Finding an Embedding for Music Auto-Complete: An LSTM Approach

Nathaniel Patterson
Advisor: Dr. Rodney Summerscales

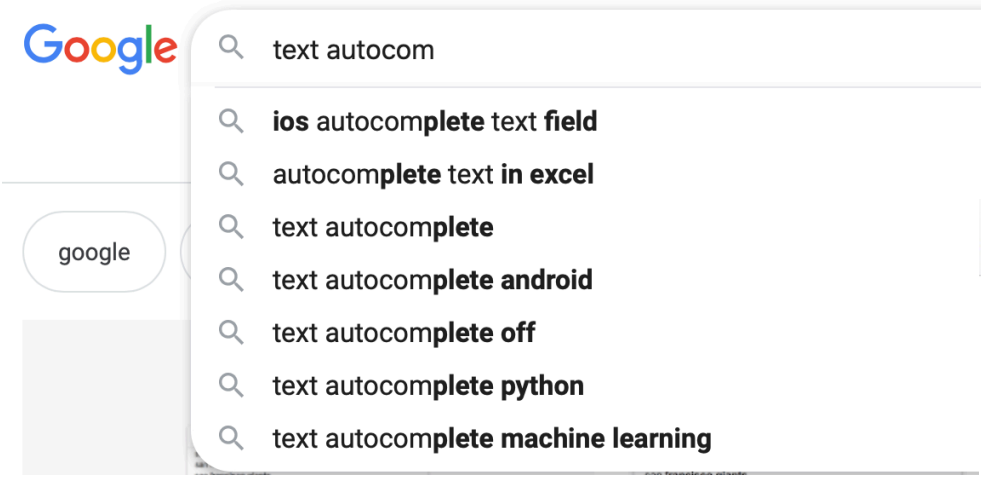
Department of Computing
School of Business Administration, Andrews University

J.N. Andrews Honors Program
Andrews University

Statement of Research Goals

- Introduce Music Autocomplete as a new problem
- Explore two embedding strategies using a Long Short Term Memory Recurrent Neural Network
- Provide commentary on the philosophical consideration of computer generated art and the ”predictability” of an artist

Understanding Autocomplete

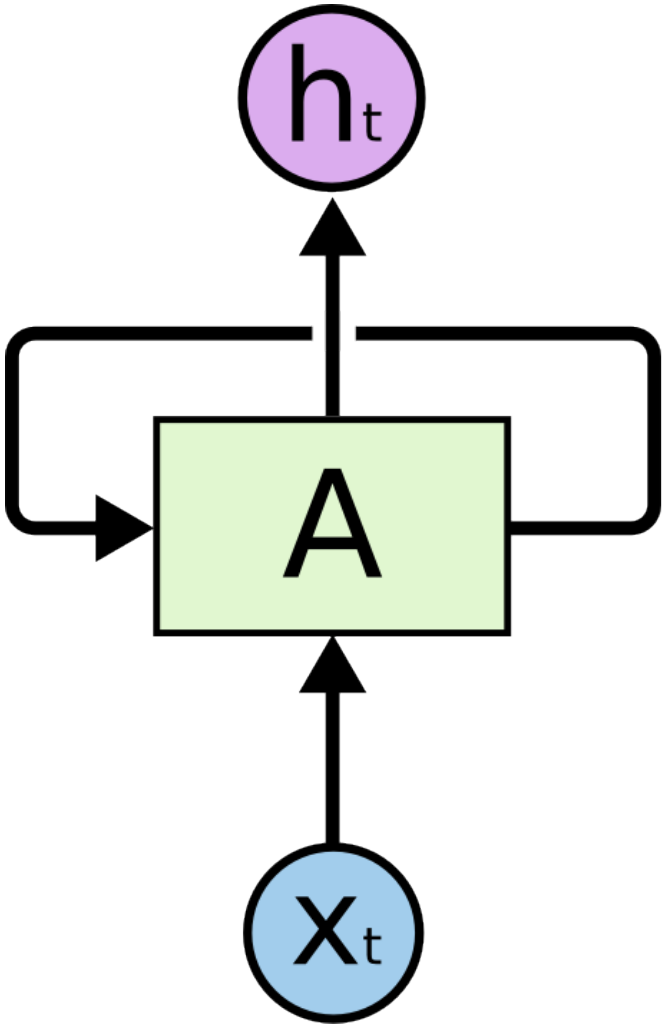


Background Music Autocomplete

- Polyphonic music generation using “Bi-Axial” LSTM (Johnson 2017)
- “Bachbot” was user tested to discriminate between Bach compositions and generated compositions (Liang, et al 2017)
- Text autocomplete is well-developed:
  - Story Scrambler (Pawade, et al 2018) focused on LSTM architecture and evaluated generated text stories when trained on books in the public domain
  - Many other models and embeddings have seen great success in research and industry applications

Background Music Autocomplete

- New Problem
- Similar to:
  - Music Generation
  - Text Autocomplete
- LSTM-RNN (Hochreiter 1997) approach provides a good baseline

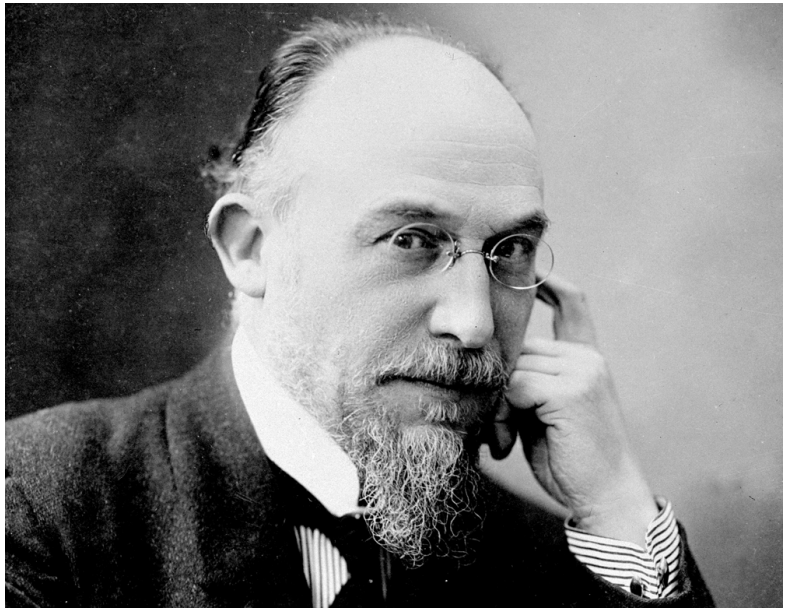


Methodology

- Select Artist
- Gather Data
- Prepare Data
- Build Model
- Train the Model
- Evaluate the Model

Note on Erik Satie

- Lived 1866-1925, France
- Witty, Satirical style in league with Surrealist Movement
- Music characterized by a refusal to be involved in any degree of transcendent significance, while disregarding traditional tonal norms
  - *Trois morceaux en forme de poire (1903)*
  - *Sports et divertissements (1914)*
- We recognized 157 piano compositions



Data

- Data is formatted as Musical Instrument Digital Interface (MIDI) files
- Classical music is in the Public Domain
- We were able to locate MIDI files for roughly 48% (76) of the 157 piano compositions
  - Some compositions of shorter length were combined with other movements. Total file count was 49 MIDI files
- Music21 Python library used to translate between MIDI and string

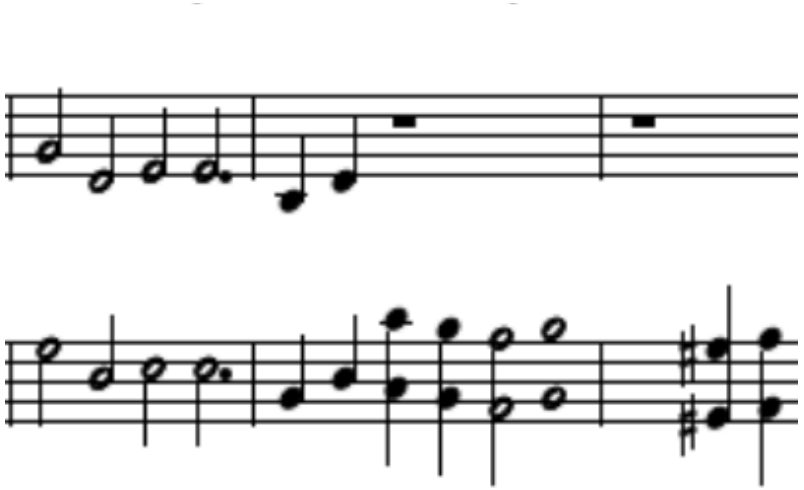
Data Representation

- All songs in training set are embedded as a string of notes with spaces separating each timestep

```
'A4 C5 F#5F#4 E5D5E4 F#5F#4 G5B4G4 G4 A4 D5D4 C5B4C4 D5D4 E5G4E4 G4 G4F#4G1G2 E4 B4D4A1A2 A4 C
5F#4B-1B-2 E4 D5D4B1B2 F#1F#2 E4C4A1A2 B1B2 C4C2C3 C4G#3 C#4B1B2 A3 D4B3A1A2 G1G2 E4C#4F#1F#2
E4C#4 F#4E1E2 D4 A4F#4 G4E4 E4C#4 E-4B3C#3 C#4A3 B3G#3E-3 F#4 E-4E3 C#4G3 E3G3B3 E1E2 E3A3C4 A
1A2 G3B3D4 G1G2 G5B5 A4 F#5A5 C5 E5G5 D5 D4F#4 D5 C4E4 G4 B3D4 B4 B3F#1F#2 B2D3F#3 E1E2 A2C3E3
G2B2D3D1D2 B3F#1F#2 B2D3F#3 E1E2 A2C3E3 G2B2D3D1D2 D4 F#4 B4B2B3 A4G4A2A3 B4B2B3 C5E4C3C4 C4 D
4 G4G2G3 F#4E4F#2F#3 G4G2G3 A4C4A2A3 C4 C4B3C1C2 A3 E4G3D1D2 D4 F#4B3E-1E-2 A3 A3G4D4E1E2 B3G4
```

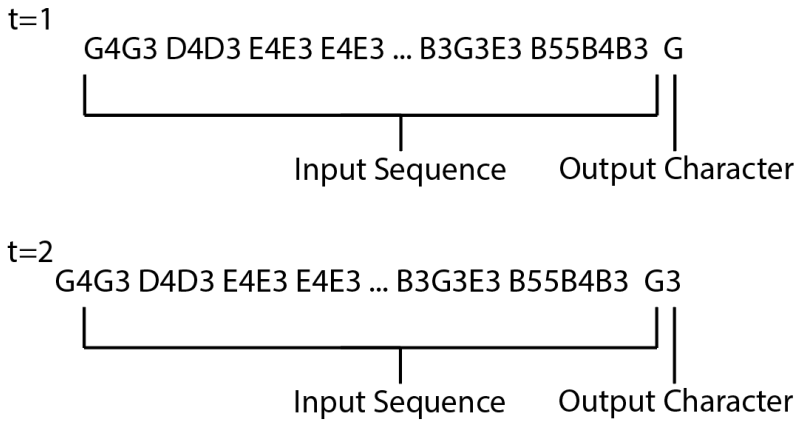
- Every note or chord played at a timestep regardless of note stem or clef is considered one chord for our purposes

Data Representation

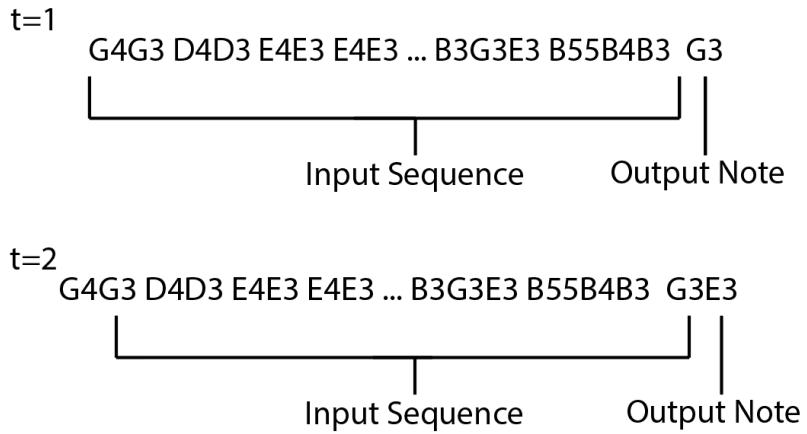


G4G3 D4D3 E4E3 E4E3 B3B2 D4D3
C#4C#3 B3B2 A2A3 B2B3 G#2G#3 A2A3…

Character strategy:



Note Strategy:



Data Embeddings

- Test two different embedding strategies
- Notes as Characters and Notes as Objects
- Both take a string approach with timesteps separated by space characters

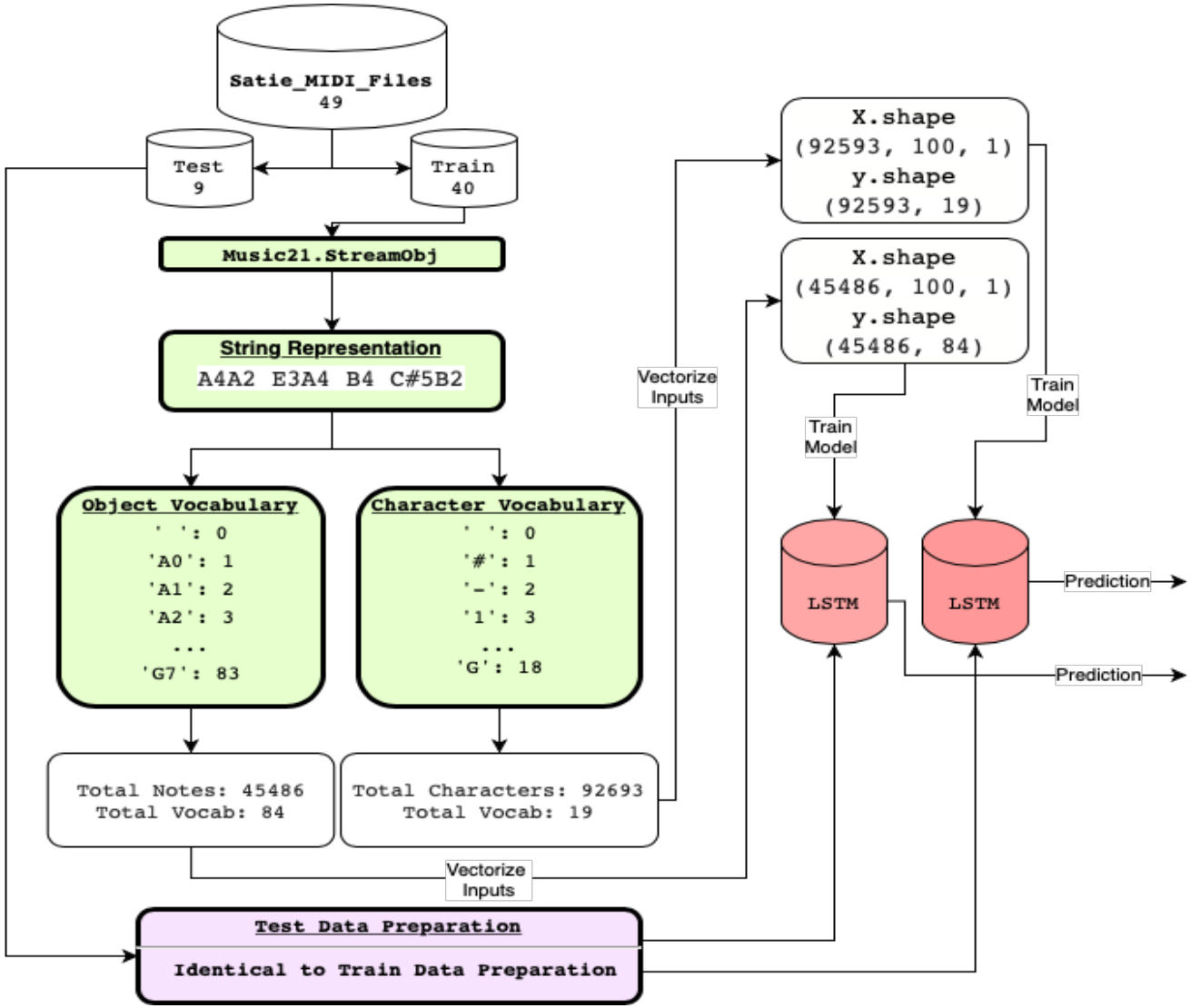
```
{' ': 0,
 'A0': 1,
 'A1': 2,
 'A2': 3,
 'A3': 4,
 'A4': 5,
 'A5': 6,
 'A6': 7,
 'B-0': 8,
 'B-1': 9,
 'B-2': 10,
 'B-3': 11,
 'B-4': 12,
 'B-5': 13,
 'B-6': 14,
 'B0': 15,
 'B1': 16,
 'B2': 17,
 'B3': 18,
 'B4': 19,
 'B5': 20,
 'B6': 21,
 'C#1': 22,
 'C#2': 23,
 'C#3': 24,
 'C#4': 25,
 'C#5': 26,
 'C#6': 27,
 'C#7': 28,
 'C-1': 29,
 'C1': 30,
 'C2': 31,
 'C3': 32,
 'C4': 33,
 'C5': 34,
 'C6': 35,
 'C7': 36,
 'D1': 37,
 'D2': 38,
 'D3': 39,
 'D4': 40,
 'D5': 41,
```

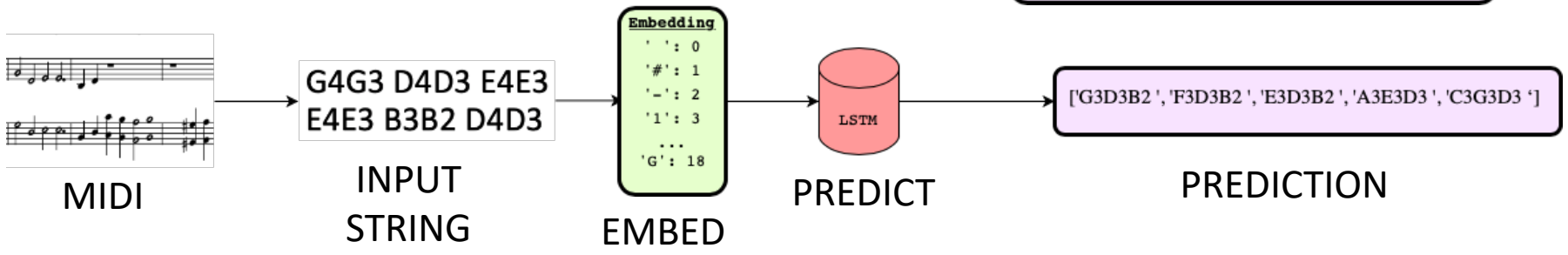
```
{' ': 0,
 '#': 1,
 '-': 2,
 '0': 3,
 '1': 4,
 '2': 5,
 '3': 6,
 '4': 7,
 '5': 8,
 '6': 9,
 '7': 10,
 '9': 11,
 'A': 12,
 'B': 13,
 'C': 14,
 'D': 15,
 'E': 16,
 'F': 17,
 'G': 18}
```

Training Process

- Data split into Training and Testing files 80/20
- Training files were split into training and validation sets
- Loss over validation set was tracked to avoid overfitting
- Process repeated three times and results averaged

Data Flow


Satie_MIDI_Files
49
Test
9
Train
40
Music21.StreamObj
String Representation
A4A2 E3A4 B4 C#5B2
Object Vocabulary
' ': 0
'A0': 1
'A1': 2
'A2': 3
...
'G7': 83
Character Vocabulary
' ': 0
'#': 1
'-': 2
'1': 3
...
'G': 18
Total Notes: 45486
Total Vocab: 84
Total Characters: 92693
Total Vocab: 19
Vectorize Inputs
Vectorize Inputs
X.shape
(92593, 100, 1)
y.shape
(92593, 19)
X.shape
(45486, 100, 1)
y.shape
(45486, 84)
Train Model
Train Model
LSTM
LSTM
Prediction
Prediction
Test Data Preparation
Identical to Train Data Preparation


G4G3 D4D3 E4E3
E4E3 B3B2 D4D3
Embedding
' ': 0
'#': 1
'-': 2
'1': 3
...
'G': 18
LSTM
['G3D3B2 ', 'F3D3B2 ', 'E3D3B2 ', 'A3E3D3 ', 'C3G3D3 ']
MIDI
INPUT STRING
EMBED
PREDICT
PREDICTION

Model Evaluation: Char-LSTM

- Over each independent trial we saw a <20% success rate.
- Success was determined by whether or not the true note occurred in the top five suggestions list from the data.
- Example output:

satogv02.mid

G4G3 D4D3 E4E3 E4E3 B3B2 D4D3 C#4C#3 B3B2 A2A3 B2B3 G#2G#3 A2A3 F#2F#3 E2E3 F#2F#3 A2A3 B2B3 D3D4 C#3C#4 B2B3 B2B3 A2A3 G6D6B5G5G4D4B3G3 D6B5F#5D5D4B3F#3D3 E6B5G5E5E4B3G3E3 E6B5G5E5E4B3G3E3 B5G5D5B4B3

Input Sequence

['G3D3B2 ', 'F3D3B2 ', 'E3D3B2 ', 'A3E3D3 ', 'C3G3D3 ']

Top 5 Predictions

success: 'G3D3B2'

True Value

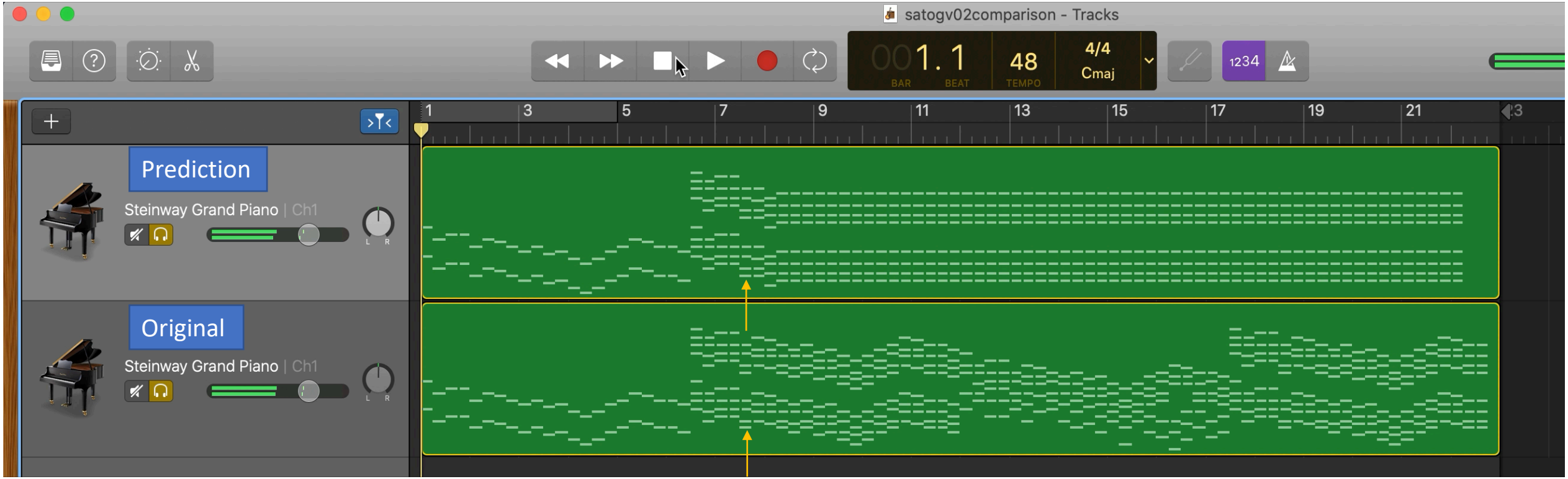
satgno05.mid

D5 G5G2 D4B3G3 F#5 D5 F#5 E5C5C3 E5 E5 E4C4G3 A2 E4C4A3 E2 F#5 G5 A5 E4B5B3G3 C6 D6 E6 F#6 E6 D6G2 B5 D6 D4B3G3 C6 D6 A5D2 D4A3F#3 B5 F#5B1 B4D4B3F#3 D5 E5E2 B4D5 E5F#5 D5 G5 E4E5B3G3 D5 A5D2 D4A3F#3

['E4 ', 'C4 ', 'G4 ', 'B4 ', 'D4 ']

correct note/chord is: B5B1

Visualizing Predictions: Char-LSTM



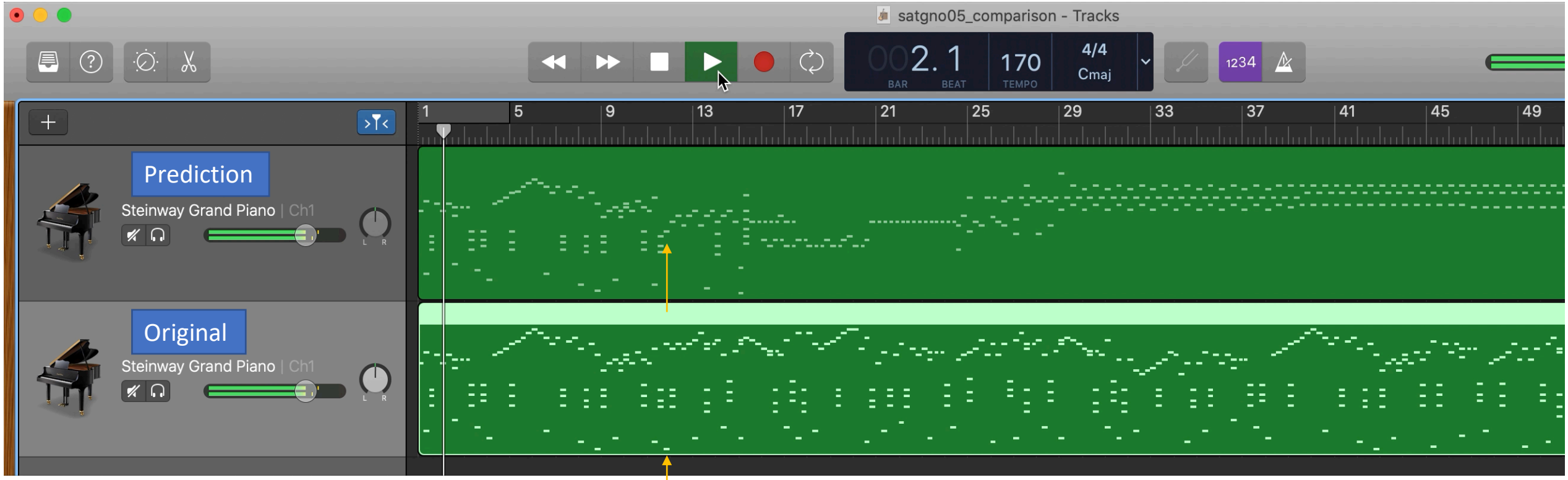
satogv02.mid

G4G3 D4D3 E4E3 E4E3 B3B2 D4D3 C#4C#3 B3B2 A2A3 B2B3 G#2G#3 A2A3 F#2F#3 E2E3 F#2F#3 A2A3 B2B3 D3D4 C#3C#4 B2B3 B2B3 A2A3 G6D6B5G5G4D4B3G3 D6B5F#5D5D4B3F#3D3 E6B5G5E5E4B3G3E3 E6B5G5E5E4B3G3E3 B5G5D5B4B3

['G3D3B2 ', 'F3D3B2 ', 'E3D3B2 ', 'A3E3D3 ', 'C3G3D3 ']

success: 'G3D3B2'

Visualizing Predictions: Char-LSTM



satgno05.mid

D5 G5G2 D4B3G3 F#5 D5 F#5 E5C5C3 E5 E5 E4C4G3 A2 E4C4A3 E2 F#5 G5 A5 E4B5B3G3 C6 D6 E6 F#6 E6 D6G2 B5 D6 D4B3G3 C6 D6 A5D2 D4A3F#3 B5 F#5B1 B4D4B3F#3 D5 E5E2 B4D5 E5F#5 D5 G5 E4E5B3G3 D5 A5D2 D4A3F#3

['E4 ', 'C4 ', 'G4 ', 'B4 ', 'D4 ']

correct note/chord is: B5B1

Model Evaluation: Note-LSTM

```
fantaisie_valse.mid
G#4C#3 B-4F4C#4G#3 C5 C#5 E-5G#2 F5 F#5F4C#4G#3 G#5 B-5 F5 G#5E-3 B-4 C5F#4C4G#3 G#5 C5 C#5 G#5G#2 B-4 G#4F#4C4G#3 F#4C4G#3 G#4C#3
Prediction: [' ', ' ', ' ']

Enfantillages_Pittoresques1.mid
E4 C4 B3 A3 F4 D4 B3 A3 G4 D4 B3 G3 G4 D4 B3 G3 E4 C4 B3 A3 F4 D4 B3 A3 G4 D4 B3 G3 G4 D4 B3 G3 A3 B3 C4
Prediction: [' ', ' ', ' ']

valse_ballet.mid
F4 F#4 G4 A4 C5 G4 B-4C3 A4E-4A3F3 C5 D5E-4A3F3 A4 C5F2 A4E-4A3F3 C5 D5 A4 C5B-2 B-4 F5D4B-3F3 F4D4B-3F3 F2 F4D4B-3F3 A4 B-4 G4
Prediction: [' ', ' ', ' ']

morceaux_en_forme_de_poire_3.mid
C3C2 C5 E-4G3C4 G#2G#1 C5 G#3E-4C4 C5C3C2 E-5 E-4G3C4 C5 E-2E-1 B4 C5 C#4B3E-4G3 C#5 E-5G#2G#1 F#3E-4C4 B-2B-1 C#5G3C#4F4 C5C3C2 E-
Prediction: [' ', ' ', ' ']

satieson.mid
A4A2 E3A4 B4 C#5B2 E5E3 A4C#3 E3A4 B4 C#5D3 F#5E3 A5A3 G#5 F#5A2 E5 G#3D5 C#5 A2B4 A4 F#3B4 A4 D3G#4 F#4 E3E4 D3 C#3 B2 A2A3
Prediction: [' ', ' ', ' ']

gymsat01.mid
G2 F#4D4B3 D2 F#4C#4A3 G2 F#4D4B3 D2 F#4C#4A3 G2 F#4D4F#5B3 A5 G5D2 F#4C#4F#5A3 C#5 B4G2 F#4D4C#5B3 D5 A4D2 F#4C#4A3 F#4G2 D4B3 D2
Prediction: [' ', ' ', ' ']

satogv04.mid
F4F3 D4D3 E4E3 C4C3 C4C3 D4D3 A3A2 C4C3 D4D3 E4E3 D4D3 C4C3 F4F3 D4D3 E4E3 C4C3 C4C3 D4D3 A3A2 C4C3 B3B2 B3B2 A3A2 F6
Prediction: [' ', ' ', ' ']

Avant-dernières_3.mid
A4 D5G5 A4 D5G5 A4 D5G5 A4 D5G5 A4 D5G5 A4 D5G5 A4B3 D5G5 A4 D5G5E4 A4 D5G5 A4F#4 D5G5 A4 D5G5 A4 D5G5C#4 A4D4 D5G5
Prediction: [' ', ' ', ' ']

flabbypr_1-3.mid
F4 A4 D4 B4 F4 D5 A4 B4 F4 A4 D4 B4 F4 D5 A4 B4 F4D3D2 A4 D4 B4 F4 D5 A4 B4 G#4C2C3 C5 F4 D5 G#4F1F2 F5 C5 D5
Prediction: [' ', ' ', ' ']
```

Model Evaluation: Note-LSTM

- Results not reportable
  - Embedding strategy was not well suited for an LSTM
- Too little data and too many notes
- String representation overfitted to space characters

```
count=0
for item in objectsNum:
  if item == 0:
    count+=1
count/len(objectsNum)
```

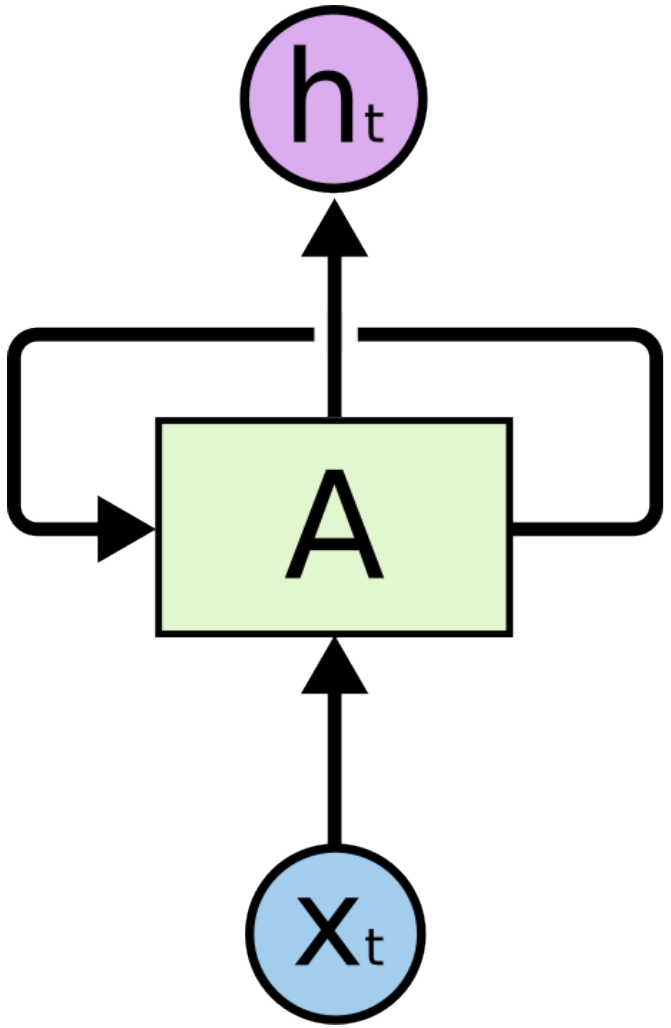
```
0.3194008502522945
```

Discussion

- Saw interesting results with the Character based model with a <20% success rate over three trials
- Important to let embeddings drive model choice
- Is it fair to assume a model could learn an artist’s style solely on his/her works?
- Would the LSTM model do better if trained on every Satie piece?

What is a good prediction?

- Should more attention be given to how we determine success?
- Is it fair to penalize just because it didn't get the exact chord or note?
- Other metrics



Future Work

- A non-string approach that used an 88-length vector for each timestep with a 1 for each note played using another model architecture should be explored
- Attempting with other artists
- Develop a good Network Architecture for this problem
  - RBM?
  - LSTM?
  - GAN?
- Determine a good metric for analyzing results. What is fair to the model?
- Interdisciplinary work

Acknowledgements

- Special thank you to the Honors Program and Department of computing for their support in conducting this research and presenting in the COVID-19 era.
- Special thanks to Dr. Summerscales for his advice and assistance
- Special thank you to Musicalion for free use of their MIDI library for research purposes

Selected Bibliography

Hochreiter, Sepp, and Jürgen Schmidhuber. “Long Short-Term Memory.” *Neural Computation*, vol. 9, no. 8, 1997, pp. 1735–1780., doi:10.1162/neco.1997.9.8.1735.

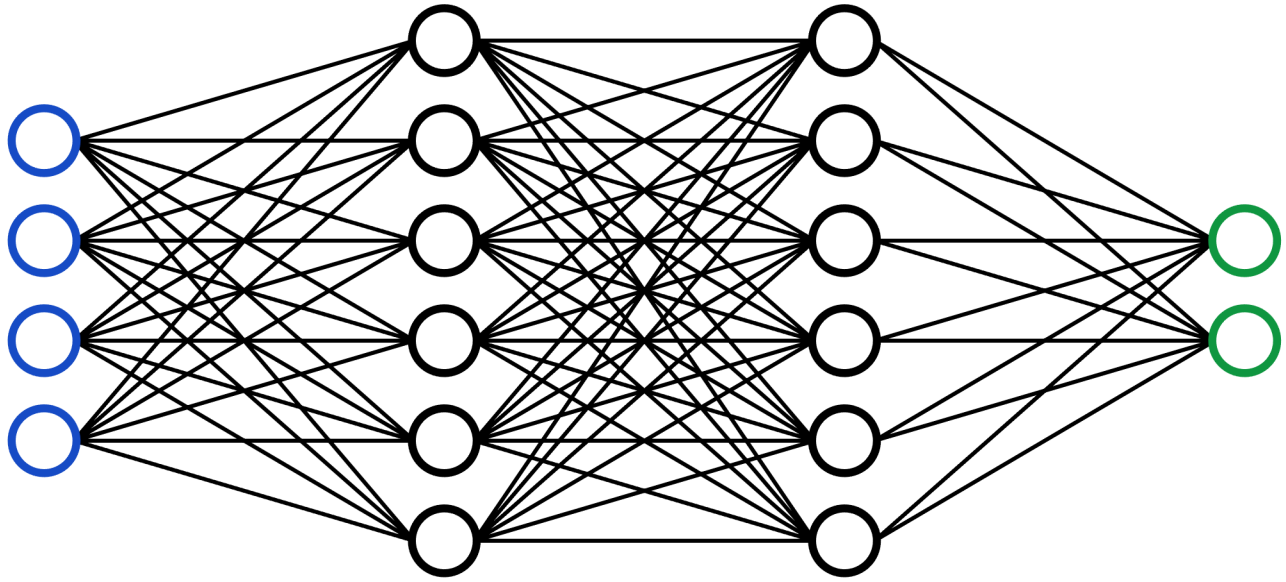
Johnson, Daniel D. “Generating Polyphonic Music Using Tied Parallel Networks.” Computational Intelligence in Music, Sound, Art and Design Lecture Notes in Computer Science, 2017, pp. 128–143., doi:10.1007/978-3-319-55750-2_9.

Liang, Feynman, et al. “ 18th International Society for Music Information Retrieval Conference.” ISMIR, *Proceedings of the 18th ISMIR Conference, Suzhou, China, October 23-27, 2017*, 2017, pp. 449–456.

Pawade, Dipti, et al. “Story Scrambler - Automatic Text Generation Using Word Level RNN-LSTM.” *International Journal of Information Technology and Computer Science*, vol. 10, no. 6, 2018, pp. 44–53., doi:10.5815/ijitcs.2018.06.05.

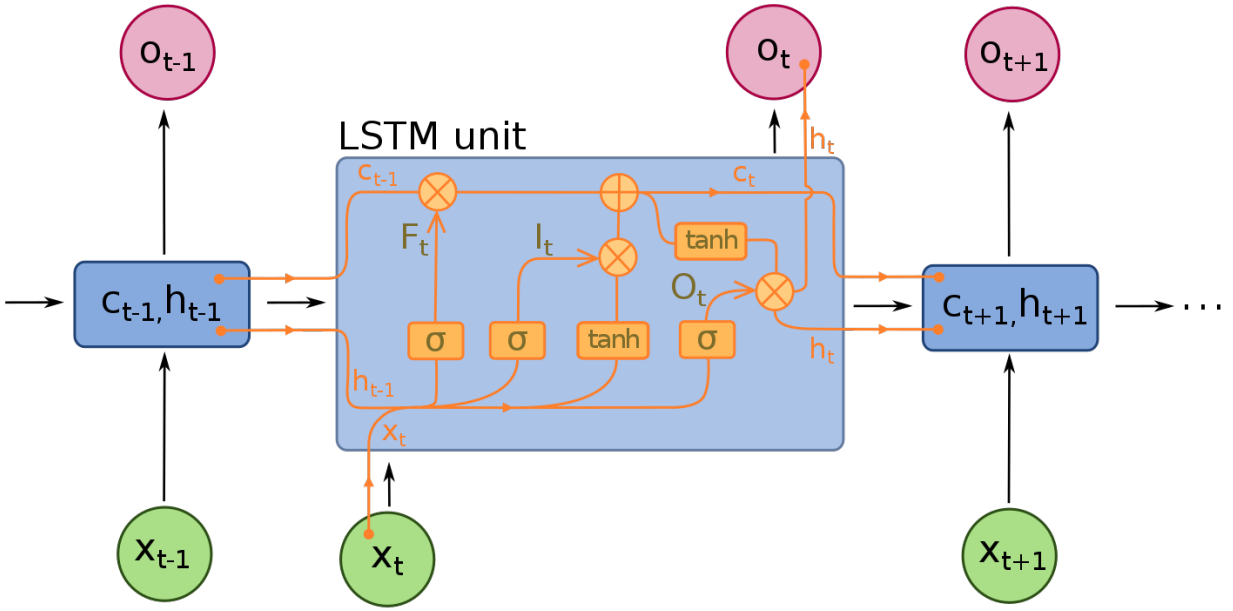
Neural Networks

- Goal: learn a function that maps input to a given output
- Different architectures better suited to different problems
- Recurrent Neural Network (RNN) architecture suited to sequential data



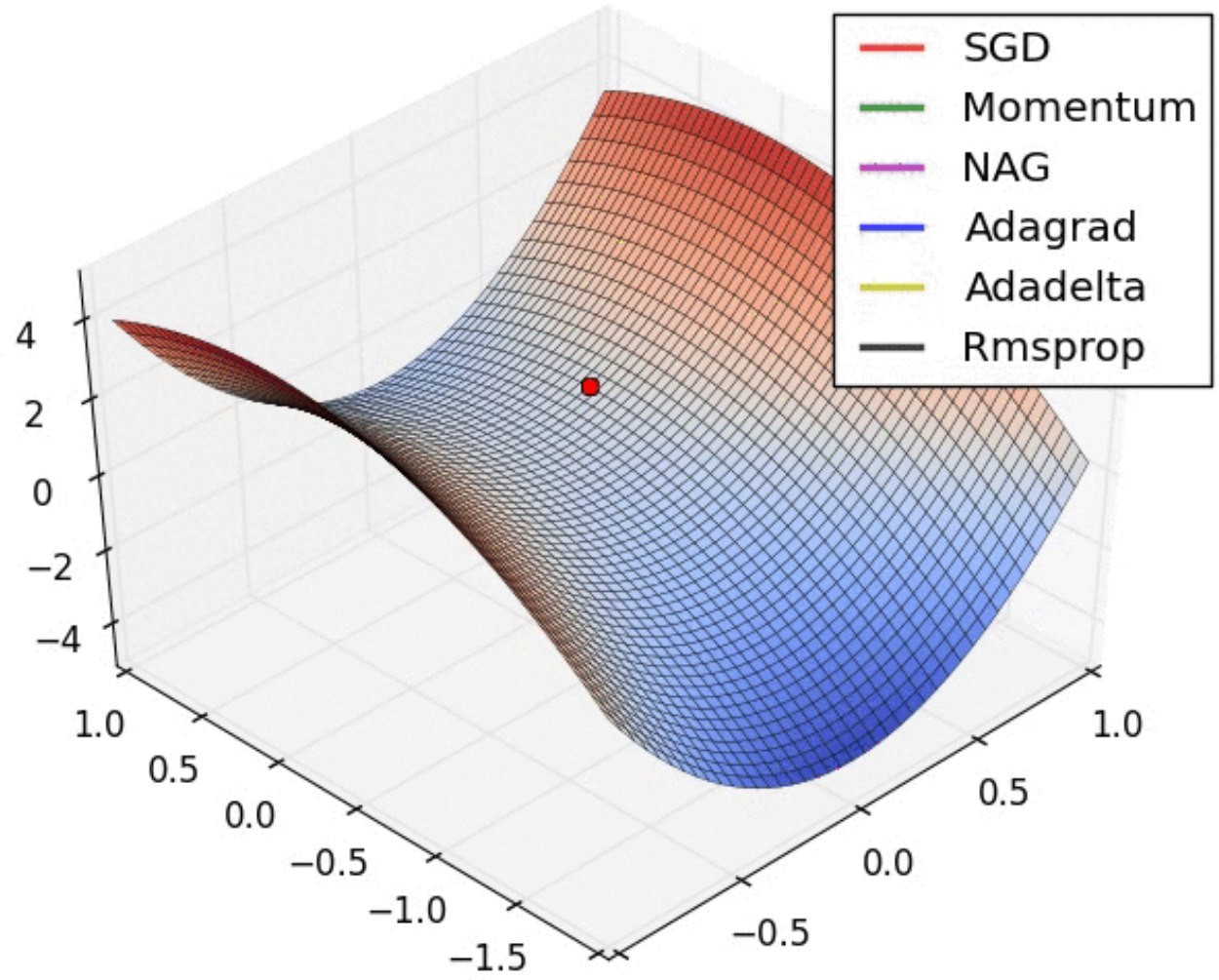
Note on Long-Short Term Memory RNN

- LSTM-RNN solved the issue of "Long-Term Dependencies"
- Neural Network for modeling sequential data
- Along with learning function, also learns what values to keep and forget



How do models learn?

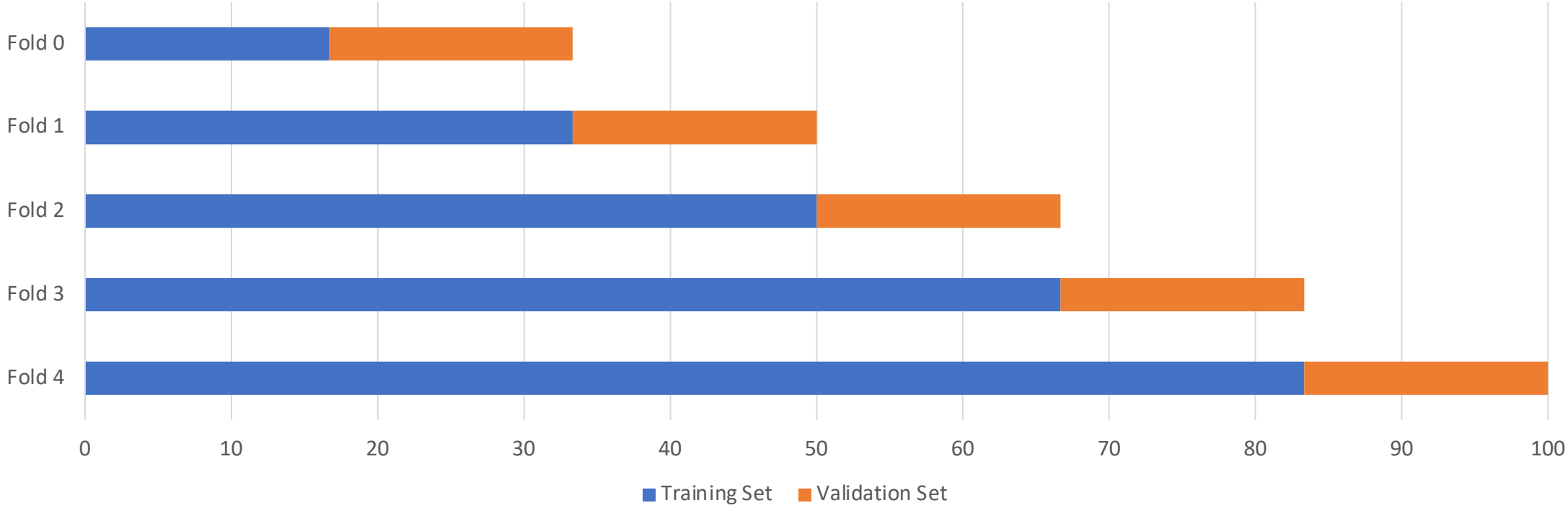
- Optimization
- Gradient Descent
- Optimize on loss taken over predicted values



https://ruder.io/content/images/2016/09/saddle_point_evaluation_optimizers.gif

Time Series Split

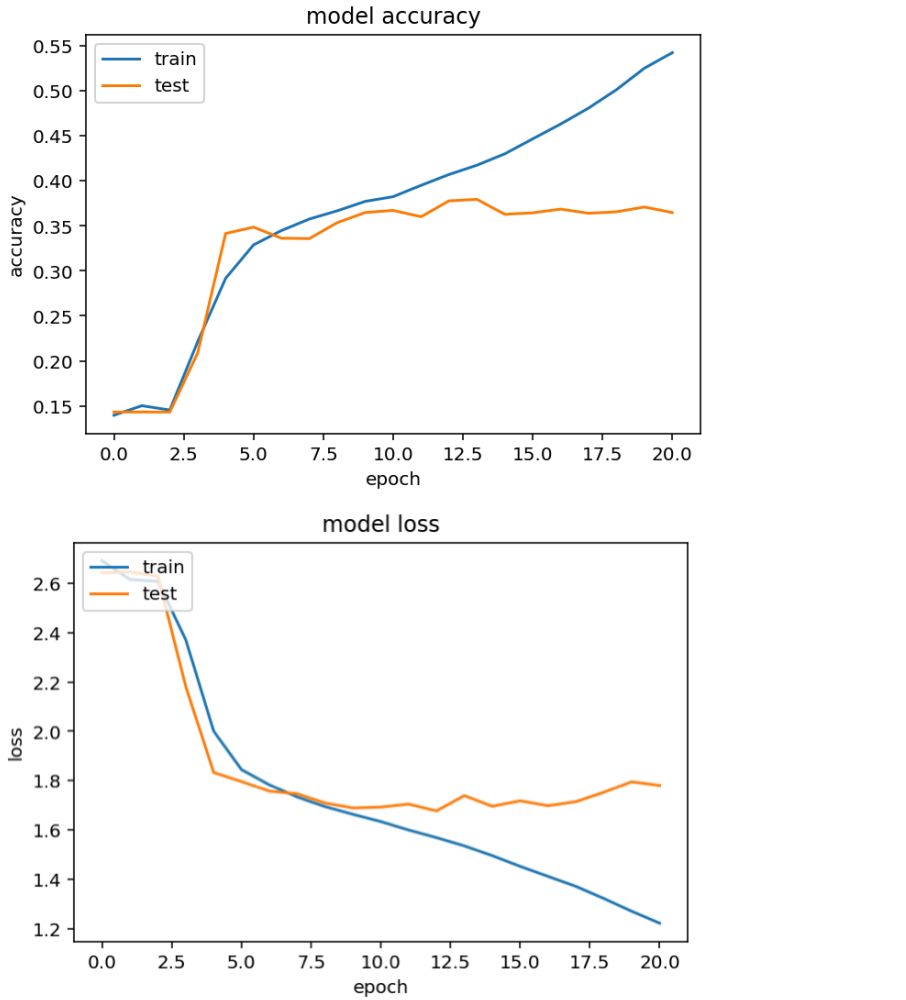
- 40 training values tracked over 5 different folds to see where overfitting happens and validation loss stops improving
- Once proper number of epochs are decided, then we train on 100 percent of the data



Note on model training

- Find where Accuracy stops increasing and Loss stops decreasing
- Over each trial we found the average overfitting point was at ~8 epochs

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```



Model Architecture

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 200, 256)          264192
_________________________________________________________________
dropout (Dropout)            (None, 200, 256)          0
_________________________________________________________________
lstm_1 (LSTM)                (None, 200, 512)          1574912
_________________________________________________________________
dropout_1 (Dropout)          (None, 200, 512)          0
_________________________________________________________________
lstm_2 (LSTM)                (None, 256)               787456
_________________________________________________________________
dense (Dense)                (None, 256)               65792
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 19)                4883
_________________________________________________________________
activation (Activation)      (None, 19)                0
=================================================================
Total params: 2,697,235
Trainable params: 2,697,235
Non-trainable params: 0
_________________________________________________________________
```

The simple LSTM-RNN model used consists of three LSTM layers with dropout 0.3 and 512, 256, and 256 nodes per layer respectively and two dense layers one with 256 nodes and the other with node count equal to the length of the output vector.

Model Evaluation: Char-LSTM

- Model Training over three independent trials