MetaMix: Towards Corruption-Robust Continual Learning with Temporally Self-Adaptive Data Transformation

Zhenyi Wang[1] Li Shen [2,*] Donglin Zhan [3] Qiuling Suo [1]
Yanjun Zhu [1] Tiehang Duan [4] Mingchen Gao [1]
[1]State University of New York at Buffalo, USA; [2]JD Explore Academy, China
[3] Columbia University, USA; [4] Meta, USA
{zhenyiwa, qiulings, yjzhu, mgao8}@buffalo.edu; dz2478@columbia.edu
{mathshenli, tiehang.duan}@gmail.com

Abstract

Continual Learning (CL) has achieved rapid progress in recent years. However, it is still largely unknown how to determine whether a CL model is trustworthy and how to foster its trustworthiness. This work focuses on evaluating and improving the robustness to corruptions of existing CL models. Our empirical evaluation results show that existing state-of-the-art (SOTA) CL models are particularly vulnerable to various data corruptions during testing. To make them trustworthy and robust to corruptions deployed in safety-critical scenarios, we propose a meta-learning framework of self-adaptive data augmentation to tackle the corruption robustness in CL. The proposed framework, MetaMix, learns to augment and mix data, automatically transforming the new task data or memory data. It directly optimizes the generalization performance against data corruptions during training. To evaluate the corruption robustness of our proposed approach, we construct several CL corruption datasets with different levels of severity. We perform comprehensive experiments on both task- and class-continual learning. Extensive experiments demonstrate the effectiveness of our proposed method compared to SOTA baselines.

1. Introduction

Humans constantly acquire new information throughout their lifespan and easily recognize information shifts such as structure and style variations in images. Continual learning (CL) aims at imitating human's ability to learn from non-stationary data distributions without forgetting the previously learned knowledge. The past few years have witnessed rapid progress in CL research [1, 30, 31, 36, 45]. Despite the success, existing CL systems overlook the robustness against unforeseen data shifts during testing. They assume that training and test images for each task follow the same distribution. However, as data distributions evolve and new scenarios occur, test images often encounter various corruptions such as snow, blur, pixelation, and combinations, resulting in a shifted distribution from the training set. For example, Figure 1 shows various corruptions applied on one image from Split-miniImageNet.

Data corruption can drastically impair the performance of existing image recognition systems. A recent study [21] shows that classification accuracy of various architectures has dropped significantly on the ImageNet test set with some simple and natural corruptions. Our empirical evaluation results show that state-of-the-art CL models are even more vulnerable to these corruptions during testing. For example, the accuracy of DER++ [5] for task-continual learning decreases from 93.9% to 50.5% on split-CIFAR10; accuracy drops to 10.6 % from 75.6 % on split-CIFAR100; accuracy decreases to 9.8% from 61.3% on split-miniImageNet by applying those common corruptions on test data of each CL task. This severe issue makes existing CL models highly unreliable in safety-critical applications. Thus, improving the robustness of CL models to foster their trustworthiness when deployed in real-world scenarios is essential.

Training a CL model robust to various corruptions is difficult due to the following challenges. 1) Unseen corruptions could perturb the test set far beyond those encountered during training. A model that naively augments training images with seen corruptions cannot generalize to the new ones during testing. Also, it is unrealistic to enumerate all possible corruptions during training since there are infinite types of corruptions and their combinations. 2) With the ever-evolving data distributions in CL, an effective augmentation strategy learned on previous tasks may gradually become less effective because the optimal augmentation strategies are task-dependent and dynamically change over tasks [11].

*Corresponding author

(a) Original data (b) Brightness (c) Contrast (d) Defocus Blur (e) Elastic (f) Fog

Figure 1. Visualization of five types of different corruption operations on the testing images of Split-miniImageNet.

Although we can adopt a memory buffer to store data from previous tasks, augmenting and replaying them at later training iterations, the augmented memory may gradually become less effective as the model could memorize the stored information after replay runs. Recent approaches, such as Augmix [22] composes and combines multiple pre-defined augmentation operations with different depths and widths, show efficacy in improving robustness under traditional supervised classification tasks. However, these approaches are not directly applicable to corruption-robust CL since they only use *a fixed random augmentation strategy* that is often not optimal for non-stationary data distributions in CL.

To address these unique challenges of corruption-robustness in CL, we propose a temporally self-adaptive Augmix within a meta-learning framework, named MetaMix. It adaptively augments the memory buffer data and the currently received new data by learning to mix the augmentation operations tailored to the evolving data distributions. In particular, our automatic self-adaptive MetaMix is a bi-level optimization, simulating the evaluation process on unseen corruptions. We randomly divide the training augmentation operations into pseudo-seen and pseudo-unseen operations at each CL step. The lower-level optimization is to optimize the model performance on the pseudo-seen operations; the upper-level optimization is to optimize the generalization of the pseudo-unseen operations. The augmentation strategy is governed by an LSTM, which inputs context information and outputs the corresponding mixing parameters for the augmentations. The proposed MetaMix ensures the augmentation strategy automatically adapts to non-stationary data distribution. Furthermore, the objective is to optimize the performance of the pseudo-unseen corruption operations, which aligns with our goal during testing and encourages the generalization to unseen corruptions.

To evaluate the corruption robustness of existing and the proposed methods, we propose a new challenging benchmark where various corruptions perturb the testing data of each CL task. To facilitate future research, we construct several new datasets, including split-CIFAR-10-C, split-CIFAR-100-C, and split-miniImageNet-C. Extensive experiments on the constructed benchmarks demonstrate the effectiveness of our proposed MetaMix approach compared with several SOTA data-augmentation approaches adapted for CL. We summarize our contributions as follows:

- To our best knowledge, we are the first to study the corruption-robustness of CL methods. Accordingly, we propose the first set of novel benchmarks for evaluating the corruption-robustness of existing CL methods and moving towards trustworthy CL.
- We propose a self-adaptive augmentation method, MetaMix, by learning to mix and augment the training data of each CL task to achieve corruption-robustness on unseen corruptions for each CL task.
- Our method is versatile and can be seamlessly integrated with existing CL methods. Extensive experiments with both task/class continual learning demonstrate the effectiveness of MetaMix.

2. Related Work

2.1. Continual Learning

CL focuses on learning non-stationary data distribution without forgetting previous knowledge. These methods can be categorized into: 1) retaining memory for future replay [1, 4, 8, 9, 17, 31, 35, 42, 46, 48, 49]; 2) designing tailored network architectures [14, 24, 37, 41, 54]; 3) performing proper regularization during parameter updates [7, 26, 40, 56]; 4) introducing Bayesian methods for model parameter inference [13, 23, 34, 53]; 5) subspace-based methods [27, 30, 38] and 6) prompt-tuning [29] based methods [50, 51]; 7) transformer-based methods [47, 52]. However, all these works focus on the simplified setting in which training and testing data of each CL task follow the same distribution. In practice, the testing data may come from another different distribution due to the variations in weather, light, snow, etc. These distortions pose significant robustness challenges for existing CL methods. Existing literature largely ignored these trustworthy properties of CL. According to the terminology of [43], CL settings can be categorized into three scenarios based on whether task identities of the testing data are available to the CL model during test time. Among the three scenarios, task-continual learning studies the scenario that the task identities are available during testing and is the easiest learning setting; domain incremental learning focuses on the case that all the tasks during CL solve the same classification task with different data distribution and the task identities are unknown during testing; class-continual learning focuses

on the scenario that task identities are unavailable and is the most difficult learning setting. We focus on task/class CL scenarios.

Recently, robust CL began to attract researchers' attention. Self-Purified Replay [25] and PuriDivER [3] consider CL on noisy data stream with *label noise*. Their work is orthogonal to ours since they consider noisy data during training; we instead consider the robustness to *data corruptions* during testing. A closely related work to our is Continual Active Adaptation (CAA) [32], which is different from and orthogonal to ours since they focus on the setting that for each data distribution, the label space for each learning task is the same. Each task only encounters a single data corruption. During testing, CAA focuses on generalization to previously *seen* corruption operations. In contrast, we focus on the setting that each task solves a different task. During testing, the CL model should be robust to *unseen* data corruptions.

2.2. Data Augmentation

We categorize existing data augmentation techniques into two classes: 1) improving performance on clean data; 2) improving performance and being robust to data distribution shift and corruptions.

Augmentations for Improving Clean Data Performance. Standard data augmentation aims to improve the generalization performance on clean data. To improve the performance of neural network, standard supervised learning adopts common augmentations, such as random flips, rotations, and crops. Recently, Cutout [12], Mixup [57], CutMix [55] and MaxUp [16] show excellent performance on clean datasets. Other techniques include automatically learning augmentation policy from data, e.g., AutoAugment [10] and RandAugment [11]. However, AutoAugment is computationally impractical for CL problems since it needs a huge computation cost and separate proxy tasks to train the policy network.

Augmentations for Being Robust to Corruptions. Augmix [22] is one of the SOTA methods on the data corruption benchmarks, such as ImageNet-C, CIFAR100-C, etc. DeepAugment [20] and AdversarialAugment [6] transform and augment clean images by perturbing the representations/features of neural networks to generate new augmented images. However, this increases a lot of computation costs. Thus, DeepAugment [20] and AdversarialAugment [6] are impractical for CL. We thus choose the more efficient and effective AugMix and propose a new self-adaptive AugMix for CL.

3. Methodology

In this section, we first define the corruption robustness of CL and then describe the proposed MetaMix approach. We describe the notation in the main text. We also provide a notation table in Table 4 in Appendix for ease of reading.

3.1. Problem Setup

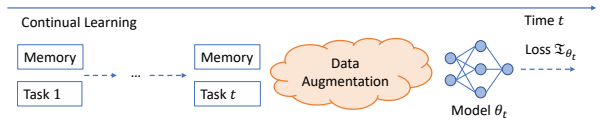
Corruption robustness of CL. We consider the problem of learning a sequence of tasks denoted as $\mathcal{D}^{tr} = \{\mathcal{D}_1^{tr}, \mathcal{D}_2^{tr}, \cdots, \mathcal{D}_N^{tr}\}$, where N is the number of training tasks. The k-th task training data $\mathcal{D}_k^{tr}$ consists of many labeled examples $(\boldsymbol{x}^k, y^k, \mathcal{T}_k)$, where $\boldsymbol{x}^k$ is the data example in the task, y^k is the corresponding data label, and $\mathcal{T}_k$ is the task identifier. The goal is to learn a model $f_{\boldsymbol{\theta}}$ on the training task sequence $\mathcal{D}^{tr}$ so that it performs well on the testing set of all the learned tasks $\mathcal{D}^{te} = \{\mathcal{D}_1^{te}, \mathcal{D}_2^{te}, \cdots, \mathcal{D}_N^{te}\}$ without forgetting previously learned knowledge. In particular, we focus on improving the model robustness on the testing set for each task $\mathcal{T}_k$ under a collection of corruption operations $\mathcal{C}$. Below, we define the CL corruption robustness.

Definition 1 (CL corruption robustness)**.** *Given a sequence of N tasks with test data $\mathcal{D}^{te} = \{\mathcal{D}_1^{te}, \mathcal{D}_2^{te}, \cdots, \mathcal{D}_N^{te}\}$, and a collection of corruption operations $\mathcal{C}$, the corruption robustness is the expectation accuracy after applying corruptions $\mathcal{C}$ on the test data of each task k, i.e.,*

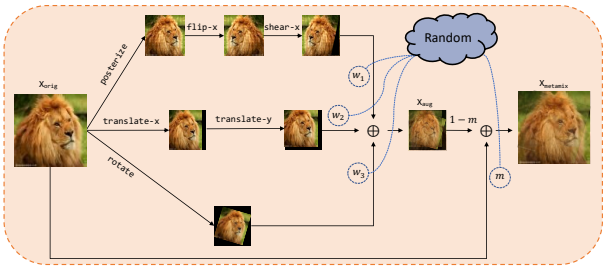
$$ACC = \frac{1}{N}\sum_{k=1}^{k=N} \mathbb{E}_{c\sim\mathcal{C}}[\mathbb{E}_{(\boldsymbol{x}^k, y^k, \mathcal{T}_k)\sim D_k^{te}} \mathbb{I}(f(c(\boldsymbol{x}^k)) = y^k)] \tag{1}$$

where $c(\boldsymbol{x}^k)$ is the resulting testing data after applying the corruption operation c on $\boldsymbol{x}^k$. $\mathbb{I}()$ is the indicator function. ACC measures the average robust accuracy after applying a collection of corruption operations $\mathcal{C}$.

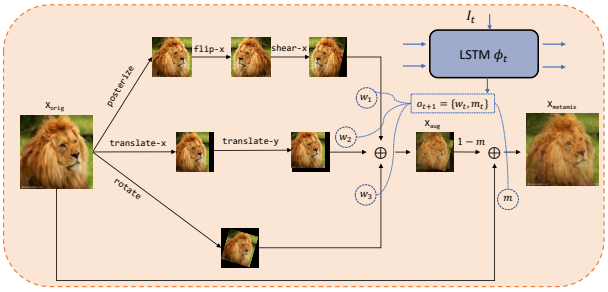
Preliminary of AugMix. AugMix [22] is a recently proposed SOTA augmentation technique that significantly improves model robustness in standard supervised learning. Considering that CL requires the algorithm to be both efficient and effective, we thus choose the efficient and effective AugMix to adapt to CL. AugMix maintains augmented data diversity by mixing several augmentation chains. Suppose we have an augmentation operation set $\mathcal{O} = \{op_1, op_2, \cdots, op_M\}$ during training. The standard AugMix implementation samples three operations $op1$, $op2$ and $op3$ from $\mathcal{O}$, and composes them with varying depths to form several operation chains; where each chain consists of one to three randomly selected augmentation operations. The augmented image is a convex mix of A chains (A denotes the number of augmentation chains), i.e., $\boldsymbol{x}_{aug} = w_1 \cdot chain_1(\boldsymbol{x}) + w_2 \cdot chain_2(\boldsymbol{x}) + ... + w_A \cdot chain_A(\boldsymbol{x})$, where mixing weights $\boldsymbol{w} = (w_1, w_2, ..., w_A)$ is sampled from the distribution, $\text{Drichelet}(\alpha, \alpha, ..., \alpha)$ with $\alpha = 1$. With the mixed image $\boldsymbol{x}_{aug}$, AugMix combines it with the original image through another random convex combination to obtain the final image $\boldsymbol{x}_{augmix}$, i.e., $\boldsymbol{x}_{augmix} = m\boldsymbol{x} + (1-m)\boldsymbol{x}_{aug}$, where m is sampled from a Beta(α, α) distribution. AugMix adopts a fixed random augmentation policy, i.e., Dirichlet and Beta distributions, during the entire training process.



(a) Continual Learning Process



(b) AugMix



(c) MetaMix (Ours)

Figure 2. Comparisons between AugMix [22] and our proposed MetaMix. (a) Tasks sequentially arrive during CL. (b) At each CL step, AugMix adopts a *fixed and random* augmentation policy during the entire learning process, thus lacking adaptation to the non-stationary data in CL and may not be the optimal strategy for each task. (c) MetaMix automatically adapts to the non-stationary data distribution during CL by meta-learning the mixing parameters of augmentation. At time t, we sample a batch of data $(\boldsymbol{x}_r, y_r)$ from the memory buffer and concatenate them with the current received data $(\boldsymbol{x}_t, y_t)$ as $(\boldsymbol{x}_b, y_b)$. We apply MetaMix to augment each data example. Specifically, we feed the context information $\boldsymbol{I}_t$ into the LSTM cell with parameters $\boldsymbol{\phi}_t$, and obtain the mixing weights $\boldsymbol{w}_t$ and m_t. The CL model $f_{\boldsymbol{\theta}_t}$ calculates the losses for the augmented data. We adopt bi-level optimization to improve model generalization on unseen corruptions by splitting the augmentation operations into a pseudo-seen set $\mathcal{S}_t$ and pseudo-unseen set $\mathcal{U}_t$. The loss $\mathcal{L}_{\boldsymbol{\phi}_t}$ is calculated on data augmented with $\mathcal{U}_t$ to update $\boldsymbol{\phi}_t$, and loss $\mathcal{L}_{\boldsymbol{\theta}_t}$ is calculated on data augmented with $\mathcal{S}_t$ to update $\boldsymbol{\theta}_t$. We repeat this process for all tasks in the sequence.

The augmentation procedure of AugMix is illustrated in Figure 2 (b). The optimization objective of AugMix is shown in the following:

$$\boldsymbol{\theta}_* = \arg\min_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{op\in\mathcal{O}} \mathcal{L}(\boldsymbol{x}, y, \boldsymbol{\theta}) + \lambda JS(\boldsymbol{x}, \boldsymbol{x}_{augmix1}, \boldsymbol{x}_{augmix2}), \tag{2}$$

where $\boldsymbol{x}$ is the original data and y is the data label; $\mathcal{L}(\boldsymbol{x}, y, \boldsymbol{\theta})$ is the loss function on the labeled data (cross-entropy for classification); $\boldsymbol{x}_{augmix1}$ and $\boldsymbol{x}_{augmix2}$ are two augmentations of original data $\boldsymbol{x}$ by AugMix; JS denotes the Jensen Shannon divergence for regularizing the output consistency between the original and augmented data. We will define it in the following section.

3.2. MetaMix Approach

Direct extension of AugMix [22] for CL is to augment data examples sampled from the memory buffer, and the received new data at each time point for memory-based method across the sequentially arrived tasks. In the following, we present our methods for memory-based CL methods, but our methods can be easily extended to other families of CL methods as well, also demonstrated in Section 4. However, since the memory buffer is usually small for CL models, using a fixed random augmentation strategy (like AugMix) may make the CL model gradually memorize the training data corruption operations, but fail to generalize to unseen corruptions during testing. Moreover, with the non-stationary nature of the data, the optimal augmentation strategy for image mixing is data-dependent, and the mixing weights should be specifically optimized for each task. However, since we cannot repeatedly tune the mixing weights for each task during CL learning process, an automatic self-adaptive process is necessary to achieve good performance. We thus introduce the details of our self-adaptive data augmentation approach, namely MetaMix, for improving robustness in CL. Figure 2 (c) illustrates the training process of MetaMix at time t. Specifically, we adopt meta-learning to learn a set of data-dependent parameters $\boldsymbol{w}_t$ and m_t, which can dynamically control the mixing weights of augmentation operations and automatically adapt to the non-stationary data distribution.

First, to memorize previous knowledge, we allocate a small memory buffer $\mathcal{M}$ to store data from previous tasks by reservoir sampling, similar to [35]. The data in the memory buffer will be replayed later with a new task together. Suppose we have M augmentation operations $\mathcal{O} = \{op_1, op_2, \cdots, op_M\}$ during training, CL model parameters at time t are $\boldsymbol{\theta}_t$, and the function represented by the network with $\boldsymbol{\theta}_t$ is $f_{\boldsymbol{\theta}_t}(\boldsymbol{x})$.

To capture the sequential dependencies among previous tasks, we design a self-adaptive LSTM mixing strategy. The LSTM inputs the current context information and outputs the adaptive mixing parameters. We use an LSTM (dynamic) model to dynamically change the mixing weights instead of an MLP (static) model since the tasks in CL sequentially arrive. The LSTM can encode the information of previous tasks into an additional hidden state while the MLP can only capture the data information of the current task. It does not capture the information of previous tasks without the hidden state. Formally, the LSTM adaptive process is defined as:

$$\boldsymbol{o}_{t+1}, \boldsymbol{h}_{t+1}, \boldsymbol{g}_{t+1} = \mathrm{LSTM}_{\boldsymbol{\phi}_t}(\boldsymbol{I}_t, \boldsymbol{h}_t, \boldsymbol{g}_t), \tag{3}$$

where ϕ_t is the LSTM MetaMixer parameters at time t, $\boldsymbol{h}_t$ is the hidden state of LSTM at time t, $\boldsymbol{g}_t$ is the cell state of LSTM at time t, $\boldsymbol{I}_t$ is the context information encoding as input to LSTM at time t; $\boldsymbol{o}_{t+1}$ is the output of LSTM at time $t+1$ for the mixing parameters, i.e., $\boldsymbol{o}_{t+1} = \{\boldsymbol{w}'_t, m'_t\}$, where m'_t is the mixing weight for the augmented data and original raw data and $\boldsymbol{w}'_t$ are the mixing weights for different chains in the augmentation. We apply softmax function to $\boldsymbol{w}'_t$ and sigmoid function to m'_t to ensure they are within [0,1] interval, i.e.,

$$m_t = \text{Sigmoid}(m'_t), \boldsymbol{w}_t = \text{Softmax}(\boldsymbol{w}'_t). \tag{4}$$

The context $\boldsymbol{I}_t$ contains essential information to characterize data in current and previous tasks. It is shown as:

$$\boldsymbol{I}_t = \{\nabla\mathcal{L}_{\boldsymbol{\theta}_t}(\boldsymbol{x}_r, y_r) \cdot \nabla\mathcal{L}_{\boldsymbol{\theta}_t}(\boldsymbol{x}_t, y_t), \mathcal{L}_{\boldsymbol{\theta}_t}(\boldsymbol{x}_r, y_r), \boldsymbol{e}_r, \boldsymbol{e}_t\}, \tag{5}$$

where $(\boldsymbol{x}_r, y_r)$ is the labeled mini-batch data sampled from the memory buffer and $(\boldsymbol{x}_t, y_t)$ is the labeled data received at time t. We adopt this information for several reasons: (1) $\nabla\mathcal{L}_{\boldsymbol{\theta}_t}(\boldsymbol{x}_r, y_r) \cdot \nabla\mathcal{L}_{\boldsymbol{\theta}_t}(\boldsymbol{x}_t, y_t)$ is the gradient dot product between memory data gradient and current mini-batch data gradient. It determines the degree of interference and transfer between memory data and current task mini-batch data. It is commonly used in existing CL as context information [18, 35]; (2) we also input the memory mini-batch data loss, as it indicates the degree of forgetting of previous tasks; (3) $\boldsymbol{e}_r$ and $\boldsymbol{e}_t$ are data encoding for mini-batch data, $\boldsymbol{x}_r$, sampled from memory buffer and currently received new mini-batch data, $\boldsymbol{x}_t$, respectively. That is, $\boldsymbol{e}_r$ and $\boldsymbol{e}_t$ are the last layer features outputted by ResNet-18. This determines the data-dependent features for calculating the data-dependent mixing weights.

Since corruption operations on testing data are unseen during training, we cannot directly optimize the augmentation strategy for these corruptions. To make the CL model generalize better to unseen corruptions, at each training step t, we randomly split the augmentation operations $\mathcal{O}$ into non-overlapping pseudo-seen $\mathcal{S}_t$ and pseudo-unseen $\mathcal{U}_t$ operations, i.e., $\mathcal{S}_t \cup \mathcal{U}_t = \mathcal{O}$ and $\mathcal{S}_t \cap \mathcal{U}_t = \emptyset$. We use the pseudo operation sets to simulate the scenario in which training and testing corruptions have no overlap. We formulate MetaMix as a bi-level optimization problem to make the CL model learn to augment and mix. This ensures that the update of model parameters $\boldsymbol{\theta}$ on the pseudo-seen set can improve the optimization of MetaMixer parameters $\boldsymbol{\phi}$ on the pseudo-unseen set, which could, in turn, be beneficial to improve the CL model generalization. We formulate the MetaMix as the following bi-level optimization:

$$\boldsymbol{\phi}_* = \arg\min_{\boldsymbol{\phi}} \mathop{\mathbb{E}}_{op\in\mathcal{U}_t} \mathcal{L}(\boldsymbol{x}_b, y_b, \boldsymbol{\theta}_*, \boldsymbol{\phi}) + \lambda JS(\boldsymbol{x}_b, \widehat{\boldsymbol{x}}_{b1}, \widehat{\boldsymbol{x}}_{b2}), \tag{6}$$

$$\boldsymbol{\theta}_* = \arg\min_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{op\in\mathcal{S}_t} \mathcal{L}(\boldsymbol{x}_b, y_b, \boldsymbol{\theta}, \boldsymbol{\phi}) + \lambda JS(\boldsymbol{x}_b, \boldsymbol{x}'_{b1}, \boldsymbol{x}'_{b2}),$$

where we concatenate the mini-batch data $(\boldsymbol{x}_r, y_r)$ from the memory buffer and the current received mini-batch data $(\boldsymbol{x}_t, y_t)$ together as $(\boldsymbol{x}_b, y_b)$; $\widehat{\boldsymbol{x}}_{b1}$ and $\widehat{\boldsymbol{x}}_{b2}$ are the two mini-batch data augmented by applying the pseudo-unseen augmentation operations $\mathcal{U}_t$ on $(\boldsymbol{x}_b, y_b)$; $\boldsymbol{x}'_{b1}$ and $\boldsymbol{x}'_{b2}$ are the two mini-batch data augmented by applying the pseudo-seen augmentation operations $\mathcal{S}_t$ on $(\boldsymbol{x}_b, y_b)$. λ is the regularization constant for the consistency loss. The lower-level optimization is to optimize the performance on the pseudo-seen augmentation operations $\mathcal{S}_t$ at the training step t; and the upper-level optimization is to optimize the generalization on the pseudo-unseen augmentation operations $\mathcal{U}_t$. This bi-level optimization for learning to augment and mix can be efficiently solved by first-order gradient descent method, similar to the first-order MAML [15].

The advantage of MetaMix (Eq. (6)) over AugMix (Eq. (2)) are two folds. First, MetaMix improves the generalization to unseen corruptions since we split the training data augmentation operations into pseudo-seen and unseen operations. This is similar to the train-validation data split in standard meta-learning, which improves the generalization compared to the training without data split [39]; It could be understood as introducing an implicit regularization so that it can benefit from low sample complexity [39]. Second, MetaMix can automatically adapt to the non-stationary data distribution in CL, while AugMix can only work well in the stationary data distribution in a single task.

Consistency Regularization To ensure the network output consistency between the original raw image data and the augmented data, we use a similar consistency loss [22] to regularize the network output to ensure smoother neural network responses. We hope the model to have similar responses to $\boldsymbol{x}_b, \widehat{\boldsymbol{x}}_{b1}, \widehat{\boldsymbol{x}}_{b2}$. We minimize the Jensen-Shannon divergence among the posterior distributions of the original sample $\boldsymbol{x}_b$ and its augmented variants $\widehat{\boldsymbol{x}}_{b1}$ and $\widehat{\boldsymbol{x}}_{b2}$. The consistency loss is defined as below:

$$p_{mean} = (p_{\boldsymbol{x}_b} + p_{\widehat{\boldsymbol{x}}_{b1}} + p_{\widehat{\boldsymbol{x}}_{b2}})/3 \tag{7}$$

$$JS(\boldsymbol{x}_b, \widehat{\boldsymbol{x}}_{b1}, \widehat{\boldsymbol{x}}_{b2}) = (\mathbb{KL}(p_{\boldsymbol{x}_b}|p_{mean}) + \mathbb{KL}(p_{\widehat{\boldsymbol{x}}_{b1}}|p_{mean}) + \mathbb{KL}(p_{\widehat{\boldsymbol{x}}_{b2}}|p_{mean}))/3 \tag{8}$$

Where $\mathbb{KL}$ denotes the KL divergence between two distributions. $p_{\boldsymbol{x}_b} = f_{\boldsymbol{\theta}_t}(\boldsymbol{x}_b)$ is the network output probabilities of each class for original raw data $\boldsymbol{x}_b$. Similarly, we can define $p_{\widehat{\boldsymbol{x}}_{b1}} = f_{\boldsymbol{\theta}_t}(\widehat{\boldsymbol{x}}_{b1})$ and $p_{\widehat{\boldsymbol{x}}_{b2}} = f_{\boldsymbol{\theta}_t}(\widehat{\boldsymbol{x}}_{b2})$. We summarize our proposed algorithm in Algorithm 3. Line 5-8 takes the context information as input and computes desired augmentation mixing parameters. Line 9-11 augments the original data with the pseudo-seen augmentation operations $\mathcal{S}_t$ and pseudo-unseen operations $\mathcal{U}_t$ by Algorithm 1. Line 12 solves the bi-level optimization problem (Eq. (6)) by Algorithm 2.

Algorithm 1 DataMix

Function DataMix ($\boldsymbol{x}_b, \boldsymbol{w}_t, m_t, \mathcal{V}$)
MetaMix augmented images initialized as $\boldsymbol{x}'_b = 0$
$\boldsymbol{x}_{orig} = \boldsymbol{x}_b$
for $i = 1$ to A **do** { A is the number of augmentation chains}
  $\boldsymbol{x}_b = \boldsymbol{x}_{orig}$
  for $l = 1$ to d **do** { d is the augmentation depth}
    randomly sample operation $op \in \mathcal{V}$ ($\mathcal{V}$ is the collection of augmentation operations)
    $\boldsymbol{x}_b = op(\boldsymbol{x}_b)$
  end for
  $\boldsymbol{x}'_b += \boldsymbol{w}_t^i \cdot \boldsymbol{x}_b$
end for
$\boldsymbol{x}'_b = m_t \boldsymbol{x}_{orig} + (1 - m_t)\boldsymbol{x}'_b$;
similar to the above procedure, we can get two different mini-batch $\boldsymbol{x}'_{b1}, \boldsymbol{x}'_{b2}$
return $\boldsymbol{x}'_{b1}, \boldsymbol{x}'_{b2}$
EndFunction

Algorithm 2 Bi-level Solver

Function Bi-level Solver($\boldsymbol{\theta}_t, \boldsymbol{\phi}_t, \boldsymbol{x}_b, y_b, \boldsymbol{x}'_{b1}, \boldsymbol{x}'_{b2}, \widehat{\boldsymbol{x}}_{b1}, \widehat{\boldsymbol{x}}_{b2}$)
optimize the *lower-level* optimization problem in Eq. (6) by $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \gamma \nabla_{\boldsymbol{\theta}} \mathbb{E}_{op \in \mathcal{S}_t}[\mathcal{L}(\boldsymbol{x}_b, y_b, \boldsymbol{\theta}_t, \boldsymbol{\phi}) + \lambda JS(\boldsymbol{x}_b, \boldsymbol{x}'_{b1}, \boldsymbol{x}'_{b2})]$
calculate the *upper-level* optimization loss function in Eq. (6) by $\mathcal{E}(\boldsymbol{\theta}_{t+1}, \boldsymbol{\phi}) = \mathbb{E}_{op \in \mathcal{U}_t}[\mathcal{L}(\boldsymbol{x}_b, y_b, \boldsymbol{\theta}_{t+1}, \boldsymbol{\phi}) + \lambda JS(\boldsymbol{x}_b, \widehat{\boldsymbol{x}}_{b1}, \widehat{\boldsymbol{x}}_{b2})]$
$\boldsymbol{\phi}'_1 = \boldsymbol{\phi}_t$
for $s = 1$ to J **do**
  $\boldsymbol{\phi}'_{s+1} = \boldsymbol{\phi}'_s - \beta \nabla_{\boldsymbol{\phi}'} \mathcal{E}(\boldsymbol{\theta}_{t+1}, \boldsymbol{\phi}'_s)$
end for
$\boldsymbol{\phi}_{t+1} = \boldsymbol{\phi}'_{J+1}$
return $\boldsymbol{\theta}_{t+1}, \boldsymbol{\phi}_{t+1}$
EndFunction

Algorithm 3 MetaMix

REQUIRE: **Augmentation Operations** $\mathcal{O} = \{osterize, rotate, \cdots, solarize\}$; augmentation width A; augmentation depth d; the number of CL tasks N; the number of training iterations for each task $\mathcal{T}_k$ is N_k; memory buffer $\mathcal{M}$; CL model parameters $\boldsymbol{\theta}$; LSTM MetaMix with parameters $\boldsymbol{\phi}$; MetaMixer LSTM update steps J; CL model parameter learning rate γ; regularization weight λ; original image $\boldsymbol{x}_{orig}$; initial hidden state and cell state $\boldsymbol{h}_t, \boldsymbol{g}_t$ are initialized as random noise $\boldsymbol{h}_1, \boldsymbol{g}_1 \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
for $k = 1$ to N **do**
  for $t = 1$ to N_k **do**
    randomly sample mini-batch data $(\boldsymbol{x}_r, y_r)$ from memory buffer $\mathcal{M}$ and concatenate it with the current received mini-batch data $(\boldsymbol{x}_t, y_t)$ to obtain $(\boldsymbol{x}_b, y_b)$
    compute context vector $\boldsymbol{I}_t = \{\nabla \mathcal{L}_{\boldsymbol{\theta}_t}(\boldsymbol{x}_r, y_r) \cdot \nabla \mathcal{L}_{\boldsymbol{\theta}_t}(\boldsymbol{x}_t, y_t), \mathcal{L}_{\boldsymbol{\theta}_t}(\boldsymbol{x}_r, y_r), \boldsymbol{e}_r, \boldsymbol{e}_t\}$
    $\boldsymbol{o}_{t+1}, \boldsymbol{h}_{t+1}, \boldsymbol{g}_{t+1} = \mathrm{LSTM}_{\boldsymbol{\phi}_t}(\boldsymbol{I}_t, \boldsymbol{h}_t, \boldsymbol{g}_t)$
    generate mixing parameters: $\boldsymbol{w}'_t, m'_t = \boldsymbol{o}_{t+1}$;
    $m_t = \mathrm{Sigmoid}(m'_t), \boldsymbol{w}_t = \mathrm{Softmax}(\boldsymbol{w}'_t)$
    randomly split augmentation operations $\mathcal{O}$ into non-overlapping pseudo-seen $\mathcal{S}_t$ and unseen $\mathcal{U}_t$
    $\boldsymbol{x}'_{b1}, \boldsymbol{x}'_{b2}$ = DataMix ($\boldsymbol{x}_b, \boldsymbol{w}_t, m_t, \mathcal{S}_t$) (Algorithm 1)
    $\widehat{\boldsymbol{x}}_{b1}, \widehat{\boldsymbol{x}}_{b2}$ = DataMix ($\boldsymbol{x}_b, \boldsymbol{w}_t, m_t, \mathcal{U}_t$) (Algorithm 1)
    $\boldsymbol{\theta}_{t+1}, \boldsymbol{\phi}_{t+1}$ = Bi-level Solver ($\boldsymbol{\theta}_t, \boldsymbol{\phi}_t, \boldsymbol{x}_b, y_b, \boldsymbol{x}'_{b1}, \boldsymbol{x}'_{b2}, \widehat{\boldsymbol{x}}_{b1}, \widehat{\boldsymbol{x}}_{b2}$) (Algorithm 2)
    update memory buffer $\mathcal{M}$ through reservoir sampling [35] to determine whether to store $(\boldsymbol{x}_t, y_t)$
  end for
end for

4. Experiment

We evaluate the model performance for both task- and class-continual learning. The CL models are evaluated on the proposed benchmark for each dataset. We then describe experiment setup, results, and ablation study.

4.1. Experiment Setup

Below, we first construct the corruption CL benchmark and then describe the evaluation metrics, baselines, and implementation details.

Corruption CL benchmark. Existing CL benchmarks, such as Split-CIFAR10, Split-CIFAR100, and Split-miniImageNet, are commonly used in literature. However, each task's training and testing data follow the same distribution for these datasets. Thus, these benchmarks are not suitable for evaluating the corruption robustness of CL methods. To properly evaluate the corruption robustness, we follow the protocols [21] and create several new CL datasets. We apply 15 common corruption operations to the testing data of the three CL benchmarks, such as snow, frost, fog, etc. We provide a brief description for each corruption operation in Table 5 in Appendix. We perform each corruption at five different severity levels. *Note that these corruptions only appear on the testing dataset of each continually learned task and are not seen in the training data of each task.* This is to ensure that we can measure the robustness to *unseen* corruptions during testing for each compared method. To our best knowledge, we build the first CL corruption benchmark for measuring the generalization of CL models to *unseen* corruptions during testing. We name these new datasets Split-CIFAR10-C, Split-CIFAR100-C, and Split-miniImageNet-C. Specifically, for **Split-CIFAR-10-C**, we split the CIFAR-10 dataset [28] into 5 disjoint tasks, where each task has 2 classes; for **Split-CIFAR-100-C**, we split CIFAR-100 [28] consisting of 100 classes into 10 disjoint tasks, where each task has 10 classes; for **Split-miniImageNet-C**, we split miniImageNet [44] that consists of 100 classes, into 10 disjoint tasks, where each task has 10 classes.

Evaluation Metrics. We evaluate the performance of different methods with average accuracy and backward transfer at the end of CL training to measure the final performance

and the extent of forgetting for different methods. For each task, the accuracy for a specific corruption c is the average accuracy over the five severity level. Suppose for task k, $ACC^k_{c,s}$ measures the robust accuracy after applying the corruption operation c with severity level s:

$$ACC^k_{c,s} = \mathbb{E}_{(\boldsymbol{x}^k, y^k, \mathcal{T}_k) \sim D^{te}_k} \mathbb{I}(f(c_s(\boldsymbol{x}^k)) = y^k),$$

where $c_s(\boldsymbol{x}^k)$ is the corruption operation c with severity level s applied on $\boldsymbol{x}^k$. The accuracy for corruption c denoted as ACC^k_c is the average robust accuracy across all the five severity levels. The overall accuracy for task k, ACC^k, is the average accuracy across all corruption types, i.e.,

$$ACC^k_c = \frac{1}{5}\sum_{s=1}^{s=5} ACC^k_{c,s}, \text{ and } ACC^k = \frac{1}{|\mathcal{C}|}\sum_{c \in \mathcal{C}} ACC^k_c.$$

To measure catastrophic forgetting, we also evaluate the backward transfer (BWT), which measures the extent of forgetting on previous tasks after learning new ones. $BWT < 0$ reveals the occurrence of catastrophic forgetting on previous tasks, and $BWT > 0$ indicates that learning new tasks is helpful for previous tasks. Formally, the backward transfer (BWT) is defined as: $BWT = \frac{1}{N-1}\sum_{k=1}^{k=N-1}(R_{N,k} - R_{k,k})$; where $R_{N,k}$ is the testing accuracy on task k after learning on task N and $R_{N,k}$ is defined the same as the above mentioned ACC^k. For BWT, the higher, the better.

Continual Learning Setting. We apply our proposed MetaMix to task-continual learning (Task-CL) and class-continual learning (Class-CL). The former provides task identities for the CL learner to select the relevant classifier for each example during testing, whereas the latter does not.

Baseline. Our method can be seamlessly integrated with exiting CL methods. For illustration, we integrate the proposed methods with SOTA memory-based methods: *DER++* [5] and CLS-ER [2]. To evaluate the effectiveness of the proposed MetaMix, we compare it with several SOTA data augmentation approaches originally designed for traditional supervised learning: *Adversarial Training* (AT) [33], *RandAugment* (RA) [11], *Maxup* [16], *DeepAugment* [20], *Augmix* [22]. We adapt these augmentation methods for CL problems by applying them on the new task data and the memory buffer data. We provide more detailed descriptions of baselines in Appendix. Due to space limitations, we put the experiments that integrate our proposed method with CLS-ER in Appendix.

Implementation details. We use ResNet18 [19] as classifier for all datasets. All the other hyperparameter settings follow from [5]. *The types of corruption during testing are not seen during training for all the compared augmentation methods to evaluate the robustness against unseen corruptions*. Following [22], the augmentation operations performed during training are autocontrast, equalize, posterize, rotate, solarize, shear-x, shear-y, translate-x, translate-y. All the above-compared data augmentation methods apply these augmentation operations to CL baseline method to improve robustness. The memory buffer has capacity of 500 data points by default. We average the result for 5 runs for each experiment. We provide both mean and standard deviation across different runs. Due to space limitations, we provide standard deviation results in Appendix. We provide more implementation details in Appendix.

LSTM Architecture. It has one recurrent layer with 5 hidden units. A linear layer (fully connected layer) is appended next to the LSTM output to generate the mixing parameters $\boldsymbol{w}_t$ and m_t. The LSTM only has about 20K parameters and is negligible compared to ResNet18, which has more than 11.22 million parameters.

4.2. Robustness of Task-CL

Task-Continual Learning In this section, we evaluate the corruption robustness under task-continual learning, where the task identifiers are provided to the CL learner during testing. We present the results that integrate baselines and the proposed method with DER++ [5]. Due to space limitations, we put the experiments that integrate baselines and MetaMix with CLS-ER [2] in Appendix.

Results. We show the results across different corruptions and compared methods in Tables 1-2; where *Avg* is the average robust accuracy across all the corruptions. Due to the space limitations, we provide the standard deviation across different runs in Table 8 and 9 in Appendix. We also put results on split-CIFAR10-C in Table 6 and Table 7 in Appendix. We can observe that without data augmentation, the performance of DER++ is close to random guessing, indicating that the CL model does not generalize to common corruptions. The proposed self-adaptive MetaMix brings significant improvement toward corruption robustness. It outperforms the average corruption robustness accuracy of the other models by a large margin of $2.4\%, 2.6\%$ on Split-CIFAR100-C, and Split-miniImageNet-C, respectively. The improvement shows the effectiveness of proposed MetaMix, which can automatically adapt to the non-stationary data distribution. The compared methods only use fixed static/random augmentation methods which lack adaptation to the non-stationary data in CL, and thus do not perform well. Random augmentation achieves second-best results. Deepaugment would distort the image such that they are very different from original images. Adversarial training and Maxup do not help much corruption robustness among the compared methods. Adversarial training can generate similar images compared to original images. Thus, there is a large gap between training and testing corrupted data for those methods adapted for CL. Maxup aims for improving clean data performance, not designed for robust accuracy.

Table 1. Robust accuracy of **Task-CL** on **Split-CIFAR100-C** with **DER++**

Method	Noise			Blur				Weather				Digital				
	Gauss	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	JPEG	**Avg**
DER++	10.46	10.69	10.26	10.76	10.51	10.71	10.66	10.47	10.56	10.51	10.79	10.54	10.75	10.55	10.72	10.60
AT	10.36	10.45	10.25	10.23	10.24	10.27	10.25	10.12	9.88	9.98	10.16	9.83	10.19	10.3	10.31	10.19
RA	26.77	27.15	25.46	27.84	23.46	24.05	25.99	23.26	26.34	30.45	27.95	28.8	25.54	29.12	29.09	26.75
DeepAugment	11.00	11.16	10.86	11.35	11.15	11.45	11.33	10.76	10.46	10.59	11.10	10.57	11.20	11.36	11.25	11.04
Maxup	11.1	11.17	11.13	11.09	11.06	11.13	11.1	10.88	10.76	10.81	10.86	10.37	11.06	11.13	11.12	10.98
Augmix	59.58	62.68	59.92	67.95	58.88	65.93	67.03	63.39	63.93	63.24	68.62	63.27	65.82	66.06	64.07	64.02
Ours	**61.53**	**65.14**	**61.18**	**70.35**	**61.28**	**68.01**	**68.76**	**66.02**	**66.49**	**66.14**	**71.49**	**66.16**	**68.24**	**68.91**	**66.75**	**66.43**

Table 2. Robust accuracy of **Task-CL** on **Split-MiniImageNet-C** with **DER++**

Method	Noise			Blur				Weather				Digital				
	Gauss	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	JPEG	**Avg**
DER++	10.27	10.1	10.07	9.78	9.73	9.67	9.72	9.61	9.71	9.85	9.69	10.15	9.55	9.45	9.57	9.79
AT	9.07	9.24	9.27	9.65	9.62	9.6	9.48	9.17	9.10	9.01	9.12	8.38	9.54	9.56	9.35	9.28
RA	17.23	18.22	16.7	19.81	16.6	18.32	19.07	18.94	17.14	18.26	17.26	14.56	21.14	20.84	23.0	18.47
DeepAugment	10.54	10.53	10.62	10.68	10.70	10.63	10.74	10.56	10.32	10.32	10.52	9.92	10.55	10.41	10.49	10.50
Maxup	10.4	10.9	10.68	10.65	9.75	10.21	10.25	9.62	10.12	10.21	10.34	10.9	10.2	10.16	10.07	10.30
Augmix	43.45	48.91	42.20	53.78	49.68	59.88	51.83	53.22	54.02	55.63	59.59	**43.46**	61.35	**51.34**	62.75	52.74
Ours	**48.8**	**53.33**	**43.81**	**59.7**	**53.64**	**62.78**	**56.95**	**56.19**	**57.31**	**56.08**	**62.07**	41.71	**63.17**	49.78	**64.55**	**55.32**

Table 3. Robust accuracy of **Class-CL** on **Split-CIFAR100-C** with **DER++**

Method	Noise			Blur				Weather				Digital				
	Gauss	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	JPEG	**Avg**
DER++	0.75	0.75	0.88	0.75	0.86	0.77	0.68	0.88	0.96	0.92	0.75	0.96	0.78	0.78	0.79	0.82
AT	0.99	0.97	1.04	0.91	0.95	0.97	0.92	0.83	1.05	0.94	0.92	1.0	0.91	0.95	0.93	0.95
RA	6.22	6.3	5.67	6.25	4.42	4.54	5.59	5.25	6.24	7.72	6.42	6.93	5.16	6.81	7.11	6.04
DeepAugment	1.29	1.29	1.25	1.26	1.26	1.23	1.24	1.26	1.24	1.24	1.14	1.26	1.22	1.28	1.33	1.25
Maxup	1.17	1.15	1.16	1.15	1.08	1.14	1.12	1.08	1.08	1.09	1.14	1.06	1.13	1.15	1.11	1.12
Augmix	19.57	21.21	20.35	26.18	19.08	25.17	25.46	22.09	22.43	22.82	26.44	22.54	23.72	24.64	23.01	22.98
Ours	**22.32**	**24.44**	**23.15**	**29.19**	**22.13**	**28.10**	**28.35**	**25.47**	**25.2**	**26.08**	**29.56**	**25.78**	**27.01**	**28.16**	**25.82**	**26.05**

4.3. Robustness of Class-CL

Class-Continual Learning. In this section, we evaluate the corruption robustness under the more challenging class-continual learning setting, where the task identifiers are unavailable to the CL learner during testing.

Results. We present the results across different corruptions and compared methods on Split-CIFAR100-C in Table 3, we present the results on Split-miniImageNet-C in Table 10 in Appendix. Similar to the Task-CL, MetaMix significantly improves over the baseline augmentation methods in Class-CL setting due to its self-adaptivity property. It outperforms the average corruption robustness accuracy of the other models by a large margin of 3.1%, 2.2% on Split-CIFAR100-C, and Split-miniImageNet-C, respectively. This is because our proposed method can automatically adapt to the non-stationary distributions. Most compared methods do not perform well on this challenging setup.

Due to space limitations, we put BWT results in Table 15 in Appendix. In the corruption-robustness scenario, BWT is no longer a meaningful metric with such extreme accuracy differences, as the significantly lower accuracy of comparison methods results in much less space for further performance variations during backward transfer.

4.4. Ablation Study and Hyperparameter Analysis

Due to the limited space, we provide (1) ablation studies; (2) hyperparameter analyses, including λ, J, β, etc.; (3) The effect of using LSTM vs MLP and the benefit of using additional hidden state information from previous tasks; (4) computation cost in Appendix.

Effect of Memory Size. We evaluate the effect of memory size with 500 and 3000, respectively. The memory size of 500 is the default setting in the above tables. We provide experiment results on Split-CIFAR100-C, Split-miniImageNet-C with memory size 3000 on task-CL and class-CL respectively in Table 11, 12, 13, 14 in Appendix. In these cases, our method significantly outperforms baselines.

5. Conclusion

This paper tackles a more challenging problem of corruption robustness in CL with non-stationary data distribution, where the testing data distribution may significantly differ from training data distribution with various forms of common corruptions. We propose a meta-learning framework, MetaMix, for self-adaptive data augmentation specialized for CL. Our proposed MetaMix can substantially improve the model's robustness. Comprehensive experiments on both task- and class-CL settings demonstrate the effectiveness of the proposed method.

Acknowledgement We thank all the anonymous reviewers for their insightful and thoughtful comments. This research was supported in part by grant NSF III-1910492.

References

[1] Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. *Advances in Neural Information Processing Systems 32*, pages 11849–11860, 2019. 1, 2

[2] Elahe Arani, Fahad Sarfraz, and Bahram Zonooz. Learning fast, learning slow: A general continual learning method based on complementary learning system. In *International Conference on Learning Representations*, 2022. 7, 12, 17

[3] Jihwan Bang, Hyunseo Koh, Seulki Park, Hwanjun Song, Jung-Woo Ha, and Jonghyun Choi. Online continual learning on a contaminated data stream with blurry task boundaries. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9265–9274, 2022. 3

[4] E. Belouadah and A. Popescu. Il2m: Class incremental learning with dual memory. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 583–592, 2019. 2

[5] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *34th Conference on Neural Information Processing Systems*, 2020. 1, 7, 12

[6] Dan A. Calian, Florian Stimberg, Olivia Wiles, Sylvestre-Alvise Rebuffi, Andras Gyorgy, Timothy Mann, and Sven Gowal. Defending against image corruptions through adversarial augmentations. *https://arxiv.org/abs/2104.01086*, 2021. 3

[7] Sungmin Cha, Hsiang Hsu, Taebaek Hwang, Flavio Calmon, and Taesup Moon. {CPR}: Classifier-projection regularization for continual learning. In *International Conference on Learning Representations*, 2021. 2

[8] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *Proceedings of the International Conference on Learning Representations*, 2019. 2

[9] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. Continual learning with tiny episodic memories. *https://arxiv.org/abs/1902.10486*, 2019. 2

[10] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[11] Ekin Dogus Cubuk, Barret Zoph, Jon Shlens, and Quoc Le. Randaugment: Practical automated data augmentation with a reduced search space. *Advances in Neural Information Processing Systems*, 2020. 1, 3, 7, 12

[12] Terrance DeVries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *https://arxiv.org/abs/1708.04552*, 2017. 3

[13] Sayna Ebrahimi, Mohamed Elhoseiny, Trevor Darrell, and Marcus Rohrbach. Uncertainty-guided continual learning with bayesian neural networks. *Proceedings of the International Conference on Learning Representations*, 2020. 2

[14] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *https://arxiv.org/abs/1701.08734*, 2017. 2

[15] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *International Conference on Machine Learning*, 2017. 5

[16] Chengyue Gong, Tongzheng Ren, Mao Ye, and Qiang Liu. Maxup: Lightweight adversarial training with data augmentation improves neural network training. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3, 7, 12

[17] Yanan Gu, Xu Yang, Kun Wei, and Cheng Deng. Not just selection, but exploration: Online class-incremental continual learning via dual view consistency. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[18] Gunshi Gupta, Karmesh Yadav, and Liam Paull. La-maml: Look-ahead meta learning for continual learning. In *Advances in Neural Information Processing Systems*, 2020. 5

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016. 7

[20] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *International Conference on Computer Vision*, 2021. 3, 7, 12

[21] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019. 1, 6

[22] Dan Hendrycks, Norman Mu, Ekin Dogus Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple method to improve robustness and uncertainty under data shift. 2020. 2, 3, 4, 5, 7, 12

[23] Christian Henning, Maria Cervera, Francesco D'Angelo, Johannes Von Oswald, Regina Traber, Benjamin Ehret, Seijin Kobayashi, Benjamin F Grewe, and Joao Sacramento. Posterior meta-replay for continual learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. 2

[24] Hyundong Jin and Eunwoo Kim. Helpful or harmful: Inter-task association in continual learning. In *European Conference on Computer Vision*, 2022. 2

[25] Chris Dongjoo Kim, Jinseo Jeong, Sangwoo Moon, and Gunhee Kim. Continual learning on noisy data streams via self-purified replay. *International Conference on Computer Vision*, 2021. 3

[26] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural

networks. *Proceedings of the national academy of sciences*, 2017. 2

[27] Yajing Kong, Liu Liu, Zhen Wang, and Dacheng Tao. Balancing stability and plasticity through advanced null space in continual learning. In *European Conference on Computer Vision*, 2022. 2

[28] Alex Krizhevsky. Learning multiple layers of features from tiny images. *Technical report*, 2009. 6

[29] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Conference on Empirical Methods in Natural Language Processing*, 2021. 2

[30] Sen Lin, Li Yang, Deliang Fan, and Junshan Zhang. TRGP: Trust region gradient projection for continual learning. In *International Conference on Learning Representations*, 2022. 1, 2

[31] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in Neural Information Processing Systems*, 2017. 1, 2

[32] Amrutha Machireddy, Ranganath Krishnan, Nilesh Ahuja, and Omesh Tickoo. Continual active adaptation to evolving distributional shifts. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3443–3449, 2022. 3

[33] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *Proceedings of the International Conference on Learning Representations*, 2019. 7, 12

[34] Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational continual learning. *Proceedings of the International Conference on Learning Representations*, 2018. 2

[35] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *International Conference on Learning Representations*, 2019. 2, 4, 5, 6

[36] Tim G. J. Rudner, Freddie Bickford Smith, Qixuan Feng, Yee Whye Teh, and Yarin Gal. Continual learning via sequential function-space variational inference. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 18871–18887, 17–23 Jul 2022. 1

[37] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *https://arxiv.org/abs/1606.04671*, 2016. 2

[38] Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representations*, 2021. 2

[39] Nikunj Saunshi, Arushi Gupta, and Wei Hu. A representation learning perspective on the importance of train-validation splitting in meta-learning. In *Proceedings of the 38th International Conference on Machine Learning*, pages 9333–9343, 2021. 5

[40] Jonathan Schwarz, Jelena Luketina, Wojciech M. Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress and compress: A scalable framework for continual learning. In *Proceedings of the International Conference on Machine Learning*, 2018. 2

[41] Joan Serrá, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *Proceedings of the International Conference on Machine Learning*, 2018. 2

[42] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017. 2

[43] Gido M. van de Ven and Andreas S. Tolias. Three scenarios for continual learning. 2019. 2

[44] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *Advances in neural information processing systems*, 2017. 6

[45] Liyuan Wang, Xingxing Zhang, Kuo Yang, Longhui Yu, Chongxuan Li, Lanqing HONG, Shifeng Zhang, Zhenguo Li, Yi Zhong, and Jun Zhu. Memory replay with data compression for continual learning. In *International Conference on Learning Representations*, 2022. 1

[46] Zhenyi Wang, Tiehang Duan, Le Fang, Qiuling Suo, and Mingchen Gao. Meta learning on a sequence of imbalanced domains with difficulty awareness. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8947–8957, 2021. 2

[47] Zhen Wang, Liu Liu, Yajing Kong, Jiaxian Guo, and Dacheng Tao. Online continual learning with contrastive vision transformer. In *European Conference on Computer Vision*, 2022. 2

[48] Zhenyi Wang, Li Shen, Tiehang Duan, Donglin Zhan, Le Fang, and Mingchen Gao. Learning to learn and remember super long multi-domain task sequence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7982–7992, 2022. 2

[49] Zhenyi Wang, Li Shen, Le Fang, Qiuling Suo, Tiehang Duan, and Mingchen Gao. Improving task-free continual learning by distributionally robust memory evolution. In *International Conference on Machine Learning*, pages 22985–22998, 2022. 2

[50] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. *European Conference on Computer Vision*, 2022. 2

[51] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 139–149, 2022. 2

[52] Mengqi Xue, Haofei Zhang, Jie Song, and Mingli Song. Meta-attention for vit-backed continual learning. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 150–159, 2022. 2

[53] Qingsen Yan, Dong Gong, Yuhang Liu, Anton van den Hengel, and Javen Qinfeng Shi. Learning bayesian sparse networks with full experience replay for continual learning. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 109–118, 2022. 2

[54] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *International Conference on Learning Representations*, 2018. 2

[55] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. *IEEE International Conference on Computer Vision*, 2019. 3

[56] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. *https://arxiv.org/abs/1703.04200*, 2017. 2

[57] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *International Conference on Learning Representations*, 2018. 3, 12