Programming MIR Baselines from Scratch*: Three Case Studies

Ethan Manilow, Mark Cartwright, Rachel Bittner

*With preparation :)

If you are….

New to (python) programming

→ Get exposed to some common workflows

An experienced (python) programmer

→ Learn new tricks / different ways of approaching problems

Not really a programmer

→ Gain insight into how systems are often built

Goals

- 0-ish% to 100%-ish on small problems
- Show diverse coding setups & problems
- Focus on ***HOW*** more than ***WHY***

Tutorial Structure

- Three 1hr sections
  - ~50 min live coding walkthroughs
  - ~10 min Q&A / break
- Some parts will be “pre-baked”
  - More information than time 🙂 🙃 🙂
- Confused? Questions?
  - Ask in the chat! One of us will be answering
  - Q&A after each session

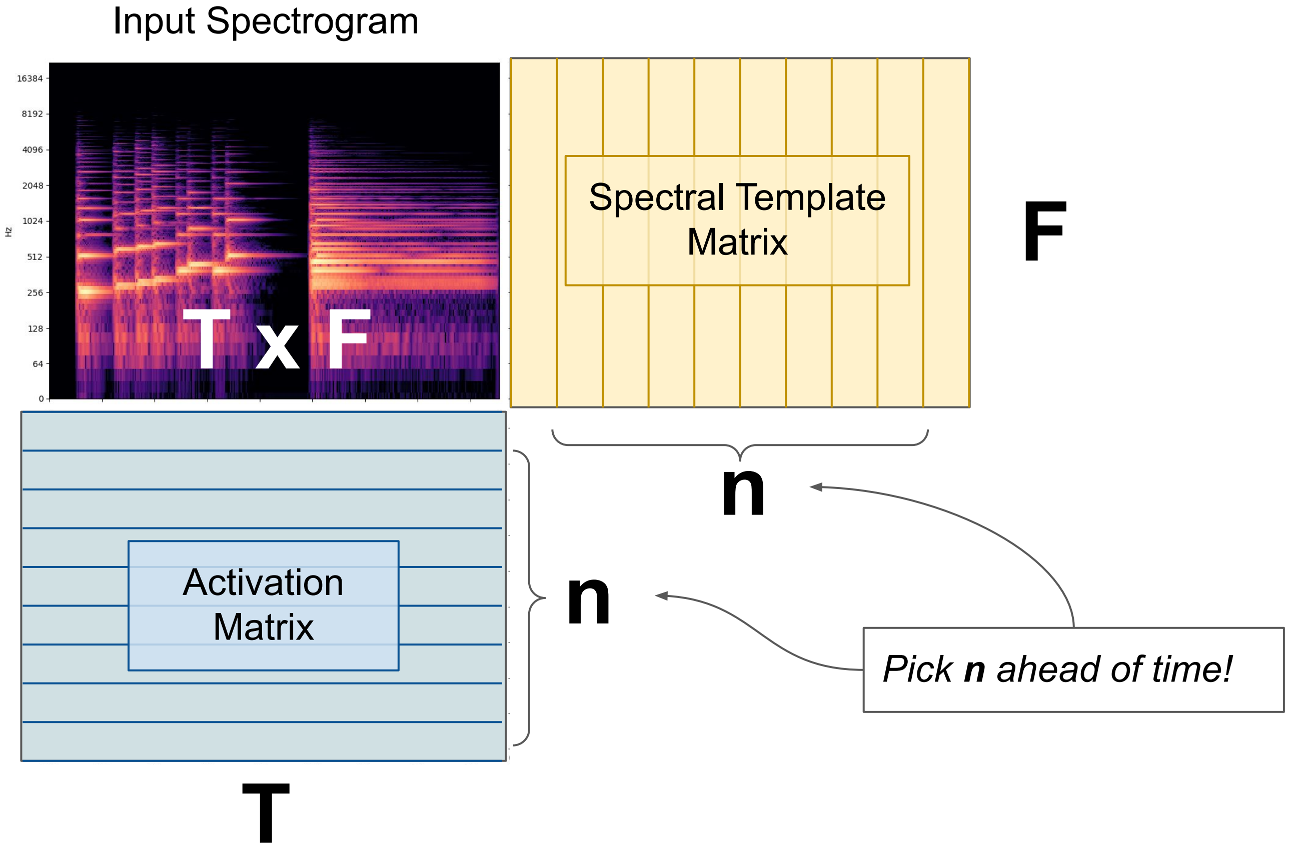
Part 1: NMF Transcription

Ethan Manilow

What is Non-Negative Matrix Factorization (NMF)?

- Matrix decomposition algorithm
  - Split an input spectrogram into a *“what”* and *“when”* matrix...
- Unsupervised
  - Don’t need paired training data
  - → Can run on *just one* clip!
- Useful for Source Separation too!
- Surpassed by deep learning (wait until Mark & Rachel’s part)

What is Non-Negative Matrix Factorization (NMF)?



Input Spectrogram

T x F

Spectral Template Matrix

F

n

Activation Matrix

n

T

*Pick **n** ahead of time!*

What is Non-Negative Matrix Factorization (NMF)?

Activation Matrix

T x n

When do things happen?

*

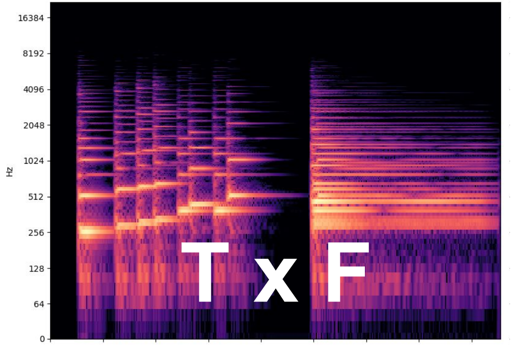
Spectral Template Matrix

n x F

What is happening?

=

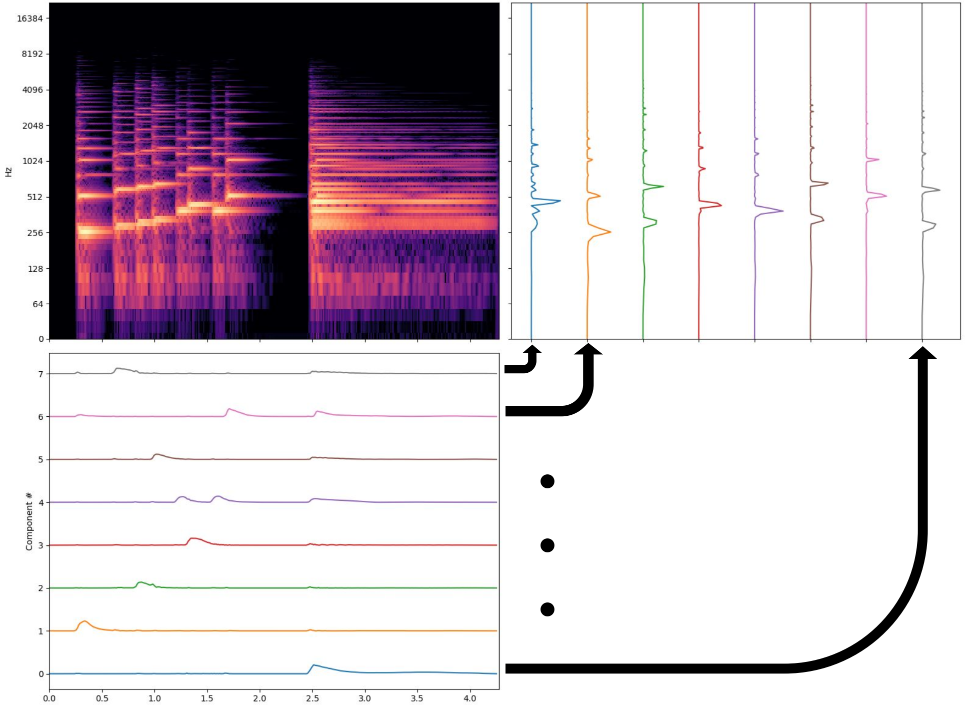
Spectrogram Reconstruction



What is Non-Negative Matrix Factorization (NMF)?

Input Spectrogram

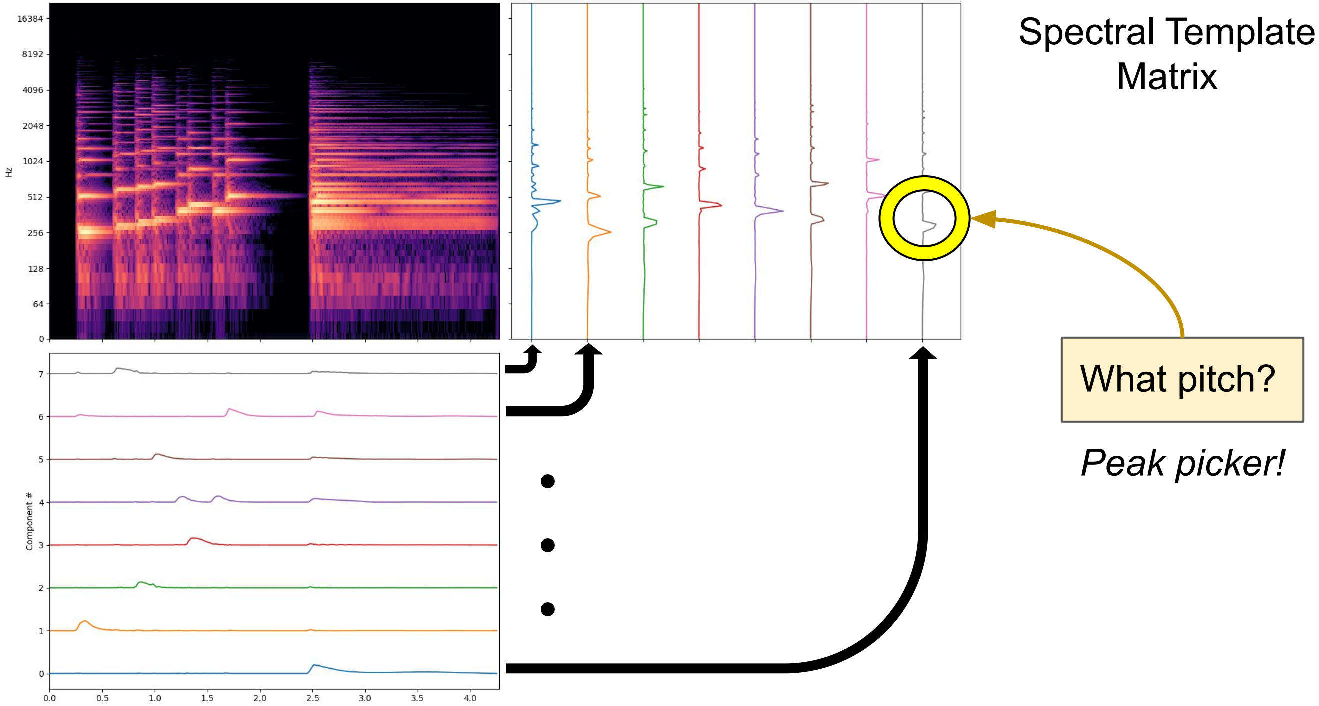
Spectral Template Matrix



Activation Matrix

How do we use NMF for transcription?

Input Spectrogram



Spectral Template Matrix

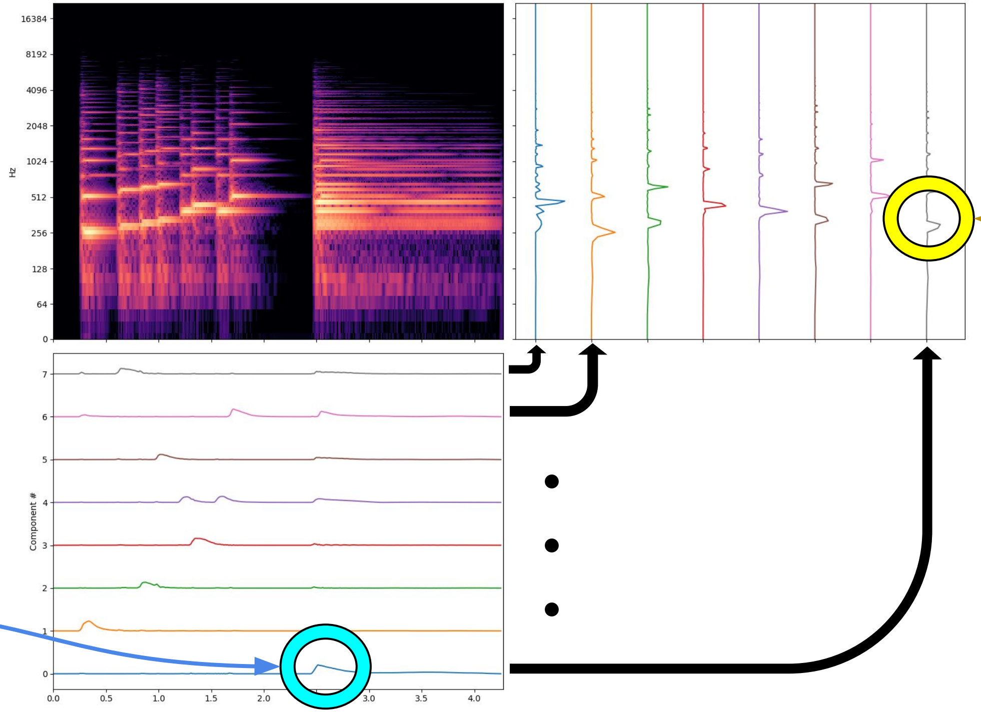
What pitch?

Peak picker!

Activation Matrix

How do we use NMF for transcription?

Input Spectrogram



Spectral Template Matrix

What pitch?

Peak picker!

When does it happen?

Peak picker!

Activation Matrix

Goal: Use NMF for transcription

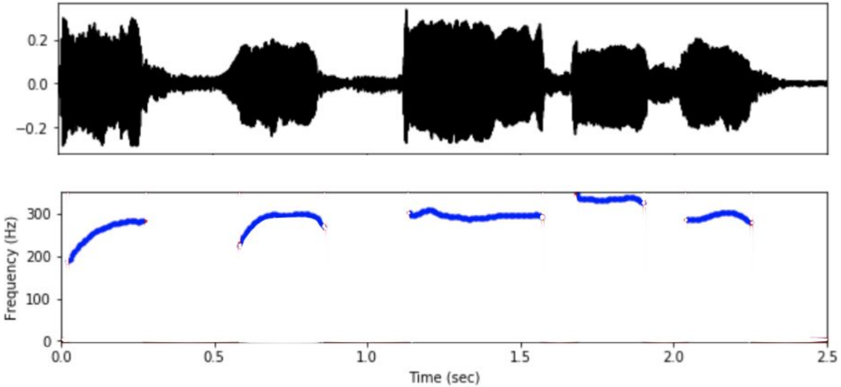
- Get set up from a blank project
- Use PyCharm & get running code
  - Interactive debugging
  - Plotting
  - Terminal
- Use `nussl` (`sklearn`) NMF to transcribe a simple audio clip

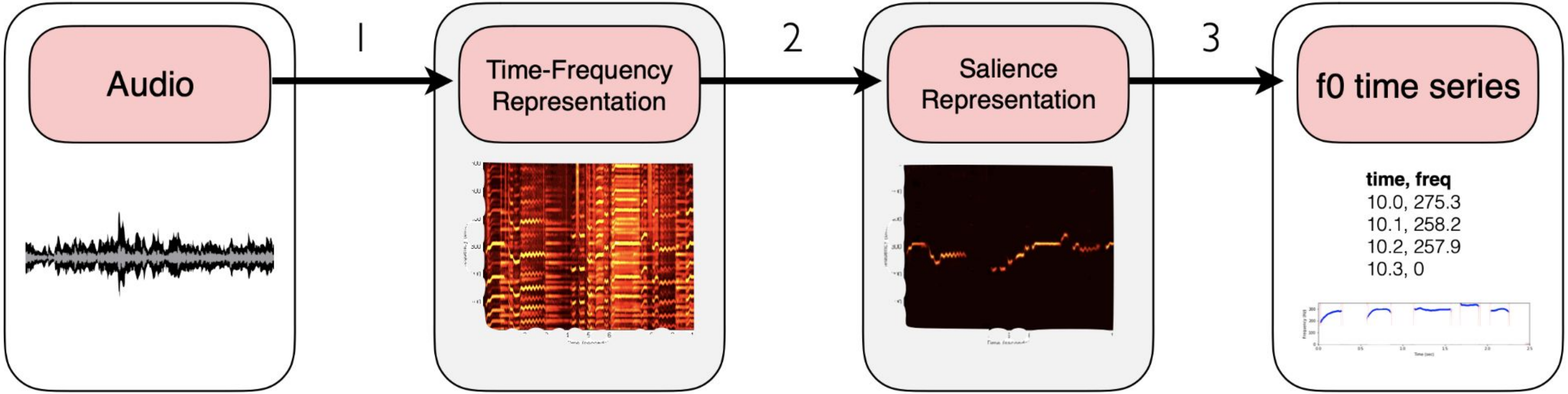
Part 2: Pitch Tracking

Rachel Bittner

Pitch Tracking

- Input: solo, monophonic music
- Output: The pitch value (in Hz) for each time frame





1. Harmonic Constant-Q Transform (HCQT)
2. A neural network
3. Viterbi decoding

Goal: Build a pitch tracker from scratch & run evaluation

Steps:

1. Preprocess the training* data
    a. Iterate over dataset/s, split full tracks into short snippets
    b. Compute input representation (HCQT) and output representation (target salience matrix)
    c. Save (hcqt, target_salience) tuples as individual npz files
2. Build & train a model
    a. A simple convnet
    b. Write the dataset/dataloader
    c. Write the training loop
    d. Add tensorboard visulalizations
3. Run evaluation
    a. mir_eval on a different dataset

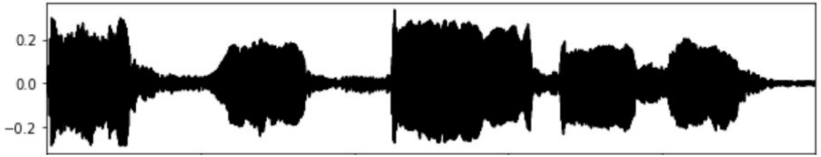
*we'll skip everything related to validation, for simplicity

Part 3: Instrument Classification

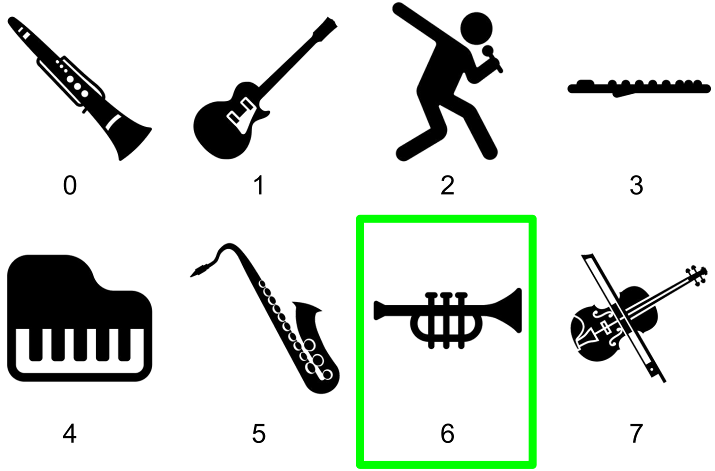
Mark Cartwright

Instrument Classification

- Input: solo instrument recording

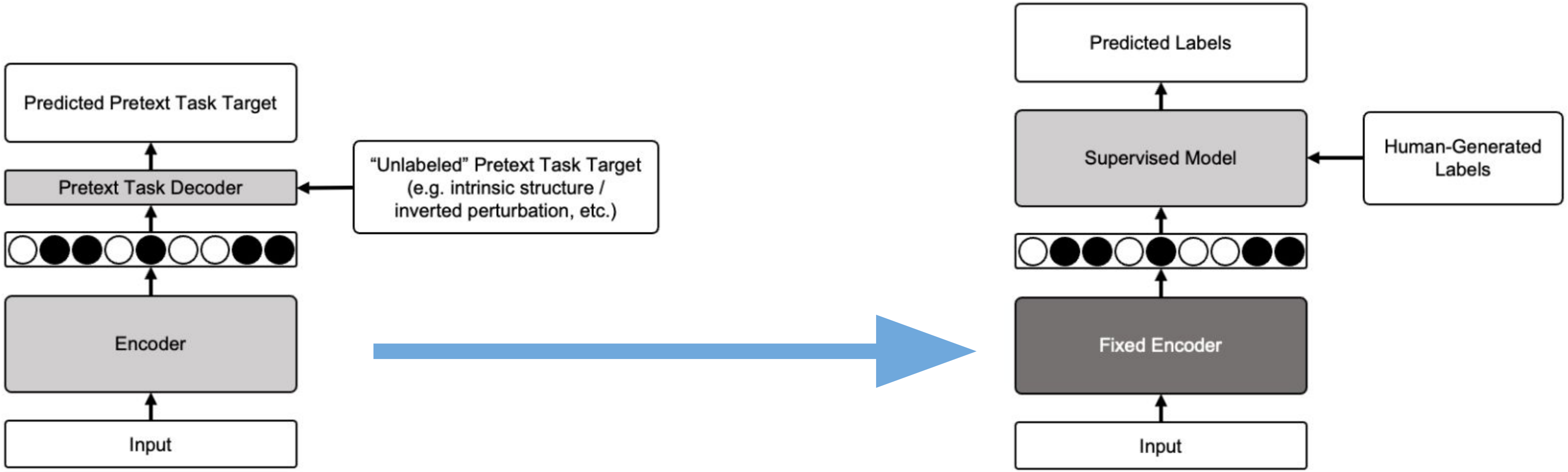


- Output: instrument class



Instrument Classification using Transfer Learning

- One form of transfer learning is to train a model on a “pretext task” with lots of data, and then use that model on a new (but related) “downstream task” for which you may not have a lot of data.

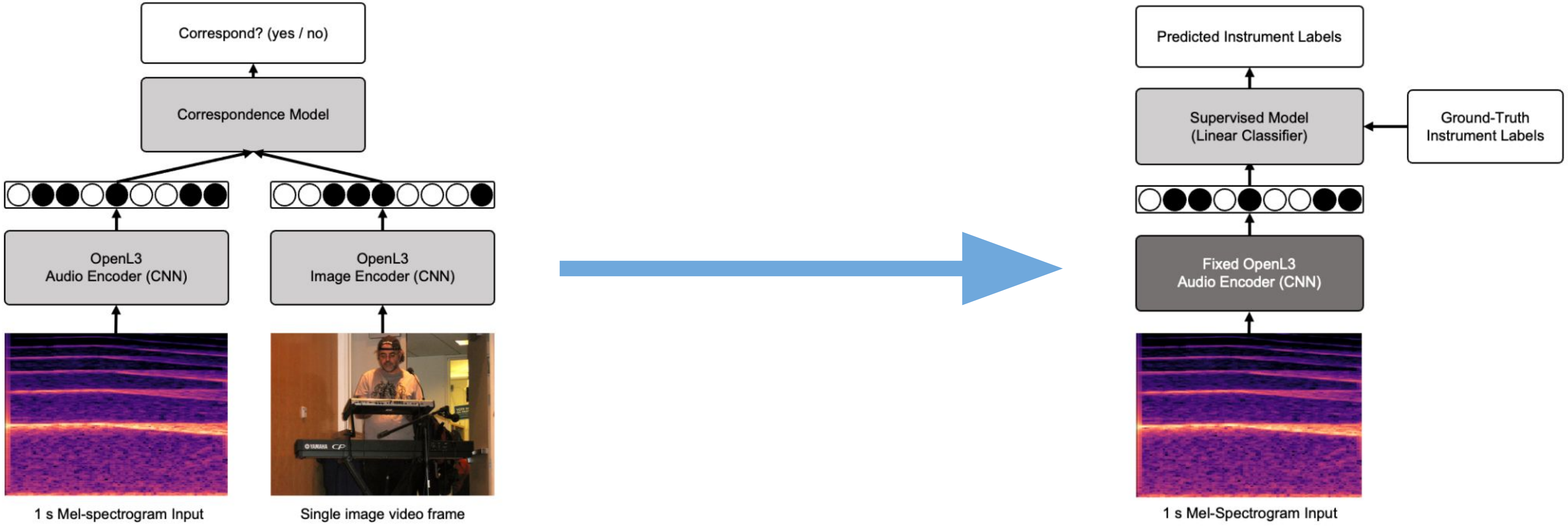

Predicted Pretext Task Target
Pretext Task Decoder
“Unlabeled” Pretext Task Target (e.g. intrinsic structure / inverted perturbation, etc.)
Encoder
Input
Predicted Labels
Supervised Model
Human-Generated Labels
Fixed Encoder
Input

Pretext Task

Downstream Task

Instrument Classification using Transfer Learning

- We'll use a model (OpenL3[1]) trained with audio-visual correspondence on YouTube videos of music (our pretext task) and apply it to instrument classification w/ Medley Solos DB (our downstream task)



Pretext Task

Downstream Task

[1] Cramer, et al. "Look, listen, and learn more: Design choices for deep audio embeddings", 2019. (based on Arandjelovic and Zisserman "Look, Listen and Learn", 2017.)

Goal: Combine a pre-trained OpenL3 model with a simple linear classifier for instrument classification

Steps:

1. Prepare data and data loaders
   a. Download data
   b. Inspect data
   c. Write data loader functions
2. Build & train a model
   a. Load OpenL3 model and freeze all layers
   b. Extend model for instrument classification
   c. Train model
   d. Save the model
3. Evaluate model
4. Fine-tune model
   a. Unfreeze some OpenL3 layers
   b. Lower learning rate
   c. Train model to fine-tune
   d. Save model