UNDER REVIEW

RESCIENCE C

Replication Study

[Re] BiRT: Bio-inspired Replay in Vision Transformers for Continual Learning

Disha Maheshwari[1,]

[1] Purdue University, West Lafayette, Indiana

Edited by
(Editor)

Reviewed by
(Reviewer 1)
(Reviewer 2)

Received
15/6/2024

Published
–

DOI
–

1 Introduction

Continual learning (CL) poses a significant challenge for deep neural networks (DNNs) due to catastrophic forgetting of previously acquired tasks when new ones are introduced. Humans, however, excel at learning and adapting to new tasks without such forgetting, a capability attributed to the rehearsal of abstract experiences through complementary learning systems in the brain. This study aims to replicate and validate the findings of BiRT, a novel approach that leverages vision transformers to enhance representation rehearsal for continual learning. BiRT introduces constructive noise at various stages of the vision transformer and enforces consistency with an exponential moving average of the working model to mitigate overfitting and enhance robustness. By replicating BiRT's methodology, we seek to verify its claimed improvements in performance over traditional raw image rehearsal and vanilla representation rehearsal on several challenging CL benchmarks. Furthermore, this study examines BiRT's memory efficiency and robustness to natural and adversarial corruptions, aiming to reinforce its practical applicability. The replication will provide critical insights into the reproducibility and generalizability of ideas introduced in the original paper.

2 Replication Summary

The foundational codebase for the ViT model utilized in our replication study, as outlined in "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale" (Dosovitskiy et al., 2020), bears resemblance to the implementation presented in [1]. However, the code implementation for all the ideas introduced in the paper including memory updation algorithm, memory structure, and training architecture is entirely original and has not been taken from any other source meticulously adheres to the concepts introduced in the original paper. It's imperative to underscore that our adaptations and modifications to the experiments were carried out to facilitate a deeper analysis of the concepts delineated in the original work.

3 Analysis of the Original Paper

1. The authors did not provide specific values for several hyperparameters and parameters used in their equations, which impedes the ability to replicate their experimental setup precisely [2].

Correspondence should be addressed to Disha Maheshwari (dsmahesh@purdue.edu)
The authors have declared that no competing interests exists.
Code is available at https://github.com/rescience-c/template.

  - They omitted detailed information regarding crucial hyperparameters α_t, α_m, α_a, and α_s, as well as those in equations (3), (4), (6), and (8) essential for their architecture-specific experiments.
  - Additionally, the paper lacks information about the pre-processing steps applied to their datasets.

2. The proposed method addresses continual learning, where the model learns sequentially from disjoint datasets representing different tasks. The paper outlines a fine-tuning strategy on a balanced dataset after each task for 20 epochs. However, observations indicate that most learning occurs during the initial epochs. For instance, training the Vision Transformer (ViT) model on CIFAR-10 and CIFAR-100 datasets for just 5 epochs already achieves significant accuracies of 45% and 18%, respectively. This raises questions about the necessity of 20 epochs of fine-tuning after each task. Furthermore, fine-tuning on a balanced dataset that includes samples from all tasks after each task undermines the core principle of continual learning, as evidenced by the experimental results.

3. It is important to recognize that the testing accuracy largely depends on the Vision Transformer (ViT) model used for training, rather than solely on the training methodology, which includes the introduction of various constructive noises and the use of semantic memory. When the ViT model is trained on the entire dataset in a conventional manner (i.e., with mixed data), it reaches a saturation accuracy of ~65% (CiFAR10) and ~35% (CiFAR100). Achieving a higher accuracy than this baseline is not feasible when employing a continual learning (CL) approach with the same model. This limitation underscores that while continual learning techniques introduced in the paper aims to mitigate catastrophic forgetting and improve knowledge retention, the maximum achievable accuracy is still constrained by the inherent performance of the model itself.

4 Dataset and Computational Set Up

The datasets used for training and testing in this replication study were CIFAR-10 and CIFAR-100.

- **CiFAR10**: It consists of 60,000 32x32 color images in 10 different classes, with 6,000 images per class. The dataset is split into 50,000 training images and 10,000 testing images.
- **CiFAR100**: It consists of 60,000 32x32 color images in 100 different classes, with 600 images per class. The dataset is split into 50,000 training images and 10,000 testing images.

The transformations applied to the input images include converting to PyTorch tensors, resizing to 32x32 pixels, randomly flipping horizontally for data augmentation, random cropping and resizing with specified scales and aspect ratios, and normalizing pixel values to the range [-1, 1]. To prepare the CIFAR-10 and CIFAR-100 datasets for continual learning, the datasets were divided by class into subsets, with the number of subsets corresponding to the number of tasks. Each task comprised distinct classes, ensuring that all tasks remained disjoint. The model was then trained sequentially on these tasks. After training on each task, the model was fine-tuned on a small, balanced dataset containing samples from all tasks to consolidate learning and mitigate forgetting. The entire model was trained on a single NVIDIA V100 GPU with 32 GB RAM.

5 Key Ideas Introduced in the Paper

The continual learning paradigm normally consists of T sequential tasks, with the data gradually becoming available over time. During each task $t \in \{1, 2, \ldots, T\}$, the samples and the corresponding labels $(x_i, y_i)_{i=1}^{N}$ are drawn from the task-specific distribution D_t.

5.1 Knowledge consolidation through complementary learning system

Complementary learning systems theory proposes that the hippocampus and neocortex possess complementary properties crucial for capturing complex brain interactions [3]. Inspired by CLS (Complementary Learning System) they propose a dual memory transformer-based learning system in which the working model encounters new tasks and consolidates knowledge over short periods of time which is then gradually aggregated into the weights of the semantic memory during intermittent stages of inactivity.

$$\theta_s = \gamma\theta_s + (1-\gamma)\theta_w \tag{1}$$

Here, θ_w represents the working model parameters, θ_s denotes the parameters stored in the semantic memory, and γ is a hyperparameter controlling the update rate.

5.2 Episodic Memory

They propose a high level representation rehearsal for vision transformers. The working model comprises two nested functions: g and f_w. The first few layers of the encoder g, processes the raw image input, and the output along with the ground truth label is stored in the episodic memory D_m, f_w and its stable counter part f_s is updated according to equation 1. They populate the episodic memory at the task boundary using iCarL herding [4] at the end of task boundry. The algorithm used to implement iCarl herding is described in the next section. The learning objective for representation rehearsal is given in equation 2

$$L_{er} = \mathbb{E}_{(x_i,y_i)\sim D_t}\left[\mathcal{L}_{ce}(f_w(g(x_i)), y_i)\right] + \alpha\, \mathbb{E}_{(x_j,y_j)\sim D_m}\left[\mathcal{L}_{ce}(f_w(r_j), y_j)\right] \tag{2}$$

Here, r_j represents the representations $g(x_j)$ stored in episodic memory, α is the hyperparameter and D_t is the dataset.

5.3 Noise and Trial-to-Trial Variability

Noise is prevalent at every level of the nervous system and has been shown to play constructive role in brain. Furthermore, injecting noise into the neural network learning pipeline has been shown to result in faster convergence to the global optimum [5] , better generalization [6], and effective knowledge distillation.

Representation Noise $\widetilde{M}$ –

$$\widetilde{r} = \lambda \mathbf{r}_i + (1-\lambda)\mathbf{r}_j \tag{3}$$

$$\widetilde{y} = \lambda \mathbf{y}_i + (1-\lambda)\mathbf{y}_j \tag{4}$$

The authors propose to linearly combine the representations sampled from episodic memory using a manifold mixup as shown in equation 3 where r_i and r_j are stored representations of two different samples, and y_i and y_j are the corresponding labels. However, this concept has not been investigated in our implementation. Instead, our focus was on evaluating the effects of various noises introduced during the training process. We opted to first assess the model's performance without introducing noise into

the training dataset. This approach allowed us to establish a baseline performance before introducing noise, enabling a more thorough examination of the impact of noise on the training process.
Furthermore,the impact of representation noise appears to be minimal, as demonstrated by ablation studies conducted on the CIFAR-100 dataset in [2]. When only representation noise is applied, the accuracy is 46.89%, which is relatively low and closely comparable to the baseline accuracy of 45.89% obtained without any noise. This suggests that representation noise has a limited effect on model performance. In contrast, the application of only supervision noise and attention noise yields higher accuracies of 49.19% and 49.30%, respectively. These findings indicate that supervision noise and attention noise have a more substantial influence on improving model performance compared to representation noise.

Attention Noise $\widetilde{A}$ – The working model $f_w(.)$ in BiRT consists of several multi-head self-attention layers that map a query and a set of key-value pairs to an output. The authors inject noise into the scaled dot-product attention at each layer of $f_w(.)$ while replaying the representation as shown in equation 5.

$$\text{Attention}(Q, K, V) = \left(\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) + \epsilon\right) V \tag{5}$$

where Q, K, and V are query, key, and value matrices, and $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is a white Gaussian noise. By stochastically injecting noise into self-attention, they discourage BiRT from overfitting.

Label Noise $\widetilde{T}$ – The author introduce a synthetic label noise $\widetilde{T}$ in which they re-assign a small percentage of the samples a random class, thus taking advantage of the fact that label noise is sparse in the real world [7].

Supervision Noise $\widetilde{S}$ – They also regularize the function learned by the working model to enforcing consistency in its predictions with respect to the semantic memory using equations 6 and 7.

$$L_{cr} = \beta_1 \mathbb{E}_{x_i \sim D_t} \|f_w(g(x_i)) - f_s(g(x_i))\|_p + \beta_2 \mathbb{E}_{r_j \sim D_m} \|f_w(r_j) - f_s(r_j)\|_p \tag{6}$$

$$f_s(r_j) \leftarrow f_s(r_j) + \delta, \tag{7}$$

where β_1 and β_2 are balancing weights, $\delta \sim \mathcal{N}(\mu, \sigma^2)$ is a white Gaussian noise, and L_{cr} represents the expected Minkowski distance between the corresponding pairs of predictions, and $p = 2$. Thus the final learning objective becomes equation 8

$$L = L_{\text{repr}} + \rho L_{\text{cr}} \tag{8}$$

5.4 Ideas Implemented in the Replication Study

The ideas implemented from the paper are

- Knowledge consolidation through complementary learning system as given in equation 1
- Episodic Memory
- Attention Noise $\widetilde{A}$ as given in equation 5
- Label Noise $\widetilde{T}$

- Supervision Noise $\widetilde{S}$ as described in equation 7
- The final learning objective which is given by equation 8 by combining equations 6, 2.

6 Models and Algorithms used in the Replication Study

6.1 Episodic Memory Architecture

During task training, the representations generated by the model g are sequentially appended to a task-specific list. Upon the completion of each task, these representations undergo a sorting process based on their associated classes. Leveraging the iCarl herding technique, representations within each class are ordered in a descending manner, from the most to the least representative. This process is illustrated in Algorithm 1. To ensure equitable representation of all classes, the first n representations from each class encountered by the model up to the current task are retained, as shown in Algorithm 2. The procedure for sampling batches from episodic memory during training is detailed in Algorithm 3.

Algorithm 1 iCaRL Construct Exemplar Set

```
Input: image set $X = \{x_1, \dots, x_n\}$ of class $y$
Input: $m$ target number of exemplars
Require: Current feature function $\phi :\to \mathrm{R}^d$
$\mu \leftarrow \frac{1}{n} \sum_{x \in X} \phi(x)$   // current class mean
for $k = 1, \dots, m$ do
    $p_k \leftarrow \underset{x \in X}{\operatorname{argmin}} \left\| \mu - \frac{1}{k}[\phi(x) + \sum_{j=1}^{k-1} \phi(p_j)] \right\|$
end for
$P \leftarrow (p_1, \dots, p_m)$
Output: exemplar set $P$
```

Algorithm 2 iCaRL Reduce Exemplar Set

```
Input: $m$                          // target number of exemplars
Input: $P = (p_1, \dots, p_{|P|})$        // current exemplar set
$P \leftarrow (p_1, \dots, p_m)$              // keep only first $m$
Output: exemplar set $P$
```

Algorithm 3 Sample Batch

```
Input: Buffer Images: $I$, Buffer Labels: $L$, Batch Size: $s$
Output: Batch Images: $b$, Batch Labels: $l$
$b \leftarrow []$
$l \leftarrow []$
for $i = 1$ to $s$ do
    $index \leftarrow \mathrm{randint}(0, \mathrm{num\_elements} - 1)$
    $b[i] \leftarrow I[index]$
    $l[i] \leftarrow L[index]$
end for
$b \leftarrow \mathrm{stack}(b, \mathrm{dim} = 0)$
$l \leftarrow \mathrm{stack}(l, \mathrm{dim} = 0)$
return $b, l$
```

6.2 Training Algorithm

The taining algorithm used in this replication study is presented in algorithm 4.

Algorithm 4 BiRT Algorithm

```
input: Data streams $\mathcal{D}_t$, buffer $\mathcal{D}_m$, working model $f_w$, hyperparameters $\gamma$, $\alpha_t$, $\alpha_m$, $\alpha_a$, $\alpha_s$
for tasks $t \in \{1, 2, .., T\}$ do
    for epochs $e \in \{1, 2, .., E\}$ do
        Sample a mini-batch $(x, y) \sim \mathcal{D}_t$
        if $\mathcal{D}_m \neq \emptyset$ then
            Sample a mini-batch $(r, y) \sim \mathcal{D}_m$
            $a, b, c, d, e \sim \mathcal{U}(0, 1)$
            $\tilde{y} \leftarrow \tilde{\mathcal{T}}(y)$            if $a < \alpha_t$
            $\tilde{A} \leftarrow \tilde{\mathcal{A}}(A)$            if $c < \alpha_a$          ▷ (Eq. 5)
            $f_s(r) \leftarrow \tilde{\mathcal{S}}(f_s(r), \delta)$  if $d < \alpha_s$          ▷ (Eq. 7)
        end if
        Compute outputs of $f_w(.)$ and $f_s(.)$
        Compute $\mathcal{L} = \mathcal{L}_{repr} + \rho \mathcal{L}_{cr}$          ▷ (Eqs. 2, 6, 8)
        $\theta_w \leftarrow \theta_w + \nabla_{\theta_w} \mathcal{L}$
        $\theta_s \leftarrow \gamma \theta_s + (1 - \gamma) \theta_w$  if $e < \alpha_e$ and $t > 1$
    end for
    if task-end = True then
        if t = 1 then
            Freeze $g(.)$
            $\theta_s$ = copy($\theta_w$)
        end if
        $\mathcal{D}_m \leftarrow (r, y)$
    end if
end for
Return: working model $\theta_w$, and semantic memory $\theta_s$
```

6.3 Vision Transformer and Working Model Architecture (f_w/f_s and g)

Key Components of the ViT Architecture — The Vision Transformer (ViT) architecture consists of several key components:

- **Patch + Position Embedding**: The input image is divided into smaller patches (e.g., 16x16 pixels). Each patch is linearly projected to create an embedding, and positional information is added to retain the spatial arrangement of the patches.
- **Transformer Encoder Blocks**: The embedded patches pass through a series of encoder blocks. Each block includes:
    - **Multi-Head Attention**: This layer captures relationships between patches, helping the model understand the global context across the image.
    - **Normalization Layers**: These layers are applied both before and after each MLP layer to stabilize training.
    - **MLP Layer**: Each encoder block contains an MLP layer for further processing.
- **MLP Head**: The final output of the Transformer encoder is passed to an MLP head, which produces the classification result.

Working Model Structure — The architecture for the BiRT model is divided into different parts for specialized processing:

- **Model** g: This part consists of the embedding layer and the first two encoder blocks of the encoder stack. It takes an input with dimensions *batch* × *channel* × *height* × *width*, and outputs features with dimensions *batch* × (*patches* + 1) × *hidden_size*.
- **Models** f_s **and** f_w: These parts include the remaining encoder blocks and the MLP head. They take inputs with dimensions *batch* × (*patches* + 1) × *latent_size* and produce an output of dimensions *batch* × 1, representing the final classification prediction.

Table 1 lists the parameter values used for the ViT architecture.

Table 1. Parameters for ViT

Variable	Description	Value
patch_size	Image patch size	4
hidden_size	Output patch size	48
num_hidden_layers	Encoder blocks	4
num_attention_heads	Attention heads	4
intermediate_size	Intermediate layer size	192
hidden_dropout_prob	Hidden dropout	0.0
attention_probs_dropout_prob	Attention dropout	0.0
initializer_range	Weight initialization range	0.02
image_size	Input image size	32
num_classes	Output classes	10
num_channels	Input channels	3
qkv_bias	Bias in projections	True
use_faster_attention	Faster attention	True

7 Tests and Results: CiFAR10

All the Training and Testing was done on V100 GPU with 32GB RAM. The training parameters used are summarized in table 2

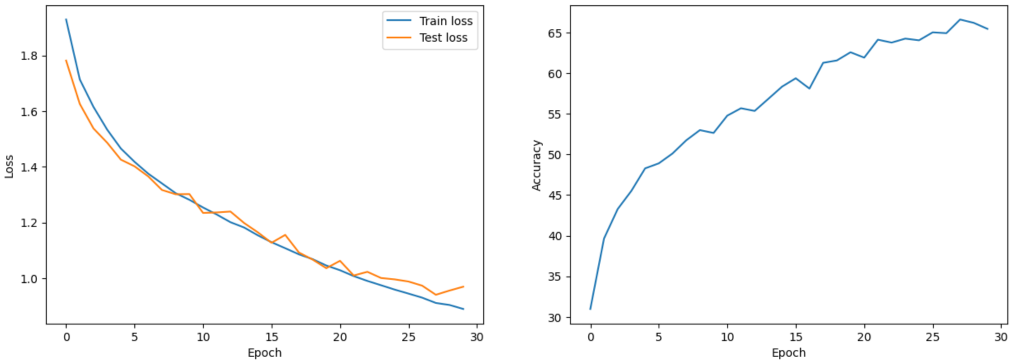
Table 2. General Training Parameters Used in the Experiments

Parameter	Value
batch_size	32
lr	0.0005
learning_rate	5e-4
weight_decay	1e-6
optimizer	Adam
loss	Cross Entropy Loss

7.1 CiFAR10 with ViT Model

The loss and accuracy of the Vision Transformer (ViT) model [8] after each training epoch are depicted in Figure 1. The parameters of the ViT model are summarized in Table 1. The dataset was not segregated into tasks for this experiment. An accuracy of 65% is achieved after training the model for 30 epochs. The time taken to train over the entire dataset per epoch is approximately 52 seconds.

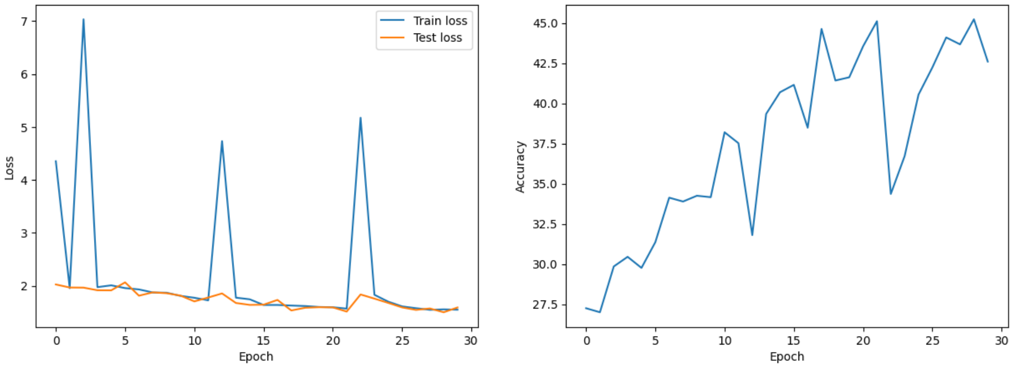
Figure 1. Accuracy and Test Loss for ViT trained on CiFAR10



7.2 CiFAR10 with ViT Model separated into g and f

The performance metrics, including loss and accuracy, of the Vision Transformer (ViT) model, which is decomposed into g and f functions as described in the BiRT paper, are illustrated in Figure 2. The model achieves a final accuracy of 42.5% after 30 epochs, closely matching the performance of the ViT model without decomposition. Detailed parameters of the ViT model are provided in Table 1. It is important to note that the dataset was not partitioned into tasks for this experiment. The training duration per epoch for the entire dataset is ~52 seconds. Partitioning the Vision Transformer (ViT) model into separate f and g components significantly destabilizes its training process, leading to a notable reduction in accuracy to 45%.

Figure 2. Test Loss and Accuracy for ViT split into g and f trained on CiFAR10



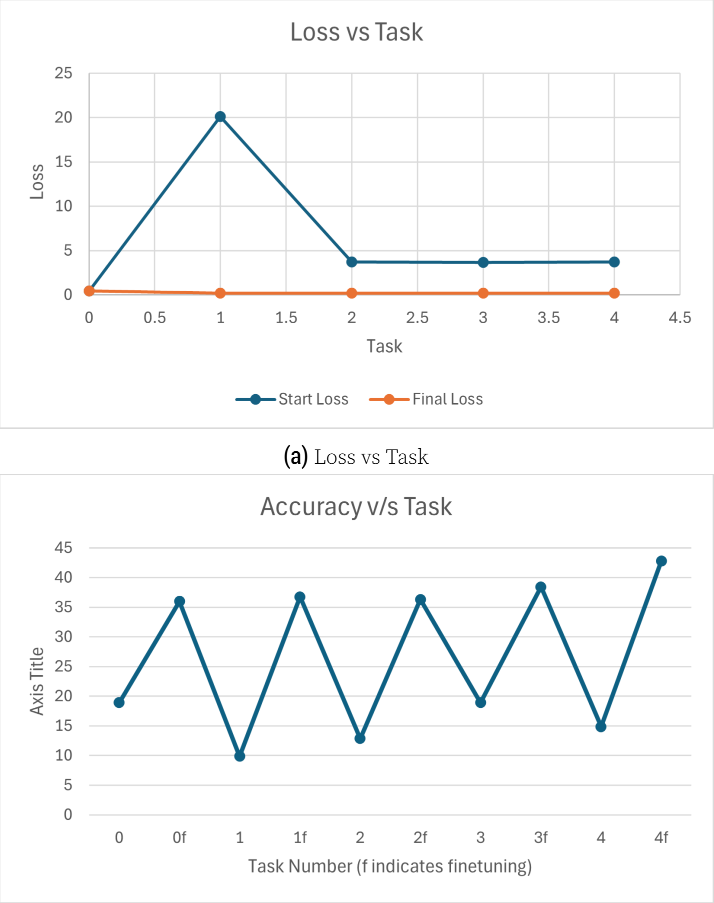
7.3 CiFAR10 with BiRT architecture for Continual Learning

The experimental setup and parameters for training CIFAR-10 using the BiRT architecture are presented in Tables 3 and 1. The CIFAR-10 dataset, comprising 10 classes, was partitioned into 5 distinct tasks, each containing 2 classes. The model underwent training for 200 epochs per task, followed by fine-tuning on a small, balanced subset of 1000 images for an additional 20 epochs post each task. The average duration for memory updates following each task was 16 minutes. The average training time per epoch per task was 55 seconds, resulting in a total of 184 minutes per task for 200 epochs for all the task except the first one for which it takes ~12 secs per epoch. The loss per epoch

Table 3. Parameters for BiRT Architecture with CiFAR-10

Parameter	Value	Parameter	Value
α_t	0.005	α_a	0.005
α_s	0.005	α_e	0.003
$\alpha_{\text{loss\_rep}}$	0.4	$\rho_{\text{loss\_cr}}$	1
$\beta_{1\text{loss}}$	0.05	$\beta_{2\text{loss}}$	0.01
γ	0.005	$PercentageChange(labelnoise)$	5
μ	0	σ^2	1
Semantic Memory Length	500		

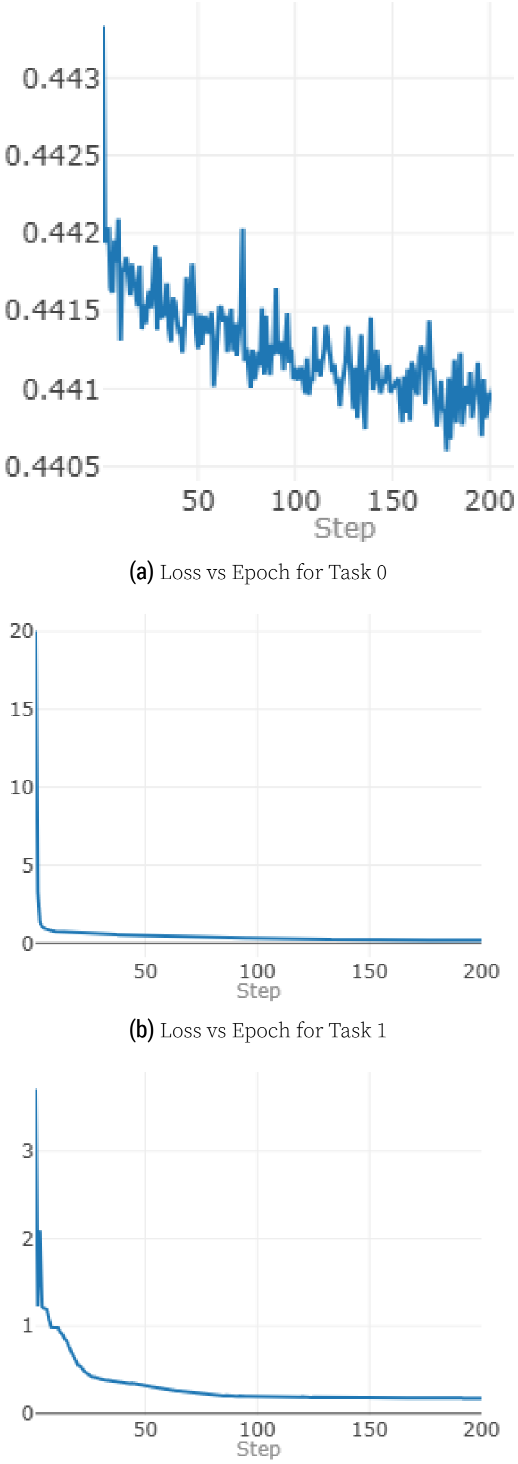
for each task is summarized in figure 3 and 4, while the accuracy metrics are detailed in figure 5.
As can be seen from figure 3 and 4, the loss at the start of each task is very high. This high initial loss indicates that the model initially struggles with new, unseen images. However, the final loss after training drops dramatically to nearly 0 for all tasks, demonstrating effective learning. A crucial observation, as can be seen from 5 is that finetuning significantly enhances accuracy after each task, indicating its effectiveness in improving model performance in sequential learning scenarios.



(a) Loss vs Task

(b) Accuracy for each Task

Figure 5. Accuracy and Loss metrics for Cifar10

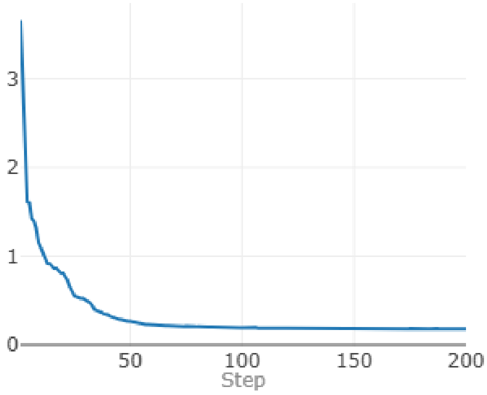


(a) Loss vs Epoch for Task 0

(b) Loss vs Epoch for Task 1

(c) Loss vs Epoch for Task 2

Figure 3. Loss for Cifar10 for 200 epochs for task 0-2



(a) Loss vs Epoch for Task 3 **(b)** Loss vs Epoch for Task 4

Figure 4. Loss for Cifar10 for 200 epochs for task 3-4

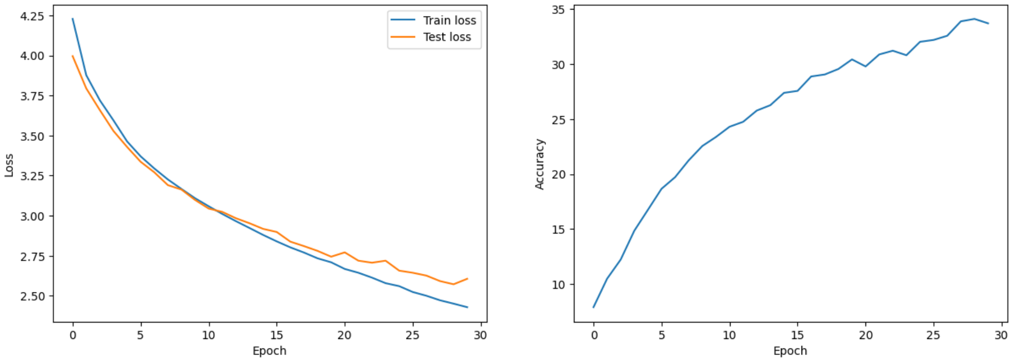
8 Test and Results: CiFAR100

The authors trained their BiRT model on 5 tasks of CiFAR 100 for 500 epochs for each task and fine tuned the model for 20 epochs on after each tasks. They report their last accuracy (accuracy on the test dataset of CiFAR 100 after the model has been done training) as 54.15 %. It took them average of 45 mins to train each tasks for 500 epochs.

8.1 ViT Model with CiFAR100

The loss and accuracy of ViT model [8] obtained after each training epoch is shown in graph 6. The parameters of ViT model are summarized in Table 1. An accuracy of 35% is achieved after training the model for 30 epochs. The dataset was not segregated into tasks for this experiment. Time taken to train over the entire dataset per epoch is ~55 secs.

Figure 6. Loss and Accuracy of ViT with CiFAR 100 after 30 epochs



8.2 ViT Model when split into g and f with CiFAR100

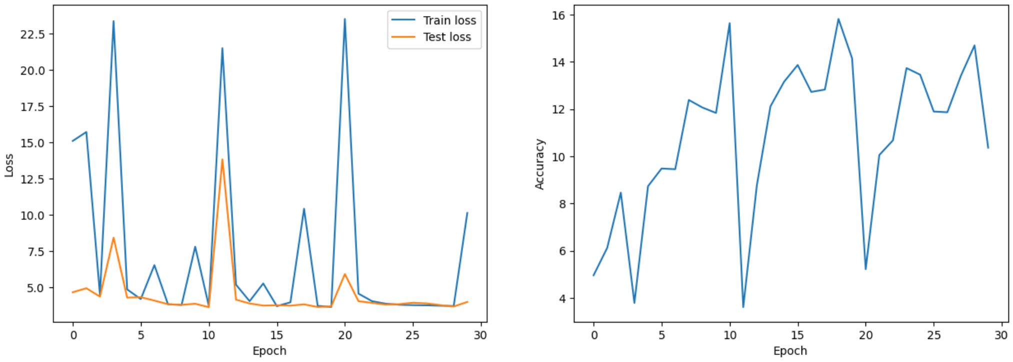
The performance metrics of the Vision Transformer (ViT) model, decomposed into g and f functions as per the BiRT paper, are shown in Figure 10. The model achieves a final accuracy of ~14% after 30 epochs. Detailed ViT parameters are in Table 1. The dataset was not partitioned into tasks, and training per epoch took ~52 seconds. Similar

Table 4. Parameters for BiRT Architecture with CiFAR-100

Parameter	Value	Parameter	Value
α_t	0.005	α_a	0.005
α_s	0.005	α_e	0.003
$\alpha_{\text{loss\_rep}}$	0.4	$\rho_{\text{loss\_cr}}$	1
$\beta_{1\text{loss}}$	0.05	$\beta_{2\text{loss}}$	0.01
γ	0.005	$PercentageChange(labelnoise)$	5
μ	0	σ^2	1
Semantic Memory Length	500		

to CiFAR10 experiment, partitioning the ViT model into separate f and g components significantly destabilizes its training process, leading to a notable reduction in accuracy to ~14%.

Figure 7. Loss and Accuracy of ViT when split into f and g with CiFAR 100 after 30 epochs

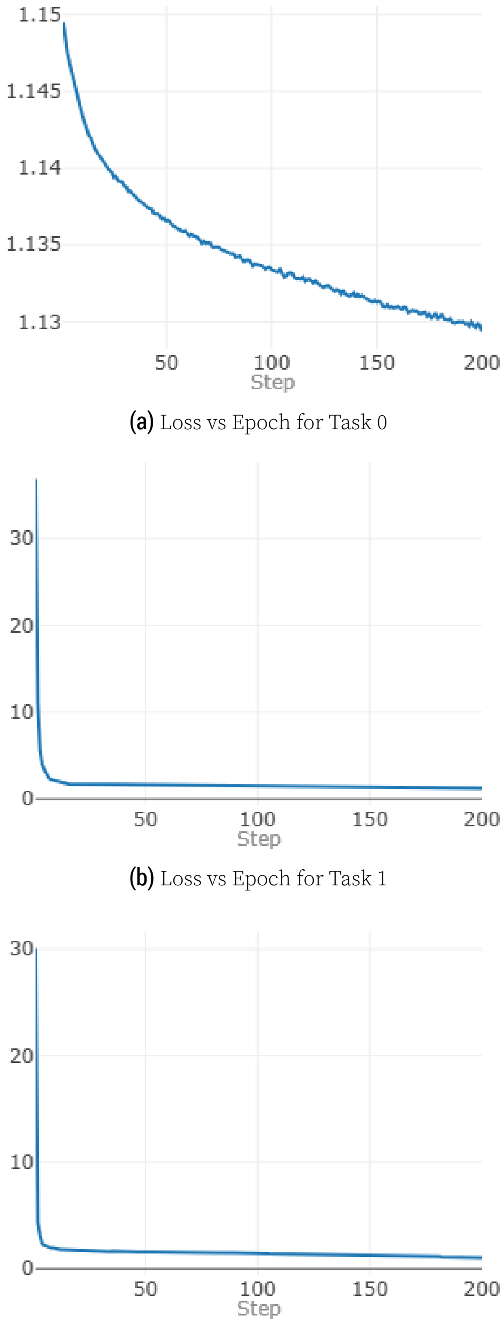


8.3 BiRT Model with CiFAR 100

The BiRT model in this implementation was trained on V100 GPU with 32 GB RAM. The dataset was divided into 5 task each with 20 classes in each.

8.4 Training BiRT on CiFAR 100 for 200 epochs

The experimental setup and parameters for training CIFAR-100 using the BiRT architecture are presented in Tables 4 and 1. The CIFAR-100 dataset, comprising 100 classes, was partitioned into 5 distinct tasks, each containing 20 classes. The model underwent training for 200 epochs per task, followed by fine-tuning on a small, balanced subset of 1000 images for an additional 20 epochs post each task. The average duration for memory updates following each task was 16 minutes. The average training time per epoch per task was 58 seconds, resulting in a total of ~196 minutes per task for 200 epochs for all the task except the first one for which it takes ~12 secs per epoch. The loss per epoch for each task is summarized in figures 8 and 9, while the accuracy metrics are detailed in figure 10. The model was trained for 200 epochs instead of 500 epochs as reported in [2], primarily due to the observation from training graphs indicating diminishing returns in loss reduction beyond 100 epochs. Similar to CiFAR10 experiment, figure 10 reveals that the initial loss for each task is quite high, signifying the model's difficulty with new, unseen images at the start. Nevertheless, the final loss drops significantly to almost zero after training, indicating successful learning. Notably, finetuning greatly improves accuracy after each task, underscoring its critical role in boosting model performance in sequential learning settings.

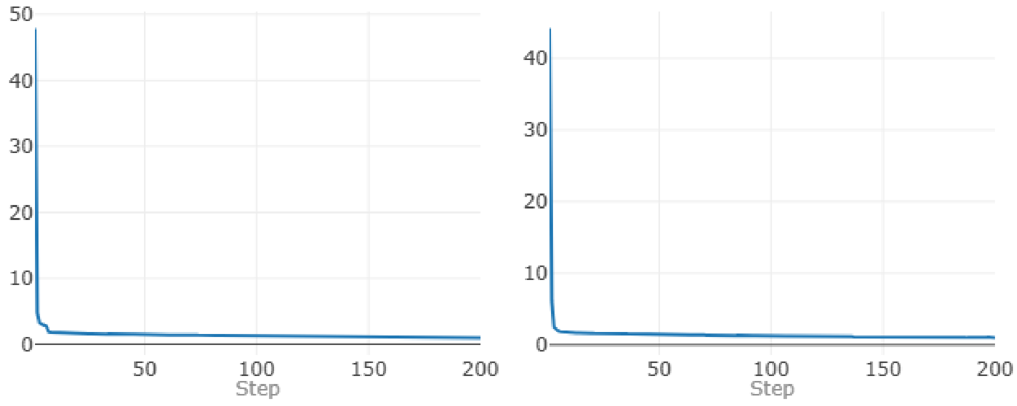


(a) Loss vs Epoch for Task 0

(b) Loss vs Epoch for Task 1

(c) Loss vs Epoch for Task 2

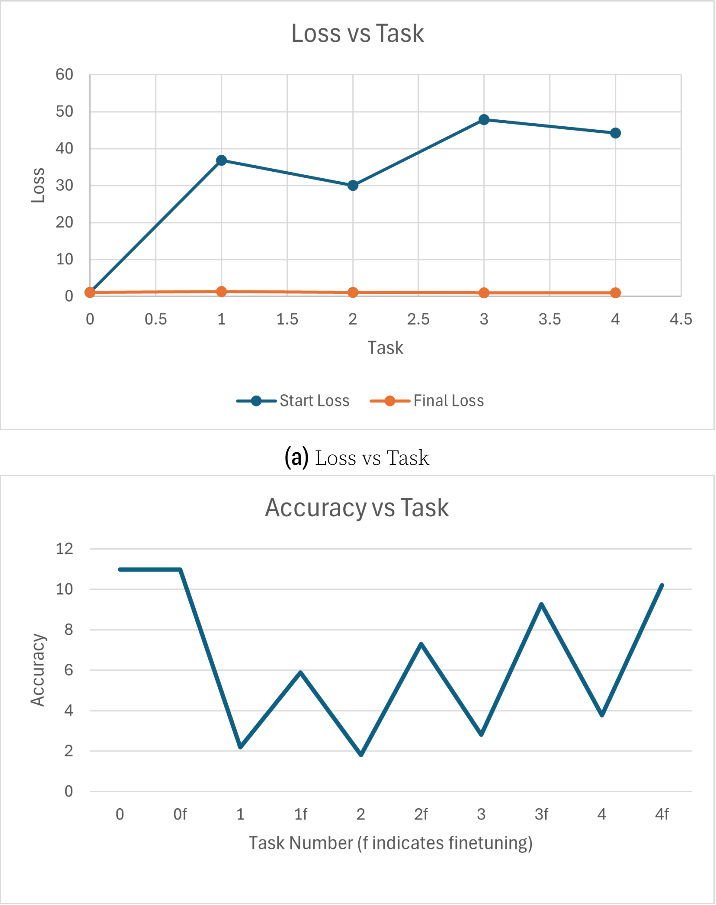
Figure 8. Loss for Cifar100 for 200 epochs for Task 0-2



(a) Loss vs Epoch for Task 3

(b) Loss vs Epoch for Task 4

Figure 9. Loss for Cifar100 for 200 epoch for Task 3-4



(a) Loss vs Task

(b) Accuracy for each Task

Figure 10. Accuracy and Loss metrics for Cifar100

8.5 Conclusion

In this replication study, we observe that the separation of the ViT model into f and g components introduces considerable instability in training, as evidenced by the loss and accuracy metrics. It is essential for the authors to clearly specify the exact structure of the ViT model utilized, given that achieving an accuracy beyond the saturation level of the ViT model in Continual Learning scenarios is not possible. Additionally, the initial loss for each task is notably high, indicating the model's difficulty with new, unseen images at the outset. However, the final loss significantly decreases to nearly zero after training, demonstrating substantial learning. Finetuning markedly improves accuracy after each task, which runs counter to the principles of Continual Learning, where tasks are trained sequentially without such interventions. Therefore, the authors should present results without finetuning after every task to accurately assess the true implications of their method.

9 Implementation Repository

The implementation code for this study can be found at https://github.com/disha101003/ReScience.

References

1. Tintn. **Vision Transformer from Scratch**. https://github.com/tintn/vision-transformer-from-scratch.
2. K. Jeeveswaran, P. Bhat, B. Zonooz, and E. Arani. "BiRT: Bio-inspired Replay in Vision Transformers for Continual Learning." In: **arXiv preprint arXiv:2305.04769** (2023).
3. B. L. McNaughton and R. C. O'Reilly. "Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of." In: **Psychological Review** 102.3 (1995), pp. 419–457.
4. S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. "icarl: Incremental classifier and representation learning." In: **Proceedings of the IEEE conference on Computer Vision and Pattern Recognition**. 2017, pp. 2001–2010.
5. M. Zhou, T. Liu, Y. Li, D. Lin, E. Zhou, and T. Zhao. "Toward understanding the importance of noise in training neural networks." In: **International Conference on Machine Learning**. PMLR. 2019, pp. 7594–7602.
6. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." In: **The Journal of Machine Learning Research** 15.1 (2014), pp. 1929–1958.
7. S. Liu, Z. Zhu, Q. Qu, and C. You. "Robust Training under Label Noise by Over-Parameterization." In: **International Conference on Machine Learning**. PMLR. 2022, pp. 14153–14172.
8. A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." In: **arXiv preprint arXiv:2010.11929** (2020).