Analysis and implementation of Ego audio classification

Chi-Jui Wu
Samuel Ginn College of Engineering
Auburn University
Auburn, AL
czw0131@auburn.edu

***Abstract*— This project investigates the application of deep learning models - the Audio Spectral Transformer (AST) and the Convolutional Neural Network (CNN) - to audio classification in the context of the Charades-Ego dataset, which first provides daily activities Dual perspective and third person video editing. The goal is to advance the technology of augmented reality, assistive technology, and interactive systems by enhancing the performance of models in identifying and classifying audio signals that correspond to specific actions. This project aims to solve and alleviate problems such as class imbalance and model generalization through specialized preprocessing and training strategies. Experimental results highlight the superiority of CNN models over AST in terms of verification accuracy and ability to effectively classify complex audio patterns. The study also discusses challenges encountered, such as limitations in audio clip length and memory management during model training, and suggests future work directions to improve model robustness and efficiency.**

Keywords*— *Audio Spectrogram Transformer, CNN, Audio extraction, MFCC, Spectrogram, sigmoid function

I. INTRODUCTION

In this project, the application of advanced deep learning models, specifically the Audio Spectrogram Transformer (AST) and the Convolutional Neural Network (CNN), are explored to address audio classification challenges in the context of the Charades-Ego dataset. Our goal is to improve the model's performance in identifying and classifying various audio signals corresponding to various actions depicted in first-person video clips. Sourced from the Allen Institute for AI, this data set provides a unique dual perspective on daily activities and is ideal for developing systems that require a user-centered understanding. This project aims to improve the predictive capabilities of these models to facilitate advances in areas such as augmented reality, assistive technologies, and interactive systems. This project provides possible solutions to key issues such as category imbalance and model generalization, as well as possible future work directions. It is expected that subsequent researchers can improve the overall accuracy and efficiency.

II. ENVIRONMENTS

A. Version of Python and Anaconda

The environment versions used in this project are Python 3.11.5 and Anaconda3-2023.09-0-Linux-x86_64.

B. Version of Function Packages

This project uses the following function packages and versions, including pytorch: 2.2.1+cu121, sklearn: 1.4.1.post1, tqdm: 4.66.2, librosa: 0.10.1, and moviepy: 1.0.3. The PyTorch library supports CUDA 12.1 in version 2.2.1, which facilitates efficient computing on compatible GPUs. Version 1.4 .1.post1 of the scikit-learn library, this project uses the train_test_split function to split the data set. To visualize progress during long operations, tqdm was used in version 4.66.2, which enhanced the user interface with real-time feedback. Audio processing tasks are handled by librosa version 0.10.1, which is known for its comprehensive audio analysis capabilities as well as the noise reduction capabilities of the noisereduce suite. Video processing was performed with moviepy version 1.0.3, which supports video editing tasks critical to multimedia projects. In general, these function package versions are the latest version currently in progress for this project. If there is a version update, it is recommended to use the latest version without affecting the model.

C. Address of the Datasets and Documents

The cloud database for this project is provided by Dr. Aakur and is located at Auburn University. The main code for the model and information acquisition process is located in the cloud database "/home/chijuiwu/snap/snapd-desktop-integration/83/Desktop/CapstoneProject". This directory serves as the central hub for main script execution and intermediate processing steps. Video data critical to the visual analysis component of the project is stored in the cloud database at "/media/DiskDrive1/CharadesEgo". Additionally, the metadata and tag details required to train and test the model can be found in the CSV file: `/home/chijuiwu/snap/snapd-desktop-integration/83 /Desktop/CapstoneProject/CharadesEgoInfo/CharadesEgo_v1_test_only1st.csv`.

D. Location of codes

The project code storage location is "/home/chijuiwu/snap/snapd-desktop-integration/83/Desktop/CapstoneProject". This address contains three main parts, audio extraction, AST model, and CNN model. The file names of Audio extraction are "Audio_Extract_singal_prosess_RegularClasses.ipynb" and "Audio_Extract_singal_prosess_ListClasses.ipynb" respectively. The AST model is stored in the file name "AST_Training_with_noaction_LsitClasses.ipynb". CNN model is "CNN_Training_with_noaction_Final.ipynb"

Audio extraction includes two types of storing the labels of the training data set after extraction. Because an audio clip may have different "actions" at a single point in time. First, store only a single category of data points, and separate different "actions" at a single time point to create a data set. Another method is to list the categories according to "action" or not in time units. For example, this data has a total of 158 categories, and one time point has the first and third actions. The corresponding label of this time point is [1,0,1,0...0] (1 represents that time contains this action, 0 otherwise).

III. Data Description

Data for this project are derived from the Charades-Ego dataset, which can be accessed at Allen Institute for AI, as part of a broader effort to advance research in computer vision and machine learning, focusing on the challenges of egocentric perception—understanding First person point of view[2]. The Charades-Ego dataset extends the original Charades dataset by including first-person video clips and third-person clips depicting daily activities. This dual perspective is particularly valuable for developing models that understand and explain human behavior from the user's perspective, making it ideal for applications in augmented reality, assistive technology, and interactive systems.

The CSV file set located at `/home/chijuiwu/snap/snapd-desktop-integration/83 /Desktop/CapstoneProject/CharadesEgoInfo/CharadesEgo_v1_test_only1st.csv` is the annotation data set of Charades-Ego. The archive contains movie files annotated with various actions, where each action is marked with a start and end time within the clip. The data includes video files in MP4 format corresponding to the video file name "/media/DiskDrive1/CharadesEgo". and corresponding comments for each video file detailing the action ID and its associated time. For preprocessing, this project will extract audio data and store features and corresponding classes in units of 1 second. This project uses several steps to refine and prepare data for further analysis and model training, including audio extraction, noise reduction, feature extraction, and feature classes correspondence.

A. Audio Extraction

The audio extraction of this project uses "moviepy" as the main way to extract audio files. Since the subsequent model will learn to determine the corresponding actions per second in the audio file, the extracted audio file will be divided into one audio clip of 1 second, so that features and classes can be added to the learning data set later.

B. Noice Reduction

Each audio clip extracted from the film is processed with noise reduction to minimize background noise, thereby improving the clarity of the audio features relevant to the action being performed. This is achieved by using the noise reduction library "noisereduce", which applies noise reduction algorithms to audio data.

C. Feature Extraction

This project extracted two main audio features including MFCC (Mel Frequency Cepstrum Coefficient) and spectrogram, using the librosa library. The MFCC feature represents the short-term power spectrum of sound. MFCC is crucial for capturing the temporal dynamics of audio, which represents different actions in the film. A spectrogram is a visual representation of the frequency spectrum of a sound over time, providing a rich, visual-like representation of sound.

D. Generate Input Data

Each audio clip is processed second by second and the MFCC and spectrogram are calculated for each segment. The resulting material is then formatted into a structured array for input into a machine learning model, ensuring that each segment is linked to its corresponding action label. The above methods are designed to promote efficient training and accurate performance of the deep learning models used in the project.

E. Normalizing

Normalization is a key pre-processing step in this project, aiming to enhance the model's ability to effectively learn and generalize audio features. This process involves scaling the MFCC and spectrogram datasets to a standard range, thereby mitigating issues related to the scale of variation between features that can adversely affect the performance of machine learning algorithms. Specifically, this is done by subtracting the fixed minimum (-4.2677393) and dividing by the range (twice the maximum, 4.5689974), effectively mapping the data to a more consistent scale.

IV. Models and Model Parameters

In this project, the Audio Spectrogram Transformer (AST) and the Convolutional Neural Network (CNN) are respectively established as the main models for audio classification and compared between the two. Designed to process spectrograms into image-like data, AST uses a transformer architecture that is good at capturing complex patterns in audio signals, allowing it to effectively represent and classify sound events[1]. The model is configured using "ASTConfig" and instantiated using "ASTModel", providing a powerful framework for learning from spectrogram inputs[2]. CNNs are used to address different aspects of audio processing, exploiting their spatial hierarchy to extract features directly from spectrogram data. The combination of these models aims to exploit local and global features of the message to improve classification accuracy. The model optimizer is set to Adam for adaptive learning rate adjustment to ensure that the model converges efficiently during the training process.

A. Audio Spectrogram Transformer

The Audio Spectrogram Transformer (AST) in this project is configured as follows: The activation function is set to "Sigmoid" to provide non-linearity in the encoding and pooling stages, since audio clips may have different operations simultaneously per second. The hidden layer size is 384 and consists of 6 hidden layers with 6 attention heads each, reducing complexity while maintaining a large amount of learning capabilities. The size of the middle layer is 3072 to facilitate the model to learn detailed feature representation. It is worth noting that the dropout probability of the hidden layer is increased to 0.4, introducing more regularization to combat overfitting of specific audio data, while the attention probability will not be affected by dropout, making it possible to retain the learned attention Pattern integrity. The initialization range is set to 0.02, ensuring that the weights start close to zero but have enough diversity for effective learning. The input spectrogram has a patch size of 16, a frequency span of 1, and a time span of 1, providing fine-grained analysis of the spectrogram input. The model can handle spectrograms with time dimensions up to 32 and frequency dimensions up to 1025 Mel frequency bins, and is optimized for capturing detailed audio patterns.

B. Convolutional Neural Network

The architecture of the CNN in this project consists of three convolutional layers, with the depth of each layer gradually increasing from 16 to 32 and then to 64 filters, with a kernel size of 3x3 and padding set to 1 to preserve the input spectrogram. spatial dimensions. These layers are activated

using the ReLU function to introduce nonlinearity, thereby enhancing the model's ability to learn complex features in the material. The pooling layer is placed after each convolutional layer with a 2x2 window and a stride of 2 to reduce the dimensionality of the feature map while retaining important features. This downsampling mechanism helps reduce computational load and control overfitting. The output of the convolutional basis is flattened and fed into a fully connected layer, which maps the extracted features to the number of target categories.

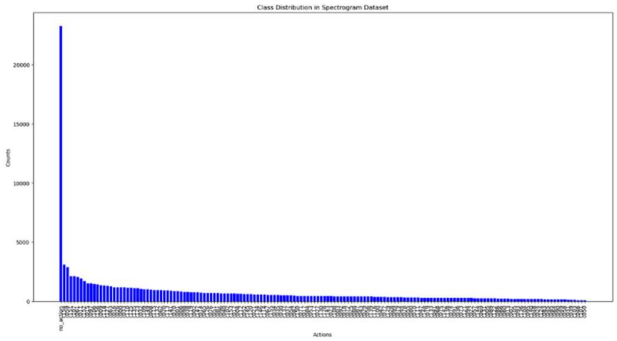

Figure 1: Class Distribution Figure, the training data set shows a highly unbalanced distribution, with the number of samples in some categories being significantly higher than in other categories, therefore the project has tried to use weights for balance.

In addition, According to Fig. 1, due to the unbalanced proportion of categories in the data, some categories have significantly more samples than others. For this reason, class weights are input into the model to handle imbalanced datasets. This adjustment helps to assign higher importance to less frequent sessions during the training phase, thus counteracting the bias against more frequent sessions. Using the formula total_samples/ (num_classes x count), the overall sample is divided by the total number of classes and the number of samples in that class, and each class is given a weight inversely proportional to its frequency in the dataset[4]. The calculated weights are then used in the model's loss function, effectively increasing the penalty for misclassifying underrepresented classes. This ensures that the model does not ignore a few categories and improves its generalization ability across all categories.

C. *Performance of each model*

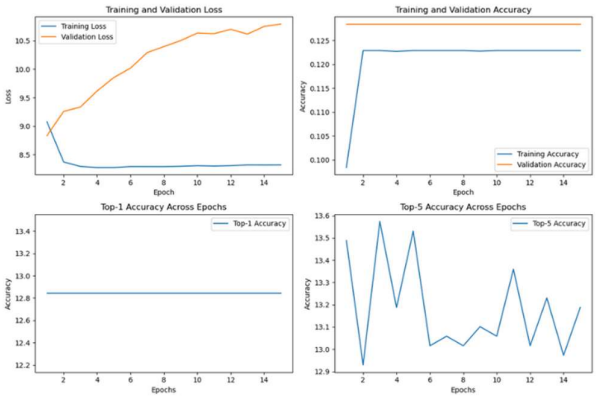

Figure 2: AST model performance

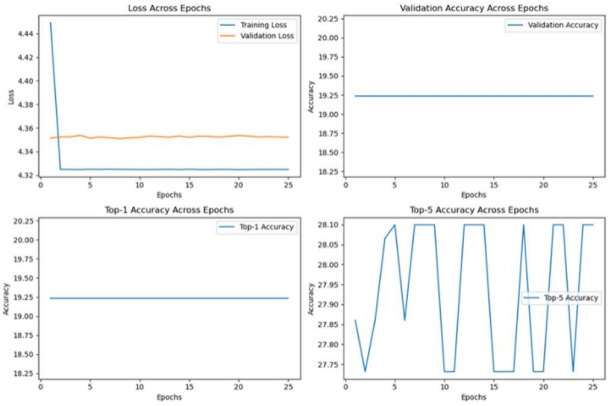

Figure 3: CNN model performance

Table 1: Performance comparison for AST model and CNN model

Model	Best Validation Accuracy	Best Top-1 Accuracy	Best Top-5 Accuracy
AST	12.84%	12.84%	13.57%
CNN	19.24%	19.24%	28.1%

According to Figure 1, Figure 2 and Table 1, the model needs to further refine the model architecture, training parameters or refine the data to improve performance. For CNN models, the training loss quickly drops to a fixed value, while the validation loss remains relatively stable, indicating that the model may converge prematurely or be overfitted. The AST model, on the other hand, exhibits considerably lower loss levels from the beginning, indicating better initial performance and generalization stability, but suffers from overfitting issues.

In terms of Top-1 and Top-5 accuracy, the CNN model is better than the AST model in both Top-1 and Top-5 accuracy indicators. Specifically, CNN's best verification accuracy for Top-1 and Top-5 is 19.24% and 28.10% respectively, which is significantly higher than AST's performance of 12.84% and 13.57%. This suggests that at this stage of the project, the CNN model is better at processing and classifying the data set for this specific task.

V. Discussion

A. *Challenges*

On this project, I encountered several significant obstacles that affected the progress and results of my work. First, the current audio processing program cannot handle audio clips shorter than 1 second, while the original goal of the project required audio clips as short as 0.1 seconds to train the model. This limitation requires finding alternatives or adapting the processing pipeline to accommodate the required audio clip length. In addition, the model training phase also faces the problem of overfitting, where the model performs well on training data but poorly on unseen data, indicating the need for better generalization techniques. Finally, memory limitations were a serious challenge, as I encountered "CUDA Error: Out of Memory," indicating that the GPU resources were insufficient for the scale I was trying to train the model if the model was too complex. This requires optimizing the memory usage of the model or leveraging a more powerful hardware solution to do it efficiently.

B. *Future Work Recommendations*

Based on the above issues, several strategies are proposed to overcome the challenges encountered and enhance this work.

First, to address the limitation of 1 second audio clips, it is crucial to explore alternative audio processing tools or libraries that support finer clips (e.g., 0.1 second clips). For example, libraries such as "pydub" or "librosa" can provide the flexibility needed to efficiently process shorter audio clips. Alternatively, developing a custom solution to preprocess the audio to the required length before feeding it into the training pipeline may be the way to go.

Second, regarding the challenge of model overfitting, more robust regularization techniques can be employed, such as dropout, batch normalization, or introducing a complexity penalty during the training process. In addition, expanding the dataset through finer clipping or through data augmentation techniques can help improve the model's generalization ability.

Finally, to address memory limitations, it is crucial to optimize the architecture of existing models to improve memory efficiency. This may involve reducing the complexity of the model by pruning unnecessary layers or neurons that have little impact on the output. Additionally, implementing techniques such as mixed-precision training can reduce memory usage during model training without significantly impacting performance. The above methods all require time to test and verify, which will be helpful for future work and can also assist in the development of more complex projects.

ACKNOWLEDGMENT

. I would like to extend my deepest gratitude to Dr. Aakur for his invaluable guidance and unwavering support throughout the duration of this project. His expertise and insightful critiques have been instrumental in shaping both the direction and success of this work. Dr. Aakur's dedication to fostering a learning environment that encourages curiosity and rigor has profoundly impacted my educational journey. I am immensely thankful for his patience and the considerable time he invested in mentoring me, which has greatly enriched my understanding and appreciation of this field.

REFERENCES

[1] Y. Gong, Y.-A. Chung, and J. Glass, "AST: Audio Spectrogram Transformer," arXiv (Cornell University), Apr. 2021, doi: https://doi.org/10.48550/arxiv.2104.01778.

[2] " Audio Spectrogram Transformer," huggingface.co. https://huggingface.co/docs/transformers/model_doc/audio-spectrogram-transformer

[3] "Perceptual Reasoning and Interaction Research - Charades-Ego," prior.allenai.org. https://prior.allenai.org/projects/charades-ego

[4] hengtao tantai, "Use weighted loss function to solve imbalanced data classification problems," Medium, Feb. 27, 2023. https://medium.com/@zergtant/use-weighted-loss-function-to-solve-imbalanced-data-classification-problems-749237f38b75