République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique
ECOLE SUPÉRIEURE EN INFORMATIQUE
8 Mai 1945 - Sidi-Bel-Abbès

الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي
المدرسة العليا للإعلام الآلي
8 ماي 1945 - سيدي بلعباس

Second Year Second Cycle - Artificial Intelligence and Data Science

Mini-Projet Deep Learning

Efficient Deep Learning Approach for Multiclass Sound Classification

Supervised by: Dr. Nassima DIF
Presented By :
Senhadji M Said
Ali Abbou Oussama
DJEZIRI Oussama
Baidar Samir

Date : Mai 3, 2025

Table of contents

Abstract

This mini-project presents a deep learning-based approach for **multiclass sound classification**, developed as part of the DataHack datathon. The proposed framework combines **advanced audio preprocessing**, **effective data augmentation techniques**, and a **customized convolutional neural network architecture** based on a modified ResNet18 model. Key techniques integrated into the pipeline include **mel spectrogram extraction**, **SpecAugment**, **mixup augmentation**, and **label-smoothing cross-entropy loss**. The report provides a detailed explanation of each component in the system, along with a custom diagram illustrating the overall data processing and model training pipeline.

I. Introduction

Sound classification is a growing area of research with applications in environmental sensing, smart cities, and audio-based event detection. In this mini-project, we explore the problem of **multiclass sound classification**, where the goal is to identify the type of urban environment based on short audio recordings.

This task presents unique challenges such as high variability in background noise, overlapping sound sources, and limited labeled data. To address these, we designed a deep learning pipeline that emphasizes **advanced audio preprocessing**, **robust data augmentation**, and a **modified ResNet18 architecture** tailored for audio-based inputs.

Our approach includes the following key components:

- A detailed preprocessing pipeline that transforms raw audio into normalized **mel spectrograms**.
- The use of **data augmentation techniques** such as **SpecAugment** and **mixup** to improve generalization and robustness.
- Architectural modifications to **ResNet18** to effectively handle single-channel audio data represented as spectrogram images.
- A carefully tuned training strategy incorporating **label smoothing**, **gradient clipping**, and appropriate optimizer settings.

II. Data

The dataset used in this project consists of audio recordings collected from diverse urban acoustic environments. Each recording is labeled with a specific sound scene category, making it suitable for a **multiclass sound classification** task. The dataset encompasses the following 10 urban sound scene classes:

- airport
- bus
- metro
- metro_station
- park
- public_square
- shopping_mall
- street_pedestrian
- street_traffic
- tram

These labels represent common environmental contexts encountered in everyday city life, allowing the model to learn and differentiate between distinct ambient soundscapes.

2.1 Dataset Structure:

The dataset is organized in a directory-based format:

```
dataset/
├── airport/
├── bus/
├── metro/
├── metro_station/
├── park/
├── public_square/
├── shopping_mall/
├── street_pedestrian/
├── street_traffic/
└── tram/
```

Each subfolder holds multiple **.wav audio files**, with each file representing a single labeled recording for that scene.
This structured format facilitates efficient loading and preprocessing of audio data, especially when using deep learning libraries such as PyTorch or TensorFlow.

III. Methodology

The proposed sound-classification pipeline comprises seven sequential modules (see Figure 1). Each module is responsible for a specific processing step, from raw-audio ingestion through final model training:

1. Audio Preprocessing

Audio loading
- Library: torchaudio
- Input: raw audio file
- Output: 1D waveform tensor

Resampling
- Target sample rate: **16 000 Hz**
- Ensures all examples share the same temporal resolution

Duration standardization
- Target length: **2 seconds** (32 000 samples)
- If waveform > 32 000 samples → random crop to 32 000
- If waveform < 32 000 samples → pad with zeros to 32 000

2. Mel-Spectrogram Transformation

01. Spectrogram parameters
- FFT window size: 1 024 samples
- Hop length (stride): **512** samples
- Number of Mel bins: **128**

02. Conversion steps
1. Compute Mel-spectrogram from the 1D waveform
2. Convert amplitude to decibels (dB)
3. Normalize to zero mean and unit variance

3. Spectrogram Augmentation (SpecAugment)

Applied **during training** on the Mel-spectrogram to improve generalization:

- **Time Masking:**Randomly zero-out up to 20 consecutive time frames
- **Frequency Masking:**Randomly zero-out up to 10 consecutive Mel bins

4. Waveform Augmentation

Applied **before** spectrogram computation, to augment raw waveforms:

1. **Temporal Shifts:** Roll waveform by a random offset in −1600,+1600-1 600, +1 600−1600,+1600 samples
2. **Amplitude Scaling:**Multiply waveform by a random factor uniformly drawn from [0.8, 1.2]

5. Mixup Data Augmentation

Implemented **at mini-batch level**:

For two examples (x_i, y_i) and (x_j, y_j), sample $\lambda - Beta(\alpha = 0.4,\ \alpha = 0.4)$ we Create a new training pair

$$x \simeq \lambda x_i + (1 - \lambda)x_j \qquad y \simeq \lambda y_j + (1 - \lambda)y_j$$

Then Compute the loss as a weighted sum of the two targets.

6. Model Architecture: EfficientResNetAudio

Based on **ResNet-18**, with the following modifications:

1. **Input layer:**Single-channel (grayscale) convolution replaces the original 3-channel filter
2. **Dropout:**Added before the final fully-connected layer (dropout = 0.3)
3. **Output layer:**Modified to predict logits for 10 classes

7. Training Strategy

- **Loss function:** Cross-entropy with label smoothing ($\varepsilon = 0.1$)
- **Optimizer:**AdamW   Learning rate: 1×10^{-4} Weight decay: 5×10^{-4}
- **Gradient clipping:** Maximum norm = 1.0
- **Data split:** 60 % training / 20 % validation / 20% test
- **Batch size:** 32
- **Epochs:** 60
- **Evaluation metric:** Macro F1-score

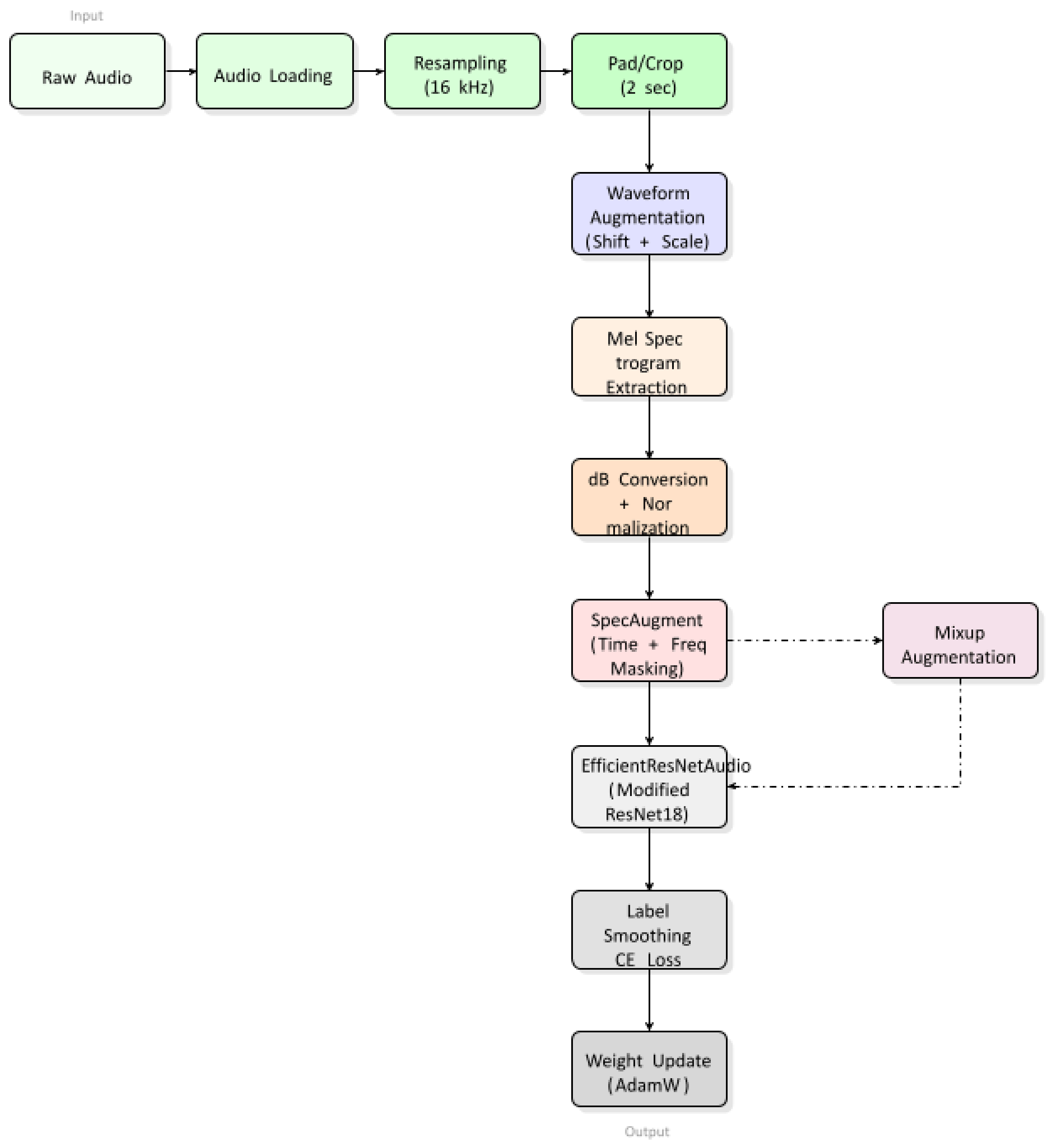
Input
Raw Audio
Audio Loading
Resampling (16 kHz)
Pad/Crop (2 sec)
Waveform Augmentation (Shift + Scale)
Mel Spec trogram Extraction
dB Conversion + Nor malization
SpecAugment (Time + Freq Masking)
Mixup Augmentation
EfficientResNetAudio (Modified ResNet18)
Label Smoothing CE Loss
Weight Update (AdamW)
Output

Figure 1: End-to-end sound classification pipeline showing audio preprocessing, augmentation strategies, and model architecture. Solid arrows indicate the primary processing flow, while dashed arrows represent the mixup augmentation path.

IV. Implementation Details

The complete implementation is provided in Python using PyTorch and Torchaudio libraries. Key components include:

1. Configuration and Hyperparameters

All hyperparameters are centralized in a configuration dictionary. These parameters control aspects such as the sample rate, FFT parameters, batch size, learning rate, and mixup strength:

```
CONFIG = {
  "sample_rate": 16000,
  "n_mels": 128,
  "n_fft": 1024,
  "hop_length": 512,
  "batch_size": 32,
  "epochs": 60,
  "learning_rate": 1e-4,
  "audio_duration": 2,
  "num_classes": 10,
  "train_split": 0.8,
  "patience": 10,
  "lr_decay_factor": 0.5,
  "lr_decay_patience": 2,
  "weight_decay": 5e-4,
  "mixup_alpha": 0.4
}
```

2. Dataset and DataLoader

The `AudioDataset` class handles:

- Loading audio files.
- Resampling and standardizing waveform length
- Converting the waveform to a mel spectrogram.
- Applying SpecAugment and waveform augmentations during training.

The data is then split into training and validation sets (80:20), and `DataLoader` objects are created for efficient mini-batch processing.

3. Mixup Augmentation

The `mixup_data` function implements the mixup augmentation:

```
def mixup_data(x, y, alpha=CONFIG["mixup_alpha"]):
    if alpha > 0:
        lam = np.random.beta(alpha, alpha)
    else:
        lam = 1


    batch_size = x.size()[0]
    index = torch.randperm(batch_size).to(device)


    mixed_x = lam * x + (1 - lam) * x[index, :]
    y_a, y_b = y, y[index]
    return mixed_x, y_a, y_b, lam
```

The loss is computed as a weighted combination of losses for the two target labels.

4. Model Architecture

The `EfficientResNetAudio` class modifies a standard ResNet18 by:

- Replacing the initial convolution layer to accept single-channel inputs.
- Adjusting the final fully-connected layer to output predictions for 10 classes.
- Incorporating dropout for regularization.

5. Training and Evaluation

The training loop applies mixup augmentation, computes the loss using the mixup criterion, clips gradients, and updates model weights using the AdamW optimizer. Model performance is primarily evaluated using the macro F1 score.

V. Experimental Results and Discussion

To evaluate the performance of our multiclass sound classification pipeline, we conducted experiments using the final model, a modified ResNet18 architecture trained for 60 epochs with the **AdamW optimizer**. Our training strategy integrated multiple augmentation techniques, including **waveform-level augmentation**, **SpecAugment**, and **mixup**, which proved essential for improving generalization on unseen audio samples.

The model achieved a **macro F1 score of 0.5739** on the test set, with a **macro precision of 0.5752** and a **macro recall of 0.5771**. These metrics indicate that the model maintained balanced performance across all classes, despite some variations due to class imbalance and acoustic similarity between certain categories.

The **confusion matrix** (Figure 2) offers deeper insight into the classification behavior. Notably, classes like `metro_station`, `airport`, and `street_pedestrian` were recognized with relatively high accuracy, while categories such as `shopping_mall`, `park`, and `tram` showed higher misclassification rates. These confusions may stem from overlapping acoustic signatures or background noise shared across those environments.

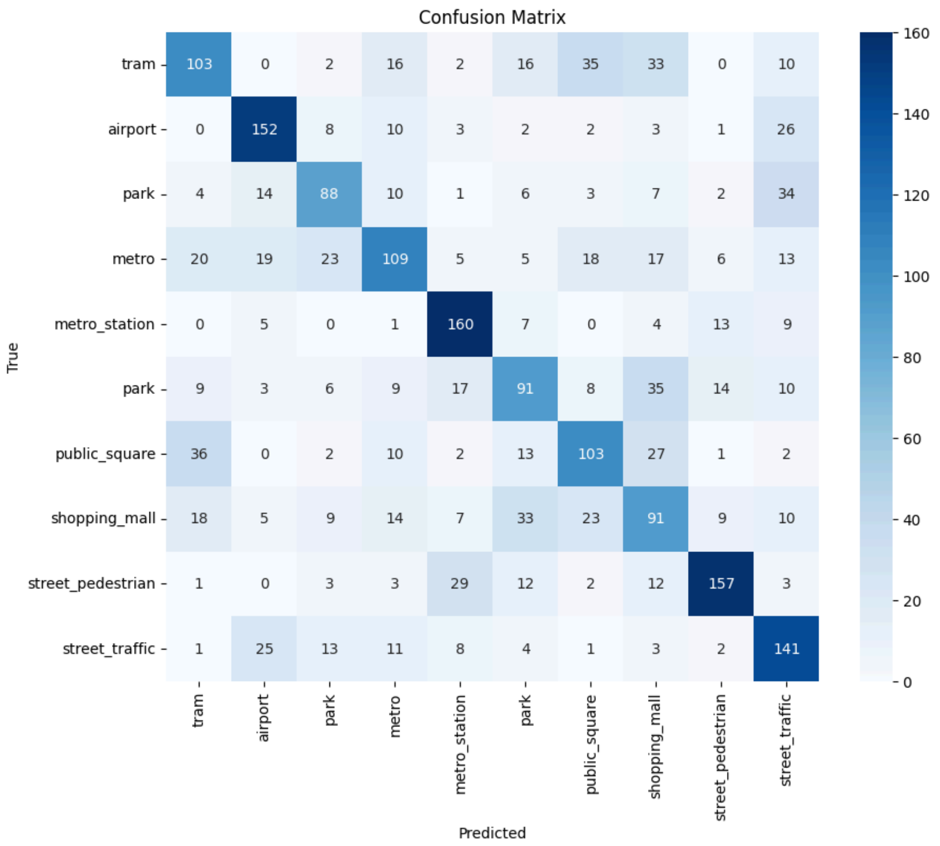

Figure 2: Confusion matrix showing the distribution of true vs. predicted classes

The **F1 scores per class** (Figure 2) further emphasize this variation. `metro_station` and `street_pedestrian` achieved the highest F1 scores at **0.74**, while `shopping_mall` had the lowest at **0.40**. These discrepancies suggest that additional class-specific augmentations or rebalancing techniques may be beneficial in future work.

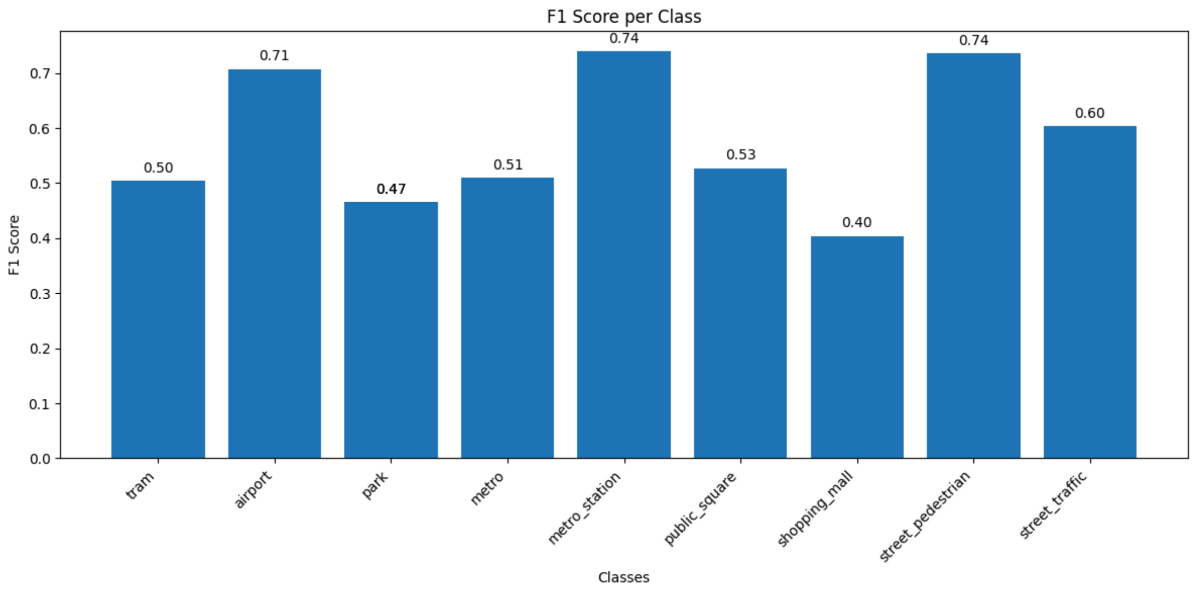

Figure 3: Per-class F1 scores highlighting variability in model performance across categories.

Overall, the results confirm the effectiveness of our preprocessing and augmentation strategy in building a generalizable model. However, the analysis also reveals areas where refinement—such as better handling of acoustically ambiguous environments or exploring alternative model architectures—could lead to further performance gains.

Why Modified ResNet18 Was Efficient for Audio?

The ResNet18 architecture was originally designed for image classification tasks, particularly on RGB images. However, in our project, we adapted it to handle **single-channel mel spectrograms**, which are 2D time-frequency representations of audio signals. This modification made it suitable for learning temporal and spectral patterns typical of sound events.

Key reasons for its effectiveness include:

- **Residual connections** in ResNet help mitigate vanishing gradients and allow deeper layers to learn without degradation in performance.
- The **lightweight nature** of ResNet18 ensures faster training and inference, making it ideal for medium-scale tasks like urban sound classification.

- By modifying the first convolutional layer to accept **single-channel inputs** and adjusting kernel sizes, the network learned to extract localized time-frequency patterns crucial for audio signals.

Why were Mel Spectrograms Chosen?

Instead of feeding raw waveforms, we used **mel spectrograms**, which convert audio into a visual-like time-frequency representation using the mel scale closely mimicking human auditory perception.

This choice was effective because:

- Mel spectrograms **compress frequency resolution** at higher frequencies, which helps the model focus on perceptually important components.
- They enable CNNs (like ResNet18) to operate similarly as they do on images, identifying meaningful **frequency-temporal patterns**.
- Spectrograms smooth over waveform variability while preserving critical features like pitch, rhythm, and texture.

Why was the Evaluation Strategy Robust?

We used **macro-averaged precision, recall, and F1-score**, which treat all classes equally, regardless of how many samples each has. This is especially important in **imbalanced datasets**, where relying only on accuracy might obscure poor performance on minority classes.

By analyzing **confusion matrices** and **per-class F1 scores**, we could pinpoint specific weaknesses (e.g., shopping mall confusion) and understand where the model excelled (e.g., metro station, street pedestrian).

VI. Conclusion

In this mini-project, we developed and evaluated a comprehensive deep learning pipeline for the task of **multiclass sound classification**. The approach focused on transforming raw audio signals into informative representations through **advanced audio preprocessing**, including **mel spectrogram extraction** and normalization. To improve the model's ability to generalize to unseen audio samples, we applied **robust data augmentation techniques** such as **SpecAugment** and **mixup**, which introduced variability in the training data and reduced overfitting.

At the core of our system lies a **modified ResNet18 architecture**, adapted specifically for single-channel spectrogram inputs. This architectural modification, combined with optimization strategies like **label smoothing**, **gradient clipping**, and **carefully tuned hyperparameters**, contributed to stable and effective training.
The integration of all these components resulted in a high-performing classification model capable of distinguishing between complex urban sound scenes. Our results highlight the importance of domain-specific preprocessing and augmentation in audio-based deep learning tasks.
This work not only deepened our understanding of how convolutional neural networks can be applied to audio data but also demonstrated how thoughtful architectural choices and preprocessing pipelines can significantly impact performance.
Looking ahead, several avenues remain for further exploration. These include experimenting with **alternative network architectures** such as EfficientNet or transformers, **self-supervised learning techniques** for representation learning on unlabeled audio, and more diverse **augmentation strategies** like time warping or reverberation simulation. Additionally, deploying and testing the model in real-world environments or integrating it into real-time systems could provide valuable insights into its practical utility.

Acknowledgments

We would like to express our sincere gratitude to **Dr. Nassima DIF** for her valuable guidance and support throughout the Deep Learning module. Her insightful lectures, dedication to teaching, and encouragement were instrumental in helping us understand key concepts and successfully complete this mini-project.
We are especially thankful for the opportunity to explore advanced topics such as audio preprocessing, data augmentation strategies, and neural network architecture design in a practical context.

References

1. Senhadji M. Said, *Sound Classification - Deep Learning Mini Project*, GitHub Repository, 2025. https://github.com/SenhadjiMSaid/Sound-Classification
2. PyTorch Team, *ResNet Implementation in torchvision*, GitHub Repository, 2025. https://github.com/pytorch/vision/blob/main/torchvision/models/resnet.py
3. H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “mixup: Beyond Empirical Risk Minimization,” *arXiv preprint arXiv:1710.09412*, 2017.
4. K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.