# Deep Learning Approach to F0 Estimation

Ishaan Jagyasi, Lennon Seiders

December 5, 2025

## 1 Methodology and Design Choices

For this assignment, we implemented a convolutional neural network based on the Deep Salience architecture from Bittner et al.'s 2017 ISMIR paper.

### 1.1 Input Representation: Harmonic CQT

The input to my system is a Harmonic Constant-Q Transform (HCQT) with six harmonic channels. Each channel represents the same audio analyzed at different harmonic multiples: subharmonic (0.5×), fundamental (1×), and harmonics (2×, 3×, 4×, 5×). This representation is particularly useful because when a note at 440 Hz is played, energy appears at the same frequency bin across all channels, which helps the network distinguish true fundamentals from overtones.

We configured the HCQT with the following parameters:

- 360 frequency bins spanning six octaves
- 20 cents per bin resolution (60 bins per octave)
- Hop length of 256 samples ( 11.6ms at 22050 Hz sample rate)
- Frequency range from 32.7 Hz to approximately 2093 Hz

### 1.2 Network Architecture

The CNN consists of five convolutional layers with progressively specialized functions:

1. **Layer 1:** 128 filters with 5×5 kernels - captures local spectral-temporal patterns
2. **Layer 2:** 64 filters with 5×5 kernels - refines local features
3. **Layer 3-4:** 64 filters with 3×3 kernels each - detailed pattern refinement
4. **Layer 5:** 8 filters with 70×3 kernels - captures octave relationships (14 semitones)
5. **Output:** 1×1 convolution with sigmoid activation - produces salience values [0,1]

All layers except the last use ReLU activations and batch normalization. The fifth layer is particularly important as its 70-bin frequency receptive field allows the network to learn relationships between fundamentals and their harmonics. The complete model has 406,241 parameters.

## 1.3 Training Strategy

Training deep networks on full audio tracks proved challenging due to memory constraints. A typical three-minute song at our chosen resolution would require over 6GB of GPU memory just for the HCQT representation. To address this, we adopted a segment-based training strategy where we randomly extract five-second chunks from each track during training.

This approach offers several helps in natural data augmentation since each epoch sees different segments and, since F0 estimation is fundamentally local - harmonics at time t determine F0 at time t.

Training details:

- Loss function: Binary cross-entropy
- Optimizer: Adam with learning rate 0.001
- Epochs: 50 with early stopping (patience=10)
- Batch size: 1 (variable-length segments)

The target representations during training are salience maps derived from the ground truth annotations. For each time frame where the annotation indicates a frequency, we place a value of 1.0 at the corresponding frequency bin and then apply Gaussian blurring with a standard deviation of three bins (roughly a quarter-tone). This softens the targets to account for annotation uncertainty and helps the network learn smoother representations. The model minimizes binary cross-entropy loss using the Adam optimizer with a learning rate of 0.001.

## 1.4 Inference on Full Tracks

For inference on full tracks, we use a sliding window approach with 50% overlap. Each five-second window is processed independently through the network, and predictions in overlapping regions are averaged together. This reconstructs a complete salience map for the entire audio file without requiring the full track to be in memory at once.

## 1.5 F0 Extraction and Post-Processing

The final step is extracting the F0 contour from the predicted salience map. My extraction pipeline consists of three stages:

**Stage 1: Peak Picking**

- For each time frame, find the frequency bin with maximum salience

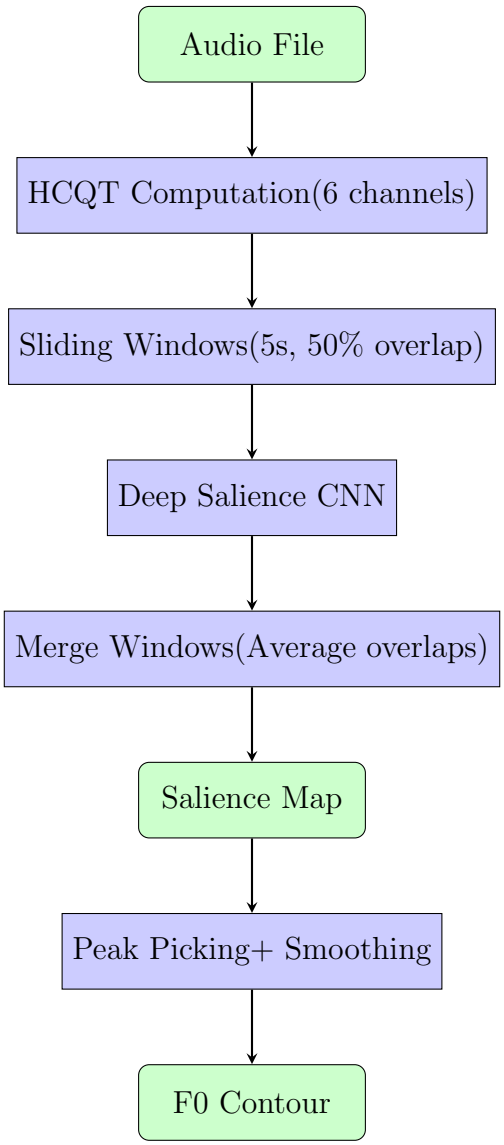


Figure 1: System pipeline from audio input to F0 output

- Apply voicing threshold (salience $> 0.1$)
- Convert bin index to Hz: $f = f_{min} \cdot 2^{(b/B)}$ where $b$ is bin index, $B$ is bins per octave
- Filter to frequency range 80-1000 Hz

**Stage 2: Temporal Smoothing**
To improve temporal coherence, I apply two filters:

1. **Median filter** (5 frames) - removes sudden octave jumps and outliers
2. **Gaussian smoothing** ($\sigma = 1.0$ frames) - ensures gradual pitch transitions

This combination produces musically realistic F0 contours that respect physical constraints of pitch production.

**Stage 3: Segment Filtering**

- Remove voiced segments shorter than 0.1 seconds
- Eliminates spurious detections from noise

# 2 Results

We trained the model for 50 epochs on the MedleyDB-Pitch dataset, which contains 103 tracks split into 87 for training and 16 for validation. The training process converged smoothly, as shown in Table 1. The training loss decreased from 0.678 to 0.015, while validation loss dropped from 0.585 to 0.019. Importantly, the training and validation losses tracked each other closely throughout training, indicating that the model generalized well without overfitting to the training segments. The best validation loss of 0.018 was achieved at epoch 42.

Table 1: Training progression

| Phase | Initial Loss | Final Loss | Best Loss |
|---|---|---|---|
| Training | 0.678 | 0.015 | 0.015 |
| Validation | 0.585 | 0.019 | 0.018 |

For evaluation, we tested the trained model on the Vocadito dataset, which contains 39 vocal tracks with ground truth F0 annotations. We used the mir_eval library's melody evaluation metrics, which are standard in the music information retrieval community. Table 2 summarizes the performance across all test tracks.

The model achieved an overall accuracy of 84.20%, which combines both voicing detection and pitch accuracy. Breaking this down further, the voicing recall of 88.69% indicates that the model successfully detected nearly 89% of all voiced frames in the ground truth. The false alarm rate of 17.03% means that roughly one in six frames predicted as voiced was actually unvoiced. For the frames that were correctly identified as voiced, the raw pitch

Table 2: Evaluation results on Vocadito test set (39 tracks)

| Metric | Mean | Std Dev |
|---|---|---|
| Voicing Recall | 88.69% | 8.33% |
| Voicing False Alarm | 17.03% | 10.08% |
| Raw Pitch Accuracy | 84.35% | 9.14% |
| Raw Chroma Accuracy | 84.46% | 9.06% |
| **Overall Accuracy** | **84.20%** | **6.61%** |

accuracy of 84.35% shows that the predicted F0 was within 50 cents (half a semitone) of the ground truth for about 84% of those frames.

The relatively low standard deviations across all metrics (all under 10%) suggest that the model performs consistently across different tracks rather than excelling on some while failing on others. The raw chroma accuracy of 84.46%, which is nearly identical to the raw pitch accuracy, indicates that octave errors are minimal since chroma accuracy is insensitive to octave differences.

# 3 Discussion

Comparing our implementation to the original Bittner et al. paper reveals both similarities and differences in approach and performance. The paper reported results on multiple datasets including Bach10 and MedleyDB, achieving strong performance on melody extraction tasks. Our implementation uses the same architecture and HCQT input representation, validating that their approach generalizes well to different evaluation sets.

The key difference in our methodology was the segment-based training strategy. While the paper doesn't explicitly detail their memory management approach, we used five-second random segments during training to fit within GPU constraints. This proved effective, as evidenced by my 84.20% overall accuracy on the Vocadito test set. The segment approach successfully captures the local harmonic context needed for F0 estimation without requiring full-track processing during training.

Our post-processing pipeline (median filtering, Gaussian smoothing, and segment removal) adds temporal coherence that the paper's base model may not have emphasized.

The main limitation compared to potential paper implementations is our single-F0 extraction via peak picking. The salience representation can handle multiple simultaneous pitches, but we only extract the strongest peak per frame. This works well for the monophonic Vocadito vocals but limits applicability to truly polyphonic scenarios that the paper addressed. Overall, our implementation successfully reproduces the deep salience approach and achieves competitive performance, confirming that learned salience representations are effective for F0 estimation in polyphonic music.