Domain-Calibrated Routing and Parameter-Isolated Training for Domain-Agnostic Incremental Audio Classification

Qianqian Li *Guoqing Chen* *Yanxiong Li*[*]

School of Electronic and Information Engineering, South China University of Technology, Guangzhou, China, 1210517956@qq.com, 1695390037@qq.com, eeyxli@scut.edu.cn

ABSTRACT

This technical report describes three systems to DCASE 2026 Task 7, Domain-Agnostic Incremental Learning for Audio Classification. The task requires sequential adaptation from the released D1 baseline to D2 and D3 without revisiting previous-domain audio and without using external data or pretrained models. Systems 1 and 2 share an Mutil-Cnn14 model with domain-specific batch normalization, squeeze-and-excitation attention, residual adapters, and domain-specific classifier heads; they differ only in deterministic inference calibration. System 3 uses a late-private high-level branch architecture, where D1 retains the original high-level path while D2/D3 use private block5/block6 modules and domain classifiers. Its inference uses Prototype-Likelihood Routing (PLR), which compares test features with saved D2/D3 prototype statistics and falls back to entropy routing when the prototype match is unreliable. System 2 gives the best D2/D3 mean (66.04%), whereas system 1 gives the best D3 score (61.76%). The report focuses on the modeling choices, routing policies, rule compliance, reproducibility, and development-set behavior of the three submitted systems.

Index Terms— domain-incremental learning, audio classification, catastrophic forgetting, domain-specific batch normalization, prototype-likelihood routing

1. INTRODUCTION

Continual learning aims to adapt a model to a non-stationary stream while retaining knowledge that was useful for previous tasks. In many class-incremental benchmarks, the label set expands over time[1-5]. DCASE 2026 Task 7 [6] instead keeps the sound-class and changes the acoustic domain sequentially. This setting makes the final decision depend on two coupled problems: learning a classifier for each revealed domain and deciding, at evaluation time, which domain branch should be trusted for an unlabeled audio file.

The organizer's baseline follows a domain-incremental design in which convolutional filters are shared and domain-specific Batch-Normalization (BN) layers absorb distribution shifts. This design is stable and parameter efficient, but our development experiments showed two practical limitations. First, BN-only adaptation may be insufficient when high-level discriminative cues shift across domains. Second, uncertainty routing can select a wrong branch that is nevertheless confident, causing domain-selection errors that dominate within-domain classification errors.

We therefore submit three complementary systems, as illustrated in Figure 1, where Systems 1 and 2 improve inference behavior through calibrated routing rules, while System 3 modifies the architecture with private high-level branches.

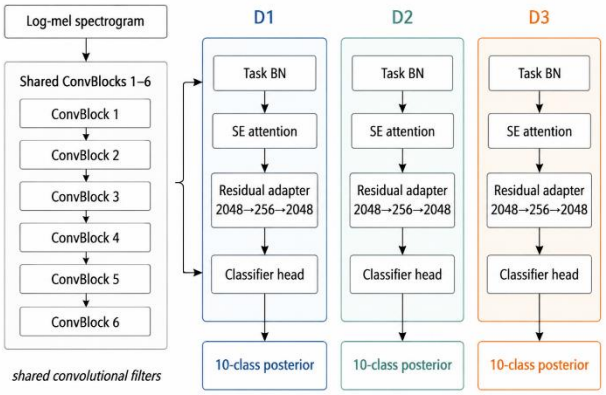

Figure 1. The framework of the systems 1, 2

The overall design of System 3 is further detailed in Figure 2, which shows the separation between shared low-level convolutional layers and domain-specific high-level modules.

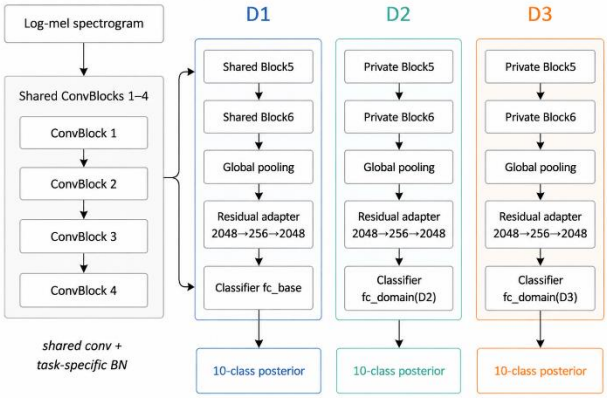

Figure 2. The framework of the system 3

2. TASK SETUP AND CONSTRAINTS

The dataset of the task contains ten target classes: alarm, baby_cry, bark, engine, fire, footsteps, knock, telephone_ringing, piano, and

[*] Corresponding author

speech. The D1 knowledge is embedded in the released baseline model. The development set provides D2 and D3 data, with class and domain labels for training and validation. The evaluation set contains audio from all three domains but provides only audio files. Therefore, the submitted system must classify each test file without knowing whether it belongs to D1, D2, or D3.

We explicitly follow the task constraints. In Step 2, the systems load only D2 training data; and in Step 3, they load only D3 training data. Previous-domain waveforms are not replayed, future-domain data is not accessed, and no external data, pretrained embeddings, or pretrained models are used. The word CNN14 in this report denotes an architecture family aligned with the baseline; it does not mean that PANNs pretrained weights are imported. Any statistics used by the methods, such as BN running statistics or PLR prototypes, are computed only from the corresponding permitted training domain.

The official ranking metric is domain-balanced. Accuracy is computed separately for each domain, domain accuracy is calculated as the mean of the class-wise accuracies within that domain, and the final score averages D1, D2, and D3 with equal weight. This motivates conservative routing: over-optimizing for the larger or easier domain can hurt the final score if another domain is under-selected.

3. BASELINE MOTIVATION AND METHOD OVERVIEW

The baseline architecture uses six convolutional blocks with channel sizes 64, 128, 256, 512, 1024, and 2048. Each block contains two 3 x 3 convolutional layers followed by BN, ReLU, average pooling, and dropout, whose structure is similar to the convolutional blocks in [7]. The front-end computes log-mel energies from 4 s, 32 kHz waveforms using a 1024-sample Hamming window, a 320-sample hop, 64 mel bands, and a 50 Hz to 14 kHz frequency range. Global pooling converts the final feature map into a fixed-length clip embedding.

Table 1: High-level comparison of the submitted systems

Aspect	Systems 1/2	System 3
Shared part	CNN14 convolutional filters	blocks 1-4 conv filters
Private part	Batch-Normalization, Squeeze-and-Excitation attention, adapter, classifier head	Batch-Normalization + private blocks 5/6 + classifier
Training focus	stable current-domain adaptation	high-level domain-specific semantics
Routing focus	confidence calibration and gates	prototype likelihood + entropy fallback

The central design is illustrated in Table 1, which compares the structural differences between the three systems. The incremental adaptation process is shown in Figure 3, where we highlight the frozen and trainable components across D2 and D3 stages. A fully shared backbone reduces forgetting but can underfit new domains. Full fine-tuning or full private copies increase capacity but are risky under limited current-domain data and no old-domain replay. Our submitted systems occupy two intermediate points. Systems 1/2 add lightweight domain modules to a mostly shared model, whereas System 3 privatizes only the last two convolutional blocks, where the representation is closest to semantic class boundaries.

4. ARCHITECTURE OF OUR SYSTEMS

Systems 1 and 2 use an Mutil-Cnn14 model. The trunk is based on the six-block CNN structure and produces a 2048-dimensional embedding by summing temporal max pooling and temporal average pooling. In every domain branch, task-specific BN handles low-order distribution shift, squeeze-and-excitation [8] (SE) attention reweights feature channels, a residual bottleneck adapter performs parameter-efficient feature correction, and a domain-specific classifier head produces a 10-class posterior.

The residual adapter adopts a bottleneck architecture with a down-projection from a 2048-dimensional feature space to a 256-dimensional latent representation, followed by an up-projection back to 2048 dimensions. It is initialized to approximate an identity mapping, ensuring that the initial behavior remains close to the released baseline while enabling lightweight high-level feature refinement for the target domain. The SE module is also trained per domain, so the branch can emphasize different acoustic channels without changing the shared convolutional filters.

During incremental adaptation, shared convolutional filters are frozen. For D2, only D2 BN affine parameters, D2 BN statistics, D2 SE parameters, D2 adapter parameters, and the D2 classifier head are updated. For D3, the same rule is applied to the D3 branch. Previous-domain branches are not placed in training mode; this prevents auxiliary forward passes from corrupting BN running means and variances for older domains. Systems 1 and 2 therefore have identical acoustic checkpoints and differ only in inference-time routing policy.

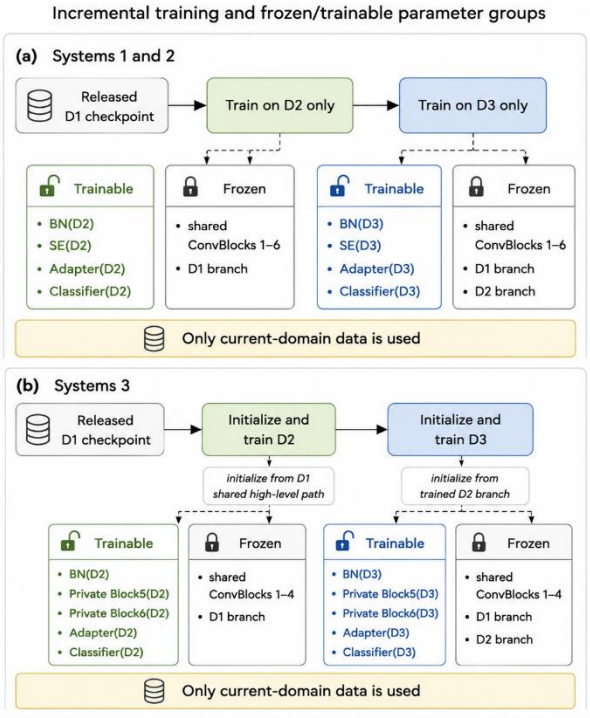

Figure 3: The incremental training flow

System 3 is built around a late-private high-level branch. The first four convolutional blocks remain shared across domains

because they mainly encode generic local time-frequency patterns, such as onsets, energy edges, and stable frequency-band structures. These layers still contain task-specific BN, allowing D1, D2, and D3 to maintain separate normalization statistics. The last two blocks are closer to class semantics and are therefore allowed to become domain-private for D2 and D3.

Concretely, D1 uses the original shared conv_block5, conv_block6, and fc_base. D2 and D3 use private_conv_block5, private_conv_block6, and fc_domain. The private convolutional blocks keep the same topology as the corresponding baseline blocks but have private convolution and BN parameters. This gives the new domains high-level feature-reconstruction capacity without copying the entire network or changing the D1 high-level path.

Initialization is important because D2/D3 data are limited. Before D2 training, the D2 BN parameters are copied from D1, private_conv_block5/6 are initialized from D1 shared block5/6, and fc_domain 1 is initialized from fc_base. Before D3 training, the D3 BN parameters, private block5/6, and fc_domain 2 are copied from the trained D2 branch. Thus each new domain begins from a meaningful acoustic representation rather than from random high-level filters.

Training is deliberately conservative. The model first freezes all parameters, then unfreezes only the current task BN layers, current private block5/6, and current domain classifier. The D1 path, old-domain private branches, old-domain classifiers, and shared low-level convolutional filters remain frozen. The training loss is standard cross-entropy on the active domain branch. This differs from replay-based continual learning because no previous-domain audio examples are stored or revisited.

5. ROUTING FOR OUR SYSTEMS

Because evaluation files do not contain domain labels, each system must infer the branch at test time. Systems 1 and 2 evaluate all visible domain branches and use test-time augmentation by circularly shifting each waveform by 0.0, 0.25, and 0.5 of the clip length. For every shifted view, each branch returns a class posterior. The shift-averaged posterior is then processed by a deterministic routing policy fixed before evaluation.

The inference-time decision process is illustrated in Figure 4, where entropy-based routing, bias correction, and class-specific gating are combined into a unified policy.

The base routing score is derived from prediction uncertainty. For each branch t, the class posterior is denoted as p_t. The entropy is used to measure the uncertainty of the prediction and is defined as follows:

$$H_t = -\sum p_t(y) \log p_t(y) \quad (1)$$

Where H_t denotes the entropy of branch t, and y is the class index.

Based on the entropy, a soft top-2 routing strategy is adopted. The routing score of each branch is computed as:

$$S_t = -\frac{H_t}{T} + b_t \quad (2)$$

Where T is a temperature coefficient controlling the sharpness of routing decisions, and b_t is a predefined domain prior bias that does not depend on evaluation data. The top-2 branches with the highest routing scores are selected, and their predictions are fused through a normalized weighting scheme:

$$P_{route} = \alpha_1 \cdot p_1 + \alpha_2 \cdot p_2 \quad (3)$$

where the weights α_1 and α_2 are computed as:

$$\alpha_t = \frac{e^{S_t}}{e^{S_1} + e^{S_2}} \quad (4)$$

System 1 adopts D1-aware soft routing with a temperature coefficient $T = 2.0$ and a domain prior bias vector (1.2, 0.0, 0.0) for (D1, D2, D3), respectively. The initially routed posterior is further interpolated with the D2 branch prediction using a fixed weighting factor of 0.40 to ensure stable performance for D2 samples.

If the maximum posterior probability of the D2 branch exceeds a confidence threshold of 0.50, the D2 prediction directly replaces the routed output. In addition, a D3 expert gate is activated when its class-specific confidence exceeds a predefined threshold, allowing it to override the current prediction. Finally, a D1 entropy-based fallback mechanism is applied to recover D1 predictions when the D1 branch exhibits sufficiently low uncertainty.

System 2 starts from zero-bias base routing with the same D2 protection threshold and D3 thresholds. It then evaluates an internal alternative branch with D1-biased routing. If the base predicted class is one of alarm, baby_cry, fire, or footsteps, the output is replaced by the D1-biased alternative. This class-conditional rule was chosen because development errors were not uniformly distributed across classes; several classes benefited from stronger D1 protection, whereas other classes were harmed by it.

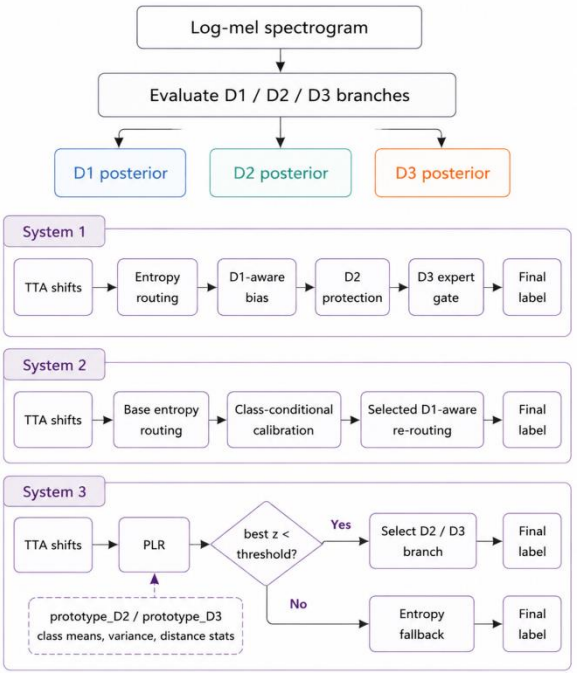

Figure 4. The pipeline of the inference method.

System 3 introduces Prototype-Likelihood Routing (PLR) to improve robustness against low-entropy misclassification. After training on D2 and D3, the system extracts feature embeddings from each domain branch and computes class prototypes by averaging embeddings within each class.

A shared variance term is estimated from the within-class feature variations to normalize feature distances. Based on these statistics, the system measures how close a test sample is to each domain's prototype distribution by computing its minimum normalized distance to all class prototypes.

The resulting distance is then standardized using the mean and variance of training distances, producing a PLR score that reflects

how well the sample fits the learned distribution of each domain. A smaller score indicates a better match.

The system compares only the PLR scores of D2 and D3 branches. If the best score is below a predefined threshold τ, the corresponding branch prediction is selected. Otherwise, the system falls back to entropy-based routing across all available branches to handle cases where prototype statistics are unreliable or unavailable (e.g., D1).

Variable-length evaluation files are handled by fixed-length windows. Short audio is zero-padded, long audio is processed by sliding windows with tail coverage, and window-level class posteriors are averaged. The final output is the class with the maximum file-level posterior. This procedure preserves the task rule that each test sample is classified independently.

5. DEVELOPMENT RESULTS AND ANALYSIS

Table 2 reports the results of our systems.

Table 2: The results of our systems

System	Main difference	D2	D3	Mean
Baseline	-	59.0	46.1	52.5
System.1	D1-aware routing + D2/D3 gates	69.70	61.76	65.73
System.2	class-conditional calibration	71.24	60.84	66.04
S.3	late-private branch + PLR	65.54	55.48	60.51

For System 1, the Step 2 results on Domain 2 (development set) are as follows: alarm (75.38%), dog (86.96%), engine (66.67%), fire (63.89%), footsteps (83.67%), knock (72.22%), piano (60.82%), and speech (85.43%), achieving an average accuracy of 74.38%.

After adaptation to Domain 3 (Step 3), System 1 achieves the following results on Domain 2: alarm (62.31%), dog_bark (91.30%), engine (57.97%), fire (55.56%), footsteps (85.71%), knock (58.83%), piano (65.98%), and speech (80.40%), with an average accuracy of 69.70%. On Domain 3, the class-wise accuracies are: alarm (79.03%), baby_cry (50.00%), dog_bark (58.44%), engine (74.12%), fire (40.54%), footsteps (52.24%), telephone_ringing (70.97%), piano (95.89%), and speech (34.63%), yielding an average accuracy of 61.76%.

These results indicate that performance degradation under domain shift is highly class-dependent. In particular, classes such as fire, engine, and telephone_ringing suffer significantly larger drops, while relatively stable classes such as piano and speech remain robust. This observation motivates class-aware adaptation in System 2 and prototype-based distance modeling in System 3.

For System 2, the Step 2 performance on Domain 2 is identical to System 1, with an average accuracy of 74.38%.

After adaptation to Domain 3 (Step 3), System 2 achieves an average accuracy of 71.24% on Domain 2, with class-wise results: alarm (56.15%), dog_bark (91.30%), engine (68.12%), fire (55.56%), footsteps (85.71%), knock (66.67%), piano (65.98%), and speech (80.40%). On Domain 3, System 2 achieves an average accuracy of 60.84%, with class-wise results: alarm (75.81%), baby_cry (50.00%), dog_bark (58.44%), engine (77.65%), fire (40.54%), footsteps (43.28%), telephone_ringing (70.97%), piano (95.89%), and speech (34.98%).

Overall, System 2 consistently improves or maintains performance compared to System 1, particularly on Domain 2 after full adaptation. This suggests that removing strong D1 bias during routing improves generalization across most classes, while class-specific D1-biased corrections are beneficial only for a limited subset of categories.

For System 3, the Step 2 results on Domain 2 are: alarm (71.07%), dog_bark (82.61%), engine (75.36%), fire (38.89%), footsteps (79.59%), knock (80.56%), piano (63.92%), and speech (83.33%), with an average accuracy of 71.92%.

After adaptation to Domain 3 (Step 3), System 3 achieves an average accuracy of 65.54% on Domain 2, with class-wise results: alarm (57.98%), dog_bark (91.30%), engine (39.13%), fire (27.78%), footsteps (75.51%), knock (80.56%), piano (63.92%), and speech (88.14%). On Domain 3, System 3 achieves an average accuracy of 55.48%, with class-wise results: alarm (66.13%), baby_cry (41.67%), dog_bark (50.65%), engine (71.76%), fire (35.14%), footsteps (54.48%), telephone_ringing (38.71%), piano (87.67%), and speech (53.19%).

These results demonstrate that System 3 exhibits stronger domain-specific separation but also a more pronounced performance drop under domain shift. This supports the hypothesis that explicitly modeling domain-specific feature distributions enables more structured adaptation, although it may reduce robustness in certain shared categories.

6 REPRODUCIBILITY AND SYSTEM FILES

Each system package contains an output TSV file, metadata YAML, model definition, and state-dictionary files for the incremental steps. The output file has no header and contains one filename and one predicted sound class per row. The model file implements load_model, which returns a model loaded with the state dictionary corresponding to the requested incremental step. The connection between the submission label, YAML metadata, output file, and model files is kept explicit.

Systems 1 and 2 are reproduced from the same acoustic checkpoint family. Their difference is the deterministic policy implemented in the inference code: System 1 corresponds to D1-aware routing with D2 protection and a D3 expert gate; System 2 corresponds to base routing with internal class-conditional calibration. System 3 is reproduced with checkpoint_D3.pth, prototype_D2.pth, prototype_D3.pth, and route_task_by_plr. The prototype files contain only class means, valid masks, diagonal variance, and training-distance summary statistics from the permitted D2/D3 training domains.

7. DISCUSSION

The three systems explore complementary strategies for domain-agnostic audio classification. Systems 1 and 2 rely on sufficiently expressive acoustic representations and focus on improving performance through deterministic routing and conservative adaptation. System 3 instead introduces domain-specific private branches to enhance high-level feature adaptation for domain-sensitive semantics.

A main limitation is that routing decisions depend on heuristically tuned thresholds derived from development behavior, although all rules are fixed before evaluation without using test annotations or statistics. A more principled solution would learn a lightweight router or model domain densities under the same data constraints.

Overall, the systems emphasize constraint-compliant incremental learning, branch-level protection, and transparent routing, while avoiding replay and external pretraining. Performance is often limited more by domain selection than by classifier capacity.

8. REFERENCES

[1] W. Xie, Y. Li, Q. He, and W. Cao, "Few-shot class-incremental audio classification via discriminative prototype learning," Expert Systems With Applications, 2023, vol. 225, 120044, pp. 1-13.

[2] Y. Li, W. Cao, W. Xie, J. Li and E. Benetos, "Few-Shot Class-Incremental Audio Classification Using Dynamically Expanded Classifier With Self-Attention Modified Prototypes," in IEEE Transactions on Multimedia, vol. 26, pp. 1346-1360, 2024, doi: 10.1109/TMM.2023.3280011.

[3] Y. Li, J. Li, Y. Si, J. Tan and Q. He, "Few-Shot Class-Incremental Audio Classification With Adaptive Mitigation of Forgetting and Overfitting," in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 32, pp. 2297-2311, 2024, doi: 10.1109/TASLP.2024.3385287.

[4] Y. Si, Y. Li, J. Tan, G. Chen, Q. Li and M. Russo, "Fully Few-Shot Class-Incremental Audio Classification With Adaptive Improvement of Stability and Plasticity," in IEEE Transactions on Audio, Speech and Language Processing, vol. 33, pp. 418-433, 2025, doi: 10.1109/TASLPRO.2025.3527147.

[5] Y. Li, W. Cao, J. Tan, Q. Li and G. Chen, "Few-Shot Class-Incremental Audio Classification Using Pseudo-Incrementally Trained Embedding Learner and Continually Updated Stochastic Classifier," in IEEE Transactions on Audio, Speech and Language Processing, vol. 33, pp. 3880-3895, 2025, doi: 10.1109/TASLPRO.2025.3610050.

[6] Manjunath Mulimani and Annamaria Mesaros. *Domain-incremental learning for audio classification.* In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2025.

[7] Y. Si, Y. Li, S. Huang, and B. Liu, "Cross domain few-shot class-incremental audio classification Via adversarial contrastive learning," in Proc. of Interspeech, 2026.

[8] J. Hu, L. Shen, and G. Sun, "Squeeze-and-Excitation Networks," in Proc. IEEE CVPR, pp. 7132-7141, 2018.