Supplementary Material for DS-AL: A Dual-Stream Analytic Learning for Exemplar-Free Class-Incremental Learning

Huiping Zhuang[1], Run He[1], Kai Tong[1], Ziqian Zeng[1], Cen Chen[2,3*], Zhiping Lin[4]

[1]Shien-Ming Wu School of Intelligent Engineering, South China University of Technology, China
[2] School of Future Technology, South China University of Technology, China
[3] Pazhou Laboratory, China
[4] School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore
{hpzhuang, zqzeng, chencen}@scut.edu.cn, {wirh, wikaitong}@mail.scut.edu.cn, ezplin@ntu.edu.sg

A: Proof of Theorem 1

Proof. According to (9), at phase $k-1$, we have

$$\hat{\boldsymbol{W}}_{\mathrm{M}}^{(k-1)} = \left(\boldsymbol{X}_{\mathrm{M},0:k-1}^T \boldsymbol{X}_{\mathrm{M},0:k-1} + \gamma \boldsymbol{I}\right)^{-1} \boldsymbol{X}_{\mathrm{M},0:k-1}^T \boldsymbol{Y}_{0:k-1}. \tag{A.1}$$

Hence, at phase k we have

$$\hat{\boldsymbol{W}}_{\mathrm{M}}^{(k)} = \left(\boldsymbol{X}_{\mathrm{M},0:k}^T \boldsymbol{X}_{\mathrm{M},0:k} + \gamma \boldsymbol{I}\right)^{-1} \boldsymbol{X}_{\mathrm{M},0:k}^T \boldsymbol{Y}_{0:k}. \tag{A.2}$$

We have defined the iACM $\boldsymbol{R}_{\mathrm{M},k-1}$ in the paper via

$$\boldsymbol{R}_{\mathrm{M},k-1} = \left(\boldsymbol{X}_{\mathrm{M},0:k-1}^T \boldsymbol{X}_{\mathrm{M},0:k-1} + \gamma \boldsymbol{I}\right)^{-1}. \tag{A.3}$$

To facilitate subsequent calculations, here we also define a cross-correlation matrix $\boldsymbol{Q}_{\mathrm{M},k-1}$, i.e.,

$$\boldsymbol{Q}_{\mathrm{M},k-1} = \boldsymbol{X}_{\mathrm{M},0:k-1}^T \boldsymbol{Y}_{0:k-1}. \tag{A.4}$$

Thus we can rewrite (A.2) as

$$\hat{\boldsymbol{W}}_{\mathrm{M}}^{(k-1)} = \boldsymbol{R}_{\mathrm{M},k-1} \boldsymbol{Q}_{\mathrm{M},k-1}. \tag{A.5}$$

Therefore, at phase k we have

$$\hat{\boldsymbol{W}}_{\mathrm{M}}^{(k)} = \boldsymbol{R}_{\mathrm{M},k} \boldsymbol{Q}_{\mathrm{M},k}. \tag{A.6}$$

From (A.3), we can recursively calculate $\boldsymbol{R}_{\mathrm{M},k}$ from $\boldsymbol{R}_{\mathrm{M},k-1}$, i.e.,

$$\boldsymbol{R}_{\mathrm{M},k} = \left(\boldsymbol{R}_{\mathrm{M},k-1}^{-1} + \boldsymbol{X}_{\mathrm{M},k}^T \boldsymbol{X}_{\mathrm{M},k}\right)^{-1}. \tag{A.7}$$

According to the Woodbury matrix identity, we have

$$(\boldsymbol{A} + \boldsymbol{U}\boldsymbol{C}\boldsymbol{V})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{U}(\boldsymbol{C}^{-1} + \boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{U})^{-1}\boldsymbol{V}\boldsymbol{A}^{-1}.$$

Let $\boldsymbol{A} = \boldsymbol{R}_{\mathrm{M},k-1}^{-1}, \boldsymbol{U} = \boldsymbol{X}_{\mathrm{M},k}^T, \boldsymbol{C} = \boldsymbol{I}, \boldsymbol{V} = \boldsymbol{X}_{\mathrm{M},k}$ in (A.7), we have

$$\boldsymbol{R}_{\mathrm{M},k} = \boldsymbol{R}_{\mathrm{M},k-1} - \boldsymbol{R}_{\mathrm{M},k-1}\boldsymbol{X}_{\mathrm{M},k}^T(\boldsymbol{I} + \boldsymbol{X}_{\mathrm{M},k}\boldsymbol{R}_{\mathrm{M},k-1}\boldsymbol{X}_{\mathrm{M},k}^T)^{-1}\boldsymbol{X}_{\mathrm{M},k}\boldsymbol{R}_{\mathrm{M},k-1}. \tag{A.8}$$

Hence, $\boldsymbol{R}_{\mathrm{M},k}$ can be recursively updated using its last-phase counterpart $\boldsymbol{R}_{\mathrm{M},k-1}$ and data from the current phase (i.e., $\boldsymbol{X}_{\mathrm{M},k}$). This proves the recursive calculation of the iACM in (12).

Next, we derive the recursive formulation of $\hat{\boldsymbol{W}}_{\mathrm{M}}^{(k)}$. To this end, we also recurse the cross-correlation matrix $\boldsymbol{Q}_{\mathrm{M},k}$ at phase k, i.e.,

$$\boldsymbol{Q}_{\mathrm{M},k} = \boldsymbol{X}_{\mathrm{M},0:k}^T \boldsymbol{Y}_{0:k} = \boldsymbol{Q}'_{\mathrm{M},k-1} + \boldsymbol{X}_{\mathrm{M},k}^T \boldsymbol{Y}_{\mathrm{M},k} \tag{A.9}$$

where

$$\boldsymbol{Q}'_{\mathrm{M},k-1} = \begin{bmatrix} \boldsymbol{Q}_{\mathrm{M},k-1} & \boldsymbol{0}_{d_{\mathrm{fe}} \times d_{y_k}} \end{bmatrix}. \tag{A.10}$$

Note that the concatenation in (A.10) is due to fact that $\boldsymbol{Y}_{0:k}$ at phase k contains more data classes (hence more columns) than $\boldsymbol{Y}_{0:k-1}$.

Let $\boldsymbol{K}_k = (\boldsymbol{I} + \boldsymbol{X}_{\mathrm{M},k}\boldsymbol{R}_{\mathrm{M},k-1}\boldsymbol{X}_{\mathrm{M},k}^T)^{-1}$. Since

$$\boldsymbol{I} = \boldsymbol{K}_k \boldsymbol{K}_k^{-1} = \boldsymbol{K}_k(\boldsymbol{I} + \boldsymbol{X}_{\mathrm{M},k}\boldsymbol{R}_{\mathrm{M},k-1}\boldsymbol{X}_{\mathrm{M},k}^T),$$

we have $\boldsymbol{K}_k = \boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{X}_{\mathrm{M},k}\boldsymbol{R}_{\mathrm{M},k-1}\boldsymbol{X}_{\mathrm{M},k}^T$. Therefore,

$$\begin{aligned}
&\boldsymbol{R}_{\mathrm{M},k-1}\boldsymbol{X}_{\mathrm{M},k}^T(\boldsymbol{I} + \boldsymbol{X}_{\mathrm{M},k}\boldsymbol{R}_{\mathrm{M},k-1}\boldsymbol{X}_{\mathrm{M},k}^T)^{-1} \\
&= \boldsymbol{R}_{\mathrm{M},k-1}\boldsymbol{X}_{\mathrm{M},k}^T\boldsymbol{K}_k \\
&= \boldsymbol{R}_{\mathrm{M},k-1}\boldsymbol{X}_{\mathrm{M},k}^T(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{X}_{\mathrm{M},k}\boldsymbol{R}_{\mathrm{M},k-1}\boldsymbol{X}_{\mathrm{M},k}^T) \\
&= (\boldsymbol{R}_{\mathrm{M},k-1} - \boldsymbol{R}_{\mathrm{M},k-1}\boldsymbol{X}_{\mathrm{M},\boldsymbol{k}}^{\boldsymbol{T}}\boldsymbol{K}_k\boldsymbol{X}_{\mathrm{M},k}\boldsymbol{R}_{\mathrm{M},k-1})\boldsymbol{X}_{\mathrm{M},k}^T \\
&= \boldsymbol{R}_{\mathrm{M},k}\boldsymbol{X}_{\mathrm{M},k}^T.
\end{aligned} \tag{A.11}$$

As $\hat{\boldsymbol{W}}_{\mathrm{M}}^{(k-1)\prime} = \begin{bmatrix} \hat{\boldsymbol{W}}_{\mathrm{M}}^{(k-1)} & \boldsymbol{0} \end{bmatrix}$ has expended its dimension similar to what $\boldsymbol{Q}'_{\mathrm{M},k-1}$ does, we have

$$\hat{\boldsymbol{W}}_{\mathrm{M}}^{(k-1)\prime} = \boldsymbol{R}_{\mathrm{M},k-1}\boldsymbol{Q}'_{\mathrm{M},k-1}. \tag{A.12}$$

Hence, $\hat{\boldsymbol{W}}_{\mathrm{M}}^{(k)}$ can be rewritten as

$$\begin{aligned}
\hat{\boldsymbol{W}}_{\mathrm{M}}^{(k)} &= \boldsymbol{R}_{\mathrm{M},k}\boldsymbol{Q}_{\mathrm{M},k} \\
&= \boldsymbol{R}_{\mathrm{M},k}(\boldsymbol{Q}'_{\mathrm{M},k-1} + \boldsymbol{X}_{\mathrm{M},k}^T\boldsymbol{Y}_{\mathrm{M},k}) \\
&= \boldsymbol{R}_{\mathrm{M},k}\boldsymbol{Q}'_{\mathrm{M},k-1} + \boldsymbol{R}_{\mathrm{M},k}\boldsymbol{X}_{\mathrm{M},k}^T\boldsymbol{Y}_{\mathrm{M},k}.
\end{aligned} \tag{A.13}$$

By substituting (A.8) into $\boldsymbol{R}_{\mathrm{M},k}\boldsymbol{Q}'_{\mathrm{M},k-1}$, we have

$$\begin{aligned}
&\boldsymbol{R}_{\mathrm{M},k}\boldsymbol{Q}'_{\mathrm{M},k-1} = \boldsymbol{R}_{\mathrm{M},k-1}\boldsymbol{Q}'_{\mathrm{M},k-1} \\
&-\boldsymbol{R}_{\mathrm{M},k-1}\boldsymbol{X}_{\mathrm{M},k}^T(\boldsymbol{I} + \boldsymbol{X}_{\mathrm{M},k}\boldsymbol{R}_{\mathrm{M},k-1}\boldsymbol{X}_{\mathrm{M},k}^T)^{-1}\boldsymbol{X}_{\mathrm{M},k}\boldsymbol{R}_{\mathrm{M},k-1}\boldsymbol{Q}'_{\mathrm{M},k-1} \\
&= \hat{\boldsymbol{W}}_{\mathrm{M}}^{(k-1)\prime} - \boldsymbol{R}_{\mathrm{M},k-1}\boldsymbol{X}_{\mathrm{M},k}^T(\boldsymbol{I} + \boldsymbol{X}_{\mathrm{M},k}\boldsymbol{R}_{\mathrm{M},k-1}\boldsymbol{X}_{\mathrm{M},k}^T)^{-1}\boldsymbol{X}_{\mathrm{M},k}\hat{\boldsymbol{W}}_{\mathrm{M}}^{(k-1)\prime}
\end{aligned} \tag{A.14}$$

*Corresponding author

Algorithm 1: DS-AL training

Input: The training data $\mathcal{D}_0^{\text{train}}, \mathcal{D}_1^{\text{train}}, \dots, \mathcal{D}_K^{\text{train}}$
Output: Weight matrix at last phase of *main stream* $\hat{\boldsymbol{W}}_{\text{M}}^{(K)}$ and *compensation stream* $\hat{\boldsymbol{W}}_{\text{C}}^{(K)}$

```
begin
    // BP-based Training
    Do conventional training with BP on base dataset D_0^train to obtain the weight of the backbone W_CNN.
    Freeze the backbone.
    for phase k ← 0 to K do
        if k = 0 then
            // AL-based Re-training
            // - The main stream
            Obtain the main stream activation X_{M,0} with (3).
            Calculate inverted auto-correlation matrix (iACM) R_{M,0} of the main stream with (10).
            Obtain the main stream weight matrix Ŵ_M^0 with (5).
            // - The compensation stream
            X_{C,0} with (14).
            Use the obtained main stream weight Ŵ_M^0 and the label Y_0^train to calculate the residue Ỹ_0 with (13).
            Obtain iACM of the compensation stream R_{C,0} ← (X_{C,0}^T X_{C,0} + γI)^{-1}.
            Obtain compensation stream weight matrix Ŵ_C^0 ← (X_{C,0}^T X_{C,0} + γI)^{-1} X_{C,0}^T Ỹ_0.
        else
            // AL-based CIL procedures
            // - The main stream
            Obtain main stream activation X_{M,k} with (7).
            Update iACM of the main stream R_{M,k} with (12).
            Update main stream weight matrix Ŵ_M^(k) with (11).
            // - The compensation stream
            Obtain the compensation stream activation X_{C,k} with (14).
            Use the updated main stream weight Ŵ_M^(k) and the label Y_k^train to calculate the residue Ỹ_k with (13).
            Use the PLC process to obtain cleansed residue label {Ỹ_k}_PLC with (15).
            Update iACM of the compensation stream R_{C,k} with (18).
            Update weight matrix Ŵ_C^(k) with (17).
        end
    end
end
```

According to (A.11), (A.14) can be rewritten as

$$\boldsymbol{R}_{\text{M},k}\boldsymbol{Q}'_{\text{M},k-1} = \hat{\boldsymbol{W}}_{\text{M}}^{(k-1)\prime} - \boldsymbol{R}_{\text{M},k}\boldsymbol{X}_{\text{M},k}^T\boldsymbol{X}_{\text{M},k}\hat{\boldsymbol{W}}_{\text{M}}^{(k-1)\prime}. \tag{A.15}$$

By inserting (A.15) into (A.13), we have

$$\begin{aligned}\hat{\boldsymbol{W}}_{\text{M}}^{(k)} &= \hat{\boldsymbol{W}}_{\text{M}}^{(k-1)\prime} - \boldsymbol{R}_{\text{M},k}\boldsymbol{X}_{\text{M},k}^T\boldsymbol{X}_{\text{M},k}\hat{\boldsymbol{W}}_{\text{M}}^{(k-1)\prime} + \boldsymbol{R}_{\text{M},k}\boldsymbol{X}_{\text{M},k}^T\boldsymbol{Y}_{\text{M},k} \\ &= \hat{\boldsymbol{W}}_{\text{M}}^{(k-1)\prime} + \boldsymbol{R}_{\text{M},k}\boldsymbol{X}_{\text{M},k}^T(\boldsymbol{Y}_k - \boldsymbol{X}_{\text{M},k}\hat{\boldsymbol{W}}_{\text{M}}^{(k-1)\prime})\end{aligned}$$

which completes the proof. □

B: Pseudo-code of DS-AL

The algorithm of DS-AL contains two parts, i.e., training and inferencing. The pseudo-codes of training part are shown in Algorithm 1. The training part of DS-AL contains mainly three parts, including BP-based training, AL-based re-training and AL-based CIL procedures. The details of inference are shown in Algorithm 2.

C: Training Details

For conventional BP training in the BT agenda, we train the network using SGD for 160 (90) epochs for ResNet-32 on CIFAR-100 (ResNet-18 on ImageNet-100 and ImageNet-Full). The learning rate starts at 0.1 and it is divided by 10 at epoch 80 (30) and 120 (60). We adopt a momentum of 0.9 and weight decay of 5×10^{-4} (1×10^{-4}) with a batch size of 128. For ResNet-18 used on CIFAR-100, we follow the setting in (Petit et al. 2023). The input data are augmented with random cropping, random horizontal flip and normalizing. For fair comparison, this base training setting is identical to that of many CIL methods (e.g., (Hou et al. 2019; Liu et al. 2020)). For the re-training and incremental learning steps, no data augmentation is adopted, and the training ends within only one epoch. The results for the DS-AL are measured by the average of 3 runs on an RTX 3090 GPU workstation. Note that in DS-AL, the CNN is only trained during the BP base training phase. After that, the parame-

Algorithm 2: DS-AL inference at phase k

Input: Backbone $\boldsymbol{W}_{\text{CNN}}$, test data $\mathcal{D}_k^{\text{test}}$, *main stream* weight $\hat{\boldsymbol{W}}_{\text{M}}^{(k)}$ and *compensation stream* weight $\hat{\boldsymbol{W}}_{\text{C}}^{(k)}$

Output: Compensated test inference $\hat{\boldsymbol{Y}}_k^{(\text{all})}$

begin

Obtain the *main stream activation* $\boldsymbol{X}_{\text{M},k} \leftarrow \sigma_{\text{M}}(B(f_{\text{flat}}(f_{\text{CNN}}(\boldsymbol{X}_k^{\text{test}}, \boldsymbol{W}_{\text{CNN}}))))$

Obtain the *compensation stream activation* $\boldsymbol{X}_{\text{C},k} \leftarrow \sigma_{\text{C}}(B(f_{\text{flat}}(f_{\text{CNN}}(\boldsymbol{X}_k^{\text{test}}, \boldsymbol{W}_{\text{CNN}}))))$.

Calculate the prediction combining both streams $\hat{\boldsymbol{Y}}_k^{(\text{all})}$ with (19).

end

ters of the CNN backbone are fixed during the incremental phases (i.e, phase #1 to #K).

References

Hou, S.; Pan, X.; Loy, C. C.; Wang, Z.; and Lin, D. 2019. Learning a Unified Classifier Incrementally via Rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Liu, Y.; Su, Y.; Liu, A.-A.; Schiele, B.; and Sun, Q. 2020. Mnemonics Training: Multi-Class Incremental Learning Without Forgetting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Petit, G.; Popescu, A.; Schindler, H.; Picard, D.; and Delezoide, B. 2023. FeTrIL: Feature Translation for Exemplar-Free Class-Incremental Learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 3911–3920.