# Re-evaluating Continual Learning Scenarios: A Categorization and Case for Strong Baselines

# Abstract

Continual learning has received a great deal of attention recently with several approaches being proposed. However, evaluations involve a diverse set of scenarios making meaningful comparison difficult. This work provides a systematic categorization of the scenarios and evaluates them within a consistent framework including strong baselines and state-of-the-art methods. The results provide an understanding of the relative difficulty of the scenarios and that simple baselines (Adagrad, L2 regularization, and naive rehearsal strategies) can surprisingly achieve similar performance to current mainstream methods. We conclude with several suggestions for creating harder evaluation scenarios and future research directions. The code is available at <https://github.com/GT-RIPL/Continual-Learning-Benchmark>.

最近，持续学习受到了很多关注，提出了几种方法。但是，评估所涉及到的场景使有意义的比较变得困难。

这项工作提供了系统的场景分类并且用统一的框架(包括强大的基线和最新方法)对其进行评估。

结果提供了一个理解场景的相对难度和简单的基线(Adagrad, L2 regularization, and naive rehearsal strategies)可以令人惊讶地实现与目前主流方法的表现相似。

我们总结了一些建议，用于创建更难的评估场景和未来的研究方向。

该代码可在<https://github.com/GT-RIPL/Continual-Learning-Benchmark>上获得。

# Introduction

While current learning-based methods can achieve high performance on tasks, they only perform well when the testing data is similarly distributed as the training data. In other words, they cannot adapt continuously in dynamic environments where situations can significantly change. Such adaptation is desirable for any intelligent system, and is the hallmark of learning in biological systems. One approach to this problem is continual learning, where models are updated incrementally as data streams in. However, deep neural networks, which are currently state of art for many applications, are known to suffer catastrophic interference or forgetting when incrementally updated through gradient-based methods. This leads to the model forgetting how to solve old tasks after being exposed to a new one due to interference caused by parameter updates.

虽然当前基于学习的方法可以在任务上实现高性能，但是他们只能当测试数据与训练数据类似分布时表现良好。换句话说，他们不能持续适应情况可能发生明显变化的动态环境。这种适应适用于任何智能系统，是生物系统的学习特点。

解决这个问题的一种方法是连续学习，其中当数据流进入的时候，模型持续更新。然而，深度神经网络会遭受灾难性干扰或遗忘当通过基于梯度方法逐步更新的时候。这导致模型遗忘如何在输入新任务后由于参数更新的干扰解决旧任务。

To address this problem, several approaches have been proposed and a number of experimental methodologies (i.e. datasets, learning curricula, and architectures) have been used for evaluation. In this paper, we argue that the current set of evaluations have significant limitations, including lack of uniformity across the different experimental methodologies, lack of hyper-parameter tuning of reasonable baselines under similar tuning budgets as the proposed methods, and simplicity of the tasks (e.g. short task queues). Towards this end, we make several contributions in this paper: 1) A categorization of a large number of experimental methodologies into a few canonical settings along with a comparison of their difficulty, 2) A uniform but flexible framework for generating scenarios under this categorization and systematic evaluation of current state of art methods, and 3) Demonstration that very simple baselines can be surprisingly effective if used properly and result in comparable or better performance against the current state of art. We have released our framework (written in PyTorch) to enable fair and uniform evaluation to aid the community, and conclude with some suggested modifications to the scenarios to increase realism of the evaluation.

为了解决这个问题，已经提出了几种方法和一些实验方法（即datasets, learning curricula, and architectures）已用于评估。

在本文中，我们认为当前的评估集有很大的局限性，包括缺乏不同实验方法的统一性，缺乏超参数调整在与提出的方法类似的调整预算下的合理基线，以及简单的任务（例如短任务队列）。

为此，我们在本文中做了几点贡献：

1. 将大量实验方法分类为几个规范设置，并比较他们的难度。
2. 统一且灵活的框架，生成该分类下的情景和对当前方法的系统评价。
3. 证明非常简单的基线，如果使用得当并且产生与当前技术可比较的或更好的性能表现。

我们发布了框架（以PyTorch编写），以实现公平和统一的评估，并得出一些建议方案修改增加评估真实性的结论。

# 2.Generating Task Sequences for Evaluating Continual Learning

## 2.1Existing Experimental Methodologics

In order to evaluate continual learning methods, scenarios are commonly generated from datasets using two operations: permutation and splitting. The typical source dataset is MNIST [1], an image dataset of hand-written digits. The Permuted MNIST experiment [2] involves ten-digit classification, where each task consists of different permutations of the pixels in the images. The number of different permutations represents the length of the task sequence. This evaluation scenario is widely adopted

[3, 4, 5, 6, 7, 8, 9], despite criticism that it is less challenging in terms of forgetting [10]. Another typical scenario, the Split MNIST experiment, was initially introduced in a multi-headed form where the ten digits are split into five two-class classification tasks (the model has five output heads, one for each task) [9, 8, 6], and the task identity (1 to 5) is given at testing time. This scenario is argued to be easier since the selection of output head is given by the task identity [10]. Farquhar and Gal [10] propose a single-headed variant which does not require task identity, where it always requires the model to make a prediction over all classes (digits 0 to 9). Such single-headed Split MNIST is known as incremental class learning [11, 12, 13]. Van and Tolias [14] propose another variant of single-headed Split MNIST, where the model always predicts over two classes instead of ten classes. Furthermore, the similar multi-headed/single-headed strategies in Split MNIST can apply to Permuted MNIST [14] resulting in many combinations. These scenarios are used in different works, and therefore there is a lack of coherent comparison. This paper addresses this problem by providing a systematic interpretation of the differences between an old task and a new one (see Section 2.2).

为了评估连续学习方法，通常使用两个操作从数据集生成场景：置换（permutation）和分裂（splitting）。

典型的源数据集是MNIST [1]，一个手写数字的图像数据集。Permuted MNIST实验[2]涉及十个数字的分类，其中每个任务由图像中像素的不同排列组成。不同排列的数量表示任务序列的长度。该评估方案被广泛采用[3, 4, 5, 6, 7, 8, 9]，尽管批评该方案在遗忘方面不那么具有挑战性。

另一个典型方案，Split MNIST实验，最初以多头形式（multi-headed form）引入，即10个数字被分为5组二分类任务(该模型有5个输出头，每个任务一个) [9, 8, 6]，并在测试时给出任务标识（task identity）(1 to 5)。由于输出头的选择有任务标识给出，因此该方案被认为更容易[10]。

Farquhar和Gal [10]提出了一种不需要任务识别的单头变体，它总是要求模型对所有类别（数字0到9）进行预测。这种单头分裂MNIST被称为增量类学习(incremental class learning)[11,12,13]。

Van和Tolias [14]提出了单头Split MNIST的另一种变体，其中模型总是预测两个类而不是十个类。

此外相似的多头/单头Split MNIST策略可以应用到Permuted MNIST[14]。这些方案用于不同的工作，因此缺乏相关的比较。

本文通过对旧任务和新任务的差异提出系统解释来解决这个问题。(2.2)

## 2.2 A Categorization of Current Scenarios

**Incremental Domain Learning：**

* 输入分布(域)差异(input distribution(domain) difference)在转移学习(transfer learning)的背景下引起广泛讨论，主要作为域适应问题(domain adoption)。
* 与旨在将旧任务的知识转移到仅考虑新任务表现的新任务的域适应不同，连续学习(continual learning)设置旨在保持旧任务的性能，同时在新任务上实现合理的性能以及使用单一模型。
* 这种域差异由置换和分隔策略引起。实际上，最广泛使用的Permuted MNIST实验产生了这样的域差异。然而，由随机置换协议(random permutation protocol)产生的输入高度不相关，因此他们不能代表所有可能的方案。
* 为了生成更相关任务的方案，应该避免随机置换而保持原来图像像素之间的原始空间相关性，从而允许任务之间共享特征的可能性。
* 这个要求可以通过Split MNIST的变形满足，其中十个数字被分为5个二进制分类任务，并且该模型只含有二元分类器(单头)。该方案在图1的中间栏中说明。
* 使用单头模型的要求对于控制其他类型的差异特别重要。具体说来，单头模型确保相同的输出空间{Y1}={Y2}(任务标识变得不那么重要)，并且等量的MNIST数字保证二分类的输出分布是一样的P(Y1)=P(Y2).
* 结果，唯一的区别在图1中的中间列中显示，即输入图像x对于标签{0,1}从数字0/1变为1/2.

**Incremental Class Learning:**

* 该方案设计到多分类。
* 序列中的每个任务数据集中类的独占子集。
* P(X1)不等于P(X2)为真。
* 所有类标签都在相同的命名空间(单头)，输出节点数量等于任务序列中的总类数。
* 由于多分类的特性，P(Y1)不等于P(Y2)。
* 图1中的最右列演示了split-dataset strategy[11,10]如何产生这个方案。permutation strategy 同样可以生成任务序列[14]。
* 在permutation strategy的情况下，每个置换数字代表一个新类，因此累的总数需乘以排列组合数。

**Incremental Task Learning:**

* 输出空间在任务之间是不相交的。
* P(Y1)不等于P(Y2)为真，同时P(X1)不等于P(X2)为真，因为semantic classes differ。输出空间之间的差异是输出维度及其相关的语义。
* 例如，旧任务可以是5个类的分类问题，而新任务是单个值得回归任务。为了允许一个模型产生某特定任务的输出，该模型需要特定于任务的输出组件，这些输出组件通过附加信息和任务标识器t的选择。
* 一个典型的神经网络有多头输出层(每个任务一个头)。在测试时，只有头与t匹配才能被激活来进行预测。
* 生成该方案的一种常见方法是有多个数据集(ex: MNIST, CIFAR10, SVHN, AudioSet, CUB-200, etc.),并且在某任务中使用其中之一。
* 分裂和置换策略也可以生成该方案的任务序列，如图1和附录图2。
* 在本段已提到的工作中通常具有测试期间给出的任务标识; 因而在此实验工作遵循相同的设置。

# 3.Experiments

We now describe the experimental configuration. Here, we use the MNIST dataset with the splitting strategy (Figure 1) to generate the three continual learning scenarios in Table 1. The standard train/test split was used, with 60k training images (6k per digit) and 10k test images (1k per digit). The preprocessing of images includes zero padding to 32x32 pixels and a standard normalization to zero mean with unit variance. No other data augmentation, (e.g. random translation) is applied.

我们现在描述实验配置。此处使用MNIST数据集进行分割策略(图1)来生成表1中的三个连续学习场景。标准训练/测试分割，60k训练图像（每个数字6k）和10k测试图像(每个数字1k)。这种图像的预处理包括零填充到32x32像素和标准归一化(期望为0，方差为1)。无其他应用数据增加。

For a fair comparison, all methods use the same neural network architecture, which is a multi-layered perceptron with two hidden layers of 400 nodes each, followed by a softmax output layer. Both hidden layers use ReLU for the activation function. The loss function is a standard cross-entropy for classification in all methods and scenarios. All models are trained for 4 epochs per task with mini-batch size 128 using the Adam optimizer(…) as the default unless explicitly described. In all experiments, the optimizer is never reset.

为了公平比较，所有方法都使用相同的神经网络结构，这是一个多层感知器，有两个隐藏层，每层有400个节点，接着是softmax输出层。两个隐藏层使用ReLU作为激活函数。

损失函数是标准交叉熵，适用于所有方法和场景的分类。

所有模型(默认设置)：4 epochs per task, mini-batch(Adam optimizer) 128。

We use three baseline strategies. The most common baseline used in prior work is a neural network sequentially trained on all tasks in the standard way, in that the parameters learned from old tasks are fine-tuned to the new task. Such a model is usually optimized with Adam [21], but here we use different optimizers including Adam, SGD, and Adagrad [22]. In fact, we show that Adam is in general a poor choice for this task. The latter two optimizers use 0.01 for the learning rate without momentum in all scenarios. Another baseline, L2 regularization, prevents the parameters from deviating too much from those previously learned. Note that this is similar to EWC in that the identity matrix replaces the Fisher information matrix. In other work this is only evaluated in one specific scenario (permuted MNIST) with limited length (3) of task sequence [3]. Similar to other regularization methods, the L2 regularization requires tuning of the single-valued regularization coefficient, which is done by a grid search [3, 9].

我们使用三个基线策略。

先前工作中最常使用的基线是神经网络依次在所有任务上以标准方式进行训练，从旧任务中学习参数并精心调整到新任务。这种模型通常用Adam进行优化，但是这里我们使用不同的优化器包括Adam, SGD, Adagrad [22]。实际上，我们说明了Adam通常是糟糕的选择。后两个优化器使用0.01的学习率(无动力)。

另一个基线是L2 regularization，可防止参数偏离之前已学习的参数太多。注意，这与EWC类似，单位矩阵代替Fischer信息矩阵。换句话说，仅在一个任务序列限制的特定场景(permuted MNIST)中评估。与其他正则化方法类似，L2正则化要求调整单值正则化系数，这可以通过网格搜索完成。

The third baseline is a naive rehearsal strategy, which is sometimes called experience replay. The model has a small replay buffer to store a fraction of previous data randomly. While training a new task, each mini-batch is constructed by an equal amount (64/64) of new data and the rehearsal data. The buffer size is predefined and fixed to match the space overhead introduced by online EWC and SI (…), which converts to 1.1k images when the pixel value is saved in a 32-bit floating number (named Naïve rehearsal). Additionally, with the same memory space, it can store more images when compression is used. One naive compression is using an 8-bit integer to represent the pixel value; thus 4.4k images can be stored (named Naive rehearsal-C). For the buffer management, all tasks seen so far have an equal amount of images in the buffer while keeping the total number the same. This management is similar to iCaRL [11], except that we randomly pick the images for staying in the buffer.

第三个基线是天真排练策略(naïve rehearsal strategy), 有时成为经验重现(experience replay)。

该模型有一个小型的重放缓冲器，随机存储之前的一部分数据。

在训练新任务是，每一格mini-batch由相同数量的新数据(new data)和排练数据(rehearsal data)组成。缓冲器大小已提前设置并固定，为了匹配由再现EWC和SI(#parameter…)引起的空间开销，当像素转换为1.1k图像值以32位浮点数保存(名为naïve rehearsal)。

此外，使用相同的内存空间，它可以在使用压缩时储存更多图像。一泓压缩方法是使用8-bit整数代表像素值，因此4.4k张图像可以被存储(称为Naïve rehearsal-C)。

对于缓冲器管理，到目前为止所有任务被视为缓冲器中具有相同数量的图像，同时保持总数不变。这种管理类似于iCaRL,不同之处在于我们随机选取图像来保存在缓冲器中。

For comparison, we pick several popular methods (EWC[3], online EWC[23], SI[9], MAS[24], LwF[25]) and state-of-the-art rehearsal-based methods (DGR[8], RtF[14]) with generative model. The hyperparameter is tuned by a grid search, and the results with the best setting are reported. The memory overhead is the same among online EWC, SI, MAS, and DGR. RtF has only half the overhead of DGR since its classification model is shared with its generative model. The hyperparameter of EWC, online EWC, SI, and MAS is tuned with grid search. We use the results from Ven and Tolias [14], which provides an analysis and comparison developed concurrently with our work, for LwF, DGR, and RtF since the same model and training procedures are used.

为了比较，我们选择几种生成模型的流行方法（EWC [3]，online EWC [23]，SI [9]，MAS [24]，LwF [25]）和最先进的基于排练的方法（DGR [ 8]，RtF [14]）。

通过网格搜索调整超参数，并报告了最佳设置的结果。

memory overhead: EWC, SI,MAS, DGR相同。RtF是DGR的一半(由于分类模型与生成模型共享)。

超参数: EWC, online EWC, SI 和 MAS通过网格搜索微调。

我们使用Ven 和 Tolias的结果[14],它提供了LwF, DGR 和RtF的分析和比较。

# 4.Results