a software framework for

Musical Data Augmentation

Brian McFee*, Eric J. Humphrey, Juan P. Bello

NYU Center for Data Science

MuseAmi

MARL

Modeling music is hard!

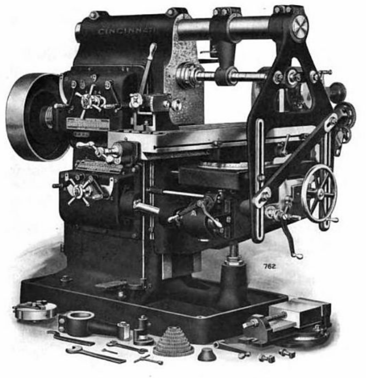
- ❑ Musical concepts are necessarily complex
- ❑ Complex concepts require big models
- ❑ **Big models need big data!**
- ❑ … but good data is hard to find

https://commons.wikimedia.org/wiki/File:Music_Class_at_St_Elizabeths_Orphanage_New_Orleans_1940.jpg


http://photos.jdhancock.com/photo/2012-09-28-001422-big-data.html

Data augmentation

Training data

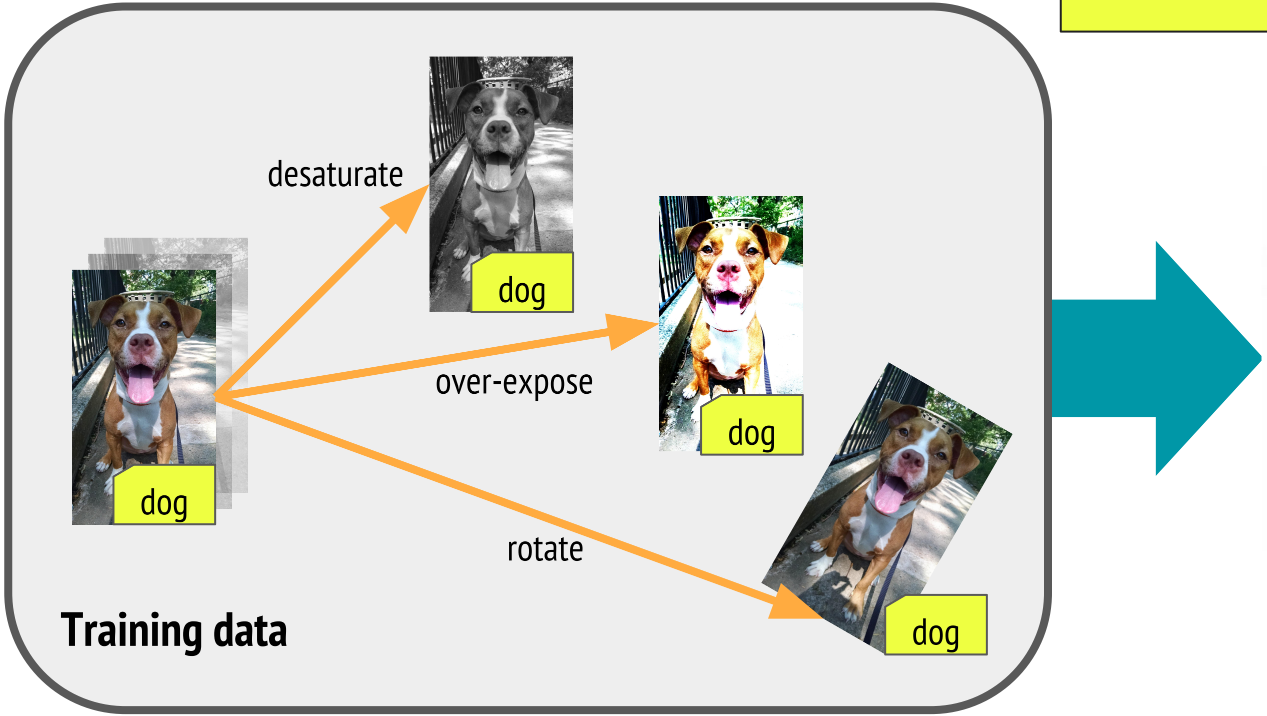


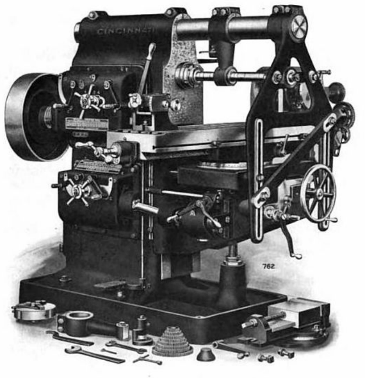
Machine learning

https://commons.wikimedia.org/wiki/File:Horizontal_milling_machine--Cincinnati--early_1900s--001.png

Data augmentation

Note: test data remains unchanged



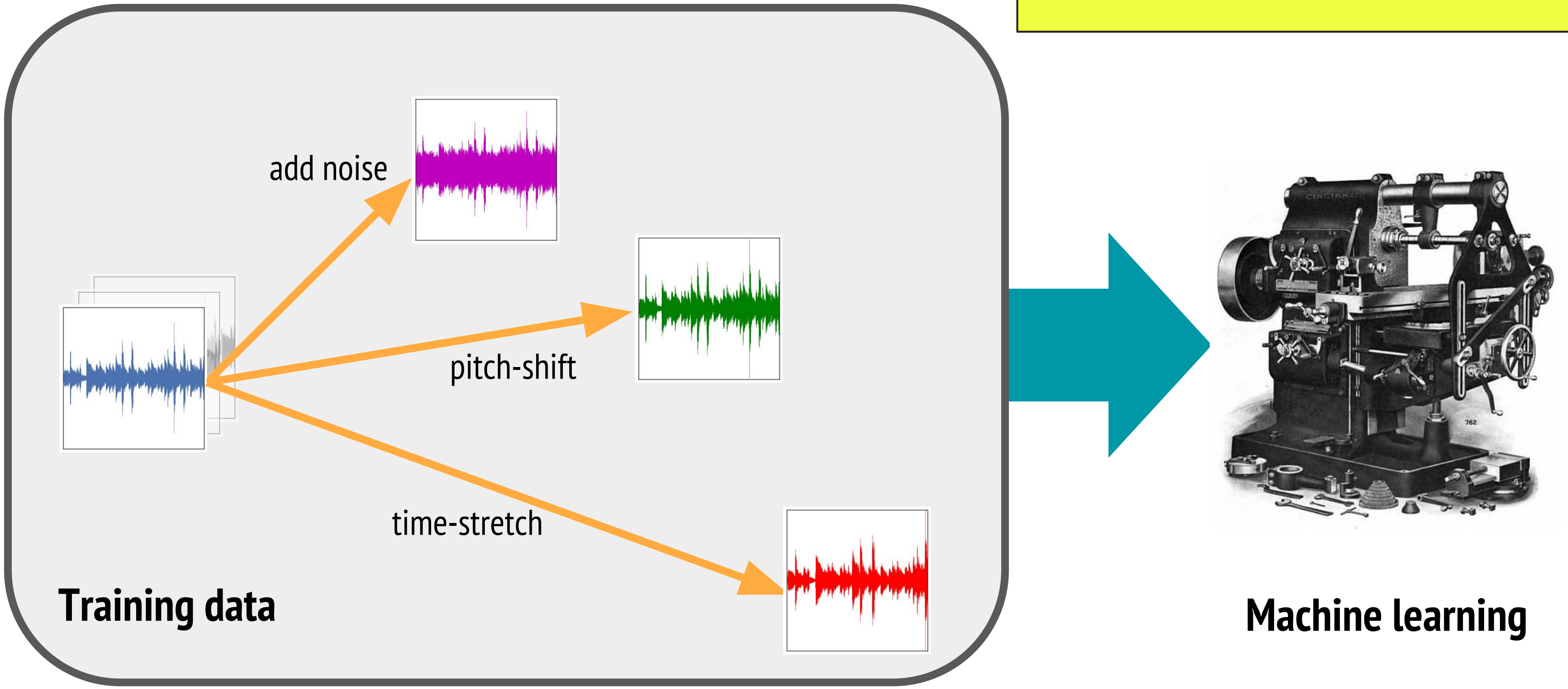


Machine learning

https://commons.wikimedia.org/wiki/File:Horizontal_milling_machine--Cincinnati--early_1900s--001.png

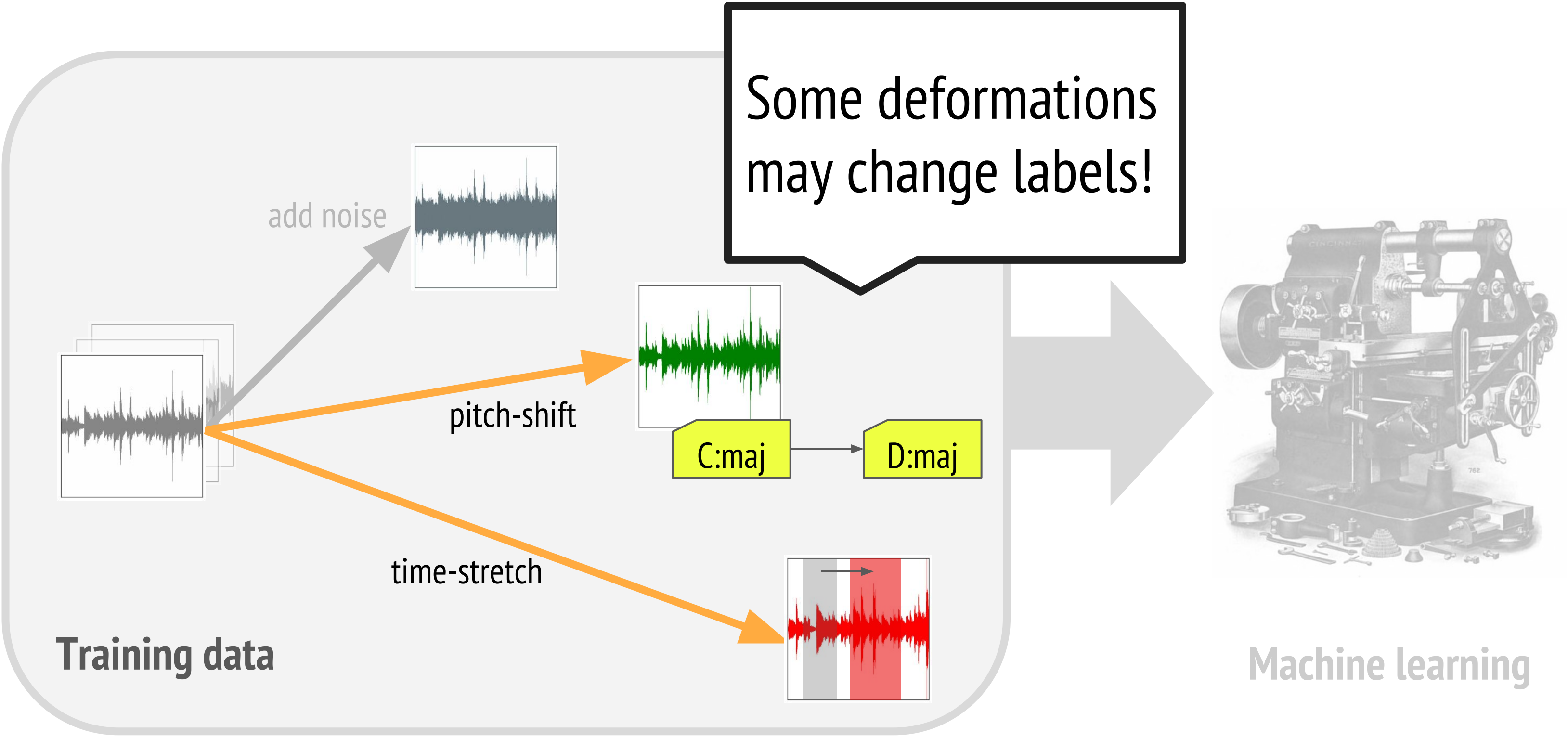
Deforming inputs and outputs

Note: test data remains unchanged



https://commons.wikimedia.org/wiki/File:Horizontal_milling_machine--Cincinnati--early_1900s--001.png

Deforming inputs and outputs



https://commons.wikimedia.org/wiki/File:Horizontal_milling_machine--Cincinnati--early_1900s--001.png

The big idea

Musical data augmentation applies to **both**

input (audio) and **output** (annotations)


... but how will we keep everything contained?
https://www.flickr.com/photos/shreveportbossier/6015498526

JAMS

JSON Annotated Music Specification
[Humphrey et al., ISMIR 2014]

- ❏ A simple container for all annotations
- ❏ A structure to store (meta) data
- ❏ *But* v0.1 *lacked a unified, cross-task interface*

Pump up the JAMS: v0.2.0

- ❏ Unified annotation interface
- ❏ DataFrame backing for easy manipulation
- ❏ Query engine to filter annotations by type
  - ❏ chord, tag, beat, *etc.*
- ❏ Per-task schema and validation

	time	duration	value	confidence
0	00:00:00	00:00:01.511000	N	1
1	00:00:01.511000	00:00:03.425000	C	1
2	00:00:04.936000	00:00:01.742000	G:9	[illegible]

chord

	time	duration	value	confidence
0	00:00:00	00:00:01.437000	silence	1
1	00:00:01.437000	00:00:35.111000	intro	1
2	00:00:36.548000	00:00:12.864000	verse	[illegible]

segment

	time	duration	value	confidence
0	00:00:01.561542	0 days	1	1
1	00:00:02.008526	0 days	2	1
2	00:00:02.446077	0 days	3	1
3	00:00:02.886395	0 days	4	[illegible]

beat

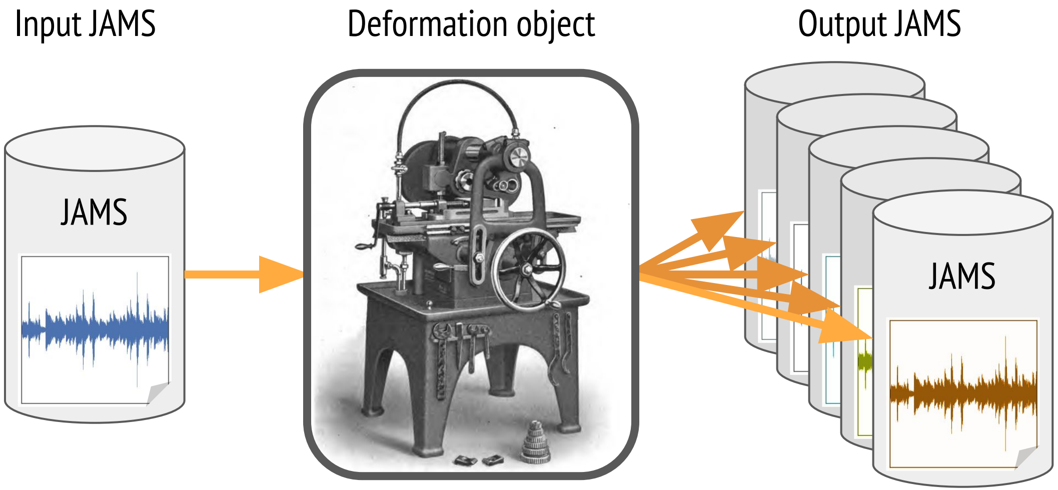
Musical data augmentation

```
In [1]: import muda
```

Deformer architecture

transform(input JAMS ***J_orig***)

1. For each state ***S***:
    a. ***J*** := copy ***J_orig***
    b. modify ***J.audio*** by ***S***
    c. modify ***J.metadata*** by ***S***
    d. Deform each annotation by ***S***
    e. Append ***S*** to ***J.history***
    f. yield ***J***



Deformer architecture

transform(input JAMS ***J_orig***)

1. For each state ***S***:
    a. ***J*** := copy ***J_orig***
    b. modify ***J.audio*** by ***S***
    c. modify ***J.metadata*** by ***S***
    d. Deform each annotation by ***S***
    e. Append ***S*** to ***J.history***
    f. yield ***J***

- State encapsulates a deformation's parameters
- Iterating over states implements 1-to-Many mapping
- Examples:
    - pitch_shift ∈ [-2, -1, 0, 1, 2]
    - time_stretch ∈ [0.8, 1.0, 1.25]
    - background noise ∈ sample library

Deformer architecture

transform(input JAMS ***J_orig***)

1. For each state ***S***:
    a. ***J*** := copy ***J_orig***
    b. modify ***J.audio*** by ***S***
    c. modify ***J.metadata*** by ***S***
    d. Deform each annotation by ***S***
    e. Append ***S*** to ***J.history***
    f. yield ***J***

- ❑ Audio is temporarily stored within the JAMS object
- ❑ All deformations depend on the state ***S***
- ❑ All steps are optional

Deformer architecture

transform(input JAMS ***J_orig***)

1. For each state ***S***:
    a. ***J*** := copy ***J_orig***
    b. modify ***J.audio*** by ***S***
    c. modify ***J.metadata*** by ***S***
    d. Deform each annotation by ***S***
    e. Append ***S*** to ***J.history***
    f. yield ***J***

- Each deformer knows how to handle different annotation types, *e.g.*:
    - PitchShift.**deform_chord**()
    - PitchShift.**deform_pitch_hz**()
    - TimeStretch.**deform_tempo**()
    - TimeStretch.**deform_all**()
- JAMS makes it trivial to filter annotations by type
- Multiple deformations may apply to a single annotation

Deformer architecture

transform(input JAMS ***J_orig***)

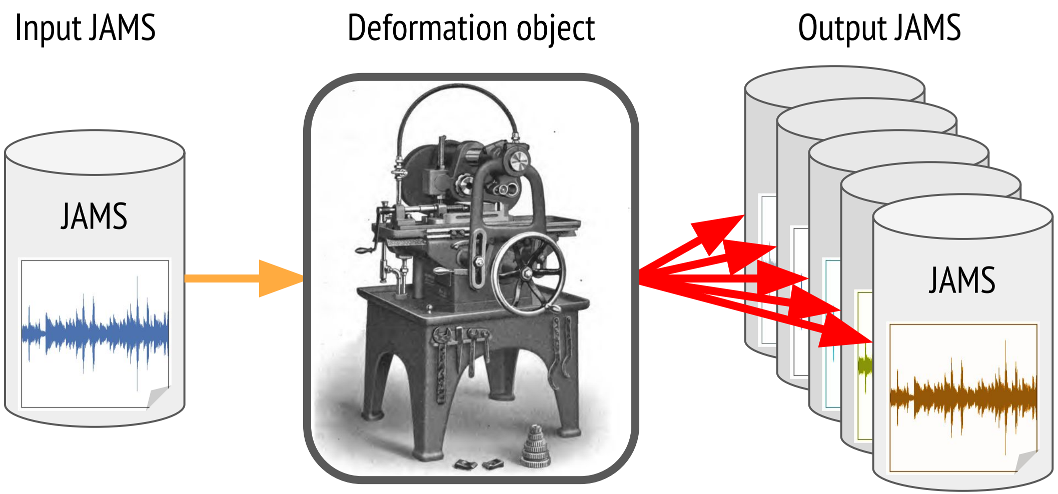
1. For each state ***S***:
   a. ***J*** := copy ***J_orig***
   b. modify ***J.audio*** by ***S***
   c. modify ***J.metadata*** by ***S***
   d. Deform each annotation by ***S***
   e. Append ***S*** to ***J.history***
   f. yield ***J***

- This provides **data provenance**
- All deformations are **fully reproducible**
- The constructed JAMS contains all state and object parameters

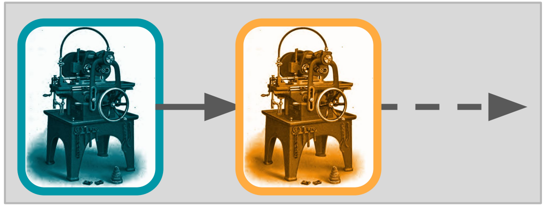
Deformer architecture

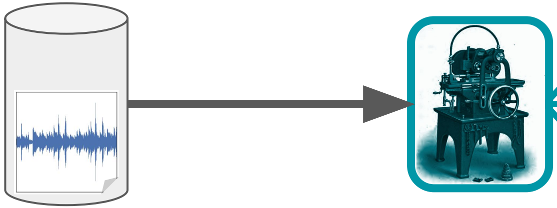
transform(input JAMS ***J_orig***)

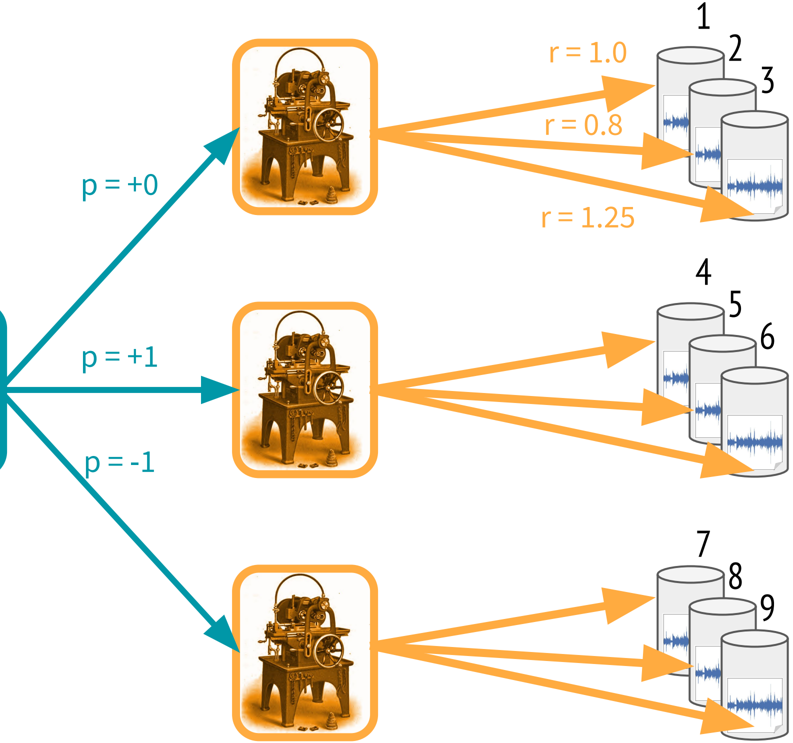
1. For each state ***S***:
    a. ***J*** := copy ***J_orig***
    b. modify ***J.audio*** by ***S***
    c. modify ***J.metadata*** by ***S***
    d. Deform each annotation by ***S***
    e. Append ***S*** to ***J.history***
    f. yield ***J***



Deformation pipelines








```
for new_jam in jam_pipe(original_jam):
        process(new_jam)
```

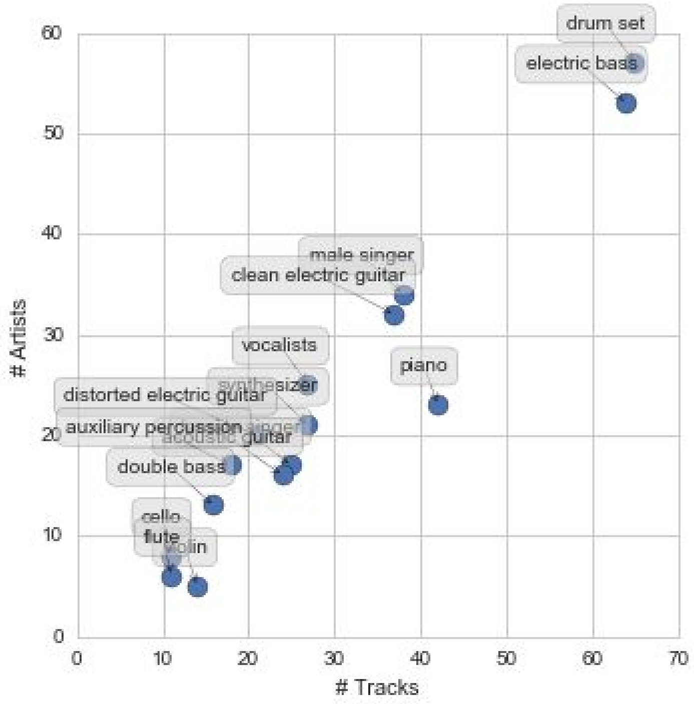
Example application

instrument recognition in mixtures

https://commons.wikimedia.org/wiki/File:Instruments_on_stage.jpg

Data: MedleyDB

- 122 tracks/stems, mixed instruments
  [Bittner et al., ISMIR 2014]
- 75 unique artist identifiers
- We model (the top) 15 instrument classes
- Time-varying instrument activation labels



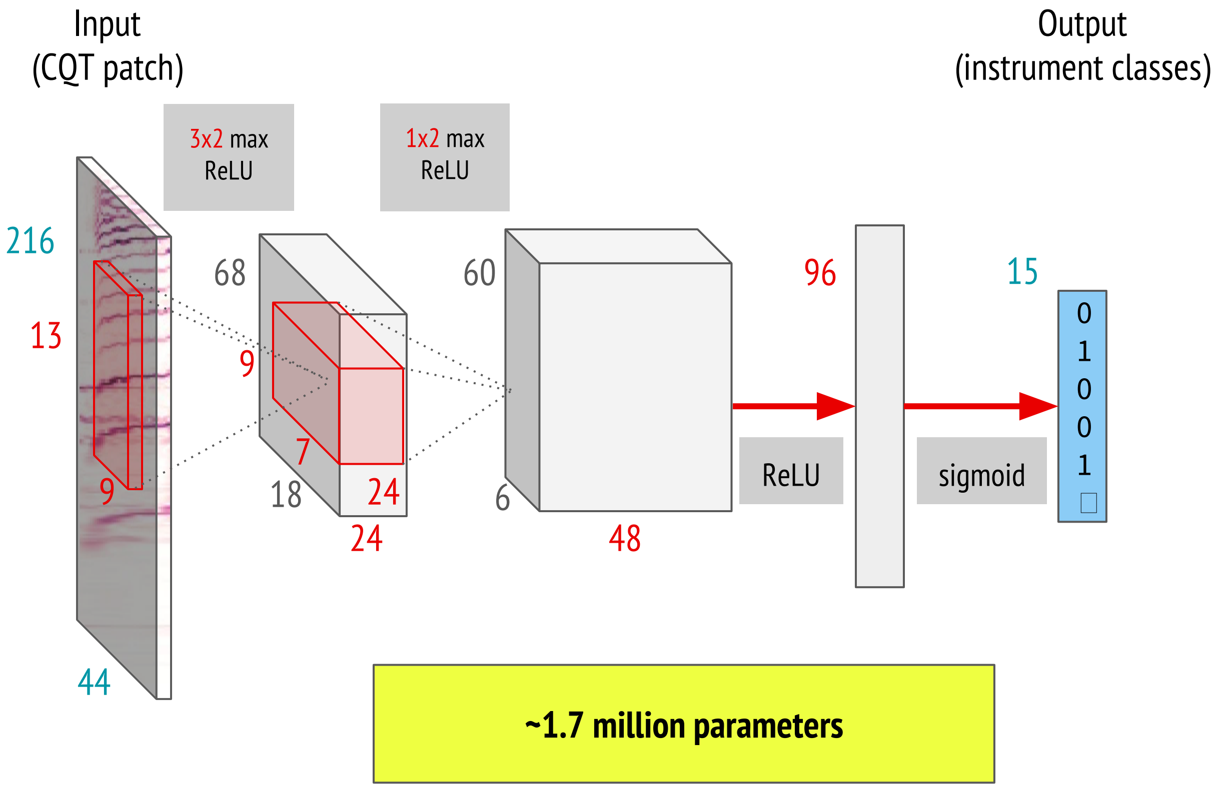
http://medleydb.weebly.com/

Convolutional model

- Input
  a. ~1sec log-CQT patches
  b. 36 bins per octave
  c. 6 octaves (C2-C8)

- Convolutional layers
  a. 24x ReLU, 3x2 max-pool
  b. 48x ReLU, 1x2 max-pool

- Dense layers
  a. 96d ReLU, dropout=0.5
  b. 15d sigmoid, ℓ_2 penalty



Experiment

How does training with data augmentation impact model stability?

Note: test data remains unchanged

- ❏ Five augmentation conditions:

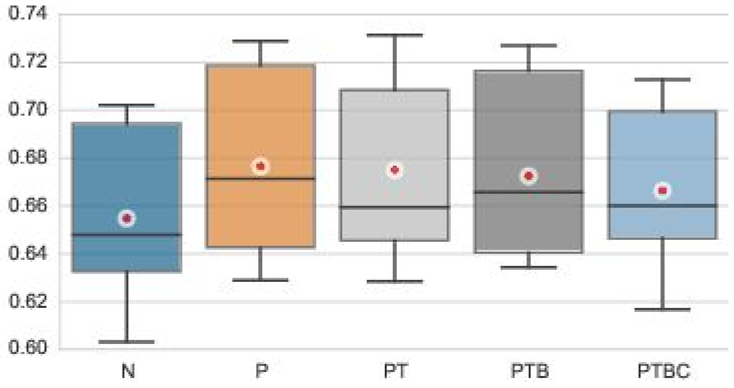
N	Baseline
P	pitch shift [+- 1 semitone]
PT	+ time-stretch [√2, 1/√2]
PTB	++ background noise [3x noise]
PTBC	+++ dynamic range compression [2x]

- ❏ 1 input ⇒ up to 108 outputs
- ❏ 15x (artist-conditional) 4:1 shuffle-splits
- ❏ Predict instrument activity on 1sec clips

Results across all categories

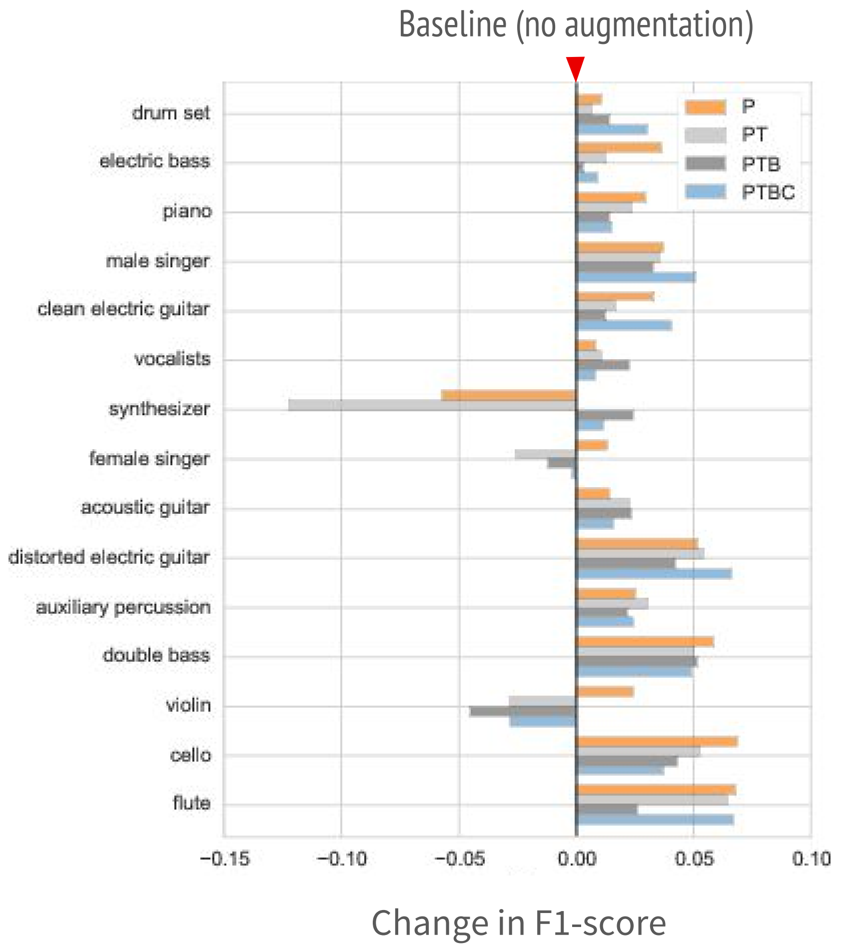
- ❑ Pitch-shift improves model stability
- ❑ Additional transformations don’t seem to help (on average)
- ❑ But is this the whole story?

Label-ranking average precision



Results by category

- All augmentations help for most classes
- synthesizer may be ill-defined
- Time-stretch can hurt high-vibrato instruments



Conclusions

- We developed a general framework for musical data augmentation

- Training with augmented data can improve model stability

- Care must be taken in selecting deformations

- Implementation is available at https://github.com/bmcfee/muda
  soon: pip install muda

Thanks!

brian.mcfee@nyu.edu

https://bmcfee.github.io

https://github.com/bmcfee/muda