CS 429/529 Introduction to Machine Learning

Project 3: Neural Networks

Welcome back, data wizards! It's time to elevate your journey through our machine learning course to new heights as we dive into the second phase of our audio challenge. In this phase, we're raising the stakes and unleashing the power of deep learning as we explore the enchanting world of audio classification. Building upon the foundations laid in our first phase, where you mastered the art of feature extraction and classical machine learning models, we're now venturing into the realm of multilayer perceptrons (MLPs), convolutional neural networks (CNNs), and the magic of transfer learning.

Picture this: You're on a quest to revolutionize how we understand and interact with music, armed with the most advanced tools and techniques in the field of artificial intelligence. Your mission: to harness the intricate patterns and nuances hidden within audio data and unlock the secrets of musical genres.

In this phase of the competition, you'll embark on an adventure of experimentation and discovery: from crafting powerful MLP architectures to sculpting intricate CNN models tailored for spectrogram analysis, you'll wield an arsenal of deep learning techniques to conquer the challenge before you.

But the journey doesn't end there – we're also delving into the realm of transfer learning, where you'll harness the knowledge distilled from pre-trained models to amplify your predictive prowess. By leveraging the hidden wisdom accumulated from vast repositories of data, you'll breathe new life into your models and propel them to unprecedented levels of accuracy and performance.

As you embark on this epic quest, remember that the path ahead is filled with twists and turns, challenges and triumphs. But fear not, for you are not alone – your fellow classmates stand beside you, ready to lend their support and share their insights as we embark on this collective journey of discovery.

So, sharpen your wits and prepare to unleash the full force of your creativity and ingenuity. The stage is set, the competition awaits – may the algorithms be ever in your favor![1]

[1] Intro created with ChatGPT

Project Specifications

For this project you are asked to perform three tasks: 1) Build and train MLPs with your structured data, 2) extract spectrograms from the audio data and train CNNs, 3) use transfer learning on either modality. **Your data is the SAME data as for project 2**

Feature Extraction and Data Preparation:

- **Structured data:** Use the best set of features from project 1 as your structured dataset.
- **Images:** Generate spectrograms from the audio data using audio libraries (e.g., Librosa, scipy).
    - Ensure that the spectrograms are properly formatted and ready for use with CNNs. That means, pay attention to the image sizes, whether you will use 1 or 3 (or more) channels, pay attention to the colormap scales across images.

Multilayer Perceptrons (MLPs):

- Implement MLPs using deep learning frameworks like TensorFlow or PyTorch, use of Matlab or R is okay as well..
- Design the architecture of the MLP including the number of layers, neurons per layer, activation functions, and regularization (e.g., L1, L2, dropout).
- Train the MLPs on the extracted structured data.
- Tune hyperparameters (e.g., learning rate, batch size), select appropriate loss functions and optimizer..

Convolutional Neural Networks (CNNs):

- Design and implement CNN architectures suitable for spectrogram classification.
- (optional - if you do this, they need to be trained from scratch) Experiment with different CNN architectures (e.g., LeNet, VGG, ResNet) and adjust parameters like kernel size, number of filters, and pooling layers.
- Train the CNNs on the spectrogram data. Training in this module needs to be done from scratch.
- Utilize techniques such as data augmentation to improve model generalization.

Transfer Learning:

- Choose one of the data modalities (either feature-based or spectrogram-based) for transfer learning.
- Utilize pre-trained models (e.g., VGG, ResNet, Inception) from deep learning frameworks like TensorFlow or PyTorch.

- Fine-tune the pre-trained models on the audio classification task using the selected data modality.
- Experiment with freezing certain layers or adjusting learning rates during fine-tuning.
- Compare the performance of transfer learning with models trained from scratch.

Model Evaluation and Comparison:

- Evaluate the performance of MLPs, CNNs, and transfer learning models using appropriate metrics (e.g., accuracy, precision, recall, F1-score).
- Conduct statistical tests or cross-validation to ensure robustness of results.
- Compare the performance of different models and analyze their strengths and weaknesses.
- Visualize the training/validation curves and confusion matrices for deeper insights.

DIscussion:

- Document, in your report, the entire process including data preprocessing, model architectures, training procedures, and evaluation results.
- Submit your results to kaggle

Image Preparation

1.- Optionally preprocess the audio signals if necessary (e.g., resampling, normalization, noise reduction) to enhance spectrogram quality.

2.- Compute the Short-Time Fourier Transform (STFT) of the audio signal. The STFT represents how the frequency content of the signal changes over time. Choose parameters such as window size, hop size, and FFT size:

- Window Size: Determines the duration of each analysis window. A longer window captures more frequency detail but reduces temporal resolution.
- Hop Size: Determines the amount of overlap between adjacent windows. A smaller hop size provides better temporal resolution but increases computational cost.
- FFT Size: Determines the frequency resolution of the spectrogram. A larger FFT size captures more frequency detail but may result in longer computation time.

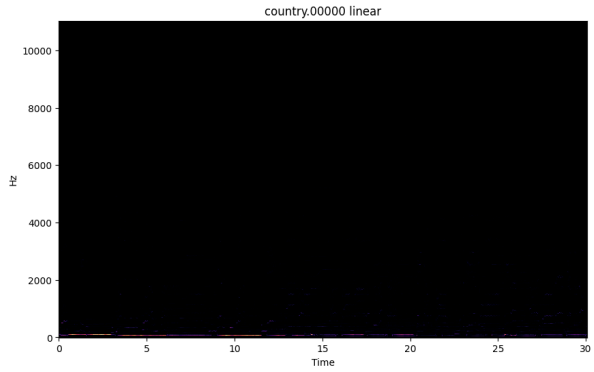
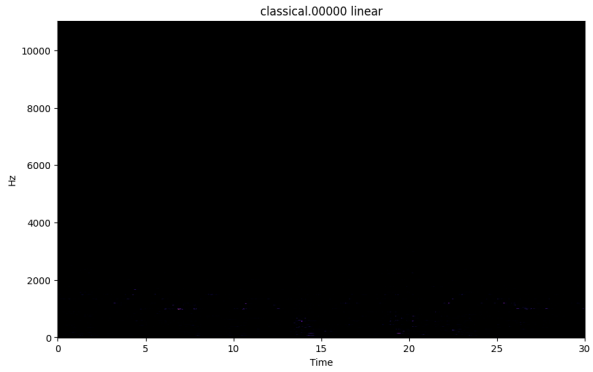
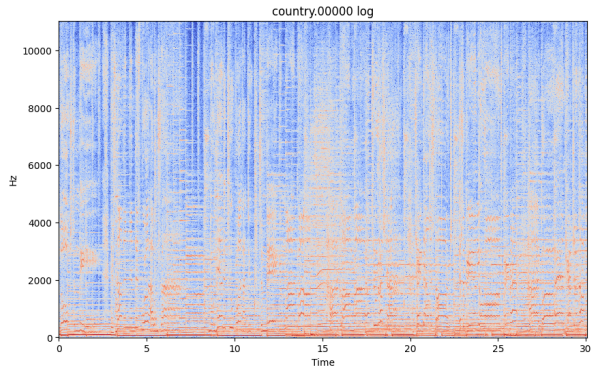
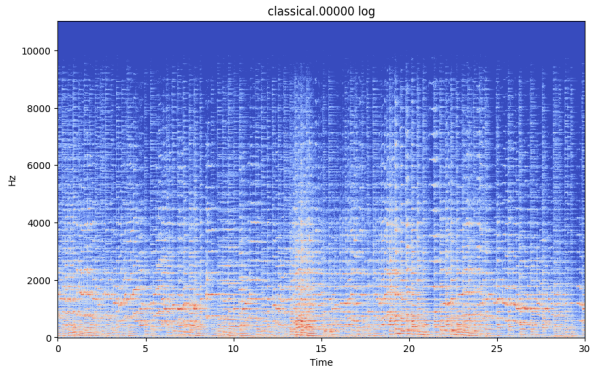
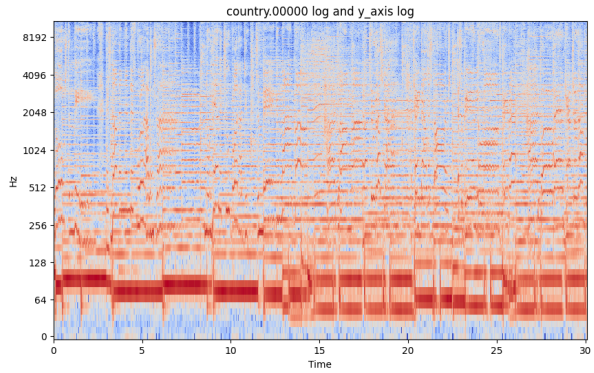
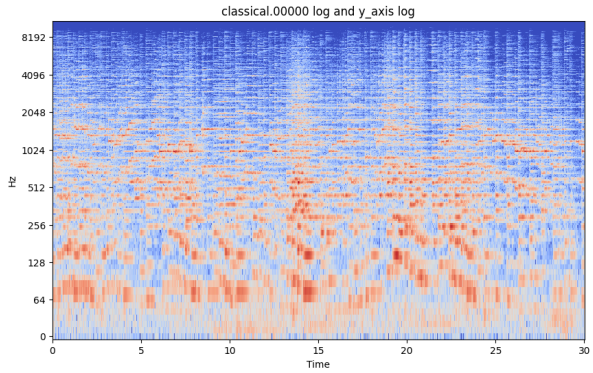
- Experiment with different parameter values to balance time-frequency resolution and computational efficiency based on the specific requirements of your application.

The following code (see notebook in drive folder) is based on this kaggle notebook: https://www.kaggle.com/code/alifrahman/spectrograms-extraction-using-librosa

```
import os
import librosa
import librosa.display
import IPython.display as ipd
import numpy as np
import matplotlib.pyplot as plt

def plot_spectrogram(title, y, sr, hop_length, y_axis = "linear"):
   plt.figure(figsize=(10,6))
   librosa.display.specshow(y, sr = sr, hop_length = hop_length, x_axis = "time", y_axis =
y_axis)
   #plt.colorbar(format="%+2.f")
   plt.title(title)
   plt.show()

def extract_and_plot(audio_data, frameSize, hopSize, title):
   audio, sample_rate = librosa.load(audio_data)
   stft_audio = librosa.stft(audio, n_fft = frameSize, hop_length = hopSize)
   y_audio = np.abs(stft_audio) ** 2
   plot_spectrogram(title+' linear', y_audio, sample_rate, hopSize)
   y_log_audio = librosa.power_to_db(y_audio)
   plot_spectrogram(title+' log', y_log_audio, sample_rate, hopSize)
   plot_spectrogram(title+' log and y_axis log', y_log_audio, sample_rate, hopSize, y_axis =
"log")


frameSize = 2048
hopSize = 512

audio1 = lpath+"/country.00000.au"
extract_and_plot(audio1, frameSize, hopSize,'country.00000')

audio2 = lpath+"/classical.00000.au"
extract_and_plot(audio2, frameSize, hopSize,'classical.00000')
```

Example code to extract and visualize spectrograms

Using this code you can plot spectrograms for different songs and evaluate what parameters give you the best type of images.

country.00000	classical.00000
	
	
	

Example of spectrograms for 2 different songs

3.- Decide whether to preserve multiple color channels or use a single-channel representation: If you preserve 3 Color Channels, you may represent the spectrograms as RGB images where color channels can capture different information, such as different frequency bands or time-frequency representations (e.g., magnitude, phase). Using a single-channel representation (e.g., grayscale) simplifies the spectrogram and reduces memory/storage requirements. Consider the trade-offs between information richness, simplicity, and ability of the model to converge, based on the complexity of your classifier and computational resources.

4.- Pixel Intensity and Normalization: scale the spectrogram values to a suitable range for pixel intensity representation:

- For RGB images: Scale each color channel to the range [0, 255] or [0, 1] based on the requirements of your deep learning framework.
- For single-channel representation: Normalize the spectrogram values to a standardized range (e.g., [0, 1] or [-1, 1]) to facilitate model convergence and stability during training.
- Decide whether this normalization needs to be done image per image or on a per-dataset basis. You may apply normalization techniques such as min-max scaling or z-score normalization to ensure consistent pixel intensity distribution across spectrograms. Consider the dynamic range and distribution of spectrogram values to prevent saturation or loss of information.

Rubric

Deliverables (10 points)
Submit 2 separate files in canvas: a pdf report and an compressed file of your code and README

- (2 points) Report must be PDF. Report will not be graded otherwise.
- (2 points) Code packaged in archive (tar or zip); no github links, do not include the dataset
    - Include README; at a minimum, describe how to run code and where to specify path to the dataset.
    - (2 points) File manifest :: describe each file (one sentence, must be more descriptive than “this file contains all code”. Even if one file does hold everything, describe what’s in there.)
    - (2 points) Team names; contributions from each member

- (2 points) Final Kaggle score, accuracy, and date run

Data Preparation (10 points)

- Data Conversion (5 points)
  - Proper handling of audio data and conversion to appropriate input format for MLPs and CNNs.
- Data Splitting (2 points)
  - Proper splitting of data into training, validation.
  - Ensure no data leakage between sets.
- Data Augmentation (3 points)
  - Implementation of data augmentation techniques (if applicable) to increase the diversity of training data.

Multilayer Perceptrons (MLPs) Implementation (15 points)

- Architecture Design (10 points)
  - Well-designed MLP architecture, discuss design decisions with respect to number of layers, neurons, and activation functions.
  - Regularization techniques (e.g., dropout) applied to prevent overfitting.
- Training and Hyperparameter Tuning (5 points)
  - Training of MLPs with appropriate hyperparameters (e.g., learning rate, batch size).
  - Utilization of techniques like grid search or random search for hyperparameter tuning.

Convolutional Neural Networks (CNNs) Implementation (15 points)

- Architecture Design (10 points)
  - Well-designed CNN architecture tailored for spectrogram classification.
  - Effective choice of convolutional layers, pooling layers, and activation functions.
- Training and Hyperparameter Tuning (5 points)
  - Training of CNNs with appropriate hyperparameters.
  - Utilization of techniques like learning rate scheduling and early stopping for efficient training.

Transfer Learning (15 points)

- Selection of Modality and Pre-trained Models (5 points)
  - Clear rationale for selecting the data modality for transfer learning (e.g., feature-based or spectrogram-based).
  - Proper selection of pre-trained models suitable for the classification task.

- Fine-tuning and Performance Improvement (5 points)
    - Effective fine-tuning of pre-trained models on the chosen modality.
    - Demonstrated improvement in model performance compared to training from scratch.
- Analysis and Interpretation (5 points)
    - Insightful analysis of the effectiveness of transfer learning compared to other approaches.
    - Discussion on the impact of fine-tuning strategies and model architecture on performance.

Model Evaluation and Comparison (15 points)

- Performance Metrics (5 points)
    - Calculation and reporting of appropriate evaluation metrics (e.g., accuracy, precision, recall, F1-score) for all models.
    - Clear interpretation of performance results and comparison between different approaches.
- Visualization and Analysis (10 points)
    - Visualizations of training/validation curves, confusion matrices, and any other relevant plots.
    - Detailed analysis of the strengths and weaknesses of each model and approach.

Documentation and Code Quality (10 points)

- Documentation (5 points)
    - Clear documentation of code, including inline comments and docstrings.
    - Readable and well-organized code structure.
- Code Quality (5 points)
    - Efficient and optimized code implementation.
    - Proper error handling and defensive programming techniques.

Report quality (10 points)

- Clear and well-structured sections of project methodology, results, and conclusions.
- Effective communication of technical concepts.
- Use of figures/captions/tables and visual aids.
- Demonstrated understanding of the project's technical aspects.