CLOUDY
STUDIO

/HOME /USR /VAR /BIN

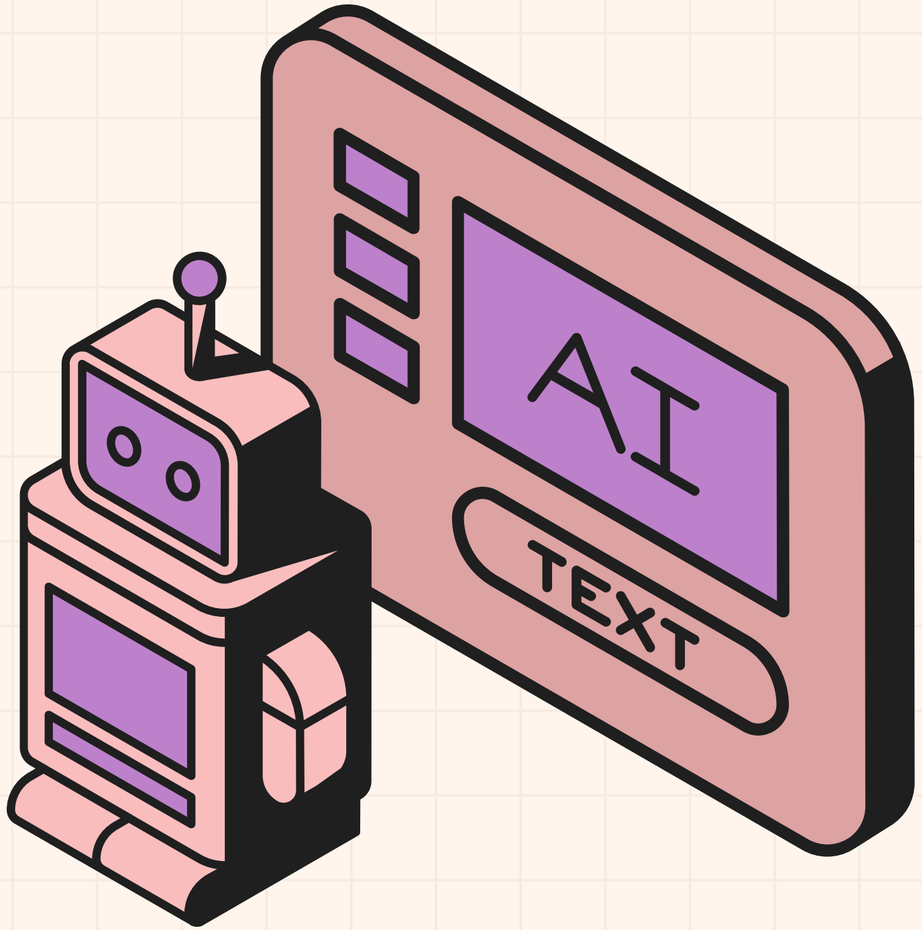
AUDIO EMOTION ANALYSIS (FINAL PROJECT)

Cloudy’s OS Members

1) 6688077 Bhumipat Pengpaiboon
2) 6688108 Napas Siripaskrittakul
3) 6688142 Krerkkiat Wattanaporn

FEATURE EXTRACTION

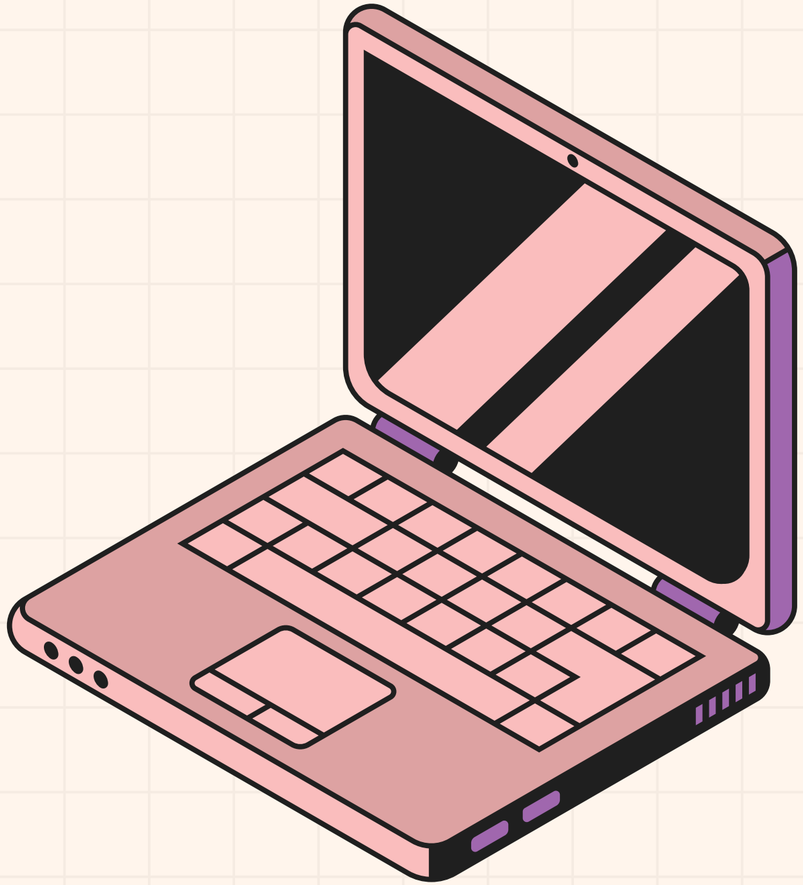
- Each audio file sampled at 44.1 kHz with 13 MFCCs were extracted using the librosa library.
- All feature sequences are standardized to 500 time steps by padding the shorter ones.
- The resulting 2D array is flattened. (500 time step x 13 MFCCs)
- Each Audio sample becomes 1D vector of 6500 features for Machine learning input.



MODEL

Multi-Layer Perceptron (MLP) classifier

- Implemented by using scikit-learn (MLPClassifier).
- Followed by feature scaling using by StandardScaler.



THE VERSION OF CODE

The first version

```
X_list = []
# ...
for index, row in tqdm(df_meta.iterrows()):
    features = extract_features(row['path'])
    X_list.append(features.flatten())
X = np.array(X_list)
```

The final version

```
filepaths = df_meta['path'].tolist()
results = Parallel(n_jobs=-1, backend='loky')(
    delayed(extract_features)(fp) for fp in tqdm(filepaths)
)
X_list = [res.flatten() for res in results]
X = np.vstack(X_list)
```

COMPARISION TWO VERSIONS OF CODE

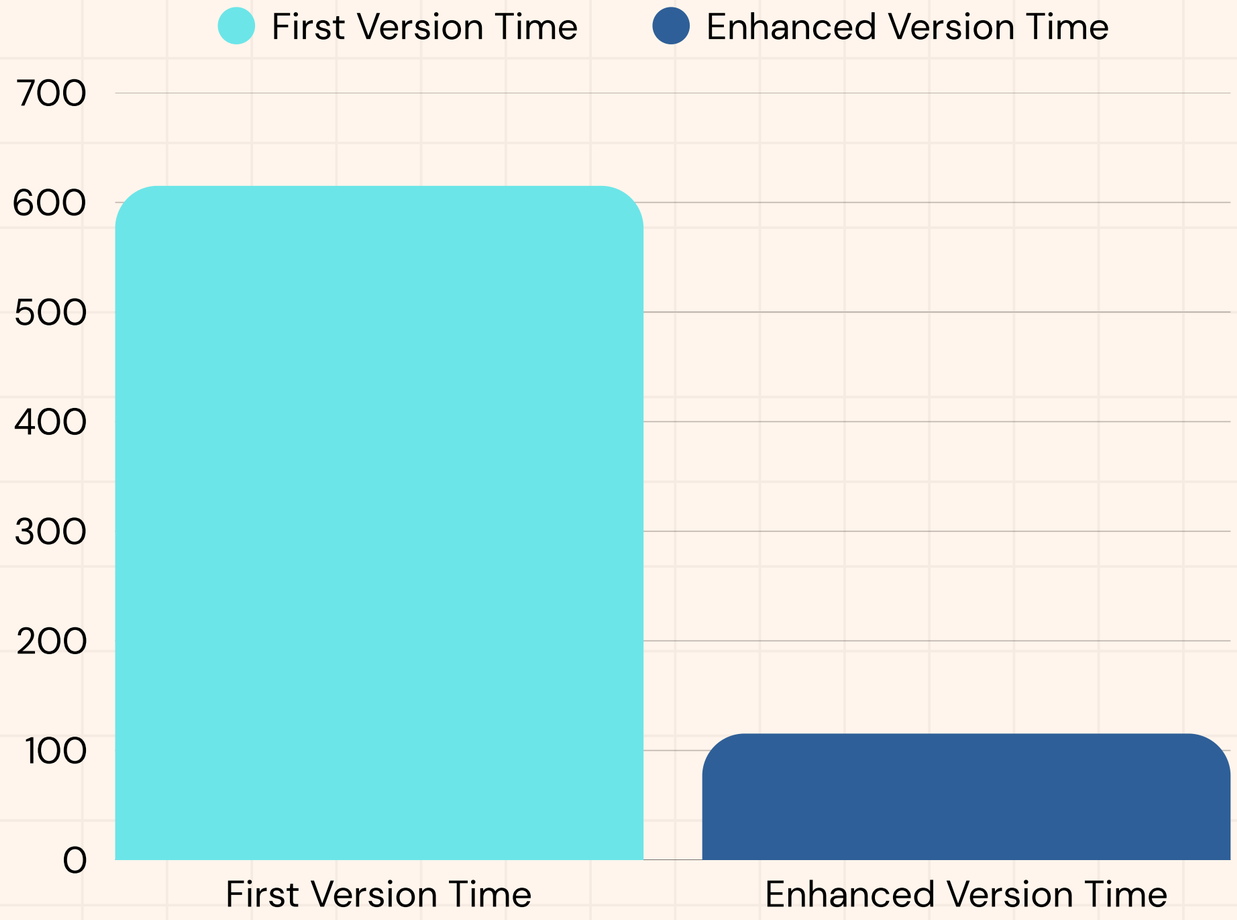
	The first version	The final version
Processing Method	Processes audio files sequentially	Processes audio files in parallel
CPU Utilization	Utilizes only a single CPU core	Leverages multiple CPU cores
Performance	Slower because its sequential, single-core nature.	Faster because it processes files concurrently across multiple cores.
Implementation Technique	Uses a standard Python for loop	Employs the joblib.Parallel utility along with delayed

Why we chosen this method?

- CUDA (GPU) Incompatibility: Project’s libraries (librosa, scikit-learn) lack native CUDA support, preventing GPU acceleration.
- Python's Global Interpreter Lock (GIL): GIL limits CPU parallelism, hindering speed gains for sequential audio loading and MFCC extraction.
- Python multiprocessing: The joblib library enables multiprocessing, allowing parallel computation for scikit-learn tasks.

PERFORMANCE ANALYSIS

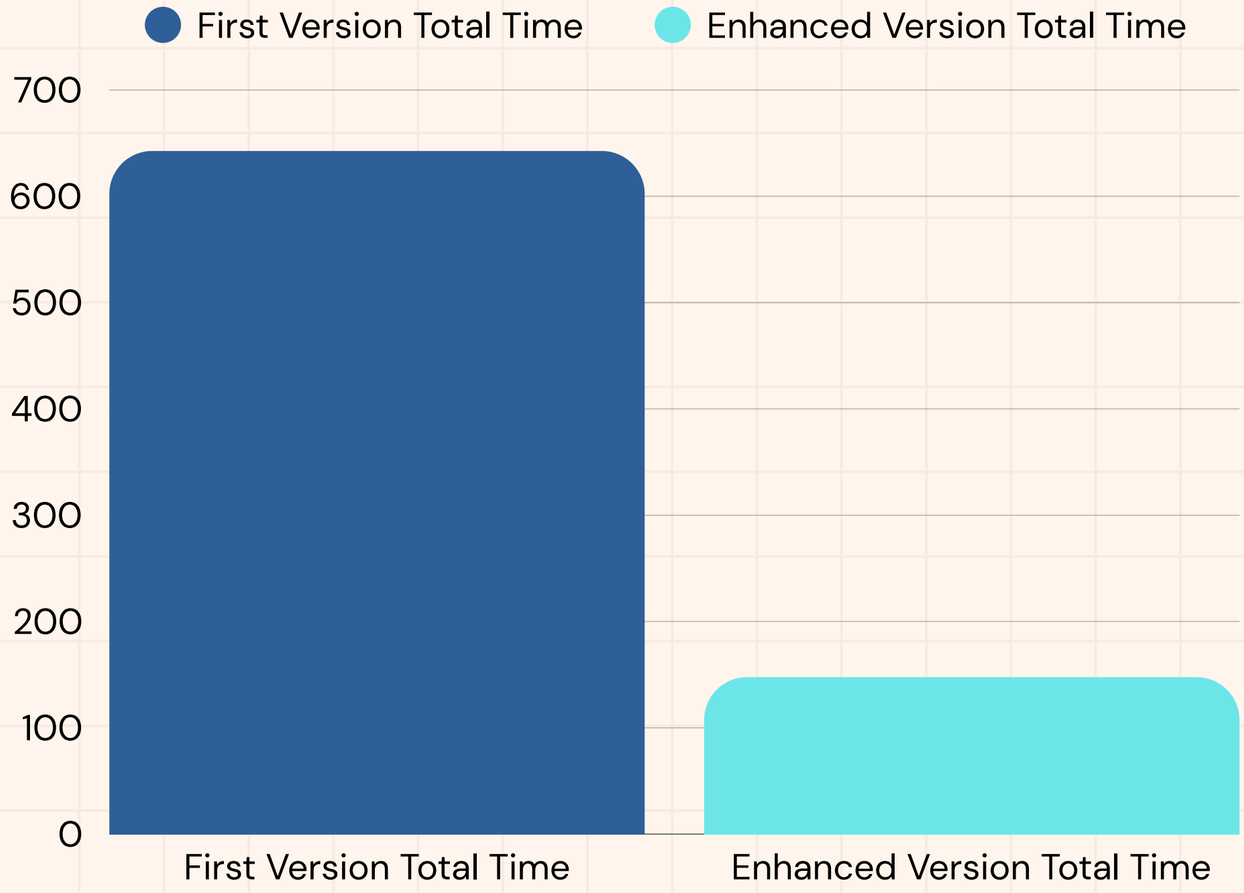
FEATURE EXTRACTION COMPARISON



- THE ENHANCED VERSION ACHIEVED APPROXIMATE **5.3X SPEEDUP** BY EFFECTIVELY UTILIZING MULTIPLE **CPU CORES** TO PROCESS FILES .

PERFORMANCE ANALYSIS

OVERALL EXECUTION TIME COMPARISON



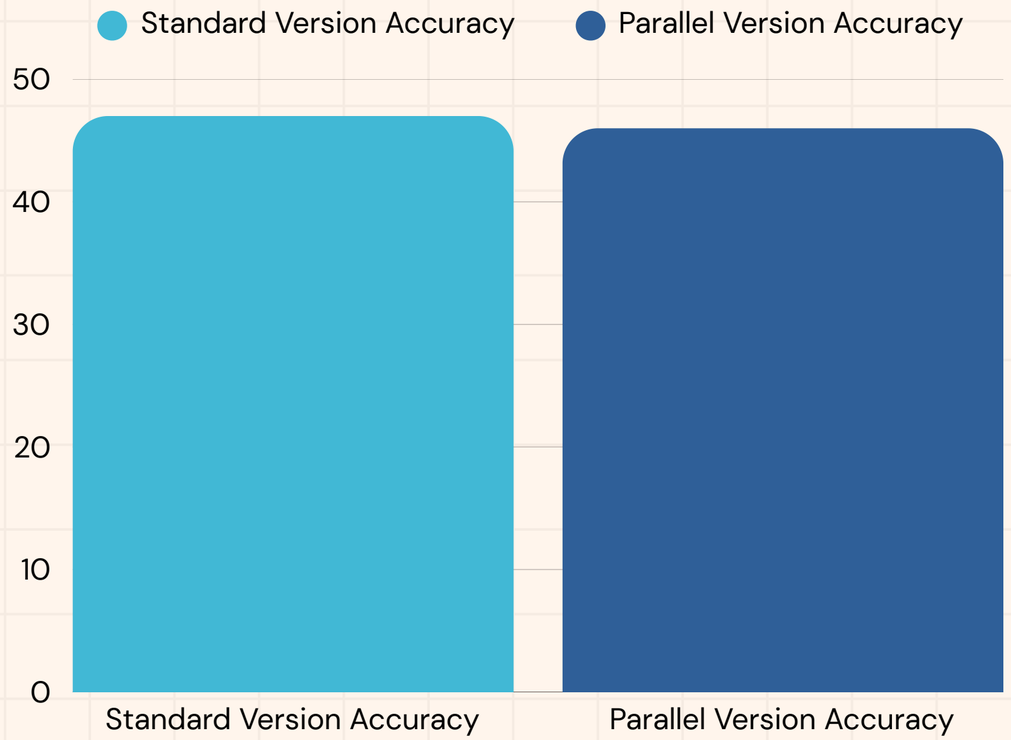
- THE ENHANCED VERSION ACHIEVED APPROXIMATE **4.35X SPEEDUP** BECAUSE THE COMPUTATIONAL BOTTLENECK.

RESOURCE USAGE

CPU & MEMORY

- **WALL-CLOCK TIME** WAS SIGNIFICANTLY REDUCED IN THE ENHANCED VERSION.
- **PEAK MEMORY USAGE** WAS SLIGHTLY HIGHER (~3632 MB VS ~3146 MB)
- **INCREASING MEMORY** WAS DUE TO **MANAGING CONCURRENT PROCESS** AND **OPTIMIZING DATA HANDLING**

MODEL ACCURACY ANALYSIS



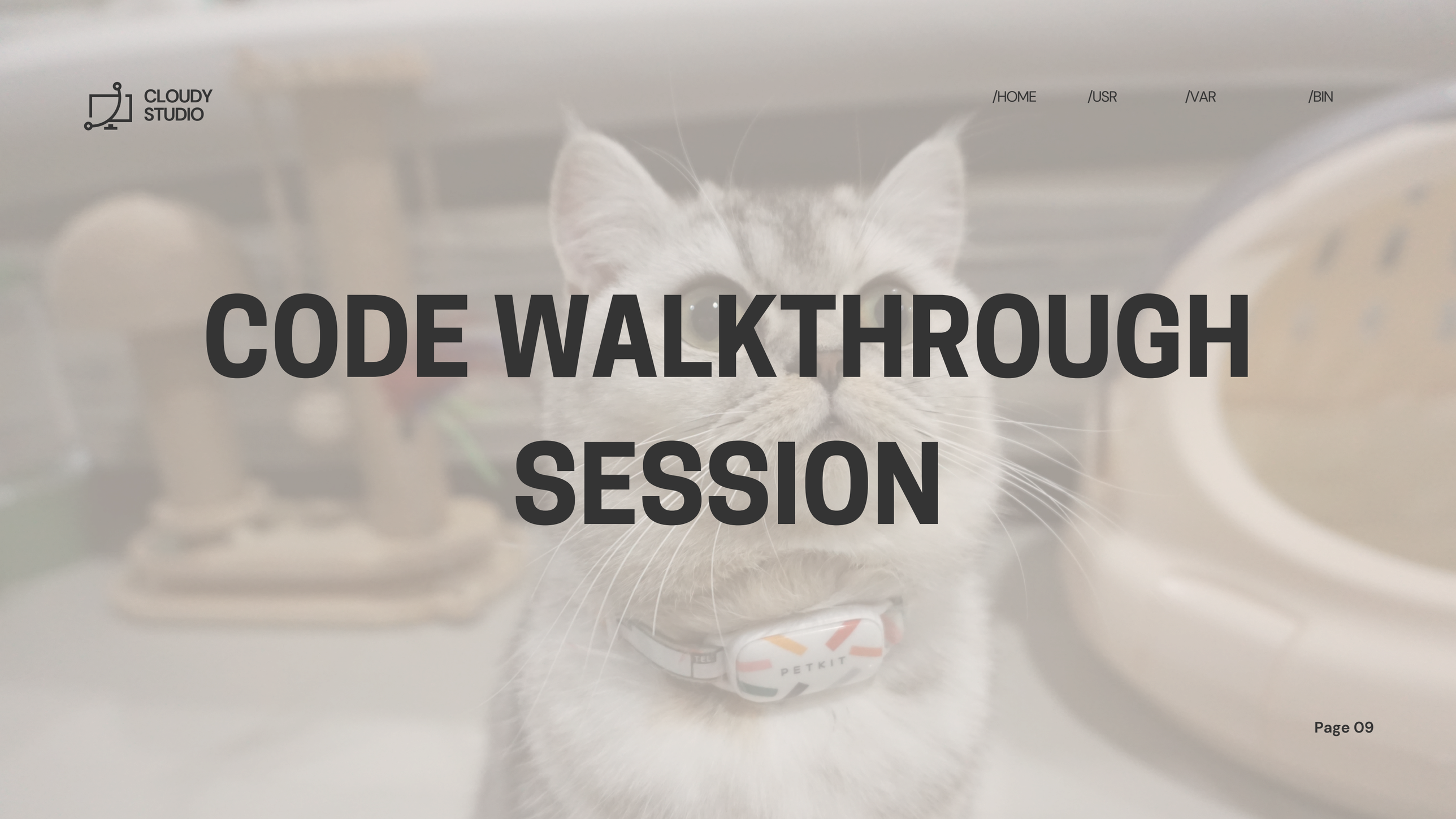
- **Standard Version Accuracy**: 47.14%
- **Parallel Version Accuracy**: 46.56%

MODEL ACCURACY ANALYSIS

- **Neutral emotion** achieved the highest F1-score (0.55)
- **Sad emotion** is getting the lowest F1-score (0.32)
- **Other classes (Angry, Frustrated, Happy)** showed moderate F1-scores (0.43-0.47)

The overall weighted average F1-score was 0.46-0.47 and it is moderate success in this multi-class classification task with these features and architecture.

CONCLUSION



CODE WALKTHROUGH SESSION

THANK YOU