Classification of Audio Embeddings

Subject: Introduction to Machine Learning

Supervised by: Bùi Duy Đăng, Nguyễn Thanh Tình

Location: Thành phố Hồ Chí Minh, Việt Nam

Trường Đại học Khoa Học Tự Nhiên, tp Hồ Chí Minh

Prepared by: Group 30

Date: June 01, 2025

1 Group Information

This project was completed by the group **Group 30**. The team members are listed below:

Name	**Student ID**
Bùi Kim Phúc	21120112
Lê Hoàng Sơn	21120127

Bảng 1: Group Members

2 About the Project

The objective of this project is to develop a machine learning model to classify audio clips based on the presence of turkey sounds. The input data consists of audio embeddings, each represented as a matrix of shape $[10, 128]$, where 10 is the number of frames and 128 is the dimensionality of each frame's feature vector. These embeddings are extracted from audio clips and provided in a JSON file (`train.json`). The task is a binary classification problem, where the model predicts whether an audio clip contains turkey sounds (`is_turkey = 1`) or not (`is_turkey = 0`). The classification is performed using a MLP model using Pytorch framework, leveraging the flattened embeddings (size 1280) as input features. Finally, the model's performance is evaluated on a test set, and the results are submitted in a CSV file (`submission.csv`) for Kaggle competition.

3 Data Preprocessing

The dataset was sourced from a JSON file (`train.json`) containing audio embeddings and binary labels. The preprocessing steps included:

- Loading the JSON data into a pandas DataFrame.
- Extracting `audio_embedding` (shape [10, 128]) and `is_turkey` (binary labels: 0 or 1).
- Flattening each audio embedding to a 1D array of size 1280 (10 $\times$ 128).
- Padding or truncating embeddings to ensure a consistent size.
- The data contains a total of 1195 samples and is split into 836 training samples (70%), 179 validation samples (15%), and 180 test samples (15%) with shuffling to ensure randomness.

4 Model Description

This project was conducted in three phases, employing different model architectures:

4.1 Phase 1: Traditional Machine Learning Models

Initially, we implemented two classical machine learning models using Scikit-learn:

- **Logistic Regression**:
  - **Algorithm**: Logistic regression with the `liblinear` solver, suitable for small datasets.
  - **Input Features**: Flattened audio embeddings of size 1280 (10 frames $\times$ 128 dimensions).
  - **Output**: Binary classification (0 or 1) for the `is_turkey` label.
  - **Hyperparameters**: C=1.0, max_iter=1000, random_state=42.
- **Random Forest**:
  - **Algorithm**: Random forest classifier, an ensemble method using multiple decision trees.
  - **Input Features**: Flattened audio embeddings of size 1280 (10 frames $\times$ 128 dimensions).
  - **Output**: Binary classification (0 or 1) for the `is_turkey` label.
  - **Hyperparameters**: n_estimators=100, max_depth=None, random_state=42.

4.2 Phase 2: Neural Network Approach

In the second phase, we implemented a more sophisticated neural network model using PyTorch:

- **Multilayer Perceptron (MLP)**:
  - **Architecture**: A flexible neural network with configurable hidden layers.
  - **Input Layer**: Flattens the input embeddings from [10, 128] to a vector of size 1280.
  - **Hidden Layers**: Multiple fully connected layers with ReLU activation functions.
  - **Regularization**: Dropout layers with 0.3 dropout rate to prevent overfitting.
  - **Output Layer**: A fully connected layer producing logits for binary classification.

MLP model compared to models in Phase 1 offers several advantages:

- More complex pattern recognition through non-linear transformations using Rectified Linear Units (ReLU) activation functions.
- Better handling of the high-dimensional audio embedding features
- Enhanced regularization strategies to prevent overfitting

5 Platform

The project was developed and executed on the following platform:

- **Google Colab**: The primary development environment, providing a cloud-based Jupyter notebook interface with GPU support for efficient model training and evaluation.
- **Google Drive**: Used for storing and accessing the dataset (`train.json`) and saving the submission file (`submission.csv`).
- **Libraries**: Pytorch framework for model implementation, pandas for data processing, NumPy for numerical operations, and other standard Python libraries.
- **Latex**: The report is formatted using LaTeX for professional presentation.

6 Training

The training process involved the following steps for both the logistic regression and random forest models:

- **Data Loading**: The JSON dataset was loaded using pandas and processed to extract features and labels.
- **Data Splitting**: The dataset was split into:
  - Training set: 70% of the data, including 836 samples.
  - Validation set: 15% of the data, including 179 samples.
  - Test set: 15% of the data, including 180 samples.

  Splitting was performed using Scikit-learn's `train_test_split` with a random state of 42 for reproducibility.
- **Training Approach**:
  - Loss Function: Cross-Entropy Loss
  - Optimizer: Adam with learning rate of 0.003
  - Weight Decay: 1e-5 for L2 regularization
  - Early Stopping: Implementation of patience (16 epochs) to prevent overfitting
  - Pocket Algorithm: Saves the best model based on validation accuracy
  - Maximum Epochs: 64

7 Evaluation Metrics

The model's performance was evaluated using the following metrics:

- **Accuracy**: The proportion of correct predictions on the validation and test sets.

8 Accuracy Report

The performance of our models is reported below:

- **Phase 1: Traditional Machine Learning Models**:
  - **Logistic Regression**:
    - **Validation Accuracy**: 0.9497
    - **Test Accuracy**: 0.9167
    - **Kaggle Score**: 0.9334
  - **Random Forest**:
    - **Validation Accuracy**: 0.8939
    - **Test Accuracy**: 0.9000
- **Phase 2: Neural Network Approach**:
  - **Multilayer Perceptron (MLP)**:
    - **Validation Accuracy**: 0.9385
    - **Test Accuracy**: 0.9611
    - **Kaggle Score**: 0.9411

The MLP model achieved its best validation accuracy of 93.85% after just 3 epochs of training, with early stopping triggered after 16 epochs without improvement. The final test accuracy was 96.11%, indicating a significant improvement over the traditional models.

9 Submission

Submission files include:

- `Group30_Turkey30.ipynb`: The Jupyter notebook containing the code for data preprocessing, model training, and evaluation.
- `This report`: The LaTeX report detailing the project, methodology, and results.
- `submission.csv`: Our neural network approach achieved a higher score of 94.11% on Kaggle.

10 Conclusion

In this project, we successfully implemented a machine learning model using MLP in Pytorch framework. Although the Kaggle score result did not improve significantly compared to the logistic regression model, we gained valuable insights into the classification of audio embeddings. The project demonstrated the effectiveness of traditional machine learning models and neural networks in handling audio data.

11 References

- Pytorch tutorial: `https://docs.pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html`
- Pandas Documentation: `https://pandas.pydata.org/`
- Google Colab: `https://colab.research.google.com/`
- Kaggle competition page: `https://www.kaggle.com/competitions/introduction-to-machine-learning-project-cq-`

12 AI Usage

The project utilized AI tools including GPT-4, Grok, and Claude Sonnet 3.7 for various tasks such as:

- Code syntax support and debugging.
- Report helping and formatting in LaTeX assistance.

We acknowledge the use of AI tools in enhancing our productivity and code quality, while ensuring that the core logic and implementation were developed by the team.

Chat conversations with AI can be found here:

- `https://grok.com/share/bGVnYWN5_b1b4cc44-6408-4ae0-ad5e-ed776476da01`