t'}$ with $t' \le t$, we firstly assume that $\boldsymbol{B}_{t'-1}^* = \boldsymbol{B}_0^*$. Then there exists $\boldsymbol{v}_0$ such that $\mathcal{L}_{t'}(\boldsymbol{B}_{t'-1}^*, \boldsymbol{v}_0) = \mathcal{L}_{t'}(\boldsymbol{B}_{t'}^*, \boldsymbol{v}_{t'}^*)$. By the proof of Proposition A.21 in [2] we have $\boldsymbol{v}_{t'}^{lp} = \boldsymbol{v}_0$. Since overparametrized model can achieve zero loss, we have $\mathcal{L}_{t'}\left(\boldsymbol{B}_{t'-1}^*, \boldsymbol{v}_{t'}^{lp}\right) = \mathcal{L}_{t'}(\boldsymbol{B}_0^*, \boldsymbol{v}_0) = 0$. So the gradient is also zero, and thus the feature extractor does not change at all, which means $\boldsymbol{B}_{t'}^* = \boldsymbol{B}_0^*$ as well.

When $t' = 1$, the assumption $\boldsymbol{B}_{t'-1}^* = \boldsymbol{B}_0^*$ is naturally satisfied. Then inductively, for all task $\mathcal{T}_{t'}$ with $t' \le t$, we have $\boldsymbol{B}_{t'}^* = \boldsymbol{B}_0^*$, which leads to $\mathcal{L}_{t'}(\boldsymbol{B}_t^*, \boldsymbol{v}_{t'}^*) = \mathcal{L}_{t'}(\boldsymbol{B}_{t'}^*, \boldsymbol{v}_{t'}^*)$ as we desire.

B. Experiment Details

B.1. Dataset Split

For CIFAR100 and CUB200, we use the official train/test split. For Caltech101 and Flowers102, we randomly choose 80% images as training set, and the others as test set. During hyperparameter search, we further randomly choose 10% images in training set for validation. The details of dataset statistics are in Tab. 1.

B.2. Implementation

All experiments are implemented using *Pytorch* 1.10 with CUDA 10.2 on NVIDIA 1080Ti GPU. The datasets and dataloaders are built via *Avalanche* [3]. We use ResNet18 implemented by *timm* [6], and its pre-trained parameters are downloaded from *torchvision*. The baselines are reproduced on the basis of the official codebases of [1,4,5], in which we only modify the models and dataset interfaces.

B.3. Data Pre-processing

For each image, we pre-process it as follows (Pytorch style):

```
Compose(
    Resize(size=256),
    CenterCrop(size=(224, 224)),
    ToTensor(),
    Normalize(
        mean=[0.4850, 0.4560, 0.4060],
        std=[0.2290, 0.2240, 0.2250]
    )
)
```

There is no extra data augmentation applied.

B.4. Hyperparameter Search

For each baseline, we extensively search the best hyperparameter, which is shown in Tab. 2. We train the model for a fixed number of epochs for each method, which depends on the complexity of the dataset. We randomly choose 10% images in training set for validation, exclude them from training, and use them to determine the best hyperparameters. After the hyperparameter search, we retrain the model with the whole training set and report the results on test set. Note that our ConFiT does not need hyperparameter search.

Method	Hyperparameter	Search Range	CIFAR100	CUB200	Caltech101	Flowers102
EWC	λ	{3e2, 1e3, 3e3, 1e4, 3e4, 1e5, 3e5}	1e3	3e3	3e4	1e5
SI	c	{1, 2, 5, 10, 20, 50}	2	5	10	20
RWalk	λ	{5, 10, 20, 50, 100, 200}	20	20	50	100
MAS	λ	{1, 3, 10, 30, 100, 300}	10	30	30	30
CPR	β	{0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5}	0.2	0.02	0.01	0.1
AFEC	λ_e	{0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000}	0.01	100	1000	10

Table 2. Results of hyperparameter search.

C. Difference between IRS and *Internal Covariate Shift*

We notice that there is another concept called *Internal Covariate Shift* (ICS) about BN. To avoid confusion, we here clarify the difference between IRS and ICS.

Firstly, we would emphasize that ICS and IRS are totally different concepts. ICS describes the instability of intermediate representations' distribution during the training stage, i.e., the inputs of intermediate layers are unstable *in training*, which impedes the learning of networks. One of the motivations behind BN is to address ICS, so as to make the network easier to train.

Whereas IRS is defined in the scenario of continual learning, which means the intermediate representations of *data of previous tasks* were shifted, because the network fitted the data of the newest task. The shifted representation will disrupt the function of BN *in testing*, since the running moments will no longer be representative of the true moments of intermediate representations.

Overall, BN can solve ICS, but will suffer from IRS in continual learning, which is what we have attempted to address.

References

[1] Sungmin Cha, Hsiang Hsu, Taebaek Hwang, Flávio P. Calmon, and Taesup Moon. CPR: classifier-projection regularization for continual learning. In *ICLR*, 2021. 2

[2] Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *ICLR*, 2022. 1, 2

[3] Vincenzo Lomonaco, Lorenzo Pellegrini, Andrea Cossu, Antonio Carta, Gabriele Graffieti, Tyler L. Hayes, Matthias De Lange, Marc Masana, Jary Pomponi, Gido M. van de Ven, Martin Mundt, Qi She, Keiland Cooper, Jeremy Forest, Eden Belouadah, Simone Calderara, German Ignacio Parisi, Fabio Cuzzolin, Andreas S. Tolias, Simone Scardapane, Luca Antiga, Subutai Ahmad, Adrian Popescu, Christopher Kanan, Joost van de Weijer, Tinne Tuytelaars, Davide Bacciu, and Davide Maltoni. Avalanche: An end-to-end library for continual learning. In *CVPR Workshops*, 2021. 2

[4] Pravendra Singh, Vinay Kumar Verma, Pratik Mazumder, Lawrence Carin, and Piyush Rai. Calibrating cnns for lifelong learning. In *NeurIPS*, 2020. 2

[5] Liyuan Wang, Mingtian Zhang, Zhongfan Jia, Qian Li, Chenglong Bao, Kaisheng Ma, Jun Zhu, and Yi Zhong. AFEC: Active forgetting of negative transfer in continual learning. In *NeurIPS*, 2021. 2

[6] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 2