Supplementary Material: Memory-Efficient Incremental Learning Through Feature Adaptation

Ahmet Iscen\textsuperscript{1}, Jeffrey Zhang\textsuperscript{2}, Svetlana Lazebnik\textsuperscript{2}, and Cordelia Schmid\textsuperscript{1}

\textsuperscript{1}Google Research \hspace{1cm} \textsuperscript{2}University of Illinois at Urbana-Champaign

A Algorithm

An overview of our framework is described in Algorithm 1.

\begin{algorithm}
\caption{Memory-efficient incremental learning}
\begin{algorithmic}[1]
\Procedure{Algorithm}{Training examples $X$, labels $Y$}
\State Given $T$ tasks
\State *** Train first task ***
\State $X^1, Y^1 \in X,Y$
\State $\theta, W = \text{Optimize}(L_{CE}(f_{\theta, W}(X^1), Y^1))$
\State $h^1_{\theta} = h_{\theta}$
\State $M^1 = h^1_{\theta}(X^1)$
\State $M^1 = \text{Herding}(M^1)$
\For{$t \in [2, \ldots, T]$}
\State *** Train incremental tasks ***
\State $X^t, Y^t \in X,Y$
\State $\theta, W = \text{Optimize}(L(f_{\theta, W}(X^t), Y^t))$
\State $h^t_{\theta} = h_{\theta}$
\State $\phi_{\psi} = \text{Feature Adaptation}(h^t_{\psi}, h^{t-1}_{\psi}, X^t, Y^t)$
\State $M^t = h^t_{\psi}(X^t)$
\EndFor
\State $\tilde{W} = \text{Train Classifier}(M^t, Y_1, \ldots, Y^t)$
\EndProcedure
\Procedure{FeatureAdaptation}{$h^{old}_{\psi}, h^{new}_{\psi}, X, Y$}
\State *** Returns transformation function ***
\State $\mathcal{V} = h^{old}_{\psi}(X)$
\State $\mathcal{V} = h^{new}_{\psi}(X)$
\State $\psi = \text{Optimize}(L_{FA}(\mathcal{V}, \mathcal{V}, Y))$
\State \Return $\phi_{\psi}$
\EndProcedure
\end{algorithmic}
\end{algorithm}

B Feature Adaptation Quality

We evaluate the quality of our feature adaptation process by measuring the average similarity between the adapted features and their ground-truth value. We compute the feature adaptation quality as explained in Section 5.4. However, we compute two distinct measurements this time. $\omega_{t-1}$ measures the average feature adaptation quality of features extracted in the previous task ($i.e.$ $y \in C^{t-1}$). This measurement does not
Fig. 1. Feature adaptation quality on CIFAR-100 for $M = 10$. $P$ refers to the number of images preserved in the memory. Solid and dashed lines correspond to vectors from previous ($\omega_{t-1}$) and first ($\omega_1$) task respectively.

track the quality over time, but shows feature adaptation quality between two tasks. $\omega_t$ measures the feature adaptation quality of features originally extracted in the first task (i.e. $y \in C^1$), showing how much the adapted features can diverge from their optimal representation due to accumulated error. Adaptation quality is computed for all $L = 500$ feature descriptors per class.

Figure ?? shows the adaptation quality $\omega_{t-1}$ and $\omega_1$ on CIFAR-100 with $M = 10$. We report the quality measures for when $P$ number images are also preserved in the memory. We observe that $P = 0$ achieves $\omega_{t-1}$ greater than 0.9 in all tasks. It increases as more classes are seen, most likely due to the network becoming more stable. After 10 tasks, $\omega_1$ is still close to 0.8, indicating our feature adaptation is still relatively successful after training 9 subsequent tasks with no preserved images. Adaptation quality improves as $P$ increases, showing that preserving images also helps with learning a better feature adaptation.

**C Balanced Feature Classifier Training**

Wu et al. [53] illustrated that training a classifier on fewer examples for previous classes introduces a bias towards new classes. To verify the robustness of our method under this setting, we investigate the effect of training our feature classifier on class-balanced and class-imbalanced training sets.

In our main experiments, we train our feature classifier $g_{\hat{W}}$ with a balanced number of examples per class. We repeat our ImageNet-100 experiment in Table 1 without balancing the classifier training samples. In the unbalanced setting, the old classes contain 250 features per class in memory, whereas the new classes each contain $\sim 1300$ feature vectors. In the balanced setting, all classes contain 250 feature vectors. On ImageNet-100, we achieve 0.893 accuracy with class-imbalanced training, compared to 0.913 accuracy with class-balanced training (reported in Table 1). This shows that even though more training data is utilized in the class-imbalanced setting, the imbalanced class bias leads to a drop in overall performance.

Our method addresses this problem by building a large balanced feature set for training. Our feature adaptation method not only reduces the memory footprint compared to [53] (see Table 1), but also allows substantially more stored data points from old classes (250 features per class compared to 20 images per class for [53]). This may explain some improvement in our results over previous methods. Lastly, the significant increase in number of stored examples provides flexibility to remove examples to keep classes balanced.