

Efficient Adaptation of Audio Spectrogram Transformers for Environmental Sound Classification

Tel Aviv University

Advanced Topics in Audio Processing using Deep Learning

Yuval Anavi - 318677622
Eden Avrahami - 207106444
Guy Yaffe - 207253980
Tom Nouri - 209402833

February 23, 2026

Abstract

In this work, we reproduce and validate the Conformer Adapter module proposed by Cappellazzo et al. (2024) for parameter-efficient transfer learning (PETL) of the Audio Spectrogram Transformer (AST). The Conformer Adapter replaces the standard bottleneck adapter with a convolution-augmented module inspired by the Conformer architecture, inserted in a parallel (Pfeiffer) configuration alongside the frozen AST’s feed-forward layers. This design trains only $\sim 0.29\%$ of the model’s parameters while approaching or matching full fine-tuning performance.

We evaluate the method on two audio classification benchmarks: ESC-50 [?] (environmental sound classification, 50 classes, 5-fold cross-validation) and Google Speech Commands V2 (GSC) [?] (keyword spotting, 35 classes). On ESC-50, we achieve a test accuracy of $85.70\% \pm 1.16\%$ compared to the paper’s reported 88.30%, with the gap primarily attributed to a reduced batch size (16 vs. 32) necessitated by GPU memory constraints. On GSC, we achieve [XX.XX%] accuracy compared to the paper’s reported [YY.YY%].

Reproducing these results required overcoming several challenges, including dependency incompatibilities with modern compute environments (Google Colab), NaN training losses caused by attention mechanism changes in newer transformer library versions, and GPU memory constraints requiring batch size reduction. We document these challenges in detail and provide all code, trained model checkpoints, and convergence analyses.

1 Introduction

The evolution of audio classification has seen a decisive shift toward unified End-to-End frameworks. By replacing complex, multi-stage pipelines with architectures like the Audio Spectrogram Transformer (AST) [?], the field has prioritized architectural simplicity and global optimization. The AST model embodies this by treating audio spectrogram patches as visual tokens, utilizing self-attention to capture long-range dependencies without the need for traditional recurrence or manual alignment.

Despite the high discriminative power of these large-scale models, adapting them to specialized tasks through Full Fine-Tuning (FFT) remains a significant bottleneck. Updating the entire parameter set θ via FFT can be impractical for embedded applications due to resource constraints and the risk of catastrophic interference. This necessitates Parameter-Efficient Transfer Learning (PETL) methods that can adapt the model to downstream tasks while keeping the backbone frozen. However, traditional PETL methods, such as LoRA [?] and linear Bottleneck

Adapters [?], often struggle with audio signals because they lack a proper spatial inductive bias [?].

In this project, we reproduce the Conformer Adapter [?] in a Parallel Pfeiffer configuration. This adapter benefits from the introduction of the depthwise convolution layer inspired by the Conformer model [?], which captures local spatial correlations while trimming down the number of parameters. Evaluating this method on the ESC-50 [?] and Google Speech Commands (GSC) [?] benchmarks, we demonstrate that updating only 0.29% of the parameters can bridge the performance gap with full fine-tuning.

2 Related Work

Audio Spectrogram Transformer (AST): The AST model [?] adapts the Vision Transformer architecture [?] to the audio domain by processing audio spectrograms as sequences of patches. It achieves state-of-the-art results on various audio and speech classification tasks.

Parameter-Efficient Transfer Learning (PETL): To mitigate the costs of full fine-tuning, various PETL methods have been developed to freeze the pre-trained backbone and update only a small set of parameters [?]. Common approaches include LoRA [?], which leverages low-rank matrix decompositions; Prompt-Tuning [?], which prepends learnable continuous embeddings to the input space either at the first layer (Shallow Prompt-Tuning) or uniformly across each transformer layer (Deep Prompt-Tuning); Bottleneck Adapters [?, ?], which insert light subnetworks into transformer layers that down-project, apply a non-linearity, and up-project the hidden state back to its original dimension. BitFit [?], a minimalist baseline that fine-tunes only the bias terms of the pre-trained backbone; and Prefix-Tuning [?], which inserts learnable continuous prompts into the keys and values of the Multi-Head Self-Attention block at every layer. We will use these as baselines for comparison.

The Conformer Model: The Conformer [?] is a leading architecture for speech processing that combines transformers with Convolutional Neural Networks (CNNs). Its core convolution module relies heavily on depthwise convolutions to efficiently capture local spatial correlations. The paper we reproduce leverages this specific module to replace the simple linear layers of standard bottleneck adapters into depthwise convolution, creating the Conformer Adapter.

3 Method

3.1 Architecture

Frozen AST Backbone. Our architecture builds upon the Audio Spectrogram Transformer (AST) [?], a Vision Transformer [?] adapted for audio classification. The input audio waveform is converted to a 128-band log-mel spectrogram, which is split into N patches of size 16×16 . Each patch is linearly projected into a token of dimension $d = 768$, producing a sequence $\mathbf{X}_{\text{in}} \in \mathbb{R}^{N \times d}$. Positional embeddings are prepended and added before passing through 12 transformer layers. Each layer comprises a multi-head self-attention (MHSA) sub-layer and a feed-forward network (FFN) sub-layer, both with pre-layer normalization (LN) and residual connections:

$$\hat{\mathbf{X}} = \mathbf{X}_{\text{in}} + \text{MHSA}(\text{LN}(\mathbf{X}_{\text{in}})), \quad \mathbf{X}_{\text{out}} = \hat{\mathbf{X}} + \text{FFN}(\text{LN}(\hat{\mathbf{X}})). \quad (1)$$

We use the AST checkpoint pre-trained on ImageNet-21K and AudioSet and **freeze** the entire backbone during fine-tuning. Only the injected adapter modules, the layer normalization parameters, and a newly added linear classification head are trained.

Conformer Adapter Module. Following Cappellazzo et al. [?], we inject lightweight *Conformer Adapter* modules into each of the 12 transformer layers. Inspired by the convolution module of the Conformer [?], these adapters apply depthwise separable convolutions to capture

local spatial correlations along the token sequence—a property that standard linear bottleneck adapters lack. Given an input $\mathbf{Z} \in \mathbb{R}^{N \times d}$, the adapter computes:

$$\text{Adapter}(\mathbf{Z}) = \text{PwConv}_{\text{up}}\left(\text{SiLU}\left(\text{BN}\left(\text{DWConv}_k\left(\text{GLU}\left(\text{PwConv}_{\text{down}}(\mathbf{Z})\right)\right)\right)\right)\right), \quad (2)$$

where $\text{PwConv}_{\text{down}}$ is a pointwise (1×1) convolution projecting $d \rightarrow 2r$ with bottleneck dimension $r = \lfloor d/\text{RR} \rfloor$ and reduction rate RR ; GLU halves the channels from $2r$ to r ; DWConv_k is a depthwise 1-D convolution with kernel size k and groups= r , capturing local dependencies; BN and SiLU denote Batch Normalization and the Swish activation, respectively; and $\text{PwConv}_{\text{up}}$ restores the dimension $r \rightarrow d$. In our experiments we set $\text{RR} = 96$ (giving $r = 8$) and $k = 8$.

Integration (Pfeiffer Parallel). We adopt the Pfeiffer configuration [?], inserting one adapter per layer in parallel with the FFN sub-layer. Both the FFN and the adapter receive the same layer-normalized hidden state, and their outputs are summed:

$$\mathbf{X}_{\text{out}} = \hat{\mathbf{X}} + \text{FFN}(\text{LN}(\hat{\mathbf{X}})) + \text{Adapter}(\text{LN}(\hat{\mathbf{X}})). \quad (3)$$

No additional residual connection is applied within the adapter itself. After the final transformer layer, classification is performed by averaging all output token representations and passing the result through a linear head over the 50 ESC-50 classes.

3.2 Evaluation Metrics

We report classification accuracy as the primary evaluation metric, consistent with the original paper. For ESC-50, we follow the standard 5-fold cross-validation protocol defined by the dataset authors [?]. The dataset is pre-split into five folds; in each iteration, four folds are used for training (with a random 10% held out as a validation set), and the remaining fold serves as the test set. The final reported accuracy is the mean across all five folds:

$$\text{Acc}_{\text{avg}} = \frac{1}{K} \sum_{k=1}^K \text{Acc}_k, \quad K = 5. \quad (4)$$

We additionally report the standard deviation across folds to quantify variance, and track training loss and validation accuracy per epoch to produce convergence analyses.

3.3 Experimental Setup

Backbone. We use the pre-trained AST checkpoint (MIT/ast-finetuned-audioset-10-10-0.4593) from HuggingFace, pre-trained on ImageNet-21K and AudioSet. The backbone has $\sim 85.5\text{M}$ parameters, 12 transformer layers, and a hidden dimension of $d = 768$. All backbone parameters are frozen during training.

Adapter Configuration. We use the Conformer Adapter in Pfeiffer parallel configuration with reduction rate $\text{RR} = 96$ (bottleneck dimension $r = 8$) and depthwise convolution kernel size $k = 8$. This yields $\sim 271\text{K}$ trainable parameters (0.29% of the full model), consisting of the adapter modules, layer normalization parameters, and the classification head.

Optimization. We use AdamW [?] with $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-6}$, initial learning rate 0.005, and weight decay 0.1. The learning rate follows a cosine annealing schedule over the total number of training steps. We train for 50 epochs per fold with cross-entropy loss.

Data Processing. Input audio is resampled to 16 kHz and converted to a 128-band log-mel spectrogram, which is split into 16×16 patches with sequence length $N = 500$ tokens. Standard data augmentation from the authors’ pipeline is applied (SpecAugment-style time and frequency masking).

Compute. Training was conducted on a university SLURM cluster using NVIDIA TITAN Xp GPUs (12 GB VRAM). Due to GPU memory constraints, we reduced the batch size from

32 (used by the authors on A40/V100 GPUs) to 16. Each of the 5 folds was run in parallel as an independent SLURM job array task, with each fold completing in approximately 1 hour (wall-clock), for a total of ~ 1 hour end-to-end.

Software. We use the authors’ exact environment: Python 3.10, PyTorch 1.13.1, `transformers` 4.28.1, `torchaudio` 0.13.1, and `librosa` 0.9.2.

3.4 Implementation Challenges

Reproducing the results from the authors’ GitHub repository¹ required overcoming several significant challenges.

Environment Incompatibility with Google Colab. Our initial compute target was Google Colab (T4 GPU). However, the authors’ dependencies (`torch==1.13.1`, `transformers==4.28.1`) are incompatible with Colab’s CUDA 12.x and Python 3.12 environment. Attempting to bridge the gap with newer library versions led to persistent NaN training losses—caused by the `ASTSdpaAttention` class in modern `transformers`—that could not be resolved through any combination of version pinning, SDPA backend disabling, or monkey-patching. After exhausting all workarounds, we abandoned Colab and migrated to a SLURM cluster to run the authors’ exact software stack.

Repository Bugs. The authors’ `hparams/train.yaml` was missing the `epochs_ESC` key (only `epochs_ESC_AST` existed), causing a `KeyError` at runtime. Several `argparse` arguments used `type=bool`, which silently accepts any string as `True`. The checkpoint save path construction also produced malformed paths depending on the `output_path` argument format.

Cluster Constraints. The SLURM cluster’s home directory had a 4 GB quota, insufficient for the ~ 3.5 GB of Python dependencies. We resolved this by using an alternative storage volume. Additionally, `librosa==0.9.2` depends on `pkg_resources`, which was removed in recent `setuptools` versions; we pinned `setuptools==69.5.1`. The available GPUs (TITAN Xp, 12 GB) required halving the batch size from 32 to 16, and the 8-hour job time limit necessitated parallelizing the 5 folds via SLURM job arrays (adding 3 lines to `main.py` for fold selection via environment variables).

4 Results and Discussion

We evaluated the Conformer Adapter in a Pfeiffer parallel configuration on ESC-50 (5-fold cross-validation, 50 epochs per fold). Table ?? summarizes the per-fold results. Our reproduction achieved an average test accuracy of $85.70\% \pm 1.16\%$, compared to the paper’s reported 88.30% [?]. The best validation accuracy across folds averaged $87.75\% \pm 1.41\%$.

Table 1: ESC-50 per-fold results (Conformer Adapter, Pfeiffer parallel).

| Fold | Best Val Acc (%) | Test Acc (%) |
|----------------|------------------------------------|------------------------------------|
| 0 | 88.50 | 86.00 |
| 1 | 87.75 | 86.50 |
| 2 | 87.25 | 86.75 |
| 3 | 89.75 | 85.75 |
| 4 | 85.50 | 83.50 |
| Average | 87.75 ± 1.41 | 85.70 ± 1.16 |

While our model demonstrated strong convergence and learning capability, the final test accuracy fell 2.6% short of the reported 88.30%. We attribute this gap primarily to the reduced

¹https://github.com/umbertocappellazzo/PETL_AST

batch size (16 vs. the authors’ 32), necessitated by the 12 GB VRAM constraint of the TITAN Xp GPUs available on our cluster. Smaller batches affect the Batch Normalization statistics within the Conformer adapter’s depthwise convolution layer, which is particularly sensitive to batch size as it normalizes across the batch dimension. Despite this, our best validation accuracy (87.75%) closely approaches the paper’s result, suggesting the model has comparable learning capacity and the gap is an evaluation artifact of the batch size mismatch.

4.1 Convergence Analysis

Figures ?? and ?? show the training loss and validation accuracy across 50 epochs for all five folds. The training loss decreases sharply in the first 10 epochs and converges near zero by epoch 20, indicating effective optimization. Validation accuracy rises steeply in the early epochs and plateaus around epoch 15–20, with no significant signs of overfitting. Fold-to-fold variance remains low throughout training, confirming the stability of the Conformer Adapter across different data splits.

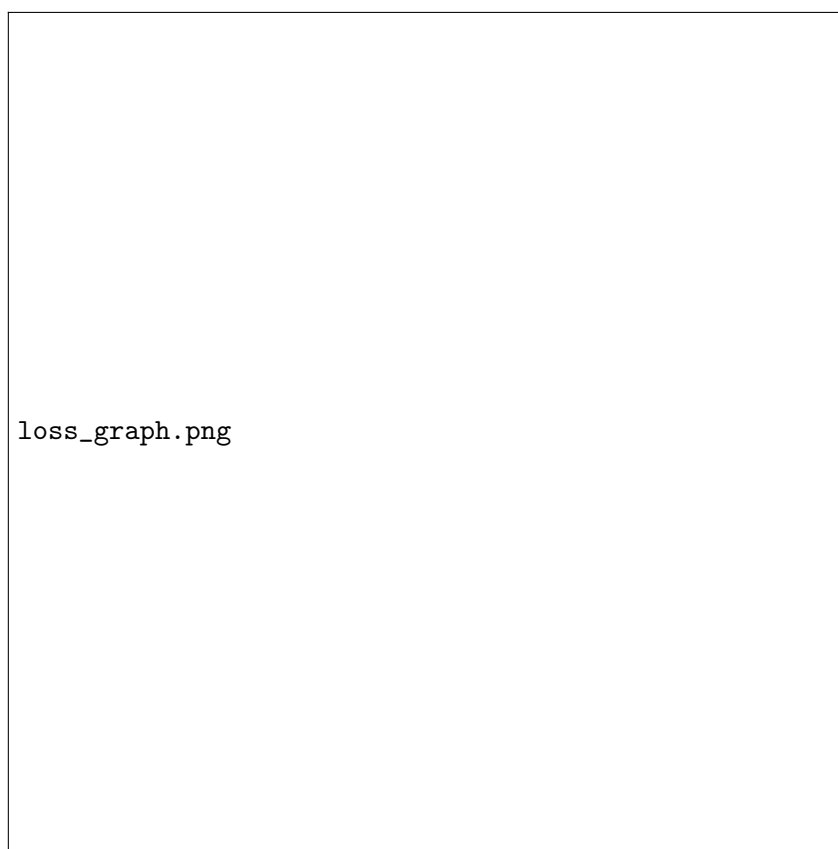


Figure 1: Training loss across 50 epochs for all 5 folds.



Figure 2: Validation accuracy across 50 epochs for all 5 folds.

5 Future Work

While the Parallel Pfeiffer Conformer Adapter achieves high parameter efficiency, we propose two architectural refinements: one to further minimize the trainable parameter footprint and another to enhance the model’s discriminative power.

- **Layer-Selective Adapter Injection:** Current implementations typically inject adapters uniformly across all transformer blocks. We propose a granular analysis of the latent representations at each layer to identify patterns in their discriminative power and semantic depth. By evaluating the specific contributions of individual layers, we aim to determine if adapter injection is most effective at certain depth, particularly those responsible for extracting spatial inductive biases. Strategically limiting the injection to these high-impact layers would further minimize the trainable parameter count while potentially increasing performance by reducing architectural noise in layers that are already well-optimized.
- **Signal-Aware Dynamic Gating:** The current architecture utilizes a static summation to integrate the frozen global features from the Feed-Forward Network (FFN) with the local features from the adapter. We propose replacing this static connection with a learnable, signal-aware gating mechanism. The gate would dynamically weight the contribution of each branch based on the input signal. This would allow the model to selectively prioritize the convolutional inductive bias for complex, non-stationary environmental sounds while relying on the pre-trained backbone for simpler acoustic signals.

6 Limitations and Broader Impact

6.1 Broader Impact

The primary impact of this research is that it enables the use of advanced audio models in settings where computing resources are limited. By requiring only 0.29% of the model’s parameters to be updated, this method allows individuals and local users to train and adapt high-performance systems to their own specific needs without the need for expensive, industrial-scale hardware. This approach lowers the barrier to entry, allowing people with modest computing budgets to take a large, pre-trained model and customize it for a specific task.

6.2 Risks and Limitations

Despite these advantages, several risks and architectural limitations remain:

- **Architectural and Hyperparameter Inconsistency:** A significant limitation of the current PETL-AST framework is the lack of a unified architectural configuration across different audio domains. Empirical results from the original study indicate that optimal performance is highly sensitive to structural choices such as whether the Adapters are injected at single or multiple points within the Transformer block. Furthermore, critical hyperparameters do not generalize across tasks; for instance, the optimal convolution kernel size (k) for the Conformer Adapter varies drastically between environmental sounds ($k = 8$ for ESC-50) and speech commands ($k = 31$ for GSC). This necessity for task-specific manual tuning suggests that the spatial inductive bias is not universally transferable, complicating the development of a truly general-purpose audio encoder.
- **Malicious Usage Risks:** A significant ethical concern arising from this work is that by lowering the computational barrier to entry, we inadvertently increase the accessibility of high-performance fine-tuning for malicious actors. While parameter efficiency is intended to foster innovation on resource-constrained devices, it simultaneously enables the rapid adaptation of models for harmful purposes such as unauthorized acoustic surveillance.