# A APPENDIX

## A.1 EVOLUTION OF MASK ACROSS GENERATIONS

In this section, we present the evolutionary process of the mask over multiple generations and its impact on the performance and generalization capabilities of the deep neural network (DNN). We evaluate the effectiveness of our proposed method, Selective Knowledge Evolution (SKE), in dynamically adapting and evolving the mask throughout the training process. The ResNet18 architecture with CUB200 is used for this evaluation.

Figure 5 illustrates the evolution of the mask across generations. As the training progresses, the mask undergoes iterative updates based on the data-aware dynamic masking criteria employed by SKE. The mask becomes more refined and selective with each generation, preserving important connections while pruning less relevant ones.

To quantify the evolution of the mask, we measure the overlap percentage of parameters retained between the first and the corresponding generations. We observe a gradual decrease in overlap from the initial generation to subsequent generations, indicating the emergence of masks in an evolutionary training scenario. This progressive mask evolution contributes to the network's enhanced capacity for learning and generalization, evident from the test accuracy.

In conclusion, our results highlight the evolutionary nature of the mask throughout generations in the SKE framework. The dynamic adaptation and refinement of the mask lead to effective masking and improved performance and generalization of the DNN. These findings support the effectiveness of our approach in leveraging the evolutionary training paradigm to enhance the learning and generalization capabilities of deep neural networks compared to KE.

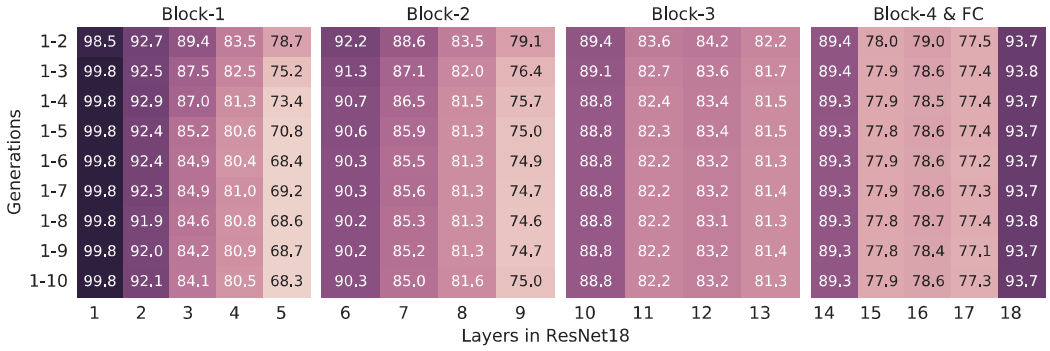


Figure 5: Layer-wise percentage overlap of the retained parameters between first and corresponding generations.

**Algorithm 1** Selective Knowledge Evolution (SKE)

**input:** Train Data $D_t \ \forall \ t \ \in \{1, ..., T\}$, Model $f_\Theta$.
Sparsity factor $k$, learning rate $\eta$, Binary Mask $M$, parameters $\Theta$, Small subset of dataset ($\pi$= $0.2|D_t|$)

1: **for all** Generation $g \in \{1, 2, .., N\}$ **do**
2: $f_g \leftarrow$ Train $f_\Theta$ for e epochs with learning rate $\eta$; ▷ Training step
3: $M \leftarrow$ Importance Estimation$(f_g, \pi, k)$
4: Retain the task specific weights based on $M$ ▷ Knowledge Selection
5: Randomly reinitialize the non-important parameters in $f_g$.
6: Model with this new initialization for next generation training

## A.2 DIFFERENCE WITH TRANSFER LEARNING

Our approach, Selective Knowledge Evolution (SKE), indeed differs significantly from the domain of transfer learning. Unlike transfer learning, which primarily focuses on leveraging pre-trained

Table 4: Comparison of SKE with transfer learning.

| Baselines | CUB | Aircraft | Dog | Flower |
|---|---|---|---|---|
| Smth + transfer learning (f3) | 65.63 $\pm 0.21$ | 61.02 $\pm 0.23$ | 63.84 $\pm 0.17$ | 57.62 $\pm 0.19$ |
| Smth + SKE (f3) | **68.56** $\pm 0.24$ | **64.37** $\pm 0.19$ | **65.72** $\pm 0.15$ | **62.13** $\pm 0.23$ |

models from different domains to boost task performance on downstream tasks, SKE is intricately designed to tackle the intricate challenge of enhancing generalization in the presence of inherently limited or small datasets. A key issue with transfer learning arises when the pre-trained model's source domain significantly differs from the target domain of interest. This discrepancy between domains often leads to domain shifts, where the knowledge transferred from the pre-trained model fails to adapt well to the specificities of the target domain, thereby resulting in suboptimal performance.

In particular, in scenarios like medical applications, obtaining sufficient labeled data that closely aligns with the task at hand is exceptionally challenging. The need for domain expertise, privacy concerns, and the uniqueness of each application domain make it exceedingly difficult to find a pre-trained model that seamlessly fits. This is where the limitations of transfer learning become apparent. Even if a pre-trained model is available, its application might lead to compromised performance due to domain shifts, negatively affecting the accuracy and generalization of the model on a specific task with limited data.

SKE, on the other hand, offers a novel solution to these intricate challenges. By employing data-aware dynamic masking and selective reinitialization, SKE fosters the gradual evolution of the network, enabling it to adapt more effectively to the characteristics of the specific dataset. This process circumvents the problems of domain shifts that often plague transfer learning methods. Thus, while transfer learning remains valuable in contexts with abundant and well-aligned data, SKE stands out as a specialized approach to address the unique hurdles faced in scenarios of limited data availability, where the domain shift problem can severely hinder model performance and generalization.

Furthermore, we have included a comparative analysis in Table 4 involving an instance of transfer learning within the iterative training process. In this particular case, weights are directly transferred from one generation to the next without undergoing reinitialization.

This comparison serves to highlight the unique effectiveness of the Selective Knowledge Evolution (SKE) method. Our results distinctly demonstrate that SKE enhances the process of generalization, showcasing superior performance in comparison to the approach of directly transferring the complete network's weights across generations. This outcome further underscores the distinct advantage of SKE in evolving the network's capacity for better adaptation and learning in the evolutionary training paradigm.

### A.3 Additional comparison with the layerwise reinitialization methods

For a more thorough evaluation, we compare SKE with the layerwise reinitialization methods and provide a detailed comparison to showcase the advantages and uniqueness of our proposed approach.

Zhou et al. (2022) (LLF) propose the forget and relearn hypothesis, which aims to harmonize various existing iterative algorithms by framing them through the lens of forgetting. This approach operates on the premise that initial layers capture generalized features, while subsequent layers tend to memorize specific details. Accordingly, they advocate for the repeated reinitialization and retraining of later layers, effectively erasing information related to challenging instances. Similarly, the LW (Alabdulmohsin et al., 2021) approach progressively reinitializes all layers. Table 5 demonstrates a comparison with these methods.

Notably, SKE showcases comparable, or slightly enhanced performance compared to LW and LLF. Also, these methods (LLF, LW) are underpinned by architecture-specific assumptions that are independent of the data. They rely on the assumed properties that are inherent to the model and its learning. These methods lack a priori knowledge of where and what features, layers, etc. should be reinitialized in general settings. Furthermore, as the model's architecture scales, the complexity of these methods increases accordingly, potentially leading to scalability challenges like which layers to reinitialize. Our proposed method, in contrast, leverages data-aware connection sensitivity

Table 5: Additional comparison with the layerwise reinitialization methods.

| Baselines | CUB | Aircraft | Dog | Flower |
|---|---|---|---|---|
| LW (N8) | 70.50 $\pm 0.26$ | 67.10 $\pm 0.32$ | 65.76 $\pm 0.36$ | 66.92 $\pm 0.20$ |
| LLF (N8) | **71.30** $\pm 0.14$ | **68.87** $\pm 0.12$ | 66.35 $\pm 0.22$ | 67.20 $\pm 0.24$ |
| SKE (N8) | 70.87 $\pm 0.16$ | 66.10 $\pm 0.25$ | **66.56** $\pm 0.18$ | **68.50** $\pm 0.27$ |

through the employment of SNIP, enabling us to select connections for reinitialization dynamically based on their redundancy, contributing to improved generalization.

### A.4 Evaluating the effectiveness of the sparse model

In this section, we assess the effectiveness of the sparse model (containing 20% fewer parameters than the full/dense model) obtained through the selective neurogenesis process during the inference phase. We examine the sparse and dense models' test performance compared to the original KE framework. For this, we measure the performance of the ResNet18 model trained on CUB200. Table 6 presents the accuracy results obtained by the sparse model compared to the dense model. Surprisingly, despite the significant reduction in the number of parameters, the sparse model achieves comparable accuracy compared to the dense model in the SKE framework. Furthermore, SKE demonstrates superior performance in both the full and sparse model scenarios compared to the KE. This indicates that the selective neurogenesis process successfully retains the critical connections necessary for accurate predictions while eliminating redundant or less informative connections. Our evaluation demonstrates that the sparse model obtained through the selective neurogenesis process offers several benefits during inference. It maintains high accuracy while achieving improved computational efficiency compared to the KE. These results highlight the practicality and efficacy of leveraging selective neurogenesis for creating efficient and compact deep learning models that can be readily deployed in real-world scenarios.

Table 6: Evaluating the effectiveness of the sparse model.

| Method | Full model | Sparse model |
|---|---|---|
| KE ($f_{10}$) | 66.51 | 66.21 |
| SKE ($f_{10}$) | **71.37** | **70.08** |

### A.5 Summary of Datasets and implementation details

Taha et al. (2021) employs various image resizing techniques for different datasets; however, they do not provide specific details about the resizing parameters in their paper. To ensure consistency across our experiments, we resize all datasets to a fixed size of (256, 256). Moreover, to fine-tune the hyperparameters, we utilize a validation split, and the reported results are based on the test set whenever it is available.

For experiments on large datasets, we used the following settings. The experiments were conducted on three different datasets: CIFAR-10/100, Tiny-ImageNet. For CIFAR-10/100, the training was performed for 160 epochs. A batch size of 64 was used, along with a step-based learning rate scheduler. The learning rate decay was applied between epochs 80 and 120, with a decay factor of 10. The momentum was set to 0.9, and l2 regularization was applied with a coefficient of 5e-4. Initial learning rate used was 0.1. There were no warmup epochs in this case.

For the Tiny-ImageNet dataset, the training was also conducted for 160 epochs. The batch size was reduced to 32, and a step-based learning rate scheduler was used. Similar to CIFAR-10/100, the learning rate decay occurred between epochs 80 and 120, with a decay factor of 10. The momentum and l2 regularization were set to 0.9 and 5e-4, respectively. Additionally, 20 warmup epochs were applied. Throughout all experiments, a resetting ratio of 20% is used for all generation. All the training and evaluation is done on NVIDIA RTX-2080 Ti GPU. The time required to approximately train 10 generation of SKE on CUB200 with ResNet18 is approximately 1.68 hours. It's worth mentioning that for comparing our method with other baselines, we utilized the results presented in

the KE paper (Taha et al., 2021) as a point of reference. For the hyperparameters used in training small datasets, please refer to Section 4.

Table 7: shows the statistics of five classification datasets.

| Datasets | Classes | Train | Validation | Test | Total |
|---|---|---|---|---|---|
| CUB-200 (Wah et al., 2011) | 200 | 5994 | N/A | 5794 | 11788 |
| Flower-102 (Nilsback & Zisserman, 2008) | 102 | 1020 | 1020 | 6149 | 8189 |
| Aircraft (Maji et al., 2013) | 100 | 3334 | 3333 | 3333 | 10000 |
| Standford-Dogs (Khosla et al., 2011) | 120 | 12000 | N/A | 8580 | 20580 |