Parameter-efficient Continual Learning for Enhancing Plasticity without Forgetting under Limited Model Capacity

Supplementary Material

1. Detailed Dataset Splits

In this work, we construct six task sequences using eight widely used image classification datasets. Each dataset is divided into a series of balanced tasks, and images undergo dataset-specific preprocessing. Details are as follows:

- **MNIST, FashionMNIST, KMNIST:** These datasets each contain 10 grayscale categories with image resolution 28×28. To enable compatibility with RGB networks, grayscale images are converted into 3-channel format by duplicating the single channel across all three channels. During training, data augmentation includes random rotation within $\pm 10^{\circ}$. All pixel values are normalized using dataset-specific mean and standard deviation, replicated across all channels.
- **TinyImageNet:** This dataset consists of 200 RGB categories at 64×64 resolution. Training images are augmented with random crops and horizontal flips, while testing images are center-cropped. Normalization is applied using per-channel mean and standard deviation computed from the dataset.
- **QMNIST:** The QMNIST dataset contains 10 grayscale digit classes at 28×28 resolution. Preprocessing and augmentation are identical to MNIST: grayscale to 3-channel conversion, random rotation, and normalization, with no resizing.
- **SVHN:** The Street View House Numbers dataset provides 10 RGB digit categories in 32×32 resolution. During training, images are padded with 4 pixels on each side and randomly cropped back to 32×32, followed by normalization using channel-wise dataset statistics.
- **CIFAR-10:** This dataset includes 10 RGB object categories with 32×32 resolution. Training images are augmented via random cropping with 4-pixel padding and horizontal flipping. Normalization is applied using CIFAR-10-specific mean and standard deviation per channel.
- **Food-101:** We select the first 100 alphabetically sorted categories from the original Food-101 dataset. The original high-resolution images are resized to 72×72. For training, random crops of 64×64 are taken after resizing; for testing, center crops are used instead. Training data also undergo horizontal flipping. Finally, images are normalized to the $[-1, 1]$ using per-channel transformation.

To maintain consistency across tasks and facilitate implementation, all class labels are remapped such that task-specific indices start from 0. This design choice ensures that each task operates in a standardized label space, which in turn supports meaningful performance comparisons under continual learning settings.

2. State of Reinforcement Learning

In GRAPA, the pruning rate for each task t is determined by a reinforcement learning agent, which observes a five-dimensional state vector at each step k. This state vector is defined as

$$s_t^k = [\rho_t^{k-1}, acc_t^{k-1}, c_t,\ \kappa_t,\ \varsigma_t],$$

where ρ_t^{k-1} denotes the pruning rate of the last step, acc_t^{k-1} is the previous task accuracy. The step-wise states are normalized to $[0, 1]$. c is the number of classes normalized by a constant. κ represents input complexity, and σ denotes class separability. Both κ and σ are computed using features extracted by a frozen convolutional feature extractor. This lightweight backbone is shared across all tasks and remains fixed throughout training. It incurs minimal computational overhead and is used solely to estimate task-level input complexity and class separability. The extracted features serve as auxiliary information for state construction in the reinforcement learning controller and do not interfere with the main model training process.

Input complexity. The input complexity κ is designed to measure the visual richness of a task. For each image, we compute its spatial standard deviation and channel-wise standard deviation, denoted as σ_s and σ_c, respectively. These capture texture variation and color contrast. The two terms are linearly combined as:

$$\kappa = \alpha \cdot \sigma_s + \beta \cdot \sigma_c,$$

where $\alpha = 1.0$ and $\beta = 2.0$. All images are converted to RGB (if necessary), and the final complexity score is the average κ across multiple training samples.

Class separability. The separability score σ characterizes how distinguishable different classes are in a given task. We first extract a fixed number of embeddings per class using the lightweight convolutional network. For each class, we compute the average intra-class distance (denoted as d_{intra}), i.e., the mean distance between embeddings and their class centroid. We also compute the average inter-class distance

(denoted as d_{inter}), i.e., the mean distance between class centroids. The final separability is then computed as:

$$\varsigma = \frac{d_{\text{inter}}}{d_{\text{intra}} + \varepsilon},$$

where ε is a small constant to avoid division by zero. A higher value of ς indicates better separation between classes.

In GRAPA, static task-descriptive states—including the number of classes, input complexity, and class separability—are introduced to stabilize the reinforcement learning agent's exploration process. Since the pruning rate and accuracy are dynamic and can fluctuate significantly during each task, relying solely on these changing signals can lead to inefficient learning and slow convergence. The static states serve as task-level priors, offering contextual information that helps the agent adapt its policy more efficiently to the characteristics of the current task. Meanwhile, the dynamic states provide essential feedback for policy optimization, guiding the agent to adjust the pruning rate for improved performance.