A Effect of Pruning Criteria

We seek to illustrate the effectiveness of different pruning criteria in TriRE. As explained in Section 3.1, the dense network is first pruned using k-WTA criteria, resulting in a subnetwork of the most activated neurons, and then this subnetwork is pruned using CWI criteria, resulting in a final extracted subnetwork at the end of *Retain* stage. Table 3 demonstrates the comparison of Class-IL accuracy between various pruning criteria, namely, magnitude-based, Fisher information-based, and CWI-based, across all three datasets. The idea behind magnitude pruning is that small valued weights impact the network's output less and can be safely pruned without significantly affecting performance. Fisher information-based pruning evaluates the importance of connections based on their contributions to the Fisher information matrix. Connections with low contributions, indicating less relevance or importance, are pruned or set to zero. However, both these criteria calculate the importance of weights within the current task, but do not consider the possibility of it being crucial for other tasks. On the other hand, CWI considers the significance of weights with respect to data saved in the rehearsal buffer as well, resulting in superior performance across all datasets.

Table 3: Comparison of the effect of various pruning criteria in TriRE on different datasets.

Dataset	Magnitude	Fisher Information	CWI
Seq-CIFAR10	65.09±0.83	64.40±0.43	**68.17**±0.33
Seq-CIFAR100	41.89±0.74	40.26±0.21	**43.91**±0.18
Seq-TinyImageNet	19.07±0.97	18.16±0.75	**20.14**±0.19

B Model analysis

B.1 Task Recency Bias

In any CL setting, the model entails learning on a few or no samples from previous tasks while aplenty of the most recent task [21]. This tilts learning toward the most recent task, resulting in decisions biased toward new classes and confusion among the old classes. However, the CL model should ideally have predictions distributed evenly across all tasks with the least possible recency bias. Figure 6 provides the confusion matrix for various CL models to evaluate the task recency bias. After training on Seq-CIFAR100 for 5 tasks with a buffer size of 200, the model is deemed to have correctly predicted the task label if it predicts any of the classes that make up the sample's true task label. As can be seen, ER and DER++ have a propensity to frequently classify the majority of samples as classes in the most recent task. However, TriRE's predictions are uniformly distributed across the diagonal. TriRE essentially decreases interference between tasks, captures task-specific information through extracted sub-networks, and produces the least recency bias.

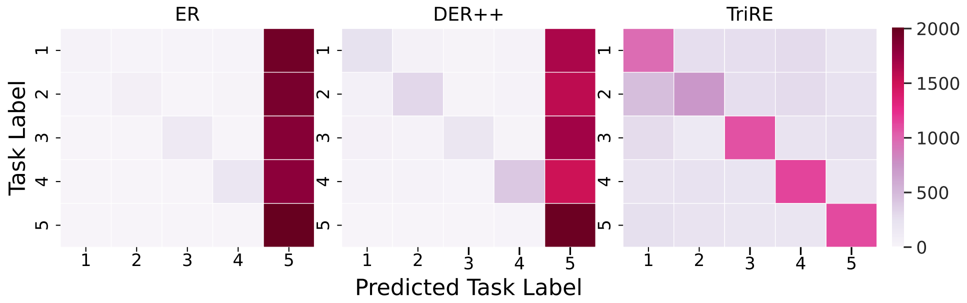

Figure 6: Confusion matrix of different rehearsal-based CL models. Unlike ER and DER++, TriRE predictions are evenly distributed across the tasks with the least recency bias.

B.2 Stability-Plasticity Dilemma

A CL model is said to be stable if it can retain previously learned information, and plastic if it can effectively acquire new information. The stability-plasticity dilemma refers to an inherent trade-off

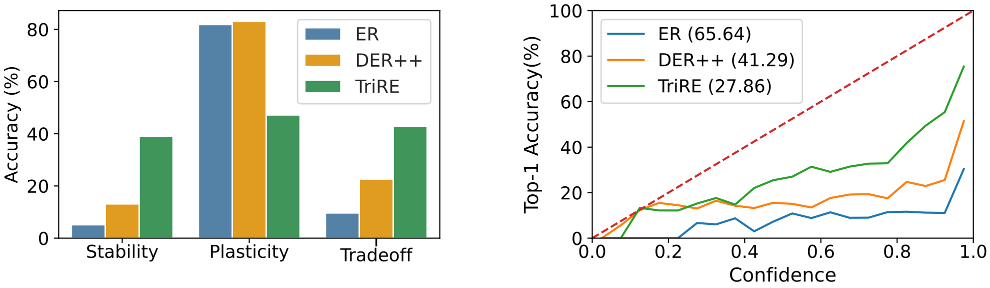

Figure 7: (Left) Stability-Plasticity Trade-off for CL models trained on Seq-CIFAR100 with 5 tasks. ER and DER++ are more plastic than stable leading to recency bias. TriRE maintains a better balance between stability and plasticity and achieves the highest trade-off amongst the baselines. (Right) Reliability diagram depicting model calibration: The red dashed line represents the ideal scenario. Compared to the other two methods, TriRE is better calibrated with the lowest ECE value. All models were trained on Seq-CIFAR100 with 5 tasks.

in which the CL model masters one of these aspects at the expense of the other. Sarfraz et al. (2022) [43] introduced a trade-off measure that serves as an approximation of how the model balances its stability and plasticity. Once the model completes the final task T, its stability (S) is assessed by calculating the average performance across all preceding $T-1$ tasks as follows:

$$S = \sum_{i=0}^{T-1} A_{Ti} \tag{5}$$

The plasticity of the model (P) is evaluated by computing the average performance of each task after its initial learning i.e.,

$$P = \sum_{i=0}^{T} A_{ii} \tag{6}$$

Thus, the trade-off measure determines the optimal balance between the stability (S) and the plasticity (P) of the model. This measure is calculated as the harmonic mean of S and P.

$$\textit{Trade-off} = \frac{2SP}{S+P} \tag{7}$$

Figure 7 (Left) provides the stability-plasticity trade-off measure for different CL methods across different datasets for a buffer size of 200. ER and DER++ exhibit high plasticity, enabling them to rapidly adapt to new information. However, they lack the ability to effectively retain previously acquired knowledge. On the other hand, TriRE exhibits substantially high stability with low plasticity, resulting in a higher stability-plasticity trade-off.

B.3 Model Calibration

Ensuring the reliability of safety-critical CL systems necessitates the presence of a well-calibrated model. Calibration refers to the task of accurately predicting probability estimates that reflect the true likelihood of correctness. Miscalibration, on the other hand, refers to the disparity between confidence and accuracy expectations. To assess the degree of miscalibration in classification, the Expected Calibration Error (ECE) involves partitioning the predictions into bins of equal size and calculating the difference between the weighted average of accuracy and confidence within each bin. A lower ECE value indicates better calibration in the underlying models. In Figure 7 (Right) shows a comparison of different CL approaches using a calibration framework trained on Seq-CIFAR100 with a buffer size of 200. Well-calibrated CL systems accurately represent the true likelihood of accuracy (indicated by the red dashed line). Among the baselines, TriRE achieves the lowest ECE value and exhibits high calibration, demonstrating its effectiveness in minimizing task interference and reducing overconfidence in CL, thus enabling more informed decision making.

C Limitations

We proposed TriRE, a novel CL paradigm that encompasses *retaining* the most prominent neurons for each task, *revising* and solidifying the knowledge extracted from current and past tasks, and actively promoting less active neurons for subsequent tasks through *rewinding* and relearning. As TriRE leverages the advantages of multiple orthogonal CL approaches, the selection of such approaches needs careful consideration, as these approaches may not always be complementary to each other. In addition, having multiple objective functions naturally expands the number of hyperparameters, thereby requiring hyperparameter tuning to achieve optimal performance. We also highlight that Retain, Revise, and Rewind steps are mainly proposed for CNN-based architectures. Therefore, more diligence is necessary when extending our method to other architectures such as vision transformers.

D Hyperparameter Selection

The hyperparameters required to replicate the results of TriRE can be found in Table 4. These hyperparameters were determined through a tuning process involving different random initializations and a small portion of the training set reserved for validation. All experiments were conducted using a batch size of 32 and trained for 50 epochs. TriRE was optimized using the Adam optimizer [23] implemented in PyTorch. Furthermore, the number of epochs allocated to each phase specified in Algorithm 1 was consistently set at a ratio of $E_1 : E_2 : E_3 = 3 : 1 : 1$.

Table 4: Best hyperparameters of TriRE chosen for optimal performance on different datasets.

Dataset	η	η'	γ	λ	EMA Parameters		Rewind Percentile
					μ	ζ	
Seq-CIFAR10	0.0006	0.0001	0.4	0.06	0.999	0.18	0.9
Seq-CIFAR100	0.002	0.0001	0.2	0.04	0.999	0.12	0.9
Seq-TinyImageNet	0.002	0.0001	0.3	0.05	0.999	0.01	0.8

E Datasets and Settings

We assess the effectiveness of our approach in two different types of CL scenarios: Class Incremental Learning (Class-IL) and Task Incremental Learning (Task-IL). In Task-IL and Class-IL, each task consists of a predetermined number of new classes that the model needs to learn. A CL model learns multiple tasks in sequence while being able to differentiate between all classes it has encountered so far. Task-IL is similar to Class-IL, but it has the advantage of having access to task labels during the inference process, making it one of the easiest scenarios.

To evaluate the performance of our method in Task-IL and Class-IL scenarios, we employ three different datasets: Seq-CIFAR10, Seq-CIFAR100, and Seq-TinyImageNet. These datasets are derived from CIFAR10, CIFAR100, and TinyImageNet, respectively. In Seq-CIFAR10, CIFAR10 is divided into five tasks, each task containing two classes. Similarly, in Seq-CIFAR100, CIFAR100 is divided into five tasks, each consisting of 20 classes. Lastly, in Seq-TinyImageNet, we partition TinyImageNet into ten tasks, each of which comprises 20 classes. These datasets are designed to introduce more challenging scenarios for a comprehensive analysis of various CL methods. By increasing the number of tasks or the number of classes per task, we can thoroughly examine the effectiveness of different CL approaches in handling different levels of complexity. Following [6], we used ResNet-18 as the backbone in all our experiments. The training process remains consistent for both Class-IL and Task-IL. To compare various state-of-the-art approaches, we present the average accuracy across all tasks encountered in Class-IL. According to Task-IL conventions, we take advantage of task identity and selectively deactivate neurons in the linear classifier that are not related to the current task.

Contrary to the common practice of using dense CL models, dynamic sparse approaches take a different approach by starting with a sparse network and maintaining the same level of connection density throughout the learning procedure to incorporate sparsity into a CL model; it is necessary to disentangle interfering units to prevent forgetting and establish new pathways to encode new knowledge. This presents challenges when implementing batch normalization and residual connections for both the NISPA and CLNP methods. Consequently, these methods do not employ the ResNet-18 architecture. Instead, they opt for a simpler convolutional neural network architecture without ‘skip connections’ and batch normalization.