Anish Shah & Nicholas Torba

Algorithm Description:
Midis are hexadecimal files for a synthesizer that contain instructions on how long notes (with velocity) an instrument (defined in a program) should play. As such to prevent manually transcribing these hexadecimal files, a package called PrettyMidi was used to read these instructions into a human-readable format and to easily manipulate the useful information. So first all the midi files were collected and were given a two letter indicator of the genre at the beginning of the name as our code will take those two letters as the label that corresponds to its data.
The indications are listed below:
CL = classical
CN = country
RO = rock
HH = hip-hop/r&b
MT = metal
ED = edm
PP = pop
Using glob, all the collected midi files stored in "Midi-Files" folder (must be named that) were iterated over to create a matrix which contains the note played at a sampled time splice of the song for an instrument.
Example displayed below:

	0.1 seconds	...	<end> seconds
Instrument 1	0	...	0
...	...	...	...
Instrument 128	42	...	56

So along the x-axis are each uniform time slice, the y-axis contains every possible instrument (128) that a midi file can contain where -1 means that there was a rest ,i.e. no note was played, and a number value corresponds to a note and its octave where the numbering matters and is uniform across notes. For this reason no centering or normalization was done.
To create this matrix a midi was opened as a PrettyMIDI object. The total amount of time slices were calculated and the instruments were known to have a total value of 128 so a 128, <Calculated time slices> numpy placeholder was made for the desired data matrix. For every instrument the piano roll was found with the same calculated time slice as before. This was done as the piano roll of an instrument is the 128, <Calculated time slice> matrix which shows what note will play at a time slice with a velocity. We looked for these matrices to only find the note that plays, but as many instruments play multiple notes simultaneously (chords), this had to be changed to match the form of our desired matrix. As such we found that in music theory the root note of triad chords can be algorithmically found. As such all the root notes of the triads were found. For anything not a triad, a random note was treated as the root chord to prevent bias of

any algorithm (i.e. only selecting lowest notes as root) on the net when training. The notes of a chord were replaced with a -1 unless they were the root note. The root note was given the value of the midi representation of that note. As each matrix had all -1's or a note for every column, the instrument can be vectorized to show the note played at time slices. Every instrument that was played was given their appropriate note vector, if not played they would stay -1.

Completed So Far:
All midi files needed for training and testing have been collected and labeled. The code written processes the midi files into a workable format to train the RNN.

RoadBlocks:
The biggest roadblock we have hit is the use of chords by certain instruments. Chords are when the same instruments play two or more notes simultaneously. The matrices used to represent the music that will be feed to our Neural Network do not allow for multiple numbers to represent the different notes used in chords. Unfortunately, the music theory behind the transformation of a chord into a single note is complex and we do not have any background in music theory. However, in our search for a music theory solution, we found that each chord has a root note. Finding the root note of a chord is much less complex than transforming the chord to an equivalent note.
In other projects we found, researchers simply threw away instruments that used chords. This is a complete waste of **precious** data. Instead, we decided that when possible, we would find and use the root note in place of chords for any instrument. As of now, the only chords we have been able to programmatically convert to the root note is triads. Triads are chords consisting of three notes. For any other chord, our current solution is to simply choose one of the notes in the chord at random. Randomly selecting a note from a chord is obviously not the best solution possible, but we do not want to lose data from instruments that use chords. In addition, triads are the most commonly used chords.
We are currently working with music students to see if it is possible for us to find the root note of more complex chords through code. We plan to have developed this algorithm by the time we train the RNN.
Another roadblock was that there is no way to represent both the note and velocity in two dimensions. At first thought, our idea was to expand the feature matrix into three dimensions to allow for the use of note and velocity, but quickly realized that RNN's cannot work with input in that format. Other projects have avoided this problem by using only one instrument so each channel could represent a note. However, that does not work for our genre classification project. To move forward, we decided that note/pitch at each time step is more important to genre classification than velocity. For that reason, we decided to structure the input matrices as time on the x-axis, instrument on the y-axis, and each number represents the note(or rest) at each time slice.

Experimental Plan:
- Fix the chord problem algorithmically via music theory if possible
- Create a vanilla RNN for us to train on to create a baseline in which can be optimized
    - This RNN will consist of nodes that will use the LSTM model

    - The concept of dropouts will be used to remove nodes that may skew the classification/music creation process to prevent outliers (such as irregular instruments in certain MIDI representations) from overfitting those nodes of instruments to misclassify
    - The input would be 128 as of now as there are 128 instruments per midi file.
    - The output would be similar to regression in which we would give a percentage of how much a midi file is of each genre. The genre it is the highest amount.
- Possible idea: How to represent the velocity along with the notes
    - After training and getting results on our RNN with the dataset described, we are going to “stack” our data matrix on top of a matrix of velocities.
    - Essentially we can feed these into the RNN (mostly for music) to create outputs that would define the note, and velocity for time slices.