DCASE 2026 TASK 7 SUBMISSION: ALIGNED LOGIT STACKING AND LOW-RANK ADAPTER EXPERTS

Technical Report

Yacine Bel-Hadj, Ricardo Fracasso, Jan Van Rompaey

Department of Mechanical Engineering
Vrije Universiteit Brussel
Brussels, Belgium
yacine.bel-hadj@vub.be

ABSTRACT

This technical report describes the VUB systems submitted to DCASE 2026 Task 7, domain-agnostic incremental learning for audio classification. We start from the official CNN14-style baseline with domain-specific batch-normalization branches. At inference time the domain label is unavailable, so the baseline evaluates all branches and selects the prediction with the lowest entropy. Our submissions keep the same branch structure but replace this fixed selection rule with a learned decision stage. BelHadj_VUB_task7_1 trains an aligned logit stacker on branch logits, posterior probabilities, confidence measures, and branch-disagreement statistics. BelHadj_VUB_task7_2 first adds task-specific low-rank convolutional adapters to the backbone, then trains the same type of branch-level stacker on the adapter-enhanced logits. The two systems therefore test two related strategies: learning a better domain-agnostic decision rule and adding parameter-efficient domain-specific capacity. Both systems use only the data and checkpoints released for the challenge, and each evaluation recording is classified independently.

Index Terms— DCASE 2026, domain-incremental learning, audio classification, logit stacking, low-rank adaptation

1. INTRODUCTION

DCASE 2026 Task 7 addresses domain-agnostic incremental learning for sound classification [1]. The system is exposed to acoustic domains sequentially and must retain previously learned knowledge while adapting to later domains. The additional difficulty is that the test-time domain identity is hidden. A model that stores domain-specific parameters must therefore decide, from the audio itself, which branch or combination of branches should be trusted for classification.

This setup reflects practical audio-recognition scenarios in which the recording conditions or deployment environment can change after the initial model has been trained. Fully retraining the model after each change is expensive, while updating all parameters can cause forgetting on earlier domains. Domain-incremental learning methods reduce this problem by keeping a shared representation and isolating a limited set of domain-specific parameters [2].

The baseline domain-agnostic inference rule is simple and reproducible. It forwards each recording through all available domain branches, computes the entropy of each branch posterior, and keeps the branch with the lowest entropy. This uses confidence as a proxy for domain suitability. However, entropy alone does not describe how branches disagree, whether two classes are competing, or whether a branch is confidently wrong. Our submissions are built around the idea that these branch outputs contain richer information than the minimum entropy value [2].

We submit two systems. BelHadj_VUB_task7_1 keeps the baseline backbone fixed and learns a stacking classifier over the three branch outputs. BelHadj_VUB_task7_2 adds low-rank convolutional adapters to the backbone, trains adapters for the available incremental domains, and then applies the same stacking principle. The first system studies decision-level fusion of existing branches. The second system combines this fusion with a constrained feature-adaptation mechanism.

2. SYSTEM DESCRIPTION

The submitted systems share the same audio frontend, class set, baseline checkpoint, and three-branch domain structure. A branch corresponds to selecting one set of domain-specific batch-normalization layers inside the same convolutional network. This section first describes the common processing pipeline, then gives the two submitted extensions.

2.1. Frontend and backbone

All audio is loaded as mono waveform input at 32 kHz. Each waveform is padded with zeros or truncated to 4 s, corresponding to 128000 samples. The frontend inside the network computes a log mel representation with 64 mel bands, an FFT size of 1024 samples, a hop size of 320 samples, and a frequency range from 50 Hz to 14 kHz. These preprocessing settings are unchanged from the official baseline.

The acoustic model is a CNN14-style convolutional network. It contains six convolutional blocks with output channel sizes 64, 128, 256, 512, 1024, and 2048. Each block has two 3-by-3 convolutional layers followed by domain-specific batch-normalization layers and ReLU activations. Average pooling is applied after each block, and dropout with probability 0.2 is used during training. After the final block, temporal max pooling and temporal mean pooling are added together before a fully connected layer predicts the ten classes: alarm, baby, dog, engine, fire, footsteps, knock, phone, piano, and speech.

The three domain branches share convolutional and classifier weights but use separate batch-normalization parameters. Let $z_t(x) \in \mathbb{R}^{10}$ be the logits for input x when branch t is selected. The corresponding posterior vector is

$$p_t(x) = \mathrm{softmax}(z_t(x)). \tag{1}$$

The baseline domain-agnostic rule computes

$$H_t(x) = -\sum_c p_t(c|x) \log p_t(c|x) \tag{2}$$

and selects the prediction from the branch with minimum entropy. Our systems still evaluate all three branches, but the final class prediction is produced by a learned meta-classifier rather than by hard entropy selection.

2.2. Aligned logit stacker

BelHadj_VUB_task7_1 is a pure decision-level extension of the baseline. The baseline checkpoint is frozen, and for each recording the model is evaluated with all three domain branches. This produces a tensor of branch logits $[z_1(x), z_2(x), z_3(x)]$. Since logits from different batch-normalization branches can have different scales and offsets, each branch first passes through a separate linear aligner

$$\tilde{z}_t(x) = A_t(z_t(x)) = W_t z_t(x) + b_t. \tag{3}$$

The aligners are initialized as identity mappings, so the initial stacked representation is equivalent to the original baseline logits.

The meta-classifier is inspired by stacked generalization [3]. It receives a compact description of each branch's evidence and of the disagreement between branches. From the aligned logits we compute posterior probabilities, predictive entropy, maximum posterior probability, and the margin between the largest and second-largest posterior probabilities. Across branches, we compute the mean class posterior and the per-class standard deviation of the posteriors. The complete feature vector is

$$\begin{aligned}\phi(x) = [&\tilde{z}_1, \tilde{z}_2, \tilde{z}_3, p_1, p_2, p_3, H_1, H_2, H_3,\\ &m_1, m_2, m_3, \Delta_1, \Delta_2, \Delta_3, \bar{p}, \sigma_p],\end{aligned} \tag{4}$$

where m_t is the maximum posterior probability, Δ_t is the top-two posterior margin, $\bar{p}$ is the mean posterior vector, and σ_p is the per-class branch-disagreement vector. For three branches and ten classes this gives an 89-dimensional input vector: 30 aligned logits, 30 posterior probabilities, 9 scalar confidence features, and 20 cross-branch statistics.

The classifier consists of layer normalization, two hidden linear layers with 256 units, ReLU activations, dropout rates 0.25 and 0.15, and a final ten-class linear layer. The loss is class-balanced cross-entropy with weights computed from the D2/D3 development training labels. The retained run uses AdamW [4] with learning rate 10^{-3}, weight decay 10^{-4}, batch size 256, 60 epochs, and random seed 1193. No waveform-level fine-tuning is performed for this system; only the branch aligners and the stacker parameters are updated.

2.3. Low-rank convolutional adapters

BelHadj_VUB_task7_2 modifies the representation before applying the same stacking idea. It adds trainable low-rank convolutional adapters to the baseline backbone. The design follows low-rank adaptation [5], but applies the residual low-rank path to two-dimensional convolutions. Each of the twelve convolutional layers in the six convolutional blocks is wrapped as a frozen base convolution plus a task-specific adapter.

For an input feature map h, the adapted convolution is

$$y = Wh + \frac{\alpha}{r} U_t V_t h, \tag{5}$$

where W is the frozen baseline convolution, V_t is a rank-r down-projection convolution, U_t is a 1-by-1 up-projection convolution, and t denotes the active branch. The submitted model uses rank $r = 8$ and scale parameter $\alpha = 16$. The down-projection keeps the same kernel size, stride, padding, dilation, and grouping as the base convolution. The up-projection is initialized to zero, so each adapter initially contributes no residual and the adapter-augmented model starts from the baseline function.

During adaptation, only the adapter parameters for the current branch are trainable; the baseline convolutional filters, classifier weights, and adapters for the other branches remain fixed. This gives each incremental domain a small residual update without overwriting the shared convolutional filters. We train the D2 adapter using branch 1 and the D3 adapter using branch 2. Branch 0 keeps its zero-initialized adapter residual and therefore remains the original baseline path. The D2 and D3 adapters are trained for 10 epochs per domain with class-balanced cross-entropy, AdamW learning rate 10^{-4}, weight decay 10^{-4}, batch size 32, and adapter dropout 0.05. After adapter training, branch logits are generated from the adapter-enhanced model and a 256-dimensional aligned stacker is trained for 40 epochs with learning rate 10^{-3}.

Table 1: Development data used for training and validation.

Domain	Train recordings	Test recordings
D2	1530	639
D3	1882	806

3. TRAINING AND EVALUATION PROTOCOL

Both submissions use only the released Task 7 data and the official baseline checkpoint after learning D3. No external audio data, external labels, or external pretrained audio embeddings are used. The development setup available for our experiments contains D2 and D3 train/test splits. Table 1 summarizes the number of recordings used for training and validation.

For BelHadj_VUB_task7_1, the baseline checkpoint is used to build a fixed logit bank on the D2/D3 training split. The stacker is trained on this bank; at validation and evaluation time, branch logits are recomputed for the input recording and passed through the trained stacker. For BelHadj_VUB_task7_2, adapters are first trained sequentially on D2 and D3. After this stage, a second logit bank is generated from the adapter-enhanced model and used to train the final stacker.

The official evaluation recordings are used only to generate one predicted label per file for the submitted CSV files. We do not use evaluation labels, evaluation-domain labels, pseudo-labels, or aggregate statistics from the evaluation set for training, calibration, or model selection. The submitted prediction files contain one tab-separated filename and class label per evaluation recording.

Development performance is measured with class-balanced accuracy. For a domain d, let $n_{ij}^{(d)}$ be the number of examples from

Table 2: Development class-balanced accuracy on the available D2/D3 split.

System	D2 (%)	D3 (%)	Average (%)
BelHadj_VUB_task7_1	70.94	61.22	66.08
BelHadj_VUB_task7_2	70.11	63.43	66.77

class i predicted as class j. With $C = 10$ classes, the score is

$$\mathrm{Acc}_d = \frac{1}{C} \sum_{i=1}^{C} \frac{n_{ii}^{(d)}}{\sum_{j=1}^{C} n_{ij}^{(d)}}. \tag{6}$$

The reported average is the arithmetic mean of the D2 and D3 scores. In Table 2, scores are reported as percentages.

4. SUBMITTED SYSTEMS

Two systems were submitted to the challenge. The first submission, BelHadj_VUB_task7_1, is the aligned logit stacker. It keeps the original baseline backbone unchanged and learns only the branch aligners and final meta-classifier. This submission isolates the effect of replacing minimum-entropy branch selection with a supervised decision rule over branch evidence.

The second submission, BelHadj_VUB_task7_2, is the low-rank adapter expert. It keeps the same final stacking mechanism, but first adapts the convolutional representation through branch-specific LoRA-style residual paths. This submission tests whether the learned stacker benefits from branch logits that come from a parameter-efficient adapted backbone.

Both systems use a single retained checkpoint for the official evaluation predictions. No ensembling over random seeds or checkpoints is used. The two submissions are therefore compact and differ primarily in whether the convolutional representation is adapted before branch-level stacking.

5. EXPERIMENTAL RESULTS

Table 2 reports the class-balanced development accuracy on the available D2/D3 test split. These scores were computed after recreating the two training pipelines and generating the retained checkpoints used for the submission.

The aligned stacker obtains the better D2 score, while the low-rank adapter expert obtains the better D3 score and the best average among the two submitted systems. The difference between the two rows is small but consistent with the design of the methods. BelHadj_VUB_task7_1 can only reinterpret the existing baseline branch outputs. BelHadj_VUB_task7_2 changes the branch outputs themselves by adding a small domain-specific residual path in each convolutional layer before the final stacker is trained.

These results suggest that branch-level stacking and adapter-based feature adaptation address different parts of the domain-agnostic problem. Stacking uses evidence already present in the branch logits, including confidence, class margins, and branch disagreement. The adapter system modifies the features that produce those logits, which is especially beneficial on D3 in our development experiments. Since both systems still rely on the same official baseline checkpoint and no external data, the comparison mainly reflects the effect of adding the low-rank residual adaptation stage.

6. CONCLUSION

This report described the two VUB systems submitted to DCASE 2026 Task 7. Both systems keep the official CNN14-style baseline as their starting point and focus on the test-time domain-agnostic decision. BelHadj_VUB_task7_1 learns an aligned stacking classifier over the three domain branches. BelHadj_VUB_task7_2 adds low-rank convolutional adapters for domain-specific feature adaptation and then applies the same learned branch-fusion mechanism.

On the available D2/D3 development split, the adapter-based submission gives the best average score among the two systems, while the pure stacker remains a compact alternative that does not alter the baseline backbone. Future work could train the adapters and stacker jointly, add explicit calibration losses for branch probabilities, or explore stronger augmentation within the data allowed by the challenge rules.

7. REFERENCES

[1] R. Casciotti, M. Mulimani, M. Harju, J. R. Jensen, and A. Mesaros, "Domain-agnostic incremental learning for sound classification. a dcase 2026 challenge task," *arXiv preprint arXiv:2606.02173*, 2026.

[2] M. Mulimani and A. Mesaros, "Online incremental learning for audio classification using a pretrained audio model," in *2025 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2025, pp. 1–5.

[3] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.

[4] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. [Online]. Available: https://openreview.net/forum?id=Bkg6RiCqY7

[5] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022. [Online]. Available: https://openreview.net/forum?id=nZeVKeeFYf9