A Reproduction of "Summarizing Stream Data for Memory-Constrained Online Continual Learning"

Nikola Vasić[1], Nevena Denić[2]

Faculty of Science and Mathematics, University of Niš

1 Introduction

Continual learning aims to enable models to learn sequentially from evolving data streams. Commonly, data is split into distinct *tasks* through time, where each task might represent a specific dataset, problem domain, etc. A particularly challenging setting is *online class-incremental continual learning*, in which each task contains a new set of classes and no task identifiers are provided to the model. The data also comes from an online stream - each sample, unless saved, is processed only once.

In online class-incremental continual setting, **replay methods** have generally demonstrated the best performance for mitigating *catastrophic forgetting*. These methods maintain a small subset of past data in a memory buffer (which is constantly updated using typically reservoir sampling) for replay. At time t, the model is trained on incoming batch from the stream $\mathcal{B}_t$ concatenated with a random sample $\mathcal{M}_t$ from the memory buffer $\mathcal{M}$.

2 Method

One common problem in this setting is that the standard Softmax classifier needs architectural changes when new classes arrive and has a task-recency bias. This is addressed by Mai et al., 2021 [2] by discarding the Softmax classifier and employing the **Nearest Class Mean Classifier** along with supervised contrastive loss. At the end of each task, the model is evaluated using the NCM classifier by creating the class means using embeddings of data currently present in memory buffer.

Under circumstances with limited storage space, the informativeness of the memory becomes crucial for effective replay. This is addressed by Gu et al., 2024 [1] by constructing **SSD** - Summarizing Stream Data. SSD enhances standard replay by supplementing original samples in a dynamic memory buffer with "summarized" samples - samples which will be optimized using batches from the data stream using dataset distillation, to condense as much information from the stream into them. Using SSD involves:

- **Dynamic Memory:** Utilizing reservoir sampling for original samples ($\mathcal{M}_O$) and dedicated slots for summarized samples ($\mathcal{M}_S$), which are initialized and updated per class. Random sampling for the main model training is done from $\mathcal{M} = \mathcal{M}_S \cup \mathcal{M}_O$.
- **Summarizing Model:** Adding a smaller *summarizing model* θ with a standard softmax classifier which will be consistently trained on B_t concatenated with a sample from $\mathcal{M}_O$ (*Past-assisted Summarizing*) using $\mathcal{L}_t$ - standard cross-entropy loss. $\mathcal{F}(\theta, \mathcal{X})$ is defined as the activation of the penultimate layer of θ, on input $\mathcal{X}$.
- **Data Summarization:** Each τ batches from the stream, optimizing $\mathcal{M}_S$ using the gradients (*gradient matching loss* - $\mathcal{L}_g$) and representations (*relationship matching loss* - $\mathcal{L}_r$) of θ. $\mathcal{L}_g$ is employed for $\mathcal{M}_S$ to match the gradients of real streaming data batches (B_t), while $\mathcal{L}_r$ is employed for ensuring consistency between current samples and past summarized samples. The summarization loss is $\mathcal{L}_s = \mathcal{L}_g + \gamma \mathcal{L}_r$.

We define $\mathcal{M}_{ps}$ as a sample from a subset of $\mathcal{M}_s$ containing only summarized samples from previous tasks. We define $\mathcal{B}_c$ as all samples from $\mathcal{B}_t$ for class c, and $\mathcal{M}_c$ as all samples from

[1]Worked on the training loop, memory buffers, gradient matching loss and relationship matching loss.

[2]Worked on the training loop, dataset handling, summarizing model and the full SCR+NCM training part.

$\mathcal{M}_s$ of class c. $\mathbf{D}$ can be any distance metric, such as Euclidean distance. $\mathcal{L}_g$ and $\mathcal{L}_r$ are defined as:

$$\mathcal{L}_g = \mathbf{D}\left(\nabla_\theta \mathcal{L}'_t(\theta; \mathcal{M}_c), \nabla_\theta \mathcal{L}'_t(\theta; \mathcal{B}_c)\right), \qquad \begin{aligned} \mathcal{L}_r &= \mathbf{D}\left(\rho(\mathcal{M}_c, \mathcal{M}_{ps}, \theta), \rho(\mathcal{B}_c, \mathcal{M}_{ps}, \theta)\right), \\ \rho(\mathcal{X}, \mathcal{Y}, \theta) &= \mathbf{D}\left(\overline{\mathcal{F}(\theta; \mathcal{X})}, \mathcal{F}(\theta; \mathcal{Y})\right). \end{aligned}$$

3 Implementation & Experiments

Our implementation is available here. The official implementation of SSD [4] is built on top of an already existing project [5] containing multiple other online continual learning solutions (including SCR). This leads to the code being more complex to read and maintain, and it also leads to the main SSD code being stored in `utils/buffer/summarize_update.py`. Because of this, we resorted to implementing both SCR and SSD from scratch. Considering multiple uncertainties unstated in the paper arose during implementation (ex. dynamic memory organization details, order of operations, some batch sizes, model details etc.), we did occasionally consult the official implementation.

During our analysis of the implementation, we even observed things that were just done differently from being described in the paper (ex. $\mathcal{M}_{ps}$ is referred to as $\mathcal{M}_s \setminus \mathcal{M}_c$ in the paper, while being implemented as described above, the paper states $\mathbf{D}$ to be Eucledian distance while it is almost never implemented like that, the implementation assumes that each batch from the stream is balanced even though that's not stated anywhere, they do 2 steps of summarizing model update for each batch, they employ a differentiable augmentation step in data summarization, etc.), and some of them affected the results of the experiments.

As SSD builds upon SCR, which in turn improves upon simple experience replay [3] (which itself is an enhancement over standard sequential memory-less training, "fine-tuning"), we designed the training loop to make it easy to switch between these methods. This structure highlights the incremental improvements each new technique introduces (shown in Table 2).

We design four datasets to experiment on (by splitting them into tasks of 10 classes): **Sequential CIFAR-100**, **Sequential Mini-ImageNet**, **Sequential Tiny-ImageNet** and **Sequential Food-101** [6]. We evaluate our models on them using *average end accuracy* (AEA) of all tasks. Comparing the results of first three to the original paper, we observe very similar results. As shown in Table 1, the most notable impact of SSD is observed with smaller memory sizes. As of the application deadline, we are still investigating the reasons behind the differences.

Table 1: Average end accuracy (AEA) on CIFAR-100

Gu et al., 2024 results		Our results			
SCR	SSD	SCR	SSD	SSD+IS	$\|\mathcal{M}\|$
9.0	12.1	9.1	12.4	12.8	100
20.6	23.0	19.7	22.2	22.8	500
26.6	28.8	26.4	27.8	28.4	1000

Table 2: Food-101

Method	AEA
finetune	3.76
ER	4.36
SCR	10.64
SSD	10.68
SSD + IS	11.14

During our experiments, we notice that implementing summarizing model with an incremental softmax (IS) that updates the last layer whenever new classes arrive (as opposed to creating a new model for each task, as in the official implementation) improves performances by a small margin. We also notice that due to the way the memory buffer organized, after the first few batches of the last task, $\mathcal{M}_O$ is empty until the end of the training session. Our experiments show that mitigating this by organizing the memory differently could also potentially enhance the final model performance.

References

[1] Gu, Jianyang & Wang, Kai & Jiang, Wei & You, Yang. (2024). Summarizing Stream Data for Memory-Constrained Online Continual Learning. Proceedings of the AAAI Conference on Artificial Intelligence. 38. 12217-12225. 10.1609/aaai.v38i11.29111.

[2] Mai, Z.; Li, R.; Kim, H.; and Sanner, S. 2021. Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning. In CVPR, 3589–3599.

[3] Chaudhry, A.; Rohrbach, M.; Elhoseiny, M.; Ajanthan, T.; Dokania, P. K.; Torr, P. H.; and Ranzato, M. 2019. On tiny episodic memories in continual learning. arXiv preprint arXiv:1902.10486.

[4] Official SSD implementation: `https://github.com/vimar-gu/SSD`

[5] Official SCR implementation: `https://github.com/RaptorMai/online-continual-learning`

[6] Food-101 Dataset: `https://www.kaggle.com/datasets/dansbecker/food-101`