A Continual Deepfake Detection Benchmark: Dataset, Methods, and Essentials
(Supplementary Material)

Chuqiao Li\textsuperscript{1}, Zhiwu Huang\textsuperscript{2,*}, Danda Pani Paudel\textsuperscript{1}, Yabin Wang\textsuperscript{3,2}, Mohamad Shahbazi\textsuperscript{1}, Xiaopeng Hong\textsuperscript{4}, Luc Van Gool\textsuperscript{1,5}
\textsuperscript{1}ETH Zürich, Switzerland \textsuperscript{2}Singapore Management University, Singapore
\textsuperscript{3}Xi’an Jiaotong University, China \textsuperscript{4}Harbin Institute of Technology, China \textsuperscript{5}KU Leuven, Belgium
chuqli@student.ethz.ch, zhiwu.huang@gmail.com,
\{paudel, mshahbazi, vangool\}@vision.ee.ethz.ch, iamwangyabin@stu.xjtu.edu.cn, hongxiaopeng@ieee.org

Abstract

In the supplementary material, we further present (1) illustration of the suggested approaches for the adaptation of class incremental learning (CIL) methods to the continual deepfake detection problem (CDD), (2) training details of the evaluated methods on the proposed continual deepfake detection benchmark (CDDDB), (3) study on more essential components of CIL methods on the suggested CDDDB, (4) evaluation on the GANFake \cite{8} dataset, (5) experiments on the generalization capability of the suggested CIL methods to unseen domains and corrupted test images, and (6) more overviews of the suggested CDD evaluations.

1. Illustrations of Adapting CIL to CDD

In the main paper, we study three main adaptations of CIL methods for CDD. Fig.1 illustrates the suggested three approaches that adapts the exiting CIL methods to the context of CDD.

2. Training Settings

We evaluated three types of the state-of-the-art class incremental learning (CIL) methods on the suggested CDDDB. 

**Gradient-based Method.** We used the official code of the null space class incremental learning method (NSCIL) \cite{13}, which is one of the state-of-the-art gradient-based methods. We followed the official code’s setup to tune the hyperparameter $\lambda$ from the default setup $\{10, 30\}$ to the new settings $\{100, 1\}$, which is used to select the smallest singular values corresponding to the null space. The larger $\lambda$ leads to larger approximate null space, increasing the plasticity to learn new tasks while decreasing the memorization of old tasks \cite{13}. We trained the NSCIL model for 12 epochs in total, as we found that training more epochs for NSCIL resulted in performance degradation. The initial learning rate is 0.001 for the first task and 0.0001 for all other tasks and is divided by 2 after 4 and 8 epochs for the EASY evaluation. For batch normalization layers, the learning rate starts from $5 \times 10^{-5}$. The other parameters are the same as official NSCIL implementation.

**Memory-based Method.** We employed latent replay class incremental learning (LRCIL) \cite{10}, which is one of the most effective and efficient memory-based methods. LRCIL was trained in the NSCIL’s framework, with 20 epochs in total, the initial learning rate is 0.001 for the first task and 0.0001 for all other tasks and is divided by 2 after 10 and 15 epochs for the EASY, HARD, LONG evaluations. For batch normalization layers, the learning rate starts from $5 \times 10^{-5}$. For the GANfake evaluation, we used 60 epochs instead and the initial learning rate is divided by 2 after 20 and 40 epoch. Following original LRCIL’s implementation, there is no knowledge distillation loss. We also tried to add distillation loss with a factor $\gamma_d = 0.3$. And we chose the second layer as latent layer. We used Adam with the batch

*Corresponding Author
size 32 to train the network.

**Distillation-based Method.** We utilized five state-of-the-art distillation-based methods, i.e., incremental classifier and representation learning (iCaRL) [11], Learning a Unified Classifier Incrementally via Rebalancing (LUCIR) [6], Dynamic Token Expension (DyTox) [4], as well as Compositional Class Incremental Learning (CCIL) [9], which is additionally evaluated in the Supp. Material. Besides, we also evaluated the multi-task variant of iCaRL (iCaRL-SumLog) [8].

For iCaRL, we trained its model for 30 epochs in total. The learning rate starts from 0.0001 and is divided by $\frac{10}{3}$ after 10 and 20 epochs for the EASY, HARD and LONG evaluations. For GANfake sequences, we used 60 epochs instead and the initial learning rate 0.001 is divided by $\frac{10}{3}$ after 20 and 40 epoch. Following the original iCaRL’s implementation, $\gamma_d$ was set to 1 for knowledge distillation loss $\ell_{distill}$ with temperature as $T = 1$. We used Adam with the batch size 32 to train the network.

For CCIL, we used its official code\footnote{https://github.com/sud0301/essentials_for_CIL}. The initial learning rate, total epochs and learning decay rate are the same as iCaRL. We tried the method with ($\gamma_d = 0.3$) or without ($\gamma_d = 0$) knowledge distillation $\ell_{distill}$. Also, we tried it with or without applying mixup and label smoothing techniques to the whole training process which is different from the original implementation (only for the first training phase), and we followed the relevant parameters (e.g. mixup weight). We used Adam with the batch size 32 to train the network. The other parameters are the same as the official implementation of CCIL.

For LUCIR, the learning rate starts from 0.0001 and is divided by $\frac{10}{3}$ after 10 after 16 and 33 epochs (50 epochs in total) for the LONG evaluation. For the EASY, HARD and GANfake evaluations, we used 40 epochs and the learning rate is divided by 10 after 13 and 26 epoch. $\gamma_d$ is set to 0.5 for knowledge distillation loss $\ell_{distill}$. $\gamma_m$ was set to 0.1 for supplementary loss $\ell_{supp}$. Within $\ell_{supp}$, the number of closest class embeddings $J$ was set to 2 and the threshold $\tau$ was set to 0.2 for all experiments. The other parameters are the same as official LUCIR implementation. We used Adam with the batch size 32 to train the LUCIR network.

For DyTox [4], we used its official implementation\footnote{https://github.com/arthurdouillard/dytox} with the default settings on its hyperparameters. DyTox [4] originally suggests training its backbone from scratch. However, this case’s performance (about 56%) is much lower than that of training from the ImageNet pre-trained base-ConViT\footnote{We empirically find that the performance (AA=96.14%) of ImageNet pre-trained ConViT is very comparable to that (AA=96.29%) of the one that is further fine-tuned on ProGAN for the EASY1500 evaluation. Therefore, for the transformer-based networks, we suggest only using ImageNet pre-trained models without the further warm-up step on ProGAN.} on CDDB-Hard500. Therefore, we apply the pre-trained base-ConViT for DyTox throughout the paper.

For a fair comparison, except for DyTox that is based on an ImageNet pre-trained transformer ConViT [5], we used the state-of-the-art deepfake CNN detector (CNNDet) [14], which applies a ResNet-50 pretrained on ImageNet [3] and ProGAN [7] as the backbone for all the remaining methods over the proposed CDDB. For the multi-task (MT) learning variants of the evaluated CIL methods, the study on the trade-off hyperparameter $\lambda$ is presented in Table 1. In Table 1, we aim to compare all the colorized triplets (LRCIL, iCaRL, LUCIR)’s performances to see which triplets consistently perform well. Each triplet should be of the same MT variant and the same $\lambda$ value (e.g., LRCIL-SumLogit-0.3, iCaRL-SumLogit-0.3, LUCIR-SumLogit-0.3). By comparing these triplets in terms of their average performances on (LRCIL, iCaRL, LUCIR), we can see that the SumLogit case ($\lambda = 0.3$) works the most promisingly on the CDDDB-EASY1500 evaluation. Furthermore, it is not feasible to tune the hyperparameters on the test data in the real-world scenario. Accordingly, we used the same hyperparameter $\lambda = 0.3$ for all the MT variants of LRCIL, iCaRL and LUCIR throughout the main paper.

### 3. Study on More Essentials

Table 2 studies more essentials in CIL on CDDB. They are cosine normalization on linear fully connected (FC) layer, label smoothing and mixup techniques.

**Linear FC (LinFC).** The main paper finds that performing cosine normalization on FC (CosFC) layers result in LUCIR [6] and CCIL [9] is clearly inferior performance than using LinFC. This is because CosFC is originally designed to address the class imbalance issue, which however is much less serious in the CDD context. We additionally apply LinFC to CCIL* [9] that further drops knowledge distillation. We can see that the LinFC version of CCIL* also performs better than its corresponding CosFC version, enhancing the conclusion that CosFC is too hard for class imbalance based normalization so that it really hurts the CDD performance.

**Label Smoothing (LabelSM).** We also study LabelSM that is commonly used to improve generalization and reduce overconfidence of classification models [12, 9]. This technique affects the cross-entropy loss for classification by interpolating the one-hot labels with a uniform distribution over the possible classes [9]. From the results in Table 2, we can discover that LabelSM slightly improves the AA scores in the cases of iCaRL [11] and LUCIR [6], while hurting the AA scores in the case of LRCIL [10] and CCIL [9].

**Mixup.** We further employ Mixup that is used a form of data augmentation for better clarification in general. The mixup technique is to generate training samples by linearly combining pairs of training samples (i.e., images and labels). Following [9], we applied the mixup data to the training data and merely used them (no original training data).
4. Evaluation on the GANfake Dataset

For the GANfake [8] dataset, Table 3 summarizes the evaluation results of the four main evaluated methods, i.e., the variant of iCaRL method [8], our suggested SumLogit variants of LRCIL [10], iCaRL [11], and LUCIR [6]. The results show that the suggested MT variants mostly work better than the original one (i.e., SumLog). From the overall results, we can find that the highest AA score (i.e., 93.19) for the case of memory budget=1024 on GANfake is clearly higher than the one (i.e., 82.53) on our suggested CDBD-HARD1000 (memory budget=1000). Besides, the highest AA score (i.e., 91.88) for the case of memory size=512 on GANfake is much higher than the one (i.e., 80.77) on the suggested CDBD-EASY1500 (memory size=1500). This implies that our suggested CDBD is clearly more challenging than GANfake, and therefore we believe it will have a high potential to promote more solid research in the domain of continual deepfake detection.

5. Experiments on the Generalization Capability of Suggested Methods

The knowledge accumulation nature of the suggested CDBD allows for a better generalization to deepfakes from unseen domains. To verify this, we compare the adapted continual learning methods (e.g., LRCIL [10], iCaRL [11], LUCIR [6]) against CNNDet [14] (pre-trained on ProGAN) and CNNDet [14]-Finetune (starts from ProGAN to the

<table>
<thead>
<tr>
<th>Learning System</th>
<th>Evaluation Method</th>
<th>AA</th>
<th>AF</th>
<th>AA-M</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRCIL[10]</td>
<td>SumLog(8)</td>
<td>90.58</td>
<td>87.64</td>
<td>86.63</td>
</tr>
<tr>
<td>iCaRL[11]</td>
<td>SumLog(8)</td>
<td>92.75</td>
<td>90.31</td>
<td>87.57</td>
</tr>
<tr>
<td>LUCIR[6]</td>
<td>SumLog(8)</td>
<td>91.00</td>
<td>88.05</td>
<td>85.10</td>
</tr>
<tr>
<td>Multi-task (MT) learning</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>iCaRL[11]-SumLogit</td>
<td></td>
<td>89.16</td>
<td>86.97</td>
<td>84.03</td>
</tr>
<tr>
<td>LUCIR[6]-SumLogit</td>
<td></td>
<td>90.48</td>
<td>88.23</td>
<td>85.75</td>
</tr>
<tr>
<td>LRCIL[10]-Max</td>
<td></td>
<td>91.75</td>
<td>89.03</td>
<td>86.11</td>
</tr>
<tr>
<td>iCaRL[11]-Max</td>
<td></td>
<td>91.28</td>
<td>89.23</td>
<td>86.48</td>
</tr>
</tbody>
</table>

Table 1. Empirical study on different multi-task learning trade-off parameter λ values for the suggested CDBD EASY1500 evaluation. The reported results are Average Accuracies (AA) for continual deepfake detection. Bold: best green/blue/red results. Underline: second/third best green/blue/red results. The best-second-best/third-best LRCIL, iCaRL, LUCIR results are in green, blue, and red respectively.

Table 2. Benchmarking results on essentials of CIS methods on CDDB’s EASY1500 evaluation. The reported results are Average Forgetting degree, AA-M: Average Accuracy for deepfake recognition. The AA results of LRCIL, iCaRL, LUCIR’s performances to see which triplets consistently perform well. Each triplet should be of the same MT variant and the same λ value.

Table 3. Continual GANfake detection accuracies of the main evaluated methods on the GANfake dataset [8] after the training on the last GAN using different memory budgets (1024 and 512).

Since the GANfake [8] dataset does not release the detailed train/val/test splits, we can merely evaluate the methods on our own splits following the description in the original paper. Accordingly, the reported results are not really comparable with those reported in [8].
CDDB-Hard data stream). Due to the CDD setup, we cannot compare CNNDet trained directly on the whole data stream. Table 4 reports the results, showing the clearly better generalization of our adapted ones. On the other hand, we also test the generalization ability of these methods to corrupted test images using the same strategy, i.e., Blur+JPEG (0.5), from [14]. Note that [14] suggests Blur+JPEG (0.5) for data augmentation during training, and testing on clean images. Despite no Blur+JPEG augmentation for training, our adapted ones (especially iCaRL) generally work better than the competitors (Table 5).

6. More Overviews of Five CDDB Evaluations

Fig.2 presents radar plots on the four evaluation metrics, i.e., AAs, AFs, mAPs and AA-Ms, of the evaluated methods for EASY1500, LONG1500, HARD1500, HARD1000, and HARD500, where 1500, 1000, 500 are three studied memory budgets, AA, AF, AA-M, mAP are Average Accuracy for deepfake detection, Average Forgetting degree, Average Accuracy for deepfake recognition, and mean Average Precision respectively. The mAP is calculated by the mean of areas under the precision-recall (PR) curves. For those multi-class and multi-task learning methods, we feed the PR calculation function with the normalized value of the maximum activation output on fakes. Formally, the normalized value is 

\[ p_F = \frac{M_F}{M_F + M_R} \]

where \( M_R \) and \( M_F \) are the maximum activation values for reals and fakes respectively.

From the overall results in Fig.2, we can see that LUCIR-MT (multi-task learning) generally works the best than the other evaluated methods in terms of AA, AF, AA-M and mAP for the five evaluations. By comparing the five different evaluations, HARD500’s AA, AF, AA-M, mAP scores are clearly worse than the other four evaluations. This implies that the detection over HARD500 is the most challenging. In particular, it suffers from much more serious catastrophic forgetting due to the smaller memory budget. Fig.3 and Fig.4 present the PR curves of the LUCIR method and the DyTox method for the five evaluations EASY1500, LONG1500, HARD1500, HARD1000, and HARD500. The PR curves demonstrate that the most difficult tasks are generally the ones on SAN [2], WildDeepfake [15], and WhichFaceReal [1] for the different evaluations. This is because SAN is a small data set, and WildDeepfake/WhichFaceReal consists of unknown deepfakes in the wild scenes.

References


