Generalization-Preserved Learning: Closing the Backdoor to Catastrophic Forgetting in Continual Deepfake Detection

Supplementary Material

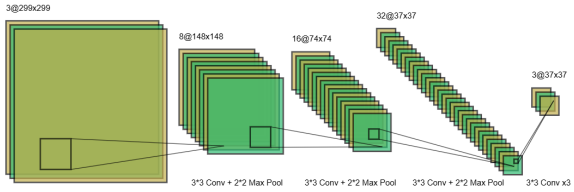

Figure 1. We propose to simulate the physics-based exposure process using a shared neural network F_Θ in a progressive manner.

A. Architecture of the Watermark Generator

The architecture of the watermark generator is designed as a lightweight convolutional network to efficiently adapt incremental data without modifying the backbone network. As shown in Fig. 1, the generator consists of multiple convolutional layers with progressively reduced spatial resolution. It begins with an input resolution of 299×299 and undergoes sequential downsampling through three 3×3 convolutional layers, each followed by a 2×2 max pooling operation. The feature maps increase in channel depth from 3 to 8, 16, and finally 32, allowing the generator to extract hierarchical representations of the input image. The final output is a three-channel residual mask of size 37×37, which is applied to the input image to generate the watermarked version. This lightweight design ensures minimal computational overhead while effectively aligning incremental data distributions.

B. Algorithm for Generalization-Preserved Learning

As shown in Algorithm 1, we provide a concisely summarized algorithm for better comprehension in the detailed implementation of the proposed Generalization-Preserved Learning.

Algorithm 1 Generalization-Preserved Learning

Require: Feature extractor $\mathcal{F}(x, \theta_f)$, classifier $\mathcal{C}(x, \theta_c)$, base watermark generator $\mathcal{G}(x, \theta_g^{1,learn})$, dataset sequence $S = \{D_1, ..., D_K\}$, clusters $C_{\text{real}}, C_{\text{fake}}$
Ensure: Optimized watermark generator $\mathcal{G}(x, \theta_g^{K,learn})$

for $t = 2, ..., K$ **do**
  Initialize $\theta_g^{t,fix}, \theta_g^{t,learn} \leftarrow \theta_g^{t-1,learn}$
  for mini-batch $(x, y) \in D_t$ **do**
    Generate feature:
    $x^{w,fix} = x + \lambda_g \mathcal{G}(x, \theta_g^{t,fix})$
    $x^{w,learn} = x + \lambda_g \mathcal{G}(x, \theta_g^{t,learn})$
    $z^{learn} = \mathcal{F}(x^{w,learn}, \theta_f)$
    Compute total loss:
    $\pi^{fix} = \text{Softmax}(\mathcal{C}(\mathcal{F}(x^{w,fix}, \theta_f), \theta_c))$
    $\pi^{learn} = \text{Softmax}(\mathcal{C}(\mathcal{F}(x^{w,learn}, \theta_f), \theta_c))$
    $\mathcal{L}_{\text{cls}} = -\sum_{i=1}^{N} \left[y_i \log \pi_i^{learn} + (1 - y_i) \log(1 - \pi_i^{learn}) \right]$
    $\mathcal{L}_{\text{nce}} = -\log \frac{\exp(-d_{\mathbb{H}}(z^{learn}, C_y)/\tau)}{\sum_{C_j \in \{C_y, C_{\neg y}\}} \exp(-d_{\mathbb{H}}(z^{learn}, C_j)/\tau)}$
    $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{cls}} + \lambda_{\text{nce}} \mathcal{L}_{\text{nce}}$
    Adjust gradient
    $\mathcal{L}_{\text{KL}} = \sum_j \pi^{fix}(y_j|x) \log \frac{\pi^{learn}(y_j|x)}{\pi^{fix}(y_j|x)}$
    $G_{task} = \nabla_{\theta_g^{t,learn}} \mathcal{L}_{\text{total}}$
    $G_{base} = \nabla_{\theta_g^{t,learn}} \mathcal{L}_{\text{KL}}$
    $G_{\text{align}} = G_{task} - \lambda_{\text{kl}} \frac{G_{task} \cdot G_{base}}{\|G_{base}\|^2} G_{base}$
    Update watermark generator:
    $\theta_g^{t,learn} = \theta_g^{t,learn} - \eta G_{\text{align}}$
  end for
end for
return trained watermark generator $\mathcal{G}(x, \theta_g^{K,learn})$

C. Calculation of the Loss Landscape

To analyze the loss landscape, a reference parameter point θ^* is first determined, typically corresponding to the optimal parameters obtained after training a neural network to convergence. Next, two independent direction vectors, δ and η, are selected, usually sampled randomly from a Gaussian distribution to ensure randomness. However, due to the differences in parameter scales may lead to visualizations of the loss landscape that are not directly comparable. To address this issue and enhance comparability across different network architectures or optimization methods, a "filter normalization" strategy can be employed. This technique normalizes each direction vector at the filter level so that its scale matches that of the original weights, thereby eliminating the impact of parameter scale variations. Specifically, for a neural network with parameters θ, a random direction vector d is first generated with dimensions identical to θ. Then, each filter in d is normalized to have the same norm as the corresponding filter in θ, as given by:

$$d_{i,j} \leftarrow d_{i,j} \frac{\|\theta_{i,j}\|}{\|d_{i,j}\|},$$

where $d_{i,j}$ represents the jth filter of the ith layer in d, and $\|\cdot\|$ denotes the Frobenius norm. The normalized direc-

tion vectors are subsequently used for parameter transformations to construct a two-dimensional parameter space for loss landscape evaluation. In the computation of the loss landscape, let α and β be two control variables. The transformation in the parameter space is defined as:

$$\theta(\alpha, \beta) = \theta^* + \alpha\delta + \beta\eta,$$

where α and β vary within a certain range, forming a set of sampled points in a two-dimensional plane. For each transformed parameter $\theta(\alpha, \beta)$, the corresponding value of the loss function is calculated as:

$$L(\alpha, \beta) = L(\theta(\alpha, \beta)).$$

By computing the loss matrix with different values of α and β, the loss landscape can ultimately be visualized using three-dimensional surface plots. In our case, α and β represent the x and y axes, while $L(\alpha, \beta)$ represents the z axis. This method provides an intuitive representation of the geometric properties of different network architectures or optimization methods in the parameter space, which is frequently used to explore the generalizability of models. Experiments have shown that flat surface generally corresponds to better generalization performance, while sharp surface may imply that the model overfits the training data.

D. Learnable Watermark Embedding for Incremental Alignment

To effectively address catastrophic forgetting in continual deepfake detection, we introduce a *learnable watermark generator* $\mathcal{G}$ that embeds structured perturbations into incremental data, aligning them with the base set's distribution in hyperbolic space. The watermarking process consists of two key components:

1. **Global Watermark for Cross-Task Alignment**: A shared watermark is optimized across all incremental tasks to ensure that new samples maintain distributional consistency with base set forgery patterns. This global watermark enforces feature alignment across different incremental datasets, reducing task recency bias and stabilizing long-term feature representations.
2. **Special Watermark for Sample-Specific Adaptation**: In addition to the global watermark, a task-specific watermark is dynamically generated per sample, introducing fine-grained perturbations that adapt to individual forgery instances while preserving the underlying decision boundary.

Process Overview:

- The backbone network $\mathcal{F}$ first extracts feature embeddings from raw images.
- The attention-guided watermark mask selectively enhances forgery-relevant regions by computing a task-aware attention map.
- The watermark generator $\mathcal{G}$ applies a mixture of global and special perturbations to incrementally align new forgery distributions with the base set's hyperbolic space.
- The refined images are reprocessed through the network to ensure feature consistency across tasks, minimizing distributional drift.

By integrating both **global and special watermarking strategies**, our method effectively projects new forgery samples onto the learned hyperbolic manifold of past tasks. This prevents overfitting to recent forgeries, while ensuring the model retains generalizable knowledge of prior deepfake patterns.

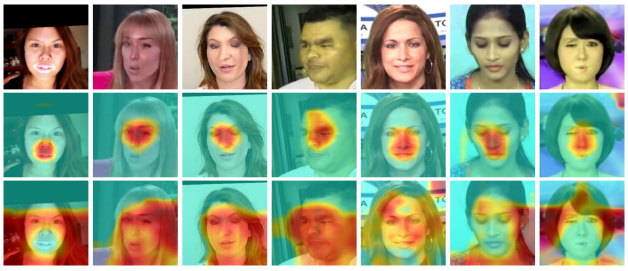

Figure 2. Grad-CAM Visualization on FF++ Dataset

Figure 3. Grad-CAM Visualization on DFDCP Dataset

Figure 4. Grad-CAM Visualization on DFD Dataset

E. Visualization of Model Attention via Grad-CAM.

We utilize Grad-CAM to visualize the attention regions of the backbone network, comparing the feature focus of DFIL and our method. The results are shown in Fig. 2, 3, 4,

Figure 5. Grad-CAM Visualization on CDF Dataset

and 5 indicate that DFIL exhibits a more dispersed attention distribution, easily prioritizing background regions. In contrast, our method focuses more on key forgery-related details in facial areas, such as the eyes, nose, and mouth, demonstrating a more stable and localized attention distribution for forgery detection.

F. Convergence Proof

In this section, we provide a convergence analysis for our proposed training scheme under standard assumptions. We specifically analyze the convergence behavior of the watermark generator parameter updates using the Generalized Gradient Projection strategy, combined with the alternating training strategy.

F.1. Assumptions

We make the following standard assumptions for the convergence analysis:

1. **Smoothness:** The total loss function

$$\mathcal{L}_{\text{total}}(\theta_g) = \mathcal{L}_{\text{cls}}(\theta_g) + \lambda_{\text{nce}}\mathcal{L}_{\text{nce}}(\theta_g),$$

   is continuously differentiable with an L-Lipschitz continuous gradient:

$$\|\nabla\mathcal{L}_{\text{total}}(\theta_1) - \nabla\mathcal{L}_{\text{total}}(\theta_2)\| \leq L\|\theta_1 - \theta_2\|, \quad \forall\, \theta_1, \theta_2.$$

2. **Boundedness:** The loss function $\mathcal{L}_{\text{total}}$ is bounded from below by a finite value.
3. **Learning Rate:** The learning rate η satisfies $0 < \eta < \frac{1}{L}$, ensuring stable gradient descent steps.
4. **Non-expansiveness of Projection:** The projection operation defined in the Generalized Gradient Projection strategy does not increase the norm of the gradient:

$$\|G_{\text{align}}\| \leq \|G_{\text{task}}\|.$$

F.2. Update Rule and Gradient Projection

At incremental task t, the watermark generator parameters $\theta_g^{t,learn}$ are updated via:

$$G_{\text{task}} = \nabla_{\theta_g}\mathcal{L}_{\text{total}}(\theta_g), \tag{1}$$

$$G_{\text{base}} = \nabla_{\theta_g}\mathcal{L}_{\text{KL}}(\theta_g), \tag{2}$$

$$G_{\text{align}} = G_{\text{task}} - \lambda_g \frac{\langle G_{\text{task}}, G_{\text{base}}\rangle}{\|G_{\text{base}}\|^2} G_{\text{base}} \tag{3}$$

leading to the parameter update:

$$\theta_g^{(t+1)} = \theta_g^{(t)} - \eta G_{\text{align}},$$

where η is the learning rate and λ_g controls the projection strength.

F.3. Convergence Analysis

Under the smoothness assumption, using the Descent Lemma, we have:

$$\mathcal{L}_{\text{total}}(\theta_g^{(t+1)}) \leq \mathcal{L}_{\text{total}}(\theta_g^{(t)}) + \langle\nabla\mathcal{L}_{\text{total}}(\theta_g^{(t)}), -\eta G_{\text{align}}\rangle + \frac{L\eta^2}{2}\|G_{\text{align}}\|^2. \tag{4}$$

Considering the non-expansiveness assumption, we have:

$$\|G_{\text{align}}\| \leq \|G_{\text{task}}\| = \|\nabla\mathcal{L}_{\text{total}}(\theta_g^{(t)})\|,$$

Choosing the learning rate sufficiently small, e.g., $\eta \leq \frac{1}{L}$, ensures a monotonically decreasing loss sequence:

$$\mathcal{L}_{\text{total}}(\theta_g^{(t+1)}) \leq \mathcal{L}_{\text{total}}(\theta_g^{(t)}) - \frac{\eta}{2}\|\nabla\mathcal{L}_{\text{total}}(\theta_g^{(t)})\|^2,$$

indicating that each update step guarantees descent in the loss.

F.4. Block Coordinate Descent and Convergence

Our training scheme employs an alternating (block coordinate descent) optimization strategy: the watermark generator and backbone network parameters are updated in an alternating fashion. Given the non-convexity of the neural network objective, convergence is understood in the sense of convergence to a stationary point (local minimum or saddle point).

Block coordinate descent (BCD) algorithms with Lipschitz continuous gradients ensure convergence to stationary points under standard non-convex optimization conditions Specifically, by fixing the backbone and classifier parameters, the incremental learning phase reduces to optimization with respect to a single parameter block (the watermark generator). Thus, standard convergence results for single-block optimization apply directly.

As a result, our alternating training strategy (BCD between backbone/classifier and watermark generator) combined with the gradient projection mechanism ensures that

each incremental update provides a monotonically decreasing loss sequence, ultimately converging to a stationary point. Formally, as $t \to \infty$, we obtain:

$$\lim_{t\to\infty} \|\nabla \mathcal{L}_{\text{total}}(\theta_g^{(t)})\| = 0,$$

which guarantees convergence to a stationary point.

F.5. Conclusion

Given the standard non-convex nature of deep neural network optimization, the proposed Generalized Gradient Projection combined with alternating block coordinate training guarantees convergence to a stationary point (local minimum or saddle point). This theoretical analysis supports the stability and effectiveness of our proposed GPL framework for incremental deepfake detection tasks.

G. Comparison of Training a New Model with Few Data

G.1. Experimental Setup

To further evaluate the effectiveness of our proposed method, we conduct a comparative experiment involving the following two approaches:

1. Training a new model from scratch, using only a limited number of new forgery samples.

2. Using our incremental learning method, where the model learns new forgery techniques while retaining knowledge from previous tasks.

We follow the experimental setup of DFIL, selecting a small number of samples from FF++, DFDCP, DFD, and CDF datasets. The newly trained model is trained only on these limited samples, whereas our incremental learning method continuously learns new data while preserving prior knowledge(FF++).

G.2. Results and Analysis

Table 1. Comparison of training a new model vs. using our incremental learning method.

Method	FF++ (%)	DFDCP (%)	DFD (%)	CDF (%)	Avg (%)
Train from scratch	-	82.44	95.05	74.63	84.04
DFIL	-	88.87	**96.92**	84.68	90.16
Ours	-	**89.85**	94.32	**93.29**	**92.49**

Tab. 1 compares the performance of continual deepfake detection under limited data conditions, focusing on the traditional Train from Scratch approach, DFIL, and our proposed Generalization-Preserved Learning (GPL) method.

For the first row (Train from Scratch), the model is trained and tested using the full training set of three datasets without applying incremental learning. For the second and third rows (incremental learning methods), the model is first pre-trained on the FF++ dataset. Then, DFDCP, DFD, and CDF are sequentially introduced as incremental training datasets, where only 25 fake and 25 real videos are selected from each dataset to evaluate the generalization ability of different methods in a low-sample incremental learning scenario. The results demonstrate that our method achieves the best performance across all datasets, with an average classification accuracy (Avg.) of 92.49%, outperforming DFIL by 2.33% and Train from Scratch by 8.45%, validating GPL's effectiveness in incremental learning under limited data conditions.

On the DFDCP dataset, our method achieves 89.85% detection accuracy, surpassing DFIL by 1.98% and Train from Scratch by 7.41%, demonstrating that GPL maintains strong forgery detection capability even in a low-sample setting. On the CDF dataset, our method achieves 93.29% detection accuracy, significantly outperforming DFIL (+8.61%) and Train from Scratch (+18.66%). This result suggests that GPL can better adapt to distribution shifts introduced by different forgery techniques, enhancing feature stability during incremental learning. On the DFD dataset, although our method is slightly lower than DFIL (-2.6%), it still maintains high detection performance and achieves the best overall average accuracy.

These experimental results not only demonstrate the effectiveness of continual deepfake learning, where incremental learning can yield better results than training on full data from the current task alone, but also further confirm that GPL effectively regulates the incremental data feature space through watermark perturbations and hyperbolic visual alignment. This enhances the model's generalization ability while better retaining detection capability for historical tasks, ensuring stability and robustness throughout the incremental learning process.

H. Ablation Study on λ_{nce}

To analyze the effect of the hyperparameter λ_{nce}, which balances the classification loss and the contrastive loss, we conduct an ablation study by varying λ_{nce} while keeping other settings fixed. The total loss function is:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{cls}} + \lambda_{\text{nce}} \mathcal{L}_{\text{nce}}, \tag{5}$$

where $\mathcal{L}_{\text{cls}}$ is the classification loss and $\mathcal{L}_{\text{nce}}$ is the contrastive learning loss in the hyperbolic space. We evaluate the performance using different values of λ_{nce} and report the classification accuracy in Tab. 2 presents the accuracy across different values of λ_{nce}. The results indicate that the performance is optimized when λ_{nce} is set to an appropriate value, balancing the influence of contrastive learning and classification loss.

The results show that overly small or large values of λ_{nce} lead to suboptimal performance. A moderate value, such

Table 2. Ablation study on the effect of λ_{nce} in the loss function.

λ_{nce}	AA (%) ↑	AF (%) ↓
0.01	79.92	3.48
0.05	88.11	5.29
0.1	92.14	1.42
0.2	90.72	3.68
0.3	85.49	2.21
0.5	88.01	3.43
0.8	67.83	10.23
1.0	66.48	11.52

as $\lambda_{\text{nce}} = 0.1$, achieves the best trade-off between classification accuracy and contrastive learning, improving the model's ability to distinguish real and fake samples effectively.

H.1. Training Overhead

During incremental training, only the watermark generator ($\mathcal{G}$) is optimized, while the backbone ($\mathcal{F} + \mathcal{C}$) remains frozen, significantly reducing training costs. To quantify this efficiency, we analyze the parameter count of both the watermark generator and the backbone, as shown in Tab. 3. The watermark generator accounts for only 1.06% of the total parameters, whereas the backbone comprises 98.94%.

Table 3. Comparison of parameter counts between the watermark generator and backbone

Module	Parameter Count	Proportion (%)
Watermark Generator ($\mathcal{G}$)	0.22M	1.06 %
Backbone ($\mathcal{F} + \mathcal{C}$)	20.8M	98.94 %

This indicates that our method requires updating only a minimal fraction of the parameters during incremental training, enabling efficient adaptation to new forgery distributions while maintaining computational efficiency. By minimizing the number of trainable parameters, our approach reduces training overhead, making continual learning more efficient and resource-friendly.