MESH: MULTI-EMBEDDING SYSTEM WITH HIERARCHICAL PROXY LEARNING FOR DCASE 2026 TASK 1

Technical Report

Sarang Han[1], *Minsik Jo*[2], *Eunseo Ha*[2],
Minju Chae[2], *Hyeonguk Kang*[2], *Geonwoo Lee*[2,*],

[1]Department of Data Science, Chosun University,
[2]Department of AI Software, Chosun University,
Gwangju 61452, Republic of Korea
{6002tkfkd, mswd81, murru8989, minju9642, solokho, geonwoo}@chosun.ac.kr

ABSTRACT

This technical report describes a multi-embedding system with hierarchical proxy learning (MESH) for DCASE 2026 Challenge Task 1. The task is based on the Broad Sound Taxonomy (BST) for hierarchical audio classification, where each sample is assigned to a lower-level category within a top-level class. Fixed audio embeddings were extracted from multiple pretrained models and pooling configurations, and the final embedding set was selected using validation hierarchical F-score (hF) and embedding diversity. The selected audio embeddings were concatenated with fixed CLAP text embeddings, and the classifier was trained through BSD35k-CS pretraining followed by BSD10k-v1.2 fine-tuning. To incorporate the BST hierarchy, a hierarchical proxy loss included distinct proxy sets for top-level and lower-level classes. Final predictions were aggregated with selective kernel-based output fusion and OOF-based ensemble selection, and the best OOF ensemble achieved 84.59% hF among the four submitted systems.

***Index Terms*—** Heterogeneous Audio Classification, Broad Sound Taxonomy, Multi-Embedding, Hierarchical Proxy Loss, Two-Stage Training

1. INTRODUCTION

DCASE 2026 Challenge Task 1 targets hierarchical audio classification using the Broad Sound Taxonomy (BST) [1, 2, 3]. In this task, each audio sample is assigned to a lower-level category under a top-level class. The dataset include music, speech, sound effects, instrument samples, and soundscapes. This heterogeneity requires representation selection and taxonomy-aware classification. The challenge provides the verified BSD10k-v1.2 dataset. It also provides the larger crowd-sourced BSD35k-CS dataset, which contains noisy labels [4].

These task-specific constraints require a system design that considers for representation diversity, label reliability, and the BST hierarchy. First, the heterogeneous audio samples requires representations not limited to a single pretrained model. Second, the difference between BSD35k-CS and BSD10k-v1.2 calls for a training strategy that distinguishes noisy pre-training data from reliable fine-tuning data. Finally, the BST hierarchy should be preserved during training instead of treating top-level and lower-level classes as independent labels. Accordingly MESH evaluates and combines multiple pretrained audio embeddings, applies data filtering with two-stage training, and employs a hierarchical proxy-based hierarchical proxy loss.

We propose the Multi-Embedding System with Hierarchical Proxy Learning (MESH) for submission to DCASE 2026 Task 1. MESH fits classifiers on fixed audio embeddings concatenated with fixed CLAP text embeddings, without fine-tuning the audio embedding backbones. A set of candidate audio embeddings were evaluated on BSD10k-v1.2, and the final embedding set was selected using validation hF and embedding diversity. The classifiers were trained with BSD35k-CS for noisy pre-training followed by BSD10k-v1.2 for reliable fine-tuning. At inference, classifier outputs were aggregated with selective kernel-based output fusion and OOF-based ensemble selection.

2. PROPOSED SYSTEM

The overall pipeline of MESH is shown in Fig. 1. MESH is organized into three stages, including audio embedding model selection, two-stage classifier training, and classifier output aggregation. In this system, only the audio embedding models are selected through the embedding evaluation, while CLAP text embeddings are treated as fixed features throughout classifier training.

Specifically, audio embeddings are extracted from multiple pretrained audio encoders, and candidate audio embedding models are selected using validation hF and embedding diversity. The selected audio embeddings are concatenated with fixed CLAP text embeddings and used as classifier inputs. The classifiers are trained sequentially with BSD35k-CS pre-training followed by BSD10k-v1.2 fine-tuning. The trained classifier outputs are aggregated using Selective Kernel-based output fusion and OOF-based ensemble selection followed by weight optimization. The detailed settings of each stage are described in the following subsections.

*Corresponding author.

†This research was supported in part by the MSIT (Ministry of Science and ICT), Korea, under the National Program for Excellence in SW supervised by the IITP (Institute of Information & Communications Technology Planning & Evaluation) in 2026 (2024-0-00062), and by "Project for Science and Technology Opens the Future of the Region" program through the Innopolis Foundation funded by Ministry of Science and ICT (2022-DD-UP-0312).

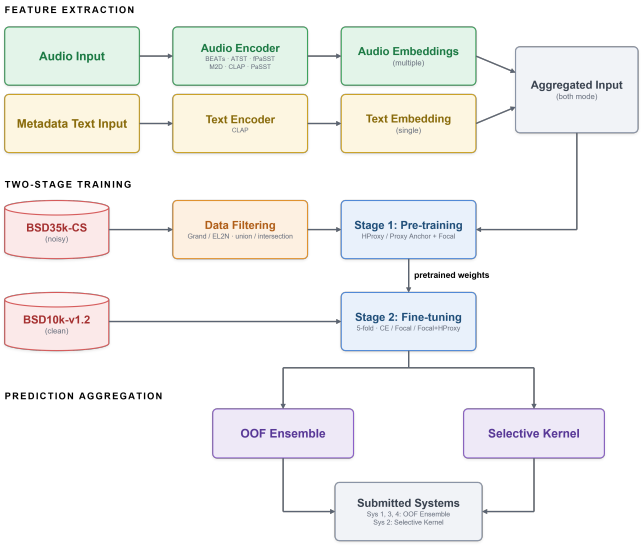

Figure 1: Overall pipeline of MESH.

2.1. Audio Embedding Extraction and Model Selection

To select audio embedding models for DCASE 2026 Task 1, we conduct an embedding search based on the BSD10k-v1.2 dataset. Table 1 summarizes the pretrained audio models and pooling strategies considered in the search. For each model, multiple audio embedding sets are extracted by varying the checkpoint and pooling strategy.

Table 1: Summary of audio embedding search space.

Model	Pooling strategy
BEATs	sequence pooling + chunk pooling
ATST-Frame	sequence pooling + chunk pooling
fPaSST	sequence pooling + chunk pooling
EfficientAT	chunk pooling
PaSST	chunk pooling
CLAP	chunk pooling
M2D	clip-level pooling

The final audio embedding models are selected using validation hF and embedding diversity. A single pretrained model may capture specific acoustic characteristics in the target dataset. Therefore, embeddings from different architectures, checkpoints, and pooling strategies are combined. Sequence pooling summarizes frame- or patch-level embeddings using statistics such as mean, maximum, and standard deviation. Chunk pooling divides an audio clip into multiple chunks and aggregates the chunk-level embeddings.

Table 2 summarizes the audio embedding models used for the final ensemble candidates. The selected audio embeddings are then concatenated with fixed CLAP text embeddings and used as classifier inputs in the two-stage training pipeline.

Table 2: Selected audio embedding models.

Backbone / checkpoint	Description	Dim.
CLAP music checkpoint	chunk mean pooling	512
CLAP 630K checkpoint	chunk mean+max pooling	1024
BEATs AS2M FT1	sequence mean+std, chunk mean pooling	1536
PaSST KD AudioSet	chunk max pooling	768
BEATs AS2M	sequence mean+max+std, chunk max pooling	2304
ATST-Frame weak checkpoint	sequence max+std, chunk mean+std pooling	3072
fPaSST weak checkpoint	sequence mean+max, chunk max+std pooling	3072
M2D CLAP ViT-base	clip-level mean+std pooling	7680

2.2. Data Filtering

BSD35k-CS is a relatively large-scale dataset obtained through a crowd-sourced process. Compared with BSD10k-v1.2, it provides more training samples, while being more likely to contain noisy labels. Accordingly, MESH applies data filtering before using BSD35k-CS for pre-training.

Table 3 summarizes the BSD35k-CS filtering strategies adopted in the system. Grand pruning removes samples that are unnecessary for training or likely to be noisy according to a precomputed Grand score. Error L2-Norm (EL2N) pruning removes hard samples with high training difficulty based on the EL2N score [5]. The EL2N score is computed from the difference between the model prediction and the ground-truth label in the early training stage.

Union and intersection strategies are also used to combine multiple filtering results. In addition, we compare the use of confidence score 1 information from the BSD10k-v1.2 dataset as an additional filtering condition. The no_conf1 setting includes this confidence score 1 condition whereas the conf1 setting does not include it.

Table 3: Data filtering strategies for BSD35k-CS.

Strategy	Description
Grand pruning	Removes samples that are unnecessary for training or likely to be noisy based on the Grand score
EL2N pruning	Removes hard samples with high EL2N scores at an early stage of training
Union filtering	Keeps samples selected by at least one filtering result
Intersection filtering	Keeps only samples commonly selected by multiple filtering results
no_conf1 setting	Includes the confidence score 1 information from BSD10k-v1.2 as an additional condition
conf1 setting	Does not use the confidence score 1 information as an additional condition

2.3. Two-Stage Training

MESH employs a two-stage training pipeline to handle for the different label reliability of BSD35k-CS and BSD10k-v1.2. In this pipeline, an embedding projection layer and a classifier are trained on selected audio embeddings concatenated with fixed CLAP text embeddings. All training configurations use the both mode, where the classifier input consists of audio and text embeddings.

To improve the stability of proxy-based training, the classifier head is simplified compared with the baseline structure. The origi-

nal baseline classifier includes a deep multilayer perceptron (MLP) projection and residual classifier blocks. In contrast, our proxy-based models use a single linear projection layer to produce a compact 128-dimensional embedding z.

In the pre-training stage, BSD35k-CS is used for the initial learning of classifier representations under noisy labels. We compare hierarchical proxy (HProxy)-only training with a joint objective that combines Proxy Anchor loss [6] and FocalLoss [7]. In the fine-tuning stage, 5-fold cross-validation is performed using the verified BSD10k-v1.2 dataset. CrossEntropyLoss, FocalLoss, and a joint objective combining FocalLoss and HProxy are compared in this stage.

To incorporate the two-level hierarchy of the Broad Sound Taxonomy (BST), we introduce level-specific proxy sets for top-level and lower-level classes, denoted as $P_{\text{top}} \in \mathbb{R}^{5\times d}$ and $P_{\text{second}} \in \mathbb{R}^{23\times d}$, respectively. HProxy is implemented as a cross-entropy-based hierarchical proxy classification loss. It consists of lower-level proxy classification, top-level proxy classification, proxy alignment across label levels, within-parent proxy regularization, and top-level proxy separation.

$$L_{\text{HProxy}} = L_{\text{second}} + \alpha L_{\text{top-cls}} + \beta L_{\text{align}} + \gamma L_{\text{sib}} + \delta L_{\text{top-sep}}. \tag{1}$$

$$L_{\text{second}} = L_{\text{proxy}}(z, P_{\text{second}}, y_{\text{second}}) \tag{2}$$

$$L_{\text{top-cls}} = L_{\text{proxy}}(z, P_{\text{top}}, y_{\text{top}}) \tag{3}$$

$$L_{\text{proxy}}(z, P, y) = \text{CE}\left(\frac{s(z, P)}{\tau}, y\right). \tag{4}$$

Here, CE denotes cross entropy, $s(z, P)$ denotes the cosine similarity between the embedding and proxy vectors, and τ is the temperature parameter. In our implementation, τ is set to 0.07. L_{align} encourages lower-level proxies to align with their corresponding top-level proxies. L_{sib} penalizes overly similar lower-level proxies that share the same top-level class, while $L_{\text{top-sep}}$ penalizes overly similar top-level proxies. The coefficients α, β, γ, and δ control the relative weights of the corresponding terms. Through this objective, HProxy encourages hierarchical consistency in the proxy space while maintaining discriminability among fine-grained lower-level categories.

Table 4: Summary of training objectives in the two-stage pipeline.

Stage	Dataset	Objective
Pre-train	BSD35k-CS	HProxy
Pre-train	BSD35k-CS	Proxy Anchor + Focal
Fine-tune	BSD10k-v1.2	CE
Fine-tune	BSD10k-v1.2	Focal
Fine-tune	BSD10k-v1.2	Focal + HProxy

For each audio embedding model, multiple classifier configurations are constructed by varying the pre-training objective, fine-tuning objective, data filtering strategy, and early stopping metric. Early stopping uses either validation hF or validation accuracy, while hF is retained as the main metric for final performance comparison.

2.4. Ensemble and Selective Kernel

In the final prediction stage, MESH aggregates classifier outputs obtained from multiple audio embedding models and training configurations. Since each pretrained audio model captures different acoustic patterns, classifier output aggregation can combine complementary predictions for heterogeneous audio classification.

We use a selective kernel-based output fusion module, inspired by selective kernel networks [8], as a separate classifier output aggregation branch. This module operates on classifier outputs and does not modify the pretrained audio embedding backbones. It combines outputs from multiple trained classifiers into a fused prediction. In MESH, selective kernel fusion is treated as an output-level strategy rather than a feature extraction backbone. It is kept separate from the OOF-based greedy ensemble candidate pool.

The OOF-based ensemble is constructed using OOF softmax outputs from the 5-fold training process. For each fold, OOF outputs are the predictions produced when that fold is held out from training. Ensemble members are selected with greedy forward selection. At each step, each remaining candidate is temporarily added to the current ensemble set, and hF is computed from the ensemble prediction. The candidate with the highest hF is retained, and the same process is repeated.

After each selection step, SciPy Nelder–Mead-based weight optimization is applied to the accumulated ensemble. For each OOF-based submitted system, the final ensemble is selected from the step with the highest optimized hF.

3. RESULT AND SUBMISSION

Using the classifier output aggregation procedures described in Section 2.4, we constructed four submitted systems from different candidate pools and aggregation settings. Table 5 summarizes the submitted system configurations and their validation hF. System 1 used the full OOF-based ensemble candidate pool, including all pre-training groups and fine-tuning configurations. System 2 used Selective Kernel-based output fusion, where four models were selected from 64 SK-fusion candidate models using greedy forward selection. System 3 used only HProxy pre-training configurations, whereas System 4 used only Proxy Anchor pre-training configurations.

4. EXTERNAL RESOURCES

In addition to BSD10k-v1.2 and BSD35k-CS provided in DCASE 2026 Task 1, the submitted systems used pretrained audio embedding models as external resources. The external audio embedding models were the CLAP music checkpoint, CLAP 630K checkpoint, BEATs AS2M, BEATs AS2M FT1, PaSST KD AudioSet, ATST-Frame weak checkpoint, fPaSST weak checkpoint, and M2D CLAP ViT-base. These models were used as fixed audio feature extractors during audio embedding extraction.

Following the DCASE 2026 Task 1 rules, the system did not directly use any data uploaded to Freesound after April 1, 2025 for training. It also did not use any external dataset or pretrained model containing Freesound data after the cutoff date. CLAP text embeddings were used as fixed text features based on the provided metadata.

Table 5: Submitted system configurations and performance.

System	Pool	Models	Sel.	hF
System 1	All configs	M2D×3, C630×5, fPaSST×1, Cmus×11, ATST×1, BEATs-FT1×2, PaSST×2	25	84.59%
System 2	Selective Kernel	M2D×2, Cmus×2	4	84.38%
System 3	HProxy pre-train only	M2D×2, C630×2, ATST×2, PaSST×1	7	84.35%
System 4	Proxy Anchor pre-train only	C630×3, Cmus×3, BEATs×2	8	84.57%

C630: CLAP 630K, Cmus: CLAP music, BEATs: BEATs AS2M, BEATs-FT1: BEATs AS2M FT1.

5. CONCLUSION

In this report, we presented MESH, a multi-embedding system with hierarchical proxy learning for DCASE 2026 Task 1. MESH selects audio embedding models from multiple pretrained audio encoders and concatenates the selected audio embeddings with fixed CLAP text embeddings. The classifiers are trained with BSD35k-CS pre-training followed by BSD10k-v1.2 fine-tuning to account for the different label reliability of the two datasets. Hierarchical proxy learning is used to incorporate the two-level BST hierarchy through top-level and lower-level class relations.

For the final submissions, we used two classifier output aggregation strategies. Selective Kernel-based output fusion was used as a separate output-level branch for combining classifier outputs. OOF-based greedy forward selection with weight optimization was used to select and combine classifier outputs from different candidate pools. Among the four submitted systems, the best OOF-based ensemble achieved 84.59% hF. These results suggest that combining multiple audio embedding models and training configurations is useful for heterogeneous audio classification under the BST hierarchy.

6. REFERENCES

[1] P. Anastasopoulou, X. Serra, and F. Font, "A general-purpose sound taxonomy for the classification of heterogeneous sound collections," in press. [Online]. Available: https://www.researchsquare.com/article/rs-7206795/v1

[2] P. Anastasopoulou, J. Torrey, X. Serra, and F. Font, "Heterogeneous sound classification with the Broad Sound Taxonomy and Dataset," in *Proc. Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2024.

[3] P. Anastasopoulou, F. A. Dal Rí, X. Serra, and F. Font, "Hierarchical and multimodal learning for heterogeneous sound classification," in *Proc. Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2025.

[4] P. Anastasopoulou and F. Font Corbera, "BSD35k-CS (Broad Sound Dataset 35k – Crowd Sourced)," Zenodo, Mar. 2026, doi: 10.5281/zenodo.19187100.

[5] M. Paul, S. Ganguli, and G. K. Dziugaite, "Deep learning on a data diet: Finding important examples early in training," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 20596–20607.

[6] S. Kim, D. Kim, M. Cho, and S. Kwak, "Proxy anchor loss for deep metric learning," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 3238–3247.

[7] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, 2017, pp. 2980–2988.

[8] X. Li, W. Wang, X. Hu, and J. Yang, "Selective kernel networks," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 510–519.