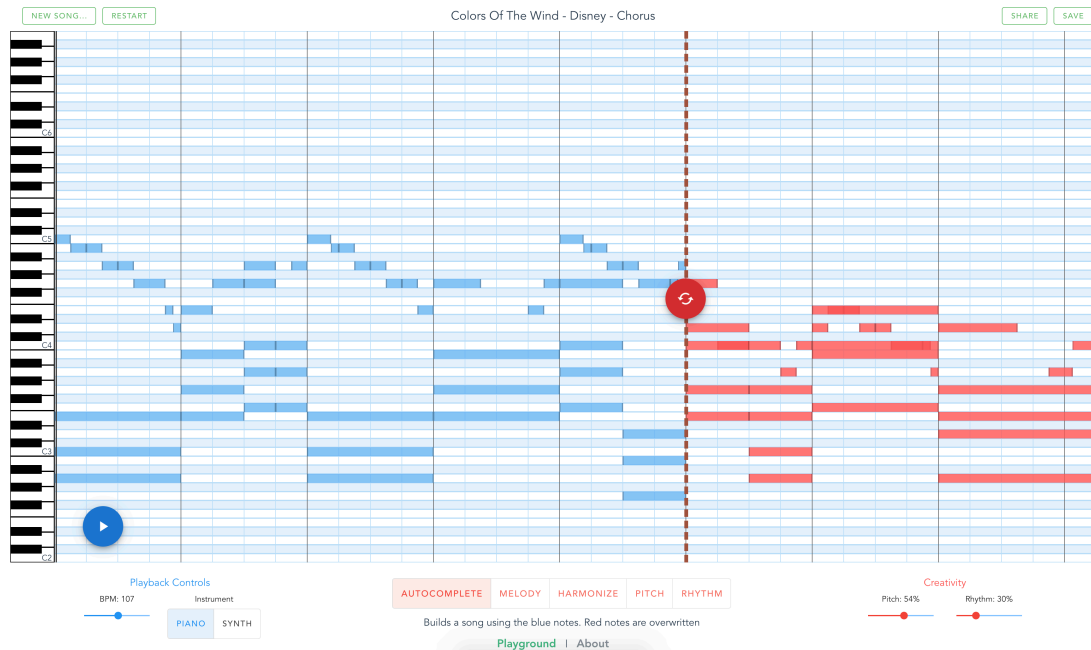


---

# MusicAutobot

Using Deep Learning to generate pop music!

You can also experiment through the web app - [musicautobot.com](https://musicautobot.com)



## Overview

Recent advances in NLP have produced amazing results in generating text. Transformer architecture is a big reason behind this.

This project aims to leverage these powerful language models and apply them to music. It's built on top of the fast.ai library

## Implementation

**MusicTransformer** - This basic model uses Transformer-XL to take a sequence of music notes and predict the next note.

**MultitaskTransformer** - Built on top of MusicTransformer, this model is trained on multiple tasks.  
\* Next Note Prediction (same as MusicTransformer) \* BERT Token Masking \* Sequence To Sequence Translation - Using chords to predict melody and vice versa.

---

Training on multiple tasks means we can generate some really cool predictions (Check out this Notebook): 1. Harmonization - generate accompanying chords 2. Melody - new melody from existing chord progression 3. Remix tune - new song in the rhythm of a reference song 4. Remix beat - same tune, different rhythm

## How it works

Details are explained in this 4 part series: \* Part I - Creating a Pop Music Generator \* Part II - Implementation details \* Part III - Multitask Transformer \* Part IV - Composing a song with Multitask

## Example Notebooks

1. Play with predictions on Google Colab
  - MusicTransformer Generate - Loads a pretrained model and shows how to generate/predict new notes
  - MultitaskTransformer Generate - Loads a pretrained model and shows how to harmonize, generate new melodies, and remix existing songs.
2. MusicTransformer
  - Train - End to end example on how to create a dataset from midi files and train a model from scratch
  - Generate - Loads a pretrained model and shows how to generate/predict new notes
3. MultitaskTransformer
  - Train - End to end example on creating a seq2seq and masked dataset for multitask training.
  - Generate - Loads a pretrained model and shows how to harmonize, generate new melodies, and remix existing songs.
4. Data Encoding
  - Midi2Tensor - Shows how the library internally encodes midi files to tensors for training.
  - MusicItem - MusicItem is a wrapper that makes it easy to manipulate midi data. Convert midi to tensor, apply data transformations, even play music or display the notes within browser.

## Pretrained Models

Pretrained models are available as MusicTransformer and MultitaskTransformer (small and large).

---

Each model has an additional **keyC** version. **keyC** means that the model has been trained solely on music transposed to the key of C (all white keys). These models produce better results, but expects the input to all be in the key of C.

1. MusicTransformer (600 MB) - AnyKey | KeyC
2. MultitaskTransformer
  - Small (700 MB) - AnyKey | KeyC
  - Large (2.1 GB) - AnyKey | KeyC

For details on how to load these models, follow the Generate and Multitask Generate notebooks

## Source Code

- musicautobot/
  - numpy\_encode.py - submodule for encoding midi to tensor
  - music\_transformer.py - Submodule structure similar to fastai's library.
    - Learner, Model, Transform - MusicItem, Dataloader
- multitask\_transformer.py - Submodule structure similar to fastai's library.
  - Learner, Model, Transform - MusicItem, Dataloader

## Scripts

CLI scripts for training models:

**run\_multitask.py** - multitask training

```
1 python run_multitask.py --epochs 14 --save multitask_model --batch_size=16 --bptt=512 --lamb --data_parallel --lr 1e-4
```

**run\_music\_transformer.py** - music model training

```
1 python run_music_transformer.py --epochs 14 --save music_model --batch_size=16 --bptt=512 --lr 1e-4
```

**run\_ddp.sh** - Helper method to train with multiple GPUs (DistributedDataParallel). Only works with run\_music\_transformer.py

```
1 SCRIPT=run_multitask.py bash run_ddp.sh --epochs 14 --save music_model --batch_size=16 --bptt=512 --lr 1e-4
```

Commands must be run inside the **scripts/** folder

---

## Installation

1. Install anaconda: <https://www.anaconda.com/distribution/>
2. Run:

```
1 git clone https://github.com/bearpelican/musicautobot.git
2
3 cd musicautobot
4
5 conda env update -f environment.yml
6
7 source activate musicautobot
```

3. Install Musescore - to view sheet music within a jupyter notebook

Ubuntu:

```
bash sudo apt-get install musescore
```

MacOS - download

## Flask Server

Installation:

```
1 cd serve
2
3 conda env update -f environment.yml
```

**S3 Bucket** You need to setup an s3 bucket to save your predictions. After you've created a bucket, update the config api/api.cfg with the new bucket name.

Development:

```
1 python run.py
```

Production:

```
1 gunicorn -b 127.0.0.1:5000 run_guni:app --timeout 180 --workers 8
```

## Data

Unfortunately I cannot provide the dataset used for training the model.

Here's some suggestions:

- 
- Classical Archives - incredible catalog of high quality classical midi
  - HookTheory - great data for sequence to sequence predictions. Need to manually copy files into hookpad
  - Reddit - 130k files
  - Lakh - great research dataset

## **Acknowledgements**

This project is built on top of fast.ai's deep learning library and music21's incredible musicology library.

Inspired by bachbot and clara

Special thanks to SPC and PalapaVC