goto;

GOTO
CHICAGO 2023

#GOTOchgo

Stretch

A Composer's Guide to Creating with Generative NNs

Molly Jones

Build smarter software.

We're on a mission to build impactful and high-quality software for exceptional clients.

SPAN TREE

TaPIR

SSHRC

Canada

My Practice

My Practice

Shannon’s Entropy

What you're going to see

in this presentation

- the technical side of creating a proof-of-concept composition
- an artist training a neural network
- data challenges
- infrastructure challenges
- ethical quandaries

What you're not

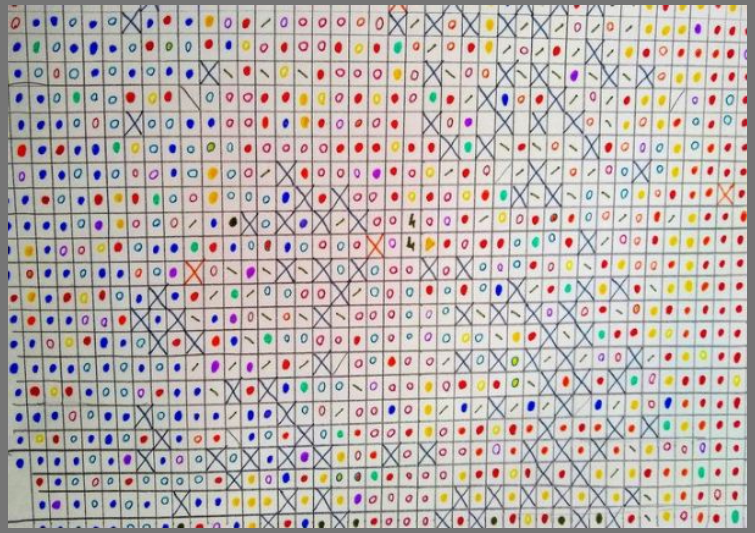
going to see

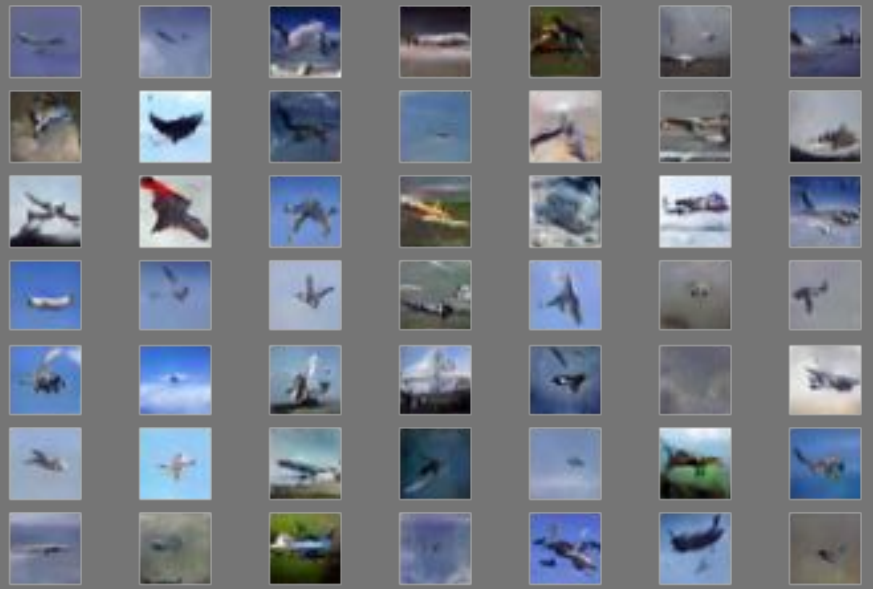
- pre-trained, proprietary models
- prompt engineering
- nonconsensual data use
- beautiful code

Outline

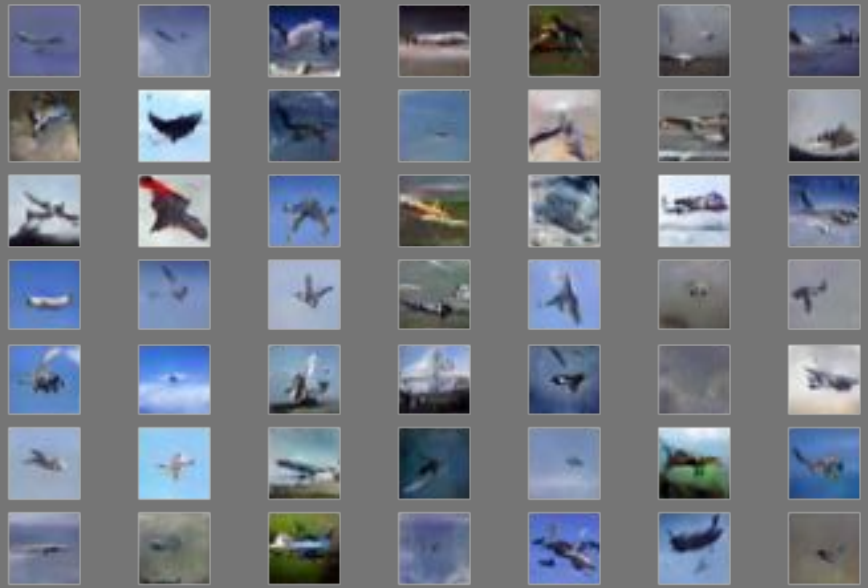
1. Background and Goals
2. Data Collection
3. Model Selection
4. Infrastructure
5. Development
6. Deciding When It’s Done
7. Composition
8. Future Directions

1. Background + Goals






Following
MothBot
@bot_moth
Follows you
:: a bot by @MollyEJones using Tracery (@GalaxyKate)
:: angsty moth poetry
github.com/v21/tracerybot
Joined October 2021
4 Following
14 Followers
Followed by (Asian)Steve, Milwaukee Ave Bike Lane, and 3 others you follow
Tweets
Tweets & replies
Media
Likes
MothBot @bot_moth · Oct 31, 2021
Below the unfathomable cosmos, I swim
MothBot @bot_moth · Oct 31, 2021
Floating, flying
silently above the ether
MothBot @bot_moth · Oct 31, 2021
My antenna reaches past the vibrationless plasma
1



no 1

no 2

no 3

approximations

a piece for accordion, percussion, and neural networks

approximations - May 19, 2023

WOLFGANG AMADEUS MOZART

Musikalisches Würfelspiel

Table of Measure Numbers

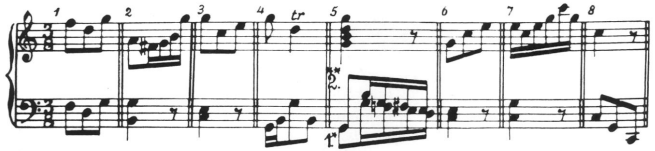
Part One

	I	II	III	IV	V	VI	VII	VIII
2	96	22	141	41	105	122	11	30
3	32	6	128	63	146	46	134	81
4	69	95	158	13	153	55	110	24
5	40	17	113	85	161	2	159	100
6	148	74	163	45	80	97	36	107
7	104	157	27	167	154	68	118	91
8	152	60	171	53	99	133	21	127
9	119	84	114	50	140	86	169	94
10	98	142	42	156	75	129	62	123
11	3	87	165	61	135	47	147	33
12	54	130	10	103	28	37	106	5

Part Two

	I	II	III	IV	V	VI	VII	VIII
2	70	121	26	9	112	49	109	14
3	117	39	126	56	174	18	116	83
4	66	139	15	132	73	58	145	79
5	90	176	7	34	67	160	52	170
6	25	143	64	125	76	136	1	93
7	138	71	150	29	101	162	23	151
8	16	155	57	175	43	168	89	172
9	120	88	48	166	51	115	72	111
10	65	77	19	82	137	38	149	8
11	102	4	31	164	144	59	173	78
12	35	20	108	92	12	124	44	131

Table of Measures

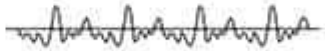


TIMBRE

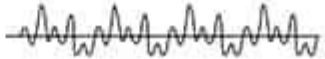
Tuning fork

Flute

Voice



Guitar



Two Stories of Confusion

Incorrect Assumption #1:
artists != engineers

Incorrect Assumption #2: proprietary models are better

2. Data Collection

Ethical Quandary:
data provenance

Louis Pino + Matti Pulkki

Challenge: data volume

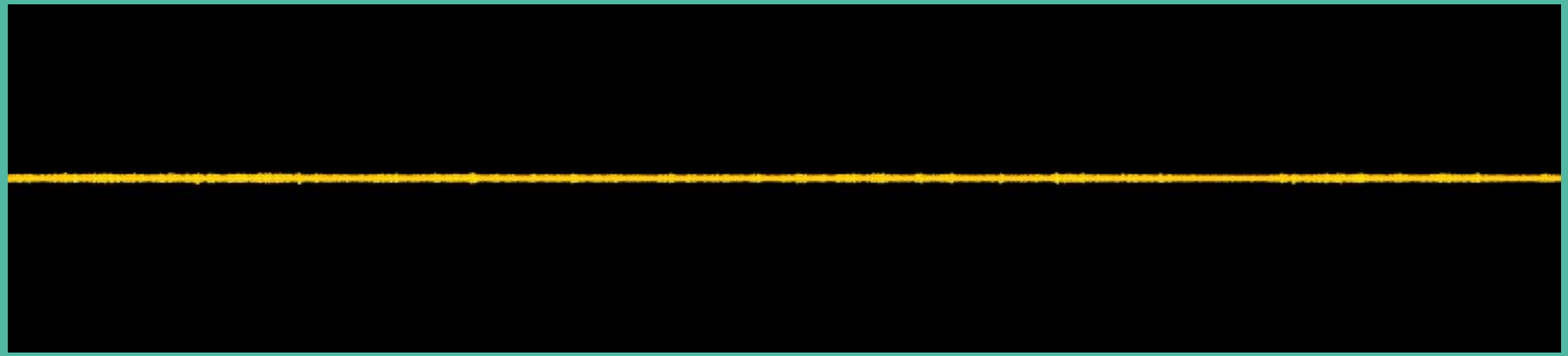
Training Data

- Percussion

- Accordion

Challenge: multi-modal data

Challenge: network too clever



3. Model Selection

"I started with a literature search. I pulled papers on adversarial audio synthesis and a variety of techniques for generation, including some published since January 2021 when I first started playing around with audio generative networks. I did a deep read of the Wavenet paper, and then I realized I may as well start with the structures I already know since I'm unlikely to have time to do a full review of the entire field."

-journal, 10/20/2022

- WaveNet
- DanceDiffusion
- GANSynth
- WaveGAN
- MelNet
- AudioSinGAN
- SampleRNN
- StrawNet (variant of WaveNet)
- RAVE
- Catch A Waveform
- DDSP

- WaveNet (2016)
- DanceDiffusion (2022)

Challenge: lack of time

Challenge: technical debt

2.1 DILATED CAUSAL CONVOLUTIONS

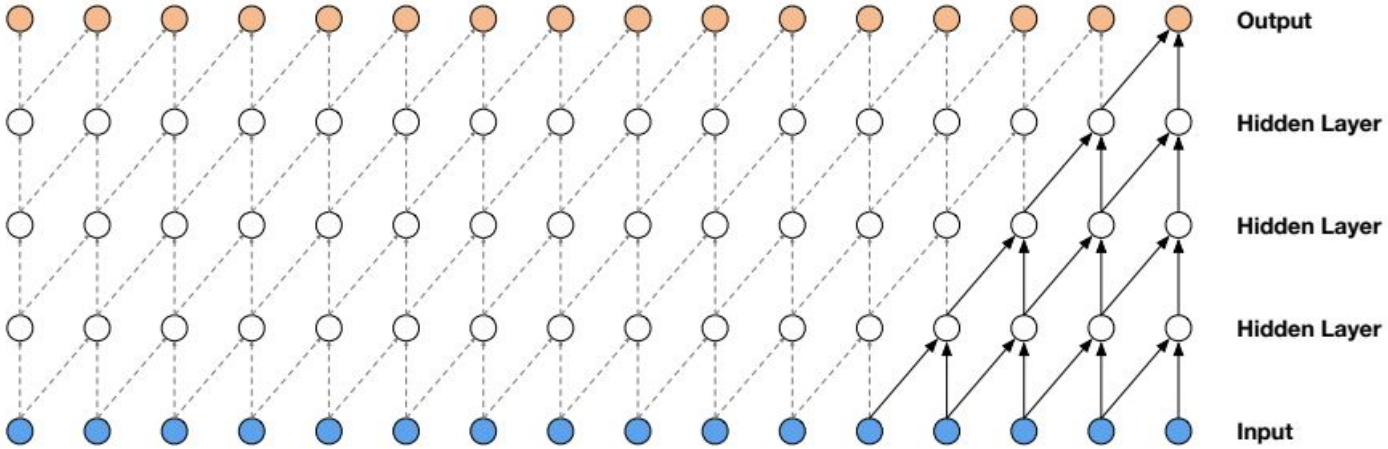
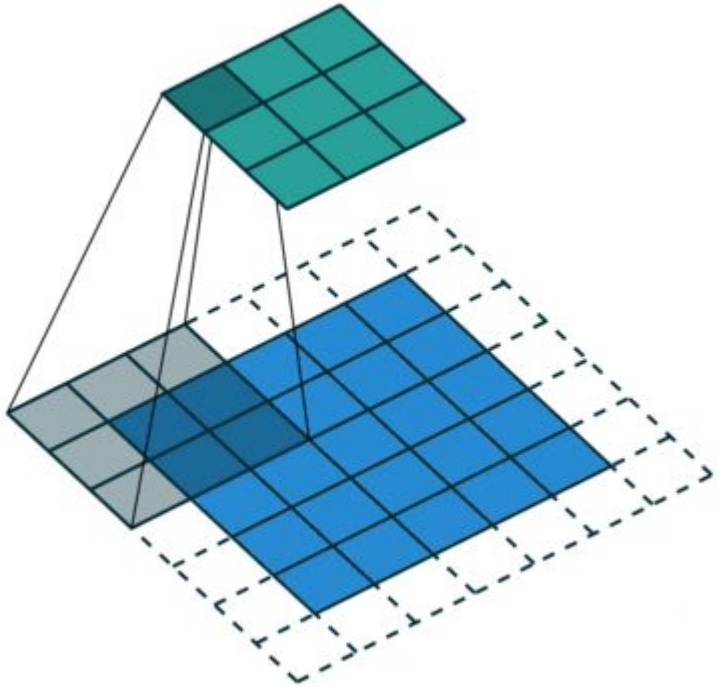
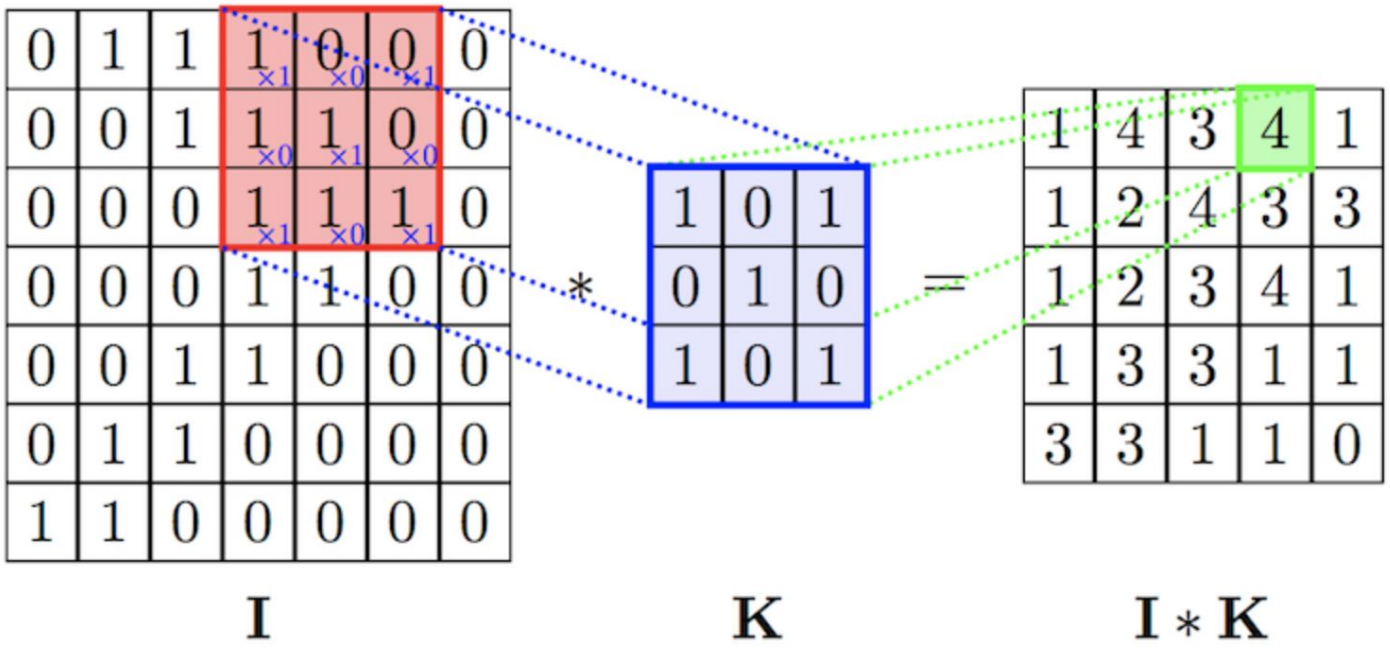


Figure 2: Visualization of a stack of causal convolutional layers.

Wavenet Architecture

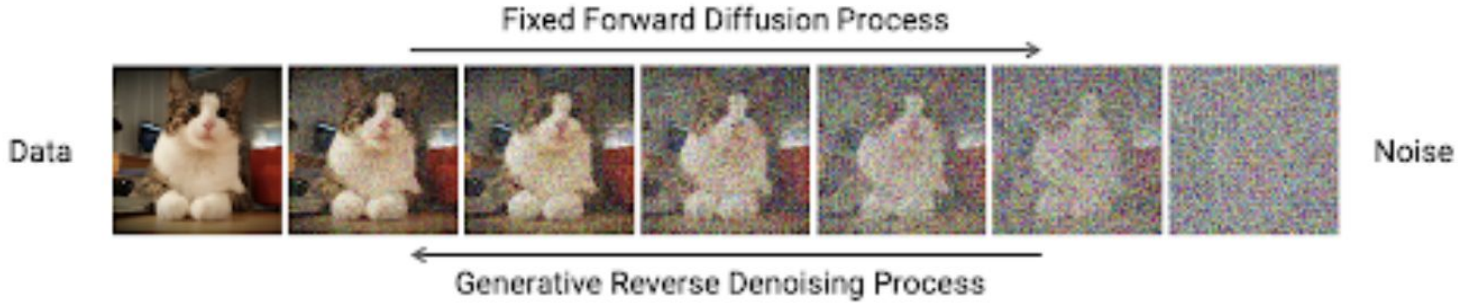


Wavenet Architecture



Wavenet Architecture

DanceDiffusion

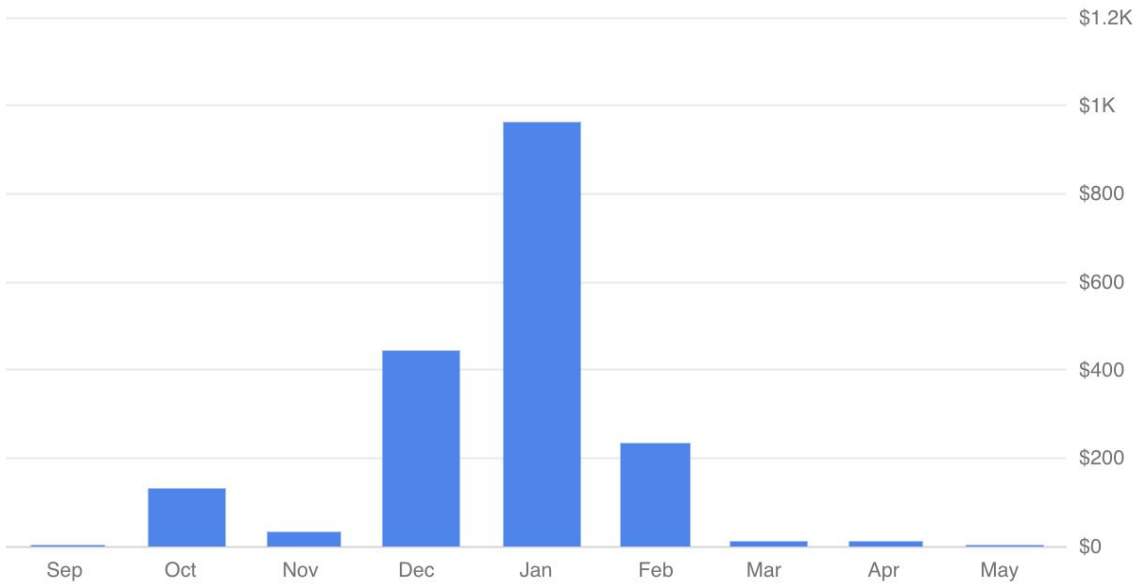


4. Infrastructure

Cloud VMs

- old converted TF + GPUs = hard
- Linux OOM killer = frustrating
- budget limitations = real
- cost structure = stressful + inexact

Cloud VMs



Challenge: MLOps

5. Development

B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Model	Performer	Data Augmentation	Audio length	Sample rate	Training Steps	Filter width	Initial filter width	Learning rate	Sample Size	Silence Threshold	L2 Strength	Momentum	Loss Behavior	Next Steps
Wavenet (Ibab)	Pino	Trimmed silence, double compressed (Audacity), time shifted, pitch shifted	4.5 hrs	combo 44100 and 22050; set as 22050 in wavenet_params	100,000	2	32	1.00E-04	100,000	0.1	0	0.9	vascillates anywhere from 0.2 up to 4.8	resample the one file to 22050 // triple the amount of audio in there through more augmentation // change sample_size to be one second of audio (22050 samples)
Wavenet (Ibab)	Pino	added Oct resampled to 22050 plus time/pitch shifted versions of Oct	8.25 hrs	22050	100,000	2	32	1.00E-04	22050	0.1	0	0.9	drops to 1.4-3.4 and then vascillates	Will try dropping the learning rate; if this doesn't help, may try increasing the number of training steps OR may try processing Matti's audio and seeing what difference that makes
Wavenet (Ibab)	Pino	added Oct resampled to 22050 plus time/pitch shifted versions of Oct	8.25 hrs	22050	100,000	2	32	1.00E-05	22050	0.1	0	0.9	vascillates more widely between 0.9 and 5	Don't think that will get better with higher # of training steps; will try increasing the sample size
Wavenet (Ibab)	Pino	added Oct resampled to 22050 plus time/pitch shifted versions of Oct	8.25 hrs	22050	100,000	2	32	1.00E-04	250000	0.1	0	0.9	N/A	ran out of memory almost immediately
Wavenet (Ibab)	Pino	added Oct resampled to 22050 plus time/pitch shifted versions of Oct	8.25 hrs	22050	100,000	2	32	1.00E-07	100000	0.1	0	0.9	vascillates between 3 and 5.5	
Wavenet (Ibab)	Matti	silence-removed, plus time and pitch shifted versions	8.25 hrs	22050	100,000	2	32	1.00E-04	100000	0.1	0	0.9	Ugh! It seems to have been converging consistently, loss was down to consistently around 2.0-2.1, and then it was killed for some reason. And when I restarted it with the logdir to continue training, loss went back to vacillating among 3 and 4.4	

Wavenet Architecture

Tunable Hyperparameters

- filter width = 2
- learning rate = 0.0001
- sample size / analysis window size = 50,000
- silence threshold = 0.1
- dilation channels = 32
- skip channels = 1024
- dilation layer structure…

Tunable Hyperparameters


```
{
    "filter_width": 2,
    "sample_rate": 16000,
    "dilations": [1, 2, 4, 8, 16, 32, 64, 128, 256, 512,
                  1, 2, 4, 8, 16, 32, 64, 128, 256, 512,
                  1, 2, 4, 8, 16, 32, 64, 128, 256, 512,
                  1, 2, 4, 8, 16, 32, 64, 128, 256, 512,
                  1, 2, 4, 8, 16, 32, 64, 128, 256, 512,
                  1, 2, 4, 8, 16, 32, 64, 128, 256, 512,
                  1, 2, 4, 8, 16, 32, 64, 128, 256, 512,
                  1, 2, 4, 8, 16, 32, 64, 128, 256, 512],
    "residual_channels": 32,
    "dilation_channels": 32,
    "quantization_channels": 256,
    "skip_channels": 1024,
    "use_biases": true,
    "scalar_input": false,
    "initial_filter_width": 32
}
```

Challenge: iteration speed

Challenge: cost

$635.00

Infrastructure Budget

$1,831.65

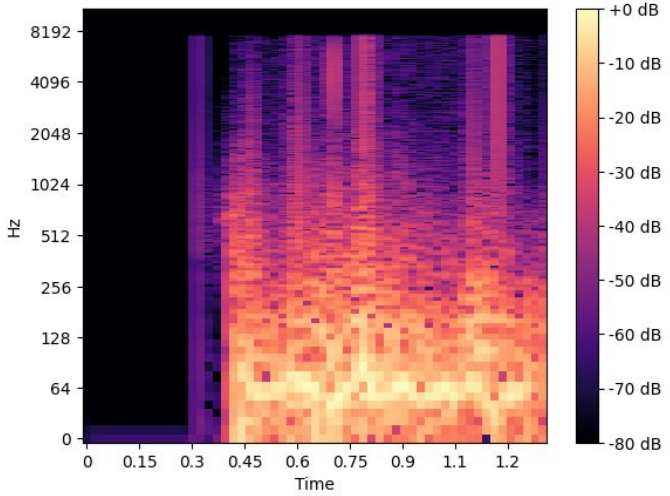
Infrastructure Costs*

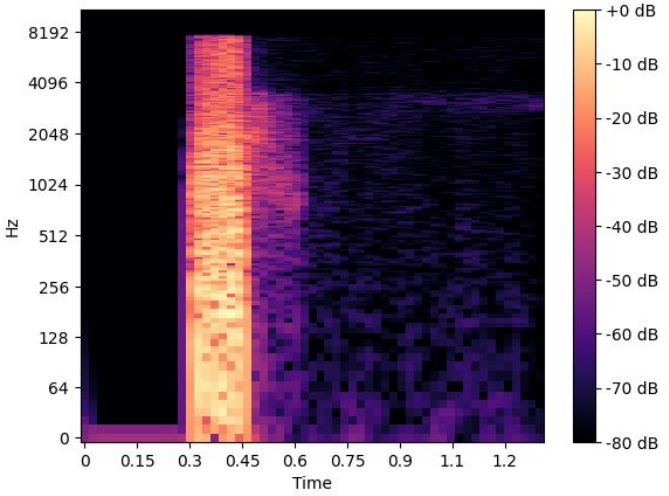
*plus ~$22 for coffee with knowledgeable ppl

Developer / Artist Hours

6. When is it done?

STFT (short time Fourier transform)





Accordion Sounds

7450 training steps

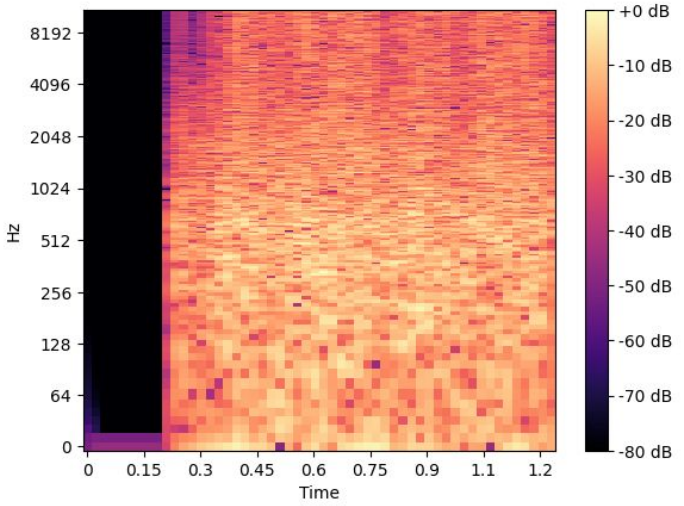
54,850 training steps

92,150 training steps

149,300 training steps

345,000 training steps

Original



Accordion Sounds

7450 training steps

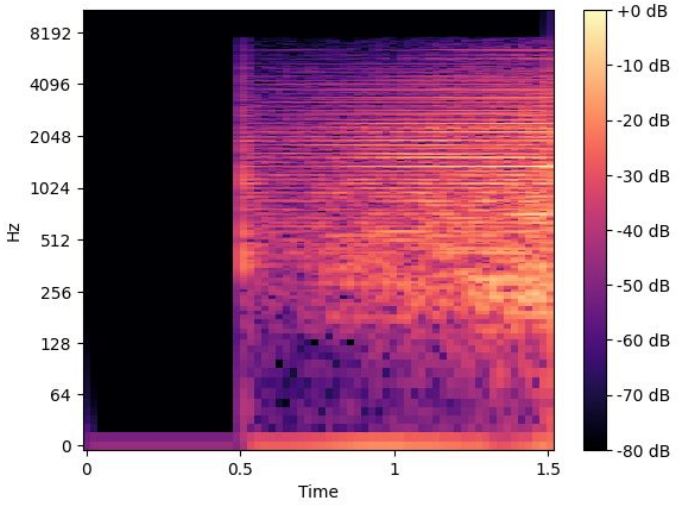
54,850 training steps

92,150 training steps

149,300 training steps

345,000 training steps

Original



Accordion Sounds

7450 training steps

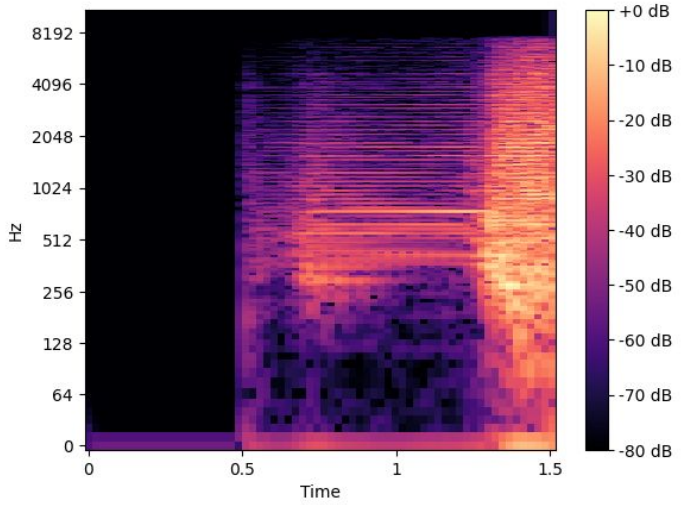
54,850 training steps

92,150 training steps

149,300 training steps

345,000 training steps

Original



Percussion Sounds

19,500 training steps

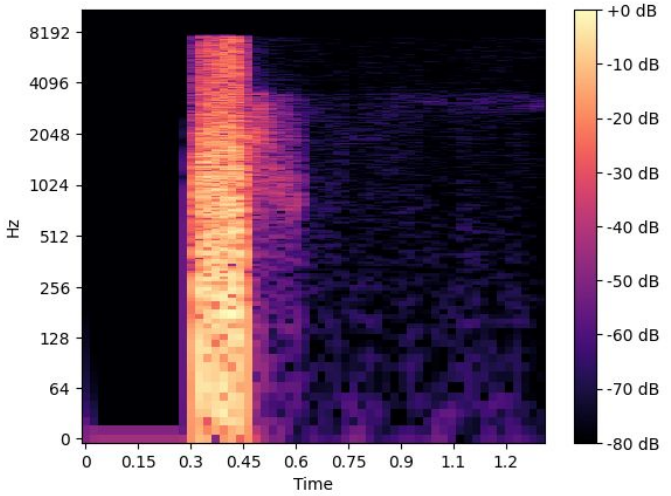
63,000 training steps

99,999 training steps

275,150 training steps

299,999 training steps

Original



Percussion Sounds

19,500 training steps

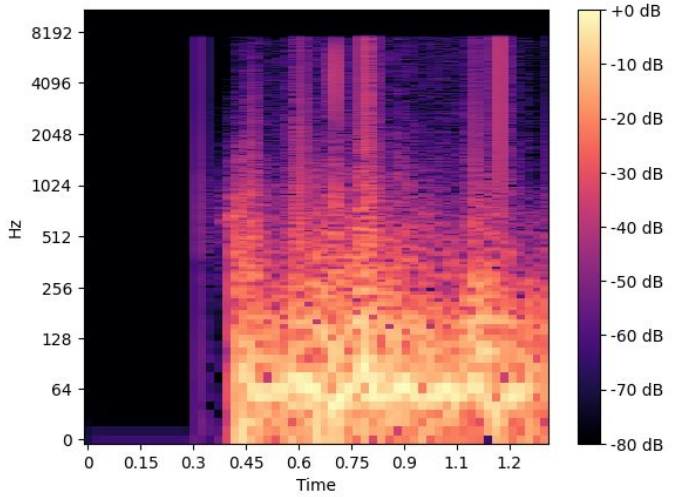
63,000 training steps

99,999 training steps

275,150 training steps

299,999 training steps

Original



Percussion Sounds

19,500 training steps

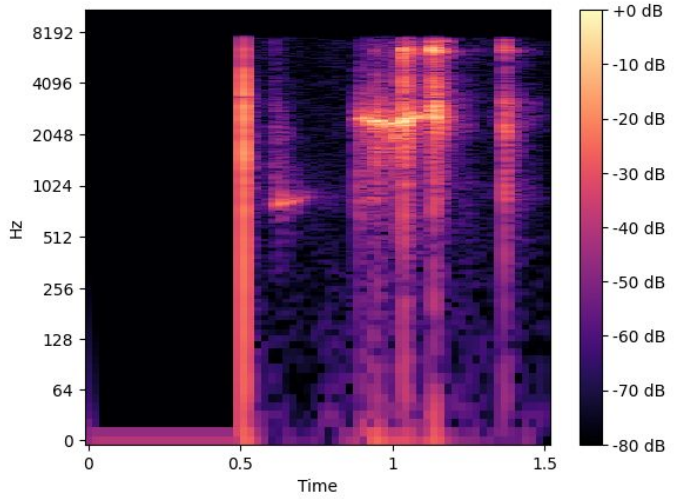
63,000 training steps

99,999 training steps

275,150 training steps

299,999 training steps

Original



~1 month

Training Time

7. Composition

```python
import os      Molly Jones, 2 months ago • add short pulse …
import numpy as np
import soundfile as sf
from common.load_audio import load_audio_file
from common.grains import make_one_grain, make_one_repitched_grain, repeat_grain


def grain_hold(
    audio_dir, filename, tempo, length, write_out=False, output_file_name=None
):
    audio_data, sr, filelength = load_audio_file(audio_dir, filename)

    original_audio_data = audio_data.tolist()
    pulse_audio = []

    grain_length = (
        round(1 / tempo * sr) if filelength > round(1 / tempo * sr) else filelength
    )
    print(grain_length)
    grain_to_repeat = make_one_grain(grain_length, original_audio_data, sr)

    while len(pulse_audio) < length * sr:
        pulse_audio.extend(grain_to_repeat)

    if write_out:
        audio_output_path = os.path.join("generated", "held_grain", output_file_name)
        sf.write(audio_output_path, pulse_audio, sr, subtype="PCM_24")
        print("Saved held grain audio file to /pulse!")

    return pulse_audio, sr


def short_pulse(
    audio_dir, filename, tempo, length, write_out=False, output_file_name=None
):
    audio_data, sr, filelength = load_audio_file(audio_dir, filename)
```

```python
def accordion_salad(
    audio_dir, length=10, max_length=1, write_out=False, output_file_name=None
):
    num_audio_files, buffer_objs = load_audio_dir(audio_dir)
    initial_sample_rate = buffer_objs[0]["sample_rate"]

    salad_left = []
    salad_right = []

    while len(salad_left) < length * initial_sample_rate:
        selected_file = np.random.randint(num_audio_files)
        audio_data = buffer_objs[selected_file]["data"]
        sr = buffer_objs[selected_file]["sample_rate"]
        minimum_grain_length = (
            sr * 0.5 if len(audio_data) > sr * 0.5 else len(audio_data) * 0.5
        )

        grain_length_left = (
            np.random.randint(minimum_grain_length, sr * 1.5 * max_length)
            if len(audio_data) > sr * 1.5 * max_length
            else np.random.randint(minimum_grain_length, len(audio_data))
        )
        grain_length_right = (
            np.random.randint(minimum_grain_length, sr * 1.5 * max_length)
            if len(audio_data) > sr * 1.5 * max_length
            else np.random.randint(minimum_grain_length, len(audio_data))
        )

        # silence random grains
        grain_to_silence = np.random.choice([0, 1, 2], 1, p=[0.9, 0.05, 0.05])
        if grain_to_silence == 0:
            grain1 = make_one_repitched_grain(grain_length_left, audio_data, sr)
            grain2 = make_one_repitched_grain(grain_length_right, audio_data, sr)
        elif grain_to_silence == 1:
            grain1 = [0.0] * grain_length_left
            grain2 = make_one_repitched_grain(grain_length_right, audio_data, sr)
        else:
            grain1 = make_one_repitched_grain(grain_length_left, audio_data, sr)
            grain2 = [0.0] * grain_length_right

        # extract harmonic components of some of the grains
        if np.random.choice([0, 1], 1, p=[0.85, 0.15]) == 0:
```

Ethical Quandary: composer / performer relationship

Molly Jones

approximations

for Accordion, Vibraphone,
Bass Drum, + Fixed Media

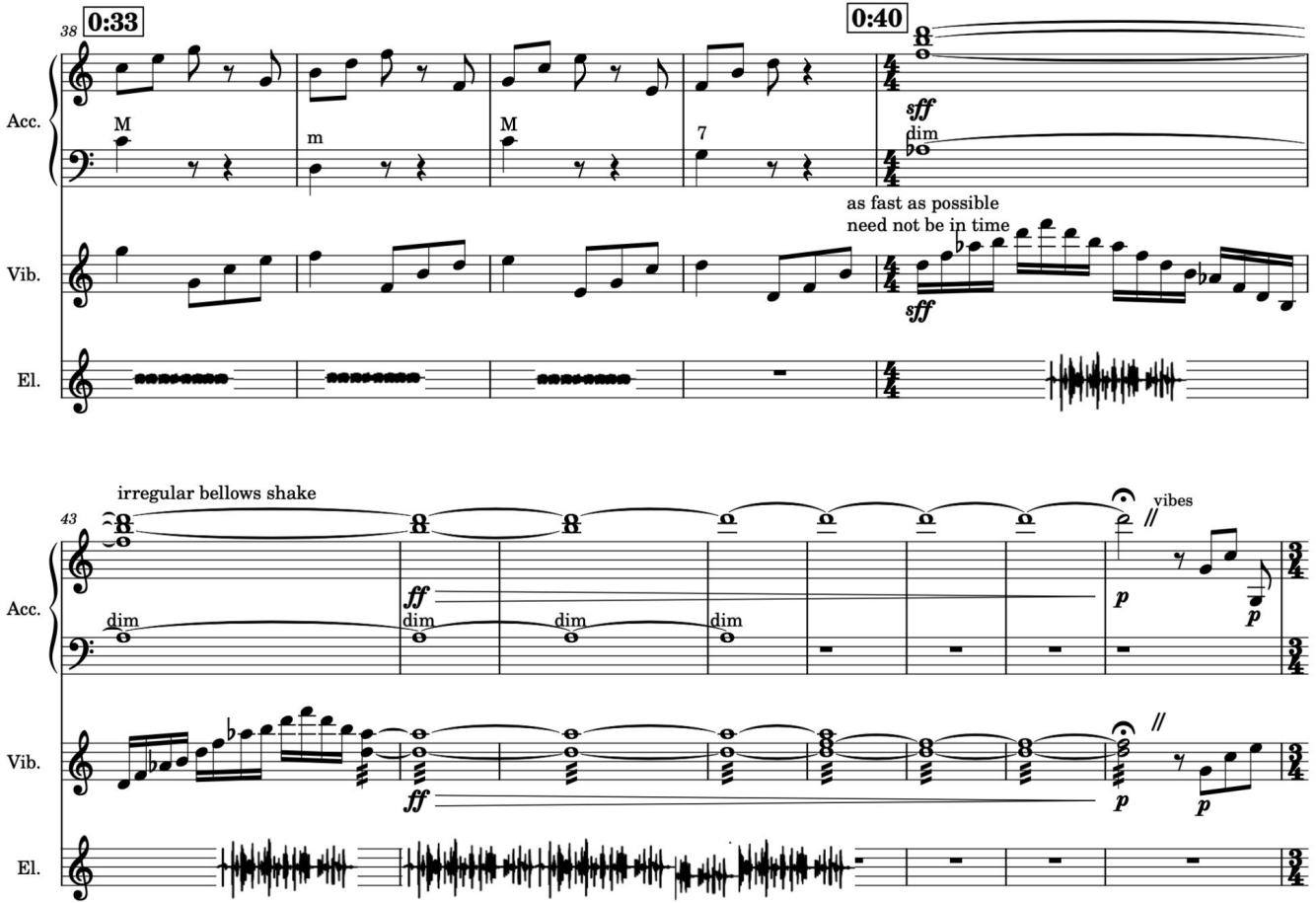
11:38

April 2023

mov't iv

mov't i

mov't i - from the fog of randomness	3:22
mov't ii - making friends with the robots	1:55
mov't iii - call + response	1:14
mov't iv - mutual listening	3:00
mov't v - educated guessing	1:05


38
0:33
0:40
Acc.
Vib.
El.
M
m
M
7
sff
dim
as fast as possible
need not be in time
sff
irregular bellows shake
43
dim
ff
dim
dim
dim
vibes
p
p
ff
ff
p
p

approximations - May 19, 2023

8. Future Directions

Next Steps

- other network structures

Next Steps

- other network structures
- TensorFlow -> PyTorch ?

Next Steps

- other network structures
- TensorFlow -> PyTorch ?
- audio data augmentation

Next Steps

- other network structures
- TensorFlow -> PyTorch ?
- audio data augmentation
- buy a GPU!

Next Steps

- Chicago Creative Machines
- Synthetic Prairie

Challenges:

- data
  - volume
  - quality

Challenges:

- data
  - volume
  - quality
- time
  - skill (years)
  - development (hours)
  - iteration speed

Challenges:

- data
  - volume
  - quality
- time
  - skill (years)
  - development (hours)
  - iteration speed
- cost

Ethical Quandary:
artist / AI symbiosis

Creative Exercise

Gratitude

- Igor Babuschkin
- Zach Evans and Harmonai

- Ethan Manilow
- Alice Li
- Mansi Shah

Questions? Comments?

mojones.e@gmail.com

goto;

Don't forget to
vote for this session
in the **GOTO Guide app**

Appendix: Vocabulary

stochastic

involving chance,
randomness, or probability

neural network

definition from IBM

timbre

the quality or tone distinctive of a particular singing voice or musical instrument

(from Merriam-Webster)

hyperparameter

a human-changeable parameter affecting the structure and training of a neural network

as opposed to parameters, which are internal to a neural network and much more numerous

CNN

- convolutional neural network
- a neural network that moves a filter across multi-dimensional input data to detect patterns and structures in it
- commonly used in computer vision
- StatQuest video

RNN

- recurrent neural network
- a neural network that contains cyclical node references/feedback loops, allowing it to remember previous time steps
- commonly used for time series data and language processing
- StatQuest video

LSTM

- long short-term memory
- a fancy type of RNN that has a longer memory and also gates that control its ability to forget
- often used for natural language processing (NLP)
- StatQuest video

Algorithmic Composition

“"Supposing, for instance, that the fundamental relations of pitched sound in the signs of harmony and of musical composition were susceptible of such expression and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent"

-Ada Lovelace, 1843

WOLFGANG AMADEUS MOZART

Musikalisches Würfelspiel

Table of Measure Numbers

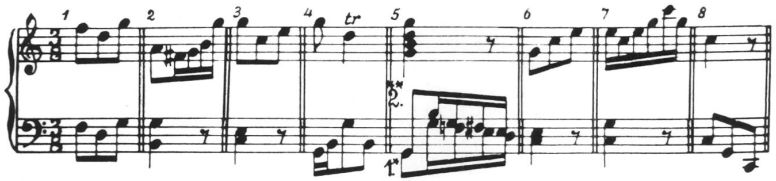
Part One

	I	II	III	IV	V	VI	VII	VIII
2	96	22	141	41	105	122	11	30
3	32	6	128	63	146	46	134	81
4	69	95	158	13	153	55	110	24
5	40	17	113	85	161	2	159	100
6	148	74	163	45	80	97	36	107
7	104	157	27	167	154	68	118	91
8	152	60	171	53	99	133	21	127
9	119	84	114	50	140	86	169	94
10	98	142	42	156	75	129	62	123
11	3	87	165	61	135	47	147	33
12	54	130	10	103	28	37	106	5

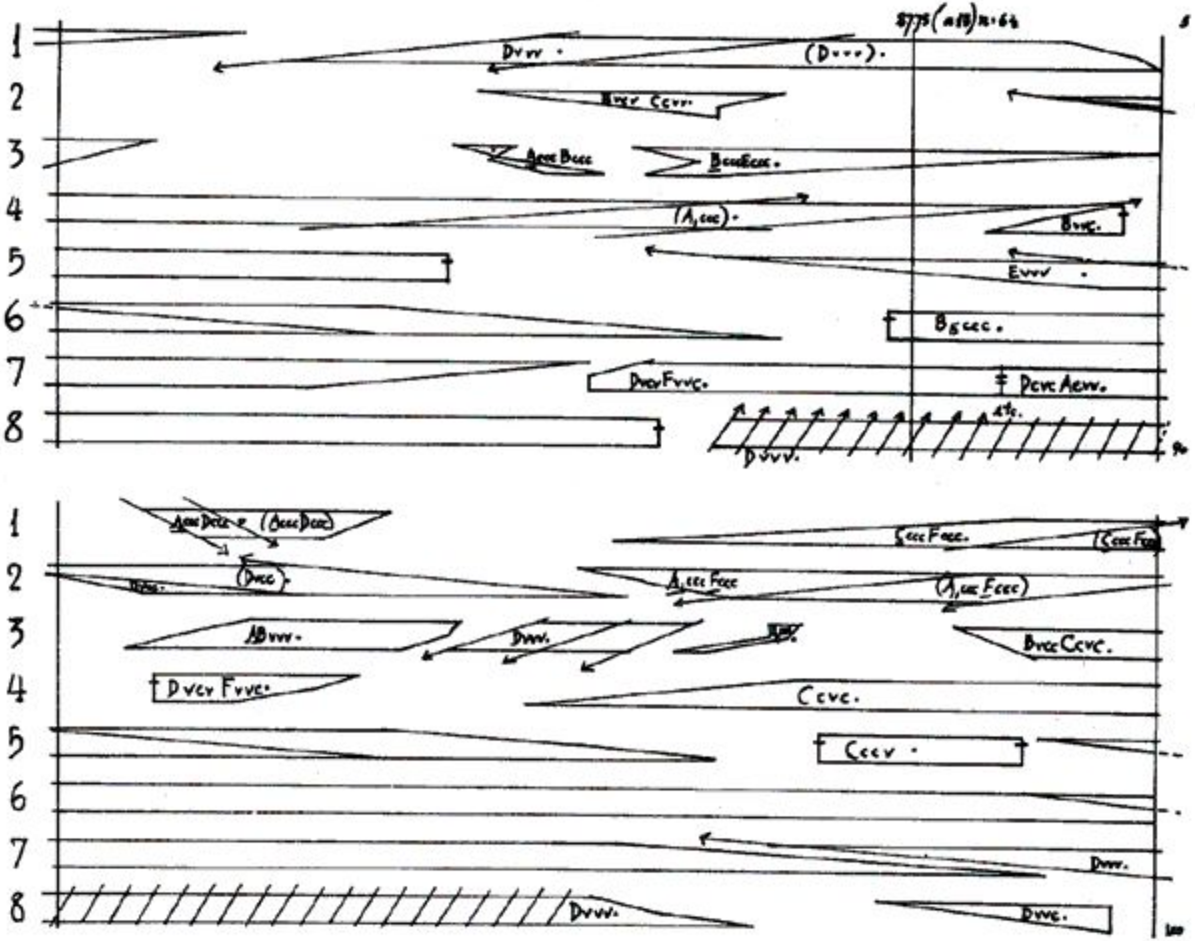
Part Two

	I	II	III	IV	V	VI	VII	VIII
2	70	121	26	9	112	49	109	14
3	117	39	126	56	174	18	116	83
4	66	139	15	132	73	58	145	79
5	90	176	7	34	67	160	52	170
6	25	143	64	125	76	136	1	93
7	138	71	150	29	101	162	23	151
8	16	155	57	175	43	168	89	172
9	120	88	48	166	51	115	72	111
10	65	77	19	82	137	38	149	8
11	102	4	31	164	144	59	173	78
12	35	20	108	92	12	124	44	131

Table of Measures



[MIDI generator]


1
2
3
4
5
6
7
8
Dvvv ·
(Dvvv).
Bgccc.
Dvcv Fvvc.
Cevc.
Cccv ·
BvccCcvc.
Dvvv.

Letter	Frequency (Hz)	Letter	Frequency (Hz)
A	440	M	660
B	458.33	N	678.33
C	476.66	O	696.66
D	495	P	715
E	513.33	Q	733.33
F	531.66	R	751.66
G	550	S	770
H	568.33	T	788.33
I/Y	586.66	U	806.66
J	605	V/W	825
K	623.33	X	843.33
L	641.66	Y/Z	861.66

Table 1: Padberg's Letter to Frequency Mapping

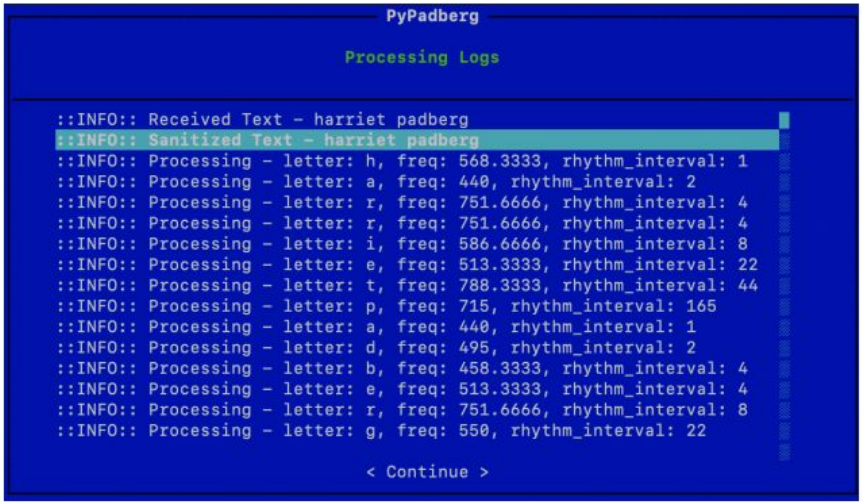


Figure 1: The processing screen in the PyPadberg Application, displaying how characters are mapped to frequencies and rhythms.

3. PYPADBERG

ML Music Generation

AI Audio Generators for Music

- AI Jukebox (OpenAI), 2020
- MusicLM (Google), text-to-music, 2023
- Style or timbre transfer
- Harmony generators