UMass | Dartmouth

Implementation of HPC for Music Genres Classification

for music recommendation systems

DSC 520-01: High Perform Scientific Compute

NIKHILESH NARKHEDE (02196141)

Problem Statement

Music genres: Build a DL model to classify music into the top 10 genres.

Audio Data ⟶ Time series data.

If we go with traditional time series architecture like RNN, the LSTM model turns out to be bulky and faces problems like vanishing gradient.

But we can convert Audio data in image format using a spectrogram with a library like Librosa.

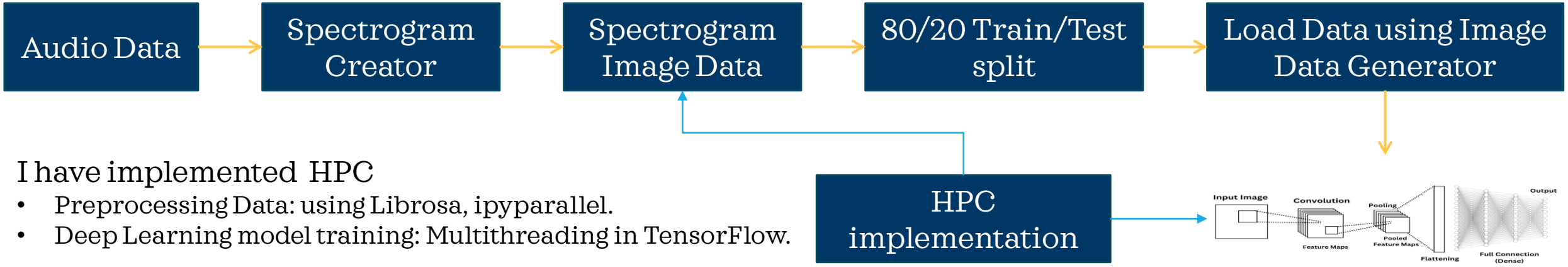
Its representation of the spectrum of frequency in sound recording as they vary over time.

Dark Area: Low-intensity frequency.

Orange and yellow Areas: High intensities frequency.

https://academo.org/demos/spectrum-analyzer/

Architecture

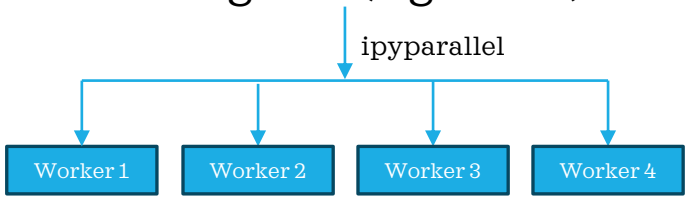


I have implemented HPC

- Preprocessing Data: using Librosa, ipyparallel.
- Deep Learning model training: Multithreading in TensorFlow.

Preprocessing Data :

- sub-genre(e.g. Rock) 100 songs



25 songs for each worker.

Deep Learning model training :

- OpenMP
- Inter-operation
- Intra-operation

UMass | Dartmouth

ipyparallel for Distributed Computing

What is ipyparallel?

- A Python library that enables parallel task execution using multiple workers, ideal for distributed computing.

Key Components:

- **Controller:** Distributes tasks and maintains the state.
- **Engines:** Execute tasks independently.

Example Use Case:

- Parallel processing of audio files into spectrogram images to reduce computation time.

Benefits:

- Improves efficiency, scales to multi-core systems, and integrates seamlessly with Python and Jupyter.

UMass | Dartmouth

Deep Learning Model Training Optimization

OpenMP

- OpenMP optimizes CPU-based computations by distributing work across multiple cores.
- Usage in this Model:
- Convolution layers (Conv2D) and matrix operations in fully connected layers (Dense) are parallelized.
- OpenMP ensures that operations like matrix multiplications for the filters in the convolution layers and the weight updates in the dense layers are executed across multiple CPU threads.

Inter-Operation Parallelism

- Manages parallel execution across different operations in the computational graph.
- If two Conv2D layers are computationally independent (e.g., during forward propagation), TensorFlow can process them concurrently.
- Pooling layers and dropout calculations can also be executed in parallel with certain independent layers.

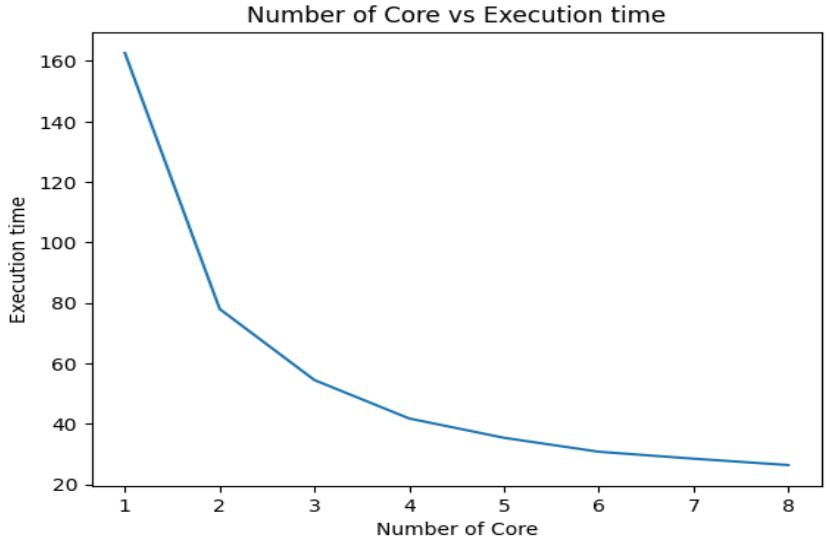
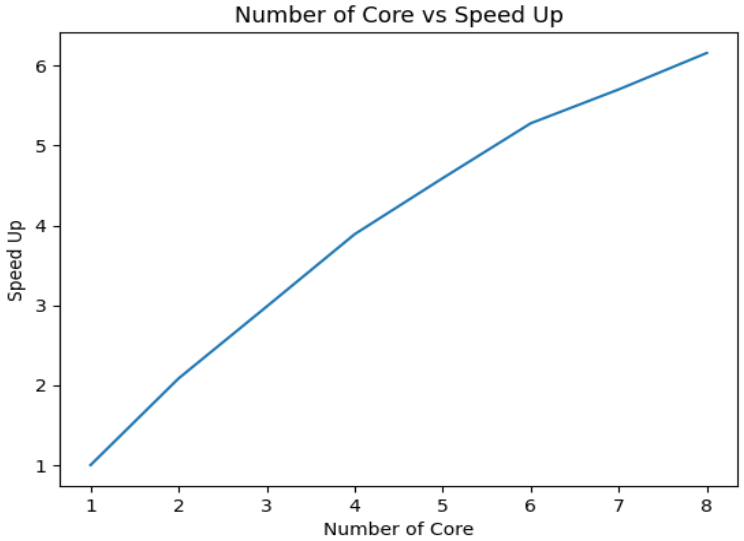
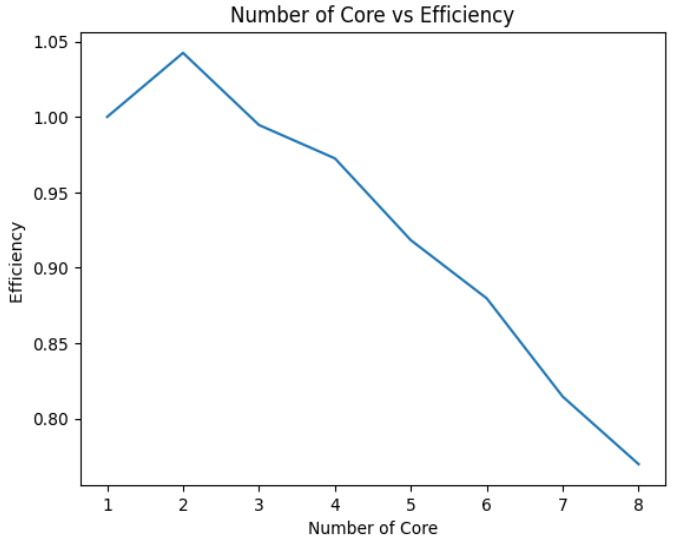
Intra-Operation Parallelism

- Optimizes parallelism within a single operation.
- Each Conv2D operation (a matrix multiplication between input data and convolution filters) is broken into smaller tasks.
- Pooling and activation computations (like ReLU) for large feature maps are divided among threads.

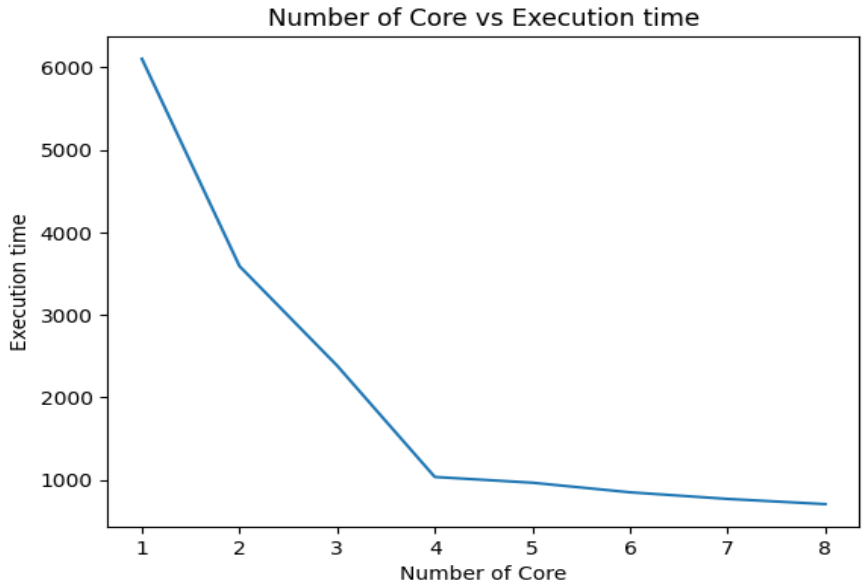
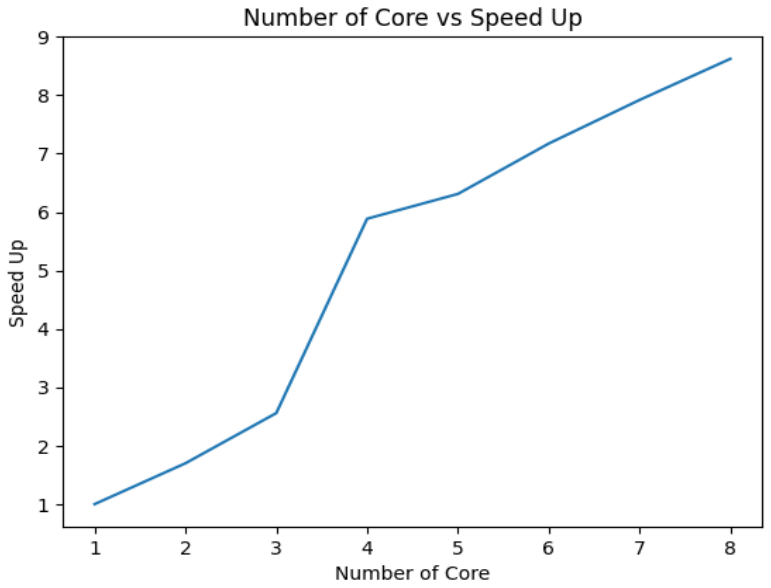
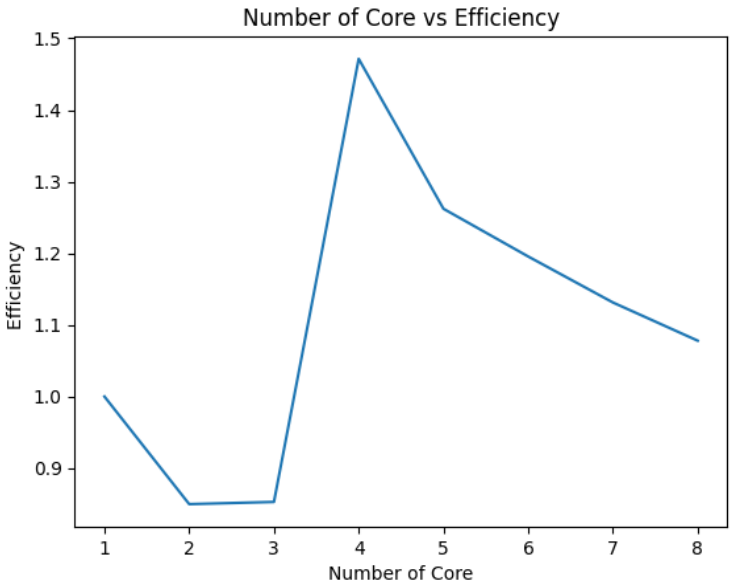
Key Benefits

- Improved efficiency and reduced execution time.
- Scalability across multi-core systems.
- Better utilization of computational resources.

Result (Preprocessing)

Number of core vs Execution time	Number of core vs Speed-up	Number of core vs Efficiency
		

Result (DL Model Training)

Number of core vs Execution time	Number of core vs Speed-up	Number of core vs Efficiency
		

Conclusion

Preprocessing Data:

- 2 workers are most preferable in this context if efficiency is given more weight
- 4 workers are most preferable in this context if execution time is given more weight

Deep Learning Model Training:

- 4 Multithreading (Worker) is most preferable in this context in terms of both execution time and efficiency.