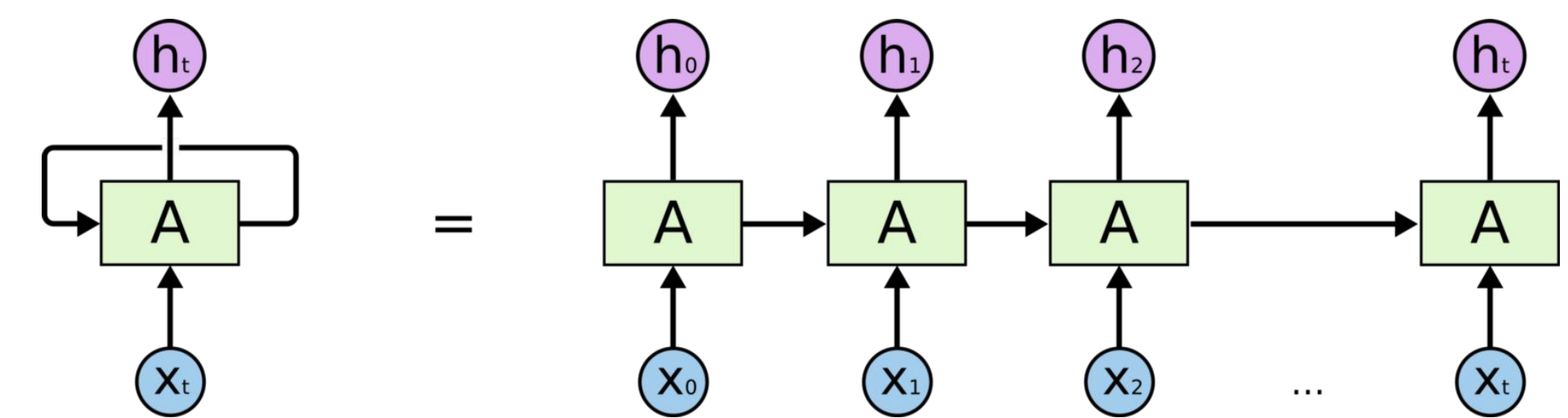


Build a model that takes in a fragment of music and outputs a predicted continuation of the music. The objective is analogous to sequence prediction, where given a segment it predicts the subsequent segments using learned weights from training examples. Musicians often improvise songs based off of a given musical fragment. We'd like to see if a machine can do the same.

The diagram illustrates the RNN model architecture for generating holiday music. It starts with an **Input Sequence** (represented by a musical staff) which is converted into a **Music21 Representation** (D4 D4 D5 A5 G4). This representation is then mapped to **Note Mapping** (50 50 54 11 32), which is fed into the **RNN** (Recurrent Neural Network) to produce the **New Sequence**.

Midi Dataset	Sequences	Unique notes	Files
Holiday Music	50168	324	82

## Recurrent Neural Network



The diagram illustrates the internal structure of two recurrent neural network units: the Long Short-Term Memory (LSTM) unit on the left and the Gated Recurrent Unit (GRU) unit on the right.

**LSTM (Left):** The LSTM unit takes an input  $x_t$  and the previous hidden state  $h_{t-1}$  as inputs. It uses four parallel gates to process these inputs: a forget gate (sigma), an input gate (sigma), a candidate cell state (tanh), and an output gate (sigma). The forget gate and input gate are multiplied element-wise with the previous hidden state and the candidate cell state, respectively, and then summed to produce the new cell state. The output gate is multiplied element-wise with the tanh of the new cell state to produce the final hidden state  $h_t$ .

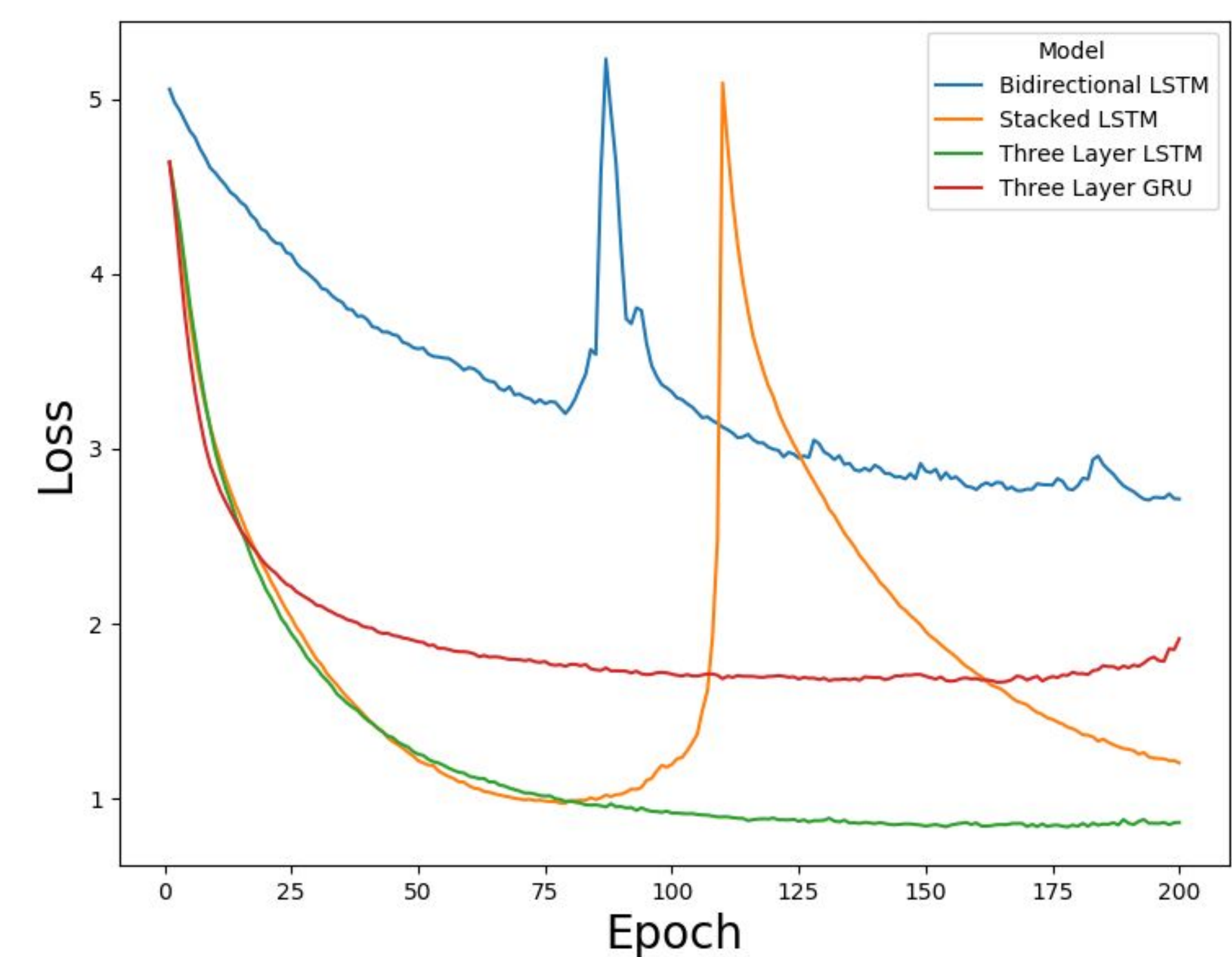
**GRU (Right):** The GRU unit takes an input  $x_t$  and the previous hidden state  $h_{t-1}$  as inputs. It uses two parallel gates: a reset gate (sigma) and an update gate (sigma). The reset gate is multiplied element-wise with the previous hidden state and then summed with the input to produce a candidate hidden state  $\tilde{h}_t$ . The update gate is multiplied element-wise with  $\tilde{h}_t$  to produce the final hidden state  $h_t$ .

$$H(p, q) = -\sum_x p(x) \log(q(x))$$

The diagram illustrates four different neural network architectures for sentiment classification, each consisting of a sequence of layers represented by colored boxes:

- Three Layered LSTM:** A vertical stack of layers: LSTM (green), Dropout (yellow), LSTM (green), Dropout (yellow), LSTM (green), Dense (blue), Dropout (yellow), Dense (blue), and Activation (purple).
- Three Layered GRU:** A vertical stack of layers: GRU (green), Dropout (yellow), GRU (green), Dropout (yellow), GRU (green), Dense (blue), Dropout (yellow), Dense (blue), and Activation (purple).
- Bidirectional LSTM:** A vertical stack of layers: Bidirectional LSTM (green), Dense (blue), Dropout (yellow), Dense (blue), Dropout (yellow), and Activation (purple).
- Stacked LSTM:** A vertical stack of layers: LSTM (green), LSTM (green), LSTM (green), Dropout (yellow), Dense (blue), and Activation (purple).

Comparisons were made on four different types of RNN units: three layered LSTM, three layed GRU, bidirectional LSTM and stacked LSTM.



	Best Epoch	Max Loss	Total Time
Three Layer LSTM	178	0.8394	4h 30min
Bidirectional LSTM	194	2.7070	2h 45min
Stacked LSTM	79	0.9755	4h 3min
Three Layer GRU	165	1.6670	3h 48min

- Categorical cross-entropy was used for loss
- The musicality of the output is subjected to human interpretation

Modified RNN to predict a segment of music instead of generating music

A test set of 10 midi songs were used to evaluate the output

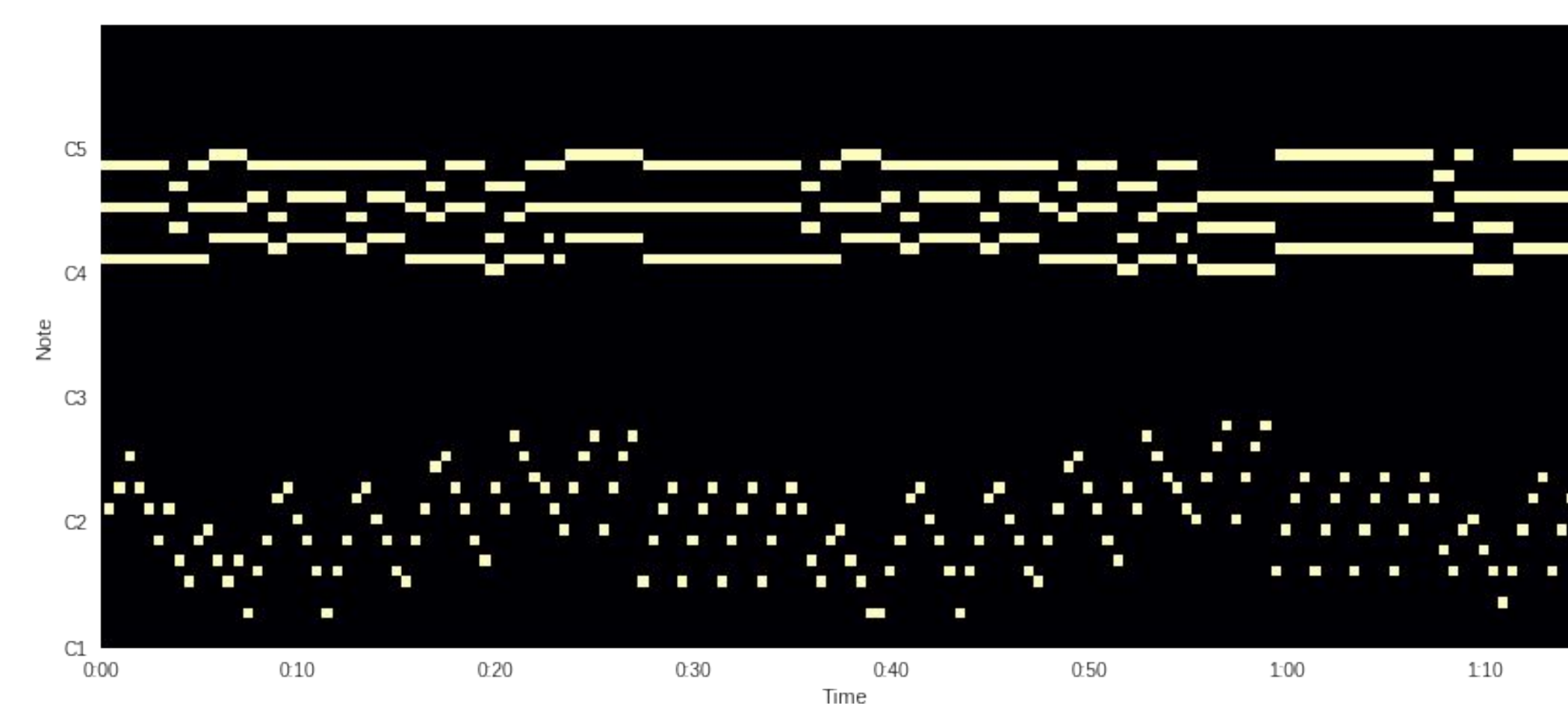
Similarity of predicted versus expected output is calculated by:

$$1 - \frac{EditDistance(expected, predicted)}{TotalNumberOfNotes}$$

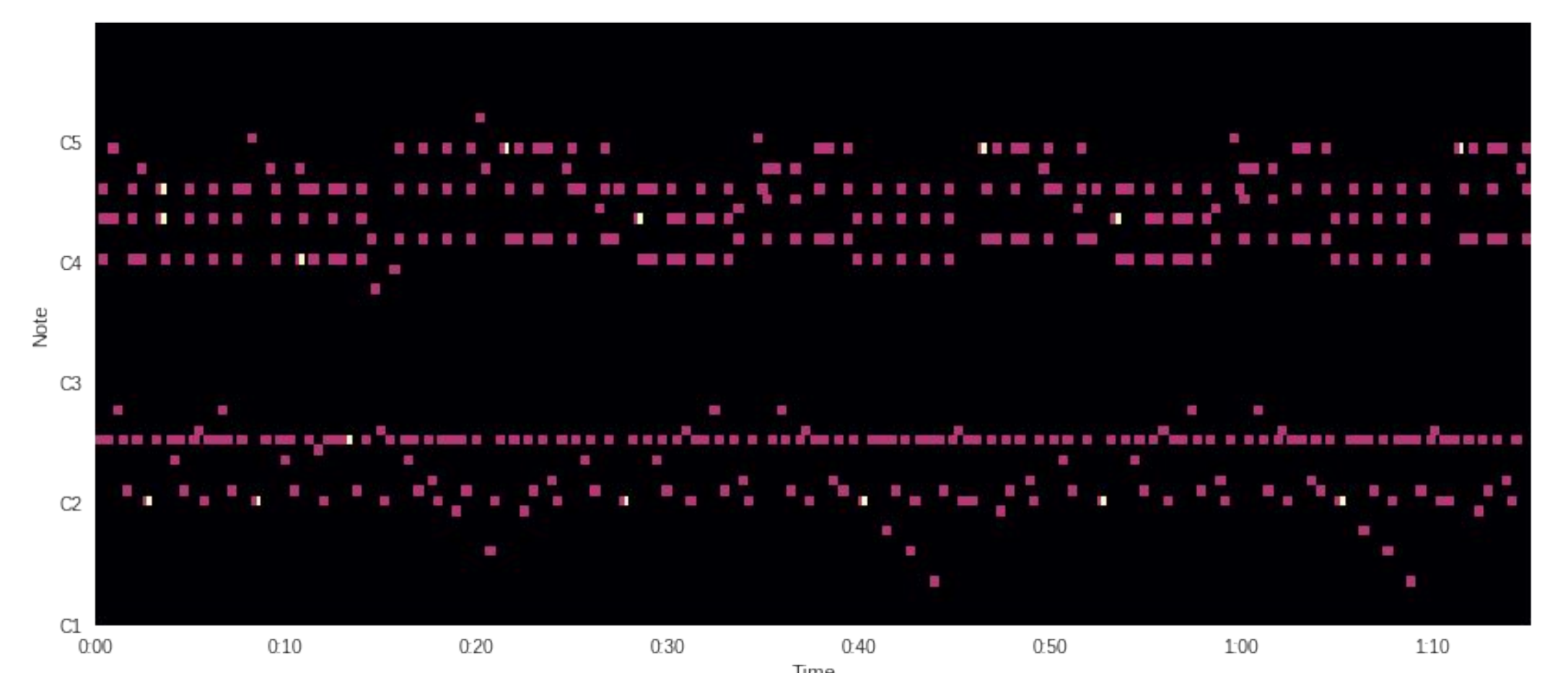
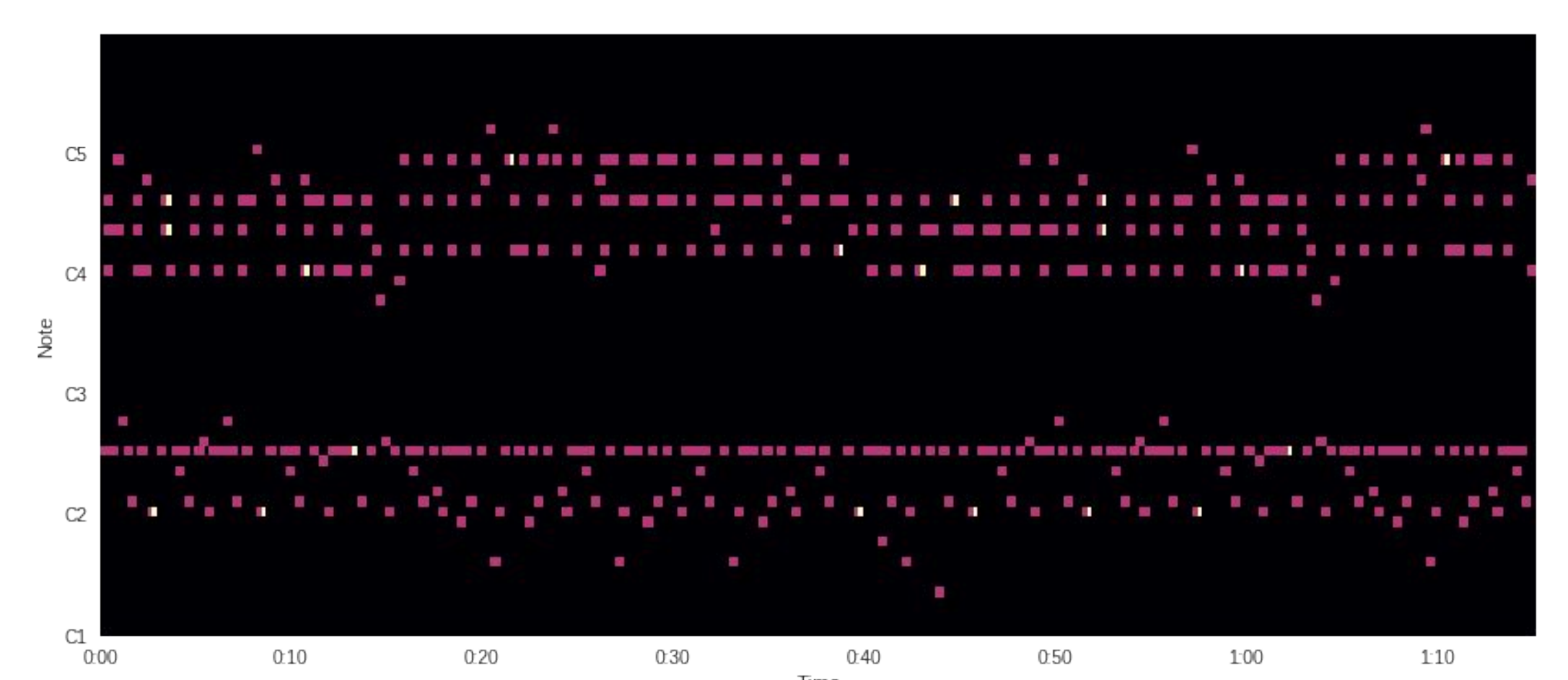
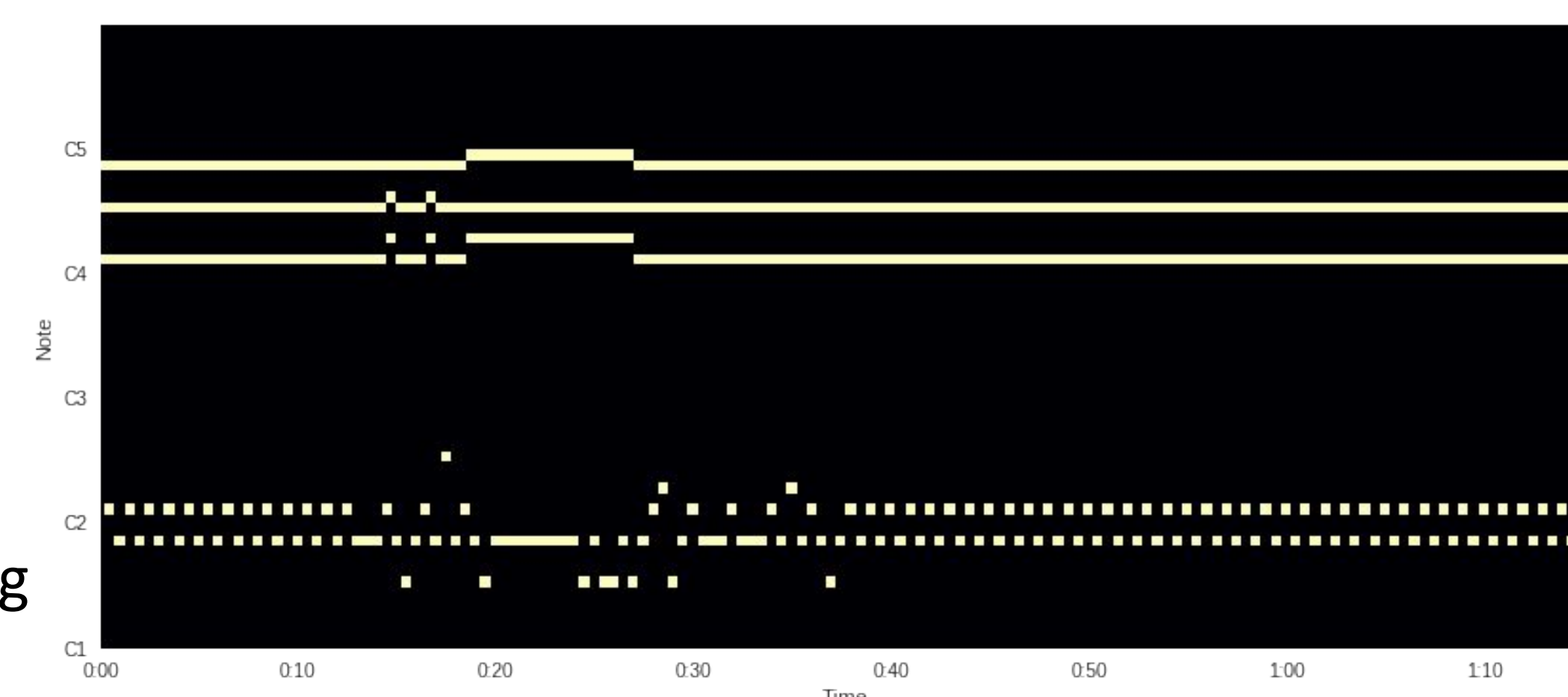
Three Layer LSTM	23.8%
Bidirectional LSTM	23.5%
Stacked LSTM	24.7%
Three Layer GRU	22.2%

The network is unable to produce notes that it has never seen before in the training set

Expected:



Prediction:



Typical outputs from the RNN.