**DATA PREPROCESSING**

In order to create music we firstly need to prepare data in a way a computer can learn.

Sound is air vibration and so is coded in mp3 and wav. How can computer understand notes? With MIDI.

With *music21* we can easily read MIDI files:

<music21.note.Note F3> 72.0  
<music21.chord.Chord A2 E3> 72.0  
<music21.chord.Chord A2 E3> 72.5  
<music21.note.Note E3> 73.5

The data splits into two object types: Notes and Chords. Note objects contain information about the pitch, octave, and offset of the Note.

Pitch refers to the frequency of the sound, or how high or low it is and is represented with the letters [A, B, C, D, E, F, G].

Octave refers to which set of pitches you use on a piano. Offset refers to where the note is located in the piece. As can be seen from the excerpt, the most common interval between notes in the midi files is 0.5. Therefore, we can simplify the data and model by disregarding the varying offsets in the list of possible outputs. It will not affect the melodies

Our prediction array will have to contain every note and chord object that we encounter in our training set.

**DATA PREPARATION**

We load each midi file in a Music21 stream object appending note/chord and pitch.

Input array: [“A2”, “E3”, “A2.C3.E3”, ….]

We then transform our input arrays from categorical to numerical (neural network perform much better with integer-based numerical data than string-based categorical data) and then perform one-hot-encoding.

**MODEL**

In our model we use four different types of layers:

LSTM layers is a Recurrent Neural Net layer that takes a sequence as an input and can return either sequences (return\_sequences=True) or a matrix.

Dropout layers are a regularisation technique that consists of setting a fraction of input units to 0 at each update during the training to prevent overfitting. The fraction is determined by the parameter used with the layer.

Dense layers or fully connected layers is a fully connected neural network layer where each input node is connected to each output node.

The Activation layer determines what activation function our neural network will use to calculate the output of a node.

model = Sequential()  
 model.add(LSTM(  
 256,  
 input\_shape=(network\_input.shape[1], network\_input.shape[2]),  
 return\_sequences=True  
 ))  
 model.add(Dropout(0.3))  
 model.add(LSTM(512, return\_sequences=True))  
 model.add(Dropout(0.3))  
 model.add(LSTM(256))  
 model.add(Dense(256))  
 model.add(Dropout(0.3))  
 model.add(Dense(n\_vocab))  
 model.add(Activation('softmax'))  
 model.compile(loss='categorical\_crossentropy', optimizer='rmsprop')

3 LSTM layers, 3 dropout layers, 2 dense layers, 1 activation layer. The last layer should always contain the same amount of nodes as the number different outputs our system has. This assures that the output of the network will map directly to our classes

The activation function is softmax: a function that takes as input a vector of K real numbers, and normalizes it into a probability distribution consisting of K probabilities. That is, prior to applying softmax, some vector components could be negative, or greater than one; and might not sum to 1; but after applying softmax, each component will be in the interval (0,1), and the components will add up to 1, so that they can be interpreted as probabilities.

To calculate the loss we use the “categorical crossentropy”

the cross entropy between two probability distributions p and q over the same underlying set of events measures the average number of bits needed to identify an event drawn from the set if a coding scheme used for the set is optimized for an estimated probability distribution q, rather than the true distribution p.

**MUSIC GENERATION**