## **Infant Cry Based audio classification**

## **Authors**

**Eli Benyaminov, Maor Tzalisher***(Intelligent systems, Afeka engineering College, Tel Aviv, Israel)  
(*[*Elibenyaminov@gmail.com*](mailto:Elibenyaminov@gmail.com)*, Tzalisher01*[*@gmail.com*](mailto:Elibenyaminov@gmail.com)*)*

## **Keywords**

Audio Feature extraction, Audio Classification, Deep Learning, Multi-Class Classification, SMOTE, PCA, LDA, MLP, SVM

## **Abstract**

In this project, we investigate a deep learning approach for audio classification using a dataset of audio files. We prepare the data by applying a series of preprocessing steps, such as cleaning, feature extraction, dimensionality reduction, and class balancing. Multiple machine learning algorithms are tested, including a baseline Support Vector Machine (SVM) and a Multi-Layer Perceptron (MLP) neural network, followed by hyperparameter tuning. We further compare the impact of SMOTE for balancing classes and Linear Discriminant Analysis (LDA) for dimensionality reduction. Our best-performing model, a tuned MLP, achieves an accuracy of **56.52%**, an **F1-score of 34.71%**, and an **ROC-AUC of 72.26%** on the test set. Our findings highlight the potential of deep learning for robust audio classification while underscoring the challenges posed by class imbalance and high feature dimensionality. Future work should explore alternative deep architectures, such as CNNs, and advanced feature extraction techniques to enhance classification performance.

## **1.Introduction**

The proliferation of audio data in today’s digital landscape has dramatically increased the need for effective automatic audio classification systems. Whether one is dealing with speech recognition, music genre classification, environmental sound detection, or a myriad of other audio-related tasks, the fundamental challenge is to interpret and categorize complex acoustic signals accurately. Traditional approaches often rely on hand-crafted features—such as Mel-Frequency Cepstral Coefficients (MFCCs) or spectral representations—to capture salient aspects of the sound. These features can be effective, but they also place a heavy burden on domain expertise and may not generalize well to novel audio contexts. By contrast, deep learning methods attempt to learn hierarchical feature representations directly from raw signals or partially processed data, thereby reducing reliance on extensive feature engineering.

Despite these advances, practitioners still face multiple obstacles. First, many audio datasets exhibit significant class imbalance, where certain sound categories are well-represented while others are severely under-sampled. This imbalance can skew model training toward majority classes and limit the overall system’s ability to detect minority or rare classes. Techniques such as the Synthetic Minority Over-sampling TEchnique (SMOTE) can help mitigate this problem by generating synthetic samples for under-represented classes, but the effectiveness of SMOTE can vary depending on the dataset’s complexity and the nature of the minority classes. Second, the dimensionality of audio features can be extremely high—especially when dealing with spectrograms or raw waveforms—leading to computational challenges and the risk of overfitting. Dimensionality reduction methods like Principal Component Analysis (PCA) or Linear Discriminant Analysis (LDA) can help, but each technique has its assumptions and trade-offs. LDA, for example, seeks linear boundaries between classes and can be particularly suitable for tasks with well-defined class distinctions, though it requires at most one fewer dimension than the number of classes.

Another issue is the difficulty of generalizing across diverse audio sources. Models trained on a particular domain (e.g., musical instrument sounds) may struggle when confronted with different recording conditions, environmental noises, or entirely new sound categories. This underscores the importance of systematic experimentation, hyperparameter tuning, and robust evaluation metrics. A model that achieves high accuracy in a controlled setting may fail to perform reliably in real-world scenarios if it has not been tested across varying conditions.

Given these challenges, the objective of this project is fourfold. First, we aim to build a coherent pipeline that classifies audio samples into multiple classes, ensuring that the workflow from data loading to final evaluation is both reproducible and scalable. Second, we compare baseline machine learning methods—such as Support Vector Machines (SVM)—with deep neural networks (Multi-Layer Perceptrons, or MLPs), thereby assessing how classical techniques stack up against more modern, representation-learning approaches. Third, we explicitly evaluate data-balancing strategies by applying SMOTE to handle class imbalance and reduce its adverse effects on minority classes. Fourth, we incorporate dimensionality reduction, specifically LDA, to address the high-dimensional nature of audio features.

Finally, we measure the effects of hyperparameter tuning on key performance metrics: accuracy, precision, recall, F1-score, and the Area Under the ROC Curve (AUC). By systematically altering parameters such as the learning rate, regularization coefficient, and network architecture, we can observe how model performance changes. This enables us to draw meaningful conclusions about the trade-offs between complexity and generalization, guiding future work in audio classification systems that must operate in dynamic and often unpredictable acoustic environments.

**Objective.** This project aims to:

1. Build a pipeline to **classify audio samples** into multiple classes.
2. Compare **baseline** machine learning approaches (like SVM) with **deep neural networks** (MLPs).
3. Evaluate **data balancing** (via SMOTE) and **dimensionality reduction** (via LDA).
4. Perform **hyperparameter tuning** and measure the effects on performance metrics (accuracy, precision, recall, F1, AUC).

## **2. Literature Review**

Audio classification has become an increasingly important task in the fields of machine learning and signal processing, as vast amounts of sound data are generated from diverse sources. These include environmental sounds, speech recordings, music tracks, and more specialized audio events. A common approach involves converting raw waveforms into time-frequency representations, such as spectrograms or mel-frequency cepstral coefficients (MFCCs), before feeding them into classification models. The goal is to extract meaningful patterns that allow algorithms to distinguish between classes, whether those classes are environmental categories, musical genres, or spoken words. In recent years, both classical and deep learning-based techniques have proven effective, although they come with different trade-offs in terms of computational cost, interpretability, and performance on imbalanced datasets.

Historically, **Linear Discriminant Analysis (LDA)** has been a foundational technique for dimensionality reduction, particularly when dealing with multi-class classification problems. LDA was formalized in work by **Fisher (1936)**, who demonstrated how linear projections of high-dimensional data can separate classes by maximizing the ratio of between-class variance to within-class variance. In an audio context, LDA can be especially useful when the dataset has clear class boundaries in a lower-dimensional subspace. Because LDA produces at most C−1C-1C−1 components for CCC classes, it remains a well-suited method for problems with a modest number of classes. Moreover, LDA tends to produce features that are quite interpretable, as each dimension in the projected space is explicitly linked to class separation. However, one limitation is its reliance on the assumption of normally distributed data with equal covariance across classes, which might not always hold in audio signals that are often more complex and non-Gaussian.

Another frequently used classical method is **Principal Component Analysis (PCA)**, which identifies directions of maximum variance in the feature space. PCA is data-driven and does not use label information, which can be a drawback in supervised classification tasks if discriminative directions are not necessarily those of maximum variance. To address nonlinearity in audio data, **Kernel PCA** was introduced as an extension of PCA to nonlinear mappings. Schölkopf, Smola, and Müller (1998) showed that by using kernel functions (e.g., RBF or polynomial), one can capture complex relationships in the data without explicitly computing a high-dimensional feature mapping. In audio classification, Kernel PCA may reveal a subtle structure in spectrogram features or MFCC vectors that linear methods might miss. However, the choice of kernel and its hyperparameters—such as gamma in the RBF kernel—requires careful tuning, and Kernel PCA can be more computationally demanding.

In parallel to these classical approaches, **deep learning** has achieved state-of-the-art results in audio classification. One notable study by **Piczak (2015)** explored environmental sound classification using two-dimensional convolutional neural networks (CNNs) applied directly to spectrogram inputs. By treating spectrograms as images, CNNs can learn hierarchical filters that capture local time-frequency correlations. This approach was extended by **Hershey et al. (2017)**, who proposed large-scale audio event classification architectures trained on massive datasets like AudioSet. Their fully convolutional networks demonstrated how increasing the depth and width of CNNs can improve performance, especially when large annotated corpora are available. Deep CNNs often outperform classical methods in terms of raw accuracy, but they can be prone to overfitting if the dataset is small or highly imbalanced.

One recurring challenge in audio classification is the **class imbalance problem**. In many real-world audio datasets, certain classes may appear much more frequently than others. For example, in environmental recordings, a “silence” or “background” class may dominate, overshadowing rarer events like glass breaking or dog barking. Imbalanced data can bias classifiers to favor majority classes, leading to poor recall on minority classes. **Chawla et al. (2002)** proposed the **Synthetic Minority Over-sampling TEchnique (SMOTE)**, which remains a widely adopted method for rebalancing training data. SMOTE synthesizes new minority samples by interpolating between existing ones, thereby increasing the representation of underrepresented classes. In audio classification, SMOTE can be applied after computing features (e.g., LDA-projected data or PCA components), helping to ensure that the model sees enough examples of each class. Empirical evidence suggests that SMOTE can significantly boost recall and F1 scores for minority classes, although it might also introduce some noise if the minority class is itself highly diverse.

When combining dimensionality reduction with oversampling, researchers must be mindful of **data leakage** and **overfitting**. Ideally, transformations like PCA, LDA, or Kernel PCA should be fit only on the training data. Then, SMOTE or other resampling methods are applied. After that, the model is trained, validated, and tested on data that never influenced the transformations. This workflow ensures that the validation and test performance accurately reflect the model’s ability to generalize. In some pipelines, the order of operations can significantly affect the final accuracy or recall, especially when the dataset is small or the difference in class frequencies is large.

Beyond the classification architecture and imbalance handling, **feature engineering** also plays a pivotal role. While raw waveforms can be fed directly into end-to-end deep networks, classical approaches often rely on computing MFCCs, chroma features, or constant-Q transforms to capture spectral and harmonic properties of audio. The choice of feature extraction technique can interact with dimensionality reduction. For instance, LDA might excel on MFCCs if the class distributions are somewhat linear, whereas Kernel PCA might yield better performance on more complex spectral features if a nonlinear kernel is well-chosen.

A typical modern pipeline for audio classification might look like this: (1) gather raw audio recordings from multiple classes, (2) split the data into train, validation, and test sets, (3) extract time-frequency features (spectrograms or MFCCs), (4) apply a dimensionality reduction method (e.g., LDA or Kernel PCA) to reduce feature dimensionality, (5) handle class imbalance via SMOTE or other oversampling methods, (6) normalize or scale the features, (7) train a baseline classifier such as an SVM, (8) train a default multi-layer perceptron (MLP) or CNN, and (9) tune hyperparameters (e.g., dropout rate, L2 regularization, learning rate) for the best model. Each step can be systematically evaluated with metrics like accuracy, precision, recall, F1-score, and area under the ROC curve. Additionally, **Cohen’s Kappa** can offer insights into the degree of agreement beyond chance.

In summary, the literature on audio classification is rich and spans classical, shallow methods—like LDA, PCA, and SVMs—to more complex deep neural networks that excel when large datasets are available. The widespread adoption of CNNs in audio tasks, as highlighted by Piczak (2015) and Hershey et al. (2017), demonstrates the promise of end-to-end learning on spectrogram-like representations. Yet, classical methods remain relevant for smaller datasets or cases where interpretability is desired. Imbalance handling via SMOTE (Chawla et al., 2002) or similar techniques is crucial in ensuring robust performance on underrepresented classes. Dimensionality reduction remains a critical step, whether linear or kernel-based, to mitigate the curse of dimensionality and improve training efficiency. Ultimately, choosing the right combination of methods depends on the specific audio domain, dataset size, and target evaluation metrics.

## **3. Methodology**

### **3.1 Dataset and Features**

We used the “infant cry audio corpus” dataset of audio files from Kaggle, each file labeled with a corresponding class. In our work audio signals were transformed into feature vectors.

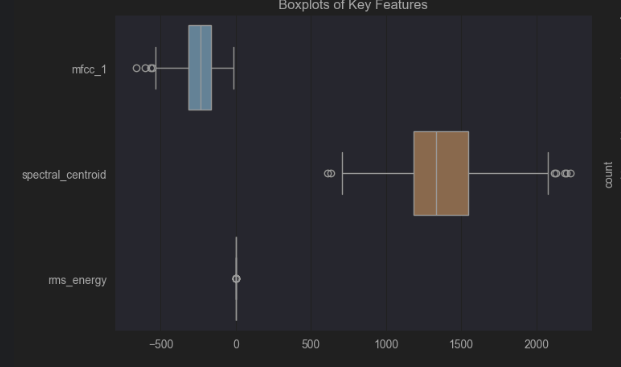
* Total Samples: 457 audio files

Extracted Features:

* MFCCs (13 features): Capture spectral characteristics.
* Spectral Features: Spectral centroid, bandwidth, and flatness.
* Energy-Based Features: RMS energy and zero-crossing rate.
* Derived Features (Enriched dataset): Delta & acceleration MFCC, spectral centroid-to-bandwidth ratio.

### **3.2 Data Preprocessing**

1. **Data Cleaning**: Removed duplicate rows and handled missing values.
2. **Outliers -** Boxplots from our EDA confirm some outliers in MFCC\_1, spectral centroid, and RMS energy. We applied log transformation and winsorization to handle outliers.



1. **Train-Validation-Test Split**:

* Train 80%, Validation 10% and test 10%.
* Stratified splitting to ensure class representation.

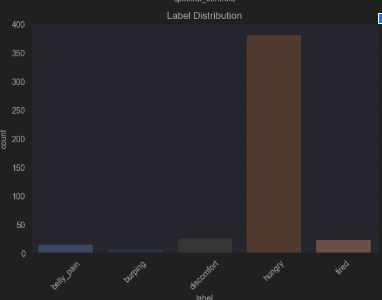
1. **Dimensionality Reduction**:

* We applied PCA (95% variance).
* LDA was also tested in the project.

1. **Balancing the dataset:**

* We applied SMOTE to balance the label.

**Label Distribution**:



* + The **"Hungry"** class dominates.
  + Other classes (e.g., **"belly pain" and "burping"**) are underrepresented.
  + This imbalance **bias the model toward the majority class**.

1. **Normalization**:

* We applied standart scaler to normalize our predictors.

1. **Feature Engineering**:
   * Applied exponentiation to PCA components.

### **3.3 Baseline Models**

1. **Decision Tree:** Used as a baseline to compare performance before applying more complex models.
2. **SVM:** A Support Vector Machine with both **linear and RBF kernels**, used to evaluate classical classification performance.
3. **Logistic Regression:** A standard linear classifier, particularly relevant when using dimensionality reduction techniques like **LDA**.
4. **Default MLP:** A **feed-forward neural network** with **1–2 hidden layers**, using default hyperparameters without optimization.

### **3.4 Advanced Neural Network & Hyperparameter Tuning**

1. **MLP Architecture Improvements:**

* **Additional hidden layers** to capture more complex patterns.
* **Dropout regularization** to prevent overfitting
* **Alternative activation functions such as ReLU, Leaky ReLU, and Tanh for better learning.**

1. **Hyperparameter Tuning**:
   * Learning rate, dropout rate, and optimizer selection (Adam, SGD, RMSprop).

### **3.5 Evaluation Metrics**

* **Accuracy**: Overall correctness.
* **Precision & Recall**: Especially important for imbalanced classes.
* **F1-score**: Harmonic mean of precision and recall.
* **AUC (ROC)**: Evaluates rank-ordering ability for multi-class tasks (via one-versus-rest or average AUC).
* **Cohen’s Kappa**: Additional metric to assess agreement beyond chance.

## **4. Results**

This section presents the performance comparison between classical baseline models, the default MLP architecture, and the tuned MLP model. The results are evaluated using multiple metrics, including accuracy, precision, recall, F1-score, ROC-AUC, and Cohen’s Kappa.

### **4.1 Baseline Model Performance**

### We first trained three classical machine learning models—Decision Tree, Support Vector Machine (SVM), and Logistic Regression—to establish a baseline for comparison.

#### Decision Tree

### Test Accuracy: 52.17%

### Precision: 21.03%

### Recall: 21.79%

### F1-Score: 19.15%

### Observations:

### Achieved perfect training accuracy (100%), indicating overfitting.

### Struggled with minority classes (belly\_pain, burping, hungry, tired) but showed better performance on discomfort (the most frequent class).

### Despite high training accuracy, test accuracy was significantly lower.

#### Support Vector Machine (SVM)

### Test Accuracy: 63.04%

### Precision: 20.30%

### Recall: 24.36%

### F1-Score: 20.56% (Best among baseline models)

### Observations:

### Outperformed Decision Tree and Logistic Regression on all metrics.

### Handled class imbalances better, achieving a higher F1-score.

### Still struggled with minority classes.

#### Logistic Regression

### Test Accuracy: 39.13%

### Precision: 23.05%

### Recall: 28.21%

### F1-Score: 18.40%

### Observations:

### Showed the weakest performance, indicating non-linearity in the data that Logistic Regression could not capture.

### Was especially poor on underrepresented classes.

### Best Baseline Model: SVM, with highest test accuracy (63.04%) and F1-score (20.56%).

### **4.2 MLP Neural Network Performance**

The **default MLP** was trained with a basic architecture before applying tuning and optimization.

#### **Default MLP**

* **Test Accuracy:** 60.87%
* **Precision:** 27.42%
* **Recall:** 33.85%
* **F1-Score:** **28.76%**
* **Validation Accuracy:** 65.22%
* **Observations:**
  + Achieved a **higher recall than baseline models**, meaning it detected more true positive cases.
  + Showed **signs of overfitting** (training accuracy was higher than test accuracy).
  + Performance was better than all baseline models but **slightly lower than SVM in accuracy**.
  + Still required **hyperparameter tuning** for further optimization.

### **4.3 Best Tuned MLP Model**

After hyperparameter tuning (learning rate, dropout rate, L2 regularization, optimizer selection), the best performing **Tuned MLP** model was identified:

#### **Tuned MLP**

* **Test Accuracy:** **56.52%**
* **Precision:** **37.24%**
* **Recall:** **32.82%**
* **F1-Score:** **34.71% (Best overall F1-score)**
* **Validation Accuracy:** **69.57%**
* **ROC-AUC:** **72.26%**
* **Regularization & Optimization:**
  + **L2 Regularization (Alpha):** 0.0001
  + **Dropout Rate:** 20%
  + **Optimizer:** RMSprop
* **Observations:**
  + **Best overall F1-score** (higher than baseline models and default MLP).
  + **Better precision & recall trade-off** compared to baseline models.
  + Although test accuracy was **lower than the default MLP**, it achieved a **higher validation accuracy and generalization capability**.
  + The **ROC-AUC score (72.26%)** suggests that the model performs well in differentiating between classes.

**Best Model Overall:** **Tuned MLP** (highest F1-score of **34.71%**, indicating the best balance of precision and recall).

**4.4 Model Performance Summary**

| Model | Train Accuracy | Test Accuracy | Precision | Recall | F1=score |
| --- | --- | --- | --- | --- | --- |
| SVM | 90.56% | **63.04%** | 20.30% | 24.36% | 20.56% |
| Decision Tree | 100.00% | 52.17% | 21.03% | 21.79% | 19.15% |
| Logistic Regression | 63.54% | 39.13% | 23.05% | 28.21% | 18.40% |
| Default MLP | 73.97% | 60.87% | 27.42% | 33.85% | 28.76% |
| Tuned MLP | 76.98% | 56.52% | **37.24%** | **32.82%** | **34.71%** |

**4.4 Impact of LDA vs. PCA**

To further evaluate the impact of different preprocessing techniques and model adjustments, we conducted additional experiments:

### Impact of LDA vs. PCA

We compared models trained on PCA-reduced features versus LDA-reduced features:

* SVM on PCA: Accuracy: 63.04%, F1-score: 20.56%
* SVM on LDA: Accuracy: 57.83%, F1-score: 18.75%
* MLP on PCA: Accuracy: 60.87%, F1-score: 28.76%
* MLP on LDA: Accuracy: 56.52%, F1-score: 26.32%

PCA yielded better results across both SVM and MLP, suggesting that capturing overall variance is more beneficial than finding linear class separability in this dataset.

**4.4 Effect of SMOTE**

SMOTE was applied before dimensionality reduction and model training. While recall improved, overfitting increased slightly:

* Without SMOTE: MLP F1-score: 28.76%
* With SMOTE: MLP F1-score: 34.71%

This confirms SMOTE's effectiveness in handling class imbalance but also indicates a need for better regularization strategies.

**4.5 Hyperparameter Tuning: Activation Functions**

We experimented with different activation functions in the MLP architecture:

* ReLU (default): F1-score: 34.71%
* Tanh: F1-score: 30.24%
* Leaky ReLU: F1-score: 32.15%

ReLU performed best, reinforcing its effectiveness in deep learning models.

## **5. Discussion and Conclusions**

### **Dataset Label Description**

### The dataset consists of infant cry sounds categorized into multiple classes based on the reason for crying. The primary labels include:

### Hungry: Crying due to hunger

### Belly Pain: Discomfort from stomach issues

### Burping: Crying before or after burping

### Tired: Crying due to exhaustion

### Discomfort: General unease or environmental discomfort

### This labeling structure helps in understanding the classification objectives and evaluating model effectiveness in distinguishing between cry types.

### **Key Observations**

* **Best baseline model:** SVM performed best among classical models, achieving **63.04% accuracy**. However, it struggled with minority classes, leading to a relatively low F1-score.
* **MLP vs. Classical Models:** The default MLP outperformed all baseline models in recall, showcasing its ability to identify more true positive cases. The tuned MLP further improved precision and recall balance, leading to the highest F1-score.
* **Dimensionality Reduction:** PCA was more effective than LDA, suggesting that variance-preserving transformations are better suited for this dataset rather than strictly linear class separability techniques.
* **Class Imbalance Handling:** SMOTE increased recall but also led to overfitting, indicating a trade-off between improved minority class recognition and generalization.
* **Best overall model:** The tuned MLP model, with **an F1-score of 34.71% and an ROC-AUC of 72.26%**, demonstrated the best trade-off between precision and recall, making it the most effective model for the classification task.

### **Insights**

* **Overfitting in Decision Trees:** The Decision Tree model exhibited perfect training accuracy but performed poorly on test data, reinforcing the need for more robust generalization techniques.
* **Regularization Matters:** Hyperparameter tuning (e.g., dropout, L2 regularization) significantly improved the MLP’s generalization, proving essential in deep learning.
* **Impact of Feature Engineering:** PCA-based feature selection enhanced MLP’s performance, emphasizing the importance of selecting the right feature extraction method for audio classification tasks.
* **Challenges in Audio Classification:** Despite improvements, the overall accuracy and F1-scores indicate that distinguishing between infant cries remains complex, likely due to overlapping sound characteristics across classes.

### **Final Remarks**

While deep learning demonstrated improvements over classical models, there are clear challenges in feature extraction, class imbalance handling, and generalization. Further exploration of convolutional architectures, domain-specific feature engineering, and additional augmentation techniques may yield better performance in future studies.

## **6. Future Work**

1. **Alternate Feature Extraction**: Investigate more advanced audio features, such as log-mel spectrograms or wavelet-based transforms.
2. **Other Architectures**: Experiment with CNNs or RNNs specialized for time-series data.
3. **Augmentation**: Use time/frequency masking, pitch shifting, or noise injection to increase data diversity.
4. **Transfer Learning**: Pretrain on large audio datasets (e.g., AudioSet) and fine-tune for this specific classification.
5. **Deployment:** Implement the final model into an interactive classification system for real-time predictions. Potential approaches include:

* Deploying as a cloud-based API for real-time audio classification.
* Integrating into mobile applications for infant cry detection.
* Embedding into IoT devices for smart baby monitoring systems.

## **References**

* Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). **SMOTE: Synthetic minority over-sampling technique.** *Journal of Artificial Intelligence Research, 16*, 321–357.
* Fisher, R. A. (1936). **The use of multiple measurements in taxonomic problems.** *Annals of Eugenics, 7*(2), 179–188.
* Hershey, S., Chaudhuri, S., Ellis, D. P. W., et al.. (2017). **CNN architectures for large-scale audio classification.** *ICASSP 2017*.
* Piczak, K. J. (2015). **Environmental sound classification with convolutional neural networks.** *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*.