Music Generation With Machine Learning

Team 135: Romal Peccia, Abhishek Paul, Donald Lleshi

Supervisor: Jason Anderson **Administrator**: John Taglione

Background and Motivation

- Personal motivation: Writing music is hard!
- Desire for an application to extend user-recorded music
- ML is widely used for **random** generation of music
- ML for **input-based** generation of music is mostly in research phase, one commercial product was announced in Dec 2019

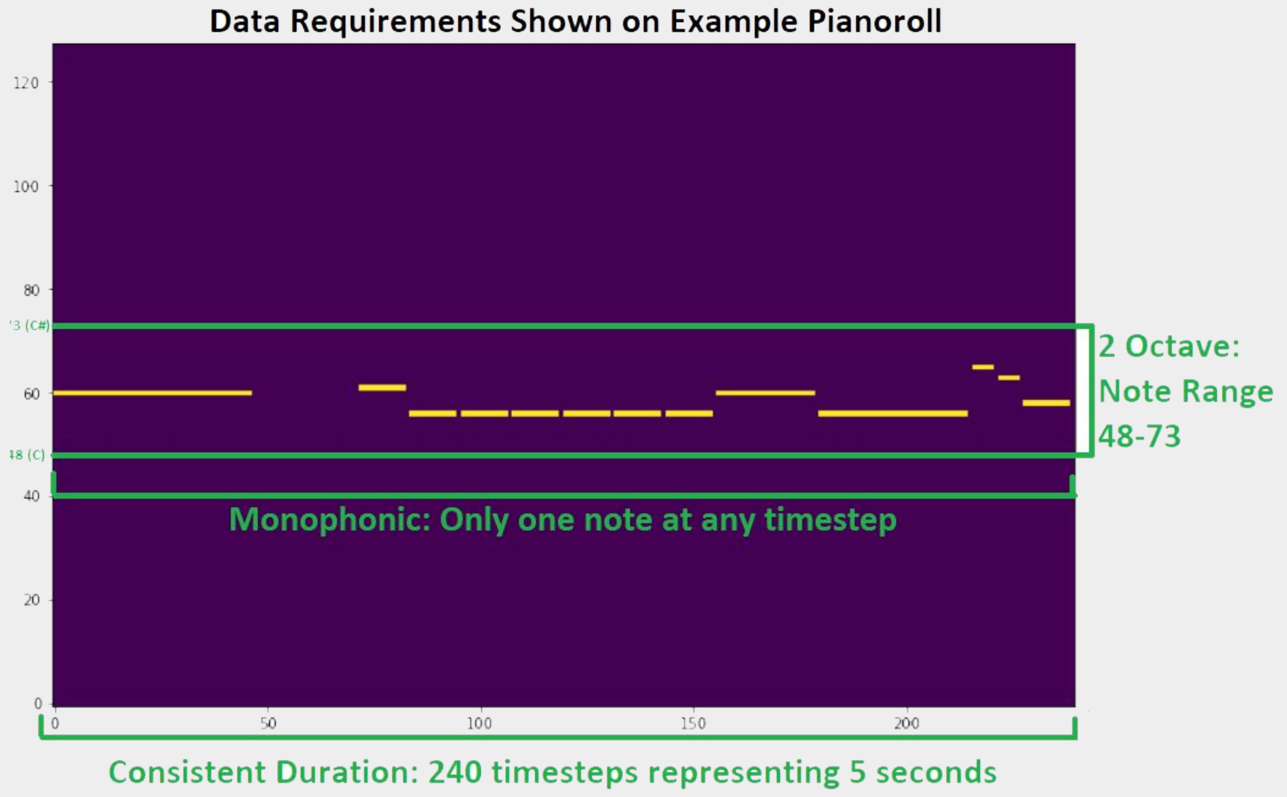
Goal

- Create a music composition application using neural networks.

Key Requirements

- **Input & Output Data:**
    - Data representation: (MIDI as Pianoroll)
    - Consistent duration: 5 seconds
    - Note range: 2 octaves
    - Musical Texture: Monophonic

Data Requirements Shown on Example Pianoroll



- **Model Architecture:**
    - Recurrent Neural Network (RNN)
- **User Interface:**
    - Interface between MIDI keyboard, computer, and RNN model
    - Return a 5 second extension of user's recorded 5 second melody

Design Challenges

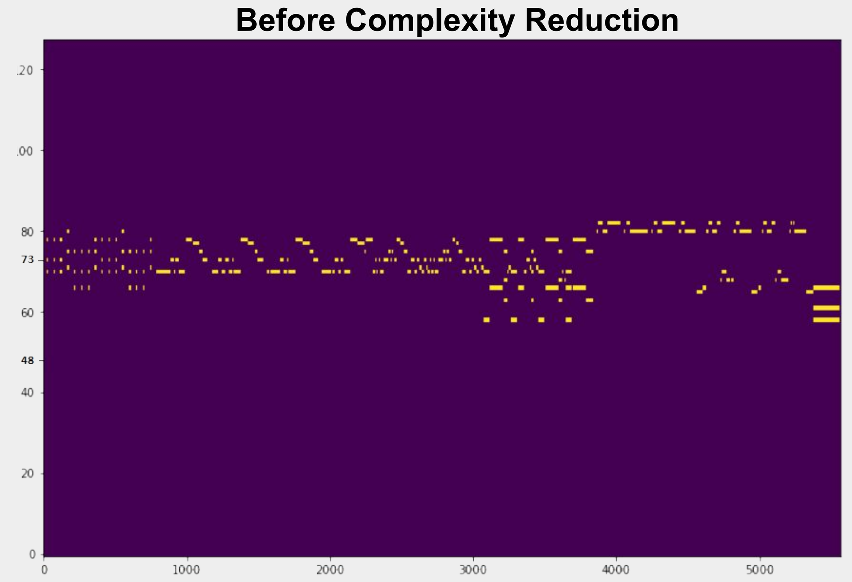
- **Data Representation of Music**
    - Typical ML tasks the team has experienced involve text, images, or spreadsheet data
    - MIDI in Pianoroll form was chosen after weighing alternatives
    - All functions that manipulated Pianoroll (other than read/write) were custom built by the team
- **Long model training times**
    - Complexity of dataset is 2^{128} combinations of notes per timestep (impossibly large)
    - Data transformations reduced complexity to 26 combinations (still days of training for a single model)
    - Reduction of scope: Focussed on training and testing one architecture (RNN) well rather than many with poorer results
- **Silent model outputs**
    - Music is more likely to be silent compared to playing any given note. Model generates silence in order to maximise accuracy
    - Custom accuracy metrics helped diagnose the problem
    - Improved data transformations were the solution

Data Processing → Model Training → User Interface

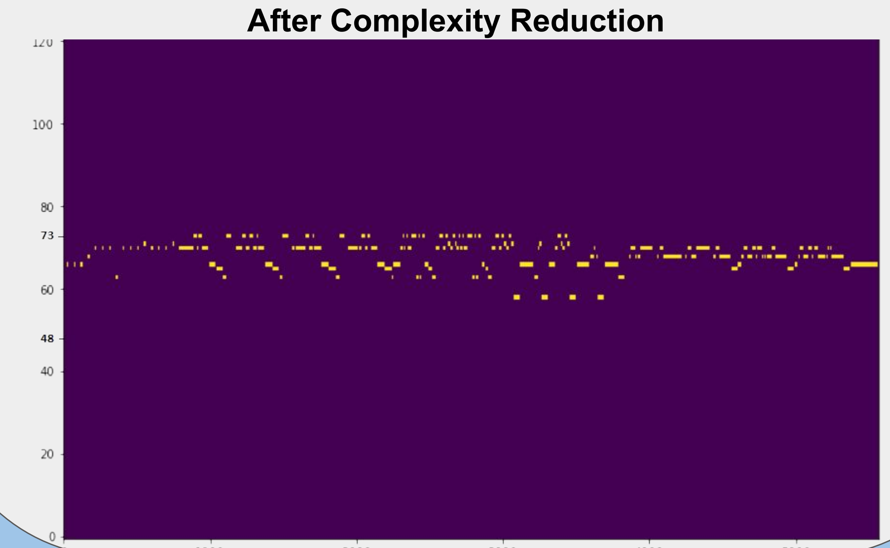
Data Processing

- Data representation: MIDI as Pianoroll
- **Acquired Dataset:**
    - 386 MIDI files (split into 6700) containing pop/classical music
- **Data Transformation**
    - Reducing Data Complexity
        - Polyphonic to monophonic
        - Note range reduction
    - Improving Data Quality
        - Split into 5s intervals
        - Tempo normalization
        - Silence threshold
    - Model Learnable Format
        - One-hot encoding
        - Melody/extension input/output pairs

Before Complexity Reduction



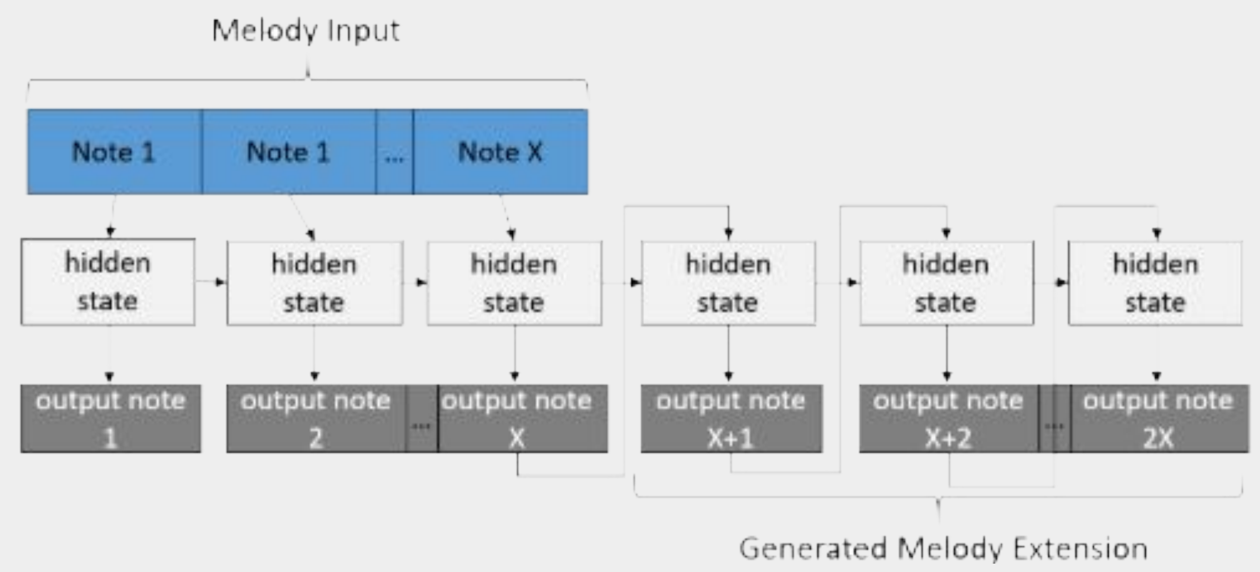
After Complexity Reduction



Model Training

Transformed Dataset → Forward Pass Function; Model → Forward Pass Function → Metrics → Model

- **Model Architecture**
    - Gated Recurrent Unit Neural Network (RNN variant)
- **Forward Pass**
    - Melodies are passed through the model, extensions are generated through process below



- **Metrics**
    - Generated extensions are compared to true extensions to calculate accuracy and loss
    - Weights of model are iteratively updated using loss
    - Custom Zero and Non-Zero Accuracy (accuracy of predicting silence or a specific note respectively) metrics developed to further assess model biases

User Interface

- Framework built to:
    - Interface the MIDI keyboard with Python code
    - Read and interpret MIDI events
    - Construct a MIDI file (recorded melody)
- Data processing executed:
    - Data transformation functions
    - Model forward pass
- Model output converted to MIDI and played back

Poll MIDI keyboard for 5 seconds → Construct a MIDI file for the recorded melody → Data transformations → Trained Model Forward Pass → Generated MIDI output

Results

- **Model Training Metrics**
    - Final model was trained for ~800 epochs (200 hours)
    - Accuracy metrics show how well model performed

Accuracy Type	Training	Validation
Zero	94.62 %	74.3572 %
Non-Zero	68.74294 %	45.6182 %
Overall	76.0272 %	48.6963%

- **Overall framework developed:**
    - Transformations of MIDI dataset to reduce complexity and pass through model
    - Model training process and custom metrics
    - Interface between keyboard, computer, and model
    - Generation of melody extensions based on user input in real time

Conclusions

- Accuracy metrics show that model was learning to generate music
- Although model can be improved, useful framework was developed to facilitate model learning and interfacing
- Future developers can mix and match datasets or models with our framework in order to find a better model, and plug their final model into our interface