IDIAP RESEARCH REPORT

ADAPTATION OF SPEECH AND BIOACOUSTICS MODELS

Eklavya Sarkar Amir Mohammadi
Mathew Magimai-Doss

Idiap-RR-05-2025

JULY 2025

Centre du Parc, Centre du Parc, Rue Marconi 19, CH - 1920 Martigny
T +41 27 721 77 11 F +41 27 721 77 12 info@idiap.ch www.idiap.ch

ADAPTATION OF SPEECH AND BIOACOUSTICS MODELS

Eklavya Sarkar *
Idiap Research Institute
Ecole Polytechnique Fédérale de Lausanne
Switzerland
eklavya.sarkar@idiap.ch

Amir Mohammadi,
Idiap Research Institute
Switzerland
amir.mohammadi@idiap.ch

Mathew Magimai.-Doss,
Idiap Research Institute
Switzerland
mathew@idiap.ch

1 INTRODUCTION

In (Sarkar & Magimai.-Doss, 2025), we examined whether fine-tuning models pre-trained on human speech could improve processing of animal vocalizations, but found no consistent gains using publicly available models fine-tuned on ASR. In this chapter, we investigate whether fine-tuning the aforementioned SSL models directly on the downstream bioacoustic data yields better performance on the same classification tasks.

Fine-tuning a pre-trained model on a downstream task or domain is the second step of the typical SSL framework. However, in standard fine-tuning, the entire parameter set of the network is updated, which can quickly become exceedingly computationally expensive or even infeasible. The advent of large foundation models has lead to the development of a number of parameter efficient fine-tuning (PEFT) techniques for downstream tasks. The core idea behind PEFT approaches is to only strategically update a small subset of parameters, while keeping the majority frozen, thereby greatly reducing the computational cost and tuning time.

To this end, we adopt Low-Rank Adaptation (LoRA) (Hu et al., 2022) for parameter-efficient fine-tuning (PEFT) and apply it to two architecturally identical models: HuBERT (pre-trained on human speech) and AVES (pre-trained on bioacoustics). We focus exclusively on the call-type identification (CTID) task and conduct systematic ablations to understand the adaptation process. Specifically, we explore which permutation of transformer projection matrices to optimize, which encoder layers permutations to fine-tune, and whether to freeze or drop the remaining layers, in order to achieve better performance. Moreover, having observed a progressive decline in representational quality across deeper layers in the previous chapter, we examine whether this layer-wise trend changes when models are fine-tuned on domain-specific data.

The remainder of this chapter is organized as follows. Section 2 provides an overview of parameter efficient fine-tuning and parameter pruning, while Section 3 outlines the research questions and experimental methodology for the different experiments. Finally, Section 4 present the results from the various studies, and Section 5 concludes the chapter.

2 PARAMETER EFFICIENT FINE-TUNING AND PARAMETER PRUNING

The following Section 2.1 gives a brief overview of Low-Rank Adaptation (LoRA), a modern PEFT adaptation technique which has gained a lot of prominence thanks to its simplicity and effectiveness. We also introduce the notion of parameter pruning in Section 2.3.

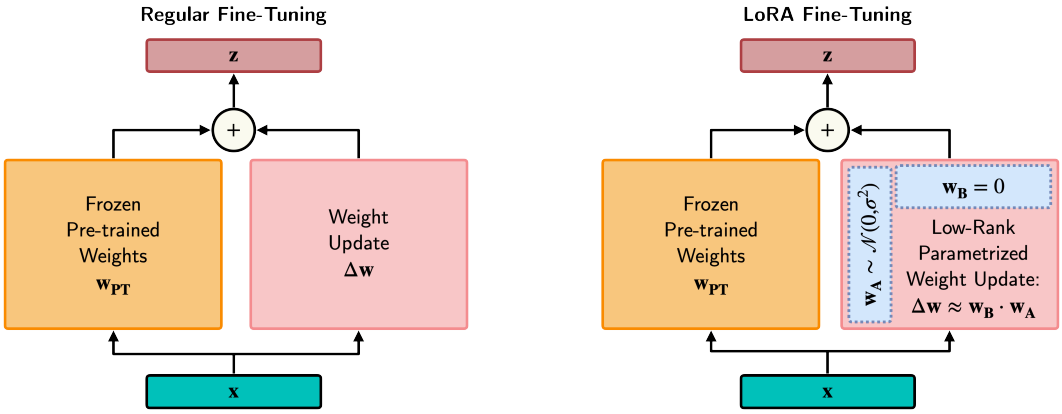

Figure 1: Regular fine-tuning compared to LoRA adaptation. $\boldsymbol{x}$ and $\boldsymbol{z}$ are the input and output.

2.1 Low-Rank Adaptation (LoRA)

During training or fine-tuning, a model's parameters are updated through backpropagation. Although these weight parameters $\boldsymbol{w}$ are full-rank matrices, they have been shown to reside in a much lower-dimensional subspace, i.e. to have low 'intrinsic dimension' (Aghajanyan et al., 2021). Likewise, (Hu et al., 2022) demonstrated that the fine-tuning updates $\Delta\boldsymbol{w}$ themselves exhibit a low 'intrinsic rank'. Consequently, one can efficiently parameterize these updates by decomposing $\Delta\boldsymbol{w}$ into the product of two low-rank matrices $\boldsymbol{w_B}$ and $\boldsymbol{w_A}$. The concept is illustrated in Figure 1 and formalized as:

$$\boldsymbol{w}_{FT} = \boldsymbol{w}_{\text{PT}} + \Delta\boldsymbol{w} \tag{1}$$

$$= \boldsymbol{w}_{\text{PT}} + \boldsymbol{w_B} \cdot \boldsymbol{w_A} \tag{2}$$

where $\boldsymbol{w_A} \in \mathbb{R}^{r\times n}$ and $\boldsymbol{w_B} \in \mathbb{R}^{m\times r}$ for a weight matrix $\boldsymbol{w} \in \mathbb{R}^{m\times n}$. Here, $\boldsymbol{w}_{FT}$ and $\boldsymbol{w}_{PT}$ denote the fine-tuned and pre-trained weights, respectively. We initialize one matrix with random Gaussian values $\boldsymbol{w_A} \sim \mathcal{N}(0, \sigma^2)$, and the other as a zero matrix $\boldsymbol{w_B} = \mathbf{0}$, ensuring that the model's initial output matches the pre-trained model. During the fine-tuning process, both the pre-trained weights $\boldsymbol{w}_{PT}$ and the new *adapters* $\boldsymbol{w_A}$ and $\boldsymbol{w_B}$ are used to compute the hidden states $\boldsymbol{z}$ during the forward pass. However, during the backward pass, only the gradients of low-rank matrices are required to be computed and optimized – the original pre-trained parameters remain frozen. This selective updating drastically reduces the computational cost compared to regular 'full' fine-tuning.

In practice, there are two additional hyperparameters: a constant scaling factor α and the rank $r \ll \min(m, n)$. The modified forward pass, where $\boldsymbol{x}$ is the input and $\boldsymbol{z}$ the output, is thus defined as:

$$\boldsymbol{z} = \boldsymbol{w}_{\text{PT}}\boldsymbol{x} + \frac{\alpha}{r}\boldsymbol{w_B}\boldsymbol{w_A}\boldsymbol{x}, \tag{3}$$

Typically LoRA is applied only to the weight matrices in the attention block of transformer-based models during fine-tuning, while the feed-forward module remains unchanged. This approach reduces the number of trainable parameters without compromising the integrity of the pre-trained representations. To the best of our knowledge, LoRA has not yet been employed to transfer models from human speech processing to the bioacoustics domain.

2.2 LoRA Adapters in Transformers

Having introduced the main principles of Low-Rank Adaptation, we now consider how these adapters $\boldsymbol{w_A}$ and $\boldsymbol{w_B}$ are integrated into the Transformer architecture shared by HuBERT and AVES. Inserting adapters at appropriate locations allows us to adapt large pre-trained models with minimal parameter updates, while preserving the bulk of the original weights.

*eklavyafcb.github.io

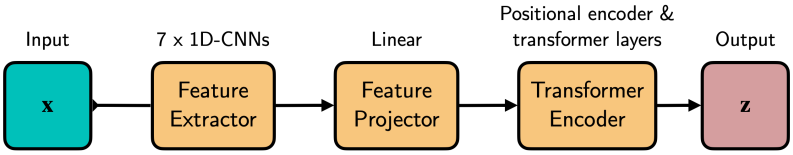

Figure 2: Transformer architecture of HuBERT and AVES.

As shown in Figure 2, the Transformer model consists of three main modules that transform raw audio $\boldsymbol{x}$ into context-aware feature representations $\boldsymbol{z}$:

- **Feature extractor**: seven 1D convolutional layers of different window lengths and shifts, alongside GeLU activation functions and LayerNorms. This block operates directly on the raw waveform, and converts the input audio signal into embeddings of size 512.
- **Feature projector**: a fully-connected layer, preceded by a LayerNorm and followed by a Dropout. This layer projects the output of the feature extractor embeddings from 512 into 768 dimensions.
- **Transformer encoder**: the core of the model, operating on 768-dim vectors, and itself consisting of:
  - One **positional encoder**: convolutional and GeLU layers that inject relative position information.
  - Multiple **Transformer (encoder) layers**: each composed of a *self-attention block* and a two-layer *feed-forward network*. Figure 3 illustrates one such layer. Note that the LayerNorm, Dropouts, skip connections, and activations have been omitted for simplicity.

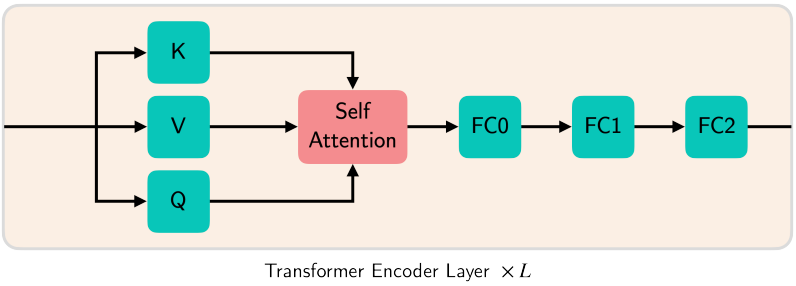

Figure 3: Simplified transformer encoder layer.

Within each Transformer encoder layer, there are multiple candidate weight matrices where LoRA adapters can be inserted to capture task-specific adjustments. In the diagram:

- The *self-attention* consists of the keys, queries, values (K, Q, V) matrices, as well as the output linear layer, often referred to as O, but here denoted as FC0..
- The *feed-forward network* consists of two fully-connected layers, henceforth referred to as FC1 and FC2.

LoRA adapters may be added to any of these projection matrices (Q, K, V, FC0, FC1, or FC2), enabling the model to learn low-rank updates at these points while keeping all other parameters fixed. By selecting different combinations of adapter placements, one can tailor the fine-tuning process to balance between parameter efficiency and adaptation flexibility.

2.3 Parameter Pruning and Layer Dropping

Large pre-trained speech foundation models can be over-parametrized for downstream tasks. Prior work in the literature has shown that structured or adaptive parameter pruning can reduce model

size while preserving strong classification performance (Peng et al., 2023). Rather than individual weights, layer dropping, i.e. removing entire layers, has also been investigated as a parameter pruning technique. Numerous layer-dropping strategies such as top-down, bottom-up, and alternating layer removals have been explored in transformer-based models, achieving up to 40% reduction in model size with only a 2% drop in downstream accuracy (Sajjad et al., 2023). Although these approaches have proven effective in NLP, their application to bioacoustics domain and effectiveness in cross-domain adaptation remains unexplored.

3 Research Questions and Experimental Methodology

This section formalizes the central research questions guiding our investigation into adapting pre-trained speech and bioacoustic models via LoRA, and defines the experimental design used to answer them.

3.1 Encoder Matrix Selection

Based on the possible adapter insertion points identified in Section 2.2, we first explore which combinations of weight matrices within the Transformer layers yield the greatest downstream classification performance when fine-tuned with LoRA. We also examine whether extending LoRA fine-tuning beyond the Transformer encoder, specifically to the feature extractor and feature projector, also leads to further improvements. To that end, we formulate the following two research questions:

Q1. Which subset or permutation of transformer projection matrices (K, Q, V, FC0, FC1, FC2) is most effective for LoRA-based fine-tuning?

To answer this, we compare the following adapter configurations:

- [FC1, FC2]: the two-layer feed-forward network only.
- [Q, K]: the self-attention query and key projections.
- [Q, K, V]: all three self-attention projections.
- [Q, K, V, FC0]: self-attention as well as the attention output projection.
- [Q, K, V, FC0, FC1, FC2]: all self-attention and feed-forward projections.

We individually fine-tune a pre-trained HuBERT under each of these five different settings on the *Train* set, and measure UAR on *Test*, thereby identifying which permutation delivers the best downstream adaptation.

Q2. Does applying LoRA adapters to the feature extraction and/or feature projection modules, in addition to the Transformer encoder, improve classification performance?

Although parameter-efficient fine-tuning typically focuses on the Transformer layers alone, strong acoustic domain shifts, such as moving from human speech to non-human animal vocalizations, may potentially benefit from adapting earlier, pre-encoder network components. To investigate this, we compare three configurations:

- *Encoder only*: LoRA adapters inserted in the Transformer encoder layers only (baseline). We insert the adapters on the optimal matrix permutation found from Q1.
- *Projector + encoder*: adapters applied to both the Transformer encoder and the feature projection fully-connected layer.
- *Extractor + projector + encoder*: LoRA adapters are applied to the Transformer encoder and the feature projection layer, while the feature extractor block is fully fine-tuned, instead of through LoRA decomposition. This is due to a limitation in the implementation of the PEFT HuggingFace library, which currently supports LoRA only on linear modules. However, we keep the feature extractor fully trainable, such that its convolutional filters can still directly adapt to the specific characteristics of bioacoustic signals.

We hypothesize that including the feature projector, a simple affine mapping, will yield additional gains, while adapting the convolutional extractor may have uncertain effects, given its role in low-level signal processing and the risk of disrupting learned acoustic filters. Moreover, the impact of full fine-tuning, as opposed to LoRA-based adaptation, may differ significantly in these components.

By systematically evaluating these configurations on our CTID task with UAR, we will identify which adapter placement strategy offers the best balance of parameter efficiency and performance.

3.2 Layer Selection Strategies

Choosing which encoder layers to adapt is an important decision in parameter-efficient fine-tuning. Rather than fine-tuning *all* layers, we investigate whether updating only a particular subset can yield comparable or better performance, and whether there exists a systematic strategy for selecting these layers. Prior work and previous chapters have shown that initial layers in speech SSL models work much better than the later layers for bioacoustics tasks. We therefore ask:

Q3. Which layer selection strategy for the Transformer encoder yields the most effective performance after fine-tuning ?

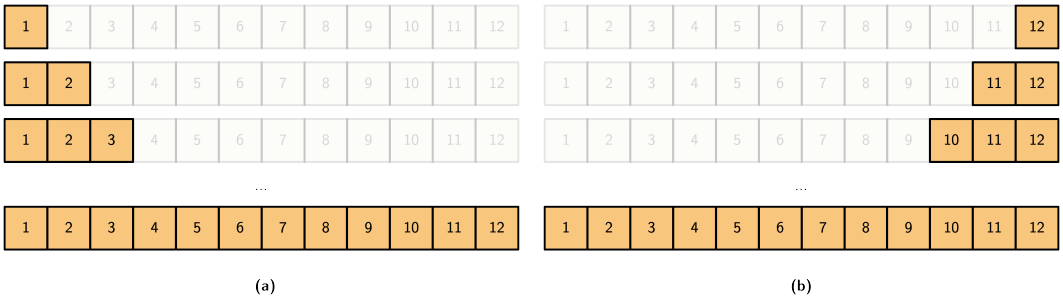

Figure 4: Layer selection strategies: (a) bottoms-up. (b) top-down. The numbers corresponds to transformer encoder layers. Each row represents a different layer permutation, eg. 1, 1–2, 1–3, etc.

To answer this question, we employ two different layer selection strategies, as shown on Figure 4. For each strategy, we compute all permutations:

- *Bottoms-up* strategy incrementally adapts the encoder from its lowest layer upwards. In one permutation, we only use the output embeddings of the first layer, then in the second combination, we use the ones of the second layer, with the input having gone through the first two layers, and so on till the final permutation where the input traverses all the layers and we use the output embeddings of the final layer.

  For L total encoder layers, denoted as $l_1, l_2, \ldots, l_L$, we define an independent fine-tuning configuration for each $k \in \{1, 2, \ldots, L\}$:

  $$\mathcal{A}_k = \{\, l_1, l_2, \ldots, l_k \}, \qquad \mathcal{F}_k = \{\, l_{k+1}, \ldots, l_L \},$$

  where $\mathcal{A}_k$ denotes the set of adapted layers with LoRA adapters, and $\mathcal{F}_k$ the set of frozen layers. We then measure downstream classification performance for each k, thereby quantifying the incremental contribution of the first k layers to the adaptation process.

  Since the later layers typically learn more task-specific information, we hypothesize that fine-tuning the lower layers could still bring substantial improvements, as these typically encode more acoustic information.

- *Top-down* strategy conversely starts by first adapting the highest-level layer embeddings only, and progressively includes lower layers. In this case, we define our configurations as:

  $$\mathcal{A}'_k = \{\, l_{L-k+1}, l_{L-k+2}, \ldots, l_L \}, \quad \mathcal{F}'_k = \{\, l_1, \ldots, l_{L-k} \},$$

  where $\mathcal{A}'_k$ are the layers adapted with LoRA, and $\mathcal{F}'_k$ are frozen. By evaluating classification performance for each k, we assess how the inclusion of progressively lower-level layers impacts adaptation.

  In this strategy, it could be argued that starting adaptation with the top layers could accelerate the domain adaptation and force the model to learn representations more relevant to the animal-specific vocalizations.

3.3 FINE-TUNING STRATEGIES: PROBING, FREEZING, AND PRUNING

Rather than simply freezing unselected layers during LoRA adaptation, parameter-pruning research detailed in Section 2.3 suggests that removing those layers from the model entirely may further improve efficiency without degrading performance. We therefore compare three distinct adaptation strategies, and formulate our question as:

Q4. Which approach yields the best downstream performance between (a) simple linear probing, (b) LoRA fine-tuning with layer freezing, and (c) LoRA fine-tuning with layer pruning ?

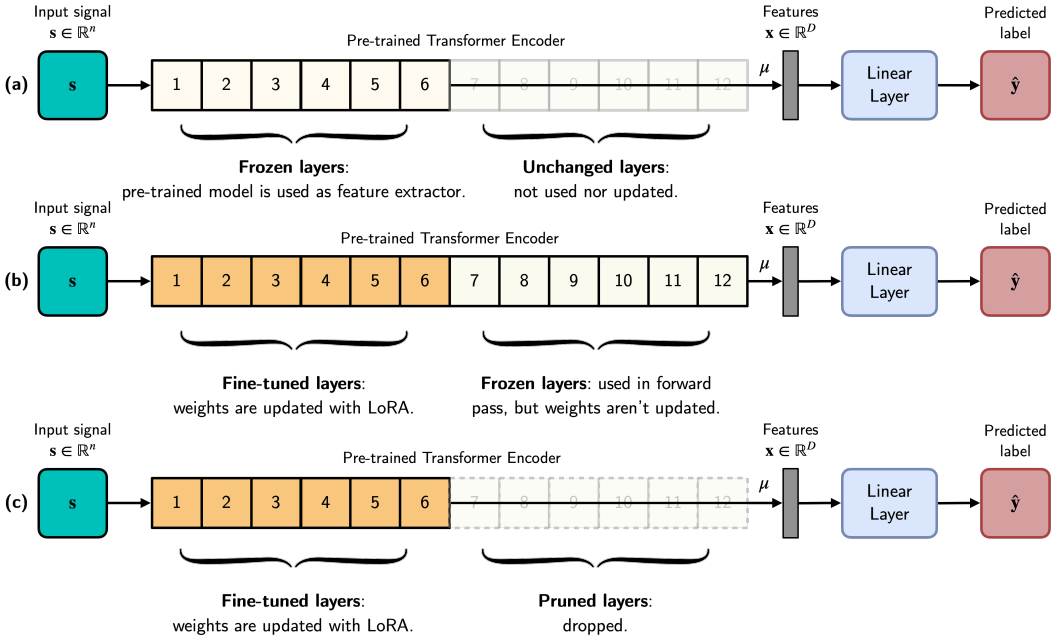

Figure 5: Three evaluation scenarios of a pre-trained SSL model using a linear classifier. This example depicts the case where layers 1–6 are selected and used for classification, while any remaining layers are either ignored, kept frozen, or pruned, depending on the scenario. **a) Linear probing**: all layers of the pre-trained model are frozen. The input signal $\boldsymbol{s}$ passes through the layers 1–6. The output embedding from layer 6 is extracted and given to a linear classifier, which is trained. The remaining layers are ignored. **b) LoRA fine-tuning with freezing**: LoRA adapters are inserted into the selected layers 1–6, which are adapted, while the others 7–12 remain frozen. **c) LoRA fine-tuning with layer pruning**: the model is pruned such that only the selected layers 1–6 are retained and then fine-tuned using LoRA, while all the others are removed from the model entirely. Note that layers 7–12 are functionally identical in scenarios (a) and (c): they are unused in both cases. We distinguish them visually to emphasize that in (c) they are explicitly removed from the model, whereas in (a) they are simply ignored. In each case, the output embeddings of the pre-trained model are mean-pooled over the temporal axis to produce a single functional feature vector $\boldsymbol{x}$. In practice, a LayerNorm layer is also implemented before the linear layer for robustness.

The three scenarios are illustrated in Figure 5, and explained below:

(a) *Linear probing*: All encoder layers remain frozen. We simply extract the output embedding of the selected layer(s), apply mean-pooling, and train a single linear classifier on top. Note that this scenario is essentially identical to the one used in Sarkar & Magimai.-Doss (2025), with the key difference that we only employ a single linear classifier instead of an MLP. Using the same classifier across all adaptation scenarios ensures a fair comparison.

(b) *LoRA + freezing*: LoRA adapters are inserted into the selected layers and fine-tuned, while all other layers remain frozen and only participate in the forward pass.

(c) *LoRA + pruning*: Selected layers receive LoRA adapters and are fine-tuned, but all other layers are removed from the model, and the classifier is applied directly on the output of the highest adapted layer.

In all scenarios, we apply the LoRA adapters to the optimal matrix permutation found from Q1. By evaluating each strategy on both HuBERT and AVES, we can determine whether dropping unused layers offers any advantage over freezing them, and how both compare to a classic linear-probing baseline. Finally, to assess which strategy performs best, we conduct all experiments on both the Abzaliev and IMV datasets.

4 Results and Analysis

4.1 Hyperparameter Selection

Given the large number of fine-tuning configurations and model permutations, we performed a preliminary grid search on HuBERT to identify a single set of LoRA hyperparameters that could then be kept constant across all subsequent experiments. The search spanned learning rate η, rank r, scaling α, dropout, weight decay, and number of epochs.

To ensure these settings generalize across different adaptation structures, we ran the search independently for each of the five matrix permutations defined in Q1, and for each of the twelve bottoms-up layer selections while keeping the unselected layers frozen. In total, this amounted to approximately 900 trials on the CTID task. The hyperparamters optimized in the grid search are given in Table 1.

Table 1: Search space to find optimal hyperparameters.

Hyperparameters	**Search Space**	**Optimal Value**
α	[1, 2, . . ., 60]	3
r	[4, 8, . . ., 64]	60
Dropout	[0, 0.1, 0.2, . . ., 1.0]	0.3
η	1e[-3, -2, -1]	1e-3
Weight decay	1e-9 – 9.67e-2	8e-09
Max. epochs	[1, 2, 3, 4, 5]	5

We found that a low learning rate (10^{-3}), a high adapter rank ($r = 60$), and a moderate scaling factor ($\alpha = 3$) produced consistently strong performance, with optimal dropout of 0.3 and minimal weight decay ($8 \cdot 10^{-9}$). These settings balance adaptation capacity against overfitting risk and are used throughout the rest of our studies.

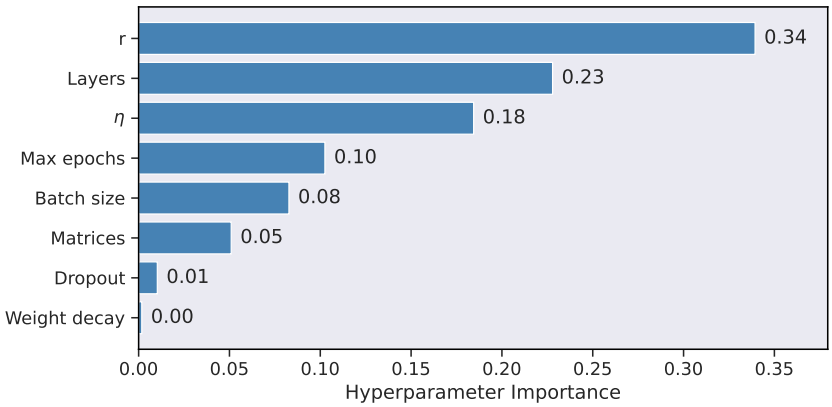

Figure 6: Hyperparameter importance on HuBERT, as estimated by the fANOVA algorithm.

The hyperparameter importance plot in Figure 6 quantifies each parameter's contribution to the variation in downstream UAR, as estimated by the fANOVA algorithm (Hutter et al., 2014). The results

indicate that adapter rank r is by far the most influential hyperparameter, reflecting the fact that increasing the latent dimensionality of the LoRA update substantially enhances the model's adaptation capacity. Next in importance is layer selection, confirming that the choice of encoder layers that receive adapters does affect the performance. The learning rate η remains critical, consistent with its central role in gradient-based optimization, but ranks below rank and layer decisions. The number of epochs and batch size exhibit moderate impact, suggesting that training duration and mini-batch stability provide incremental gains once rank, layers, and η are set. The choice of projection matrices, formulated in Q1, seems to have only a modest effect once the principal LoRA capacity and layer locations are determined. Finally, LoRA dropout and weight decay show near-zero importance, implying that explicit regularization is largely unnecessary under LoRA fine-tuning for CTID.

4.2 Matrix Selection Results (Q1)

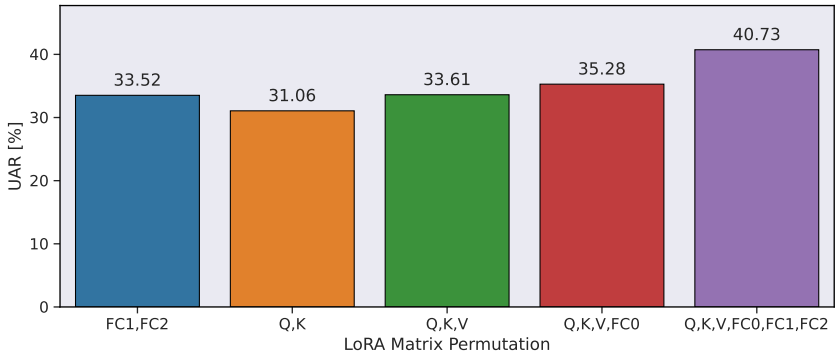

Figure 7: Best UAR [%] for each LoRA adapter configuration on layers 1–12. Fine-tuning all matrices yields the best performance.

Figure 7 shows the highest UAR score achieved for each of the five different LoRA adapter matrix configurations defined in Q1. To ensure a fair comparison across matrix combinations, we fix the selected layers to HuBERT encoder layers 1–12 for all experiments. All results are obtained on the Abzaliev dataset for the call-type classification (CTID) task. For each configuration, we report the best UAR achieved across our full hyperparameter sweep. The results exhibit a clear, monotonic progression:

$$\text{Q, K} \;<\; \text{Q, K, V} \;<\; \text{Q, K, V, FC0} \;<\; \text{Q, K, V, FC0, FC1, FC2.}$$

In other words, performance steadily increases as more projection modules are adapted. Fine-tuning only the query and key projections yields the lowest UAR, with each successive addition (value, attention output, feedforward layers) leading to higher scores. This progression highlights that granting the model greater adaptation capacity, by increasing the number of LoRA-enabled projections, consistently improves downstream accuracy, with the full set of adapters delivering the best result.

4.3 Layer Selection Strategy Results (Q2 & Q3)

Based on the previous results, we fix the matrix permutation to include all the aforementioned matrices, and now aim to identify which layer selection strategy and permutation yields the best fine-tuning results.

Figure 8 compares the best UAR scores across different layer selection strategies for both AVES and HuBERT. We observe that in both cases, fine-tuning the feature extraction (FE) layers severely degrades performance. Fine-tuning the feature projection (FP) alone does not significantly improve performance relative to other strategies, but it also does not degrade it, suggesting that FP adaptation is optional rather than essential. Furthermore, bottoms-up and top-down layer selection strategies yield comparable results, generally achieving scores in the range of 30–40% bracket for all layer permutations. Finally, neither AVES nor HuBERT consistently outperforms the other across all layer selections. However, HuBERT appears to perform slightly better in the later layers in the bottoms-up strategy, with or without feature projection tuning.

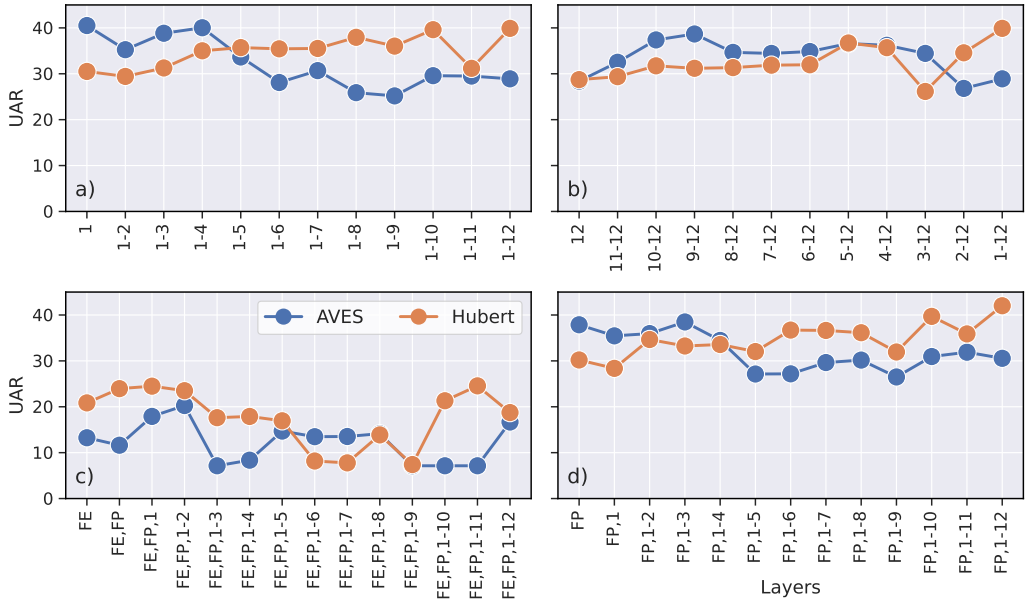

Figure 8: Layer selection strategy UAR [%] results: (a) bottoms-up, (b) top-down, (c) FE + FP + bottoms-up, (d) FP + bottoms-up.

4.4 Fine-Tuning Strategy Selection (Q4)

In this final research question, we evaluate three paradigms aforementioned in Q4, namely linear probing, LoRA with layer freezing, and LoRA with layer pruning, applied to the Transformer encoder in a bottoms-up layer selection. For fairness, we keep the feature extraction (FE) and feature projection (FP) modules unchanged, since standalone fine-tuning on these sub-modules did not yield consistent gains. We run these experiments on both the Abzaliev and IMV datasets, using AVES and HuBERT feature representations.

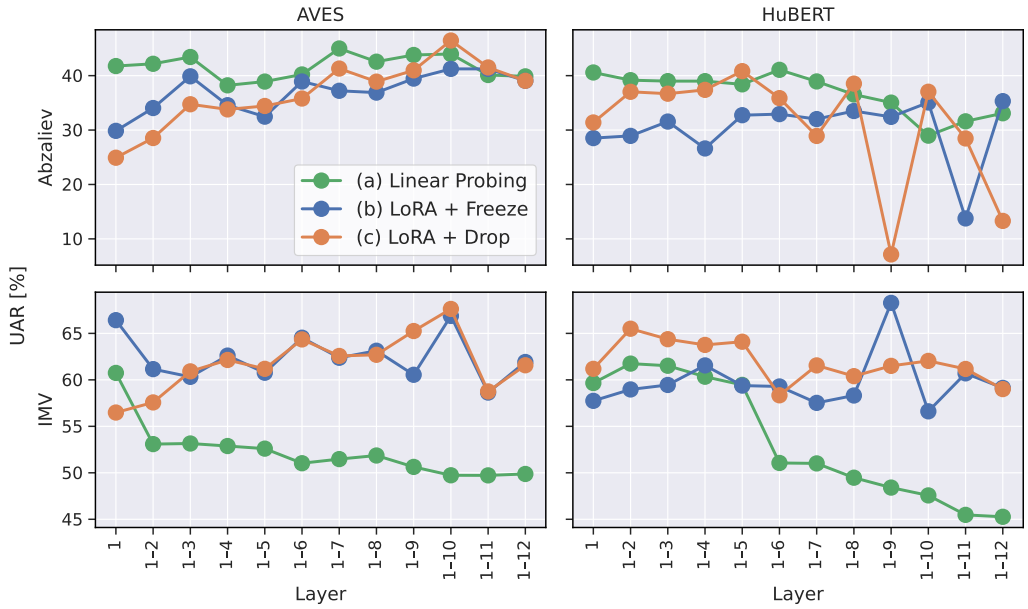

Figure 9: Layer-wise UAR [%] performance of scenarios (a), (b), and (c).

Figure 9 displays the per-layer UAR performance for each strategy. On the IMV dataset, LoRA fine-tuning, whether with freezing or pruning, consistently and significantly improves performance over simple linear probing across nearly all layers when using AVES, and shows clear gains in the later layers of HuBERT. By contrast, on the smaller Abzaliev dataset, simple linear probing almost always exceeds either LoRA performance, suggesting that LoRA tuning offers limited benefit in low-data scenarios. However, this performance gap on Abzaliev is modest compared to the substantial gains that LoRA fine-tuning delivers on the larger IMV dataset. This suggests that LoRA's advantages scale with dataset size, whereas in lower-data scenarios a simple linear probe may be more a reliable choice.

We can also observe that in the case of AVES, both LoRA-tuned models display a general *upward* trajectory for IMV and Abzaliev, whereas the linear probe continues to follow the same downward trend when going deeper in the layers, as seen in previous chapters. This demonstrates that deeper transformer layers in AVES encode increasingly abstract features that can effectively classify calls, but only when these layers are fine-tuned. A linear probe, which freezes the backbone, cannot leverage these deeper embeddings, and thus its performance declines in later layers. In contrast, LoRA injects a small number of trainable parameters into each layer, providing just enough task-specific flexibility to enable each additional layer to contribute positively, yielding a steady upward trend in performance. Practically, this implies that when extracting features from deeper layers within the transformer, one should pair them with parameter-efficient fine-tuning methods, such as LoRA, rather than relying on a fixed feature extractor alone.

4.5 Classifier Comparison: Linear Layer vs. MLP

The results obtained in the previous Section 4.4 can be directly compared, on the same datasets and feature representations, with those from Sarkar & Magimai.-Doss (2025)'s Section IIIA. In this chapter, we fine-tuned models using LoRA with a single linear output layer, as depicted in Figure 5, and compared them to a linear layer baseline. However, in previous chapters, we employed a MLP, composed of three blocks of [Linear, LayerNorm, ReLU] layers and a final linear layer, to evaluate various feature representations.

Figure 10 shows the highest scores of each scenario (a–c) from Figure 9, across all layers, alongside the corresponding MLP results from earlier chapters. This allows us to assess the potential benefit of classifier complexity, specifically, to see whether using a non-linear MLP really leads to better performance than a single linear layer.

We can observe that for the Abzaliev dataset, the MLP classifier clearly outperforms the single-layer LoRA variants, (b) and (c), for both AVES and HuBERT, suggesting that the added classifier capacity and non-linearity does help for CTID. However, for IMV, the opposite holds true: both single-layer LoRA models yield higher scores than the MLP classifier, indicating dataset-specific behavior.

Overall, these results do not allow us to draw general conclusions. While increased capacity may help in some cases, it may not be universally beneficial. Further investigation, such as fine-tuning a LoRA model with a non-linear MLP classifier, could give deeper insight into the impact of classifier capacity and non-linearity in this context.

5 Conclusions

In this chapter, we studied the potential of parameter-efficient fine-tuning (PEFT) for adapting large speech and bioacoustic SSLs models. We showed that Low-Rank Adaptation (LoRA) can greatly enhance call-type classification of animal vocalizations when sufficient labeled data is available. We systematically investigated a number of research directions by conducting a series of controlled experiments regarding LoRA adapter placements, layer selections, and fine-tuning strategies, and arrived at the following insights:

- Transformer encoder matrix selection: adapting an increasing subset of projection matrices yields steadily higher performance, with adaptation of entire self-attention and feed-forward projections achieving the best UAR.

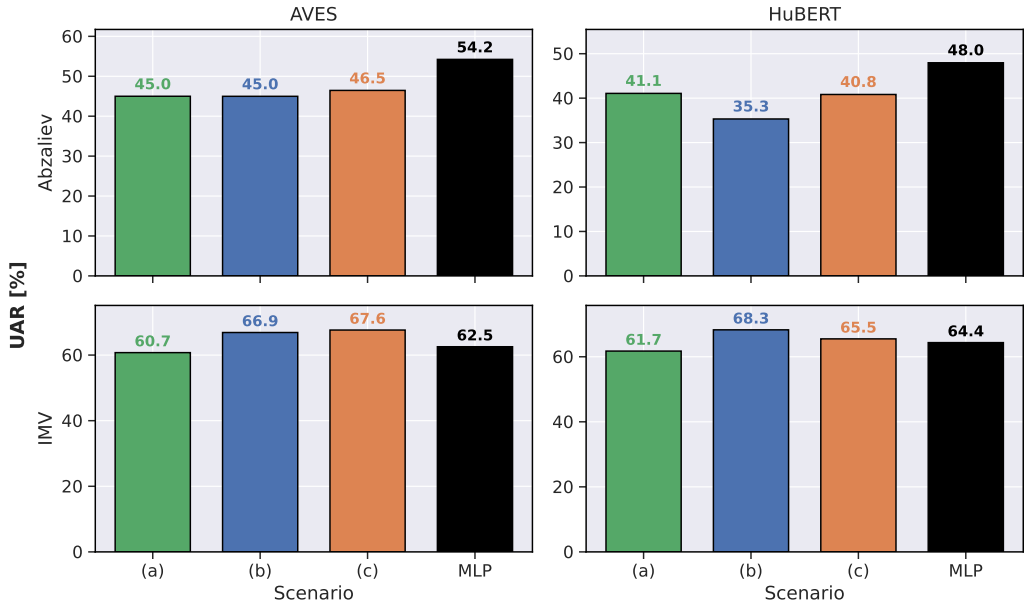

Figure 10: Best UAR results across layers for the (a), (b), and (c), scenarios defined in RQ4, using a linear layer classifier, compared to an MLP classifier.

- LoRA adapters scope: extending LoRA adapters beyond the Transformer encoder to the feature projection layer yields only marginal gains, whereas fine-tuning the convolutional feature extractor consistently and significantly degrades downstream performance.
- Layer selection strategy: neither 'bottoms-up' nor 'top-down' layer selection strategies clearly outperforms one another. Both produce comparable results when adapters are placed on the same matrices.
- Fine-tuning strategy: on the larger IMV dataset, LoRA fine-tuning (with either freezing or pruning) substantially outperforms simple linear probing across nearly all layers. In contrast, on the smaller Abzaliev dataset, simple linear probing remained more reliable, though the performance gap was modest. This indicates that LoRA's efficacy scale with dataset size.
- Classifier selection: LoRA adaptation with a single linear layer outperforms a deeper 4-layer MLP classifier head on IMV, while the reverse is seen for Abzaliev, indicating further investigation is needed to draw firm conclusions.

In conclusion, the overall results indicate that low-rank adaptation is a highly effective PEFT method and powerful tool for bioacoustic classification when ample data is available, enabling even deep transformer layers to contribute meaningfully. In low-data settings, however, a classic linear probe may still be preferable.

ACKNOWLEDGMENTS

This work was funded by Swiss National Science Foundation's NCCR Evolving Language project (grant no. 51NF40_180888).

REFERENCES

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Lin-*

guistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 7319–7328, Online, August 2021. Association for Computational Linguistics.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

F. Hutter, H. Hoos, and K. Leyton-Brown. An efficient approach for assessing hyperparameter importance. In *Proceedings of International Conference on Machine Learning 2014 (ICML 2014)*, pp. 754–762, June 2014.

Yifan Peng, Kwangyoun Kim, Felix Wu, Prashant Sridhar, and Shinji Watanabe. Structured pruning of self-supervised pre-trained models for speech recognition and understanding. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2023.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. On the effect of dropping layers of pre-trained transformer models. *Computer Speech & Language*, 77:101429, 2023. ISSN 0885-2308.

Eklavya Sarkar and Mathew Magimai.-Doss. Comparing self-supervised learning models pre-trained on human speech and animal vocalizations for bioacoustics processing. In *Proc. of ICASSP*, pp. 1–5, 2025.