CHAPTER IV

CREPE: DEEP MONOPHONIC PITCH ESTIMATION

As outlined in Chapter I, we start with a simpler formulation of music transcription that is later extended to ultimately support multi-instrument polyphonic music transcription. This chapter concerns the first step where we assume that the input is monophonic, i.e. the audio contains at most one pitched sound at each instant. Through this simplified formulation, we aim to gain insights on how a supervised learning framework for music transcription tasks can be designed. Furthermore, we seek to identify important aspects of the proposed approach that can be applied to or extended to polyphonic and multi-instrument transcription tasks. The content of this chapter is largely based on the work presented at IEEE ICASSP 2018 (J. W. Kim et al., 2018).

1 Introduction

Estimating the fundamental frequency (f0) of a monophonic audio signal, also known as pitch tracking or pitch estimation, is a long-standing topic of research in audio signal processing. Pitch estimation plays an important role in music signal processing, where monophonic pitch tracking is used as a method to generate pitch annotations for multi-track datasets (Bittner et al., 2014) or as a core component of melody extraction systems (Bosch & Gómez, 2014; Mauch et al., 2015). Pitch

estimation is also important for speech analysis, where prosodic aspects such as intonations may reflect various features of speech (Zubizarreta, 1998).

We have reviewed a few approaches for monophonic pitch tracking in Section II.2, including YIN (de Cheveigné & Kawahara, 2002) and pYIN (Mauch & Dixon, 2014). A notable trend in those methods is that the derivation of a better pitch detection system solely depends on cleverly devising a robust candidate-generating function and/or sophisticated post-processing steps, i.e. heuristics, and none of them are directly learned from data, except for manual hyperparameter tuning. This contrasts with many other problems in music information retrieval like chord ID (Humphrey & Bello, 2012) and beat detection (Böck & Schedl, 2011), where data-driven methods have been shown to consistently outperform heuristic approaches. One possible explanation for this is that since fundamental frequency is a low-level physical attribute of an audio signal which is directly related to its periodicity, in many cases heuristics for estimating this periodicity perform extremely well with accuracies (measured in raw pitch accuracy, defined later on) close to 100%, leading some to consider the task a solved problem. This, however, is not always the case, and even top performing algorithms like pYIN can still produce noisy results for challenging audio recordings such as a sound of uncommon instruments or a pitch curve that fluctuates very fast. This is particularly problematic for tasks that require a flawless f0 estimation, such as using the output of a pitch tracker to generate reference annotations for melody and multi-f0 estimation (Salamon et al., 2017; Bittner et al., 2017).

In the following sections, a novel, data-driven method for monophonic pitch tracking based on a deep convolutional neural network operating on

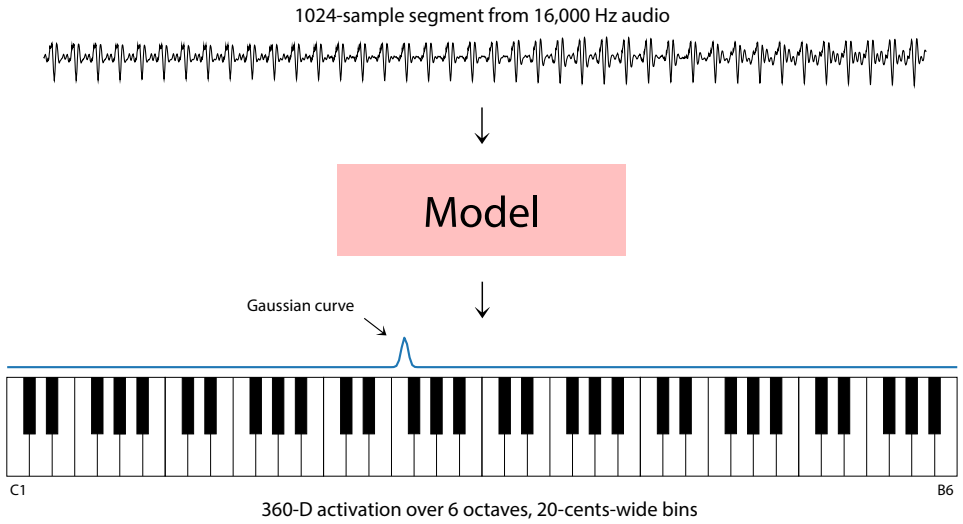

Figure 6: The input and output representations of the CREPE model. The input is 1024 time-domain samples, and the output is a 360-dimensional vector that contains a Gaussian curve centered at the ground-truth frequency (see Equation 16). For details on the convolutional architecture used in the model, see Figure 7.

the time-domain signal is presented. This approach, CREPE (Convolutional Representation for Pitch Estimation), obtains state-of-the-art results, outperforming heuristic approaches such as pYIN and SWIPE while being more robust to noise too. It is further shown that CREPE is highly precise, maintaining over 90% raw pitch accuracy even for a strict evaluation threshold of just 10 cents.

2 Architecture

CREPE consists of a deep convolutional neural network which operates directly on the time-domain audio signal to produce a pitch estimate. The input and output representations used in the model are illustrated in Figure 6. The input is a 1024-sample excerpt from the time-domain audio signal, using a 16 kHz sampling rate, and the output is a 360-dimensional vector $\hat{\mathbf{y}}$, from which the resulting pitch estimate is calculated deterministically.

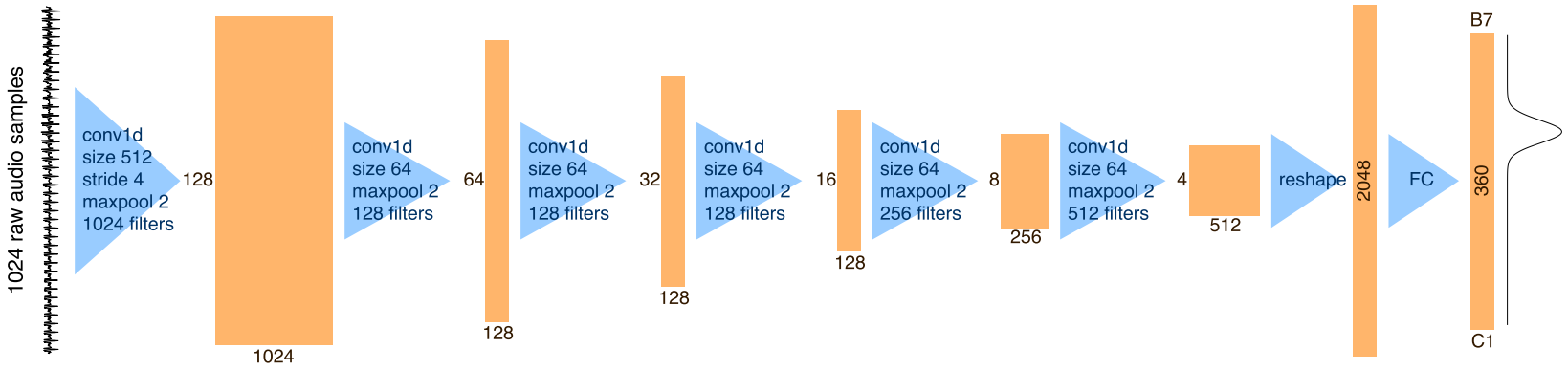

Figure 7: The architecture of the CREPE pitch tracker. The six convolutional layers predicts the Gaussian curve centered at the ground-truth frequency. The output is then used to extract the exact pitch estimate as in Equation 14-15.

Each of the 360 dimensions in the output corresponds to a specific pitch value, defined in cents. Cent is a unit representing musical intervals relative to a reference pitch f_{ref} in Hz, defined as a function of frequency f in Hz:

$$\text{¢}(f) = 1200 \cdot \log_2 \frac{f}{f_{\text{ref}}}, \tag{13}$$

where $f_{\text{ref}} = 10$ Hz throughout the experiments. This unit provides a logarithmic pitch scale where 100 cents equal one semitone. The 360 pitch values are denoted as $\text{¢}_1, \text{¢}_2, \cdots, \text{¢}_{360}$ and are selected so that they cover six octaves with 20-cent intervals between C1 and B6, corresponding to 32.70 Hz and 1975.5 Hz. The resulting pitch estimate $\hat{\text{¢}}$ is the weighted average of the associated pitches ¢_i according to the output $\hat{\mathbf{y}}$, which gives the frequency estimate in Hz:

$$\hat{\text{¢}} = \frac{\sum_{i=1}^{360} \hat{y}_i \text{¢}_i}{\sum_{i=1}^{360} \hat{y}_i}, \tag{14}$$

$$\hat{f} = f_{\text{ref}} \cdot 2^{\hat{\text{¢}}/1200}. \tag{15}$$

The target outputs we use to train the model are 360-dimensional vectors, where each dimension represents a frequency bin covering 20 cents (the same as the model's output). The bin corresponding to the ground truth fundamental frequency is given a magnitude of one. As in Bittner et al. (2017), in order to soften the penalty for near-correct predictions, the target is Gaussian-blurred in frequency such that the energy surrounding a ground truth frequency decays with a standard deviation of 25 cents:

$$y_i = \exp\left(-\frac{(\text{¢}_i - \text{¢}_{\text{true}})^2}{2 \cdot 25^2}\right), \tag{16}$$

This way, high activations in the last layer indicate that the input signal is likely to have a pitch that is close to the associated pitches of the nodes with high activations.

A detailed block diagram of the proposed architecture is provided in Figure 7. There are six convolutional layers that result in a 2048-dimensional latent representation, which is then connected densely to the output layer with sigmoid activations to produce the 360-dimensional target vector. The network is trained to minimize the binary cross entropy between the target vector $\mathbf{y}$ and the predicted vector $\hat{\mathbf{y}}$:

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^{360} \left(-y_i \log \hat{y}_i - (1 - y_i) \log(1 - \hat{y}_i)\right), \tag{17}$$

where both y_i and $\hat{y}_i$ are real numbers between 0 and 1. This loss function is optimized using the ADAM optimizer (Kingma & Ba, 2015), with the learning rate 0.0002. The best performing model is selected after training until the validation accuracy no longer improves for 32 epochs, where one epoch consists of 500 batches of 32 examples randomly selected from the training set. Each convolutional layer is preceded with batch normalization (Ioffe & Szegedy, 2015) and followed by a dropout layer (Srivastava et al., 2014) with the dropout probability 0.25. This architecture and the training procedures are implemented using Keras (Chollet, 2015).

3 Experiments

3.1 Datasets

In order to objectively evaluate CREPE and compare its performance to alternative algorithms, we require audio data with perfect ground truth annotations. This is especially important since the performance of the compared algorithms is already

very high. In light of this, we cannot use a dataset such as MedleyDB (Bittner et al., 2014), since its annotation process includes manual corrections which do not guarantee a 100% perfect match between the annotation and the audio, and it can be affected, to a degree, by human subjectivity. To guarantee a perfectly objective evaluation, we must use datasets of synthesized audio in which we have perfect control over the f0 of the resulting signal. We use two such datasets: the first, RWC-synth, contains 6.16 hours of audio synthesized from the RWC Music Database (Goto et al., 2002) and is used to evaluate pYIN in (Mauch & Dixon, 2014). It is important to note that the signals in this dataset were synthesized using a fixed sum of a small number of sinusoids, meaning that the dataset is highly homogenous in timbre and represents an over-simplified scenario. To evaluate the algorithms under more realistic (but still controlled) conditions, the second dataset we use is a collection of 230 monophonic stems taken from MedleyDB and re-synthesized using the methodology presented in (Salamon et al., 2017), which uses an analysis/synthesis approach to generate a synthesized track with a perfect f0 annotation that maintains the timbre and dynamics of the original track. This dataset consists of 230 tracks with 25 instruments, totaling 15.56 hours of audio, and henceforth referred to as MDB-stem-synth.

3.2 Methodology

We train the model using 5-fold cross-validation, using a 60/20/20 train, validation, and test split. For MDB-stem-synth, we use artist-conditional folds, in order to avoid training and testing on the same artist which can result in artificially high performance due to artist or album effects (Sturm, 2013). The evaluation of an algorithm's pitch estimation is measured in raw pitch accuracy (RPA) and raw

chroma accuracy (RCA) with 50 cent thresholds (Salamon et al., 2014). These metrics measure the proportion of frames in the output for which the output of the algorithm is within 50 cents (a quarter-tone) of the ground truth. We use the reference implementation provided in `mir_eval` (Raffel et al., 2014) to compute the evaluation metrics.

We compare CREPE against the current state of the art in monophonic pitch tracking, represented by the pYIN (Mauch & Dixon, 2014) and SWIPE (Camacho & Harris, 2008) algorithms. To examine the noise robustness of each algorithm, we also evaluate their pitch tracking performance on degraded versions of MDB-stem-synth, using the Audio Degradation Toolbox (ADT) (Mauch & Ewert, 2013). We use four different noise sources provided by the ADT: pub, white, pink, and brown. The pub noise is an actual recording of the sound in a crowded pub, and the white noise is a random signal with a constant power spectral density over all frequencies. The pink and brown noise have the highest power spectral density in low frequencies, and the densities fall off at 10 dB and 20 dB per decade respectively. Seven different signal-to-noise ratio (SNR) values are used: ∞, 40, 30, 20, 10, 5, and 0 dB.

3.3 Results

3.3.1 Pitch Accuracy

Table 1 shows the pitch estimation performance tested on the two datasets. On the RWC-synth dataset, CREPE yields a close-to-perfect performance where the error rate is lower than the baselines by more than an order of magnitude. While these high accuracy numbers are encouraging, those are achievable thanks to the highly homogeneous timbre of the dataset. In order to test the generalizability of

Dataset	Metric	CREPE	pYIN	SWIPE
RWC-synth	RPA	**0.999±0.002**	0.990±0.006	0.963±0.023
	RCA	**0.999±0.002**	0.990±0.006	0.966±0.020
MDB-stem-synth	RPA	**0.967±0.091**	0.919±0.129	0.925±0.116
	RCA	**0.970±0.084**	0.936±0.092	0.936±0.100

Table 1: Average raw pitch/chroma accuracies and their standard deviations, tested with the 50 cents threshold.

Dataset	Threshold	CREPE	pYIN	SWIPE
RWC-synth	50 cents	**0.999±0.002**	0.990±0.006	0.963±0.023
	25 cents	**0.999±0.003**	0.972±0.012	0.949±0.026
	10 cents	**0.995±0.004**	0.908±0.032	0.833±0.055
MDB-stem-synth	50 cents	**0.967±0.091**	0.919±0.129	0.925±0.116
	25 cents	**0.953±0.103**	0.890±0.134	0.897±0.127
	10 cents	**0.909±0.126**	0.826±0.150	0.816±0.165

Table 2: Average raw pitch accuracies and their standard deviations, with different evaluation thresholds.

the algorithms on a more timbrally diverse dataset, the performance is evaluated on the MDB-stem-synth dataset as well. It is notable that the degradation of performance from RWC-synth is more significant for the baseline algorithms, implying that CREPE is more robust to complex timbres compared to pYIN and SWIPE.

Finally, to see how the algorithms compare under scenarios where any deviation in the estimated pitch from the true value could be detrimental, Table 2 reports the RPA at lower evaluation tolerance thresholds of 10 and 25 cents as well as the RPA at the standard 50 cents threshold for reference. The table indicates that as the threshold is decreased, the difference in performance becomes more

accentuated, with CREPE outperforming by over 8 percentage points when the evaluation tolerance is lowered to 10 cents. The high accuracy in 10 cents despite the 20-cent resolution of the model output indicates that the weighted average solution in Equation 14 is effective at predicting the precise frequencies even between the adjacent frequency bins. This suggests that CREPE is especially preferable when even minor deviations from the true pitch should be avoided as best as possible. Obtaining highly precise pitch annotations is perceptually meaningful for transcription and analysis/resynthesis applications.

3.3.2 Noise Robustness

Noise robustness is key to many applications like speech analysis for mobile phones or smart speakers, or for live music performance. Figure 8 shows how the pitch estimation performance is affected when an additive noise is present in the input signal. CREPE maintains the highest accuracy for all SNR levels for pub noise and white noise, and for all SNR levels except for the highest level of pink noise. Brown noise is the exception where pYIN's performance is almost unaffected by the noise. This can be attributed to the fact that brown noise has most of its energy at low frequencies, to which the YIN algorithm (on which pYIN is based) is particularly robust.

To summarize, CREPE performs better in all cases where the SNR is below 10 dB while the performance varies depending on the spectral properties of the noise when the noise level is higher, which indicates that this approach can be reliable under a reasonable amount of additive noise. CREPE is also more stable, exhibiting consistently lower variance in performance compared to the baseline algorithms.

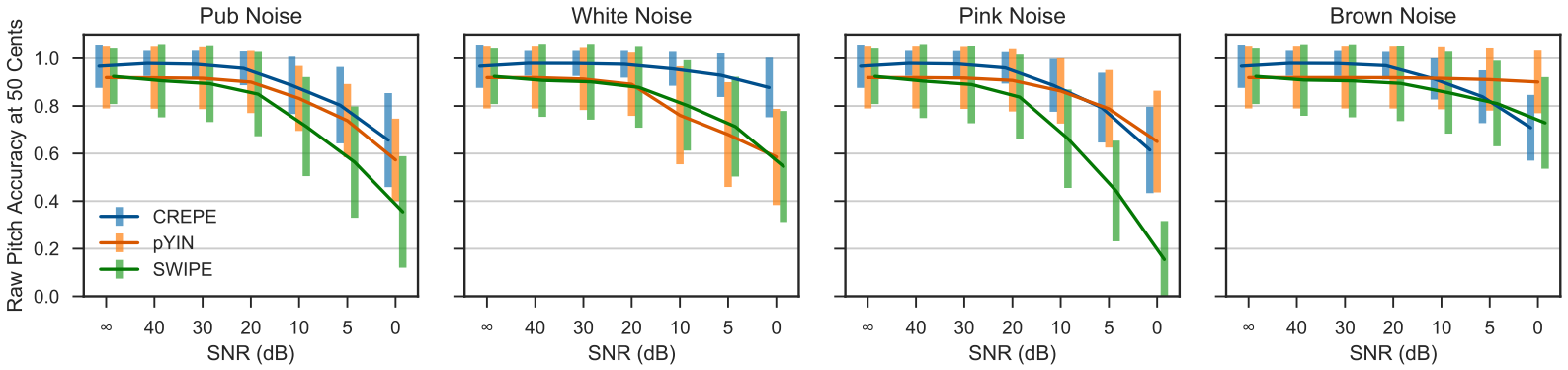

Figure 8: Pitch tracking performance when additive noise signals are present. The error bars are centered at the average raw pitch accuracies and span the first standard deviations. With brown noise being a notable exception, CREPE shows the highest noise robustness in general.

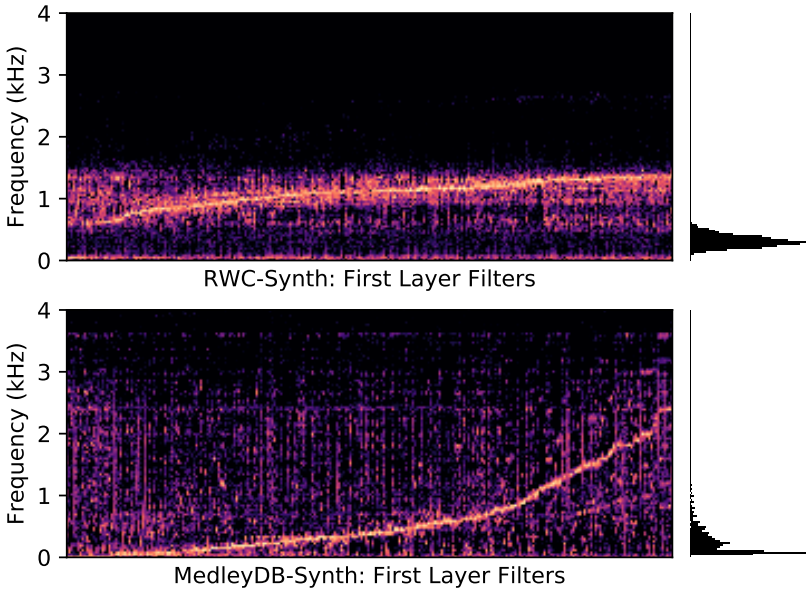

Figure 9: Fourier spectra of the first-layer filters sorted by the frequency of the peak magnitude. Histograms on the right show the distribution of ground-truth frequencies in the corresponding dataset.

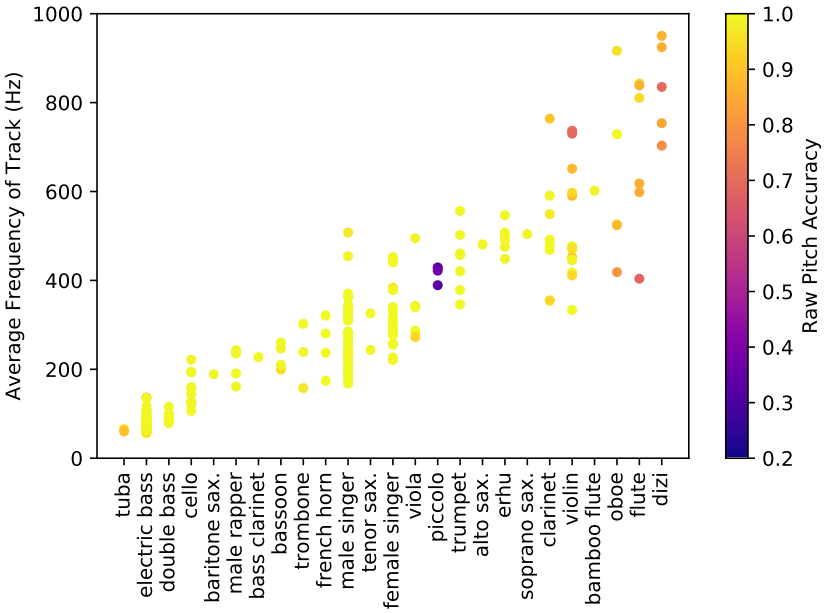

Figure 10: The raw pitch accuracy (RPA) of CREPE's predictions on each of the 230 tracks in MDB-stem-synth with respect to the instrument, sorted by the average frequency.

3.3.3 Model Analysis

To gain some insight into the CREPE model, we visualize in Figure 9 the spectra of the 1024 convolutional filters in the first layer of the neural network, with histograms of the ground-truth frequencies to the right of each plot. It is noticeable that the filters learned from the RWC-synth dataset have the spectral density concentrated between 600 Hz and 1500 Hz, while the ground-truth frequencies are mostly between 100 Hz and 600 Hz. This indicates that the first convolutional layer in the model learns to distinguish the frequencies of the overtones rather than the fundamental frequency. These filters focusing on overtones are also visible for MDB-stem-synth, where peak frequencies of the filters range well above the f0 distribution of the dataset, but in this case, the majority of the filters overlap with the ground-truth distribution, unlike RWC-synth. A possible explanation for this is that since the timbre in RWC-synth is fixed and identical for all tracks, the model is able to obtain a highly accurate estimate of the f0 by modeling its harmonics. Conversely, when the timbre is heterogeneous and more complex, as is the case for MDB-stem-synth, the model cannot rely solely on the harmonic structure and requires filters that capture the f0 periodicity directly in addition to the harmonics. In both cases, this suggests that the neural network can adapt to the distribution of timbre and frequency in the dataset of interest, which in turn contributes to the higher performance of CREPE compared to the baseline algorithms.

3.3.4 Performance by Instrument

The MDB-stem-synth dataset contains 230 tracks from 25 different instruments, where electric bass (58 tracks) and male singer (41 tracks) are the most common while there are instruments that occur in only one or two tracks. Figure 10

shows the performance of CREPE on each of the 230 tracks, with respect to the instrument of each track. It is notable that the model performs worse for the instruments with higher average frequencies, but the performance is also dependent on the timbre. CREPE performs particularly worse on the tracks with the dizi, a Chinese transverse flute, because the tracks came from the same artist, and they are all placed in the same split. This means that for the fold in which the dizi tracks are in the test set, the training and validation sets do not contain a single dizi track, and the model fails to generalize to this previously unseen timbre. There are 5 instruments (bass clarinet, bamboo flute, and the family of saxophones) that occur only once in the dataset, but their performance is decent, because their timbres do not deviate too far from other instruments in the dataset. For the flute and the violin, although there are many tracks with the same instrument in the training set, the performance is low when the sound in the tested tracks is too low (flute) or too high (violin) compared to other tracks of the same instruments. The low performance on the piccolo tracks is due to an error in the dataset where the annotation is inconsistent with the correct pitch range of the instrument. Unsurprisingly, the model performs well on test tracks whose timbre and frequency range are well-represented in the training set.

4 Open-Sourcing CREPE

Releasing an implementation of research work as an open-source software helps ensure better quality, reproducibility, and longevity of the research code (McFee et al., 2019). To facilitate easier adoption of CREPE's pitch tracking functionality for wider audience, we open-sourced[1] the implementation of CREPE under the

[1] `https://github.com/marl/crepe`

MIT License. The main purpose of the open-source release is to make the usage of the CREPE software as easy and effective as possible, so we aimed to ensure that the pitch estimation can work accurately for the broad range of the real-world monophonic signals. As shown in Figure 9, a model trained on a specific dataset specializes to the characteristics of the sounds in the dataset and will fail to work well across the previously unknown, real-world inputs. To address this problem, we used six different datasets for the pre-trained model included in the open-source release of CREPE: MIR-1K (Hsu & Jang, 2010), Bach10 (Duan et al., 2010), RWC-Synth (Mauch & Dixon, 2014), MedleyDB (Bittner et al., 2014), MDB-STEM-Synth (Salamon et al., 2017), and NSynth (Engel et al., 2017). These datasets contain various instrumental and vocal sounds, and the pre-trained model is expected to work well on these types of sounds. A visualization of the first-layer filters for the model trained with these datasets are shown in Figure 11. Compared to Figure 9, the peak-frequency curve better covers the full range of frequencies and in a smoother manner. An interesting feature of this visualization is that the curve is somewhat linear until 1 kHz and rapidly grows for the higher frequencies. This roughly coincides with the Mel scale, where the mapping from the perceived frequency to the actual frequency is linear below 1 kHz and logarithmic above (Logan, 2000).

4.1 Python Package and Command-Line Interface

CREPE is distributed as a Python package and is hosted on the Python Package Index (PyPI). It can be installed simply by running the following command:

```
$ pip install crepe
```

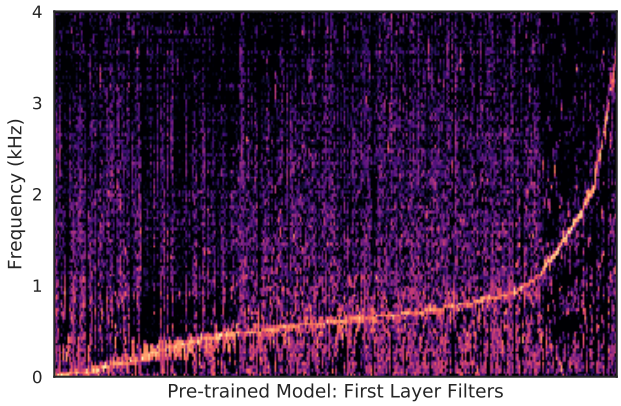

Figure 11: The first layer filters of the CREPE model trained with six different datasets, visualized using the same method as in Figure 9. Compared to the previous cases where the filters are specialized to one dataset, the peak-frequency curve is smoother and covers the full frequency range.

in a Python environment. This command will install the Python package `crepe` usable in Python scripts as well as a command-line interface that can be called with one or more audio file names:

```
$ crepe audio_file.wav
```

The estimated pitch values are saved using the CSV format, like:

```
time,frequency,confidence
0.00,185.616,0.907112
```

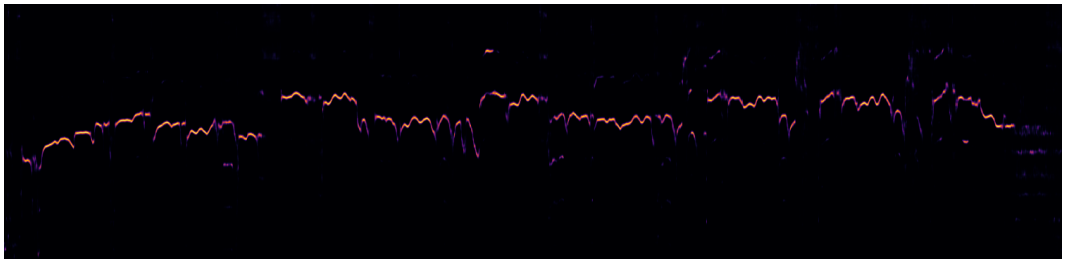

Figure 12: Visualization of the target activation, which can be saved as in image file by using the `--save-activation` option in the CREPE command-line interface. The audio clip used contains an excerpt of male singing voice.

```
0.01,186.764,0.844488
0.02,188.356,0.798015
0.03,190.610,0.746729
0.04,192.952,0.771268
0.05,195.191,0.859440
0.06,196.541,0.864447
0.07,197.809,0.827441
0.08,199.678,0.775208
...
```

where the third column indicates the confidence level of the presence of a pitch, which is obtained by taking the maximum activation value at each frame.

The command-line interface supports various options, such as to use Viterbi smoothing (Viterbi, 1967) of the pitch curves as in pYIN (Mauch & Dixon, 2014) and to save the activation in an image file as shown in Figure 12. The package contains five pre-trained models with different capacities, where the number of convolutional channels are scaled to 12.5% (tiny), 25% (small), 50% (medium), 75% (large), or 100% (full) relative to the model size used in the experiments (see Figure 7), and the specific model capacity to use can be specified using the `--model-capacity` option. Smaller-capacity models run significantly faster than the full-capacity model, suitable for resource-constrained use cases where faster computation is preferred at the cost of slightly lower pitch estimation accuracy.

4.2 Real-Time Web Demo

In addition to the Python package, we have developed a real-time web demo[1] where the CREPE model runs on the web browser using the audio recorded from the system microphone and displays the target activation as well as the predicted pitch value in real time. To enable this, the demo used the W3C Web Audio API for real-time audio input and TensorFlow.js (Smilkov et al., 2019) for running inference on the deep model. Under the hood, TensorFlow.js uses WebGL which allows GPU acceleration for faster computation of the convolutional layers. The web demo uses the "tiny" capacity model in order to run faster in browsers, which uses less than 3% of the parameters compared to the full model. The web demo is also open-source and distributed in the same Github repository as the Python package.

4.3 Argmax-Local Weighted Averaging

In the process of training the model with diverse datasets, we have observed a side-effect where the model produces more octave errors, i.e. the predicted pitch is an octave apart from the ground truth. These errors resulted in multiple peaks in the target output vector around the harmonics of the ground-truth frequency, as depicted in Figure 13. When the weighted average over the all 360 bins is used, this ends up predicting a completely different pitch. To alleviate this, we have revised Equation 14 in the open-source version of CREPE to calculate the weighted average

[1] `https://marl.github.io/crepe`

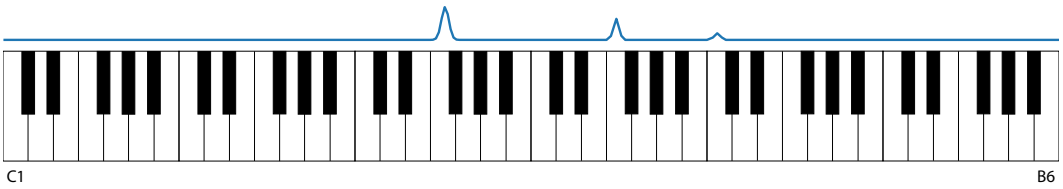

Figure 13: An example of an inaccurate target output where there exist multiple peaks around the harmonics of the ground-truth frequency. Equation 18 ensures that the predicted pitch is calculated using only the frequency bins around the highest peak.

using only the frequency bins around the highest peak in the output:

$$\hat{c} = \frac{\sum_{i=m-4}^{m+4} \hat{y}_i c_i}{\sum_{i=m-4}^{m+4} \hat{y}_i}, \qquad m = \arg\max_i \hat{y}_i. \tag{18}$$

Using Equation 18 improved the transcription accuracies of real-world signals significantly. However, a quantitative evaluation of the robustness in the real-world situations is inherently difficult and is out of the scope of this chapter, because it is virtually impossible to obtain a representative and objective dataset encompassing all such situations.

4.4 Data-Driven Models and Real-World Applications

The series of modifications on the CREPE model are taken in order to release the open-source software that are effective not only for evaluation in an academic setting but also in various real-world scenarios of pitch estimation. These give an important lesson that should be considered when developing data-driven models for music analysis tasks. Especially with the rise of deep learning models that typically have a large number of parameters, it is very easy to overfit a model to work well with a specific dataset while not being able to generalize. This problem can persist even if the dataset is properly splitted for training and evaluation,

because it is common for such dataset to have shared characteristics such as recording and mixing conditions that are not particularly representative of the broad range of the real-world data for which the model should supposedly work well. Therefore, a great amount of care should be taken when selecting the datasets to be used for evaluation of a music analysis model, and qualitative analyses on out-of-distribution real-world data need to be accompanied.

5 Conclusions

In this chapter, we presented a novel data-driven method for monophonic pitch tracking based on a deep convolutional neural network operating on time-domain input, CREPE. We showed that CREPE obtains state-of-the-art results, outperforming pYIN and SWIPE on two datasets with homogeneous and heterogeneous timbre respectively. Furthermore, we showed that CREPE remains highly accurate even at a very strict evaluation threshold of just 10 cents. We also showed that in most cases CREPE is more robust to added noise. In addition, we discussed the challenges that arose during the open-source release of the model, which included how to make the model more generalizable to the real-world sounds and the revised pitch extraction strategy to better deal with increased octave errors.

Ideally, we want the model to be invariant to all transformations that do not affect pitch, such as changes due to distortion and reverberation. Some invariance can be induced by the architectural design of the model, such as the translation invariance induced by pooling layers in our model as well as in deep image classification models. However, it is not as straightforward to design the model architecture to specifically ignore other pitch-preserving transformations.

While it is still an intriguing problem to build an architecture to achieve this, we could use data augmentation to generate transformed and degraded inputs that can effectively make the model learn the invariance. The robustness of the model could also be improved by applying pitch-shifts as data augmentation (McFee et al., 2015) to cover a wider pitch range for every instrument. In addition to data augmentation, various sources of audio timbre can be obtained from software instruments; NSynth (Engel et al., 2017) is an example where the training dataset is generated from the sound of software instruments.

Pitch values tend to be continuous over time, but CREPE estimates the pitch of every frame independently without using any temporal tracking, unlike pYIN which exploits this by using an HMM to enforce temporal smoothness. We can potentially improve the performance of CREPE further by statistically modeling how the pitch value changes over time. Employing recurrent neural networks (RNNs) is a natural choice for this, and we will explore in the following chapters how RNNs can model the temporal characteristics of music, focusing on two specific tasks: music synthesis (Chapter V) and piano transcription (Chapter VI).