Architecture Matters in Continual Learning

Seyed Iman Mirzadeh*[1], Arslan Chaudhry[2], Dong Yin[2],
Timothy Nguyen[2], Razvan Pascanu[2], Dilan Gorur[2], and Mehrdad Farajtabar†[2]

[1]Washington State University, [2]DeepMind

Abstract

A large body of research in continual learning is devoted to overcoming the catastrophic forgetting of neural networks by designing new algorithms that are robust to the distribution shifts. However, the majority of these works are strictly focused on the "algorithmic" part of continual learning for a "fixed neural network architecture", and the implications of using different architectures are mostly neglected. Even the few existing continual learning methods that modify the model assume a fixed architecture and aim to develop an algorithm that efficiently uses the model throughout the learning experience. However, in this work, we show that the choice of architecture can significantly impact the continual learning performance, and different architectures lead to different trade-offs between the ability to remember previous tasks and learning new ones. Moreover, we study the impact of various architectural decisions, and our findings entail best practices and recommendations that can improve the continual learning performance.

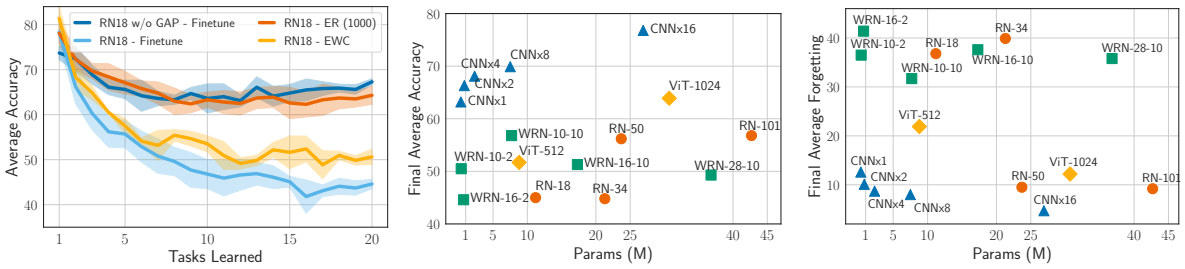
1 Introduction

Continual learning (CL) (Ring, 1995; Thrun, 1995) is a branch of machine learning where the model is exposed to a sequence of tasks with the hope of exploiting existing knowledge to adapt quickly to new tasks. The research in continual learning has seen a surge in the past few years with the explicit focus of developing algorithms that can alleviate *catastrophic forgetting* (McCloskey and Cohen, 1989)—whereby the model abruptly forgets the information of the past when trained on new tasks.

While most of the research in continual learning is focused on developing *learning algorithms*, that can perform better than naive fine-tuning on a stream of data, the role of model architecture, to the best of our knowledge, is not explicitly studied in any of the existing works. Even the class of parameter isolation or expansion-based methods, for example (Rusu et al., 2016; Yoon et al., 2018), have a cursory focus on the model architecture insofar that they assume a specific architecture and try to find an algorithm operating on the architecture. Orthogonal to this direction for designing algorithms, our motivation is that the inductive biases induced by different architectural components are important for continual learning. We seek to characterize the implication of different architectural choices.

To motivate, consider a ResNet-18 model (He et al., 2016) on Split CIFAR-100, where CIFAR-100 dataset (Krizhevsky et al., 2009) is split into 20 disjoint sets—a prevalent architecture and benchmark in the existing continual learning works. Fig. 1a shows that explicitly designed CL algorithms, EWC (Kirkpatrick et al., 2017) (a parameter regularization-based method) and experience replay (Riemer et al.,

*Work done during an internship at DeepMind.
†Correspondence to farajtabar@google.com



(a) (b) (c)

Figure 1: Split CIFAR-100: (a) While compared to naive fine-tuning, continual learning algorithms such as EWC and ER improve the performance, a simple modification to the architecture (removing global average pooling (GAP) layer) can match the performance of ER with a replay size of 1000 examples. (b) and (c) Different architectures lead to very different continual learning performance levels in terms of accuracy and forgetting. This work will investigate the reasons behind these gaps and provide insights into improving architectures.

2018) (a memory-based CL algorithm) indeed improve upon the naive fine-tuning. However, one can see that similar or better performance can be obtained on this benchmark by simply removing the global average pooling layer from ResNet-18 and performing the naive fine-tuning. This clearly demonstrates the need for a better understanding of network architectures in continual learning.

Briefly, in this work, we thoroughly study the implications of architectural decisions in continual learning. Our experiments suggest that different components of modern neural networks have different effects on the relevant continual learning metrics—namely average accuracy, forgetting, and learning accuracy (*cf.* Sec. 2.1.2)—to the extent that vanilla fine-tuning with modified components can achieve similar of better performance than specifically designed CL methods on a base architecture without significantly increasing the parameters count.

We summarize our main contributions as follows:

- We compare both the learning and retention capabilities of popular architectures. To the best of our knowledge, the significance of architecture in continual learning has not been explored before.
- We study the role of individual architectural decisions (e.g., width and depth, batch normalization, skip-connections, and pooling layers) and how they can impact the continual learning performance.
- We show that, in some cases, simply modifying the architecture can achieve a similar or better performance compared to specifically designed CL algorithms (on top of a base architecture).
- In addition to the standard CL benchmarks, Rotated MNIST and Split CIFAR-100, we report results on the large-scale Split ImageNet-1K benchmark, which is rarely used in the CL literature, to make sure our results hold in more complex settings.
- Inspired by our findings, we provide practical suggestions that are computationally cheap and can improve the performance of various architectures in continual learning.

We emphasize that the main objective of this work is to illustrate the significance of architectural decisions in continual learning, and this does not imply that the algorithmic side is not essential. In fact, one can enjoy the improvements on both sides, as we will discuss in Appendix B. Finally, we note that the secondary aim of this work is to be a stepping-stone that encourages further research on the architecture side of continual learning by focusing on the *breadth* rather than *depth* of some topics in this work. We believe our work provides many interesting directions that require deeper analysis beyond the scope of this paper but can significantly improve our understanding of continual learning.

2 Comparing Architectures

2.1 Experimental Setup

Here, for brevity, we explain our experimental setup but postpone more detailed information (e.g., hyper-parameters, details of architectures, the justification behind our experimental setup choices, etc.) to Appendix A.

2.1.1 Continual Learning Setup

We use three continual learning benchmarks for our experiments. The Split CIFAR-100 includes 20 tasks where each task has the data of 5 classes (disjoint), and we train on each task for 10 epochs. The Split ImageNet-1K includes 10 tasks where each task includes 100 classes of ImageNet-1K and we train on each task for 60 epochs. Finally, for a few experiments, we use the small Rotated MNIST benchmark with 5 tasks where the first task is the standard MNIST dataset, and each of the subsequent tasks adds 22.5 degrees of rotation to the images of the previous task. The rationale behind our MNIST setup is that by increasing the rotation degrees, the shift across tasks increases and makes the benchmark more challenging (Mirzadeh et al., 2021b). We note that the Split CIFAR-100 and Split ImageNet-1K benchmarks use a multi-head classification layer, while the MNIST benchmark uses a single-head classification layer. Thus, Split CIFAR-100 and Split ImageNet-1K belong to the so-called *task incremental learning* setting, whereas Rotated MNIST belongs to *domain incremental learning* (Hsu et al., 2018).

We work with the most common architectures in the literature (continual learning or otherwise). We denote each architecture with a descriptor. MLP-N represents fully connected networks with hidden layers of width N. Convolutions neural networks (CNN) are represented by CNN$\times$N where N is the multiplier of the number of channels in each layer. Unless otherwise stated, the CNNs have only convolutional layers (with a stride of 2), followed by a densely feed-forward layer for classification. For the CIFAR-100 experiments, we use three convolutional layers, and for the ImageNet-1K experiments, we use six convolutional layers. Moreover, whenever we add pooling layers, we change the convolutional layer strides to 1 to keep the dimension of features the same. The standard ResNet (He et al., 2016) of depth D is denoted by ResNet-D and WideResNets (WRN) (Zagoruyko and Komodakis, 2016) are denoted by WRN-D-N where D and N are the depths and widths, respectively. Finally, we also use the recently proposed Vision Transformers (ViT) (Dosovitskiy et al., 2021). For the ImageNet-1K experiments, we follow the naming convention in the original paper (Dosovitskiy et al., 2021). However, for the Split CIFAR-100 experiments, we use smaller versions of ViTs where ViT N/M stands for a 4-layer vision transformer with the hidden size of N and MLP size of M.

For *each architecture*, we search over a large grid of hyper-parameters (refer to Appendix A) and report the best results. Further, we average the results over 5 different random initializations, for the corresponding best hyper-parameters, and report the average and standard deviations. Finally, for Split CIFAR-100 and Split ImageNet-1K benchmarks, we randomly shuffle the labels in each run, for 5 runs, to ensure that the results are not biased towards a specific label ordering.

2.1.2 Metrics

We are interested in comparing different architectures from two aspects: (1) how well an architecture can learn a new task *i.e.* their *learning ability* and (2) how well an architecture can preserve the previous knowledge *i.e.* their *retention ability*. For the former, we record *average accuracy*, *learning accuracy*, and *joint/ multi-task accuracy*, while for the latter we measure the *average forgetting* of the model. We now

define these metrics.
(1) *Average Accuracy* $\in [0, 100]$ (the higher the better): The average validation accuracy after the model has been continually trained for T tasks is defined as: $A_T = \frac{1}{T}\sum_{i=1}^{T} a_{T,i}$, where, $a_{t,i}$ is the validation accuracy on the dataset of task i after the model finished learning task t.
(2) *Learning Accuracy* $\in [0, 100]$ (the higher the better): The accuracy for each task directly after it is learned. The learning accuracy provides a good representation of the plasticity of a model and can be calculated using: $\text{LA}_T = \frac{1}{T}\sum_{i=1}^{T} a_{i,i}$. Note that for both Split CIFAR-100 and ImageNet-1K benchmarks, since tasks include images with disjoint labels, the standard deviation of this metric can be high. Moreover, since all models are trained from scratch, the learning accuracy of the first few tasks is usually smaller compared to later tasks.
(3) *Joint Accuracy* $\in [0, 100]$ (the higher the better): The accuracy of the model when trained on the data of all tasks together.
(4) *Average Forgetting* $\in [-100, 100]$ (the lower the better): The average forgetting is calculated as the difference between the peak accuracy and the final accuracy of each task, after the continual learning experience is finished. For a continual learning benchmark with T tasks, it is defined as: $F = \frac{1}{T-1}\sum_{i=1}^{T-1} \max_{t\in\{1,\ldots,T-1\}} (a_{t,i} - a_{T,i})$.

2.2 Results

We first compare different architectures on the Split CIFAR-100 and Split ImageNet-1K benchmarks. While this section broadly focuses on the learning and retention capabilities of different architectures, the explanations behind the performance gaps across different architectures and the analysis of different architectural components is given in the next section.

Tab. 1 lists the performance of different architectures on Split CIFAR-100 benchmark. One can make several observations from the table. First, very simple CNNs, which are not state-of-the-art in the single image classification tasks, significantly outperform both the ResNets, WRNs, and ViTs in terms of average accuracy and forgetting. This observation holds true for various sizes of widths and depths in all the architectures. A similar overall trend, where for a given parameter count, simple CNNs outperform other architectures, can also be seen in Fig. 1b and 1c.

Second, a mere increase in the parameters count, within or across the architectures, does not necessarily translate into the performance increase in continual learning. For instance, ResNet-18 and ResNet-34 have roughly the same performance despite almost twice the number of parameters in the latter. Similarly, WRN-10-10 outperforms WRN-16-10 and WRN 28-10 despite having significantly less number of parameters. Note that we do not draw a general principle that overparametrization is not helpful in continual learning. In fact, in some cases, it indeed is helpful as can be in the across-the-board performance improvement when ResNet-34 is compared with ResNet-50 or when the WRN-10-2 is compared to the WRN-10-10. In the next section, we analyze when overparametrization can help the performance in continual learning.

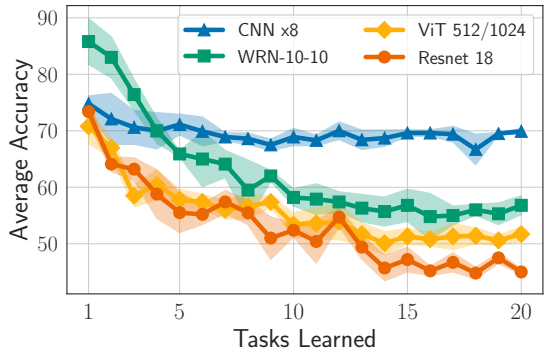
Finally, explicitly comparing the learning and retention capabilities of different architectures, one can see from the table that ResNets and WRNs have a higher learning accuracy suggesting that they are better at learning a new task. This also explains their frequent use in single task settings. However, in terms of retention, CNNs and ViTs are much better, as evidenced by their lower forgetting numbers. This is further demonstrated in Fig. 2a (CIFAR-100) and Fig. 2b (ImageNet-1K), where it can be seen that ResNets and WRNs learn each individual task much better resulting in a higher average accuracy for the first few tasks. However, as the number of tasks increases, CNNs outperforms the other architectures due to their smaller forgetting, eventually translating into an overall flatter average accuracy curve.

Table 1: Split CIFAR-100: the learning and retention capabilities can vary significantly across different architectures.

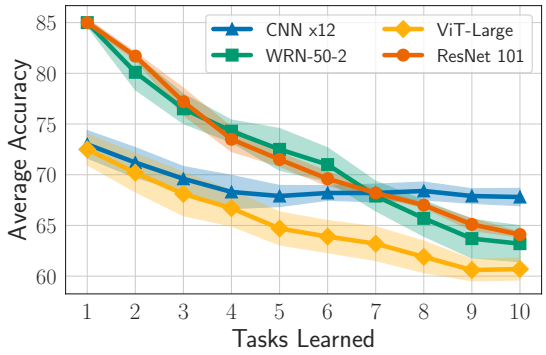
Model	Params (M)	Average Accuracy	Average Forgetting	Learning Accuracy
CNN x1	0.3	62.2 ±1.35	12.6 ±1.14	74.1±7.72
CNN x2	0.8	66.3 ±1.12	10.1 ±0.98	75.8 ±7.2
CNN x4	2.3	68.1 ±0.5	8.7 ±0.21	76.4 ±6.92
CNN x8	7.5	69.9 ±0.62	8.0 ±0.71	77.5 ±6.78
CNN x16	26.9	76.8 ±0.76	4.7 ±0.84	81.0 ±6.97
ResNet-18	11.2	45.0 ±0.63	36.8 ±1.08	74.9 ±3.98
ResNet-34	21.3	44.8 ±2.34	39.9 ±2.28	72.6 ±6.34
ResNet-50	23.6	56.2 ±0.88	9.5 ±0.38	67.8 ±5.09
ResNet-101	42.6	56.8 ±1.62	9.2 ±0.89	65.7±5.42
WRN-10-2	0.3	50.5 ±2.65	36.5 ±2.74	84.5 ±5.04
WRN-10-10	7.7	56.8 ±2.03	31.7 ±1.34	86.7 ±4.94
WRN-16-2	0.7	44.6 ±2.81	41.4 ±1.43	82.4 ±6.09
WRN-16-10	17.3	51.3 ±1.47	37.6 ±2.22	86.9 ±3.96
WRN-28-2	5.9	46.6 ±2.27	39.5 ±2.29	82.5 ±6.26
WRN-28-10	36.7	49.3 ±2.02	35.8 ±2.56	82.5 ±6.26
ViT-512/1024	8.8	51.7 ±1.4	21.9 ±1.3	71.4 ±5.52
ViT-1024/1546	30.7	60.4 ±1.56	12.2 ±1.12	67.4 ±5.57

Table 2: Split ImageNet-1K: the learning and retention capabilities can vary significantly across different architectures.

Model	Params (M)	Average Accuracy	Average Forgetting	Learning Accuracy
CNN x3	9.1	63.3 ±0.68	5.4 ±0.93	71.6 ±2.31
CNN x6	24.2	66.7 ±0.62	3.9 ±0.86	70.1 ±3.21
CNN x12	72.4	67.8 ±1.04	2.8 ±0.7	70.3 ±2.82
ResNet-34	21.8	62.7 ±0.53	17.3 ±0.58	78.4 ±2.57
ResNet-50	25.5	66.1 ±0.69	19.0 ±0.67	83.3 ±1.57
ResNet-101	44.5	64.1 ±0.72	18.9 ±1.32	81.1 ±2.89
WRN-50-2	68.9	63.2 ±1.61	21.7 ±1.73	85.8 ±1.65
ViT-Base	86.1	58.3 ±0.65	15.9 ±1.11	72.8 ±2.25
ViT-Large	307.4	60.7 ±1.31	10.6 ±1.1	73.2 ±2.12



(a) Split CIFAR-100



(b) Split ImageNet-1K

Figure 2: Evolution of average accuracy for various architectures on (a) Split CIFAR-100: CNNs have smaller forgetting than other architectures while WideResNets have the highest learning accuracy, and (b) Split ImageNet-1K WideResNets and ResNets have higher learning accuracy than CNNs and ViTs. However, the latter has smaller forgetting.

A trend similar to CIFAR-100 can also be seen in the ImageNet-1K benchmark as shown in Table 2. However, the performance difference, as measured by the average accuracy, between CNNs and other architectures is smaller compared to that of CIFAR-100. We believe that this is due to the very high learning accuracy of other architectures compared to CNNs on this benchmark, and hence their frequent use in the single task settings, resulting in an improved final average accuracy. The average forgetting of CNNs is still much smaller than other architectures.

Overall, from both tables, we conclude that ResNets and WRNs have better learning abilities whereas CNNs and ViTs have better retention abilities. In our experiments, simple CNNs achieve the best trade-off between learning and retention.

3 Role of Architecture Components

We now study the individual components in various architectures to understand how they impact continual learning performance. We start by generic *structural* properties in all architectures such as *width* and *depth* (*cf.* Sec. 3.1), and show that as the width increases, the forgetting decreases. In Sec. 3.2, we study the impact of batch normalization and observe that it can significantly improve the learning accuracy in continual learning. Then, in Sec. 3.3, we see that adding skip connections (or shortcuts) to CNNs does not necessarily improve the CL performance whereas pooling layers (*cf.* Sec. 3.4 and Sec. 3.5) can have significant impact on learning accuracy and forgetting. Moreover, we briefly study the impact of attention heads in ViTs in Sec. 3.6. Finally, based on the observations we make in the aforementioned sections, in Sec. 3.7, we provide a summary of modifications that can improve various architectures on both Split CIFAR-100 and ImageNet-1K benchmarks[1].

3.1 Width and Depth

Recently, Mirzadeh et al. (2021a) have shown that wide neural networks forget less catastrophically by studying the impact of width and depth in MLP and WideResNet models. We extend their study by confirming their conclusions on more architectures and benchmarks.

Table 3: Role of width and depth: increasing the number of parameters (by increasing width) reduces the forgetting and hence increases the average accuracy. However, increasing the depth does not necessarily improve the performance, and thus, it is essential to distinguish between scaling the models by making them deeper and wider.

Benchmark	**Model**	**Depth**	**Params (M)**	**Average Accuracy**	**Average Forgetting**	**Learning Accuracy**
Rot MNIST	MLP-128	2	0.1	70.8 ±0.68	31.5 ±0.92	96.0 ±0.90
Rot MNIST	MLP-128	8	0.2	68.9 ±1.07	35.4 ±1.34	97.3 ±0.76
Rot MNIST	MLP-256	2	0.3	71.1 ±0.43	31.4 ±0.48	96.1 ±0.82
Rot MNIST	MLP-256	8	0.7	70.4 ±0.61	32.1 ±0.75	96.3 ±0.77
Rot MNIST	MLP-512	2	0.7	72.6 ±0.27	29.6 ±0.36	96.4 ±0.73
CIFAR-100	CNN x4	3	2.3	68.1 ±0.5	8.7 ±0.21	76.4 ±6.92
CIFAR-100	CNN x4	6	5.4	62.9 ±0.86	12.4 ±1.62	77.7 ±5.49
CIFAR-100	CNN x8	3	7.5	69.9 ±0.62	8.0 ±0.71	77.5 ±6.78
CIFAR-100	CNN x8	6	19.9	66.5 ± 1.01	10.7 ±1.19	76.6 ±4.78
CIFAR-100	ViT 512/1024	2	4.6	56.4 ±1.14	15.9 ±0.95	68.1 ±7.15
CIFAR-100	ViT 512/1024	4	8.8	51.7 ±1.4	21.9 ±1.3	71.4 ±5.52

Tab. 3 shows that across all architectures, over-parametrization through increasing width is helpful in improving the continual learning performance as evidenced by lower forgetting and higher average accuracy numbers. For MLP, when the width is increased from 128 to 512, the performance in all measures improves. However, for both MLP-128 and MLP-256 when the depth is increased from 2 to 8 the average accuracy is reduced, and the average forgetting is increased with a marginal gain in learning accuracy. Finally, note that MLP-256 with 8 layers has roughly the same number of parameters as the MLP-512 with 2 layers. However, the wider one of these two networks has a better continual learning performance.

A similar analysis for ResNets and WideResNets is demonstrated in Tab. 1. ResNet-50 and ResNet-101 are four times wider than ResNet-18 and ResNet-34, and from the table, it can be seen that this

[1]In this section, we duplicate some of the results across tables to improve readability.

width translates into drastic improvements in average accuracy and forgetting. Similarly, ResNet-34 and ResNet-101 are the deeper versions of ResNet-18 and ResNet-50, respectively. We can observe that increasing the depth is not helpful in this case. Finally, wider WRN-10-10, WRN-16-10, and WRN-28-10 outperform the narrower WRN-10-2, WRN-10-10, WRN-28-10, respectively. Whereas if we fix the width, increasing the depth is not helpful. Overall, we conclude that over-parametrization through width is helpful in continual learning, whereas a similar claim cannot be made for the depth.

A possible explanation for this is through the lazy-training regime, gradient orthogonalization, and sparsification induced by increasing the width (Mirzadeh et al., 2021a). Motivated by this observation, later, we show why global average pooling layers hurt the continual learning performance and how we can alleviate that.

3.2 Batch Normalization

Batch Normalization (BN) (Ioffe and Szegedy, 2015) is a normalization scheme that is shown to increase the convergence speed of the network due to its optimization and generalization benefits (Santurkar et al., 2018; Bjorck et al., 2018). Another advantage of the BN layer is its ability to reduce the covariate shift problem that is specifically relevant for continual learning where the data distribution may change from one task to the next.

There are relatively few works that have studied the BN in the context of continual learning. Mirzadeh et al. (2020) analyzed the BN in continual learning through the generalization lens. Concurrently to this work, Pham et al. (2022) study the normalization schemes in continual learning and show that BN enables improved learning of each task. Additionally, the authors showed that in the presence of a replay buffer of previous tasks, BN facilitates a better knowledge transfer compared to other normalization schemes such as Group Normalization (Wu and He, 2018).

Intuitively, however, one might think that since the BN statistics are changing across tasks, due to evolving data distribution in continual learning, and one is not keeping the statistics of each task, the BN should contribute to an increased forgetting. This is not the case in some of the experiments that we conducted. Similar to the results in Mirzadeh et al. (2020); Pham et al. (2022), we found the BN to facilitate the learning accuracy in Split CIFAR-100 and split ImageNet-1K (*cf.* Tab. 4 and Tab. 7). We believe that this could be due to relatively unchanging BN statistics across tasks in these datasets. To verify this, in Appendix B.2, we plot the BN statistics of the first layer of CNN$\times$4 on the Split CIFAR-100 dataset, and we show that the BN statistics are stable throughout the continual learning experience. However, if this hypothesis is true, the converse – a benchmark where the BN statistics change a lot across tasks, such as Permuted MNIST – should hurt the continual learning performance. In Appendix B.3, we plot the BN statistics of the first layer of MLP-128 on Permuted MNIST. It can be seen from the figure that indeed the BN statistics are changing in this benchmark. As a consequence, adding BN to this benchmark significantly hurt the performance, as evidenced by the increased forgetting in Tab. 9.

Overall, we conclude that the effect of the batchnorm layer is data-dependent. In the setups where the input distribution relatively stays stable, such as Split CIFAR-100 or Split ImageNet-1K, the BN layer improves the continual learning performance by increasing the learning capability of the models. However, for setups where the input distribution changes a lot across tasks, such as Permuted MNIST, the BN layer can hurt the continual learning performance by increasing the forgetting.

Table 4: Role of various components for the Split CIFAR-100 benchmark: While adding skip connections does not have a significant impact on the performance, batch normalization and max polling can significantly improve the learning accuracy of CNNs.

Model	Params (M)	Average Accuracy	Average Forgetting	Learning Accuracy	Joint Accuracy
CNN x4	2.3	68.1 ±0.5	8.7 ±0.21	76.4 ±6.92	73.4 ±0.89
CNN x4 + Skip	2.4	68.2 ±0.56	8.9 ±0.72	76.6 ±7.07	73.8 ±0.47
CNN x4 + BN	2.3	74.0 ±0.56	8.1 ±0.35	81.7 ±6.68	80.2 ±0.16
CNN x4 + AvgPool	2.3	68.5 ±0.6	8.3 ±0.57	76.3 ±7.63	73.6 ±0.83
CNN x4 + MaxPool	2.3	74.4 ±0.34	9.3 ±0.47	83.3 ±6.1	79.9 ±0.53
CNN x4 + All	2.4	77.7 ±0.77	6.5 ±0.58	83.7 ±6.31	81.6 ±0.77
CNN x8	7.5	69.9 ±0.62	8.0 ±0.71	77.5 ±6.78	74.1 ±0.83
CNN x8 + Skip	7.8	70.7 ±0.31	6.8 ±0.91	77.1 ±6.87	74.4 ±0.35
CNN x8 + BN	7.5	76.1 ±0.3	5.9 ±0.16	81.7 ±6.83	80.5 ±0.27
CNN x8 + AvgPool	7.5	71.2 ±0.5	8.3 ±0.35	79.0 ±7.05	74.0 ±1.02
CNN x8 + MaxPool	7.5	77.2 ±0.53	7.1 ±0.33	84.0 ±5.81	80.6 ±0.35
CNN x8 + All	7.8	78.1 ±1.15	5.7 ±0.36	83.3 ±6.27	81.9 ±0.51
CNN x16	26.9	76.8 ±0.76	4.7 ±0.84	81.0 ±6.97	79.1 ±0.86
CNN x16 + All	27.9	78.9 ±0.27	4.5 ±0.36	82.9 ±6.48	82.1 ±0.46

3.3 Skip Connections

Skip connections (Cho et al., 2011), originally proposed for convolutional models by He et al. (2016), are crucial in the widely used ResNet architecture. They are also used in many other architectures such as transformers (Vaswani et al., 2017). Many works have been done to explain why skip connections are useful: Hardt and Ma (2016) show that skip connection tends to eliminate spurious local optima; Bartlett et al. (2018b) study the function expressivity of residual architectures; Bartlett et al. (2018a) show that gradient descent provably learns linear transforms in ResNets; Jastrzebski et al. (2017) show that skip connections tend to iteratively refine the learned representations. However, these works mainly focus on learning a single task. In continual learning problems, due to the presence of distribution shift, it is unclear whether these benefits of skip connections still have a significant impact on model performance, such as forgetting and average accuracy over tasks.

Here, we empirically study the impact of skip connection on continual learning problems. Interestingly, as illustrated in Tab. 4, adding skip connections to plain CNNs does not change the performance significantly, and the results are very close (within the standard deviation) of each other. Therefore, we conclude that skip connection may not have a significant positive or negative impact on the model performance in our continual learning benchmarks.

3.4 Pooling Layers

Pooling layers were the mainstay of the improved performance of CNNs before ResNets. Pooling layers not only add local translation invariances, which help in applications like object classification (Krizhevsky et al., 2017; Dai et al., 2021), but also reduce the spatial resolution of the network features, resulting in the reduction of computational cost. Since one family of the architectures that we study are all-convolutional CNNs, we revisit the role of pooling layers in these architectures in a continual learning setup. Towards this, we compare the network without pooling, CNN×N , against those that have pooling layers ('CNN×N +AvgPool' or 'CNN×N +MaxPool') in Tab. 4. To keep the feature dimensions fixed, we set the convolutional stride from 2 to 1 when pooling is used. We make the following observations from the table.

First, the average pooling (+AvgPool) does not have any significant impact on the continual learning metrics. Second, max pooling (+MaxPool) improves the learning capability of the network significantly, as measured by the improved learning accuracy. Third, in terms of retention, pooling layers do not have a significant impact as measured by similar forgetting numbers. All in all, we see that max pooling achieves the best performance in terms of average accuracy, owing to its superior learning capability.

The ability of max pooling to achieve better performance in a continual learning setting can be attributed to a well-known empirical observation by Springenberg et al. (2015), where it is shown that max pooling with stride 1 outperforms a CNN with stride 2 and no pooling. Further, we believe that max pooling might have extracted the low-level features, such as edges, better, resulting in improved learning in a dataset like CIFAR-100 that consists of natural images. There is some evidence in the literature that max pooling provides sparser features and precise localization (Zhou et al., 2016). This could have transferred over to the continual learning setup that we considered. It, however, remains an interesting future direction to further study the gains brought by max pooling in both the standard and continual learning setups.

3.5 Global Pooling Layers

Global average pooling (GAP) layers are typically used in convolutional networks just before the final classification layer to reduce the number of parameters in the classifier. The consequence of adding a GAP layer is to reduce the width of the final classifier. It is argued in Sec. 3.1 that wider networks forget less. Consequently, the architectures with a GAP layer can suffer from increased forgetting.

Tab. 5 empirically demonstrate this intuition. From the table it can be seen that applying the GAP layer significantly increases the forgetting resulting in a lower average accuracy. In the previous section, we already demonstrated that average pooling does not result in a performance decrease as long as the spatial feature dimensions are the same. To demonstrate that there is nothing inherent to the GAP layer, and it is just a consequence of a reduced width of the final classifier, we construct another baseline by multiplying the number of channels in the last convolutional layer by 16 and then apply the GAP. This network is denoted as "CNN x4 (16x)" in Tab. 5 and it has the same classifier width as that of the network without GAP. It can be seen this architecture has considerably smaller forgetting showing GAP affects continual learning through the width of the final classifier.

Table 5: Role of Global Average Pooling (GAP) for Split CIFAR-100: related to our arguments in Sec. 3.1, adding GAP to CNNs significantly increases the forgetting. Later, we show that removing GAP from ResNets can also significantly reduce forgetting as well.

Model	**Params (M)**	**Pre-Classification Width**	**Average Accuracy**	**Average Forgetting**	**Learning Accuracy**	**Joint Accuracy**
CNN x4	2.3	**8192**	68.1 ±0.5	8.7 ±0.21	76.4 ±6.92	73.4 ±0.89
CNN x4 + GAP	1.5	**512**	60.1 ±0.43	14.3 ±0.8	66.1 ±7.76	76.9 ±0.81
CNN x4 (16x) + GAP	32.3	**8192**	73.6 ±0.39	5.2 ±0.66	75.6 ±4.77	77.9 ±0.37
CNN x8	7.5	**16384**	69.9 ±0.62	8.0 ±0.71	77.5 ±6.78	74.1 ±0.83
CNN x8 + GAP	6.1	**1024**	63.1 ±2.0	14.7 ±1.68	70.1 ±7.18	78.3 ±0.97
CNN x16	26.9	**32768**	76.8 ±0.76	4.7 ±0.84	81.0 ±6.97	74.6 ±0.86
CNN x16 + GAP	23.8	**2048**	66.3 ±0.82	12.2 ±0.65	72.3 ±6.02	78.9 ±0.27

Inspired by this observation, in Sec. 3.7 we show that the performance of ResNets in continual learning can be significantly improved by either removing the GAP layer or using the smaller average pooling layers rather than the GAP.

3.6 Attention Heads

The attention mechanism is a prominent component in the transformer-based architectures that has had great success in natural language processing and, more recently, computer vision tasks. For the latter, the heads of vision transformers (ViTs) are shown to attend to both local and global features in the image Dosovitskiy et al. (2021). We double the number of heads while fixing the width to ensure that any change in results is not due to the increased dimension of representations. In Tab. 6, it can be seen that even doubling the number of heads in ViTs only marginally helps in increasing the learning accuracy and lowering the forgetting. This suggests that increasing the number of attention heads may not be an efficient approach for improving continual learning performance in ViTs. We, however, note that consistent with other observations in the literature (Paul and Chen, 2022), ViTs show promising robustness against distributional shifts as evidenced by lower forgetting numbers, for the same amount of parameters, on the Split CIFAR-100 benchmark (*cf.* Fig. 1c).

Table 6: Role of attention heads: for the Split CIFAR-100 benchmark, increasing the number of attention heads (while fixing the total width), does not impact the performance significantly.

Model	**Heads**	**Params (M)**	**Average Accuracy**	**Average Forgetting**	**Learning Accuracy**
ViT 512/1024	4	8.8	50.9 ±0.73	23.8 ±1.3	72.8 ±6.13
ViT 512/1024	8	8.8	51.7 ±1.4	21.9 ±1.3	71.4 ±5.52
ViT 1024/1536	4	30.7	57.4 ±1.59	14.4 ±1.96	66.0 ±5.89
ViT 1024/1536	8	30.7	60.4 ±1.56	12.2 ±1.12	67.4 ±5.57

3.7 Improving Architectures

We now provide practical suggestions to improve the performance of architectures, that are derived from our experiments in the previous sections.

For CNNs, we add BatchNormalization and MaxPooling, as they both are shown to improve the learning ability of the model and skip connections that make the optimization problem easier. Tab. 7 shows the results on both CIFAR-100 and ImageNet-1K. It can be seen from the table that adding these components significantly improves the CNNs performance, as evidenced by improvement over almost all the measures, including average accuracy.

For ResNets, we either remove the GAP, as is the case with CIFAR-100, or locally average the features in the penultimate layer by a 4x4 filter, as is the case with ImageNet-1K. The reason for not fully removing the average pooling from the ImageNet-1K is the resultant large increase in the number of parameters in the classifier layer. It can be seen from Tab. 7 that fully or partially removing the GAP, CIFAR-100, and ImageNet-1K, respectively, highly improves the retention capability of ResNets, indicated by their lower forgetting.

4 Related Work

As we have discussed before, our work focuses on *architectures* in continual learning rather than the *algorithmic* side. Hence, we start with a brief overview of continual learning algorithms, and then we focus mainly on the part of the literature that is more related to the architecture.

Table 7: Improving CNNs and ResNets architectures on both Split CIFAR-100 and ImageNet-1K benchmarks.

Model	Benchmark	Params (M)	Average Accuracy	Average Forgetting	Learning Accuracy
CNN x4	CIFAR-100	2.3	68.1 ±0.5	8.7 ±0.21	76.4 ±6.92
CNN x4 + BN + MaxPool + Skip	CIFAR-100	1.5	77.7 ±0.77	6.5 ±0.58	83.7 ±6.31
CNN x8	CIFAR-100	7.5	69.9 ±0.62	8.0 ±0.71	77.5 ±6.78
CNN x8 + BN + MaxPool + Skip	CIFAR-100	6.1	78.1 ±1.15	5.7 ±0.36	83.3 ±6.27
CNN x3	ImageNet-1K	9.1	63.3 ±0.68	5.8 ±0.93	71.6 ±2.31
CNN x3 + BN + MaxPool + Skip	ImageNet-1K	9.1	66.4 ±0.47	5.4 ±0.3	74.7 ±2.1
CNN x6	ImageNet-1K	24.2	66.7 ±0.62	3.9 ±0.86	70.1 ±3.21
CNN x6 + BN + MaxPool + Skip	ImageNet-1K	24.3	72.1 ±0.41	4.0 ±0.22	75.7 ±2.57
ResNet-18	CIFAR-100	11.2	45.0 ±0.63	36.8 ±1.08	74.9 ±3.98
ResNet-18 w/o GAP	CIFAR-100	11.9	67.4 ±0.76	11.2 ±1.98	74.2 ±4.79
ResNet-50	CIFAR-100	23.6	56.2 ±0.88	9.5 ±0.38	67.8 ±5.09
ResNet-50 w/o GAP	CIFAR-100	26.7	71.4 ±0.29	6.6 ±0.12	73.0 ±5.18
ResNet-34	ImageNet-1K	21.8	62.7 ±0.53	19.0 ±0.67	80.4 ±2.57
ResNet-34 w 4x4 AvgPool	ImageNet-1K	23.3	66.0 ±0.24	4.2 ±0.16	70.2 ±3.87
ResNet-50	ImageNet-1K	25.5	66.1 ±0.69	17.3 ±0.58	83.3 ±1.57
ResNet-50 w 4x4 AvgPool	ImageNet-1K	31.7	67.2 ±0.13	3.5 ±0.35	72.8 ±3.27

4.1 Algorithms

On the algorithmic side of research, various methods have been proposed to alleviate the forgetting problem. (Zenke et al., 2017; Kirkpatrick et al., 2017; Mirzadeh et al., 2021b; Yin et al., 2020) identify essential knowledge from the past, often in the form of parameters or features, and modify the training objective of the new task to preserve the learned knowledge. However, in practice, **replay** methods seem to be much more effective as they keep a small subset of previously seen data and either uses them directly when learning from new data (e.g., experience replay) (Chaudhry et al., 2019; Rolnick et al., 2019; Riemer et al., 2018) or incorporate them as part of the optimization process (e.g., gradient projection) (Farajtabar et al., 2020; Chaudhry et al., 2018; Balaji et al., 2020), or learn a generative model (Shin et al., 2017; Kirichenko et al., 2021) or kernel (Derakhshani et al., 2021) to do so.

Perhaps the most related side of algorithms to our work is the **parameter-isolation** methods where different parts of the model are devoted to a specific task (Yoon et al., 2018; Mallya and Lazebnik, 2018; Wortsman et al., 2020). Often, these algorithms start with a fixed-size model, and the algorithm aims to allocate a subset of the model for each task such that only a specific part of the model is responsible for the knowledge for that task. For instance, PackNet (Mallya and Lazebnik, 2018) uses iterative pruning to free space for future tasks, while in (Wortsman et al., 2020) the authors propose to fix with a randomly initialized and over-parameterized network to find a binary mask for each task. Nevertheless, even though the focus of these methods is the model and its parameters, the main objective is to have an algorithm that can use the model as efficiently as possible. Consequently, the significance of the architecture is often overlooked.

We believe our work is orthogonal to the algorithmic side of continual learning research as any of the methods mentioned above can use a different architecture for their proposed algorithm, as we have shown in Fig. 1a and Tab. 8.

4.2 Architecture

There have been some efforts on applying architecture search to continual learning (Xu and Zhu, 2018; Huang et al., 2019; Gao et al., 2020; Li et al., 2019; Lee et al., 2021). However, when it comes to architecture search, the current focus of continual learning literature is mainly on deciding how to efficiently share or expand the model by delegating these decisions to the controller. For instance, both Lee et al. (2021) and Lee et al. (2021) authors use a ResNet model as the base model and propose a search algorithm that decides whether the parameters should be reused or new parameters are needed for the new task, and the implications of architectural decisions (e.g., the width of the model, impact of normalization, etc.) on continual learning metrics have not been discussed. We believe future continual learning works that focus on the architecture search can benefit from our work by designing better search spaces that include various components that we have shown are important in continual learning.

Beyond architecture search methods, there are several works that focus on the non-algorithmic side of CL. Related to the architecture, Mirzadeh et al. (2020) study the impact of width and depth on forgetting and show that as width increases, the forgetting will decrease. Our work extends their analysis for larger-scale ImageNet-1K benchmark and various architectures. Recently, Ramasesh et al. (2022) shows that in the pre-training setups, the scale of model and dataset can improve CL performance. In this work, rather than focusing on the pre-training setups, we study the learning and retention capabilities of various architectures when models are trained from scratch. However, we show that while scaling the model can be helpful, the impact of architectural decisions is more significant. For instance, as shown in Tab. 1 and Tab. 2b, the best performing architectures are not the ones with highest number of parameters.

Finally, while in Sec. 3 we have discussed the related works to each specific architectural component, we believe our work draws a comprehensive picture of various aspects of architectures in continual learning and poses interesting directions for future works.

5 Discussion

In this work, we have studied the role of architectures in continual learning. Through extensive experiments on a variety of benchmarks, we have shown that different neural network architectures have different learning and retention capabilities, and a slight change in the architecture can result in a significant change in performance in a continual learning setting.

Our work can be extended in several directions. We have already empirically demonstrated that different architectural components, such as width, depth, normalization, and pooling layers, have very specific effects in a continual learning setting. One could theoretically explore these components vis-à-vis continual learning. Similarly, designing architectural components that encode the inductive biases one hopes to have in a model, such as a measure of or resistance to the distributional shifts, a sense of relatedness of different tasks, or fast adaptation to new tasks, could be a very interesting future direction. We hope that the community takes up these questions and answers them in future works.

Acknowledgments

We would like to thank Huiyi Hu, Alexandre Galashov, and Yee Whye Teh for helpful discussions and feedback on this project.

References

Balaji, Y., Farajtabar, M., Yin, D., Mott, A., and Li, A. (2020). The effectiveness of memory replay in large scale continual learning. *arXiv preprint arXiv:2010.02418*.

Bartlett, P., Helmbold, D., and Long, P. (2018a). Gradient descent with identity initialization efficiently learns positive definite linear transformations by deep residual networks. In *International Conference on Machine Learning*, pages 521–530. PMLR.

Bartlett, P. L., Evans, S. N., and Long, P. M. (2018b). Representing smooth functions as compositions of near-identity functions with implications for deep network optimization. *arXiv preprint arXiv:1804.05012*.

Bjorck, N., Gomes, C. P., Selman, B., and Weinberger, K. Q. (2018). Understanding batch normalization. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. (2018). Efficient lifelong learning with a-gem. In *International Conference on Learning Representations*.

Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P. K., Torr, P. H. S., and Ranzato, M. (2019). On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*.

Cho, K., Raiko, T., and Ilin, A. (2011). Enhanced gradient and adaptive learning rate for training restricted boltzmann machines. In Getoor, L. and Scheffer, T., editors, *Proceedings of the 28th International Conference on Machine Learning, ICML*, pages 105–112. Omnipress.

Dai, Z., Liu, H., Le, Q. V., and Tan, M. (2021). Coatnet: Marrying convolution and attention for all data sizes. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems, NeurIPS*.

Derakhshani, M. M., Zhen, X., Shao, L., and Snoek, C. (2021). Kernel continual learning. In *Proceedings of the 38th International Conference on Machine Learning, ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 2621–2631. PMLR.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR, 2021*.

Farajtabar, M., Azizan, N., Mott, A., and Li, A. (2020). Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3762–3773. PMLR.

Gao, Q., Luo, Z., and Klabjan, D. (2020). Efficient architecture search for continual learning. *CoRR*, abs/2006.04027.

Goodfellow, I. J., Mirza, M., Da, X., Courville, A. C., and Bengio, Y. (2014). An empirical investigation of catastrophic forgeting in gradient-based neural networks. In *2nd International Conference on Learning Representations, ICLR*.

Hardt, M. and Ma, T. (2016). Identity matters in deep learning. *arXiv preprint arXiv:1611.04231*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Hsu, Y.-C., Liu, Y.-C., Ramasamy, A., and Kira, Z. (2018). Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *arXiv preprint arXiv:1810.12488*.

Huang, S., François-Lavet, V., and Rabusseau, G. (2019). Neural architecture search for class-incremental learning. *CoRR*, abs/1909.06686.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org.

Jastrzebski, S., Arpit, D., Ballas, N., Verma, V., Che, T., and Bengio, Y. (2017). Residual connections encourage iterative inference. *arXiv preprint arXiv:1710.04773*.

Kirichenko, P., Farajtabar, M., Rao, D., Lakshminarayanan, B., Levine, N., Li, A., Hu, H., Wilson, A. G., and Pascanu, R. (2021). Task-agnostic continual learning with hybrid probabilistic models. In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6).

Lee, S., Behpour, S., and Eaton, E. (2021). Sharing less is more: Lifelong learning in deep networks with selective layer transfer. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 6065–6075. PMLR.

Li, X., Zhou, Y., Wu, T., Socher, R., and Xiong, C. (2019). Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, volume 97 of *Proceedings of Machine Learning Research*, pages 3925–3934.

Mallya, A. and Lazebnik, S. (2018). Packnet: Adding multiple tasks to a single network by iterative pruning. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*, pages 7765–7773. Computer Vision Foundation / IEEE Computer Society.

McCloskey, M. and Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165.

Mirzadeh, S. I., Chaudhry, A., Hu, H., Pascanu, R., Gorur, D., and Farajtabar, M. (2021a). Wide neural networks forget less catastrophically. *ArXiv*, abs/2110.11526.

Mirzadeh, S. I., Farajtabar, M., Gorur, D., Pascanu, R., and Ghasemzadeh, H. (2021b). Linear mode connectivity in multitask and continual learning. In *International Conference on Learning Representations*.

Mirzadeh, S. I., Farajtabar, M., Pascanu, R., and Ghasemzadeh, H. (2020). Understanding the role of training regimes in continual learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7308–7320.

Paul, S. and Chen, P.-Y. (2022). Vision transformers are robust learners. *AAAI*.

Pham, Q., Liu, C., and HOI, S. (2022). Continual normalization: Rethinking batch normalization for online continual learning. In *International Conference on Learning Representations*.

Ramasesh, V. V., Lewkowycz, A., and Dyer, E. (2022). Effect of scale on catastrophic forgetting in neural networks. In *International Conference on Learning Representations*.

Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., and Tesauro, G. (2018). Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations*.

Ring, M. B. (1995). *Continual learning in reinforcement environments*. PhD thesis, University of Texas at Austin, TX, USA.

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T. P., and Wayne, G. (2019). Experience replay for continual learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. (2018). How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Shin, H., Lee, J. K., Kim, J., and Kim, J. (2017). Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2990–2999.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. A. (2015). Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806.

Thrun, S. (1995). A lifelong learning perspective for mobile robot control. In *Intelligent robots and systems*, pages 201–214. Elsevier.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Wortsman, M., Ramanujan, V., Liu, R., Kembhavi, A., Rastegari, M., Yosinski, J., and Farhadi, A. (2020). Supermasks in superposition. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*.

Wu, Y. and He, K. (2018). Group normalization. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich*, volume 11217 of *Lecture Notes in Computer Science*, pages 3–19. Springer.

Xu, J. and Zhu, Z. (2018). Reinforced continual learning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 907–916.

Yin, D., Farajtabar, M., Li, A., Levine, N., and Mott, A. (2020). Optimization and generalization of regularization-based continual learning: a loss approximation viewpoint. *arXiv preprint arXiv:2006.10974*.

Yoon, J., Yang, E., Lee, J., and Hwang, S. J. (2018). Lifelong learning with dynamically expandable networks. In *Sixth International Conference on Learning Representations*. ICLR.

Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. In *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*. BMVA Press.

Zenke, F., Poole, B., and Ganguli, S. (2017). Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3987–3995. JMLR.

Zhou, B., Khosla, A., Lapedriza, À., Oliva, A., and Torralba, A. (2016). Learning deep features for discriminative localization. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2921–2929.

Overview

In this document, we provide additional details regarding the main work. More specifically:

- First, in Sec. A we cover the details regarding our experimental setup, design choices, architecture details, and hyper-parameters.
- Second, in Sec. B we provide additional experiments and results and a more detailed version of some of the experiments.

A Experimental Setup Details

A.1 Architectures

A.1.1 CNNs

Unless otherwise stated, the CNN models in this work solely include the convolutional layers, followed by a single feed-forward layer for classification. All CNNs use the ReLU activation function and use 3x3 kernels and a stride of 2.

For the CIFAR-100 experiments, the *base CNN* contains 3 convolutional layers with 32, 64, and 128 channels respectively, and *CNN* $\times N$ refers to the base CNN model where the number of channels in each layer is multiplied by N. Hence, *CNN* $\times 4$ refers to 3 convolutional layers with 128, 256, and 512 channels, followed by a feed-forward classification layer. In Split ImageNet-1K experiments, the *base CNN* contains 6 convolutional layers with 64 channels for the first four layers and 128 channels for the last two layers. Similar to the setup for the Split CIFAR-100 experiments, all models have a single-layer feed-forward layer for classification and use a kernel size of 3, with a stride of 2.

When we use pooling layers in CNNs(e.g., Sec. 3.4), to keep the dimension of features the same with the CNNs that don't have pooling layers, we use a stride of 1 for convolutional layers. In other words, every convolutional layer leads to a reduced feature map by a factor of 2: either the convolution has stride 2 (e.g., CNNs in Tab. 1) or else it has a stride 1, followed by a pooling layer (e.g., CNN+(Avg/Max)Pool in Tab. 4 and Tab. 7). Finally, for the experiments with skip-connections added, we add 2 skip-connections (with projections) for the 3-layer CNNs on Split CIFAR100 that adds shortcuts from layer 1 to output of layer 2 and layer 2 to layer 3. For the Split ImageNet-1K benchmark for 6-layer CNNs, we add 3 shortcuts that bypass every other layer.

A.1.2 ResNets

The ResNets we use in the ImageNet experiment are the standard models, and we do not modify them. However, the ResNets in the CIFAR-100 experiments use 3×3 kernels with a stride of 1, rather than the 7×7 kernels with a stride of 2. This is a common practice for the ResNet models for low-dimensional CIFAR images since it does not reduce the dimension of input significantly. Other than this modification for CIFAR-100 experiments, the rest of the ResNet architecture kept the same.

A.1.3 WideResNets

The WideResNets (WRN) models on both CIFAR-100 and ImageNet benchmarks follow the original implementation of WRN models. For all experiments, we use a dropout factor of 0.1 as we empirically observed increasing the dropout does not improve the performance significantly.

A.1.4 Vision Transformers

The Vision Transformer (ViT) models in our ImageNet experiment follow the same architecture as the original vision transformers. Similar to the original ViT models, we use the patch size of 16 for both ViT models in our ImageNet-1K experiments.

However, since the original vision transformer paper does not provide the details for the best practices for the CIFAR benchmark, we used smaller versions of the ViTs to match the training cost of other models. In those experiments, *ViT 512-1024* refers to a ViT model with 4 layers, with a width of 512 and MLP size of 1024. Similarly, *ViT 1024-1536* has a width of 1024 with the MLP hidden size of 1536. All models use the patch size of 4 (i.e., 64 patches for CIFAR images), but we empirically observed increasing the patch size to 8 does not impact the results significantly.

A.2 Hyperparameters

In this section, we report the hyper-parameters we used for our experiments. We include the chosen hyper-parameter for each architecture in parentheses.

A.3 Rotated MNIST

We follow the setting in (Mirzadeh et al., 2021a) for our MNIST experiments in Sec. 3.1.

```
learning rate: [0.001, 0.01 (MLP), 0.05, 0.1]
momentum: [0.0 (MLP), 0.8]
weight decay: [0.0 (MLP), 0.0001]
batch size: [16, 32, 64 (MLP)]
```

A.4 Split CIFAR-100

We use the following grid of hyper-parameters for the CIFAR-100 experiments:

```
learning rate: [0.001, 0.005, 0.01 (ViT 1024, ResNet 50/101), 0.05 (CNNs, ResNet 18/34, ViT 512, WRNs), 0.1]
learning rate decay: [1.0 (ResNet 50/101), 0.8 (CNN, ViTs, ResNet 18/34, WRNs)]
momentum: [0.0 (CNN, ViTs, ResNet 50/101), 0.8 (ResNet 18/34, WRNs)]
weight decay: [0.0 (CNNs, ViTs), 0.0001 (ResNets, WRNs)]
batch size: [8 (CNNs, ResNet18/34), 16 (WRNs), 64 (ResNet 50/101, ViT 512), 128 (ViT 1024)]
```

We note that the learning rate decay is applied after each task.

A.5 Split ImageNet-1K

Due to computation budget, we use smaller a smaller grid for the ImageNet-1K experiments. However, we make sure that the grid is diverse enough to cover various family of architectures.

```
learning rate: [0.005, 0.01, 0.05, 0.1 (All models)]
learning rate decay: [1.0, 0.75 (All models)]
momentum: [0.0, 0.8 (All models)]
weight decay: [0.0 (CNNs), 0.0001 (ResNets, WRN, ViTs)]
batch size: [64 (All models), 256]
```

B Additional Results

B.1 Algorithms vs. Architectures

In Fig. 1a, we have provided the results for the ResNet-18 model. Here, we provide the average accuracy and average forgetting for various models and algorithms on split CIFAR-100 with 20 tasks.

Table 8: Different Algorithms and Architectures

Algorithm	Parameters (M)	Architecture	Average Accuracy	Average Forgetting
Finetune	11.2	ResNet-18	45.0 ±0.63	36.8 ±1.08
EWC	11.2	ResNet-18	50.6 ±0.70	26.6 ±2.53
AGEM (Mem = 1000)	11.2	ResNet-18	61.8 ±0.45	22.9 ±1.59
ER (Mem = 1000)	11.2	ResNet-18	64.3 ±0.99	19.7 ±1.26
Finetune	11.9	ResNet-18 (w/o GAP)	67.4 ±0.76	11.2 ±1.98
AGEM (Mem = 1000)	11.9	ResNet-18 (w/o GAP)	72.8 ±1.33	5.8 ±0.39
ER (Mem = 1000)	11.9	ResNet-18 (w/o GAP)	74.4 ±0.33	4.6 ±0.54
Finetune	2.3	CNN x4	68.1 ±0.50	8.7 ±0.21
ER (Mem = 1000)	2.3	CNN x4	74.4 ±0.27	2.4 ±0.12

We remind the reader that the aim of our work is not to undermine the importance of continual learning algorithms. On the contrary, we appreciate the recent algorithmic improvements in the continual learning literature. The aim of this work is to show that *the role of architecture is significant, and a good architecture can complement a good algorithm in continual learning.*

B.2 Batchnorm Statistics on Split CIFAR-100

We show the first layer's BN statistics for CNN×4 in Fig. 3 using kernel density estimation with Gaussian kernel. As illustrated, the batch statistics and learned BN parameters do not change significantly across different tasks. Here, for simplicity, we have shown only four tasks from the beginning, middle, and late of the learning experience. Moreover, we focus on the first layer's statistics, which is the first layer of the model that operates on the data.

B.3 Batchnorm Statistics on Permuted MNIST

In Sec. 3.2, we have discussed that if the batch statistics change significantly across tasks, batch normalization can increase the forgetting. To this end, we design an experiment where we train two MLP-128 networks (with and without BN) on the Permuted MNIST benchmark(Goodfellow et al., 2014) with five

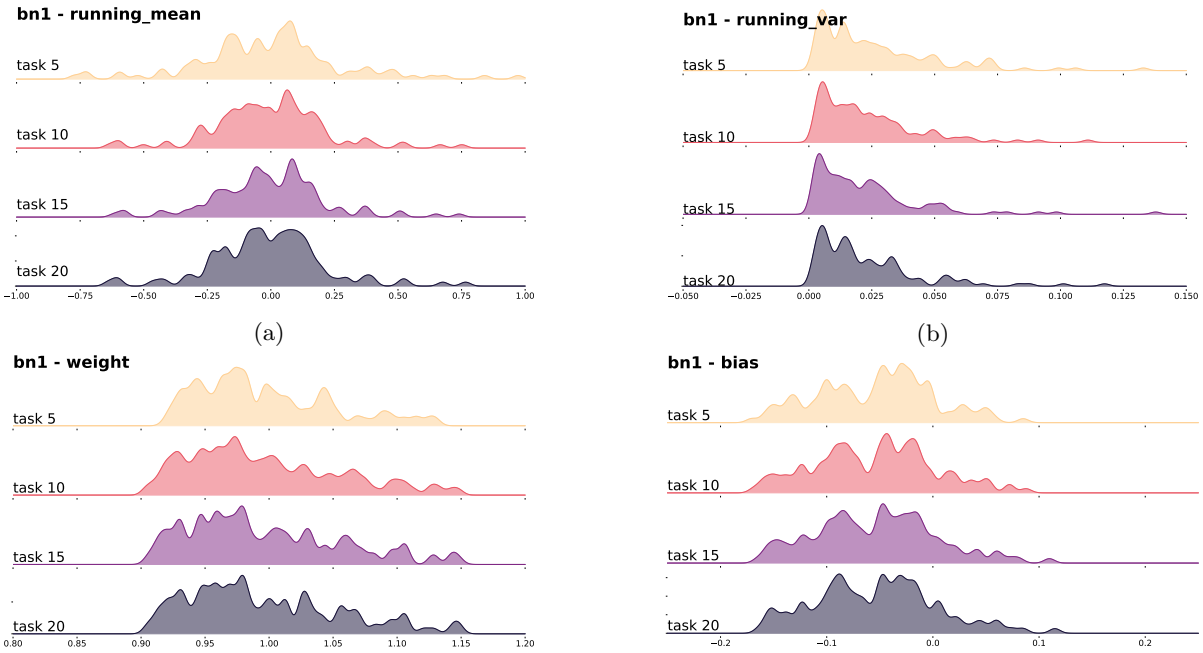


Figure 3: BN statistics for the first layer of CNN×4 on Split CIFAR-100: the statistics do not change significantly throughout the continual learning experience.

tasks. While Permuted MNIST is not a very realistic benchmark, it fits our requirements for synthetic distribution shift (i.e., shuffling pixels).

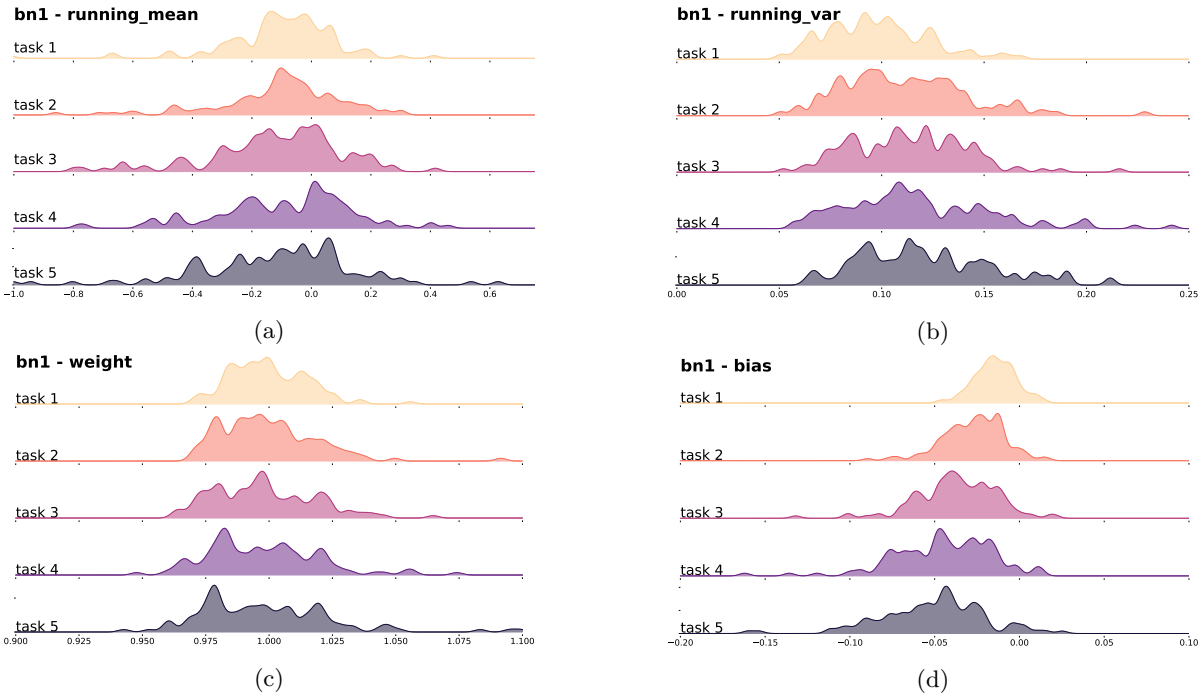
While Tab. 9 demonstrates the benefit of adding BN (i.e., improving learning accuracy), we can observe a significant increase in average forgetting as well. To investigate this more, we visualize the BN statistics in Fig. 4 where we can see compared to Fig. 3, the statistics change more, confirming our hypothesis in Sec. 3.2.

Table 9: Permuted MNIST: The MLP with BN has slightly higher learning accuracy, but significantly higher forgetting as well.

Model	Average Accuracy	Average Forgetting	Learning Accuracy
MLP-128	86.8 ±0.95	10.9 ±0.88	95.5 ±0.33
MLP-128 + BN	73.2 ±0.82	32.5 ±0.72	97.8 ±0.45

B.4 More number of epochs on Split CIFAR-100

While in our main text, we have used 10 epochs for Split CIFAR-100, here in Tab. 10 we show that the main conclusions hold even when we train the models longer.



(a) (b) (c) (d)

Figure 4: BN statistics for the first layer of MLP-128 on Permuted MNIST: the statistics change more compared to Fig. 3

Table 10: Comparing the impact of components in with two different settings: The components that are helpful in the short training time (e.g., removing GAP layers, adding pooling or batch norm layers), are also beneficial when the training time is longer.

		Epochs = 10			**Epochs = 50**		
Model	**Params (M)**	**Average Accuracy**	**Average Forgetting**	**Learning Accuracy**	**Average Accuracy**	**Average Forgetting**	**Learning Accuracy**
ResNet-18	11.2	45.0 ± 0.63	36.8 ± 1.08	74.9 ± 3.98	37.1 ± 0.59	48.9 ± 1.35	82.4 ± 4.83
ResNet-18 w/o GAP	11.9	67.4 ± 0.76	11.2 ± 1.98	74.2 ± 4.79	66.1 ± 0.44	17.3 ± 1.43	80.2 ± 4.77
ResNet-50	23.6	56.2 ± 0.88	9.5 ± 0.38	67.8 ± 5.09	53.4 ± 0.29	14.5 ± 0.64	75.3 ± 5.65
ResNet-50 w/o GAP	26.7	71.4 ± 0.29	6.6 ± 0.12	73.0 ± 5.18	71.2 ± 0.18	7.3 ± 0.22	76.5 ± 4.87
CNN x4	2.3	68.1 ± 0.5	8.7 ± 0.21	76.4 ± 6.92	62.6 ± 0.4	14.4 ± 0.62	75.2 ± 6.25
CNN x4 + BN	2.3	74.0 ± 0.56	8.1 ± 0.35	81.7 ± 6.68	68.9 ± 0.93	13.8 ± 0.68	80.7 ± 5.83
CNN x4 + Maxpool	2.3	74.4 ± 0.34	9.3 ± 0.47	83.3 ± 6.1	69.3 ± 0.79	13.5 ± 0.85	81.9 ± 5.47
CNN x8	7.5	69.9 ± 0.62	8.0 ± 0.71	77.5 ± 6.78	64.3 ± 0.82	13.2 ± 1.01	78.8 ± 6.61
CNN x8 + BN	7.5	76.1 ± 0.3	5.9 ± 0.16	81.7 ± 6.83	71.7 ± 0.79	11.5 ± 0.85	82.4 ± 6.18
CNN x8 + Maxpool	7.5	77.2 ± 0.53	7.1 ± 0.33	84.0 ± 5.81	73.6 ± 2.25	12.9 ± 1.07	84.4 ± 5.06