PER-DOMAIN RESIDUAL ADAPTERS WITH ASYMMETRIC DEPTH FOR DOMAIN-INCREMENTAL AUDIO CLASSIFICATION

Technical Report

Aakash Divakar, Angad Ripudaman Singh Bajwa*, Atharva Anand Joshi**

Independent Researcher
Austin, USA
{aakashdivakar.4869, angadisthere, atharvajoshi253}@gmail.com

ABSTRACT

We describe our submission to DCASE 2026 Task 7, Domain-Incremental Learning (DIL) for audio classification. Starting from the released ADIL baseline, which adds per-domain Batch Normalization (BN) and a per-domain classifier to a shared, frozen CNN14 backbone, we identify the limited per-domain capacity as the main bottleneck for the harder domains. We add, on top of the frozen backbone, a stack of zero-initialised *residual bottleneck adapters* after every convolutional block. Because each domain owns an independent adapter stack, the depth of the stack can be set *per domain*: the more difficult domain (D3) receives a deeper stack than the easier one (D2). Combined with a warm-started per-domain classifier head and a supervised contrastive auxiliary loss, this raises the domain-agnostic average accuracy on the dev-test set from 52.5% to 68.5%, with D2 and D3 improving from $58.6\%/46.1\%$ to $76.2\%/60.8\%$. The shared backbone and the D1 behaviour are preserved bit-for-bit, so no previously learned domain is forgotten.

***Index Terms*—** Domain-incremental learning, continual learning, acoustic classification, residual adapters, catastrophic forgetting

1. INTRODUCTION

Domain-incremental learning (DIL) requires a single model to classify the same sound classes as recording conditions change over a sequence of domains $\mathcal{D}_1 \rightarrow \mathcal{D}_2 \rightarrow \mathcal{D}_3$, without access to previous domains' data and without forgetting them [1]: naive fine-tuning overwrites earlier knowledge (catastrophic forgetting), while freezing the network entirely prevents it from adapting (poor plasticity).

The DCASE 2026 Task 7 baseline addresses this trade-off with the ADIL approach [1]: a CNN14 [2] backbone is trained on $\mathcal{D}_1$ and then *frozen*, after which only a small set of domain-specific parameters – affine Batch-Normalization (BN) and a residual classifier correction – is learned per domain. Since domain identity is unknown at inference, the model forward-passes through every domain's layers and keeps the lowest-entropy prediction (domain-agnostic, ADIL).

This design forgets nothing, since earlier-domain parameters are never touched, but its plasticity is limited: a handful of affine BN parameters is often too little capacity to absorb a large distribution shift, as we observe on the two harder development domains. Our contribution is to *increase per-domain capacity* while keeping the forgetting-free property intact (Fig. 1): we (1) add per-domain residual bottleneck adapters after every convolutional block, (2) make the adapter depth *asymmetric*, giving the harder domain more capacity than the easier one, (3) replace the shared classifier with a warm-started per-domain head, and (4) tighten the per-class embedding clusters with a supervised contrastive loss. All additions are per-domain and zero/identity-initialised, so $\mathcal{D}_1$ is reproduced exactly and the domains remain mutually independent.

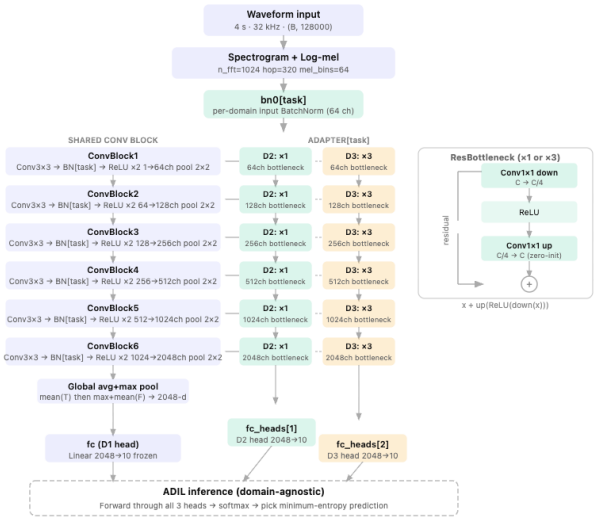

Figure 1: Architecture of the submitted system. A shared, frozen CNN14 backbone (purple) is augmented with per-domain components: an input and per-block Batch Normalization, a stack of zero-initialised residual bottleneck adapters after every convolutional block (green for D2, amber for D3, with asymmetric depth 1 vs. 3), and a per-domain linear head. At inference the input is routed through all domains and the minimum-entropy prediction is kept (ADIL). The inset details one residual bottleneck adapter.

2. METHOD

2.1. Backbone and per-domain Batch Normalization

We keep the baseline front-end and backbone unchanged. Audio is resampled to 32 kHz; a log-mel spectrogram with 64 mel bands

*The three authors contributed equally to this work.

is computed with a window of 1024 and a hop of 320 samples. The backbone is CNN14 [2]: six convolutional blocks producing $\{64, 128, 256, 512, 1024, 2048\}$ channels, each block being two 3×3 convolutions, each followed by BN and ReLU, and a 2×2 average pooling. Following the baseline, the convolutional weights are loaded from the released $\mathcal{D}_1$ checkpoint and *frozen for all domains*; only the BN layers are made domain-specific, i.e. each domain t owns its own BN parameters and statistics.

2.2. Per-domain residual bottleneck adapters

To give each domain more than the affine BN capacity, we insert a residual bottleneck adapter after each of the six convolutional blocks. For a feature map with C channels the adapter is

$$\mathrm{Adapter}(\boldsymbol{x}) = \boldsymbol{x} + W_{\text{up}}\, \sigma\big(W_{\text{down}}\, \boldsymbol{x}\big), \tag{1}$$

where W_{down} is a 1×1 convolution reducing the channels to $C/4$, σ is a ReLU, and W_{up} is a 1×1 convolution restoring C channels. W_{up} is *zero-initialised*, so at the start of training every adapter is the identity map and the pretrained features are left untouched; the adapter then learns a domain-specific feature shift on top of the frozen backbone. Domain $\mathcal{D}_1$ uses identity adapters, so its behaviour is reproduced bit-for-bit.

2.3. Asymmetric adapter depth

Because each domain owns an independent stack of adapters, the *number* of adapters stacked after each block can be chosen per domain. We stack d_t residual adapters for domain t:

$$\mathrm{Stack}_t(\boldsymbol{x}) = \big(\mathrm{Adapter}^{(d_t)} \circ \cdots \circ \mathrm{Adapter}^{(1)}\big)(\boldsymbol{x}). \tag{2}$$

Every adapter in the stack is zero-initialised, so a stack of any depth still starts as the identity. We set $d_{\mathrm{D2}} = 1$ and $d_{\mathrm{D3}} = 3$: the harder domain D3 receives three times the per-domain convolutional capacity of D2. Since the two stacks are completely independent, increasing D3's depth cannot affect D2.

2.4. Per-domain classifier head

The baseline reuses the frozen $\mathcal{D}_1$ classifier for all domains. We instead give every domain $t > 0$ its own linear head $\boldsymbol{W}_t \in R^{C \times K}$ ($K = 10$ classes), warm-started from the $\mathcal{D}_1$ classifier at checkpoint-load time. $\mathcal{D}_1$ keeps the original head. The head is trained jointly with the domain's BN and adapters.

2.5. Training objective

Each domain is trained with cross-entropy plus a supervised contrastive auxiliary loss [3] applied to the L2-normalised global embedding $\boldsymbol{z}$:

$$\mathcal{L}_{\text{con}} = \sum_i \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(\boldsymbol{z}_i \cdot \boldsymbol{z}_p / \tau)}{\sum_{a \neq i} \exp(\boldsymbol{z}_i \cdot \boldsymbol{z}_a / \tau)}, \tag{3}$$

where $P(i)$ is the set of samples sharing i's class. The total loss is $\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda\, \mathcal{L}_{\text{con}}$ with $\lambda = 0.2$ and $\tau = 0.07$, which pulls same-class embeddings together and pushes different-class embeddings apart, sharpening the per-domain decision boundaries.

2.6. Waveform-level data augmentation

We additionally evaluated a training-only waveform augmentation pipeline; the validation and test loaders remain deterministic and use the original audio. Each recording is split into non-overlapping 4 s segments, and the dataset keeps one unmodified view plus multiple augmented views of every segment (class label preserved). Each augmented view applies a random composition of conservative transforms, after DC-offset removal and crop/zero-pad to the fixed clip length:

- **Temporal:** zero-filled time shift (p=0.75, max 0.75 s); time stretching (p=0.30); speed perturbation (p=0.30); fade-in/out (p=0.25, max 0.25 s).
- **Class-aware pitch/tempo:** pitch shifting (p=0.25) with class-specific ranges, narrower for tonal/speaker-like classes (piano, speech) than texture-like classes (engine, fire, footsteps, knocking), to avoid label drift.
- **Spectral/noise:** random filtering (p=0.35); colored-noise mixing (p=0.45, SNR 18–35 dB).
- **Label-preserving interference:** a low-level distractor from another (preferably different-class) item mixed at 8–20 dB SNR, simulating background interference while keeping the target label dominant.
- **Local corruption/dynamic range:** temporal masking (p=0.30, $\leq$ 2 masks, $\leq$ 8% of clip); gain perturbation (p=0.85, $[-8, 8]$ dB); mild soft clipping (p=0.20).

The waveform is then re-cropped/padded and peak-normalised only if its peak exceeds one, keeping augmented samples in range without altering already valid clips.

2.7. Entropy-Based Domain Routing

The simplest routing strategy requires no additional training. For each test sample we compute class probabilities under each domain head, $\mathbf{p}_t = \mathrm{softmax}(\mathbf{l}_t)$, and select the domain with minimum predictive entropy [1]:

$$t^* = \arg\min_t \; H_t, \qquad H_t = -\sum_c p_{tc} \log p_{tc}. \tag{4}$$

A well-matched domain head produces a peakier, lower-entropy distribution than a mismatched one, so t^* acts as an implicit domain detector. The method is fully unsupervised at inference time: no domain labels are required and the router adds zero parameters.

2.8. Mahalanobis Distance Router

Rather than routing via classifier confidence, we explicitly model the distribution of each domain in the D1-BN feature space, following the framework of Lee et al. [4]. The D1-BN embedding $\mathbf{z} = g_0(\mathbf{x}) \in R^{2048}$ is the Global Average Pooled (GAP) output of Conv Block 6 computed with the D1 BatchNorm statistics frozen; it forms a fixed coordinate system in which D2 and D3 occupy characteristic regions.

Training phase. We extract $\mathbf{z}$ for all D2 and D3 training samples and compute domain means and covariances using streaming (online) statistics to avoid materialising the full $N \times 2048$ matrix in

memory:

$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_i \mathbf{z}_i^{(k)},$$
$$\boldsymbol{\Sigma}_k = \frac{1}{N_k - 1}\sum_i \big(\mathbf{z}_i^{(k)} - \boldsymbol{\mu}_k\big)\big(\mathbf{z}_i^{(k)} - \boldsymbol{\mu}_k\big)^\top + \lambda I, \tag{5}$$

where $\lambda = 10^{-4}$ is a diagonal regulariser. A diagonal covariance approximation ($\boldsymbol{\Sigma}_k \leftarrow \mathrm{diag}(\boldsymbol{\Sigma}_k)$) is used in practice for memory efficiency.

Inference phase. Given a test embedding $\mathbf{z}$, the Mahalanobis distance to each domain is:

$$d_k(\mathbf{z}) = \sqrt{(\mathbf{z} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{z} - \boldsymbol{\mu}_k)}, \tag{6}$$

and routing follows:

$$t^* = \begin{cases} D1 & \text{if } \min(d_{D2},\, d_{D3}) > \tau, \\ \arg\min_k \; d_k & \text{otherwise.} \end{cases} \tag{7}$$

Samples whose nearest-domain distance exceeds the rejection threshold τ are considered out-of-distribution for D2 and D3 and routed to the D1 head. The threshold was calibrated against the dev-test split ($\tau = 60$ in all reported results).

2.9. GAP-MLP Router

The third approach trains a lightweight Multi-Layer Perceptron (MLP) to discriminate domains directly, in the spirit of mixture-of-experts gating networks [5, 6]. Features are extracted by hooking Conv Block 6 and concatenating the domain-specific GAP activations from *all three* domain heads, giving a $3 \times 2048 = 6144$-dimensional descriptor that encodes how each domain's forward pass responds to the input.

The MLP has architecture Linear($6144 \to 256$) $\to$ ReLU $\to$ Linear($256 \to 2$), outputting logits for $\{D2,\ D3\}$. It is trained with cross-entropy loss on the labeled development training split using Adam for 5 epochs (lr $= 10^{-3}$). At inference the predicted domain selects the corresponding classifier head; if $\max(\mathrm{softmax}) < \tau_{\mathrm{MLP}}$ the sample is rejected to D1.

2.10. Inference

Inference is domain-agnostic and identical to the baseline ADIL rule: the input is forward-passed through every seen domain's specific layers, and the domain with the minimum prediction entropy is selected to produce the final class label.

3. EXPERIMENTS AND RESULTS

3.1. Setup

All systems are trained on the Task 7 development set, one domain at a time, starting from the released $\mathcal{D}_1$ checkpoint. We use Adam with learning rate 10^{-3}, batch size 32, and a CosineAnnealingLR schedule decaying to 10^{-7}. Only the current domain's parameters (BN, adapters, head) are updated at each step. We report the domain-agnostic accuracy on D2 and D3, and their average, after the full $\mathcal{D}_1 \to \mathcal{D}_2 \to \mathcal{D}_3$ sequence.

For the augmentation experiment, the learning rate was 10^{-3}, batch size 32, and 240 epochs. Boundary-aware regularisation was disabled ($\lambda_{\mathrm{reg}} = 0.0$), supervised contrastive learning used $\lambda_{\mathrm{con}} = 0.2$ and temperature $\tau = 0.07$, the asymmetric adapter depths were $(1, 3)$ for D2 and D3, waveform augmentation was enabled, class-balanced sampling was disabled, and test-time augmentation was disabled.

Table 1: Domain-agnostic accuracy (%) on the dev-test split, adding one component at a time. The last row is the submitted system.

System	D2	D3	Avg
Per-domain BN (baseline) [1]	59.2	40.8	50.0
+ residual adapters	73.9	53.5	63.7
+ per-domain head	77.3	54.0	65.6
+ supervised contrastive loss	77.9	55.1	66.5
+ asymmetric depth $(1, 3)$	77.2	57.7	67.4
+ longer training (180 ep.)	**76.2**	**60.8**	**68.5**

Table 2: Per-domain adapter family, same backbone and protocol (residual bottleneck selected).

Adapter family	D2	D3	Avg
FiLM (affine per channel)	56.7	38.8	47.7
Squeeze-and-excitation	70.3	45.0	57.7
Squeeze-and-excitation (deeper)	69.2	41.8	55.5
Residual bottleneck (*ours*)	**73.9**	**53.5**	**63.7**

3.2. Ablation

Table 1 adds one component at a time. Residual adapters give the largest single jump (+13.7 average over the BN baseline), confirming that per-domain capacity, not the classifier, is the bottleneck. The per-domain head and the contrastive loss mainly help D2, while the asymmetric depth and longer training are what move the harder D3 domain (from 55.1% to 60.8%). The submitted system reaches 68.5% average, an 18.5 percentage point improvement over the reproduced baseline, while preserving D1 exactly.

3.3. Choosing the adapter

Before settling on residual bottleneck adapters we compared three per-domain adapter families inserted at the same six locations, each trained under the identical protocol (Table 2). Feature-wise linear modulation (FiLM), which only scales and shifts each channel, actually performed *below* the BN baseline: a per-channel affine transform is too weak to absorb the distribution shift. Squeeze-and-excitation (SE) channel gating helped (57.7%), but stacking more SE blocks did not. The residual bottleneck, which can mix channels through its 1×1 convolutions, was clearly the strongest (63.7%) and is the one we keep. The gap is largest on the harder D3 domain ($38.8 \to 53.5\%$), confirming that the bottleneck's extra capacity is what the difficult domain needs.

3.4. Domain Routing Strategies

Table 3 compares all three routing strategies on top of the best single model (bn_res_deep_clf, asymmetric depth $1, 3$, 180 epochs). All numbers are on the dev-test split (D2: 639 samples, D3: 806 samples).

Table 3: Routing strategy comparison on the dev-test split.

Router	D2	D3	Avg
Entropy (no threshold)	**76.2**	60.8	68.5
Entropy + ensemble ($\times 2$)	74.0	50.50	60.9
Mahal. (no threshold)	70.9	**62.5**	66.7
Mahal. ($\tau = 60$, diag)	68.5	57.7	63.8
GAP-MLP (3 ep., $\tau_{\text{MLP}} = 0.5$)	76.5	62.0	**69.3**

Table 4: Waveform augmentation result

System	D2	D3	Avg
waveform augmentation	75.4	51.4	63.4

Entropy routing needs no extra training: the supervised contrastive loss calibrates per-domain heads, yielding separable entropy values across domains, similar to confidence estimation in deep ensembles [7].

Mahalanobis routing instead models each domain's feature distribution explicitly [4]: without a threshold it matches entropy routing (66.7%) while reaching 73.2% D2↔D3 routing accuracy, confirming D2/D3 occupy separable but overlapping regions in D1-BN space ($d_{D2} \approx 42.0$, $d_{D3} \approx 44.0$). With $\tau = 60$, 22.8% of samples are rejected to D1, hurting accuracy – a single global threshold is too coarse here.

GAP-MLP routing concatenates GAP features from all three heads, in the spirit of mixture-of-experts gating [5]: the best epoch (3 of 5) reaches 69.28% average accuracy but overfits quickly (epoch 5 drops D2 to 74.96%, average to 68.93%). On the unlabeled evaluation set it assigns just 0.6% of samples to D1 (61.0%/38.5% D2/D3), showing $\tau_{\text{MLP}} = 0.5$ rarely rejects out-of-distribution inputs – evidence of memorising the dev-train distribution.

Key finding. Entropy routing remains the strongest single-model router, since contrastive training already calibrates per-domain heads well [7]. Learned routers (MLP, Mahalanobis) offer interpretable domain representations and explicit D1 rejection but need careful threshold tuning to avoid overfitting and excessive permissiveness.

3.5. Effect of waveform augmentation

Table 4 reports the result of enabling the waveform-level augmentation pipeline described in Section 2.6 for the model. The augmented model was trained for 240 epochs with supervised contrastive weight 0.2, contrastive temperature 0.07, and asymmetric adapter depths $(1, 3)$.

The augmented run remains strong on D2 but loses accuracy on D3 relative to the best non-augmented asymmetric-depth system. This suggests that the current augmentation distribution may be too broad for the hardest domain: while the perturbations improve diversity, aggressive temporal warping, pitch changes, filtering, and distractor mixing can also move some samples away from the D3 domain characteristics that the adapter stack needs to specialise for. We therefore treat this pipeline as a robustness-oriented ablation rather than the final selected configuration.

Table 5: Partial backbone unfreezing for D3. No configuration beats the frozen adapter system; the shared weights couple D2 and D3.

Configuration	D2	D3	Avg
Frozen adapters (submitted)	**76.2**	60.8	**68.5**
Unfreeze 2 blk, no anchor	58.5	**66.4**	62.5
Unfreeze 2 blk, ℓ_2 anchor	75.3	57.3	66.3
Unfreeze 1 blk, ℓ_2 anchor	72.6	57.7	65.2
D3-only unfreeze 2 blk	61.5	62.3	61.9
D3-only unfreeze 3 blk	46.3	54.3	50.3

3.6. What helped and what did not

Asymmetric beats symmetric depth. Giving *both* domains depth-2 stacks scored 65.1%, whereas the asymmetric $(1, 3)$ allocation reached 67.4% at a similar parameter budget: the capacity is better spent on the domain that actually needs it (D3) than spread evenly.

Contrastive weight. Sweeping the SupCon weight $\lambda \in \{0.1, 0.15, 0.2, 0.25, 0.3\}$ peaked at $\lambda = 0.2$ (66.7%); larger values began to compete with the cross-entropy term and slightly hurt accuracy.

Augmentation was selected for the robustness The stronger waveform augmentation pipeline reached 75.4% on D2, 51.4% on D3, and 63.4% average for the run. Although this is well above the reproduced BN baseline, it is below the best non-augmented asymmetric-depth model because D3 degrades substantially. We therefore keep augmentation as an ablation/robustness experiment. Test-time augmentation and checkpoint ensembling were likewise within noise of the single frozen model and were dropped to keep the submission simple.

3.7. Frozen backbone vs. partial unfreezing

The most tempting way to gain D3 accuracy is to unfreeze the shared backbone's top convolutional blocks (Table 5). Unfreezing two blocks pushes D3 to 66.4% — our best D3 result — but collapses D2 to 58.5%, since the backbone is shared and D2's BN/adapters were tuned for the original features. An ℓ_2 anchor toward the previous checkpoint recovers D2 but cancels most of the D3 gain, and restricting unfreezing to D3's own forward pass still leaves D2 degraded; unfreezing three blocks destabilises training entirely. No unfrozen configuration beats the fully-frozen adapter system on average, so we keep the backbone frozen and rely on per-domain adapters, which add plasticity without cross-domain interference.

4. CONCLUSION

We presented a forgetting-free extension of the DCASE 2026 Task 7 baseline that increases per-domain capacity through zero-initialised residual adapters, with an asymmetric depth that gives the harder domain more capacity, a warm-started per-domain classifier, and a supervised contrastive loss. The system raises the domain-agnostic dev-test average from 52.5% to 68.5% while reproducing $\mathcal{D}_1$ exactly and keeping the domains independent. Closing the remaining gap on the hardest domain, ideally without the cross-domain interference seen when unfreezing the shared backbone, is left for future work.

5. REFERENCES

[1] M. Mulimani and A. Mesaros, "Domain-incremental learning for audio classification," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2025, pp. 1–5.

[2] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "PANNs: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 28, pp. 2880–2894, 2020.

[3] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[4] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[5] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, no. 1, pp. 79–87, 1991.

[6] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *Journal of Machine Learning Research*, vol. 23, pp. 1–39, 2022.

[7] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.