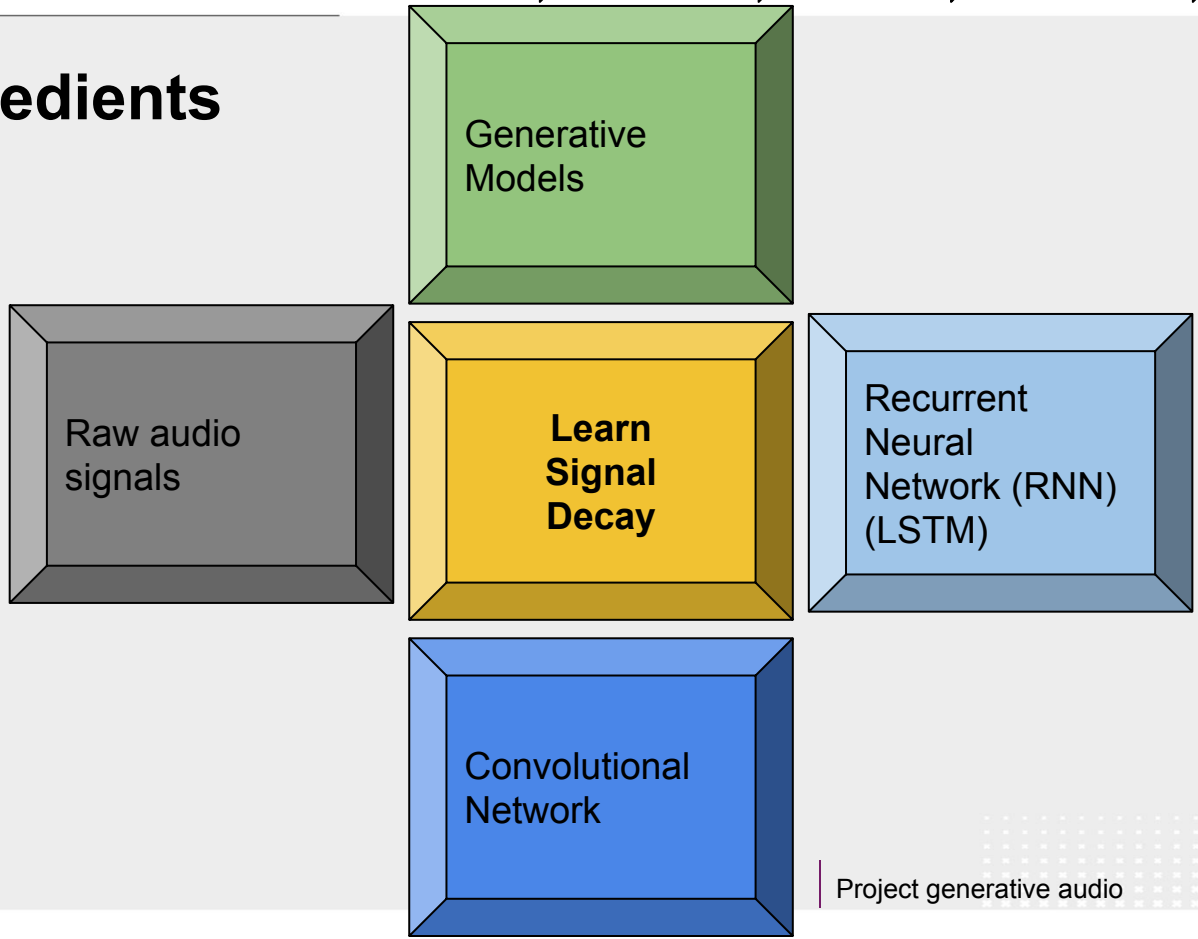
University of Amsterdam

Presentation of project *generative audio* by Frederik Harder & Jörg Sander

Generative Audio

Learning the decay of a sound signal

Ingredients



A motivation for generative models

Goal: endow computers with an understanding of the world

How?

train a **generative model** we first collect a large amount of data in some domain (e.g., think millions of images, sentences, or sounds, etc.) and then train a model to generate data like it

"What I cannot create, I do not understand."

—Richard Feynman

Project generative audio

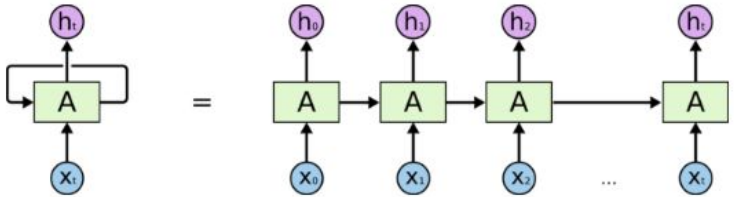
taken from https://www.openai.com/blog/generative-models/#going-forward

Recurrent Neural Network (RNN)

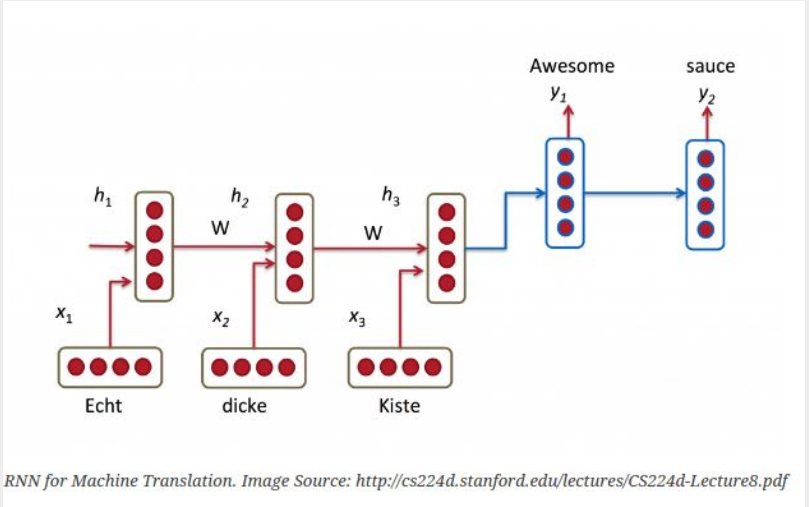
typical application is to let an RNN model a probability distribution over sequences

$$p(\mathbf{x}_{1:T}) = \prod_{t=0}^{T-1} p(x_{t+1}|\mathbf{x}_{1:t}),$$

Recurrent Neural Network (RNN)



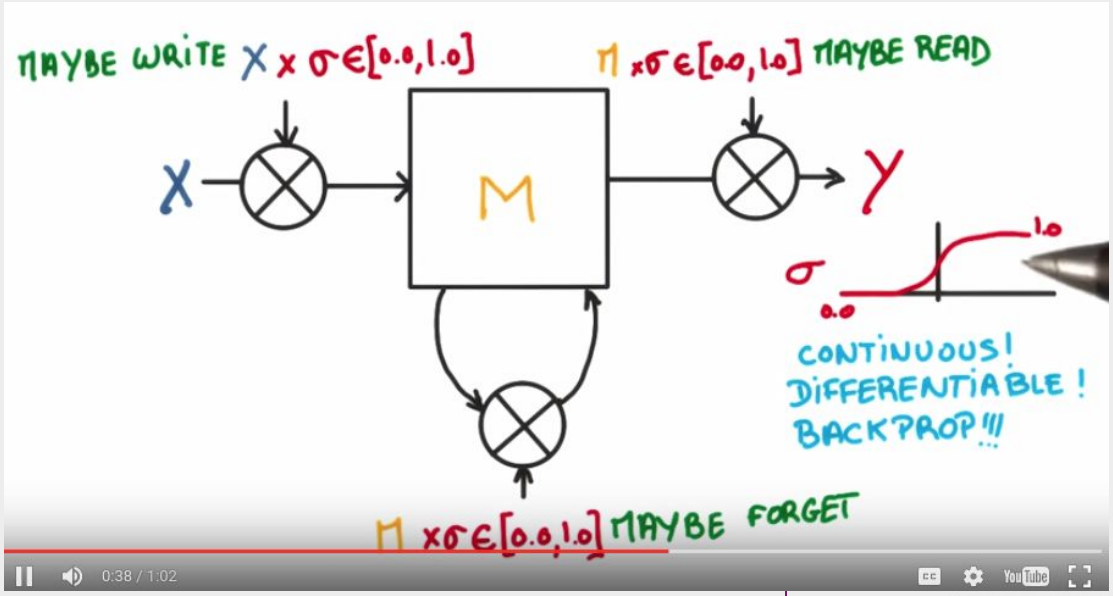
An unrolled recurrent neural network.



RNN for Machine Translation. Image Source: http://cs224d.stanford.edu/lectures/CS224d-Lecture8.pdf

Long Short Term Memory (LSTM) units

- Special kind of RNN, capable of learning long-term dependencies
- Also solve *vanishing gradient* problem



Related work

Eck & Schmidhuber (2002)

LSTM

Generate 12-bar blues chord

One input/target unit per note representation

Franklin (2006)

LSTM

Learn to generate Jazz music

Uses MIDI representation

Boulanger-Lewandowski et al. (2012)

RNN with RBM (RNN-RBM)

model unconstrained polyphonic music in the piano-roll representation (basically MIDI)

Bayer & Osendorfer (2015)

RNN with SGVB (STORN)

Generating polyphonic music

Uses MIDI representation

Karpathy (2015)

THE UNREASONABLE EFFECTIVENESS OF RNNs

char RNN

Nayebi & Vitelli (2015)

LSTM/GRU

Generating music based on David Bowie and Madeon songs

Using raw audio signals and DFT representation

Chung & Kastner et al. (2016)

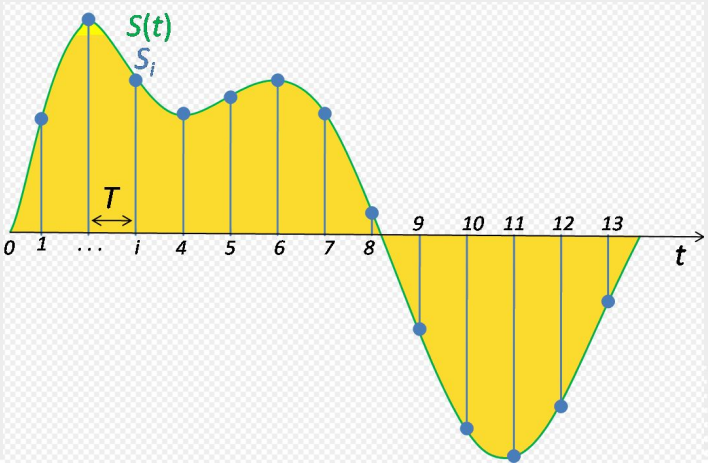
VRNN

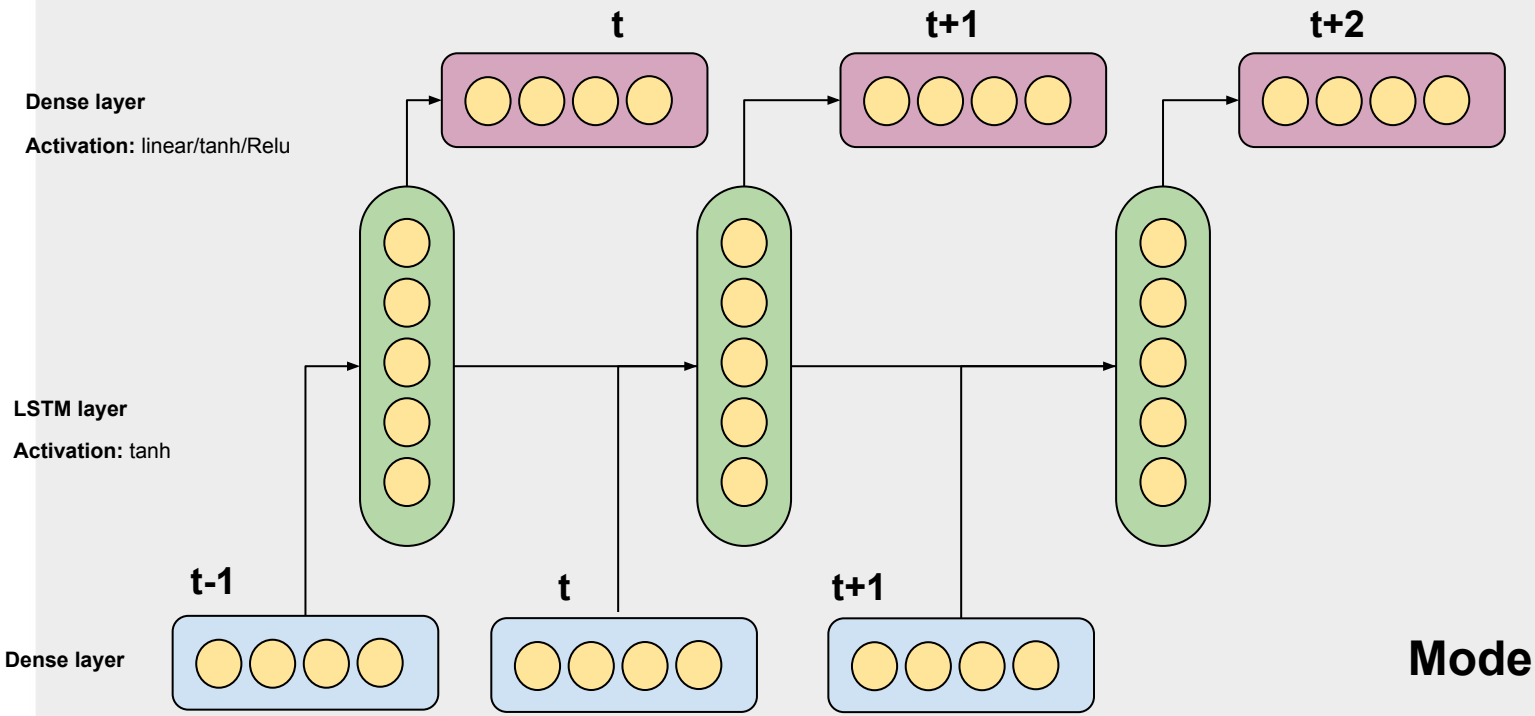
Speech modelling

Handwriting generation

Main difference between related work and our research

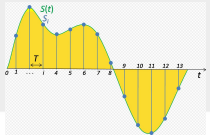
- Previous work used low dimensional sound representations such as midi files
- we attempt to eliminate this limit and learn directly on raw audio files thus giving the generative models the chance to access the entire frequency spectrum



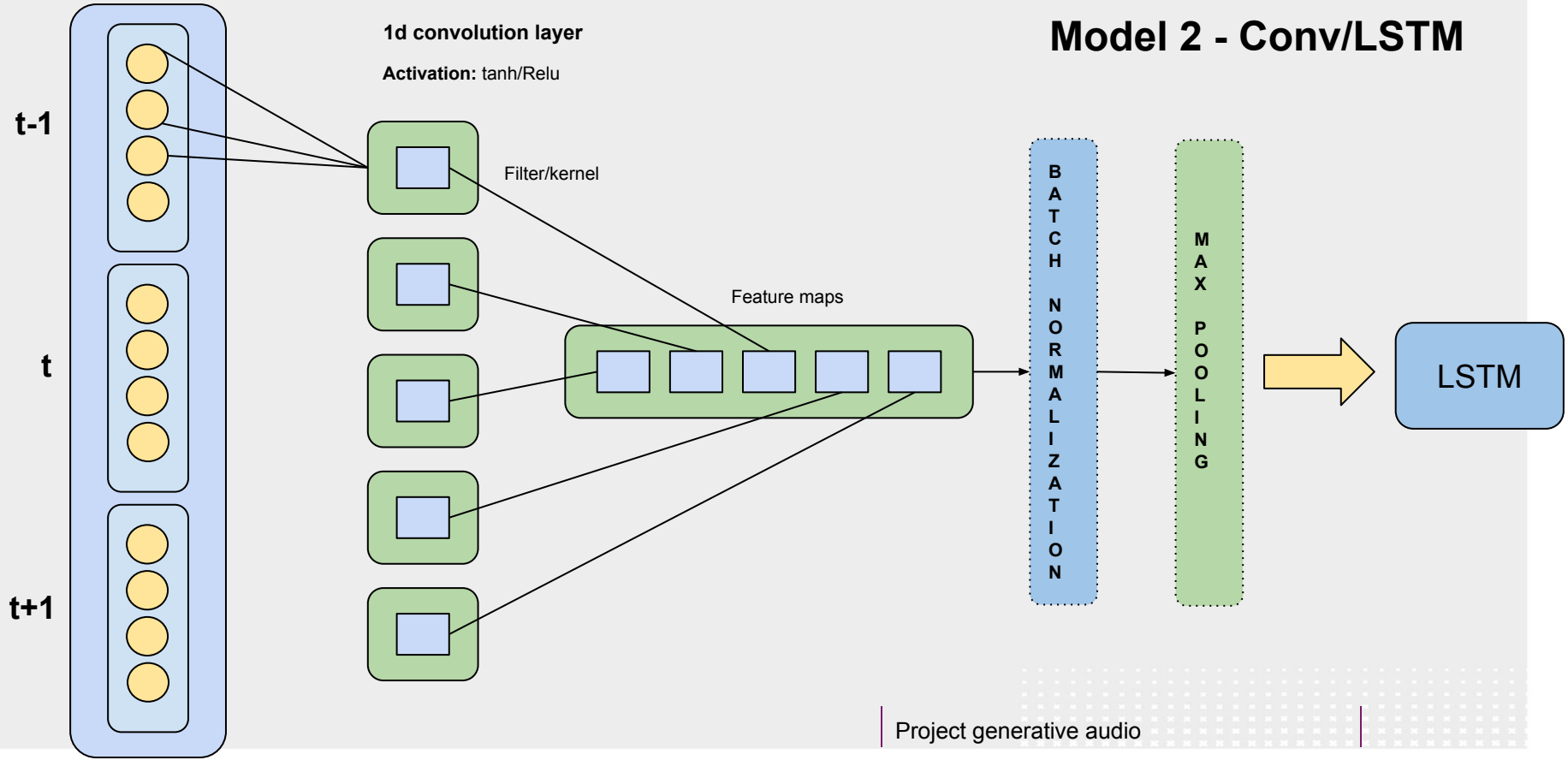


Model 1

Input signal



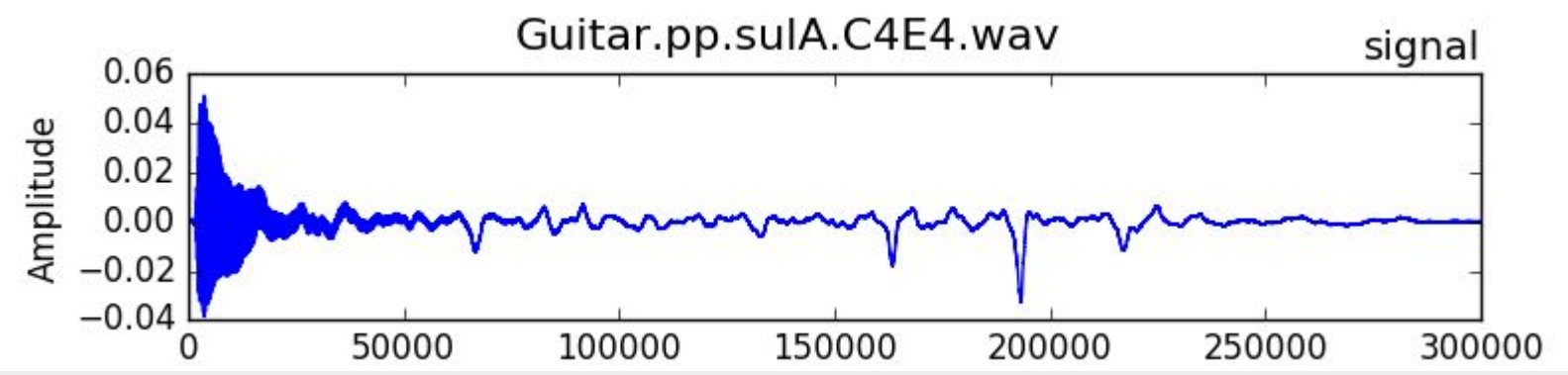
Model 2 - Conv/LSTM

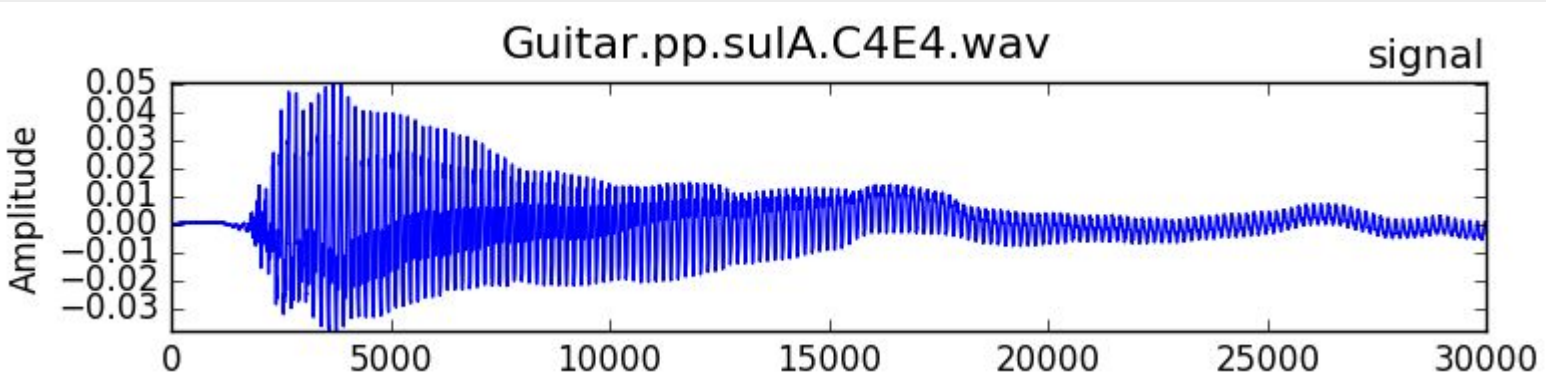


Audio Data

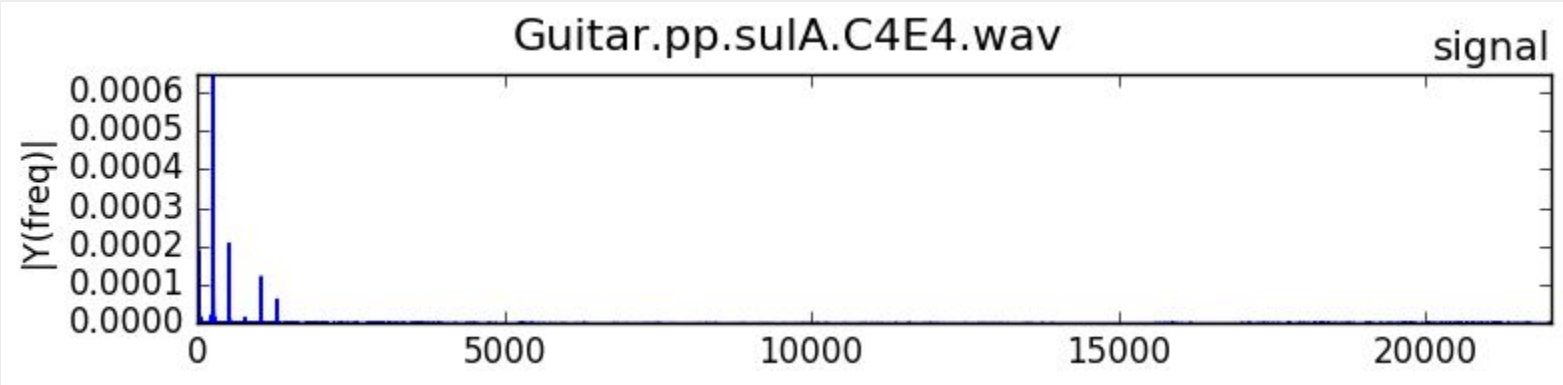
WAV format:

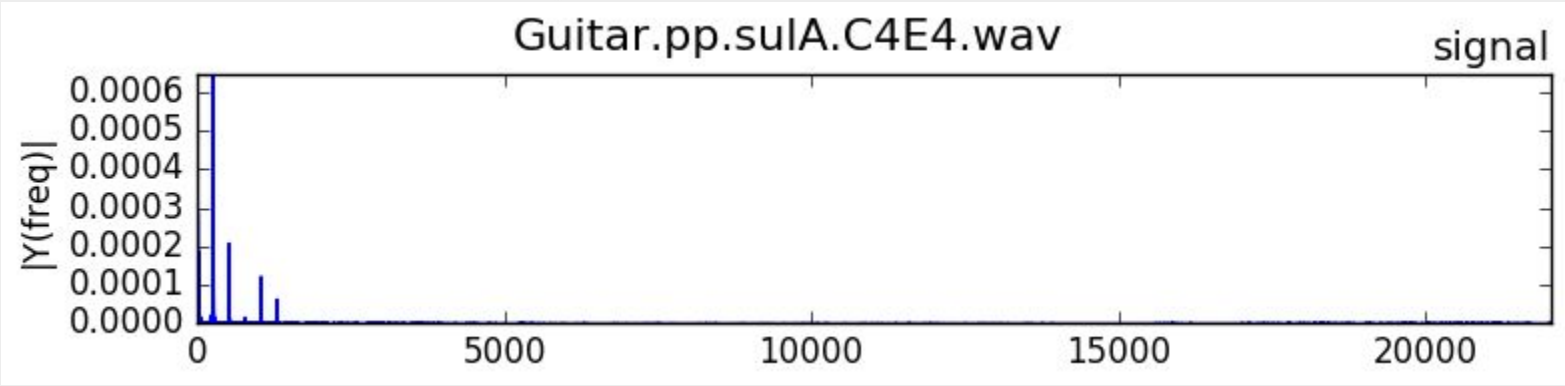
- Vector of int values
- Sample rate of 44100 values per second
  - Captures frequencies up to 22.05 kHz



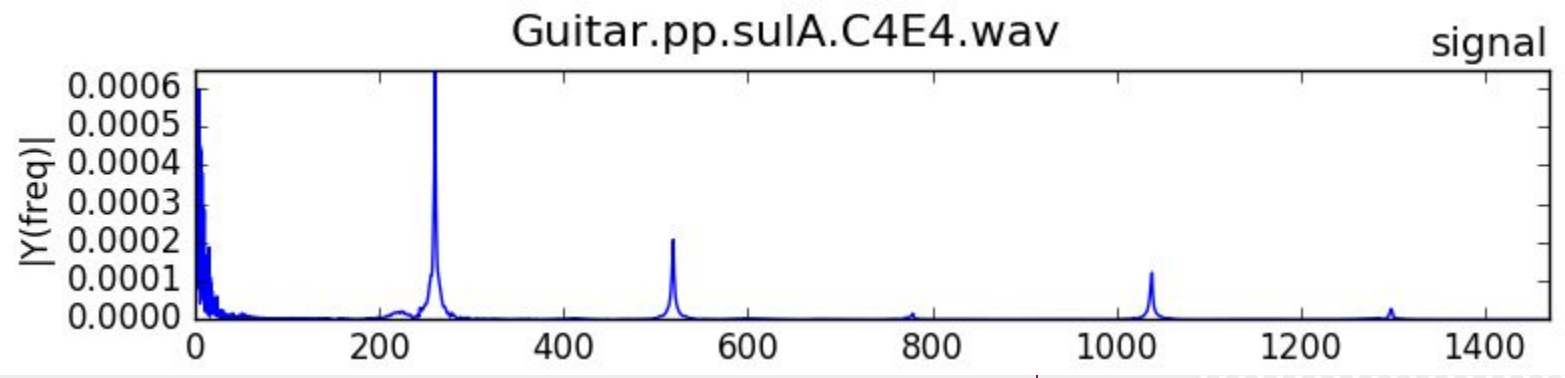


With a fourier transform, Signals can be decomposed into Frequencies





Reducing sample rate compresses the signal but loses high frequency information
(Nyquist theorem: signals can encode frequencies up to half the sampling rate)



Preprocessing and data representation

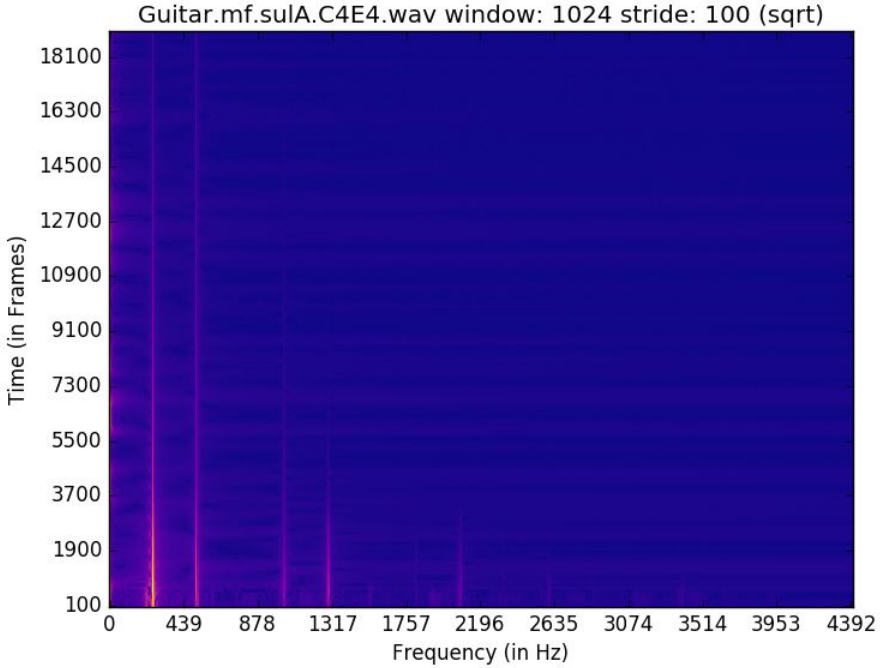
- All data is rescaled and normalized
- In most cases we downsampled the data to sample rate of
  - ~8000 for piano and cello
  - ~2800 for guitar
- Each signal is then separated into small time slices which serve as input to the RNN
- Optionally, each time slice is fourier transformed and the samples are created by composing the two representations

Data-Sets

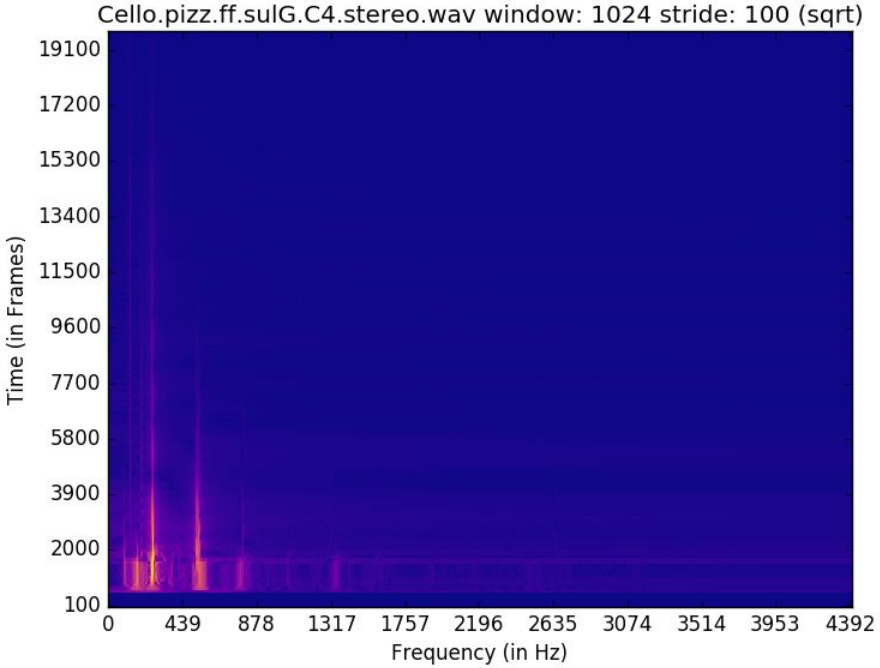
Studio recorded samples of individual notes in scales from

- Flute (39 x 2 second samples)
- Guitar (45 x ~5 sec)
- Cello (Bow and plucking) (100 x ~3 sec, 25 on each string)
- Piano (88 x ~10 sec)

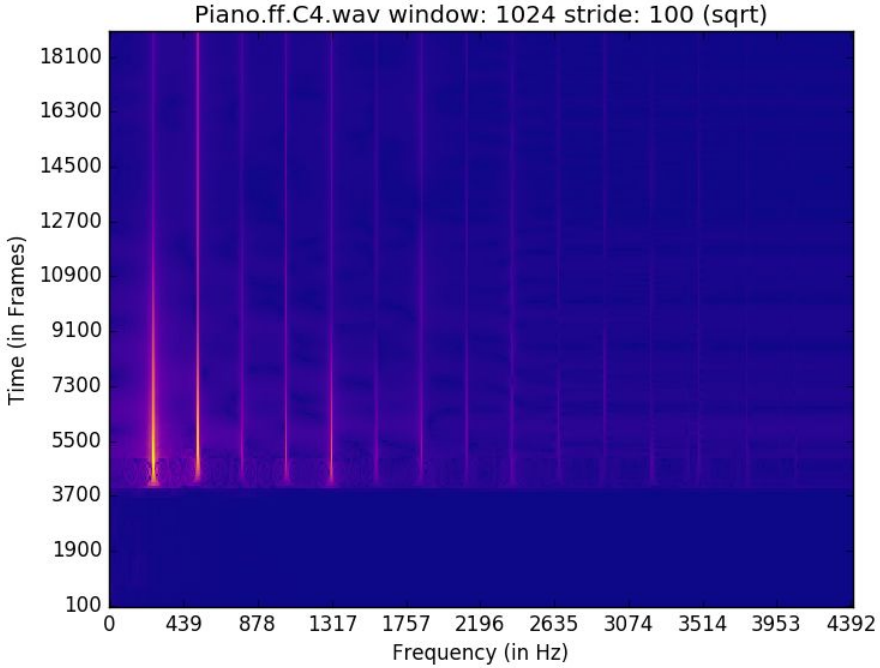
Guitar is more homogeneous than the others, containing 26 C's out of 45 samples.



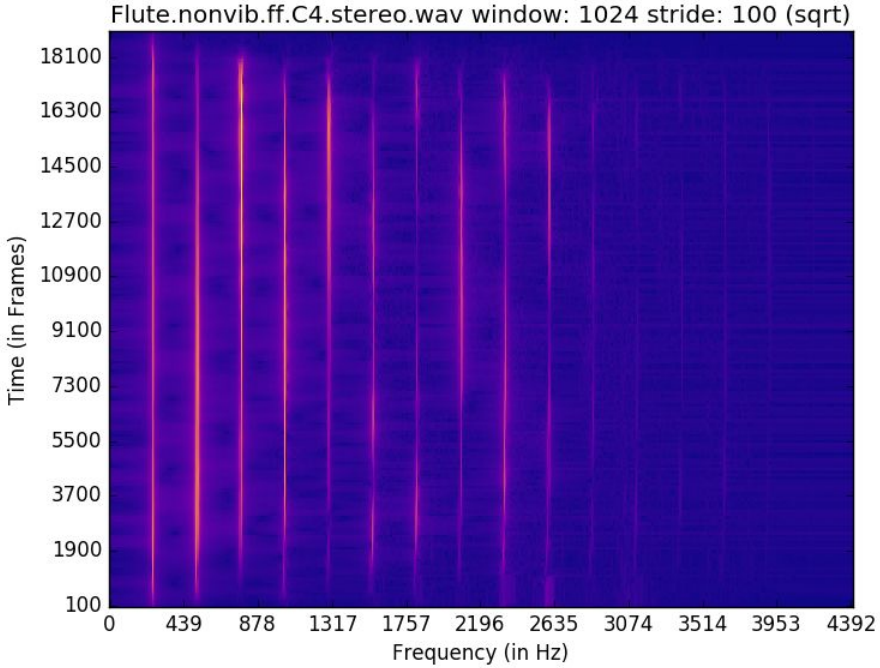
Guitar Spectrogram over time



Cello Spectrogram over time

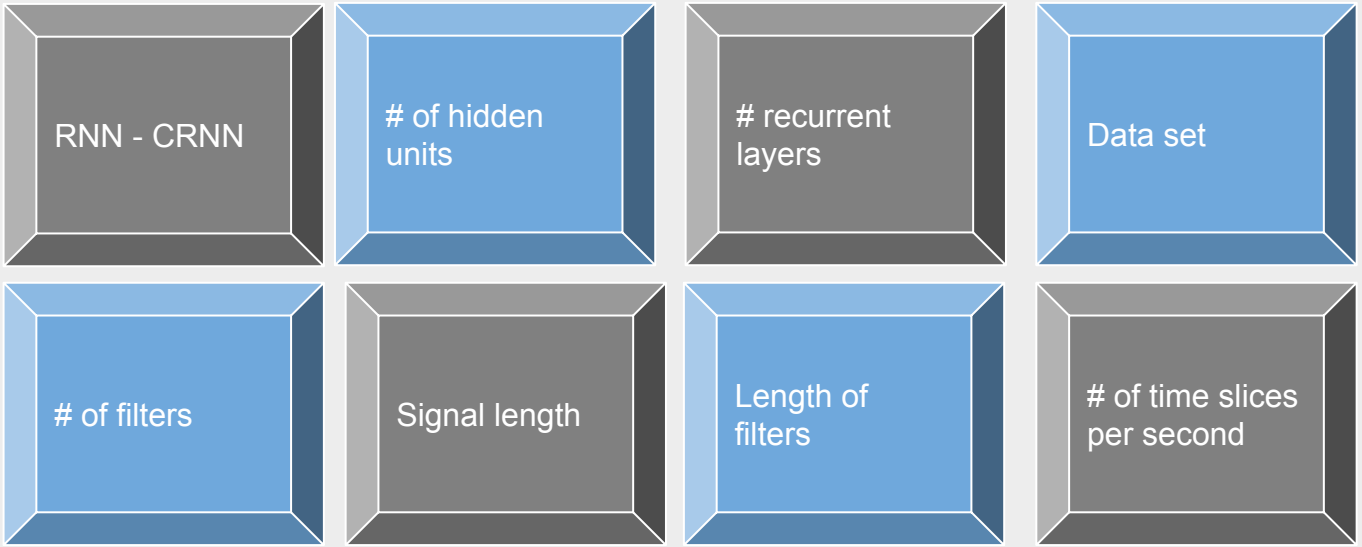


Piano Spectrogram over time

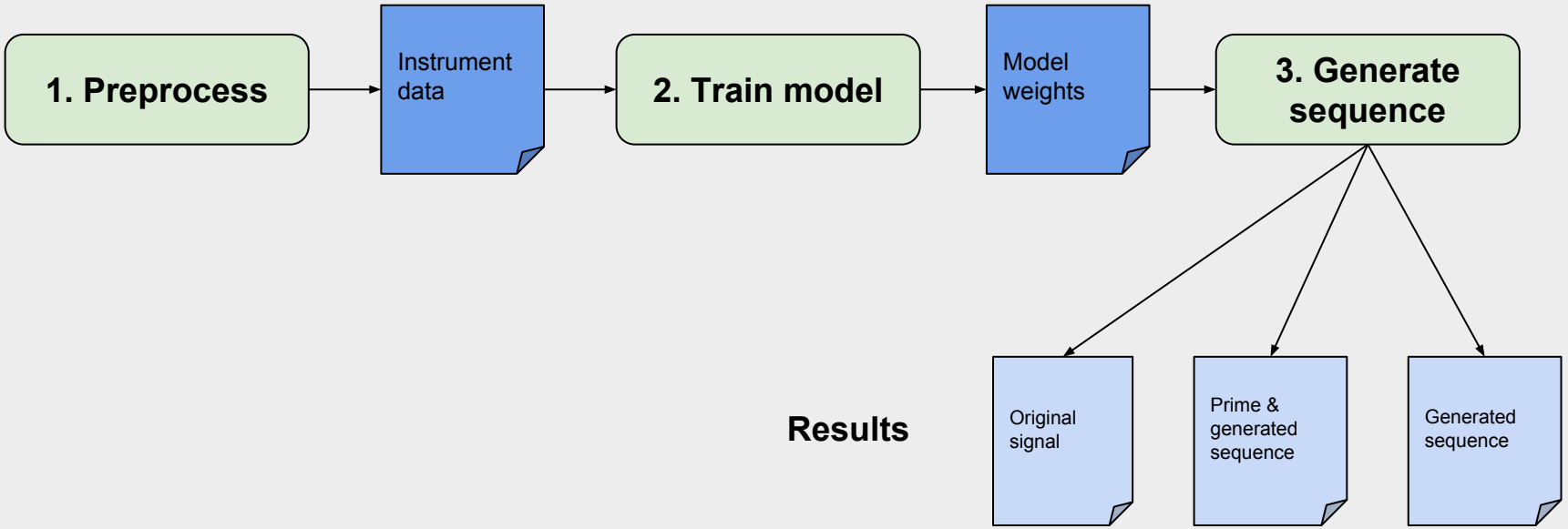


Flute Spectrogram over time

Parameters of experiments

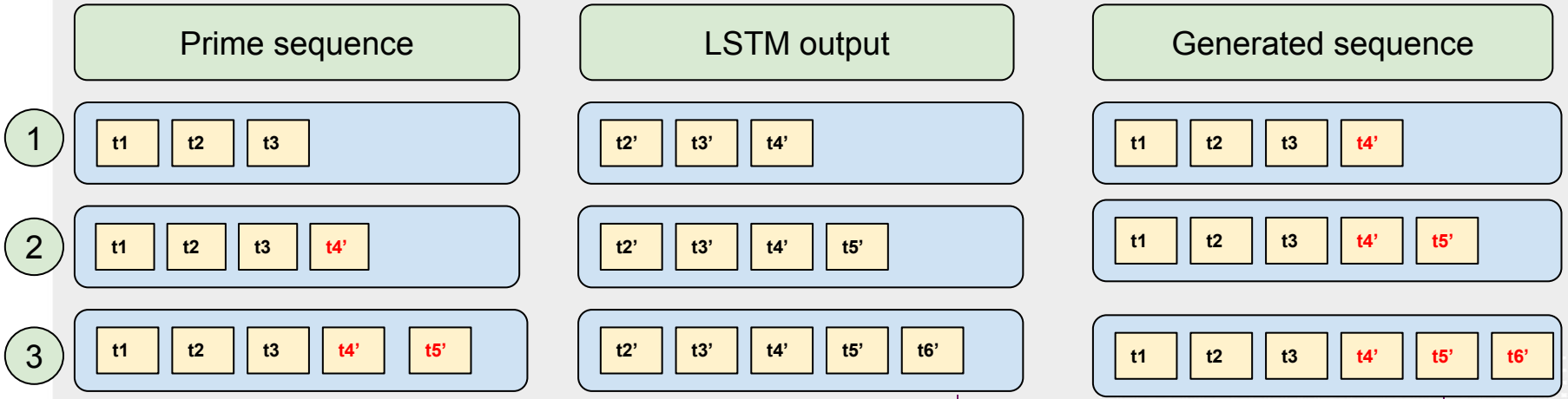


Pipeline


1. Preprocess
Instrument data
2. Train model
Model weights
3. Generate sequence
Results
Original signal
Prime & generated sequence
Generated sequence

Generating process

- In most cases the prime sequence was part of the training set (overfitting)
- Trivial: the length of the prime sequence greatly determines the quality of the generated sequence
- Depending on the instrument we mostly used ⅓ of a second for the prime signal



Selection of experiments

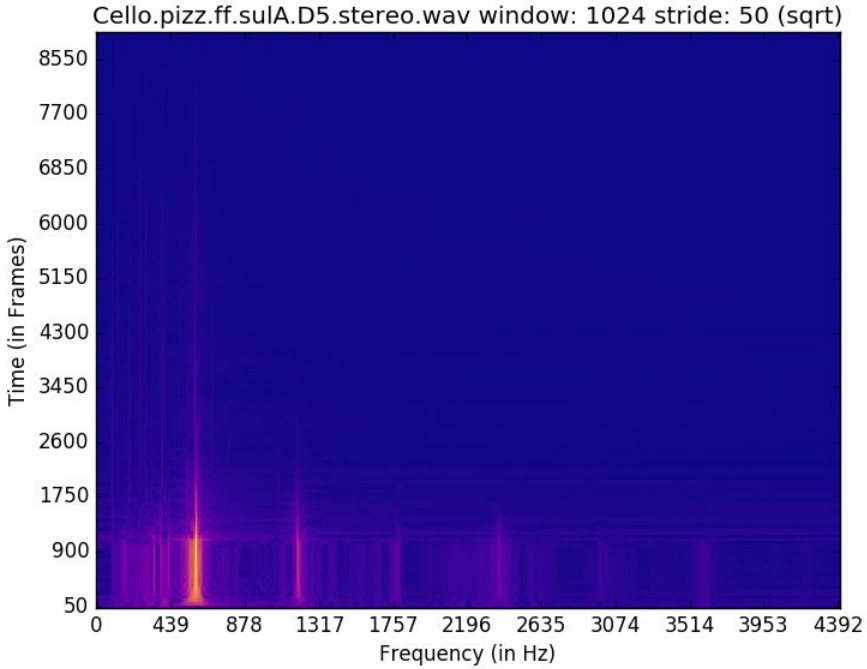
	Instrument	Time slices/sec	Signal length	# of hidden units	# of recur layers	Prime length	Remarks
1	Cello pizzicato	60	3	512	1	20	
2	Guitar	10	5	512	1	3	
3	Guitar	30	5	512	1	10	
4	Guitar	60	5	512	1	20	
5	Guitar	60	5	1024	1	20	
6	Guitar	60	5	512	2	20	
7	Piano	60	5	512	1	20	Activation tanh (instead of linear)
8	Guitar	60	5	512	1	20	prime signals were not part of the training

Note: all other parameters were held fixed. We used 180 epochs and a batch size of 10. “RMSprop” as optimizer and “mse” as loss function.

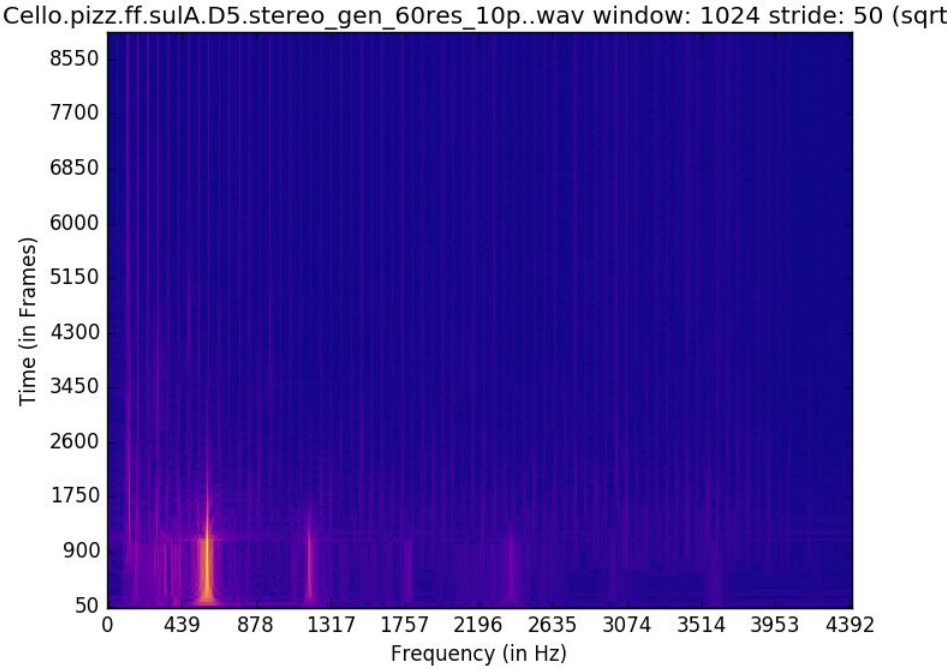
Findings

- Model 1 with one or two recurrent layers
  - Can generate good results on the homogeneous guitar dataset
  - On cello and piano, some signal is learned, but with bad decay and much noise
  - Adding spectra has no beneficial effect
- Model 2 does not learn
  - We tried different CRNNs, but so far none of them work

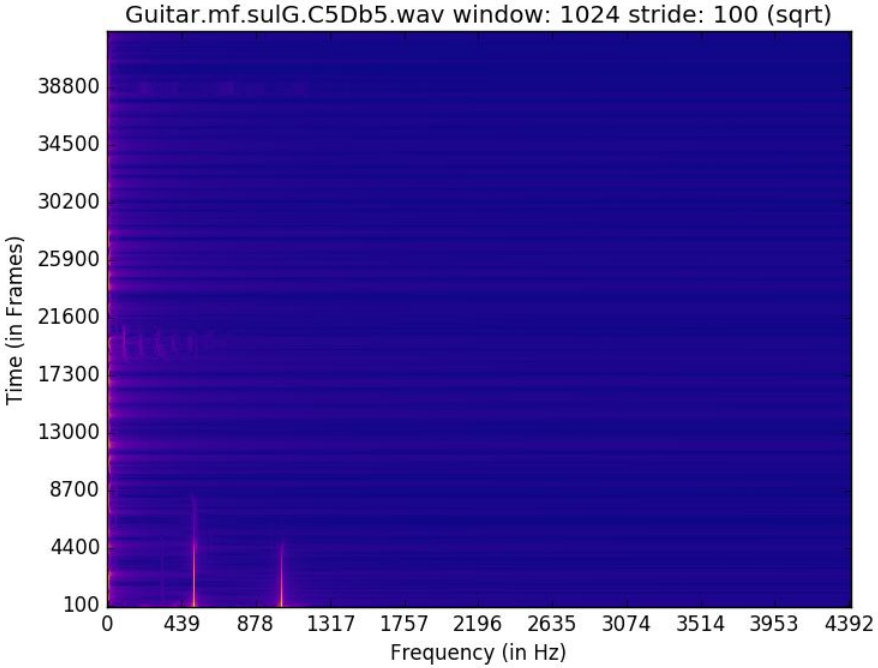
Best performance is achieved with a single LSTM layer between dense layers of 512 units each and 60 time slices per second, which, at a sample rate 8000 leads to ~130 in- and output units



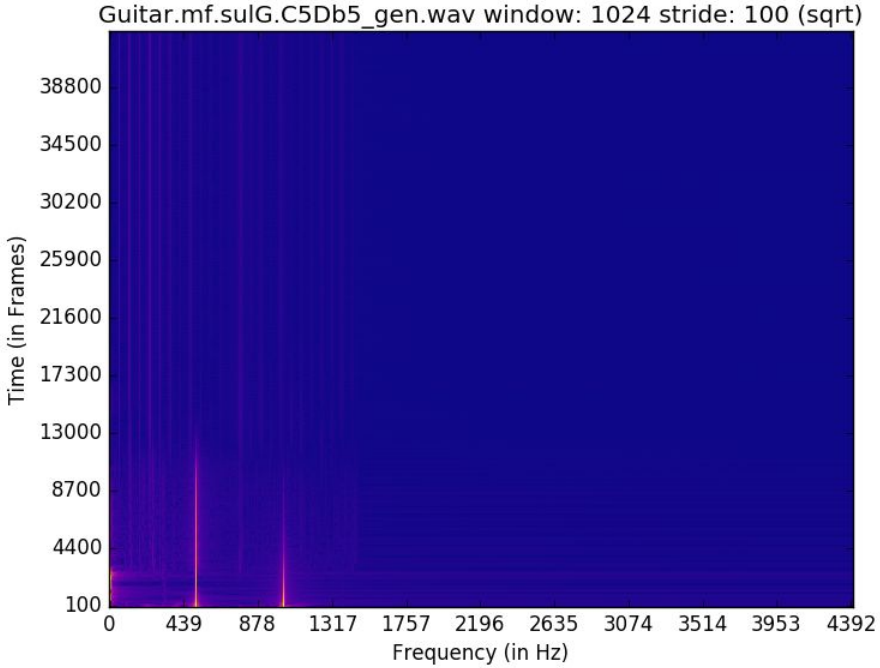
Cello Sample



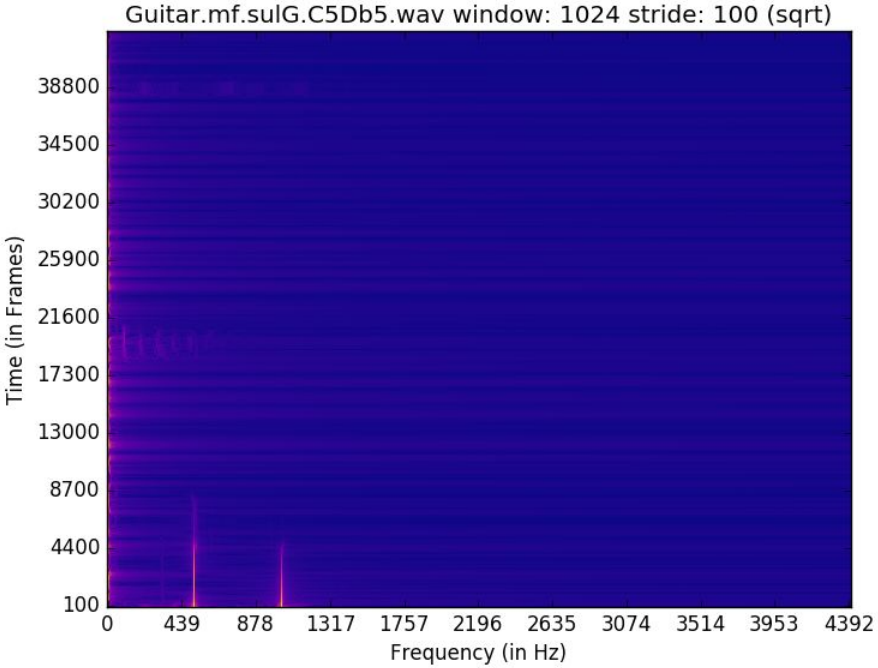
Model Reconstruction (in training set)



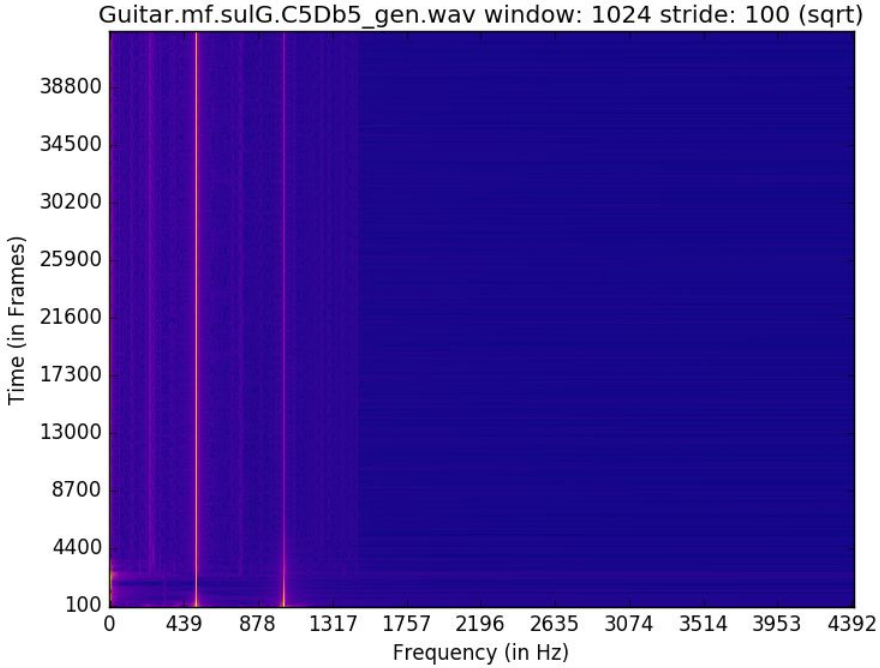
Guitar Sample



Model Reconstruction (in training set)



Guitar Sample



Model Reconstruction (in test set)

Discussion

- What works
  - Can reproduce change in tone over time with some accuracy
  - Can reach low levels of noise
- What doesn't
  - Bad performance on any unseen data
- Why not
  - Data set too small - likely
  - Architecture not powerful enough - subject of some discussion
- What's next
  - Bigger / More diverse data sets could lead to better generalization
  - Find a way to make Conv layers work
  - Move to more complex architecture (VRNN ?)