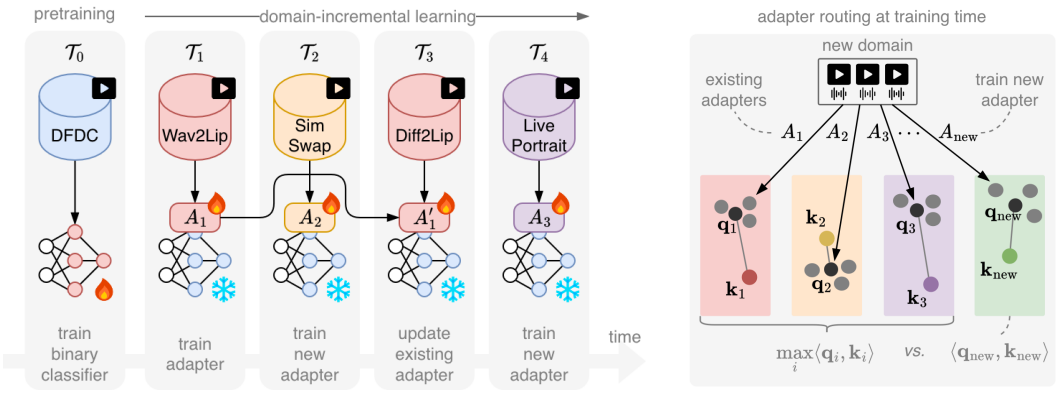
MULTIMODAL DEEPFAKE DETECTION VIA DOMAIN-INCREMENTAL MULTI-ADAPTER LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

DeepFake detectors are typically trained in a static setting and struggle to generalize to emerging manipulation techniques. As generative methods evolve rapidly, the ability to incrementally extend models to new domains without full retraining becomes critical. We introduce DEFEND, a framework for domain-incremental multimodal DeepFake detection with parameter-efficient updates. Unlike prior work that focuses on dataset-level shift, we consider a fine-grained incremental setting, where *new manipulation types* arrive sequentially. To this end, we construct a chronological stream of 23 domains, spanning from early face-swapping methods to recent multimodal forgeries. DEFEND builds on a frozen pretrained audio-visual backbone and leverages Low-Rank Adaptation (LoRA) to enable efficient domain-specific updates while mitigating catastrophic forgetting. Importantly, we introduce a dynamic adapter expansion guided by a learned key-query routing strategy. This mechanism allows the model to reuse existing adapters for similar manipulations while expanding its capacity when encountering novel ones. We evaluate DEFEND on three benchmarks, KoDF, DeepSpeak and MAVOS-DD. On BRAVEn, DEFEND reaches 89.7% average AUC under automatic routing, 8.5 points above the strongest continual learning baseline, with only 11 adapters for 23 domains; on CLIP+CLAP, it is on par with the best continual learning baselines.



(a) (b)

Figure 1: (a) **Problem statement and DEFEND.** A base model is pretrained on $\mathcal{T}_0$ (e.g., DFDC; Dolhansky et al. 2020). As new manipulation types $\mathcal{T}_1, \mathcal{T}_2, \ldots$ arrive sequentially, the model is extended via low-rank adapters. A new adapter is added *if* the incoming domain is sufficiently different from previously seen ones; otherwise the most suitable existing adapter is updated (e.g., $\mathcal{T}_3$ reuses A_1). Illustrative example. (b) **Routing at training time.** Incoming samples are embedded through each existing adapter and a freshly trained A_{new}. The resulting average query embeddings $\mathbf{q}$ are compared against the stored key $\mathbf{k}$ of each adapter to decide routing.

1 INTRODUCTION

The rapid advancement of generative models produces increasingly realistic DeepFakes that blur the boundary between authentic and manipulated content. Beyond creative applications, they enable misinformation, identity fraud, and election interference — as seen in Romania, Canada, and the

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Czech Republic in 2025 alone (Stockwell, 2025). Audio-visual manipulations are arguably the most consequential, fabricating both what a person appears to do and say. Our framework therefore operates on joint audio-visual inputs and also supports purely visual ones.

Existing DeepFake detectors (Haliassos et al., 2022; Oorloff et al., 2024; Du et al., 2025; Liu et al., 2026) assume a fixed training set and aim to generalize to unseen manipulation types. This static setup does not reflect realistic conditions: in practice, new manipulation techniques emerge continuously as generative models evolve, and retraining from scratch on every new manipulation is computationally expensive and requires storing all past training data, which may be unavailable due to storage or privacy constraints.

We instead model the continuous nature of manipulation data explicitly. Generators emerge in chronological order, each defining a distinct manipulation domain, and a detector must adapt to new types while retaining performance on earlier ones - a *domain-incremental learning* setting (Van de Ven et al., 2022). Prior work in this space (Pan et al., 2023; Tian et al., 2024; Cheng et al., 2025; Shen et al., 2025) typically uses a fixed-capacity model that is incrementally updated, which can reach a capacity limit as the number of manipulations grows (cf. the single adapter in Table 5).

To address this, we propose DEFEND (multimodal DEepFake dEtection via domain-iNcremental multi-aDapter learning), a dynamically expandable architecture whose capacity grows with the data stream (Figure 1a). We learn a set of low-rank adapters (LoRA; Hu et al. 2022) over a pretrained model. Because adapters are reused across related domains, far fewer adapters than domains are needed, while shared characteristics across similar manipulations are preserved. A learned routing mechanism (Figure 1b) decides at both training and inference time whether incoming data warrants a new adapter or should be handled by an existing one.

We evaluate on a chronological, manipulation-level benchmark of **23 domains** drawn from KoDF (Kwon et al., 2021), DeepSpeak (Barrington et al., 2024), and MAVOS-DD (Croitoru et al., 2025) — substantially broader than prior benchmarks that consider four domains. The included techniques span early face-swapping to state-of-the-art multimodal forgeries, with additional variation from recording conditions. On BRAVEn, DEFEND outperforms all incremental baselines with only 11 adapters for 23 domains, and on CLIP+CLAP it is on par with the best continual learning baselines.

2 Related work

Multimodal DeepFake detection. Detection of video DeepFakes often exploits audio-visual synchronicity, either in an unsupervised manner (Feng et al., 2023; Reiss et al., 2023; Liang et al., 2024; Smeu et al., 2025) or via supervised classifiers built on self-supervised representations (Oorloff et al., 2024; Klemt et al., 2025; Wu et al., 2025; Koutlis & Papadopoulos, 2025). We follow the latter approach, extracting visual features with CLIP (Radford et al., 2021) — proven effective for image DeepFake detection (Ojha et al., 2023; Koutlis & Papadopoulos, 2024; Khan & Dang-Nguyen, 2024) — and audio features with CLAP (Wu et al., 2023), whose cross-modal language alignment has shown strong results on DeepSpeak (Barrington et al., 2024).

(Domain-)Incremental learning in DeepFake detection. Tasks are typically defined per generator, though some works use datasets as domain boundaries (Xiong et al., 2024; Fontana et al., 2025). Common methodological choices include elastic weight consolidation (Kirkpatrick et al., 2017), knowledge distillation (Hinton et al., 2015; Li & Hoiem, 2017), and iCaRL-style replay (Rebuffi et al., 2017; Tassone et al., 2024; Laiti et al., 2024; Wang et al., 2022a; Marra et al., 2019; Li et al., 2023). Prior studies are predominantly unimodal (Zhang et al., 2023; Chen et al., 2025a; Febrinanto et al., 2025; Salvi et al., 2025; Marra et al., 2019; Li et al., 2023; Laiti et al., 2024); recent video-based work (Zhang et al., 2025) does not incorporate audio. Our approach most closely relates to Xiong et al. (2024) in its use of lightweight adapters, but replaces heuristic similarity measures with a learned routing problem that enables principled adapter reuse and expansion.

Dynamically expanding models. Growing model capacity to mitigate forgetting ranges from one model per task (Rusu et al., 2016; Aljundi et al., 2017) — connected via lateral connections or warm-started from the most similar prior model — to expanding only parts of the network (Li et al., 2019) or using low-rank adapters (Wang et al., 2025; Zhou et al., 2024; Zhang et al., 2022). Our key-query routing mechanism learns *when* to reuse versus expand: it replaces an absolute similarity threshold with a learned, relative comparison against a freshly trained candidate adapter.

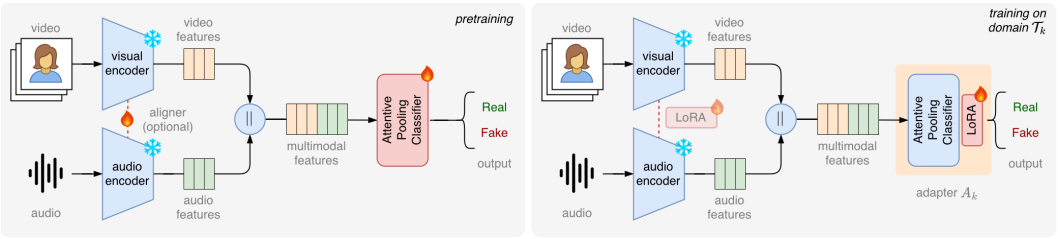


Figure 2: **DEFEND architecture for the domain-incremental learning problem.** (Left) The model is first pretrained on DFDC. (Right) Next, the model parameters are frozen, and learning on a new domain $\mathcal{T}_k$ is performed using low-rank adaptation via adapter A_k.

3 DEFEND METHODOLOGY

Let $\mathcal{F}$ be a DeepFake detector pretrained on a base domain $\mathcal{T}_0$. Given K manipulation types arriving sequentially $\mathcal{T}_1, \ldots, \mathcal{T}_K$, our goal at each step k is to adapt $\mathcal{F}$ to $\mathcal{T}_k$ while mitigating forgetting of prior domains. DEFEND (multimodal DEepFake dEtection via domain-iNcremental multi-aDapter learning) maintains a growing set of lightweight adapters (§ 3.1). For each new domain, a routing mechanism decides whether to reuse an existing adapter or allocate a new one (§ 3.2); the same mechanism operates at inference time (§ 3.4). Adapters are trained with binary cross-entropy and a margin loss that separates real from fake embeddings, which also structures the embedding space used for routing (§ 3.3).

3.1 LoRA-BASED INCREMENTAL ADAPTATION

We apply DEFEND to a dual-stream model architecture (Figure 2). The model $\mathcal{F}$ consists of frozen visual and audio encoders that extract features from their respective modalities, followed by an Attentive Pooling Classifier (Chen et al., 2024; Bardes et al., 2024) that operates on multimodal feature sequences of arbitrary length; the multimodal features are obtained by concatenating the video and audio features along the temporal dimension. The architecture also includes an optional aligner (learnable self- and cross-attention layers), used when the audio and visual encoders are not already aligned, as for CLIP+CLAP.

During pretraining on the base domain $\mathcal{T}_0$, only the aligner and the classifier are trained; then these components are kept frozen. When a new domain $\mathcal{T}_k$ arrives, DEFEND adapts by introducing additional weights through Low-Rank Adaptation (LoRA; Hu et al. 2022). Concretely, an adapter A is applied to the linear layers of the cross-attention in the aligner and of the classifier. Since LoRA introduces only low-rank parameter updates, a stable adaptation is achieved by allowing the model to incorporate new knowledge while preserving previously learned representations. Crucial for our approach is that we do not rely on a single adapter, but maintain a dynamic set of adapters $\{A_1, A_2, \ldots, A_n\}$ that can be reused or expanded depending on the incoming domain.

3.2 ROUTING AT TRAINING TIME

When data from a new domain $\mathcal{T}_k$ arrives, we must decide whether (i) to reuse and update an existing adapter with $\mathcal{T}_k$, or (ii) to create a new adapter. We reuse an adapter when the incoming domain is sufficiently similar to a previous domain; otherwise, we create a new adapter specialized for the new domain. Concretely, we maintain a prototype for the domains seen by each adapter and compare those with the prototype of the current domain (Figure 1b). We consider two routing approaches that differ in the similarity functions and decision rules (threshold-based or not).

DEFEND-SIM. Our first (naive) routing approach is based on cosine similarity. Formally, each adapter A_i is associated with a prototype centroid $\mathbf{c}_i$, computed as the average representation of the last domain it was trained on. For a new domain $\mathcal{T}_k$, we pass its samples through the model with adapter A_i and compute the resulting centroid $\mathbf{c}_i^k$. We measure how compatible adapter A_i is with domain $\mathcal{T}_k$ by computing the cosine similarity between $\mathbf{c}_i^k$ and the corresponding centroid c_i. If $\max_i \cos(\mathbf{c}_i^k, \mathbf{c}_i) > \tau$, the most similar adapter is *reused* and consequently updated using $\mathcal{T}_k$;

otherwise, a new adapter is created and trained. We refer to this routing as DEFEND-SIM and provide its pseudocode in Algorithm 2 (§ B). One drawback of this method is that each centroid is recomputed after every update from the last domain only (Algorithm 2, L11), so it no longer represents the earlier domains of the adapter, which makes routing less reliable.

DEFEND-KQ. To account for adapter representations changing over time, we propose a learnable key-query routing mechanism. The idea is to have each adapter associated with a learnable key representing the domain captured by that adapter; similarly, the incoming domain is represented by a query produced by applying a learnable function g to its audio-visual representations. Specifically, for each adapter A_i, we extract representations $\mathbf{z}_i^k$ from domain $\mathcal{T}_k$ and use g to map them to the corresponding query; queries and keys live in a 256-d routing space: g maps each embedding to it ($d \to 512 \to 256$), and each key is initialized as the average embedding of its adapter's domains, mapped to 256-d by a fixed random projection P and L2-normalized (Algorithm 1, L13, L23). To measure how well an adapter A_i already explains the new domain $\mathcal{T}_k$, we compute a compatibility score s_i^k as temperature-scaled inner product between its queries and corresponding keys: $(\mathbf{q}_i^k \cdot \mathbf{k}_i^\top)/T$, where T is the temperature parameter (L8). To understand how well a new adapter would fit the data, we train a temporary adapter A_{new} on $\mathcal{T}_k$ and compute its compatibility score s_{new} (L10–14). If this score exceeds the scores of all existing adapters, then the temporary adapter is retained and added to the adapter set (L17–18); otherwise, the highest-scoring existing adapter is reused and updated (L20–23).

Algorithm 1 Key-query routing

1: **Input**: adapter set $\mathcal{M}$, domain data $\mathcal{S}_k$
2:
3: **Function** UPDATEMODEL($\mathcal{M}$, $\mathcal{S}_k$)
4: **for each** adapter $A_i \in \mathcal{M}$ **do**
5: $\quad \mathbf{z}_i \leftarrow$ GETEMBEDDINGS($\mathcal{S}_k$, A_i)
6: $\quad \mathbf{q}_i \leftarrow$ AVG(NORMALIZE($g(\mathbf{z}_i)$))
7: $\quad \mathbf{k}_i \leftarrow A_i$.key
8: $\quad s_i \leftarrow (\mathbf{q}_i \cdot \mathbf{k}_i^\top)/T$
9: **end for**
10: $A_{\text{new}} \leftarrow$ TRAINNEWADAPTER($\mathcal{S}_k$)
11: $\mathbf{z}_{\text{new}} \leftarrow$ GETEMBEDDINGS($\mathcal{S}_k$, A_{new})
12: $\mathbf{q}_{\text{new}} \leftarrow$ AVG(NORMALIZE($g(\mathbf{z}_{\text{new}})$))
13: $\mathbf{k}_{\text{new}} \leftarrow$ NORMALIZE(P AVG($\mathbf{z}_{\text{new}}$))
14: $s_{\text{new}} \leftarrow (\mathbf{q}_{\text{new}} \cdot \mathbf{k}_{\text{new}}^\top)/T$
15: $i^* \leftarrow \arg\max_i \{s_i\}$
16: **if** $s_{\text{new}} > s_{i^*} + m$ **then**
17: $\quad \mathcal{M} \leftarrow \mathcal{M} \cup \{A_{\text{new}}\}$
18: $\quad A^* \leftarrow A_{\text{new}}$
19: **else**
20: $\quad A^* \leftarrow A_{i^*}$
21: $\quad A^* \leftarrow$ TRAINADAPTER(A^*, $\mathcal{S}_k$)
22: $\quad \mathbf{z}_{\text{new}} \leftarrow$ GETEMBEDDINGS($\mathcal{S}_k$, A^*)
23: $\quad \mathbf{k}_{\text{new}} \leftarrow$ NORMALIZE(P AVG($\mathbf{z}_{\text{new}}$))
24: **end if**
25: A^*.key $\leftarrow \mathbf{k}_{\text{new}}$
26: TRAINROUTER(g, keys, stored $\mathbf{z}$ of all seen domains + $\mathbf{z}_{\text{new}}$)

After training, the selected adapter's key is reset to the mean embedding of the current domain (Algorithm 1, L25), like the DEFEND-SIM centroid; unlike it, the key is then optimized with g on the stored embeddings of all seen domains (L26), so it represents every domain assigned to the adapter. Since $\mathbf{k}_{\text{new}}$ and $\mathbf{q}_{\text{new}}$ come from the same data, s_{new} is systematically favored; we therefore require it to exceed s_{i^*} by a reuse margin m (L16; m=0.3, tuned on validation in § 5).

After training the adapter, its embeddings $\mathbf{z}_{\text{new}}$ of $\mathcal{T}_k$ are stored, and the router is retrained on the stored adapter embeddings of all seen domains (Algorithm 1, L26). Samples from each domain are passed to g, and the scores s_i are obtained. For a given domain, g and the keys are optimized using a cross-entropy loss to maximize s_y for the specific adapter associated with that domain:

$$\mathcal{L}_{\text{router}} = -\log \frac{\exp(s_y)}{\sum_{i=1}^{n} \exp(s_i)}$$

where n is the number of adapters and y is the index of the adapter that was trained on the corresponding domain. $\mathcal{L}_{\text{router}}$ only updates the router (g and the keys), not the adapters.

Computational cost For each new domain, DEFEND-KQ trains the candidate adapter A_{new} once; if an existing adapter is reused, that adapter is additionally updated on $\mathcal{T}_k$ and A_{new} is discarded, so each domain requires at most two adapter trainings. Since adapters are LoRA modules, each training only updates a small fraction of the parameters. Computing the queries requires one forward pass over $\mathcal{T}_k$ per existing adapter, and the router is a lightweight MLP retrained on the stored embeddings, which is negligible compared to adapter training. At inference, routing adds one forward pass of the detection head and of g per sample to compute the query, before the selected adapter is applied.

3.3 TRAINING OBJECTIVE

Adapter training uses binary cross-entropy ($\mathcal{L}_{\text{bce}}$) combined with a margin loss that encourages real samples to cluster and be separated from fake samples:

$$\mathcal{L}_{\text{marg}} = \frac{1}{|\mathcal{P}|} \sum_{(i,j)\in\mathcal{P}} (1 - s_{ij}) + \frac{1}{|\mathcal{N}|} \sum_{(i,k)\in\mathcal{N}} \max(0, s_{ik} - m), \tag{1}$$

where $\mathcal{P}$ pairs authentic samples, $\mathcal{N}$ pairs authentic with fake, and m is the cosine margin. The final objective is $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{bce}} + \lambda \mathcal{L}_{\text{marg}}$. We find that $\mathcal{L}_{\text{marg}}$ leads to more adapter reuse and thus fewer parameters (§ 5).

3.4 Inference Procedure

At inference time, the input domain is unknown, so an adapter must be selected automatically. **DEFEND-SIM**, a simple baseline, selects the adapter whose centroid has the highest cosine similarity to the input representation. Our main method, **DEFEND-KQ**, projects the input into the learned routing space, which accounts for adapter drift, and selects the adapter with the highest score. Real and fake samples are routed in the same way, as neither label nor domain is known; $\mathcal{T}_0$ has no adapter and no key, so its samples go to incremental adapters, lowering the DFDC AUC. As a reference, **ground-truth routing** (GT-SIM or GT-KQ) selects the adapter trained on the true domain of the input. While not applicable in practice, it isolates classification performance from routing errors; it is not a strict upper bound, since another adapter can occasionally be better for a sample.

4 Constructing the Incremental Data Stream

We use four datasets: one for pretraining and the remaining three for the domain-incremental stream. Further details are in § A.

(1) DFDC (Dolhansky et al., 2020) is a large-scale dataset (∼120k videos) with diverse actors and face manipulations from 2020. We use it exclusively for pretraining. **(2) KoDF** (Kwon et al., 2021) contains Korean-subject videos with traditional face-swapping, reenactment, and audio-driven reenactment techniques. **(3) DeepSpeak** (Barrington et al., 2024) covers 500 identities with diverse state-of-the-art manipulations: lip synthesis, avatar and face reenactment for video, and three speech synthesis systems for audio. **(4) MAVOS-DD** (Croitoru et al., 2025) is a recent multilingual dataset with over 250 hours of real and fake videos across multiple languages, generated by seven DeepFake models.

Table 1: **Incremental data stream:** The 23 domains in chronological order and their manipulations, drawn from KoDF, DeepSpeak (DS), or MAVOS-DD (Mavos). Audio: 0 = real audio only; 1 audio manipulation = ElevenLabs; 3 = ElevenLabs, PlayAI, and Speechify.

	Manipulations					Size
$\mathcal{T}$	Video	Family	Audio	Date	Dataset	$\times 10^3$
1	FaceSwap	Face swap	0	2017-12	KoDF	7.9
2	FSGAN	Face swap	0	2019-08	KoDF	5.1
3	FOMM	Avatar	0	2020-02	KoDF	12.8
4	DeepFaceLab	Face swap	0	2020-05	KoDF	8.7
5	ATFHP +	Lip sync	0	2020-02	KoDF	3.5
	Wav2Lip	Lip sync	0	2020-08	KoDF	
6	Wav2Lip	Lip sync	1	2020-08	DS	1.6
7	SimSwap (SS)	Face swap	0	2020-10	DS	0.2
8	HiFiFace	Face swap	0	2021-06	Mavos	9.2
9	VideoRetalking	Lip sync	1	2022-11	DS	1.6
10	FaceFusion	Face swap	0	2023-03	DS	0.8
11	FaceFusion Live	Face swap	0	2023-03	DS	0.8
12	FaceFusion + GAN	Face swap	0	2023-06	DS	0.8
13	InSwapper (IS)	Face swap	0	2023-06	DS + Mavos	0.2 + 9.9
14	IS + CodeFormer	Face swap	0	2023-06	DS	0.2
15	Roop	Face swap	0	2023-07	Mavos	3.2
16	SS + RestoreFormer	Face swap	0	2023-08	DS	0.3
17	Diff2Lip	Lip sync	3	2023-08	DS	1.0
18	EchoMimic	Avatar	0	2024-07	Mavos	3.1
19	LivePortrait	Avatar	0	2024-07	DS + Mavos	1.2 + 2.8
20	HelloMeme	Avatar	0	2024-10	DS	0.4
21	Sonic	Avatar	0	2024-11	Mavos	3.7
22	Memo	Avatar	3	2024-12	DS + Mavos	1.2 + 3.7
23	LatentSync	Lip sync	3	2024-12	DS	1.3

Data Stream Construction. We define 23 domains based on manipulation types, ordered chronologically by release date.[1] The resulting sequence is reported in Table 1, grouped into three families: Face swap (identity replacement), Lip sync (lip and audio synchronization), and Avatar (full-face reenactment). When the same manipulation appears in multiple datasets, we merge their samples into a single domain.

For DeepSpeak, we retain the official test split. For DFDC, MAVOS-DD, and KoDF, we hold out 20% for evaluation (KoDF: a fixed 10k-sample test set; see § A); the remainder is split 80/20 into train and validation. Real samples are balanced to match fake sample counts; in the training set, they are drawn from DFDC and the incremental datasets in a 3:1 ratio (three DFDC samples per incremental-dataset sample); for multi-dataset domains (e.g., InSwapper), real samples are drawn from all contributing sources at this same proportion.

[1] We use the date of the earliest public release (preprint, publication, or code release).

Table 2: **LoRA placement and loss:** CLIP+CLAP, DEFEND-KQ, GT routing, validation set. N_{adp}: number of adapters.

LoRA layers	N_{adp}	AUC (%)↑	Forget↓
Class	16	74.74	**0.14**
Cross+Class	**9**	78.58	2.75
Self+Cross+Class	17	78.46	1.72
Cross+Class (BCE only)	17	**79.58**	1.83

Table 3: **Adapter design:** Comparing different designs, including dynamic expansion on 23 domains. Evaluation performed on the validation set.

	CLIP+CLAP		BRAVEn AV	
Adapter	AUC (%)↑	Forget↓	AUC (%)↑	Forget↓
Single adapter	72.33	11.54	71.21	24.40
One per domain	69.02	-	**93.27**	-
GT-SIM	**82.71**	**2.04**	84.64	4.84
GT-KQ	78.58	2.75	91.98	**1.69**

5 Results and Discussion

Evaluation Metrics We report the area under the ROC curve (AUC) per domain. For the domain-incremental setting, we aggregate over domains using two metrics. The **average AUC** is computed after all domains have been learned, using the adapter selected by the automatic router. The **average forgetting** measures, for each domain, the drop in AUC between the model immediately after that domain was trained (using its dedicated adapter) and the final model after all domains have been seen (using the router-selected adapter). Consequently, forgetting captures both catastrophic interference from later training and routing errors. We report the average forgetting with and without $\mathcal{T}_0$; the average AUC always includes $\mathcal{T}_0$.

Implementation Details We evaluate DEFEND on two frozen backbones: **CLIP+CLAP** (Radford et al., 2021; Wu et al., 2023), with a detection head of self-attention, cross-attention, and an Attentive Pooling Classifier (LoRA on cross-attention and classifier); and **BRAVEn** (Haliassos et al., 2024), with an Attentive Pooling Classifier alone (LoRA on the classifier), used both audio-visually (AV) and video-only (V). We use LoRA rank r=64, λ=0.2, a margin of 0.5, a 2-layer MLP router with temperature T=0.5, τ=0.7 for DEFEND-SIM, and a reuse margin of 0.3. Full details are in § C.

5.1 Training procedure ablations

Unless stated otherwise, all ablations (CLIP+CLAP and BRAVEn AV/V) use 500 training samples per domain, the validation set, and GT routing. All DEFEND-KQ results use a reuse margin of 0.3.

We ablate the CLIP+CLAP head architecture in Table 16 (§ F); differences between variants are within 1.5 AUC points. Removing cross-attention gives the lowest AUC and highest forgetting, so cross-modal interaction matters most; we keep the full head, which forgets the least. On BRAVEn, adding a cross-attention block was unstable across seeds ($83.27\pm7.63\%$ vs. $93.41\pm0.27\%$ GT AUC without it, w/o DFDC), so we use the Attentive Pooling Classifier alone.

Regarding LoRA rank (Table 18), we sweep r with DEFEND-SIM at τ=0.9: AUC is stable from r=8 to r=128 (80.18–82.15%) and degrades beyond (71.47% at r=512). Within this range r=64 has the lowest AUC (80.18%); we choose r=64 as a compromise between per-adapter cost (1.5M parameters) and single-adapter capacity (Table 19). With r=64, we then ablate τ: τ=0.7, the lowest value tested, gives both the highest AUC and the lowest forgetting (82.71%, 2.04), so the optimum may lie below it. The two parameters were tuned sequentially, and τ only affects DEFEND-SIM, while the rank also applies to DEFEND-KQ.

Table 2 ablates the LoRA layers and the margin loss. Applying LoRA only to the classifier gives the lowest forgetting but also the lowest AUC (74.74%). Adding cross-attention (Cross+Class) improves AUC by 3.8 points; also adding self-attention does not help (78.46%) and needs more, larger adapters (17 vs. 9; 2.56M vs. 1.5M each). Removing the margin loss gives a higher AUC (79.58% vs. 78.58%) but 17 instead of 9 adapters (12M more parameters), so we keep it at a 1.0-point AUC cost.

Table 3 compares adapter allocation strategies. A single shared adapter suffers from domain interference (72.33% CLIP+CLAP; 71.21% and 24.40 forgetting on BRAVEn AV). One adapter per domain requires the domain label at inference, so it is not deployable; it also does not improve on CLIP+CLAP (69.02%), as complete isolation prevents knowledge transfer, while on BRAVEn it reaches 93.27%, only 1.3 points above GT-KQ (91.98%). Both dynamic strategies outperform the fixed baselines on CLIP+CLAP. Under GT routing, the SIM-allocated adapters are better than the KQ-allocated ones (82.71% vs. 78.58%): GT routing only measures how well each domain is served

Table 4: **Inference routing evaluation.** Dynamic adapter expansion on 23 domains. Average AUC (%), average forgetting, and routing accuracy (%). Evaluation on the validation set.

	CLIP+CLAP			BRAVEn AV			BRAVEn V		
Routing	AUC↑	Forget↓	Routing ACC↑	AUC↑	Forget↓	Routing ACC↑	AUC↑	Forget↓	Routing ACC↑
GT-SIM	82.71	2.04	100.00	84.64	4.84	100.00	84.19	1.77	100.00
DEFEND-SIM	65.74	23.03	19.17	83.86	5.61	40.65	78.81	7.02	29.07
GT-KQ	78.58	2.75	100.00	91.98	1.69	100.00	90.40	2.65	100.00
DEFEND-KQ	74.93	6.60	86.41	90.45	3.28	82.75	86.43	6.60	75.47

by its adapter, whereas KQ also reuses adapters so that the router can select them, which merges more domains. As Table 4 shows, the SIM adapters cannot be selected reliably without domain labels.

A key hyperparameter is the *reuse margin*: the router creates a new adapter only when the best existing adapter scores below s_{new} by at least this margin. On CLIP+CLAP, a margin of 0.3 reduces the adapters from 12 to 9 with no loss in routed AUC (Table 17). Fewer adapters are only an advantage when they cost little accuracy: without the compatibility score s_{new} of the new adapter, BRAVEn (full training set) collapses to 5 adapters and the routed AUC drops from 89.68% to 79.34% (Table 22).

5.2 Inference procedure ablations

Table 4 compares GT and automatic routing on all backbones (CLIP+CLAP, BRAVEn AV, BRAVEn V). DEFEND-SIM achieves only 19.17% routing accuracy on CLIP+CLAP, which drops the AUC from 82.71% to 65.74% and raises the forgetting from 2.04 to 23.03. Since GT-SIM cannot be deployed without domain labels at inference, static similarity-based routing is not a viable choice. DEFEND-KQ reaches 86.41% routing accuracy and remains much closer to its GT counterpart (74.93% vs. 78.58% AUC, 6.60 vs. 2.75 forgetting), where the extra forgetting reflects the routing errors that automatic routing incurs. On BRAVEn AV, its AUC is only 1.5 points below GT-KQ (90.45% vs. 91.98%). This is consistent with misrouted samples reaching adapters of similar manipulations: domains routed together share an adapter well (Table 9), another adapter often scores higher than the assigned one (Table 11), and random routing is far worse (Table 14). Because its similarity routing is too unreliable, we drop DEFEND-SIM and use DEFEND-KQ in the rest of the paper. GT routing is also not a strict upper bound: for 7–10 of the 24 domains, an adapter other than the assigned one gives a higher AUC, and choosing the best adapter per domain would give 93.28% instead of 92.38% on the final BRAVEn AV model (Table 11). Finally, the routing accuracy on early domains is stable: on the first five domains it is 80.5% after $\mathcal{T}_5$ and 79.1% after $\mathcal{T}_{23}$; the lower final routing accuracy comes from later domains that resemble the earlier ones (Figure 3).

5.3 Comparing DEFEND against baselines

Results are on the full test set of $\mathcal{T}_0$–$\mathcal{T}_{23}$ using full per-domain training data.

Joint Training On CLIP+CLAP, joint LoRA training fails to learn ($60.67 \pm 0.29\%$, below the single adapter), with AUC below 50% on $\mathcal{T}_1$, $\mathcal{T}_3$ and $\mathcal{T}_4$ (Table 20), which points to a training difficulty rather than a property of the task: unlike BRAVEn, pretrained for lip reading and speech recognition, CLIP and CLAP are pretrained on generic image–text and audio–text data, so fitting all manipulations at once is harder; it is not an upper bound here. On BRAVEn, joint training reaches the highest AUC ($94.93 \pm 1.39\%$ AV, $93.83 \pm 0.86\%$ V); as a non-incremental method, it serves as an upper bound on this backbone only.

Single Adapter A single LoRA adapter trained sequentially achieves 74.28% AUC and 14.81 forgetting on CLIP+CLAP. On BRAVEn AV the degradation is much more severe (66.63%, 29.00 forgetting), as each domain overwrites the sole classifier; the video-only model is more stable (84.32%, 8.88 forgetting).

One Adapter per Domain One dedicated adapter per domain achieves 87.50% AUC on CLIP+CLAP and 93.19% (AV) and 92.91% (V) on BRAVEn, but requires 23 adapters (34.5M parameters on CLIP+CLAP, 24.2M on BRAVEn) and ground-truth domain labels at inference, making it unsuitable for deployment. Pairing it with the KQ router would remove the need for labels, but routing among

Table 5: **Comparison of DEFEND to baselines on 23 domains.** Average AUC and average forgetting are reported on the test set; mean ± std over 3 seeds where given. One Adapter per Domain uses the ground-truth domain at inference. $N_{\text{adp}} = 23$ for One Adapter per Domain; for DEFEND-KQ, $N_{\text{adp}} = 8.0$ (CLIP+CLAP), 11.3 (BRAVEn AV) and 11.3 (BRAVEn V). (V) denotes the video-only model, pretrained on DFDC with video features only. The dashed line separates standard continual learning baselines from the other comparison methods.

			Train. params (M)			Avg forget↓	
Method	Adaptation	Incr.	Per update	Total	Avg AUC (%)↑	All	w/o DFDC
CLIP+CLAP							
Joint training	LoRA	no	6.0	6.0	60.67 ±0.29	-	-
Single adapter	LoRA	yes	1.5	1.5	74.28 ±7.31	14.81±5.23	16.03±5.89
One adapter per Domain	LoRA	yes	1.5	34.5	87.50±0.18	-	-
Full finetuning	finetuning	yes	8.67	8.67	74.36±1.94	14.36±1.32	13.89±1.25
EWC	finetuning	yes	8.67	8.67	75.87 ±0.86	11.71 ±0.62	11.80 ±0.77
KD+Margin Loss	finetuning	yes	8.67	8.67	77.10 ±0.63	9.47 ±0.58	9.31 ±0.51
DER++	finetuning	yes	8.67	8.67	77.91 ±2.14	9.11 ±1.97	8.05 ±2.02
DualPrompt	prompting	yes	0.4	0.4	67.32 ±1.06	5.38 ±0.81	5.42 ±0.78
GT-KQ	LoRA	yes	1.5	12	82.81 ±2.76	4.69 ±1.58	4.90 ±1.65
DEFEND-KQ	LoRA	yes	1.5	12	77.96 ±2.31	9.65 ±1.45	9.31 ±1.53
BRAVEn AV							
Joint training	LoRA	no	4.2	4.2	94.93 ±1.39	-	-
Single adapter	LoRA	yes	1.05	1.05	66.63 ±4.41	29.00 ±4.72	29.34 ±4.64
One adapter per Domain	LoRA	yes	1.05	24.2	93.19 ±0.04	-	-
Full finetuning	finetuning	yes	11.55	11.55	57.91 ±2.21	27.80 ±2.76	27.79 ±2.83
EWC	finetuning	yes	11.55	11.55	81.23 ±0.71	10.65 ±0.78	10.53 ±0.80
KD+Margin Loss	finetuning	yes	11.55	11.55	55.27 ±3.22	29.80 ±3.95	29.79 ±4.09
DER++	finetuning	yes	11.55	11.55	78.51 ±2.51	8.50 ±2.51	7.81 ±2.62
DualPrompt	prompting	yes	0.27	0.27	78.32 ±2.01	9.87 ±2.05	9.91 ±2.05
GT-KQ	LoRA	yes	1.05	11.9	92.38 ±0.73	1.42 ±0.92	1.49 ±0.96
DEFEND-KQ	LoRA	yes	1.05	11.9	89.68 ±0.90	4.24 ±1.05	3.81 ±1.02
BRAVEn V							
Joint training	LoRA	no	4.2	4.2	93.83 ±0.86	-	-
Single adapter	LoRA	yes	1.05	1.05	84.32 ±0.48	8.88 ±0.33	8.76 ±0.37
One adapter per Domain	LoRA	yes	1.05	24.2	92.91 ±0.07	-	-
Full finetuning	finetuning	yes	11.55	11.55	71.82 ±3.91	19.31 ±3.49	19.37 ±3.59
EWC	finetuning	yes	11.55	11.55	85.44 ±0.93	5.18 ±1.00	5.26 ±1.03
KD+Margin Loss	finetuning	yes	11.55	11.55	78.21 ±1.72	14.33 ±2.11	14.19 ±1.98
DER++	finetuning	yes	11.55	11.55	78.58 ±2.28	2.18 ±1.06	1.62 ±1.02
DualPrompt	prompting	yes	0.27	0.27	80.68 ±0.12	2.26 ±0.25	2.16 ±0.29
GT-KQ	LoRA	yes	1.05	11.9	91.79 ±1.00	2.04 ±0.99	2.13 ±1.04
DEFEND-KQ	LoRA	yes	1.05	11.9	87.95 ±1.15	5.79 ±1.17	5.60 ±1.33

more adapters is harder: when the router allocates 21–23 adapters, its routing accuracy drops to 28–47% and the routed AUC to 82.7–88.0% (Table 15).

Incremental Full Finetuning Finetuning the cross-attention and classifier layers achieves 74.36% AUC and 14.36 forgetting on CLIP+CLAP (57.91% AV, 71.82% V on BRAVEn), with substantial degradation on prior domains.

Continual Learning Baselines On CLIP+CLAP, EWC (Kirkpatrick et al., 2017), KD+Margin (Hinton et al., 2015), DER++ (Buzzega et al., 2020), and DualPrompt (Wang et al., 2022b) all outperform joint training, and all but DualPrompt also outperform the single adapter. DER++ reaches the highest AUC among them (77.91±2.14%); DualPrompt has the lowest forgetting (5.38) but also the lowest AUC (67.32±1.06%). On BRAVEn AV, KD+Margin collapses to 55.27% AUC with 29.80 forgetting; EWC is the strongest continual learning baseline (81.23% AV, 85.44% V), but with higher forgetting than DER++ and DualPrompt (10.65 AV, 5.18 V).

DEFEND On CLIP+CLAP, DEFEND-KQ achieves 77.96±2.31% routed AUC (routing accuracy 74.43±5.66%, forgetting 9.65±1.45) with 8.0 adapters on average, on par with the best continual learning baselines (DER++, KD+Margin). As a reference that uses ground-truth domain labels, unlike the baselines, GT-KQ reaches 82.81±2.76% (forgetting 4.69±1.58). On BRAVEn, GT-KQ reaches 92.38±0.73% AUC (forgetting 1.42±0.92), 0.8 points below One Adapter per Domain (93.19±0.04%) but with only 11.3 adapters; automatic routing achieves 89.68±0.90% (forgetting 4.24±1.05), above all incremental baselines. The contrast with the baselines is strongest on this backbone: the Single Adapter, Full Finetuning and KD+Margin reach only 55–67% AUC with 28–30 points of forgetting, since each new domain overwrites the attentive classifier, which in BRAVEn is the only module of the head. The video-only model reaches 91.79±1.00% (GT) and 87.95±1.15%

(routed), only 0.6–1.7 points below AV, while fixed-capacity baselines are far more stable without audio (Single Adapter: 84.32% V vs. 66.63% AV), indicating that sequential updates of a shared audio-visual head are especially prone to domain interference, which DEFEND avoids through adapter isolation. Overall, the audio stream gives a modest gain. It is somewhat larger on the domains with manipulated audio (0.84 GT, 3.12 routed) than on those with real audio (0.58 and 1.68, Table 13), but three seeds are too few to establish this difference.

Domain grouping DEFEND-KQ organizes domains by manipulation similarity in the backbone representation rather than by dataset boundaries (Table 8). The groups are largely stable across seeds and mostly follow the manipulation families. On both backbones, the diffusion-based lip-sync methods Diff2Lip and LatentSync share an adapter in all runs, and InSwapper and Roop are consistently grouped together, often joined by HiFiFace and LivePortrait. The two backbones differ in how they separate lip-sync from face swapping. With BRAVEn, whose features come from a lip-reading model, Wav2Lip and Retalking always share an adapter, while SimSwap and the FaceFusion variants form a second group. With CLIP+CLAP, Wav2Lip and Retalking are merged with SimSwap and FaceFusion into one larger adapter, reflecting what each backbone encodes. To verify these groups, we hold out each domain of the two largest BRAVEn groups and train an adapter either on the remaining domains of its group or on the same number of domains from other groups: training on its own group gives a higher AUC on 11 of the 13 held-out domains (Table 9), confirming that the domains routed together benefit from sharing an adapter.

Robustness analyses Sending each test sample to a random adapter drops AUC by 20–24 points below GT routing and causes $>$20 points of forgetting, confirming that the adapters are specialized and that the router is essential (Table 14). The key-query router recovers most of this gap: DEFEND-KQ stays within 2.7–4.9 points of GT routing on all three settings, although it picks the adapter of the correct domain for only 74–78% of the fake samples. Capping the router's stored embeddings to $B{=}500$ per domain reduces the router memory from 217 to 39.7 MiB at a cost of 1.7 routed AUC points on BRAVEn AV (Table 15). With less memory the router reuses adapters less often (21–23 instead of 15.7), and most of the drop comes from these allocation decisions rather than from the final router: capping the memory only for the final router costs less (88.27% with 500 embeddings per domain). The memory also differs in nature from that of DER++: DER++ replays 500 model inputs, i.e. raw clips on CLIP+CLAP (4.6 GiB) and cached features on BRAVEn (62.5 MiB, 128 KiB per sample), while the router stores one embedding per clip (4 KiB on BRAVEn), so $B{=}500$ covers $20\times$ more clips than DER++ in less memory (39.7 MiB).

Overall, dynamic adapter expansion with learned key-query routing achieves a better balance between adaptation and knowledge retention than existing continual learning approaches, while allocating far fewer adapters than domains (8-11 for 23 domains).

6 Conclusion

We introduced a new domain-incremental multimodal DeepFake detection task that models manipulation techniques continuously emerging over time. To support this setting, we defined a chronological protocol spanning seven years (2017–2024) and 23 manipulation domains from KoDF, DeepSpeak, and MAVOS-DD. We proposed DEFEND, a framework with parameter-efficient updates that balances specialization and knowledge reuse through dynamic LoRA adapter expansion and learned key-query routing, deciding for each domain whether to reuse an existing adapter or create a new one. Experimentally, DEFEND-KQ is on par with the best continual learning baselines on CLIP+CLAP (77.96% AUC) and outperforms all incremental baselines on BRAVEn (89.68% AV, 87.95% V), while allocating only 8–11 adapters for 23 domains. Several limitations remain. DEFEND assumes that domain boundaries are known during training and that each step contains a single coherent domain, whereas a deployed detector may receive unlabeled, mixed-domain data; moreover, these boundaries follow human-defined manipulation types, while our routing results suggest a coarser structure. The router stores one embedding per clip of every past domain, so its memory grows with the stream, and routing errors can accumulate as the number of domains grows. We evaluate a single stream order, several domains contain few samples, and real samples are drawn from DFDC and the incremental datasets in a fixed 3:1 ratio, which may act as a dataset shortcut. Future work will investigate more scalable routing and adaptive adapter consolidation for long-term continual multimodal learning.

AI USE STATEMENT

In this work, we used generative AI tools for the following tasks.

Writing. It was used to edit and refine parts of the results section and of the appendix, and to improve the readability of the text.

Code. Generative AI tools were used to debug our code. The core method (the DEFEND framework, the key-query router and the training pipeline) was designed and implemented by the authors. We have reviewed all AI-assisted work: e.g., the experiment designs were discussed and approved by the authors before running them; the AI-written scripts were inspected and their outputs verified against our existing evaluation code, which reproduced the results exactly on previously evaluated runs; all AI-drafted text was reviewed and edited by the authors.

We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work aims to improve the detection of audio-visual DeepFakes, which can be used for disinformation, fraud and non-consensual content. We do not develop or release any generation method, and all manipulated videos used in this work come from existing public benchmarks.

Data. We only use publicly available datasets (DFDC, KoDF, DeepSpeak, and MAVOS-DD) and follow their licenses and terms of use; datasets that require an access agreement were obtained through it. We do not collect new data and do not redistribute any video: the release will only contain our code, the domain splits (lists of file identifiers) and the trained adapters. The datasets contain faces and voices of real people, so we do not use them for any purpose other than detection research, and we do not try to identify any individual.

Dual use. Publishing detection methods and the domains on which they fail can help attackers to design manipulations that evade detection. We believe the benefit for defenders outweighs this risk, since our results report aggregate detection performance on manipulations that are already public, and the incremental setting is designed to let detectors be updated quickly when new manipulations appear.

Bias and limitations. The domains are drawn from four datasets that differ in language, recording conditions and demographics, and some domains have few samples. Detection and routing performance differ across languages (§ E), and we did not evaluate performance across demographic groups, since the datasets do not provide consistent annotations. The method relies on face crops and on labeled data for every new manipulation. A detector like ours can make errors, and a wrong "fake" prediction can harm the person in the video. It should therefore be used as a tool to support human verification, and not as the only evidence to decide whether a video is authentic.

REPRODUCIBILITY STATEMENT

We describe the method in § 3, including the key-query router (Algorithm 1) and the training objective. The construction of the 23-domain stream, the datasets and the chronological order of the domains are given in § 4 and § A; the real and fake samples used for training, validation and testing are fixed by split files, and all methods are evaluated on the same test split. § C lists the backbones, the architecture of the detection head, all hyperparameters (LoRA rank, loss weights, margin, router dimensions, temperature, reuse margin, optimizer, learning rates, epochs and batch sizes), the pretraining on DFDC, and the settings of the baselines, together with the training cost, memory and wall-clock time. The evaluation metrics (average AUC, forgetting and routing accuracy) are defined in § 5. The main results are averaged over different seeds and reported as mean $\pm$ std.

REFERENCES

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *CVPR*, 2017.

Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, and Gang Hua. Towards open-set identity preserving face synthesis. In *CVPR*, 2018.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *arXiv preprint arXiv:2404.08471*, 2024.

Sarah Barrington, Matyas Bohacek, and Hany Farid. The DeepSpeak dataset. *arXiv preprint arXiv:2408.05366*, 2024.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: A strong, simple baseline. In *ICML*, 2020.

Renwang Chen, Xuanhong Chen, Bingbing Ni, and Yanhao Ge. Simswap: An efficient framework for high fidelity face swapping. In *Proceedings of the 28th ACM international conference on multimedia*, pp. 2003–2011, 2020.

Xiaohuan Chen, Wenhuan Lu, Ruiteng Zhang, Junhai Xu, Xugang Lu, Lin Zhang, and Jianguo Wei. Continual unsupervised domain adaptation for audio deepfake detection. In *ICASSP*, 2025a.

Xiaokang Chen, Mingyu Ding, Xiaodi Wang, Ying Xin, Shentong Mo, Yunhao Wang, Shumin Han, Ping Luo, Gang Zeng, and Jingdong Wang. Context autoencoder for self-supervised representation learning. *Int. J. Comput. Vis.*, 132(1):208–223, 2024.

Zhiyuan Chen, Jiajiong Cao, Zhiquan Chen, Yuming Li, and Chenguang Ma. Echomimic: Lifelike audio-driven portrait animations through editable landmark conditions. In *AAAI*, 2025b.

Jikang Cheng, Zhiyuan Yan, Ying Zhang, Li Hao, Jiaxin Ai, Qin Zou, Chen Li, and Zhongyuan Wang. Stacking brick by brick: Aligned feature isolation for incremental face forgery detection. In *CVPR*, 2025.

Kun Cheng, Xiaodong Cun, Yong Zhang, Menghan Xia, Fei Yin, Mingrui Zhu, Xuan Wang, Jue Wang, and Nannan Wang. VideoReTalking: Audio-based lip synchronization for talking head video editing in the wild. In *SIGGRAPH*, 2022.

Florinel-Alin Croitoru, Vlad Hondru, Marius Popescu, Radu Tudor Ionescu, Fahad Shahbaz Khan, and Mubarak Shah. Mavos-dd: Multilingual audio-video open-set deepfake detection benchmark. *arXiv preprint arXiv:2505.11109*, 2025.

Brian Dolhansky, Joanna Bitton, Ben Pflaum, Jikuo Lu, Russ Howes, Menglin Wang, and Cristian Canton Ferrer. The deepfake detection challenge (DFDC) dataset. *arXiv preprint arXiv:2006.07397*, 2020.

Yuxuan Du, Zhendong Wang, Yuhao Luo, Caiyong Piao, Zhiyuan Yan, Hao Li, and Li Yuan. CAD: A general multimodal framework for video deepfake detection via cross-modal alignment and distillation. *arXiv preprint arXiv:2505.15233*, 2025.

ElevenLabs. Elevenlabs: Ai voice generation platform. `https://www.elevenlabs.io`, 2023. Accessed: 2025-08-05.

Falih Gozi Febrinanto, Kristen Moore, Chandra Thapa, Jiangang Ma, Vidya Saikrishna, and Feng Xia. Rehearsal with auxiliary-informed sampling for audio deepfake detection. In *Interspeech*, 2025.

Chao Feng, Ziyang Chen, and Andrew Owens. Self-supervised video forensics by audio-visual anomaly detection. In *CVPR*, 2023.

Federico Fontana, Anxhelo Diko, Romeo Lanzino, Marco Raoul Marini, Bachir Kaddar, Gian Luca Foresti, and Luigi Cinque. Revisiting deepfake detection: Chronological continual learning and the limits of generalization. *arXiv preprint arXiv:2509.07993*, 2025.

Jianzhu Guo, Dingyun Zhang, Xiaoqiang Liu, Zhizhou Zhong, Yuan Zhang, Pengfei Wan, and Di Zhang. Liveportrait: Efficient portrait animation with stitching and retargeting control. *arXiv preprint arXiv:2407.03168*, 2024.

Alexandros Haliassos, Rodrigo Mira, Stavros Petridis, and Maja Pantic. Leveraging real talking faces via self-supervision for robust forgery detection. In *CVPR*, pp. 14930–14942, 2022.

Alexandros Haliassos, Andreas Zinonos, Rodrigo Mira, Stavros Petridis, and Maja Pantic. Braven: Improving self-supervised pre-training for visual and auditory speech recognition, 2024. URL https://arxiv.org/abs/2404.02098.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

Dong Huang and Fernando De La Torre. Facial action transfer with personalized bilinear regression. In *ECCV*, 2012.

Xiaozhong Ji, Xiaobin Hu, Zhihong Xu, Junwei Zhu, Chuming Lin, Qingdong He, Jiangning Zhang, Donghao Luo, Yi Chen, Qin Lin, et al. Sonic: Shifting focus to global audio perception in portrait animation. In *CVPR*, 2025.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

Sohail Ahmed Khan and Duc-Tien Dang-Nguyen. CLIPping the deception: Adapting vision-language models for universal deepfake detection. In *ICMR*, 2024.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *PNAS*, 114(13):3521–3526, 2017.

Marcel Klemt, Carlotta Segna, and Anna Rohrbach. Deepfake doctor: Diagnosing and treating audio-video fake detection. *arXiv preprint arXiv:2506.05851*, 2025.

Christos Koutlis and Symeon Papadopoulos. Leveraging representations from intermediate encoder-blocks for synthetic image detection. *arXiv preprint arXiv:2402.19091*, 2024.

Christos Koutlis and Symeon Papadopoulos. AuViRe: Audio-visual speech representation reconstruction for deepfake temporal localization. In *WACV*, 2025.

Patrick Kwon, Jaeseong You, Gyuhyeon Nam, Sungwoo Park, and Gyeongsu Chae. KoDF: A large-scale Korean deepfake detection dataset. In *ICCV*, 2021.

Francesco Laiti, Benedetta Liberatori, Thomas De Min, and Elisa Ricci. Conditioned prompt-optimization for continual deepfake detection. In *ICPR*, 2024.

Chunyu Li, Chao Zhang, Weikai Xu, Jinghui Xie, Weiguo Feng, Bingyue Peng, and Weiwei Xing. Latentsync: Audio conditioned latent diffusion models for lip sync. *arXiv e-prints*, pp. arXiv–2412, 2024.

Chuqiao Li, Zhiwu Huang, Danda Pani Paudel, Yabin Wang, Mohamad Shahbazi, Xiaopeng Hong, and Luc Van Gool. A continual deepfake detection benchmark: Dataset, methods, and essentials. In *WACV*, 2023.

Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *ICML*, 2019.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(12):2935–2947, 2017.

Yachao Liang, Min Yu, Gang Li, Jianguo Jiang, Boquan Li, Feng Yu, Ning Zhang, Xiang Meng, and Weiqing Huang. SpeechForensics: Audio-visual speech representation learning for face forgery detection. In *NeurIPS*, 2024.

Baoping Liu, Bo Liu, Ming Ding, and Tianqing Zhu. ForgeFinder: Perceptive multimodal deepfake detection via multi-grained forgery localization. *ACM Transactions on Multimedia Computing, Communications and Applications*, 22(1):1–24, 2026.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Francesco Marra, Cristiano Saltori, Giulia Boato, and Luisa Verdoliva. Incremental learning for the detection and classification of gan-generated images. In *WIFS*, 2019.

Soumik Mukhopadhyay, Saksham Suri, Ravi Teja Gadde, and Abhinav Shrivastava. Diff2lip: Audio conditioned diffusion models for lip-synchronization. In *WACV*, 2024.

Yuval Nirkin, Yosi Keller, and Tal Hassner. FSGAN: Subject agnostic face swapping and reenactment. In *ICCV*, 2019.

Utkarsh Ojha, Yuheng Li, and Yong Jae Lee. Towards universal fake image detectors that generalize across generative models. In *CVPR*, 2023.

Trevine Oorloff, Surya Koppisetti, Nicolò Bonettini, Divyaraj Solanki, Ben Colman, Yaser Yacoob, Ali Shahriyari, and Gaurav Bharaj. AVFF: Audio-visual feature fusion for video deepfake detection. In *CVPR*, 2024.

Kun Pan, Yifang Yin, Yao Wei, Feng Lin, Zhongjie Ba, Zhenguang Liu, Zhibo Wang, Lorenzo Cavallaro, and Kui Ren. DFIL: Deepfake incremental learning by exploiting domain-invariant forgery clues. In *ACM MM*, 2023.

Ivan Perov, Daiheng Gao, Nikolay Chervoniy, Kunlin Liu, Sugasa Marangonda, Chris Umé, Carl Shift Facenheim, Luis RP, Jian Jiang, Sheng Zhang, et al. Deepfacelab: Integrated, flexible and extensible face-swapping framework. *arXiv preprint arXiv:2005.05535*, 2020.

PlayAI. Instant voice cloning (playai api). `https://docs.play.ai/reference/api-create-instant-voice-clone`, 2025. Accessed: 2025-08-05.

Adam Polyak, Lior Wolf, and Yaniv Taigman. Tts skins: Speaker conversion via asr, 2020. URL `https://arxiv.org/abs/1904.08983`.

K. R. Prajwal, Rudrabha Mukhopadhyay, Vinay P. Namboodiri, and C. V. Jawahar. A lip sync expert is all you need for speech to lip generation in the wild. In *ACM MM*, 2020.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. iCaRL: Incremental classifier and representation learning. In *CVPR*, 2017.

Tal Reiss, Bar Cavia, and Yedid Hoshen. Detecting deepfakes without seeing any. *arXiv preprint arXiv:2311.01458*, 2023.

Henry Ruhs. Facefusion. `https://github.com/facefusion/facefusion`, 2024.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

s0md3v. Roop: A face swapping tool. `https://github.com/s0md3v/roop`, 2023. Accessed: 2025-08-05.

Davide Salvi, Viola Negroni, Luca Bondi, Paolo Bestagini, and Stefano Tubaro. Freeze and learn: Continual learning with selective freezing for speech deepfake detection. In *ICASSP*, 2025.

Hao Shen, Jikang Cheng, Renye Yan, Zhongyuan Wang, Wei Peng, and Baojin Huang. When generative replay meets evolving deepfakes: Domain-aware relative weighting for incremental face forgery detection. *arXiv preprint arXiv:2511.18436*, 2025.

Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. *Advances in neural information processing systems*, 32, 2019.

Stefan Smeu, Dragos-Alexandru Boldisor, Dan Oneata, and Elisabeta Oneata. Circumventing shortcuts in audio-visual deepfake detection datasets with unsupervised learning. In *CVPR*, 2025.

Speechify. Speechify v1 voices api. `https://docs.sws.speechify.com/`, 2025. Speechify Text-to-Speech API v1 voices; accessed 2025-08-05.

Sam Stockwell. From deepfake scams to poisoned chatbots: AI and election security in 2025. *CETaS Expert Analysis*, 2025. URL `https://cetas.turing.ac.uk/publications/deepfake-scams-poisoned-chatbots`.

Francesco Tassone, Luca Maiano, and Irene Amerini. Continuous fake media detection: Adapting deepfake detectors to new generative techniques. *Comput. Vis. Image Underst.*, 249:104143, 2024.

Jiahe Tian, Cai Yu, Xi Wang, Peng Chen, Zihao Xiao, Jizhong Han, and Yesheng Chai. Dynamic mixed-prototype model for incremental deepfake detection. In *ACM MM*, 2024.

Gido M Van de Ven, Tinne Tuytelaars, and Andreas S Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 2022.

Haofan Wang. INSwapper: Face swapping model based on insightface. `https://github.com/haofanwang/inswapper`, 2023. Accessed: 2025-08-05.

Huiyi Wang, Haodong Lu, Lina Yao, and Dong Gong. Self-expansion of pre-trained models with mixture of adapters for continual learning. In *CVPR*, 2025.

Yabin Wang, Zhiwu Huang, and Xiaopeng Hong. S-prompts learning with pre-trained transformers: An Occam's razor for domain incremental learning. In *NeurIPS*, 2022a.

Yuhan Wang, Xu Chen, Junwei Zhu, Wenqing Chu, Ying Tai, Chengjie Wang, Jilin Li, Yongjian Wu, Feiyue Huang, and Rongrong Ji. Hififace: 3d shape and semantic prior guided high fidelity face swapping. *arXiv preprint arXiv:2106.09965*, 2021.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. DualPrompt: Complementary prompting for rehearsal-free continual learning. In *ECCV*, 2022b.

Xuecheng Wu, Heli Sun, Danlei Huang, Xinyi Yin, Yifan Wang, Hao Wang, Jia Zhang, Fei Wang, Peihao Guo, Suyu Xing, et al. HOLA: Enhancing audio-visual deepfake detection via hierarchical contextual aggregations and efficient pre-training. In *ACM MM*, 2025.

Yusong Wu, Ke Chen, Tianyu Zhang, Yuchen Hui, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation. In *ICASSP*, 2023.

Wuti Xiong, Guoying Zhao, and Xiaobai Li. Exemplar-free incremental deepfake detection. In *ECAI*, 2024.

Ran Yi, Zipeng Ye, Zhiyao Sun, Juyong Zhang, Guoxin Zhang, Pengfei Wan, Hujun Bao, and Yong-Jin Liu. Predicting personalized head movement from short video and speech signal. *IEEE Transactions on Multimedia*, pp. 1–13, 2022. doi: 10.1109/TMM.2022.3207606.

Egor Zakharov, Aliaksandra Shysheya, Egor Burkov, and Victor Lempitsky. Few-shot adversarial learning of realistic neural talking head models. In *ICCV*, October 2019.

Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Sign. Process. Letters*, 23(10):1499–1503, 2016.

Shengkai Zhang, Nianhong Jiao, Tian Li, Chaojie Yang, Chenhui Xue, Boya Niu, and Jun Gao. Hellomeme: Integrating spatial knitting attentions to embed high-level and fidelity-rich conditions in diffusion models. *arXiv preprint arXiv:2410.22901*, 2024.

Xiaohui Zhang, Jiangyan Yi, Jianhua Tao, Chenglong Wang, and Chu Yuan Zhang. Do you remember? Overcoming catastrophic forgetting for fake audio detection. In *ICML*, 2023.

Xueyi Zhang, Peiyin Zhu, Chengwei Zhang, Zhiyuan Yan, Jikang Cheng, Mingrui Lao, Siqi Cai, and Yanming Guo. Generalization-preserved learning: Closing the backdoor to catastrophic forgetting in continual deepfake detection. In *ICCV*, 2025.

Yanzhe Zhang, Xuezhi Wang, and Diyi Yang. Continual sequence generation with adaptive compositional modules. In *ACL*, 2022.

Longtao Zheng, Yifan Zhang, Hanzhong Guo, Jiachun Pan, Zhenxiong Tan, Jiahao Lu, Chuanxin Tang, Bo An, and Shuicheng Yan. Memo: Memory-guided diffusion for expressive talking video generation. *arXiv preprint arXiv:2412.04448*, 2024.

Da-Wei Zhou, Hai-Long Sun, Han-Jia Ye, and De-Chuan Zhan. Expandable subspace ensemble for pre-trained model-based class-incremental learning. In *CVPR*, 2024.

Shangchen Zhou, Kelvin C. K. Chan, Chongyi Li, and Chen Change Loy. Towards robust blind face restoration with codebook lookup transformer. In *NeurIPS*, 2022.

A DATASETS

As introduced in § 4 we employ four different datasets: DFDC (Dolhansky et al., 2020), DeepSpeak (Barrington et al., 2024), KoDF (Kwon et al., 2021), and MAVOS-DD (Croitoru et al., 2025). Below, we report the details about the datasets.

Table 6: Summary of dataset statistics and modalities used in this work.

Dataset	Identities	Video Manipulations	Audio Manipulations	Use Case
DFDC	3,426	5 (+ refinement)	1 (TTS Skins)	Pretraining
DeepSpeak	500	14	3 (ElevenLabs, Speechify, PlayAI)	Incremental
KoDF	403	6	0	Incremental
MAVOS-DD	*N/A*	7	0	Incremental

A.1 DEEPFAKE DETECTION CHALLENGE (DFDC)

The DFDC dataset (Dolhansky et al., 2020) serves as our primary pretraining dataset due to its scale and demographic diversity. It consists of approximately $128,000$ videos, featuring high variability in lighting, backgrounds, and compression levels.

- **Manipulations:** The video fakes are generated with five methods: the DeepFake Autoencoder (DFAE, in two resolutions), *MM/NN Face Swap* (Huang & De La Torre, 2012), *NTH* (Zakharov et al., 2019), *FSGAN* (Nirkin et al., 2019), and a *StyleGAN*-based (Karras et al., 2019) face swap; a random subset is additionally post-processed with a sharpening filter (refinement).
- **Audio:** Manipulation is limited to a single audio technique: *TTS Skins Voice* (Polyak et al., 2020) (voice conversion). Consequently, most fakes retain the original audio or focus solely on visual artifacts.
- **Partitioning:** Following common practice where the private test set is inaccessible, we split the available data into 70% training, 10% validation, and 20% test sets. Due to the lack of per-manipulation labels, we utilize DFDC exclusively for learning general-purpose audio-visual features.

A.2 DEEPSPEAK

The DeepSpeak dataset (Barrington et al., 2024) is designed specifically for multimodal DeepFake detection and provides approximately $30,000$ samples from 500 unique identities. It covers face-swap, lip-synthesis and avatar-based manipulations, paired with several speech synthesis engines.

- **Manipulations:** Face swaps (*FaceFusion* (Ruhs, 2024), *FaceFusion+GAN* (Zhou et al., 2022), *FaceFusion Live*, *SimSwap* (Chen et al., 2020), *InSwapper* (Wang, 2023), and their restored variants *InSwapper+CodeFormer* (Zhou et al., 2022) and *SimSwap+RestoreFormer*), lip-synthesis (*Wav2Lip* (Prajwal et al., 2020), *VideoRetalking* (Cheng et al., 2022), *Diff2Lip* (Mukhopadhyay et al., 2024), *LatentSync* (Li et al., 2024)), and avatar-based reenactment (*LivePortrait* (Guo et al., 2024), *HelloMeme* (Zhang et al., 2024), *Memo* (Zheng et al., 2024)).
- **Audio:** Synthetic audio is generated with the *ElevenLabs* (ElevenLabs, 2023), *Speechify* (Speechify, 2025) and *PlayAI* (PlayAI, 2025) synthesis engines. The lip-synthesis and *Memo* manipulations are paired with these fake audio tracks, while the face swaps keep the original audio.
- **Partitioning:** We adopt the provided test splits and derive our validation sets from the training data using an $80/20\%$ split.

A.3 KOREAN DEEPFAKE (KODF)

KoDF (Kwon et al., 2021) is utilized to assess the model's robustness against demographic (Korean ethnicity) and linguistic shifts.

- **Composition:** The dataset contains $\approx 238,000$ videos from 403 identities, utilizing six synthesis pipelines: *FaceSwap* (Bao et al., 2018), *DeepFaceLab* (Perov et al., 2020), *FS-GAN* (Nirkin et al., 2019), *First Order Motion Model* (FOMM) (Siarohin et al., 2019), *Audio-driven Talking Face Head Pose* (ATFHP (Yi et al., 2022)), and *Wav2Lip* (Prajwal et al., 2020) (the last two are grouped into the same "class" in the dataset).
- **Selection:** We select a diverse subset of $50,000$ samples. To prevent leakage, we allocate $10,000$ samples for testing and split the remaining samples into training (80%) and validation (20%) sets.

A.4 MAVOS-DD

MAVOS-DD (Croitoru et al., 2025) is a recent multilingual audio-visual DeepFake dataset designed to capture variability across both manipulation techniques and languages.

- **Composition:** The dataset comprises over 250 hours of real ($\sim$25k samples) and manipulated ($\sim$35k samples) videos generated using seven distinct DeepFake models across multiple languages. We consider English, German, Hindi, Mandarin, Romanian, Russian, and Spanish.

  The manipulation techniques include *InSwapper* (Wang, 2023), *HiFiFace* (Wang et al., 2021), and *Roop* (s0md3v, 2023) for face swapping, as well as *EchoMimic* (Chen et al., 2025b), *LivePortrait* (Guo et al., 2024), *Memo* (Zheng et al., 2024), and *Sonic* (Ji et al., 2025) for avatar-based reenactment.
- **Partitioning:** We split the data into 70% training, 10% validation, and 20% test sets.

B METHODOLOGY

Algorithm 2 provides pseudocode for DEFEND-SIM, our cosine-similarity-based routing that selects adapters based on their centroids. The description is provided in § 3.2.

Algorithm 2 Adapter selection via similarity-based routing

Input: $\mathcal{M}$ set of adapters, $\mathcal{S}_k$ domain data , τ threshold

Function UPDATEMODEL($\mathcal{M}$, $\mathcal{S}_k$, τ)
  $(i, s) \leftarrow$ FINDMOSTSIMILARADAPTER($\mathcal{M}, \mathcal{S}_k$)
  if $s > \tau$ **and** $|\mathcal{M}| > 0$ **then**
    $\mathcal{M}[i] \leftarrow$ TRAINADAPTER($\mathcal{M}[i], \mathcal{S}_k$)
  else
    $\mathcal{M}$.append(TRAINNEWADAPTER($\mathcal{S}_k$))
    $i = -1$
  end if
  $\mathcal{M}[i]$.centroid = AVG(GETEMBEDDINGS($\mathcal{S}_k, \mathcal{M}[i]$))

Function FINDMOSTSIMILARADAPTER($\mathcal{M}$, $\mathcal{S}_k$)
  for each $i \in \{1, \ldots, |\mathcal{M}|\}$ **do**
    adapter $\leftarrow \mathcal{M}[i]$
    $\mathbf{c}_a \leftarrow$ adapter.centroid
    $\mathbf{e}_{\text{new}} \leftarrow$ GETEMBEDDINGS($\mathcal{S}_k$, adapter)
    $\mathbf{c}_{\text{new}} \leftarrow$ AVG($\mathbf{e}_{\text{new}}$)
    similarity$[i] \leftarrow$ COSINESIMILARITY($\mathbf{c}_a, \mathbf{c}_{\text{new}}$)
  end for
  return argmax(similarity), max(similarity)

C Implementation Details

CLIP+CLAP backbone. We employ CLIP (Radford et al., 2021) and CLAP (Wu et al., 2023) as our video and audio encoders, which stay frozen during pretraining and adaptation. We use the pretrained checkpoints from Huggingface: `laion/ViT-B-16-laion2b_s34b_b88k` for CLIP, `laion/clap-htsat-unfused` for CLAP. The dimensionality of the CLIP and CLAP features is 512. Self attention, cross attention and final attentive pooling layers use 8 attention heads and a feedforward dimension of the MLP layers of 2048 (4×512). Each video is cropped around the face region using MTCNN (Zhang et al., 2016) and resized to 224×224. Following AVFF (Oorloff et al., 2024), we sample 16 frames per video with a stepsize of 5. Audio is temporally aligned with the video and resampled to 48 kHz, the sampling rate used by CLAP during training. LoRA is applied to the cross-attention and to the Attentive Pooling Classifier.

BRAVEn backbone. We use the frozen audio and visual encoders of BRAVEn (Haliassos et al., 2024) (Large), finetuned on VSR and ASR, which give 1024-dimensional frame-wise features for the same 16 frames (stepsize 5, 96×96 mouth crops) and the aligned audio. Since the encoders are frozen, the features are extracted once and cached. Here the detection head has no self- or cross-attention: the Attentive Pooling Classifier (depth 1, 16 heads, feedforward dimension 4×1024) is the only module, and in the AV variant it is where the two modalities interact, with the audio and video tokens concatenated along the temporal axis. LoRA is applied to the classifier, which gives about 1.05M trainable parameters per adapter at $r = 64$. In the video-only variant (V), the classifier receives only the video tokens, and the base detector is pretrained on DFDC with the video manipulation label.

The BRAVEn base detectors are pretrained on DFDC with a learning rate of 10^{-4} for up to 150 epochs. Adapters are trained for up to 20 epochs with a batch size of 256, and all other hyperparameters match the CLIP+CLAP setting.

Training. We optimize the model using AdamW (Loshchilov & Hutter, 2019) with a learning rate of 10^{-3} and a weight decay of 0.05. We use a combination of BCE and margin loss, with $\lambda = 0.2$ and margin value of 0.5. The learning rate is reduced when the validation loss plateaus, and early stopping is also applied. We train for a maximum of 150 epochs, with a batch size of 256. In the incremental learning setting, we set the LoRA rank to $r = 64$; for the joint training model, we set $r = 256$. We apply LoRA to the attention and MLP weights of the cross-attention and classifier layers only (the classifier only for BRAVEn); the self-attention layers are not adapted (as in § 5 and Table 2). All experiments are conducted on a single NVIDIA A100 GPU, with about 31 GB of GPU memory used during training (45 GB reserved).

For the key-query router (DEFEND-KQ), we train a lightweight routing network (2-layer MLP). The query projection module uses a projecting dimension of 256 and a hidden dimension of 512, with a dropout set to 0.1. The router is also trained using AdamW, with a learning rate of 10^{-3}, weight decay of 10^{-4}, temperature scaling of 0.5, batch size of 256, and a maximum of 100 epochs. We initialize the keys using the adapter prototypes and further optimize them during training. For DEFEND-SIM, the cosine similarity threshold is set to $\tau = 0.7$.

C.1 Cost

Training cost. One DEFEND adapter update costs 1.07 GFLOPs per sample in the CLIP+CLAP head and 0.21 GFLOPs in the BRAVEn head (forward and backward), i.e. 77% and 43% of fully finetuning the same head, since the frozen base weights need no weight gradients. A router training step costs below 3 MFLOPs per stored embedding, so even a full router update with all 55,553 embeddings and 100 epochs stays around 16 TFLOPs, i.e. at most about 2% (CLIP+CLAP) and 10% (BRAVEn) of the adapter updates of one domain. The router is thus a small fraction of the cost of adapting to a new domain.

Wall-clock time. We measure the training time from the checkpoint timestamps of the runs, each on a single NVIDIA A100 GPU. With CLIP+CLAP, training a new adapter on a domain takes about 2 hours, updating a reused adapter about 1 hour, and retraining the key-query router 13-25 minutes; the full stream of 23 domains takes about 70 hours. Adapter training uses about 31 GB of GPU memory (45 GB reserved) at a batch size of 256, most of it for the frozen encoders processing the videos. With

the cached BRAVEn features, a new adapter takes 2-4 minutes, a router update 2-3 minutes, and the whole stream 3-4 hours, and adapter training uses less than 1 GB of GPU memory.

Inference cost. At inference, DEFEND-KQ runs the frozen encoders once, then the detection head twice (once to compute the query and once with the selected adapter) and the router (0.39M parameters for CLIP+CLAP and 0.66M for BRAVEn). Since the encoders dominate the cost (372.5 GFLOPs per clip for CLIP+CLAP and 161.1 GFLOPs for BRAVEn, against 0.70 and 0.17 GFLOPs for the head and below 1.5 MFLOPs for the router), the overhead over a single model is 0.2% for CLIP+CLAP and 0.1% for BRAVEn. It does not grow with the number of adapters, since only one adapter is used per sample, and GT and automatic routing have the same cost.

Storage. Each adapter adds 1.5M (CLIP+CLAP) or 1.05M (BRAVEn) parameters at $r = 64$. The router is trained on the stored embeddings of the fake training clips of all seen domains, each computed through the adapter assigned to its domain (one vector per clip), which is 55,553 vectors after the last domain, i.e. 108 MiB for CLIP+CLAP (512-d) and 217 MiB for BRAVEn (1024-d) in float32.

D Continual Learning Baselines

In Table 5, we compare our approach against four standard continual learning techniques: Elastic Weight Consolidation (Kirkpatrick et al., 2017) (EWC), Knowledge Distillation (Hinton et al., 2015) combined with our margin loss (KD+Margin), DualPrompt (Wang et al., 2022b) and Dark Experience Replay++ (DER++) (Buzzega et al., 2020).

EWC mitigates forgetting by constraining important parameters to remain close to their previous values when learning new domains. This is achieved through a regularization term weighted by the diagonal approximation of the Fisher information matrix, which we estimate using 200 samples per domain. The regularization weight is set to 1 000.

For KD+Margin, the frozen pretrained model acts as the teacher, while the incrementally adapted model acts as the student. Since the teacher is only familiar with the pretraining domain, we apply distillation exclusively to samples originating from the pretraining dataset. KD+Margin therefore replays DFDC samples during incremental training, which makes it a form of rehearsal on the pretraining domain. Instead of distilling the final binary logits, we minimize the mean squared error between the teacher and student embeddings, allowing the student model to preserve richer feature representations across domains.

DualPrompt is a prompting-based continual learning method that separates generic and task-specific knowledge through two types of prompts: shared G-prompts and task-specific E-prompts. These prompts are concatenated to the keys and values before the attention operation and optimized using the classification loss. During inference, the most similar set of E-prompts is selected. For a fair comparison, we apply the prompts only to the cross-attention layers and the final attentive pooling layer, while keeping the remaining hyperparameters from the original implementation unchanged.

DER++ maintains an experience replay buffer containing samples from previously observed domains. We use a memory buffer of 500 samples, storing the input, target label, and logits predicted when the sample was first encountered. During training, the method combines a supervised classification loss computed on replayed samples with a mean squared error loss between stored and current logits, preserving prior knowledge through rehearsal and distillation.

E Results

Per-Domain Analysis Table 7 reports, for each domain, the AUC of GT-KQ and DEFEND-KQ and the routing accuracy of the key-query router, for the three backbones on the test set. With BRAVEn, most domains are detected almost perfectly: the lip-sync and audio-driven methods ($\mathcal{T}_5$, $\mathcal{T}_6$, $\mathcal{T}_9$, $\mathcal{T}_{17}$, $\mathcal{T}_{23}$) and most KoDF domains are above 95% AUC for both the audio-visual and the video-only model. The hardest domains are the face swaps of DeepSpeak and MAVOS-DD, in particular SimSwap ($\mathcal{T}_7$), HiFiFace ($\mathcal{T}_8$) and Roop ($\mathcal{T}_{15}$), which stay below 83% even with GT routing on all backbones. On CLIP+CLAP, the face swaps of DeepSpeak ($\mathcal{T}_{12}$, $\mathcal{T}_{14}$, $\mathcal{T}_{16}$) remain close to

Table 7: **Per-domain performance of DEFEND-KQ.** AUC (%) of GT-KQ and DEFEND-KQ for each domain, and routing accuracy (%) of the key-query router, on the test set. Routing ACC is computed on the fake samples of each domain. As $\mathcal{T}_0$ does not have an adapter, routing ACC is not reported.

Backbone	Method	$\mathcal{T}_0$	$\mathcal{T}_1$	$\mathcal{T}_2$	$\mathcal{T}_3$	$\mathcal{T}_4$	$\mathcal{T}_5$	$\mathcal{T}_6$	$\mathcal{T}_7$	$\mathcal{T}_8$	$\mathcal{T}_9$	$\mathcal{T}_{10}$	$\mathcal{T}_{11}$
CLIP+CLAP	GT-KQ	98.05	93.14	97.31	83.60	87.85	96.18	82.59	74.43	79.91	81.13	70.99	81.85
	DEFEND-KQ	80.80	78.73	81.63	73.36	81.46	91.46	81.16	69.17	79.31	78.11	70.57	80.78
	Routing ACC	-	33.38	99.04	65.96	41.46	65.93	84.08	63.69	99.02	74.42	77.49	90.66
BRAVEn AV	GT-KQ	81.50	91.56	99.96	99.99	99.18	99.99	99.53	68.65	80.88	99.41	88.98	95.69
	DEFEND-KQ	67.85	84.28	99.69	99.62	95.85	99.63	96.27	69.71	79.61	97.66	86.25	93.77
	Routing ACC	-	57.95	100.00	91.49	66.59	84.51	65.50	45.83	83.79	79.94	68.58	59.45
BRAVEn V	GT-KQ	82.06	93.44	97.11	99.97	98.37	99.99	99.84	72.68	75.35	98.64	84.84	92.82
	DEFEND-KQ	72.03	88.07	94.74	99.71	97.15	99.59	98.62	66.06	70.84	98.44	81.41	92.47
	Routing ACC	-	7.72	77.31	95.13	84.42	92.57	87.79	43.45	78.24	86.41	69.43	83.23

Backbone	Method	$\mathcal{T}_{12}$	$\mathcal{T}_{13}$	$\mathcal{T}_{14}$	$\mathcal{T}_{15}$	$\mathcal{T}_{16}$	$\mathcal{T}_{17}$	$\mathcal{T}_{18}$	$\mathcal{T}_{19}$	$\mathcal{T}_{20}$	$\mathcal{T}_{21}$	$\mathcal{T}_{22}$	$\mathcal{T}_{23}$
CLIP+CLAP	GT-KQ	54.22	72.86	53.66	73.97	56.77	99.93	97.70	70.30	84.96	99.11	96.96	99.91
	DEFEND-KQ	54.80	68.17	53.71	73.23	53.38	98.50	96.53	70.68	72.59	97.14	88.01	97.86
	Routing ACC	46.71	97.21	53.74	98.52	43.97	81.36	89.94	81.98	63.29	79.78	71.54	82.84
BRAVEn AV	GT-KQ	91.06	89.08	89.81	82.62	80.86	100.00	96.69	90.91	96.30	97.55	96.96	100.00
	DEFEND-KQ	88.20	85.00	85.43	81.58	82.53	100.00	92.05	89.80	92.05	91.86	93.64	99.99
	Routing ACC	65.18	87.01	44.22	86.43	43.26	100.00	54.64	70.68	48.52	48.86	82.58	100.00
BRAVEn V	GT-KQ	87.99	91.64	83.44	77.05	82.80	99.29	99.18	95.06	98.85	98.68	96.93	97.01
	DEFEND-KQ	85.05	84.56	83.85	75.82	76.45	98.15	91.13	89.10	97.47	93.19	85.85	90.93
	Routing ACC	64.12	89.17	48.30	85.97	46.81	93.73	52.77	80.02	61.60	62.15	60.71	75.62

chance, while the lip-sync and avatar methods ($\mathcal{T}_{17}$, $\mathcal{T}_{18}$, $\mathcal{T}_{21}$–$\mathcal{T}_{23}$) are above 96% with GT routing. The routing accuracy varies a lot across domains, from above 95% (e.g., $\mathcal{T}_2$ and $\mathcal{T}_{17}$ on BRAVEn AV, $\mathcal{T}_{13}$ and $\mathcal{T}_{15}$ on CLIP+CLAP) to about 50% or less for domains that look similar to earlier ones (e.g., $\mathcal{T}_{14}$ and $\mathcal{T}_{16}$ on all backbones, and $\mathcal{T}_1$ for the video-only model). A low routing accuracy does not necessarily lead to a low AUC: on BRAVEn AV, Sonic ($\mathcal{T}_{21}$) is routed to its own adapter for only 48.86% of the fake samples but reaches 91.86% AUC, and for some domains ($\mathcal{T}_7$ and $\mathcal{T}_{16}$ on BRAVEn AV, $\mathcal{T}_{12}$ and $\mathcal{T}_{19}$ on CLIP+CLAP) the routed AUC is even slightly higher than with GT routing.

Domain grouping produced by DEFEND-KQ Table 8 lists the domains that DEFEND-KQ assigns to the same adapter for the three backbones, as discussed in § 5.3. On all backbones, each of the first KoDF domains ($\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_4$) gets its own adapter, the diffusion-based lip-sync methods Diff2Lip and LatentSync share an adapter, and InSwapper and Roop are assigned to the same adapter. CLIP+CLAP merges the lip-sync methods Wav2Lip and Retalking with SimSwap and the FaceFusion variants, while both BRAVEn models keep the lip-sync methods apart from the face swaps. Without the audio stream, BRAVEn V puts all four lip-sync methods (Wav2Lip, Retalking, Diff2Lip, LatentSync) into one adapter and gives EchoMimic, Sonic and Memo their own adapters.

Within-group generalization To check whether the groups found by the key-query router are meaningful, we run a leave-one-domain-out experiment on BRAVEn AV, seed 1708. For each domain of the two largest KQ groups of Table 8 (7 and 6 domains), we train one LoRA adapter from the DFDC base on the other domains of the same group (*in-group*) and one on the same number of domains taken from other groups (*out-of-group*), with the same number of samples, and test both on the held-out domain. We also report the zero-shot base model and the adapter trained on the held-out domain itself. Results are in Table 9. The in-group adapter is better than the out-of-group one on 11 of the 13 held-out domains, by 4.05 points on average (+2.3 for the first group and +6.1 for the second), and it almost reaches the adapter trained on the domain itself (89.55% vs. 89.71%). For several face-swap domains ($\mathcal{T}_7$, $\mathcal{T}_{10}$, $\mathcal{T}_{12}$, $\mathcal{T}_{14}$, $\mathcal{T}_{16}$) the in-group adapter is even better than the own adapter, showing that these domains can share one adapter. The second group is less homogeneous: for HiFiFace ($\mathcal{T}_8$), EchoMimic ($\mathcal{T}_{18}$) and Memo ($\mathcal{T}_{22}$) the own adapter is still 3–13 points better, and for Memo the out-of-group adapter is better; in the first group, this is also the case for HelloMeme ($\mathcal{T}_{20}$). For $\mathcal{T}_{10}$, $\mathcal{T}_{12}$ and $\mathcal{T}_{16}$ the own adapter is even below the zero-shot model (e.g., 71.71% vs. 77.32% on $\mathcal{T}_{16}$), probably because these domains are small (0.3–0.8k samples) and the adapter overfits. For the two groups with only two domains (Wav2Lip/Retalking and Diff2Lip/LatentSync) all settings are close to 100% AUC.

Table 8: **Domain grouping produced by DEFEND-KQ on the different backbones.** Domains assigned to the same adapter are grouped together; adapters are numbered in order of creation. Reuse margin 0.3, full training set, one seed shown.

Adapter	CLIP+CLAP	BRAVEn AV	BRAVEn V
adapter1	$\mathcal{T}_1$ DFFS	$\mathcal{T}_1$ DFFS	$\mathcal{T}_1$ DFFS
adapter2	$\mathcal{T}_2$ FSGAN	$\mathcal{T}_2$ FSGAN	$\mathcal{T}_2$ FSGAN
adapter3	$\mathcal{T}_3$ FOMM, $\mathcal{T}_5$ Wav2Lip+ATFHP	$\mathcal{T}_3$ FOMM	$\mathcal{T}_3$ FOMM
adapter4	$\mathcal{T}_4$ DFL	$\mathcal{T}_4$ DFL	$\mathcal{T}_4$ DFL
adapter5	$\mathcal{T}_6$ Wav2Lip, $\mathcal{T}_7$ SimSwap, $\mathcal{T}_9$ Retalking, $\mathcal{T}_{10}$ FaceFusion, $\mathcal{T}_{11}$ FaceFusion Live, $\mathcal{T}_{20}$ HelloMeme	$\mathcal{T}_5$ Wav2Lip+ATFHP	$\mathcal{T}_5$ Wav2Lip+ATFHP
adapter6	$\mathcal{T}_8$ HiFiFace, $\mathcal{T}_{13}$ InSwapper, $\mathcal{T}_{14}$ InSwapper+CodeFormer, $\mathcal{T}_{15}$ Roop, $\mathcal{T}_{19}$ LivePortrait	$\mathcal{T}_6$ Wav2Lip, $\mathcal{T}_9$ Retalking	$\mathcal{T}_6$ Wav2Lip, $\mathcal{T}_9$ Retalking, $\mathcal{T}_{17}$ Diff2Lip, $\mathcal{T}_{23}$ LatentSync
adapter7	$\mathcal{T}_{12}$ FaceFusion GAN, $\mathcal{T}_{16}$ SimSwap+RestoreFormer	$\mathcal{T}_7$ SimSwap, $\mathcal{T}_{10}$ FaceFusion, $\mathcal{T}_{11}$ FaceFusion Live, $\mathcal{T}_{12}$ FaceFusion GAN, $\mathcal{T}_{14}$ InSwapper+CodeFormer, $\mathcal{T}_{16}$ SimSwap+RestoreFormer, $\mathcal{T}_{20}$ HelloMeme	$\mathcal{T}_7$ SimSwap, $\mathcal{T}_{10}$ FaceFusion, $\mathcal{T}_{11}$ FaceFusion Live, $\mathcal{T}_{12}$ FaceFusion GAN, $\mathcal{T}_{14}$ InSwapper+CodeFormer, $\mathcal{T}_{16}$ SimSwap+RestoreFormer
adapter8	$\mathcal{T}_{17}$ Diff2Lip, $\mathcal{T}_{23}$ LatentSync	$\mathcal{T}_8$ HiFiFace, $\mathcal{T}_{13}$ InSwapper, $\mathcal{T}_{15}$ Roop, $\mathcal{T}_{18}$ EchoMimic, $\mathcal{T}_{19}$ LivePortrait, $\mathcal{T}_{22}$ Memo	$\mathcal{T}_8$ HiFiFace, $\mathcal{T}_{13}$ InSwapper, $\mathcal{T}_{15}$ Roop, $\mathcal{T}_{19}$ LivePortrait, $\mathcal{T}_{20}$ HelloMeme
adapter9	$\mathcal{T}_{18}$ EchoMimic, $\mathcal{T}_{21}$ Sonic	$\mathcal{T}_{17}$ Diff2Lip, $\mathcal{T}_{23}$ LatentSync	$\mathcal{T}_{18}$ EchoMimic
adapter10	$\mathcal{T}_{22}$ Memo	$\mathcal{T}_{21}$ Sonic	$\mathcal{T}_{21}$ Sonic
adapter11	–	–	$\mathcal{T}_{22}$ Memo

Table 9: **Within-group generalization** (BRAVEn AV, full dataset train, test set). AUC (%) on the held-out domain of an adapter trained on the other domains of its KQ group (In), on the same number of domains from other groups (Out), of the zero-shot base model, and of an adapter trained on the held-out domain (Own).

Group	Held-out domain	In ↑	Out ↑	Zero-shot	Own
A (7 domains)	$\mathcal{T}_7$ SimSwap	**70.39**	66.62	63.61	64.29
	$\mathcal{T}_{10}$ FaceFusion	**94.40**	91.95	92.60	92.17
	$\mathcal{T}_{11}$ FaceFusion Live	**98.15**	96.45	98.63	98.98
	$\mathcal{T}_{12}$ FaceFusion GAN	**95.32**	92.88	94.35	93.33
	$\mathcal{T}_{14}$ InSwapper+CodeFormer	**94.87**	94.73	87.84	89.84
	$\mathcal{T}_{16}$ SimSwap+RestoreFormer	**84.75**	77.58	77.32	71.71
	$\mathcal{T}_{20}$ HelloMeme	94.06	**95.54**	97.13	98.26
B (6 domains)	$\mathcal{T}_8$ HiFiFace	**76.78**	69.68	57.72	90.23
	$\mathcal{T}_{13}$ InSwapper	**92.99**	81.58	85.96	93.71
	$\mathcal{T}_{15}$ Roop	**85.22**	72.70	70.35	83.38
	$\mathcal{T}_{18}$ EchoMimic	**95.03**	90.19	73.17	98.04
	$\mathcal{T}_{19}$ LivePortrait	**93.46**	89.69	92.58	95.00
	$\mathcal{T}_{22}$ Memo	88.68	**91.89**	83.96	97.32
Mean (13 domains)		**89.55**	85.50	82.71	89.71

Routing over time Table 10 shows how the router of DEFEND-KQ behaves as more domains are learned (BRAVEn AV, reuse margin 0.3, 3 seeds). After each step t we take the router and the adapters as they are at that point and route the test clips of all domains seen so far. To check whether the routing of old domains degrades, we also report the routing accuracy on the clips of the first five domains only. The routing accuracy on early domains stays stable: on $\mathcal{T}_1$–$\mathcal{T}_5$ is 80.5% after $\mathcal{T}_5$ and 79.1% after $\mathcal{T}_{23}$, although 18 more domains and 5–8 more adapters were added in between. The limit of about 80% on these domains comes from confusion between the KoDF face-swap methods and is already visible after $\mathcal{T}_4$. The overall routing accuracy goes down from about 85% to 78%, mostly because of late domains that look similar to earlier ones ($\mathcal{T}_{14}$, $\mathcal{T}_{16}$, $\mathcal{T}_{20}$ and $\mathcal{T}_{21}$ are each routed correctly for 43–49% of their clips). The routed AUC follows the routing accuracy only weakly and stays between 89% and 92% after $\mathcal{T}_{10}$.

Ground-truth-domain vs. best-available adapter GT routing sends each domain to the adapter that was assigned to it during training. This is not necessarily the best adapter for that domain, since another adapter may have learned similar artifacts. In Table 11 we also report, for each domain, the AUC of the best adapter among all final adapters. The assigned adapter is not the best one for 7–10 of the 24 domains, mostly the small face-swap domains of DeepSpeak, but the difference in Avg AUC is small (0.9–2.0 points). The gap between GT and routed AUC is therefore mainly due to routing errors and not to a wrong choice of the assigned adapter.

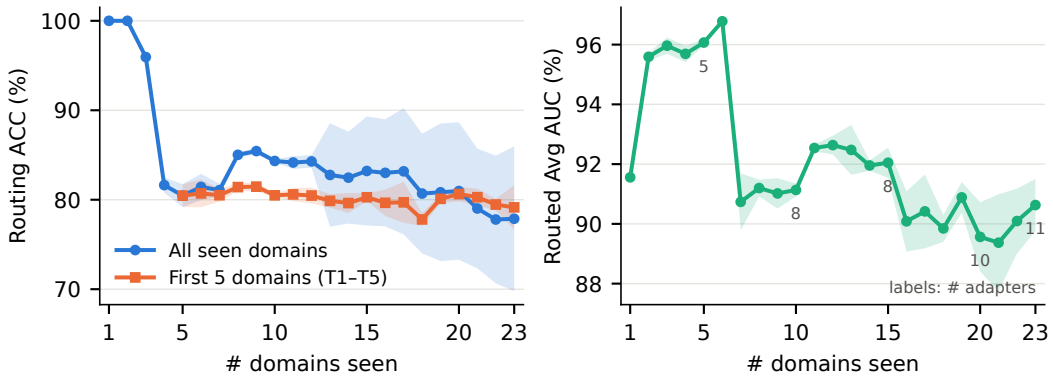


Figure 3: **Routing over time** (BRAVEn AV, DEFEND-KQ, test set, mean±std over different runs). Left: routing accuracy on all seen domains and on the first five domains. Right: routed Avg AUC over the seen domains, with the number of adapters.

Table 10: **Routing after t domains** (BRAVEn AV, DEFEND-KQ, test set, mean±std over different runs; std for the overall routing accuracy). Routing accuracy is computed on the fake clips of the seen domains, overall and per source dataset; routed AUC is averaged over $\mathcal{T}_1$–$\mathcal{T}_t$ (without DFDC).

t	N_{ADP}	ROUTING ACC	KODF	MAVOS-DD	DEEPSPEAK	ACC ON $\mathcal{T}_1$–$\mathcal{T}_5$	ROUTED AUC
5	5.0	80.46 ± 1.28	80.46	–	–	80.46	96.07
10	8.0	84.33 ± 0.17	80.49	98.19	88.43	80.49	91.14
15	8.3	83.22 ± 6.09	80.28	89.24	79.35	80.28	92.05
20	10.3	80.98 ± 7.68	80.64	84.26	72.02	80.64	89.56
23	11.3	77.88 ± 8.09	79.15	78.74	71.03	79.15	90.63

MAVOS-DD per language MAVOS-DD contains videos in seven languages, so we also report the AUC on its domains for each language separately (Table 12). Hindi is the hardest language on all backbones, with a GT AUC of about 77% against 84–94% for the other languages. For Hindi, the routing also causes a larger drop, in particular for the video-only model (76.12% → 54.93%), while for the other languages DEFEND-KQ stays within 6 points of GT routing. Russian has only 63 real test clips, so its AUC is less reliable. The differences between languages are therefore mostly due to the detection of Hindi and to the routing of the Hindi samples, and not to a general drop on non-English data.

Audio-manipulated vs. real-audio domains Several domains keep the original audio, so we split the 23 incremental domains into those with manipulated audio ($\mathcal{T}_6$, $\mathcal{T}_9$, $\mathcal{T}_{17}$, $\mathcal{T}_{22}$, $\mathcal{T}_{23}$) and those with real audio only (the other 18), following Table 1, and compare the audio-visual (AV) and the video-only (V) BRAVEn models (Table 13). AV is better than V by 0.84 points with GT routing and by 3.12 points with automatic routing on the audio-manipulated domains ($p = 0.012$ and $p = 0.001$, Welch t-test over seeds), and by 0.58 and 1.68 points on the real-audio domains ($p = 0.58$ and $p = 0.24$). With only three seeds the test has little power, so the higher p-values on the real-audio domains do not mean that audio brings no gain there, and the difference between the two groups is small under GT routing.

Random routing To check that the gains of DEFEND do not come from the adapters alone, we keep the adapters learned by DEFEND-KQ and send each test sample to a random adapter instead of the one selected by the router (Table 14). As expected, the random assignment picks the right adapter for about one sample in N_{adp} (9–13%), and the AUC drops to 59.87% on CLIP+CLAP, 68.88% on BRAVEn AV and 71.55% on BRAVEn V, i.e. 22.9, 23.5 and 20.2 points below GT routing, with more than 20 points of forgetting. The key-query router recovers most of this gap: DEFEND-KQ is only 2.7–4.9 points below GT routing on all three settings, although it selects the adapter of the right domain for only 74–78% of the fake samples. This shows that the adapters are specialized, so that a random adapter strongly hurts the detection.

Table 11: **Ground-truth-domain vs. best-available adapter.** Avg AUC (%) over $\mathcal{T}_0$–$\mathcal{T}_{23}$, test set, mean±std over 3 seeds. Best adapter: per domain, the final adapter with the highest AUC (for $\mathcal{T}_0$ the pretrained model is also a candidate). #Best≠GT: number of domains where the best adapter is not the assigned one.

BACKBONE	BEST ADAPTER	GT	DEFEND-KQ	#BEST≠GT
CLIP+CLAP	84.79 ± 2.29	82.81 ± 2.76	77.96 ± 2.31	7.3
BRAVEN AV	93.28 ± 0.70	92.38 ± 0.73	89.68 ± 0.90	6.7
BRAVEN V	93.58 ± 0.93	91.79 ± 1.00	87.95 ± 1.15	9.7

Table 12: **MAVOS-DD per language.** AUC (%) of GT-KQ and DEFEND-KQ on the pure MAVOS-DD domains ($\mathcal{T}_8$, $\mathcal{T}_{15}$, $\mathcal{T}_{18}$, $\mathcal{T}_{21}$; the mixed domains $\mathcal{T}_{13}$, $\mathcal{T}_{19}$ and $\mathcal{T}_{22}$ are excluded), computed separately for each language (fakes and reals of that language), test set, reuse margin 0.3, mean over 3 seeds. #Real / #Fake: number of real and fake MAVOS-DD test clips of each language.

		CLIP+CLAP		BRAVEN AV		BRAVEN V	
LANGUAGE	#REAL / #FAKE	GT	DEFEND-KQ	GT	DEFEND-KQ	GT	DEFEND-KQ
ENGLISH	1598 / 1799	85.74	84.27	90.70	88.45	93.05	88.69
GERMAN	855 / 960	87.20	86.99	85.44	83.37	88.36	85.20
HINDI	763 / 586	76.99	75.53	76.88	70.89	76.12	54.93
MANDARIN	452 / 933	90.08	89.10	89.74	87.91	84.52	79.65
ROMANIAN	788 / 962	87.46	87.01	90.16	87.10	94.43	88.99
RUSSIAN	63 / 399	86.03	85.19	85.02	83.93	85.19	83.74
SPANISH	604 / 1566	88.83	86.78	86.13	83.09	88.83	83.81

Router memory and stability The key-query router is retrained after each domain on the stored embeddings of the fake training clips of all seen domains, each computed through the adapter assigned to its domain, which amounts to 217 MiB after the last domain. To measure how much of this memory is needed, we train BRAVEn DEFEND-KQ keeping at most B embeddings per domain during the whole stream (Table 15). With less memory, the router reuses the existing adapters less often and allocates 21–23 adapters instead of 15.7, i.e. almost one adapter per domain. The GT AUC therefore stays high (93.2–93.5%), but the routed AUC decreases from 89.72% to 87.99% with 500 embeddings per domain and to 82.69% with 10, together with the routing accuracy (from 67.77% to 28.35%), since more adapters with fewer stored samples each are harder to tell apart. When we instead cap the memory only for the final router, keeping the adapters and the allocation decisions of the full-memory run, the loss is smaller (88.27% with 500 and 86.29% with 10 embeddings per domain), so most of the drop comes from the allocation decisions made with less memory. The two methods store different data. DER++ keeps a replay buffer of 500 model inputs in total, which are raw clips for CLIP+CLAP (about 4.6 GiB) and the cached encoder features for BRAVEn (62.5 MiB, i.e. 128 KiB per sample). The router instead keeps one embedding per fake training clip of every seen domain (2 KiB for CLIP+CLAP, 4 KiB for BRAVEn). With all 55,553 embeddings, the router needs more memory than DER++ on BRAVEn (217 vs. 62.5 MiB); with B=500 per domain it covers about 20× more clips than the DER++ buffer in less memory (39.7 MiB), with a higher routed AUC than DER++ (87.99% vs. 78.51%).

F ABLATIONS

LoRA rank with a single adapter Table 19 reports the LoRA rank ablation when a single adapter is trained incrementally on the 23 domains (CLIP+CLAP, GT routing, validation set, 500 samples per domain). Lower ranks reduce forgetting, with $r = 16$ achieving the lowest forgetting, while larger ranks improve the adaptation capacity, with $r = 64$ obtaining the highest AUC. With a single shared adapter, the rank therefore trades adaptation against forgetting, which motivates isolating the domains in multiple adapters, where $r = 64$ is used.

Joint-Training Ablation We analyze the effect of the LoRA (Hu et al., 2022) rank in the joint-training setting using a single adapter trained jointly across all 23 domains. As shown in Table 20, the LoRA rank has a significant impact on the final performance. Small ranks such as $r = 64$ and

Table 13: **Audio-manipulated vs. real-audio domains.** Mean AUC (%) over the domains of each group, test set, mean over 3 seeds. AV−V: gain of the audio-visual over the video-only BRAVEn model, with the p-value of a Welch t-test over seeds.

	GT		DEFEND-KQ	
BACKBONE	AUDIO MANIP. (5)	REAL AUDIO (18)	AUDIO MANIP. (5)	REAL AUDIO (18)
CLIP+CLAP	92.10	79.38	88.73	74.82
BRAVEN AV	99.18	91.10	97.51	88.72
BRAVEN V	98.34	90.52	94.40	87.04
AV−V	+0.84 (p=0.012)	+0.58 (p=0.58)	+3.12 (p=0.001)	+1.68 (p=0.24)

Table 14: **Random routing.** The adapters of DEFEND-KQ are kept, and each test sample is sent to a uniformly random adapter. Full training set, test set, mean ± std over 3 seeds. Average AUC and forgetting over $\mathcal{T}_0$–$\mathcal{T}_{23}$; Routing ACC on the fake samples of the incremental domains.

				FORGET↓		
BACKBONE	ROUTING	N_{ADP}	AUC (%)↑	ALL	W/O DFDC	ROUTING ACC (%)↑
CLIP+CLAP	GT	8.0	82.81 ± 2.76	4.69 ± 1.58	4.90 ± 1.65	100.00
	RANDOM		59.87 ± 1.52	27.77 ± 1.13	27.82 ± 1.19	13.08 ± 3.35
	DEFEND-KQ		77.96 ± 2.31	9.65 ± 1.45	9.31 ± 1.53	74.43 ± 5.66
BRAVEN AV	GT	11.3	92.38 ± 0.73	1.42 ± 0.92	1.49 ± 0.96	100.00
	RANDOM		68.88 ± 0.46	25.33 ± 0.50	25.57 ± 0.50	8.96 ± 1.12
	DEFEND-KQ		89.68 ± 0.90	4.24 ± 1.05	3.81 ± 1.02	77.88 ± 8.10
BRAVEN V	GT	11.3	91.79 ± 1.00	2.04 ± 0.99	2.13 ± 1.04	100.00
	RANDOM		71.55 ± 1.34	21.83 ± 1.31	21.77 ± 1.32	8.92 ± 0.38
	DEFEND-KQ		87.95 ± 1.15	5.79 ± 1.17	5.60 ± 1.33	74.34 ± 4.00

$r = 128$ provide limited adaptation capacity and lead to lower average AUC. Increasing the rank to $r = 256$ substantially improves the overall performance and achieves the best average AUC of 67.14%.

Interestingly, further increasing the rank does not provide additional gains. Both $r = 512$ and $r = 1024$ result in lower average performance, suggesting that excessively large ranks may reduce optimization stability and increase overfitting to specific domains. These results indicate that an intermediate rank provides the best tradeoff between adaptation capacity and generalization in the joint-training setting. Consequently, we use $r = 256$ for the final joint-training baseline.

DEFEND-KQ router capacity We also analyze the effect of the routing network capacity by varying the size of the projection MLP used to compute the query embeddings (see Table 21). Specifically, we compare a smaller router with a projection dimension of 128 and a hidden dimension of 256 to the default configuration with a projection dimension of 256 and a hidden dimension of 512. The results reveal a tradeoff between routing accuracy and generalization. The larger router achieves a substantially better routing accuracy (86.41% vs. 77.59%) and a higher routed AUC. The smaller router gives a slightly higher GT AUC; since the router also decides during training which adapter each domain reuses, the two routers lead to different adapter allocations, which is the only way the router affects the GT AUC.

Overall, we select the larger router as the default configuration since it provides more consistent routing behavior and better overall performance in the fully automatic routing setting, which is the practical scenario considered in this work.

Compatibility score of the new adapter The compatibility score s_{new} of the candidate adapter decides when none of the existing adapters is good enough for the incoming domain and a new adapter has to be created. To check whether it is needed, we remove it and replace it with a fixed rule: a new adapter is created only if $\max_i \cos(\mathbf{q}, \mathbf{k}_i) < \tau$. With $\tau = 0$ (w/o s_{new}) the router almost never expands. Results are in Table 22.

On BRAVEn (full training set, test set), removing s_{new} collapses the model to 5 adapters in all seeds. The routing accuracy goes up (86.66%), but only because there are fewer adapters to choose from: the

Table 15: **Router memory.** BRAVEn AV DEFEND-KQ trained with at most B stored embeddings per domain during the whole stream, without reuse margin. Average AUC and forgetting over $\mathcal{T}_0$–$\mathcal{T}_{23}$, test set, mean ± std over 3 seeds.

B	MEMORY (MIB)	N_{ADP}	GT AUC (%)↑	ROUTED AUC (%)↑	ROUTED FORGET↓	ROUTING ACC↑
ALL	217	15.7	92.92 ± 0.26	**89.72 ± 1.34**	**4.11 ± 1.40**	**67.77 ± 2.88**
500	39.7	21.0	93.22 ± 0.07	87.99 ± 0.75	5.80 ± 0.82	47.01 ± 2.05
200	17.4	22.7	**93.48 ± 0.18**	86.64 ± 0.59	7.05 ± 0.61	40.52 ± 0.35
50	4.5	23.0	93.21 ± 0.08	85.10 ± 0.75	8.45 ± 0.79	34.32 ± 1.36
10	0.9	23.0	93.35 ± 0.04	82.69 ± 2.66	11.13 ± 2.74	28.35 ± 2.59

Table 16: **Architecture ablation.** Blocks kept in the CLIP+CLAP head, trained on the **full** training set (23 incremental domains) with DEFEND-KQ and evaluated on the validation split with GT routing. Average AUC and average forgetting over $\mathcal{T}_0$–$\mathcal{T}_{23}$ ($\mathcal{T}_0$: pretrained model on DFDC).

HEAD	N_{ADP}	AUC (%)↑	FORGET↓
CROSS+CLASS	12	**84.95**	5.64
SELF+CLASS	9	83.48	6.46
SELF+CROSS+CLASS	11	83.95	**4.77**

Table 17: **Ablation of the adapter reuse margin.** CLIP+CLAP, GT and routed average AUC (%) over $\mathcal{T}_0$–$\mathcal{T}_{23}$. Train on 500 samples/domain, validation set.

MARGIN	N_{ADP}	GT AUC↑	ROUTED AUC↑
0.0	12	77.52	74.42
0.1	11	**78.72**	72.13
0.2	11	78.60	74.38
0.3	**9**	78.58	**74.93**
0.5	8	77.18	73.91

GT AUC drops by about 7 points (92.38% → 85.44%), the routed AUC by about 10 points (89.68% → 79.34%), and the forgetting grows from 1.42 to 8.51 (GT) and from 4.24 to 14.87 (routed). The variance over seeds also becomes 3–5× larger, since the result depends on which domains are forced to share an adapter. Note that DEFEND-KQ already uses only 11.3 adapters for 23 domains, so fewer adapters are not a problem by themselves; the problem is merging domains that should not be merged.

On CLIP+CLAP with 500 samples per domain the effect is smaller. Without s_{new} the GT AUC drops by 2.3 points (78.58% → 76.25%) with 5 instead of 9 adapters, and the GT forgetting grows from 2.75 to 4.41, showing that merging domains into fewer adapters increases the interference between them. Under automatic routing, instead, the model without s_{new} is slightly better (75.68% vs. 74.93% AUC, 4.96 vs. 6.60 forgetting), because routing among 5 adapters is easier (93.74% vs. 86.41% routing accuracy) and fewer samples are sent to a wrong adapter. With only 500 samples per domain the adapters are less specialized, so merging domains costs less than on the full training set, where removing s_{new} degrades both the GT and the routed results.

Table 18: **Ablation of the rank/threshold values.** CLIP+CLAP, multi-adapter model on 23 domains with GT routing. Average AUC and forgetting on the validation set. Left: LoRA rank r with $\tau = 0.9$; right: similarity threshold τ with $r = 64$.

r ($\tau = 0.9$)	8	16	32	64	128	256	512
AUC (%)↑	81.68	81.24	81.54	80.18	**82.15**	76.09	71.47
FORGET↓	**2.27**	3.33	3.40	3.80	2.36	3.04	9.71

τ ($r = 64$)	0.7	0.8	0.9	0.95
AUC (%)↑	**82.71**	82.32	80.18	81.81
FORGET↓	**2.04**	2.57	3.80	2.53

Table 19: **LoRA rank ablation with a single adapter.** A single adapter is trained incrementally on the 23 domains (CLIP+CLAP, 500 samples per domain). GT routing; average AUC and forgetting on the validation set.

RANK	AVG AUC (%) ↑	AVG FORGET ↓
$r = 16$	68.20	**4.36**
$r = 32$	69.61	5.59
$r = 64$	**72.33**	11.54

Table 20: Performance (AUC) of the Single Adapter under Joint Training across the 23 manipulation domains. Results are reported on the validation set.

Rank Value	$\mathcal{T}_0$	$\mathcal{T}_1$	$\mathcal{T}_2$	$\mathcal{T}_3$	$\mathcal{T}_4$	$\mathcal{T}_5$	$\mathcal{T}_6$	$\mathcal{T}_7$	$\mathcal{T}_8$	$\mathcal{T}_9$	$\mathcal{T}_{10}$	$\mathcal{T}_{11}$
r=64	66.88	43.21	96.06	42.50	35.88	47.91	63.20	65.93	59.75	70.16	52.61	52.61
r=128	78.72	43.85	39.01	42.49	39.01	49.55	69.62	69.25	56.32	73.01	52.58	60.25
r=256	88.81	46.41	96.66	43.17	42.75	51.01	75.63	74.36	54.11	72.25	60.95	70.09
r=512	50.63	43.71	95.88	42.60	40.01	47.97	58.83	56.29	56.63	68.18	52.78	51.16
r=1024	77.50	42.94	96.22	42.07	37.07	49.17	73.99	63.90	46.94	78.40	50.39	64.56

Rank Value	$\mathcal{T}_{12}$	$\mathcal{T}_{13}$	$\mathcal{T}_{14}$	$\mathcal{T}_{15}$	$\mathcal{T}_{16}$	$\mathcal{T}_{17}$	$\mathcal{T}_{18}$	$\mathcal{T}_{19}$	$\mathcal{T}_{20}$	$\mathcal{T}_{21}$	$\mathcal{T}_{22}$	$\mathcal{T}_{23}$	AVG
r=64	56.97	57.36	54.96	58.71	56.42	99.81	71.22	64.01	54.46	85.94	67.47	99.80	63.49
r=128	48.27	54.24	55.16	59.51	57.06	99.79	65.73	66.61	59.52	84.53	68.66	99.94	62.19
r=256	44.27	56.61	55.44	59.78	74.36	99.58	57.13	69.75	62.82	82.23	73.44	99.74	67.14
r=512	59.30	55.44	55.93	53.66	60.48	99.76	76.66	63.68	53.10	88.22	66.86	99.77	62.40
r=1024	46.66	53.77	51.30	53.89	60.23	99.98	66.11	68.20	55.75	85.04	69.37	99.97	63.89

Table 21: **Ablation of the key-query router capacity.** We compare two routing network capacities by varying the projection dimension (d_p) and hidden dimension (d_h) of the query projection MLP. Results are reported for routed inference and GT routing on the validation set; CLIP+CLAP trained on 500 samples per domain.

ROUTER	d_p	d_h	ROUTED		GT	
			AVG AUC ↑	ROUTING ACC ↑	AVG AUC ↑	ROUTING ACC ↑
SMALL CAPACITY	128	256	72.94	77.59	**78.99**	100.0
LARGE CAPACITY	256	512	**74.93**	**86.41**	78.58	100.0

Table 22: **Ablation of the compatibility score s_{new} of the new adapter.** W/o s_{new}: a new adapter is created only if $\max_i \cos(\mathbf{q}, \mathbf{k}_i) < \tau$, with $\tau = 0$; BRAVEn AV: full training set, test set, mean±std over 3 seeds. CLIP+CLAP: 500 samples/domain, validation set, seed 1708. Forgetting as in Table 5.

BACKBONE	SETTING	N_{ADP}	GT AUC ↑	GT FORGET ↓	ROUTED AUC ↑	ROUTED FORGET ↓	ROUTING ACC ↑
BRAVEN AV	DEFEND-KQ	11.3	**92.38 ± 0.73**	**1.42 ± 0.92**	**89.68 ± 0.90**	**4.24 ± 1.05**	77.88 ± 8.10
	W/O s_{NEW}	5.0	85.44 ± 3.17	8.51 ± 3.08	79.34 ± 4.51	14.87 ± 4.42	**86.66 ± 3.46**
CLIP+CLAP	DEFEND-KQ	9	78.58	**2.75**	74.93	6.60	86.41
	W/O s_{NEW}	5	76.25	4.41	**75.68**	**4.96**	**93.74**