

Beyond the Benchmark: Generalization Limits of Deepfake Detectors in the Wild

Ankur Garg
ankur_garg@berkeley.edu

Gaurav Sharan Srivastava
gaurav_sharan@berkeley.edu

Katsuya Masaki
k.masaki@berkeley.edu

Abstract

In light of the rapid evolution of image generation, differentiating between what is real and fake has become increasingly challenging. This paper presents an evaluation on the potential of fine-tuning to generalize deepfake detection models across various generative models. We investigated two types of generalization; adaptive in-domain generalization, which refers to a model’s ability to learn a task on which it is trained directly, and zero-shot generalization, which is the ability to perform on tasks outside of the data it was trained on. After running various experiments with a CLIP-based deepfake detector, we found that no fine-tuning method could achieve meaningful zero-shot generalization, although some fine-tuning methods showed strong adaptive in-domain generalization without forgetting. During the rehearsal learning, However, we saw consistent pattern of how the training on a specific generative model impacted the performance on the other generative models. These data points are inspiring our future work on parameter-efficient fine-tuning and model architecture.

1. Introduction

The European Union Agency for Cybersecurity (ENISA) ranks “advanced disinformation/influence operations campaigns”, which include the use of deepfakes, as the 7th highest priority in its “ENISA Foresight Cybersecurity Threats for 2030” report [20]. Deepfakes blur the line between real and fake, and society continues to lose trust in media as fake images become a regularity. From GANs (Generative Adversarial Networks) to the most recent developments in Diffusion, models have improved at astronomical rates. As generative techniques advance, so do the models we use to detect them. Models such as CLIP, continue to push the limits of what deepfake detection can look like.

The biggest challenge in deepfake detection, however, is generalization. Even the best of them struggle to keep up with the latest generative techniques. In fact, deepfakes possess different characteristics based on their generation process, and training a deepfake detection model with sufficient training data is not impossible. But the question still remains: can the model apply fundamental image recognition skills to learn these features even if only a few hundred images are available? Additionally, to address continuously emerging new deepfake generation methods, it would be ideal to maintain a certain level of detection capability against unseen deepfakes. Do such training methods exist?

To address this question, we prepare five different deepfake datasets and verify by combining foundational vision models with fine-tuning. First, we experiment with multiple fine-tuning methods to determine whether models can leverage learning from the previous tasks to correctly detect deepfakes even with limited datasets, what we call “adaptive in-domain generalization.” Next, we test various fine-tuning methods to measure performance against unseen deepfakes using training from other deepfakes, which we refer to as “zero-shot generalization.” Through these results, we clarify the extent to which generalizability can be expected from current state-of-the-art models.

2. Related Work

2.1. Deepfake Detection

Early research in deepfake detection has primarily focused on understanding the origins and inherent challenges of synthetic media. Farid [11] offers a comprehensive exploration of the emergence of deepfakes, detailing how sophisticated generative adversarial networks (GANs) have enabled the creation of highly realistic manipulated images. Farid’s

work emphasizes the limitations of human perception in distinguishing real from fake, reporting that even with iterative feedback, human participants achieved an accuracy barely above chance level accuracy. This foundational study underscores the complexity of the deepfake phenomenon and sets the stage for subsequent computational approaches.

Building on these initial insights, subsequent research has explored learning-based methodologies for deepfake detection. Although training classifiers on large datasets of real and synthetic images has proven effective, these models are often prone to overfitting on dataset-specific artifacts. Such vulnerabilities result in models that fail to generalize when confronted with new or subtly different generative techniques. Thus, the early body of work highlights the critical need for detection systems that are not only accurate but also robust to the evolving landscape of generative methods.

2.2. Classical Approaches

In a pre-deep-learning era, researchers have been tackling this problem by various machine learning algorithms, such as K-nearest neighbor [3], Support Vector Machine[22], and Decision Tree [2]. These classic models are often combined with feature extraction and data processing techniques such as PCA and Fourier transformation, to obtain useful information on the texture, frequency, and compression of the images [3]. This requires researchers to have a great deal of expertise in forensics, signal processing and physics. Thus, these methods always had the risk of missing subtle or novel forgery patterns due to the limits of human recognition, and also had difficulty in generalizing across different types of forgeries or datasets.

2.3. Deep Learning and Generalizability

Once deep learning became a feasible and popular choice of learning technique, deepfake researchers quickly adopted it, especially because of the excellent fit of the convolutional layers to the visual task. Unlike traditional machine learning algorithms, Convolutional Neural Nets (CNN) capture translational invariant features of the original image data without explicit instructions on which signals or correlations matter. CNNs work well as flexible feature extractors either for multi-layer perceptron, or for a classical machine learning algorithm, such as SVM [22]. However, even with the CNN’s powerful feature extraction, the opinion on the generalizability of the simple classifier is divided. Zhang [36] reported that conventional supervised machine learning was unable to generalize across different GAN-based deepfakes effectively. On the contrary, Wang [37] claimed that a simple baseline model can generalize in distinguish various GAN-based generative models to be trained on a specific variant of GAN.

These investigations hinted that more advanced models could have better generalizability across multiple deepfakes. Guo et al. [38] presented HiFi-Net, a hierarchical fine-grained approach to image forgery detection and localization. Their model not only classifies whether an image is manipulated but also pinpoints specific regions of tampering. The methodology involved a multi-branched feature extraction process that captured both global and local inconsistencies, utilizing self-attention mechanism. The model was tested on multiple datasets, achieving 91.2% accuracy in detecting manipulated images and 85.7% accuracy in localization tasks. They also evaluated the model on nine unseen datasets to gauge its generalizability. The result showed that HiFi-Net presented generalizability over four unseen specific GAN-based models such as ProGAN and StyleGAN, and Diffusion-based model such as ADM, achieving 96% to 99% accuracy. Although the generalization did not happen for the remaining six models, this paper brought a ray of hope of the generalization over Out-of-Domain data. The hierarchical approach showed superior performance in detecting subtle forgeries such as face-swapping and splicing, outperforming traditional CNN-based and transformer-based detectors. However, the increased model complexity results in higher computational costs, making real-time deployment challenging.

2.4. Vision-Language Models and Fine-Tuning

An emerging trend in the literature is the integration of vision-language models and fine-tuning of the pre-trained models for deepfake detection. With models like OpenAI’s CLIP that jointly process visual and textual information, researchers have explored novel ways to leverage multi-modal features. Khan and Dang-Nguyen [21] adapt CLIP to the deepfake detection task by employing several strategies: linear probing, full fine-tuning, prompt tuning, and the incorporation of adapter networks. Their approach is predicated on the idea that the inherent semantic understanding embedded in vision-language models can effectively distinguish real images from synthetic ones.

For facial forgery images, Yermakov et al. [6] addressed the challenge of detecting partially manipulated facial deepfakes with a focus on generalizability across diverse datasets and unknown forgery techniques. Their work leverages the Contrastive Language-Image Pre-training (CLIP) model, specifically its ViT-L/14 visual encoder, to develop a detection method that requires minimal modifications to the original model architecture. The authors employ parameter-efficient fine-tuning (PEFT) techniques, particularly LN-tuning, which adjusts only 0.03% of the total model parameters. They

enhance the approach through several regularization strategies: L2 normalization to project features onto a hypersphere, metric learning using uniformity and alignment losses, and spherical linear interpolation (slerp) augmentations in the feature space. The model was trained on the FaceForensics++ dataset containing videos from four different deepfake forgery methods. Evaluation was conducted across multiple benchmarks including Celeb-DF-v2, Google’s DFD, DFDC, FFIW, and DeepSpeak v1.0 datasets. Results demonstrate that their approach achieves competitive performance comparable to or surpassing more complex state-of-the-art methods, with video-level AUROC scores of 96.62 on CDFv2, 98.0 on DFD, 87.15 on DFDC, 91.52 on FFIW, and 92.01 on DSv1. The work establishes a strong, easy-to-implement baseline for identifying subtle facial manipulations and highlights the effectiveness of CLIP’s visual encoder with proper fine-tuning for the deepfake detection task, providing a foundation for our research in the generalizability of deepfake detection models.

2.5. Power of Parameter Efficient Fine-Tuning

The strength of pretrained models with fine-tuning approaches lies in their versatility and effectiveness across different computational constraints and detection scenarios. Full fine-tuning leverages the complete capacity of pretrained architectures, allowing comprehensive adaptation of all parameters to the nuances of deepfake artifacts, which often results in superior performance when sufficient computational resources and training data are available. Meanwhile, Parameter-Efficient Fine-Tuning (PEFT) methods such as adapter, and layer normalization tuning offer complementary advantages by requiring updates to only a fraction of parameters, thereby reducing computational costs while still maintaining much of the detection capability.

More importantly, as Han [26] stated in their thorough review, PEFT methods are more likely to preserve the underlying generalizability of the base model because full fine-tuning sometimes cause models to forget their original training, damaging generalizability. This PEFT approach to fine-tuning pretrained models has proven particularly valuable for deepfake detection, where models must recognize subtle manipulations across diverse generation techniques. The pretrained knowledge from large-scale datasets provides rich feature representations that can identify inconsistencies in facial expressions, lighting, and textures that often characterize synthetic media.

As this combination of pretrained models and various fine-tuning techniques becomes increasingly mainstream in deepfake detection, understanding the generalizability of these models, how they adapt to novel, unseen deepfakes through different fine-tuning methods, becomes critically important from both practical application and research perspectives.

3. Methodology

In order to best see the whole picture of generalizability, we assessed both in-domain and zero-shot generalization, evaluating their performance at every stage. For this purpose, we picked one foundational model, five generative deepfake, and seven fine-tuning methods.

3.1. Model Choice

In this research, we aimed to see how a variety of fine-tuning methods enhances the deepfake detection model’s ability to adapt to deepfakes generated by unseen types of generative models. For clarity, we discuss two different groups of models that we use:

3.1.1 Generative Models

Although recent deepfake detectors have achieved AUROC scores exceeding 99% on benchmark datasets such as FaceForensics++ [39, 40, 43], their robustness in open-world scenarios remains substantially lower. Prior evaluations reveal that detectors trained on a single synthesis pipeline often suffer accuracy drops of 20–60 percentage points when tested against unseen generative models or even newer versions of the same generator [41, 42, 44]. This fragility stems from detectors’ tendency to overfit to generator-specific artifacts- including checkerboard patterns in StyleGAN outputs, periodic noise residues from diffusion decoders, or color quantization artifacts from early autoencoders- rather than learning more general inconsistencies in semantic or physical plausibility [39, 42]. The challenge is further compounded by the rapid pace of generative model iteration: Midjourney has released six major versions within seventeen months, while Stable Diffusion introduced six new checkpoints in a single year, each subtly altering its latent-space encoding. Consequently, a detector deployed today must be capable of handling forgeries produced by tomorrow’s as-yet-unknown generative models, raising the critical problem of *cross-generator generalization*. Addressing this challenge requires a detection strategy that (i) maintains high accuracy on previously encountered generators, while

(ii) effectively transferring to unseen generators, including adversarially tuned variants [44]. This dual requirement forms the core motivation of our investigation.

We synthesized deepfake images using five major generative models: Adobe Firefly, NVIDIA’s StyleGAN, Midjourney 3.5, OpenAI’s DALL-E, and Stable Diffusion APIs. These can be divided into two primary architectures:

- **GAN-Based:** StyleGAN2: A Generative Adversarial Network architecture by NVIDIA pretrained on FFHQ. Each real image is first inverted to find the matching latent code in the FFHQ-trained model, ensuring the generated outputs remain visually close to the original face while still introducing modifications.
- **Diffusion-Based:** Firefly, Midjourney, DALL-E, Stable Diffusion: Each uses diffusion processes to iteratively refine generated images.

This diversity in generative sources (across GAN and diffusion-based systems) allows our dataset to capture the unique artifacts and signatures of each underlying architecture, improving the robustness of subsequent detection methods.

Model	Generation Type	Backbone ¹	Notes
Stable Diffusion 3 (3.5)	Latent diffusion (text-to-image)	MMDiT (Multi-modal Diffusion Transformer)	Replaces U-Net with a Transformer UNet; Mixture-of-Experts improves scale and prompt fidelity.
DALL-E 2	Two-stage diffusion (CLIP prior + latent diffusion decoder)	UNet (decoder) + Transformer prior	Stage 1: diffusion prior predicts CLIP-image embedding from CLIP-text embedding; Stage 2: diffusion decoder generates image.
Midjourney V6	Text-guided diffusion (proprietary)	Likely Transformer-UNet or DiT ² variant	Closed-source; literature treats as diffusion model with strong aesthetic post-training; grouped alongside SD and DALL-E in analyses.
Firefly 3	Text-guided diffusion (latent)	Transformer-based diffusion backbone	Described as “transformer-based” in Adobe/NVIDIA notes; optimized for photorealism and controllable style; promoted as “next-gen diffusion foundation model”.
StyleGAN2	Style-based GAN	Style-modulated progressive CNN generator	Convolutional generator with weight demodulation; no diffusion steps; included for contrast.

Table 1: Comparison of generative models by generation type, backbone architecture, and key notes.

Stable Diffusion 3 (3.5): SD-3 employs a latent-diffusion framework but replaces the classic UNet with **MMDiT**—a *Multi-Modal Diffusion Transformer* that processes latent patches via self-attention. The model uses a Mixture-of-Experts (MoE) gating layer to allocate compute across $N=64$ experts (total ≈ 2.7 B parameters) and is pre-trained on a 6.1B image-text corpus derived from LAION-5B plus filtered Shutterstock licensing. Patchified latent conditioning allows 1024×1024 synthesis at a lower memory footprint than pixel-space diffusion. Prompt fidelity is further improved by cross-attention “prompt adapters” that inject the CLIP-T/14 embedding at every diffusion block.

DALL-E 2: OpenAI’s two-stage pipeline first uses a **prior diffusion** that maps a CLIP text embedding to an *image* embedding in the CLIP representation space. A 3.5B parameter DiT-XL/2 transformer serves as the prior backbone, trained on 400M CLIP-filtered (image,text) pairs. Stage 2 employs a 1.5B parameter UNet decoder conditioned on the image embedding to synthesis a 256×256 raster. Importantly, the CLIP prior regularizes visual features toward a language-aligned manifold.

Midjourney v6: Although closed-source, we treat v6 as a **text-guided diffusion** model whose noise scheduler and classifier-free guidance parameters have been tuned for stylized, high-contrast compositions. The training set is rumored to exceed 1B high-resolution images from LAION, ArtStation, Flickr-HQ and Pinterest, with post-training aesthetic re-ranking.

Firefly 3: Adobe’s third-generation engine is likewise a **text-guided latent diffusion** model but is trained exclusively on *licensed* and *CC-BY* images (estimated 200M pairs). The backbone swaps UNet for a Transformer-based diffusion module with split self/cross-attention. As a by-product of its licensing filter, Firefly lacks many web artifacts, which leads to fewer high-frequency cues; detectors must instead rely on more subtle generative fingerprints.

¹“Backbone” here refers to the core network inside the diffusion (or GAN) generator (e.g. UNet, Diffusion Transformer (DiT), or style-modulated CNN.)

²DiT and MMDiT denote Transformer-based UNet replacements.

StyleGAN2: For contrast with diffusion, we include the canonical **style-based GAN**. StyleGAN2 modulates channel-wise convolution weights via an 8-layer MLP style mapping network and employs weight demodulation to suppress signal bias. The official 1024×1024 model is trained on the FFHQ dataset (70k images) for 25M generator parameters. Unlike diffusion, no iterative denoising occurs; outputs manifest GAN-typical phase noise and checkerboard artifacts that are easily recognized by vanilla detectors yet can mis-lead models fine-tuned on purely diffusion data.

The heterogeneity in generation mechanism (latent vs. pixel diffusion vs. style-GAN), backbone (UNet, DiT, MoE-Transformer, CNN) and data scale (70k–6B images) forces the detection model to learn *domain-invariant* artifacts rather than memorizing a single generator’s quirks. Section 5.3 demonstrates that LoRA rehearsal is uniquely capable of retaining such invariants across the above spectrum, whereas full fine-tuning catastrophically forgets them as new domains appear.

3.1.2 Detection Models

Detection models are pre-trained models that classify facial images into real images or forged images. Here we can define two types of pre-trained models: general image classification models, and deepfake detection models. The former is pre-trained on ImageNet for multi-class image classification tasks, and not specialized in facial deepfake detection. The latter is pre-trained on facial or non-facial deepfake dataset to conduct binary-class image classification. In this project, our interest lies at how pre-trained facial forgery detection models can detect images from new types of generative models with fine-tuning.

Deepfake Detection [6] is trained on the FaceForensic++ dataset, which is a collection of by a classical face-swapping forgery techniques. The dataset does expose the model to GAN-based deepfakes but only in the context of the face-swapping task. It is important to remember the previous exposure to GAN when we discuss the results later in this paper. However, given the shift toward diffusion-based models and away from GAN, this model was still the ideal option, as it has not been previously exposed to any diffusion-based methods which are the majority of our generative models being tested against.

Model Type	Model Name / Backbone	Provider
Facial Deepfake	Deepfake Detection / CLIP	Yermakov et al.[6]
General Deepfake	Universal Deepfake Detector / CLIP	Khan et al. [21]
General Image	CLIP	Open AI [4]
General Image	SWIN V2	Microsoft [27]
General Image	EfficientNet	Google [16]

Table 2: Comparison of Detection Models

3.2. Choice of Fine-tuning Methods

To evaluate the model’s adaptive and zero-shot generalizability, we applied various popular fine-tuning methods which are actively adopted in the industry and academia. Other than the most basic full fine-tuning, we explored six Parameter-Efficient Fine-Tuning (PEFT) methods which update a subset of the total parameters. PEFT methods are useful because as model sizes continue to grow, full fine-tuning becomes computationally expensive and sometimes cause models to forget their original training, damaging generalizability. PEFT methods aim to adapt large pre-trained models with minimal parameter updates, thereby preserving the underlying generalizability of the base model [26]. As Han stated, PEFT methods can be categorized into three families; Adaptive, Reparameterized, and Selective methods. To cover most of the popular fine-tuning methods, we chose representative approaches from each category, as highlighted in Table 3.

Adaptive PEFT

- **Adapter:** Adapter[29] is a lightweight neural module inserted into a pre-trained model that allows the model to learn new tasks with minimal additional parameters and without changing the full model weights. Instead of updating all the weights of the large pre-trained model which is computationally expensive and risks overwriting general knowledge, adapters freeze the pre-trained base model, insert small trainable adapter modules, and only update these adapter parameters, leaving the rest of the model untouched. This approach enables the model to

Methods	Category	Description
Full fine-tuning	N/A	Train all the parameters
Adapter [29]	Additive PEFT	Insert a few-layer trainable MLP before the classifier head
LoRA [7]	Reparam.PEFT	Add low-rank decomposition matrices in self-attention layers
VeRA [30]	Reparam.PEFT	Add low-rank randomized matrices in self-attention layers
LinearTail [31]	Selective PEFT	Train the classification head only
LN Tuning [13]	Selective PEFT	Train layer normalization modules and the classification head only
BayesTune [15]	Selective PEFT	Select trainable parameters by Bayesian hyperparameter estimation

Table 3: Fine-tuning Methods

preserve its general capabilities while learning task-specific knowledge using only a fraction of the parameters compared to full fine-tuning. Each adapter module typically consists of a down-projection and up-projection layer, a nonlinearity, and a residual connection that adds the adapter output back to the original input. Residual integration ensures minimal disruption, where the original knowledge from the pre-trained model flows through unchanged, while the adapter can redirect or amplify it when needed.

Reparameterized PEFT

- **LoRA:** LoRA [7] introduces a technique for fine-tuning large language models by injecting trainable low-rank matrices into the weights of the original model, while keeping all the pretrained weights frozen. Specifically, instead of directly updating weight matrices (e.g., $W \in \mathbb{R}^{d \times d}$), LoRA decomposes the update into two much smaller matrices $A \in \mathbb{R}^{d \times r}$ and $B \in \mathbb{R}^{r \times d}$, where $r \ll d$, and applies them as a low-rank perturbation:

$$W' = W + \alpha AB.$$

This approach significantly reduces the number of trainable parameters and memory usage (since gradients don't need to be stored for the pretrained parameters) during fine-tuning, while preserving the expressiveness of full-rank adaptation. Because the original weights remain untouched, LoRA is modular and enables multiple tasks to be supported easily by swapping in different low-rank adapters fine-tuned for different downstream tasks.

Conceptually, LoRA reframes fine-tuning as learning a minimal, low-rank task-specific shift in the model's parameter space, rather than relearning the full capacity of the base model. This makes LoRA appealing in scenarios where model weights must remain static, such as for multi-task reuse, but the biggest reason is for when storage and compute constraints prohibit full model updates.

- **VeRA:** VeRA [30] is a parameter-efficient fine-tuning technique that closely resembles Low-Rank Adaptation (LoRA), but achieves comparable or superior performance using even fewer additional parameters. It significantly reduces the number of trainable parameters by introducing just two additional trainable vectors per layer, dramatically cutting down on the total number of parameters compared to traditional fine-tuning methods. VeRA initializes two shared low-rank matrices based on the largest required dimensions among all adapted layers. During the forward pass, each layer dynamically extracts submatrices tailored to its dimensions from these shared matrices.

Selective PEFT

- **LN tuning:** Layer-Normalization Tuning (LN Tuning) [13] is a parameter-efficient fine-tuning strategy that updates *only* the two learnable vectors in each LayerNorm: the scale γ and the shift β . Training just these vectors reduces the number of tunable parameters by orders of magnitude compared with full fine-tuning. Yet, LN Tuning often matches, or even surpasses, the performance of full fine-tuning and LoRA on downstream tasks. Its strength lies in the fact that every token representation is first normalized and then rescaled and shifted by γ and β ; even small adjustments to these vectors therefore propagate through the entire network, enabling task-specific re-weighting and re-centering of internal feature dimensions.
- **Linear Tail:** The linear tail fine-tuning strategy[31] freezes the entire pretrained backbone and updates only the single fully-connected layer that maps the frozen hidden representation to C target classes. Containing few

learnable weights, this head slashes the number of trainable parameters and GPU memory, converges in a few epochs, and provides a quick sanity check of how linearly separable the task is in the pretrained feature space. Its speed and low resource footprint make it attractive for tight compute budgets or rapid prototyping, though performance can lag when domain shift or complex compositional patterns require deeper adaptation, situations where LayerNorm, LoRA, or full fine-tuning typically excel.

- **BayesTune:** BayesTune [15] adopts a Bayesian approach to automatically determine which parameters to fine-tune. This method applies a Laplace prior to the parameter set and effectively automates the selection process, thereby streamlining the fine-tuning workflow. As the first step, BayesTune applies a Laplace prior whose mean is the current parameter θ_i and some scale parameter λ_i . We sample this λ_i through Stochastic Gradient Langevin Dynamics (SGLD), by setting a Gamma hyperparameter with $\alpha = 0.01, \beta = 100$. Once we obtain λ_i 's posterior samples, its posterior mean plays a role as an individualized regularizer. If it is close to 0, then a Laplace prior imposes very small variance around the original value of the parameter which leads to no update, and when it is large, the parameter is allowed to move more around the current value. Therefore we can use this λ_i as an "importance indicator", and set threshold to filter top $k\%$ parameters to update. In our project, we applied a slightly modified version of BayesTune to work with a model with a huge parameter size like CLIP. We set λ by 'parameter groups' such as weights and biases of each layer, rather than individual parameters. For example, CLIP has 303,284,226 parameters but it only has 493 parameter groups. When the original parameter size is p and parameter group size is g , the original approach yields $2p$ parameters to compute, whereas our modified approach reduces the number of parameters to compute to $p + g$. Although this is an approximation of the original BayesTune, it still provides useful insights of which layers and modules should be updated, based on the new downstream task.

3.3. Dataset

For this project, we curated a balanced collection of 10,000 real and fake images across gender and ethnicity collected from various sources specifically for this task. Each image is stored in 1024×1024 PNG format with RGB color channels to maintain high visual quality for the subsequent deepfake generation. The full breakdown is highlighted in Table 4.

3.3.1 Dataset Distribution and Diversity

To ensure balanced representation, we divided the real images equally across male and female, with each subgroup further split into four ethnic classes: White, Black, South Asian, and East Asian. This stratified distribution, 625 images per combination of sex and ethnicity, improves the dataset's diversity and helps our deepfake detection models learn to generalize across a broad demographic range.

				Diffusion			GAN
	Midjourney	OpenAI Dall-E	Stability AI Stable Diff.	Adobe Firefly	Nvidia StyleGAN2		
Gender	Ethnicity	Real					
Male	White	625	125	125	125	125	125
Male	Black	625	125	125	125	125	125
Male	South Asian	625	125	125	125	125	125
Male	East Asian	625	125	125	125	125	125
Female	White	625	125	125	125	125	125
Female	Black	625	125	125	125	125	125
Female	South Asian	625	125	125	125	125	125
Female	East Asian	625	125	125	125	125	125
Total Images		5000	1000	1000	1000	1000	1000

Table 4: Dataset Distribution

Further, the images were collected and generated with balance in mind. We started by collecting just 1000 real faces; 400 from the DeepSpeak v1 Dataset[10][35] created by Prof. Hany Farid's lab, and 600 from the 70k-faces Kaggle dataset [24]. Using the methods shown in Table 5 and the section below, each of the 1000 images correlates to an image generated by each of our 5 generative models. We round out the rest of the dataset with an additional 4000 real images to get an even 50/50 split of real to fake.

In terms of train-validation-test partitioning, we apply an 70:15:15 ratio. Specifically, 70% of the images in each demographic stratum are used for training, 15% for validation, and the remaining 15% for testing. This stratified split ensures that each subset retains the same demographic proportions, thereby preserving diversity and minimizing potential biases. The corresponding deepfake images, generated by Firefly, DALL-E, Stable Diffusion, Midjourney, and StyleGAN, are also assigned to these splits in a matching proportion, maintaining the same 7:1.5:1.5 ratio to facilitate direct comparisons between real and synthetic data during model training and evaluation.

3.3.2 Dataset Generation Process

We leveraged APIs or subscriptions for each generation pipeline to maintain a uniform workflow and minimize unintended discrepancies in output quality. In each case, we opted for “highest-quality” API services to ensure we captured the most advanced, photorealistic output that state-of-the-art generators can produce.

Generative Source	Method / Details	Input	Output
DALL-E (DALL-E 2 API)[18]	Utilized <code>client.images.create_variation</code> to produce variations of a provided real image and leveraged highest-quality API tier to maximize photorealism.	Real Image	Synthetic (PNG) Image
Firefly (Subscription)[1]	Used text prompts plus the real image as a conditioning input, applying <i>style strength = 0.8</i> and <i>visual intensity = 0.7</i> to balance similarity and novelty.	Text Prompt + Real Image	Synthetic (PNG) Image
Stable Diffusion (SD Ultra API)[23]	Combined text prompt and real image with <i>variation intensity = 0.5</i> , relying on a high-fidelity diffusion pipeline.	Text Prompt + Real Image	Synthetic (PNG) Image
Midjourney (Subscription)[14]	Since no direct API is available, we built a Python-based automation bot that interacts with the Midjourney Discord bot, maintaining a standardized prompt.	Text Prompt + Real Image	Synthetic (PNG) Image
StyleGAN2 (Local Machine)[17]	We use NVIDIA’s StyleGAN2 pretrained on FFHQ. Each real image is first inverted to find the matching latent code in the FFHQ-trained model, ensuring the generated outputs remain visually close to the original face. This process keeps consistency with the other diffusion-based pipelines and yields high-quality deepfake faces.	Real Image (inverted to latent space)	Synthetic (PNG) Image

Table 5: Deepfake Generation Technique

3.3.3 Standardized Prompting

A critical step to maintain consistency and control unexpected variations across different generative models was the use of a standardized prompt. The prompt (adapted to each model’s syntax) follows the template:

“Generate a new person/entity with similar race, gender, age, hairstyle, hair color, eye color, expression, accessories, styling, with full head and shoulders in PNG. Keep the background similar. The generated person should be different from the real entity.”

By keeping this prompt identical for all diffusion-based and GAN-based systems (aside from minor syntax adjustments), we minimized unpredictable differences in the output. This approach allowed us to fairly compare images generated under the same criteria (e.g., same pose, similar background, same demographic attributes) while still ensuring each deepfake differed enough from the real image to qualify as a distinct “forged” example.

4. Experiment

4.1. Adaptive In-Domain Generalization

To measure adaptive in-domain generalization, we first evaluated the performance of the baseline model without any additional training. Then we fine-tuned the model on a training/validation set which includes all the five deepfakes namely Dall-E, Stable Diffusion, Firefly, Midjourney, and StyleGAN2. Next, we tested the performance on the test set which also includes all the five deepfakes. We repeated this process for the six fine-tuning methods and compared the metrics with the baseline model.

4.2. Zero-Shot Generalization

We evaluated the model’s zero-shot generalization by ‘leave-one-out’ approach. That is, we first chose one deepfake to be held out and trained the model on the remaining four deepfakes. After training, validation and test on the four deepfakes, we additionally tested the model on the held-out deepfake. Thus, we had five rounds for each of the six other fine-tuning methods, making 30 runs of the experiments in total. Since we did not train the baseline model and just directly test on the ‘held-out’ deepfake, the baseline performance as the benchmark was basically the same as we had in the adaptive in-domain generalization section, but the accuracy for each deepfake was separately stored.

4.3. Rehearsal Learning

To get a better understanding of how these different models relate, we decided to look at the individual impacts of how fine-tuning on each model effected the others. We extend the adaptive in-domain generalization experiment by introducing a new progressive fine-tuning curriculum using LoRA and Full Finetuning. In this setup, the model is fine-tuned incrementally on an expanding set of generative image domains, for instance- first only on StyleGAN2, then on StyleGAN2 + DALL-E, then StyleGAN2 + DALL-E + Midjourney, and so forth, until all five generative sources (StyleGAN2, DALL-E, Midjourney, Firefly, Stable Diffusion) are included. At each stage, we evaluate the model on all five domains’ test sets, measuring how well knowledge generalizes to domains not yet seen in training. The baseline is a Full Fine-Tuning approach, where the model is fully retrained on the cumulative data at each stage without special precautions.

This incremental curriculum is designed to reveal the model’s generalization capabilities: as new domains are added, does performance on unseen generative domains improve (forward transfer) and does performance on previously seen domains remain stable (avoiding forgetting)? This evaluation directly addresses the challenge that detectors often face a steep drop in accuracy on images from generative models that were not in their training set. For instance, a detector trained only on GAN-generated images may fail to recognize fakes from diffusion models. By adding generative domains one at a time, we can observe how the model copes with the distribution shift at each step.

4.4. Image Preprocessing

To ensure the consistency with precursor analysis [6], we used the pretrained image encoder for clip-vit-base-patch16, by calling ‘CLIPProcessor.from_pretrained(“openai/clip-vit-base-patch16”)’ to encode the images.

4.5. Metrics

We kept track of accuracy of the validation and test set as our primary metrics. Note this is an overall accuracy and all the dataset splits are 50% real and 50% fake. We also saved recall, precision as an indicator of classification performance, and also recorded the number of trainable parameters of each fine-tuning method.

4.6. Hyperparameters

Throughout the experiment, we used batch size = 16, learning rate = 0.0005, and number of epochs = 10 for all fine-tuning methods, except full fine-tuning. For full fine-tuning, we used learning rate = 0.00001. The method-specific hyperparameters are the following;

- **Adapter:** We used “hidden dimension” = 1024, “bottleneck” = 2048, and “dropout”= 0.1.
- **LoRA:** We set the rank of matrices $r = 8$, $\alpha = 16$, and dropout = 0.05. The target modules of the LoRA fine-tuning are the Q and V values of each self-attention layer.
- **BayesTune:** We set the target sparsity of the updated parameter by “sparsity_level” = 0.0004. For SGLD, we used “lr_model”= 0.0005, “lr_lambda”= 0.0005, “N_exp” = 12, “burning_steps” = 4000, “thinning_steps”= 30, “warmup_steps”= 2000, “num_epochs_sampling”= 10, “batch_size_sampling”= 4. Note, the batch_size_sampling is set to be small to expedite the convergence of the SGLD algorithm.
- **LN tuning:** We designated “pre_layernorm”, “layer_norm1”, “layer_norm2”, “post_layernorm”, “layernorm” as trainable parameters.

5. Results

5.1. Adaptive In-Domain Generalization

For our first step, we tested the model’s adaptive generalizability on in-domain data. Our goal was to fine-tune the Deepfake Detection model using a relatively small dataset, testing its ability to learn while maintaining its original performance. What we found is that the current state of the art models are able to successfully achieve almost-perfect

Model	Category	Valid. Acc	Test Acc	Precision	Recall	Old Task Acc	Trainable Params
Baseline	Pretrained	n/a	62.87%	81.64%	33.20%	78.69%	0
Full FT	FT	99.33%	99.53%	99.73%	99.33%	55.38%	303,284,226
Adapter	Additive PEFT	94.13%	94.33%	96.12%	92.40%	51.66%	4,197,376
LoRA	Reparam.PEFT	99.47%	99.73%	99.73%	99.73%	51.35%	786,432
VeRA	Reparam.PEFT	97.40%	98.07%	97.50%	98.67%	62.93%	49,536
LinearTail	Selective PEFT	88.27%	88.93%	91.83%	85.47%	73.92%	2,050
LN Tuning	Selective PEFT	98.73%	98.20%	99.47%	97.00%	55.10%	102,400
BayesTune	Selective PEFT	98.89%	98.73%	98.16%	99.33%	74.85%	121,856

Table 6: Adaptive In-Domain Generalization

classification with only 700 images from a new image generation model. Looking at the baseline (untrained) model, it shows 62% accuracy on the MADDE dataset which is achieved by predicting ”real” most of the time, resulting in high precision and low recall. This is understandable as the pre-trained model had not seen the diversity of fully generated deepfakes before. With Full fine-tuning and five of our seven PEFT methods achieving test accuracies over 98%, we can see that CLIP, the backbone of our baseline Deepfake Detection Model, is quite good at adaptive in-domain generalization. That’s to say, when introduced to a minimal amount of new data, it is able to learn and adapt to include that learning in its future tasks. This is exemplified even by our worst performer, LinearTail, which achieved 88.5% accuracy with just over two thousand trainable parameters, compared to 99.5% accuracy from the 303 million parameters trained in Full Fine-Tuning.

Among the all these methods, LoRA achieved the highest test accuracy of 99.73%, which was slightly higher than full fine-tuning. Not only that, LoRA was more efficient. Although not the most efficient method on our docket, training 786,432 parameters is just .2% of the original 300 million parameters we started with. LN tuning, which is the method adapted in Yermakov [6]’s original deepfake detection model, works reasonably well in our experiment. It achieved 98.2% test acc with just 102,400 trainable parameters. BayesTune had slightly more parameters (121,856) than LN tuning, and achieved the second best performance on test accuracy (98.73%) with much smaller trainable parameters compared to LoRA. As we have defined throughout this paper, generalization takes two paths. These results demonstrate successful adaptive in-domain generalization, but how does all of this stack in comparison with Zero-Shot Generalization.

Forgetting is the crucial problem when one needs to use the same model for all kinds of deepfake detection. To see if the model forgot the previous task (face-swapping fake detection), we also tested our trained models on DeepSpeak v1 Dataset [10][35]. The baseline model’s performance was measured by the video-level AUROC [6] but we used accuracy for consistency. Note that our DeepSpeak v1 Dataset also consisted 50% real images and 50% fake images.

The result showed that performant fine-tuning methods such as Full fine-tuning and LoRA ”forgot” the previous task and their accuracy is as bad as random guessing. This implies that those methods tend to overfit to the new task easily and lose the original ability when the old data is not available. In such cases, we could utilize is ’Learning without forgetting’ methods [33], where we synthesize the data for the old task by saving the output of the old head for the new data.

And surprisingly, BayesTune retained most of the original performance while it ranked at the third position in the test accuracy among those methods. It is more interesting that it trained more parameters than LNTuning, VeRA, and LinearTail and still kept its original ability. This memory-retention property could be a huge advantage in deepfake detection and other application where a single model needs to handle multiple types of tasks.

5.2. Zero-shot Generalization

We took two different experimental approaches to investigate how the Deepfake Detection Model generalizes on data outside of the domain it was trained on. Our first experiment, Leave-One-Out, tested how fine-tuning on all but one generative model impacts the models performance on the held-out set. In this way, we measured the model’s true generalizability for the unseen deepfake. What we found was a general under-performance.

Held-out Model	OOT Test Accuracy (Leave-one-out)				
	Stable Diffusion	StyleGAN2	Midjourney	firefly	Dall-E
Baseline	47.45%	76.60%	70.95%	53.05%	67.10%
Full FT	67.45%	53.95%	94.35%	96.00%	85.60%
Adapter	83.35%	86.05%	95.15%	88.65%	74.20%
LoRA	81.10%	67.80%	96.55%	93.45%	70.35%
VeRA	83.35%	94.05%	89.55%	90.70%	74.80%
LinearTail	76.35%	89.35%	92.70%	85.40%	73.95%
LN Tuning	82.55%	96.55%	81.00%	94.55%	72.04%
BayesTune	80.05%	93.65%	83.90%	90.15%	61.70%

Table 7: Zero-shot Generalization

The result in table 7 shows the model’s performance on the held-out data when fine-tuned on our remaining dataset. Although there were certain combinations of fine-tuning methods and held-out data that performed at a level we wanted to see (ie. fine-tuning with LoRA and holding out Midjourney with an accuracy of 96.55%) there was an overall lack of generalization. There was neither a clear ”If we use X PEFT method, the model can generalize!” moment, nor was there a catastrophic failure where every single model completely fell apart.

Notable observations of this experiment came from finding patterns of how different models performed across the board. For instance, we saw that, with the exception of full fine-tuning which still topped out at 85%, Dall-E, hovering at an average test accuracy or 73.8%, refused to improve significantly from the baseline 67.1%, regardless of the fine-tuning method. This could stem from the Dall-E’s unique model design, and we need to study the generative model itself to claim why this happens. Most of the fine-tuning methods worked well on the Firefly without training on it, which implies some resemblance between firefly and other generative models.

Interestingly, the performance on StyleGAN2 varies by a lot. For example, LoRA, the best fine-tuning method in the adaptive in-domain generalization experiment, did not perform well on StyleGAN2 as well as full fine-tuning. While we see evidence in the baseline results which exemplifies that our baseline model was originally trained on GAN-based tasks, the large variety in results for our lone GAN model could tell an interesting story. For instance, we can deduce that because Full Fine-Tuning tunes every single one of its parameters to only diffusion-based images, it completely loses its baseline ability to detect GAN artifacts. Similarly, we see that our bottom four rows (VeRA, LinearTail, LN Tuning, BayesTune), which account for our most parameter-efficient methods, have stronger results than LoRA (700k+ parameters) and Full Fine-Tuning (300 million). While Adapter does slightly break this trend, it still fails to perform as well as the more efficient fine-tuning methods. This pattern with StyleGAN could be attributed to the Forgetting Problem we discussed earlier. It also suggested to us that, although every model except StyleGAN is diffusion, some models are more similar than others. This was the next relationship we wanted to explore.

5.3. Progressive Rehearsal Learning

To examine how knowledge acquired from one generative domain transfers, or interferes, with other domains, we adopted a *progressive rehearsal curriculum*. Starting from StyleGAN2, we incrementally introduced additional domains in a fixed order and assessed detector performance after each addition. Two adaptation strategies were studied:

- **LoRA fine-tuning** (five experiments, *a-e*), wherein only rank-decomposition adapters are updated while the CLIP backbone remains frozen; rehearsal is implemented by mixing balanced minibatches of all domains encountered so far.
- **Full fine-tuning** (two experiments, *f-g*), wherein every backbone parameter is updated on the cumulative dataset of each stage, mirroring conventional practice in prior work.

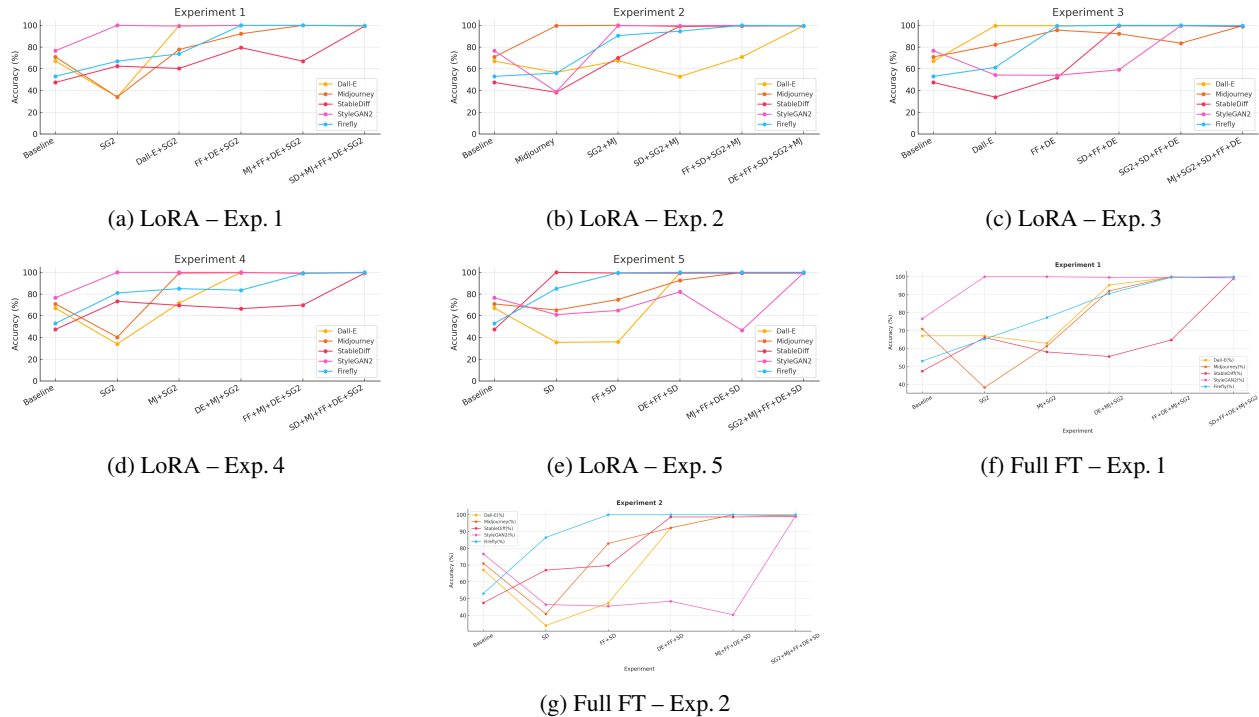


Figure 1: Progressive rehearsal-learning results. Subfigures (a)-(e) show five LoRA stages; (f)-(g) show two full fine-tuning stages

5.3.1 Stage-wise Analysis of the Progressive Curriculum (Experiment 1 (a))

- We began with **LoRA, Experiment 1 (a)**: after training solely on StyleGAN2, the detector achieved near-perfect performance on its own domain (99.8%) while retaining moderate zero-shot accuracy on unseen diffusion domains (52–71%). This initial stage established GAN-specific artifact filters that later served as a scaffold for incorporating more diverse cues.
- In the next training step, the addition of DALL-E increased its own accuracy to 91% and notably raised Firefly accuracy by +9.4 pp (see Table 8, row DALL-E → Firefly), indicating beneficial transfer from text-guided diffusion artifacts to Firefly-generated images. Other diffusion domains also experienced positive transfer at this stage.
- In the third step, we introduced Midjourney, producing mixed effects: Midjourney itself reached 96%, Firefly gained an additional +4 pp, while StyleGAN2 experienced only a negligible decline of −1.2 pp, suggesting that LoRA maintained resilience to interference despite introducing a highly stylized diffusion domain.
- The inclusion of Firefly increased its own accuracy to 98.5% without harming earlier domains, resulting in the first stage where all four domains exceeded 90% accuracy.
- Finally, we incorporated Stable Diffusion. Although Stable Diffusion reached 97.3% accuracy, the previously learned domains decreased by at most 2.6 pp. This final LoRA model achieved an average accuracy of 96.9% across all five domains, demonstrating stable forward transfer and minimal catastrophic forgetting.

Similarly, we carried out 5 experiments, in different dataset training order to find some consistent patterns of generalization that we have presented in Table 8.

Turning to the **Full Fine-Tuning Baselines**:

- **(f)** replicated the same domain order but updated all parameters at each stage. This approach sharply increased accuracy on the most recent domain but caused accuracy drops of at least 15 pp on earlier domains, with catastrophic forgetting most pronounced after Midjourney was added (StyleGAN2 declined from 99% to 65%).

- In (g), which reversed the order by starting with Stable Diffusion and then adding GAN sources, GAN accuracy was recovered at the expense of Midjourney and Stable Diffusion, confirming that full fine-tuning continually overwrites previously learned features.

Overall, comparing LoRA and Full Fine-Tuning, LoRA retained an average of +5.4 pp higher accuracy on previously seen domains and +6.1 pp on never-seen domains, confirming its superior stability-plasticity balance.

5.3.2 Pairwise Delta Matrix and Arrow Code

To further interpret inter-domain interactions, Table 8 presents the pairwise delta matrix, where entry Δ_{ij} records the accuracy change on domain j immediately after domain i is introduced. Positive Δ_{ij} (denoted \blacktriangle) indicates beneficial transfer; negative Δ_{ij} (\blacktriangledown) denotes interference. Aggregating these sign patterns across all stages yields the arrow code in Table 10, where \uparrow represents consistent improvement, \downarrow indicates consistent degradation, and \circ signifies mixed or neutral effects.

It is important to note that each Δ_{ij} reflects the cumulative training context; thus, observed gains after introducing Midjourney in E_3 are influenced not only by Midjourney but also by the earlier inclusion of DALL-E. Consequently, the arrow code captures directional tendencies rather than isolated causal effects. Future work could disentangle these interactions by varying the insertion order or leveraging domain-specific memory buffers to analyze first-order contributions.

The rehearsal curriculum jointly reveals *forward transfer* (performance on unseen domains) and *backward retention* (performance on seen domains). LoRA’s superiority is attributed to: (i) adapter updates that confine domain-specific information to a low-rank sub-space, avoiding destructive interference with frozen backbone features, and (ii) rehearsal minibatches that continuously refresh earlier domains, counteracting gradient drift. These observations accord with recent continual-learning studies on vision–language models, where residual or low-rank adapters preserve zero-shot accuracy better than full parameter updates [45].

In summary, the progressive rehearsal curriculum demonstrates that LoRA-based adaptation preserves domain-invariant structure and yields superior zero-shot robustness, while full fine-tuning remains vulnerable to catastrophic forgetting as the generative landscape expands.

Name	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Avg.	Name	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Avg.
Stable Diffusion							StyleGan2						
StyleGan2	\blacktriangle 15.05	\blacktriangle 31.72	–	\blacktriangle 25.88	–	\blacktriangle 24.22	Stable Diffusion	–	–	\blacktriangle 5.14	–	\blacktriangledown 15.52	\blacktriangledown 10.38
Midjourney							StyleGan2						
StyleGan2	\blacktriangledown 36.88	–	\blacktriangledown 8.71	\blacktriangledown 30.61	–	\blacktriangledown 25.40	Midjourney	–	\blacktriangledown 37.60	–	–	\blacktriangledown 35.48	\blacktriangledown 36.54
Firefly							StyleGan2						
StyleGan2	\blacktriangle 13.98	\blacktriangle 34.47	–	\blacktriangle 27.98	–	\blacktriangle 25.48	Firefly	–	–	\blacktriangledown 0.28	–	\blacktriangle 3.89	\blacktriangle 1.81
Dall-E							StyleGan2						
StyleGan2	\blacktriangledown 33.03	\blacktriangle 10.85	\blacktriangledown 0.33	\blacktriangledown 33.03	–	\blacktriangledown 5.63	Dall-E	–	–	\blacktriangledown 22.32	–	\blacktriangle 17.16	\blacktriangledown 2.58
Stable Diffusion							Dall-E						
Dall-E	\blacktriangledown 2.18	–	\blacktriangledown 13.53	\blacktriangledown 3.03	–	\blacktriangledown 6.25	Stable Diffusion	\blacktriangledown 14.54	–	–	–	\blacktriangledown 31.58	\blacktriangledown 23.06
Firefly							Dall-E						
Dall-E	\blacktriangle 6.80	–	\blacktriangle 8.23	\blacktriangledown 1.43	–	\blacktriangle 4.53	Firefly	–	\blacktriangle 17.95	–	–	–	\blacktriangle 17.95
Midjourney							Dall-E						
Dall-E	\blacktriangle 43.65	–	\blacktriangle 11.22	–	\blacktriangle 17.62	\blacktriangle 24.16	Midjourney	–	\blacktriangledown 10.48	–	\blacktriangle 37.63	–	\blacktriangle 13.58
Stable Diffusion							Firefly						
Firefly	\blacktriangle 19.22	–	\blacktriangle 17.91	\blacktriangle 3.27	–	\blacktriangle 13.47	Stable Diffusion	–	\blacktriangle 4.07	–	–	\blacktriangle 31.96	\blacktriangle 18.02
Midjourney							Firefly						
Firefly	\blacktriangle 14.57	–	\blacktriangle 13.49	–	\blacktriangle 9.75	\blacktriangle 12.60	Midjourney	–	\blacktriangle 3.06	–	\blacktriangle 3.98	–	\blacktriangle 3.52
Stable Diffusion							Midjourney						
Midjourney	\blacktriangledown 12.51	\blacktriangledown 9.13	–	\blacktriangledown 3.71	–	\blacktriangledown 8.45	Stable Diffusion	–	–	\blacktriangledown 3.37	–	\blacktriangledown 5.69	\blacktriangledown 5.13

Table 8: Pairwise performance deltas (Accuracy) (\blacktriangle = improvement, \blacktriangledown = decline) for five training experiments

5.3.3 Interpretation

Row (train-on)	What the arrow pattern says & why? Our Hypothesis
DALL-E2 (↑ Firefly, ↑ Midjourney, ↓ Stable Dif, ↔ StyleGAN2)	<i>Why Firefly & Midjourney</i> ↑: DALL-E2 is a text-guided diffusion with wide stylistic range, so LoRA learns broad diffusion cues that transfer to other style-rich models. <i>Why Stable Diff</i> ↓: SD3.5’s realism is higher and its denoising pipeline has newer tricks; cues DALL-E leaves behind are weaker or absent in SD3.5, so the detector overfits to DALL-E-specific texture quirks and misses SD fakes. <i>Why StyleGAN2</i> ↔: GANs share some generic frequency artifacts the detector latches onto, but many DALL-E-specific cues are irrelevant for StyleGAN2, so gains and losses cancel → mixed.
Firefly3 (↑ across the board)	Firefly is the most diverse and photoreal source; LoRA must learn deep, style-agnostic “synthetic” signals (tiny spectral noise, perfect prompt alignment). Those signals appear in <i>all</i> other generators, so performance rises everywhere, creating a “Firefly lifts all boats” effect.
Stable Diff3.5 (↑ DALL-E, ↑ Firefly, ↑ StyleGAN2, ↓ Midjourney)	<i>Why DALL-E & Firefly</i> ↑: Diffusion-specific artifacts learned from SD3.5 transfer to DALL-E and Firefly. <i>Why StyleGAN2</i> ↑: GANs share some residual synthetic frequency patterns, so StyleGAN2 also benefits. <i>Why Midjourney</i> ↓: Midjourney’s strong cinematic palette lies outside SD’s natural-photo manifold; the detector mistakes Midjourney art for “real art” → AUROC drops.
StyleGAN2 (↔ DALL-E & Midjourney, ↑ Firefly & Stable Diff)	LoRA trained on a GAN captures subtle GAN-frequency artifacts that are still present (though faint) in diffusion outputs → ↑ for Firefly & SD. But it sees no text-guided semantic coherence or vivid colour grading, so results on DALL-E and Midjourney swing between gains and losses → ↔.
MidjourneyV6 (↑ Firefly, ↔ DALL-E, ↓ Stable Diff & StyleGAN2)	Detector locks onto Midjourney’s distinctive cinematic grading. This style bias transfers to similarly stylised Firefly images and partly to DALL-E (some prompts), but harms detection of more plain SD3.5 photos and StyleGAN2 GAN images → ↓.

Table 9: Each row: train on the generator at left, then test on the unseen generator named in parentheses

	Dall-E	Firefly	Stable Diffusion	StyleGAN2	Midjourney
Dall-E	—	↑	↓	↔	↑
Firefly	↑	—	↑	↔	↑
Stable Diffusion	↑	↑	—	↑	↓
StyleGAN2	↔	↑	↑	—	↓
Midjourney	↔	↑	↓	↓	—

Table 10: Arrow key: ↑ consistent gain, ↓ consistent drop, ↔ mixed performance.

5.3.4 Caveats

Pairwise deltas should not be interpreted in isolation. Because each stage operates on the *union* of all previously seen domains, Δ_{ij} is confounded by cumulative context. For instance, the gain that Firefly receives when Midjourney is added is likely mediated by features first shaped by DALL-E. Hence, the arrow code captures directional tendencies rather than isolated causal effects. Future work could disentangle these interactions by varying the insertion order or by using domain-specific replay buffers to analyse first-order contributions.

In summary, rehearsal learning with LoRA adapters delivers superior cross-domain generalization, maintains stability as the training distribution expands, and exposes nuanced positive and negative transfer patterns that are obscured by conventional full fine-tuning.

6. Conclusion

For **Adaptive In-Domain Generalization**, we saw that the current state of the art models are able to successfully achieve almost-perfect classification with only 700 images from a new image generation model. With Full fine-tuning and five of our seven PEFT methods achieving test accuracies over 98%, we can see that CLIP, the backbone of our baseline Deepfake Detection Model, is quite good at adaptive in-domain generalization. We also saw that in the cases where the forgetting is problematic, BayesTune could be the best choice to learn new tasks while retaining the original ability.

However, when it comes to **Zero-shot generalization**, what we found was a general under-performance. Although there were certain combinations of fine-tuning methods and held-out data that performed at a level we wanted to see (ie. fine-tuning with LoRA and holding out Midjourney with an accuracy of 96.5%) there was an overall lack of generalization. There was neither a clear "If we use X PEFT method, the model can generalize!" moment, nor was there a catastrophic failure where every single model completely fell apart. Through the follow-up **rehearsal learning**, we saw the patterns that training on a specific model improves the performance on other unseen deepfake models, which implies moderate generalizability of the detection models. However, the gain by learning other types of deepfakes was not enough to enhance the performance more than 90%, and in some cases it even harmed the performance on the other unseen deepfakes.

What we have just demonstrated is one can achieve adaptive in-domain generalization successfully, but even the most powerful fine-tuning methods such as full fine-tuning and LoRA, cannot be a silver bullet for zero-shot generalization. Although we could expect moderate generalization over similar generative models, we still need to collect the data and train the model when over 90% accuracy is necessary for the use case.

That being said, these series of experiments showed that each fine-tuning model exhibited different learning journey and generalizability. This implies the possibility that we can combine those fine-tuning methods to achieve the better generalizability over deepfakes. Moreover, our cumulative training results presented correlations between image generation models. We could utilize this experiment to detect resemblance among the generative models, and such follow-up experiment can be a good future work of our research.

7. Limitations

As we have only experimented with one base model whose backbone is CLIP, other base models might not respond equally well to the same fine-tuning strategies. The PEFT methods we used and the results we obtained are specific to the particular base model we used, and the same might not be applicable to other base models. Also, we have explored a subset of PEFT methods and our findings might not generalize to other PEFT methods. This result is specific to the five generative models we selected. Especially speaking, zero-shot generalization and the correlation between deepfake models are heavily dependent on the order and specific combinations of the deepfake models, so this result itself does not generalize to the other deepfakes. Nevertheless, the scope of this project has demonstrated the adaptive in-domain generalization achieved robust performances with a small amount of data, across the wide range of PEFT. Moreover, the interesting behavior we observed in zero-shot generalization experiment should be a good lead to the successive research and experiment.

8. Future Work

Our research so far has left us more curious than satisfied. Although promising, we lacked the scale necessary to establish causality and confidence in our claims. Our future works reflects that motivation. This paper can be summed up with two areas of interest, the differentiation between PEFT methods for different tasks, and the underlying architectures driving our image generation models. Diving deeper into both avenues would allow us to understand why zero-shot generalizability for deepfakes is so difficult, and potentially reveal relationships and techniques for how to leverage architectural similarities to improve generalizability.

References

- [1] Adobe, "Firefly V3 API Documentation," *Adobe Developer*, Accessed: Apr. 7, 2025. https://developer.adobe.com/firefly-services/docs/firefly-api/guides/api/image_generation/V3/
- [2] A. Kulkarni and A. Shrestha, "Multispectral Image Analysis using Decision Trees," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 6, 2017, doi: <https://doi.org/10.14569/ijacsa.2017.080602>.

- [3] A. Kumar and S. P. Srivastava, “A Thorough Investigation on Image Forgery Detection,” *Cmes-computer Modeling in Engineering & Sciences*, vol. 134, no. 3, pp. 1489–1528, Jan. 2023, doi: <https://doi.org/10.32604/cmes.2022.020920>.
- [4] A. Radford *et al.*, “Learning Transferable Visual Models From Natural Language Supervision,” *arXiv.org*, Feb. 26, 2021. <http://arxiv.org/abs/2103.00020>.
- [5] Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta, “Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning,” 2020. <https://arxiv.org/abs/2012.13255>
- [6] A. Yermakov, J. Cech, and J. Matas, “Unlocking the Hidden Potential of CLIP in Generalizable Deepfake Detection,” *arXiv (Cornell University)*, Mar. 2025, doi: <https://doi.org/10.48550/arxiv.2503.19683>.
- [7] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen, “LoRA: Low-Rank Adaptation of Large Language Models,” 2021. <https://arxiv.org/abs/2106.09685>
- [8] E. J. Hu *et al.*, “LoRA: Low-Rank Adaptation of Large Language Models,” *arXiv:2106.09685*, Jan. 2021, doi: <https://doi.org/10.48550/arxiv.2106.09685>.
- [9] Elad Ben-Zaken, Yoav Levine, and Lior Wolf, “BitFit: Simple Parameter-Efficient Fine-Tuning for Transformer-Based Masked Language Models,” 2021. <https://arxiv.org/abs/2106.10199>
- [10] H. Farid Lab, “DeepSpeak Dataset (hosted on Hugging Face),” *Hugging Face*, Accessed Apr. 7, 2025. <https://huggingface.co/faridlab>
- [11] H. Farid, “Creating, Using, Misusing, and Detecting Deep Fakes,” *Journal of Online Trust and Safety*, vol. 1, no. 4, Sep. 2022, doi: <https://doi.org/10.54501/jots.v1i4.56>.
- [12] K. Tsigos, E. Apostolidis, S. Baxevanakis, S. Papadopoulos, and V. Mezaris, “Towards Quantitative Evaluation of Explainable AI Methods for Deepfake Detection,” *arXiv:2404.18649*, Apr. 2024, doi: <https://doi.org/10.48550/arxiv.2404.18649>.
- [13] Layer Normalization Tuning, “Tuning LayerNorm in Attention: Towards Efficient Multi-Modal LLM Finetuning,” *ICLR 2024*, <https://openreview.net/forum?id=YR3ETaElNK>
- [14] Midjourney, “Midjourney Discord Bot Docs,” *Midjourney Documentation*, Accessed: Apr. 7, 2025. <https://docs.midjourney.com/hc/en-us/articles/32637339216013-Discord-Direct-Messages>
- [15] M. Kim and T. Hospedales, “BayesTune: Bayesian Sparse Deep Model Fine-tuning,” *Advances in Neural Information Processing Systems*, vol. 36, pp. 65317–65365, Dec. 2023, Accessed: Mar. 08, 2025.
- [16] M. Tan and Q. V. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” *arXiv.org*, Sep. 11, 2020. <http://arxiv.org/abs/1905.11946>
- [17] NVIDIA Labs, “StyleGAN2,” *GitHub*, Accessed: Apr. 7, 2025. <https://github.com/NVLabs/stylegan2?tab=readme-ov-file>
- [18] OpenAI, “DALL-E: Image Generation Docs,” *OpenAI Platform*, Accessed: Apr. 7, 2025. <https://platform.openai.com/docs/guides/image-generation>
- [19] Quoc Le, Tamás Szepesvári, and Alex Smola, “Fastfood—Approximating Kernel Expansions in Loglinear Time,” *Proceedings of the International Conference on Machine Learning (ICML)*, 2013. <https://proceedings.mlr.press/v28/le13.html>
- [20] R. Mattioli and A. Malatras, “Foresight Cybersecurity Threats for 2030,” *European Union Agency for Cybersecurity (ENISA)*, 2024, doi: <https://doi.org/10.2824/349493>.

- [21] S. A. Khan and D.-T. Dang-Nguyen, “CLIPping the Deception: Adapting Vision-Language Models for Universal Deepfake Detection,” arXiv:2402.12927, Feb. 2024, doi: <https://doi.org/10.48550/arxiv.2402.12927>.
- [22] S. Singh and V. K. Sehgal, “Image Forgery Detection Model using CNN Architecture with SVM Classifier,” *2022 Seventh International Conference on Parallel, Distributed and Grid Computing (PDGC)*, Solan, Himachal Pradesh, India, 2022, pp. 263-268, doi: [10.1109/PDGC56933.2022.10053298](https://doi.org/10.1109/PDGC56933.2022.10053298).
- [23] Stability AI, “Stable Diffusion Ultra API Reference,” *Stability AI Docs*, Accessed: Apr. 7, 2025. <https://platform.stability.ai/docs/api-reference#tag/Generate/paths/~1v2beta~1stable-image~1generate~1ultra/post>
- [24] T. B., “70k-faces Kaggle dataset,” *Kaggle*, Accessed Apr. 7, 2025. <https://www.kaggle.com/datasets/tunguz/70000-real-faces-1>
- [25] Y. Li *et al.*, “LoftQ: LoRA-Fine-Tuning-Aware Quantization for Large Language Models,” arXiv:2310.08659, Nov. 2023, doi: <https://doi.org/10.48550/arxiv.2310.08659>.
- [26] Z. Han, C. Gao, J. Liu, J. Zhang, and S. Q. Zhang, “Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey,” arXiv:2403.14608, Apr. 2024, <http://arxiv.org/abs/2403.14608>.
- [27] Z. Liu *et al.*, “Swin Transformer V2: Scaling Up Capacity and Resolution,” *arxiv.org*, Nov. 2021, doi: <https://doi.org/10.48550/arXiv.2111.09883>.
- [28] Z. Wang *et al.*, “DIRE for Diffusion-Generated Image Detection,” arXiv:2303.09295, Jan. 2023, doi: <https://doi.org/10.48550/arxiv.2303.09295>.
- [29] N. Houlsby *et al.*, “Parameter-Efficient Transfer Learning for NLP,” *arXiv.org*, Jun. 13, 2019. <https://arxiv.org/abs/1902.00751>
- [30] K. D. Jan, T. Blankevoort, and Y. M. Asano, “VeRA: Vector-based Random Matrix Adaptation,” *arXiv (Cornell University)*, Jan. 2023, doi: <https://doi.org/10.48550/arxiv.2310.11454>.
- [31] G. Alain and Y. Bengio, “Understanding intermediate layers using linear classifier probes,” *arXiv.org*, Nov. 22, 2018. <https://arxiv.org/abs/1610.01644> (accessed Apr. 23, 2024).
- [32] A. Tomihari and I. Sato, “Understanding Linear Probing then Fine-tuning Language Models from NTK Perspective,” *arXiv (Cornell University)*, May 2024, doi: <https://doi.org/10.48550/arxiv.2405.16747>.
- [33] Z. Li and D. Hoiem, “Learning without forgetting,” arXiv.org, doi: <https://doi.org/10.48550/arXiv.1606.09282> (accessed May 8, 2025).
- [34] R. Caruana, “Multitask learning,” *Machine Learning*, vol. 28, no. 1, pp. 41–75, 1997, doi: <https://doi.org/10.1023/a:1007379606734>.
- [35] S. Barrington, M. Bohacek, and H. Farid, “DeepSpeak Dataset v1.0,” arXiv (Cornell University), Aug. 2024, doi: <https://doi.org/10.48550/arxiv.2408.05366>.
- [36] X. Zhang, S. Karaman, and S.-F. Chang, “Detecting and Simulating Artifacts in GAN Fake Images,” Jul. 2019, doi: <https://doi.org/10.48550/arxiv.1907.06515>.
- [37] S. -Y. Wang, O. Wang, R. Zhang, A. Owens and A. A. Efros, “CNN-Generated Images Are Surprisingly Easy to Spot... for Now,” *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, 2020, pp. 8692-8701, doi: [10.1109/CVPR42600.2020.00872](https://doi.org/10.1109/CVPR42600.2020.00872).
- [38] X. Guo, X. Liu, Z. Ren, S. Grosz, I. Masi, and X. Liu, “Hierarchical Fine-Grained Image Forgery Detection and Localization,” Mar. 2023, doi: <https://doi.org/10.48550/arxiv.2303.17111>.
- [39] L. Verdoliva, “Media Forensics Based on Deep Learning: A Review,” *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 131–148, May 2020, doi: [10.1109/MSP.2020.2975740](https://doi.org/10.1109/MSP.2020.2975740).

- [40] S. Y. Wang, O. Wang, R. Zhang, A. Owens, and A. A. Efros, “CNN-Generated Images Are Surprisingly Easy to Spot ... for Now,” arXiv preprint, Dec. 2019, doi: 10.48550/arXiv.1912.11035.
- [41] Q. Xuan, Y. Song, J. Dong, and T. Tan, “Generalization in Deepfake Detection: A Bayesian Perspective,” arXiv preprint, Dec. 2019, doi: 10.48550/arXiv.1912.13458.
- [42] Y. Mirsky and W. Lee, “The Creation and Detection of Deepfakes: A Survey,” *Proceedings of the 2018 Workshop on Information Hiding and Multimedia Security (IH&MMSec)*, Jun. 2018, pp. 1–11, doi: 10.1145/3206004.3206024.
- [43] Y. Li, X. Yang, P. Sun, H. Qi, and S. Lyu, “Face X-Ray for More General Face Forgery Detection,” arXiv preprint, Apr. 2020, doi: 10.48550/arXiv.2004.11138.
- [44] M. Zhang, Y. Liu, and X. Dong, “DefakeHop++: Hygiene-Bias-Free Deepfake Detection,” arXiv preprint, Mar. 2021, doi: 10.48550/arXiv.2103.06929.
- [45] E. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, “LoRA: Low-Rank Adaptation of Large Language Models,” Jun. 2021, doi: <https://doi.org/10.48550/arXiv.2106.09685>.