Introducing EG-IPT and ipt~: a novel electric guitar dataset and a new Max/MSP object for real-time classification of instrumental playing techniques

Marco Fiorini*
IRCAM - STMS
CNRS UMR 9912 - Sorbonne Université
Paris, France
marco.fiorini@ircam.fr

Nicolas Brochec*
Tokyo University of the Arts
Tokyo, Japan
nicolas.brochec@pm.me

Joakim Borg
IRCAM - STMS
CNRS UMR 9912
Paris, France
joakim.borg@ircam.fr

Riccardo Pasini
UNIFE Università di Ferrara
Dipartimento di Matematica e Informatica
Ferrara, Italy
riccardo01.pasini@edu.unife.it

Abstract

This paper presents two key contributions to the real-time classification of Instrumental Playing Techniques (IPTs) in the context of NIME and human-machine interactive systems: the EG-IPT dataset and the ipt∼ Max/MSP object. The EG-IPT dataset, specifically designed for electric guitar, encompasses a broad range of IPTs captured across six distinct audio sources (five microphones and one direct input) and three pickup configurations. This diversity in recording conditions provides a robust foundation for training accurate models. We evaluate the dataset by employing a Convolutional Neural Network-based classifier (CNN), achieving state-of-the-art performance across a wide array of IPT classes, thereby validating the dataset's efficacy. The ipt∼ object is a new Max/MSP external enabling real-time classification of IPTs via pre-trained CNN models. While in this paper it's demonstrated with the EG-IPT dataset, the ipt∼ object is adaptable to models trained on various instruments. By integrating EG-IPT and ipt∼, we introduce a novel, end-to-end workflow that spans from data collection, model training to real-time classification and human-computer interaction. This workflow exemplifies the entanglement of diverse components (data acquisition, machine learning, real-time processing, and interactive control) within a unified system, advancing the potential for dynamic, real-time music performance and human-computer interaction in the context of NIME.

Keywords

Instrumental Playing Techniques, Electric Guitar, Music Classification, Real-Time, Music AI, Python, Max/MSP, NIME

1 Introduction

Instrumental Playing Technique (IPT) recognition is an emerging area of significance for New Interfaces for Musical Expression (NIME), particularly in interactive systems [31] that facilitate dynamic human-computer interaction [19]. While it has its roots in Music Information Retrieval (MIR) [20], its applications in real-time, expressive musical interfaces offer new creative opportunities for performers and researchers alike.

Several tools have been developed to address IPT recognition within NIME scenarios, including FluCoMa [38], MuBu [32], and the PRiSM Music Gesture Recognition toolkit[1], which is used in Forager[2]. The latter is an extension of the seminal Voyager [16] system, now incorporating gesture recognition. These tools, all compatible with Max/MSP, provide flexible frameworks for real-time signal processing and prototyping. However, they are limited in their ability to train and deploy deep learning models directly in Max/MSP, and evaluating their performance for accuracy and generalization across diverse datasets and users remains a significant challenge. A notable breakthrough in this domain was the introduction of the RAVE generative model [6] and the nn∼ Max/MSP object[3]. This approach enables deep learning models trained in Python to be then integrated into Max/MSP for real-time applications. Although initially designed for timbre transfer, this workflow has been adopted in various NIME applications [29, 33, 39] and recently extended to IPT recognition [35]. Despite these advancements, existing implementations often focus on a limited range of playing techniques and lack systematic evaluation for generalization to new datasets or users.

In response, we propose a novel workflow specifically designed for real-time IPT recognition in interactive systems. Building on the RAVE/nn∼ paradigm, our approach integrates a Python-based deep learning training phase with a dedicated Max/MSP object, ipt∼, optimized for this task. This workflow prioritizes usability, robustness, and generalization, making it suitable for both research and performance contexts.

The contributions of this paper are as follows: (I) we introduce a new dataset, EG-IPT, specifically designed for electric guitar IPT recognition, addressing both MIR and NIME research needs; (II) we present the ipt∼ Max/MSP object for real-time IPT recognition in interactive systems, leveraging deep learning models; and (III) we propose a novel workflow that, while inspired by the RAVE/nn∼ framework, offers a new trajectory focused on practical utility in interactive performance contexts with an emphasis on robust generalization. As for the paper structure, section 2 reviews related work on IPT recognition, surveys existing guitar

*These two authors contributed equally to this paper.

NIME '25, June 24–27, 2025, Canberra, Australia

[1] https://github.com/rncm-prism/PRiSM-MusicGestureRecognition
[2] https://www.youtube.com/watch?v=AiveFGVSWSU
[3] https://github.com/acids-ircam/nn_tilde

datasets, and introduces our proposed workflow. Section 3 details the EG-IPT dataset. Section 4 describes the implementation of the ipt~ object in Max/MSP. Section 5 outlines the evaluation tests we conducted to validate our dataset. Section 6 discusses the results, and Section 7 concludes the paper.

2 Related Works

2.1 Instrumental Playing Technique Recognition

Various approaches have been tried to recognize Instrumental Playing Techniques (IPTs) from audio data. For Western instruments, studies mainly focused on acoustic Guitar [21, 26, 30, 37], piano [5, 18], violin [36], cello [9], and flute [3, 4, 10]. Eastern instrument playing techniques recognition include studies on guqin [14], a plucked seven-string Chinese instrument, guzheng [17], a plucked twenty-one-string Chinese instrument and Chinese bamboo flute [40]. A few recent systems address the real-time recognition of IPTs for real-time creative purposes, notably studies on the cello [9], acoustic guitar [21], and flute [10]. Only the study on the flute achieved promising results in classifying IPTs using training and test data from different performers, with an accuracy of 92.56% across seven classes. This success is attributed to a technique involving the simultaneous recording of IPTs with multiple microphones for the recording of IPT samples [4]. This approach captures more audio without extending recording time and improves accuracy by incorporating audio files from various microphones into the training dataset.

2.2 Guitar Datasets

The guitar is central to Music Information Retrieval (MIR) and New Interfaces for Musical Expression (NIME). As data-driven methods advance, the demand for high-quality, diverse datasets grows. However, many existing datasets lack comprehensive IPT coverage, diverse microphone and pickup setups, and real-time applicability. Below, we review notable guitar datasets:

- Guitar Playing Techniques [37] (2014): 6580 clips with 11928 notes annotated for seven techniques. Access is currently unavailable.
- IDMT-SMT-Guitar [15] (2014): 5100 note events across monophonic and polyphonic recordings on electric and acoustic guitars.
- Guitar Solo Detection [27] (2017): Focused on solo detection, with limited details on annotations.
- GuitarSet [42] (2018): 360 acoustic guitar excerpts, designed for transcription tasks, but lacks electric guitar effects.
- EGDB [7] (2022): 240 electric guitar tone renditions, without real-time or hardware effects.
- EGFxSet [28] (2022): 8970 recordings processed with real effects, suited for physical effect emulation tasks.
- EG-Solo [13] (2023): 6833 annotated note events for electric guitar solos, with a focus on techniques like palm muting and harmonics, but sourced from YouTube.
- Guitar Style [23] (2024): 549 video samples of nine techniques, constrained by a single performer and amplifier simulations.
- AG-PT [34] (2024): Over 10 hours of acoustic guitar recordings with eight techniques, but lacks electric guitar diversity.

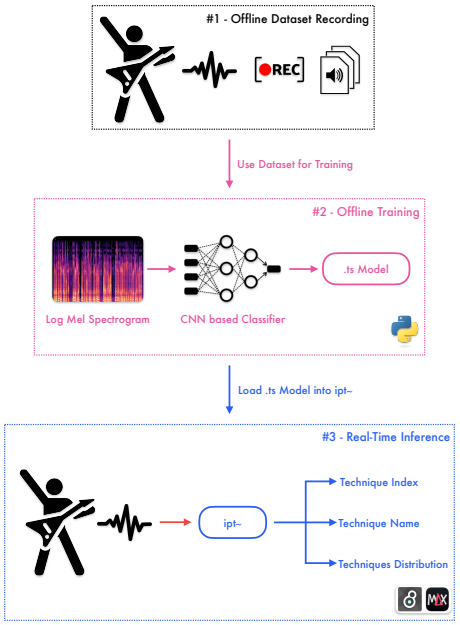

Figure 1: Diagram illustrating our end-to-end proposed workflow: (1) Record the dataset, (2) Train the CNN-based classifier in Python, and (3) Load the resulting .ts model into the ipt~ object in Max/MSP for real-time inference.

While these datasets provide valuable resources, they often lack comprehensive coverage of guitar techniques, diverse recording setups, and real-time applicability. Our dataset addresses these gaps by offering a wider range of IPTs, diverse recording conditions, and compatibility with real-time systems, particularly for NIME applications.

2.3 Proposed Workflow

Our proposed workflow for real-time IPT classification is outlined in Figure 1 and consists of three main stages:

(1) **Dataset Recording**: Audio samples are recorded using six audio sources (five microphones, one direct input) and three pickup configurations, encompassing 19 techniques. Details are in Section 3.
(2) **Model Training**: A CNN-based classifier is trained on the dataset using log mel spectrograms. Section 4 provides an overview of the training process and model evaluation.
(3) **Real-Time Inference**: The trained model is deployed in Max/MSP using the `ipt_tilde` object, enabling real-time classification of live audio input. The model, exported in TorchScript format, is detailed in Section 5.

This workflow integrates data collection, model training, and real-time interaction, offering a scalable solution for IPT classification. Its versatility makes it ideal for NIME systems, where real-time accuracy is crucial.

3 The EG-IPT Dataset

The Electric Guitar Instrumental Playing Techniques (EG-IPT) dataset [11] comprises 52320 audio files totaling 28 hours, 22 minutes, and 56 seconds of recordings, with a size of 29.77 GB. It includes 19 distinct playing techniques, capturing individual electric guitar sounds performed by a professional guitarist and

recorded by a sound engineer in a specialized studio using high-quality equipment. The dataset reflects rigorous standards, offering a diverse and comprehensive resource for studying IPTs and it's available on Zenodo[4].

3.1 Recording

The recording was made using a 2005 Gibson SG Standard electric guitar equipped with two humbucker pickups, offering three selectable positions: bridge humbucker, both pickups coupled, and neck humbucker. An EVH 5150 III 50W 6L6 tube amplifier head was used, set with all controls in a flat position and no reverb applied. The amp was paired with a Mesa 4x12" cabinet with Celestion V30 UK speakers. The recording took place in a studio room measuring 4.5 meters in width, 4.5 meters in depth, and 3 meters in height. The audio was captured at a resolution of 96kHz and 24-bit using Pro Tools Ultimate.

Following the methodology of microphone-based data augmentation [4], the dataset was recorded using 6 distinct recording sources, as detailed in Table 1. These include 5 microphones and 1 direct input line (DI box). For close-miking, a Shure SM57 dynamic microphone was paired with a FetHead and Tk Audio DP1 preamp, positioned at a distance of 2.5 cm. An AKG C414 condenser microphone, connected to a Tk Audio DP1, was also placed at 2.5 cm for additional tonal variation. A Peluso R14 ribbon microphone was used at a mid-distance of 24 cm with a Soyuz The Launcher and a Tk Audio DP1 preamp, capturing a warmer, more ambient tone. An Audio Technica 350 was placed 190 cm away in a bucket on the wall, while an Audio Technica 4050 was set at 280 cm in an omnidirectional pattern for room ambiance. Lastly, a DI Box BSS Audio AR 133 through a Midas XL48 preamp was used to record the direct guitar signal.

3.2 Electric Guitar Playing Techniques

The EG-IPT dataset includes the following techniques:

(1) *ordinario*: Playing notes with a natural, sustained tone, allowing the string to vibrate without additional effects.
(2) *staccato*: Producing short, detached notes by muting the string immediately after plucking.
(3) *muted*: Dampening the strings near the bridge with the palm, generating a percussive tone.
(4) *vibrato*: Modulating pitch by rapidly bending and releasing the string.
(5) *harmonics*: Producing tones by lightly touching string nodes, including natural and artificial harmonics.
(6) *glissando*: Sliding along the string for a continuous pitch transition.
(7) *slide*: Moving the fretting hand along the string to shift between pitches, as indicated (e.g., "1st," "1t").
(8) *bend*: Raising pitch by bending strings, including whole-tone, half-tone, or pre-bend variations.
(9) *hammer-on*: Transitioning to a higher pitch by pressing the string without re-plucking.
(10) *pull-off*: Releasing a fretted note to sound a lower pitch on the same string.
(11) *tremolo*: Rapidly repeating a single note for a sustained, continuous effect.
(12) *trill*: Alternating between two adjacent pitches in quick succession.
(13) *snap-pizz*: Plucking the string forcefully to make it snap against the fretboard.
(14) *ebow*: Using an electronic bow to create sustained string vibrations.
(15) *arco*: Bowing the strings to produce sustained tones or harmonics.
(16) *palmstrike*: Striking the strings or body of the guitar with the palm to create rhythmic impacts.
(17) *scratch*: Dragging fingers or objects along the strings to generate non-pitched, frictional sounds.
(18) *bottleneck*: Using a slide on the string to create continuous pitch transitions.
(19) *behind-nut*: Plucking or pressing the string behind the nut to produce short, high-pitched tones.

3.3 Dataset Structure

The EG-IPT dataset is organized hierarchically to facilitate efficient navigation and use in machine learning applications. Its structure, depicted in Figure 2, reflects the combinations of pickup configurations, recording sources, and instrumental playing techniques included in the dataset.

At the root level, the dataset is divided into folders representing the three pickup configurations: *HB-bridge*, *HB-couple*, and *HB-neck*, corresponding to the bridge humbucker, combined pickup setting, and neck humbucker of the guitar, respectively. Each configuration folder contains subfolders for six distinct recording sources: five microphones and one direct input (DI). These sources, detailed in Section 3.1 and in Table 1, include close-miking setups (e.g., dynamic and condenser microphones), mid-distance and room captures, as well as the DI signal, ensuring a comprehensive range of tonal characteristics. Within each recording source, subfolders represent the 19 instrumental playing techniques included in the dataset. These techniques, outlined in Section 3.2, span a wide variety of articulations, including common methods such as *ordinario*, *bend*, and *hammer-on*, as well as more specialized techniques like *ebow*, *bottleneck*, and *behind-nut*. Each recording is labeled with its pickup configuration, source, technique, string played, and distance/width (for techniques involving two notes, such as *bend*, *hammer-on*, *pull-off*, *slide*, and *trill*), along with a unique identifier for the note played. These labels facilitate precise categorization, enabling effective use in training and evaluating machine learning models.

This hierarchical organization provides a structured and extensible framework for the dataset, supporting both detailed exploration of specific techniques and broader studies of electric guitar performance. The inclusion of placeholders (...) in the tree diagram in Figure 2 illustrates the potential for extending the dataset with additional techniques or recording setups in future research.

4 Evaluation of EG-IPT

The code for all the evaluations discussed in this paper is available on GitHub[5] while a video demo of ipt~ detecting techniques from the EG-IPT dataset can be found on YouTube[6].

4.1 Proposed Methodology

4.1.1 Datasets. After preliminary tests, we conducted three experimental setups to evaluate our dataset and training architecture. Our initial hypothesis was that the multi-source recordings in the EG-IPT dataset would improve training accuracy compared to using only DI signals, as shown in a previous study

[4]https://doi.org/10.5281/zenodo.15205644

[5]https://github.com/nbrochec/nime2025.git
[6]https://youtu.be/PFiWNnOd-vg

Table 1: Microphones, preamps and specifications used to record the dataset

Name	Microphone	Preamp	Distance
dyn	Shure SM57	FetHead + Tk Audio DP1	2.5 cm
cond	AKG C414	Tk Audio DP1	2.5 cm
rib	Peluso R14	Soyuz The Launcher + Tk Audio DP1	24 cm
bucket	Audio Technica 350	Tk Audio DP1	190 cm, in a bucket on the wall
room	Audio Technica 4050	Tk Audio DP1	280 cm, omnidirectional pattern
DI	DI Box BSS Audio AR 133	Midas XL48	-

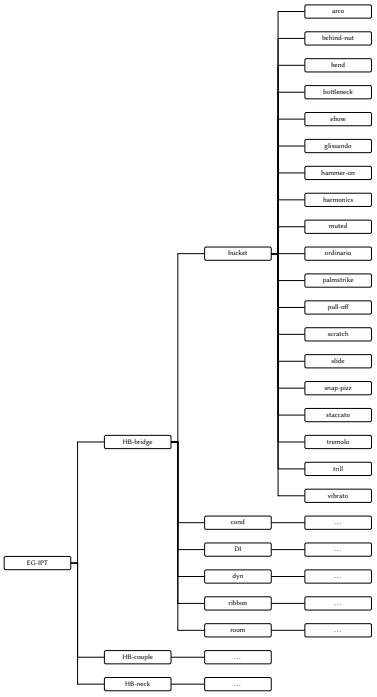

Figure 2: Tree diagram of the folder structure of the EG-IPT dataset

about the flute [4]. Additionally, we aimed to assess the extent to which our architecture could generalize by validating the dataset against a different electric guitar dataset. Out test configurations include: setup A, using only DI signals from different pickups for training, validation, and testing (see Table 2); setup B, training on full multi-source recordings from a single pickup, validated and tested with DI signals from other pickups; and setup C, training on an all-pickups DI set, validated and tested with another novel dataset we recorded for the experiment, the Strandberg Dataset.

Table 2: Overview of the datasets used in the study, detailing the training, validation, and test sets, along with the number of IPTs for each configuration setup.

Setup	Training Set	Validation Set	Test Set	n IPTs
A	HB neck DI	HB bridge DI	HB couple DI	14
B	HB neck multisource	HB bridge DI	HB couple DI	14
C	HB all pickups DI	20% Test Set	Strandberg DI	10

For evaluations A and B, we reduced the techniques to 14: *behind-nut*, *bend*, *bottleneck*, *glissando*, *legato*, *muted*, *palmstrike*, *scratch*, *snap-pizz*, *staccato*, *sustained*, *tremolo*, *trill*, and *vibrato*. This restructuring simplifies the taxonomy by grouping perceptually similar techniques. Considering that *ordinario* and *harmonics* are perceptually similar, we grouped them into the *sustained* category. We applied the same methodology to *hammer-on*, *pull-off*, and *slide* that we grouped into *legato* category. Finally, *arco* and *ebow* were excluded due to potential challenges for guitarists unfamiliar with these tools.

Evaluation setups A and B create semi-heterogeneous evaluation conditions, as recordings from the same guitarist and guitar differ due to varying pickup configurations and sources. To assess the generalization of our dataset and model under fully-heterogeneous conditions, we recorded the Strandberg Dataset (C Test Set, as referenced in Table 2). This dataset comprises 1038 audio files, totaling 35.58 minutes with a size of 310.3 MB, covering 16 techniques. Recorded by a different guitarist using a Strandberg Boden Prog 7 strings guitar with two Fishman Fluence Modern pickups, directly connected to a Scarlett Focusrite 2i2 interface, and a 2023 MacBook Pro with M3 Max processor, it simulates real-life conditions without high-end studio equipment. Unlike EG-IPT with 19 techniques, this dataset omits *behind-nut* (due to the lack of a headstock on the guitar), *arco* and *ebow* (due to guitarist's inexperience with these tools), focusing on one pickup (bridge). Given the adjustments in IPT classes for evaluations A and B, we then aligned the classes for evaluation configuration C with the Strandberg test set, resulting in 10 classes: *bend*, *glissando*, *legato*, *muted*, *palmstrike*, *staccato*, *sustained*, *tremolo*, *trill*, and *vibrato*.

4.1.2 Preprocessing. Our preliminary tests showed that downsampling the audio files to 8 kHz improved performance with our architecture and dataset setups. We removed the silence from the audio files because it is irrelevant, using the trim function from the librosa library [22] with `top_db=-60`. We sliced the audio files into adjacent sequences of ≈ 900ms, considering long-duration playing techniques from EG-IPTs such as the *bend*, *glissando*, or *legato*, which cannot be fully comprehended with a shorter temporal window.

Data augmentation is a typical technique in machine learning. It allows for mitigating overfitting by increasing the amount of data and data variability. We applied several offline data augmentations after slicing audio files. We generated new training data by separately applying three different audio transformations to the original data. We employed detuning, noise addition, and time stretching inspired by the performance conditions. We detuned the audio files with a range of ±100Hz around 440Hz the tuning frequency, considering that the electric guitar's tuning may vary across performances. We added noise, considering that amplifying the electric guitar signal may induce noise. We time-stretched the audio files with a random ratio between 0.9 and 1.1,

considering that the playing techniques can be performed at different speeds. By applying these data augmentation techniques, we multiplied our training data by a factor of 4.

Our datasets are unbalanced due to the difference in time duration of each class. Indeed, long-duration playing techniques provide more training samples, e.g., *sustained* is a longer playing technique than *staccato*. To alleviate that concern, we use a balanced batch sampler[7]. It balances batches by ensuring the same number of data samples per class is used.

4.1.3 Model Architecture. Recent studies show that models based on convolutional neural networks are adapted to recognize IPTs in real-time [8, 9, 21]. Building on a previously developed model architecture [4, 10], we augmented the number of parameters of the model by increasing the output channels of convolutional layers. To build our architecture, we defined four custom blocks, LMS, Conv2d, MaxPooling, and Linear blocks, as shown in Figure 3. Blocks are assembled to form the entire architecture, as shown in Figure 4. Kernels of the first four Conv2d blocks are set to `2x3`, and the remaining are set to `2x2`. The block LMS processes inputted raw audio and output a normalized Log-Mel-Spectrogram, using `n_fft=2048`, `n_mels=128`, `hop_length=512` and `fmin=50`. A flatten layer adapts the output dimension of the last Conv2d block to the first Linear block. Softmax is applied at the bottom end to provide the class distribution. The entire architecture has around three million parameters.

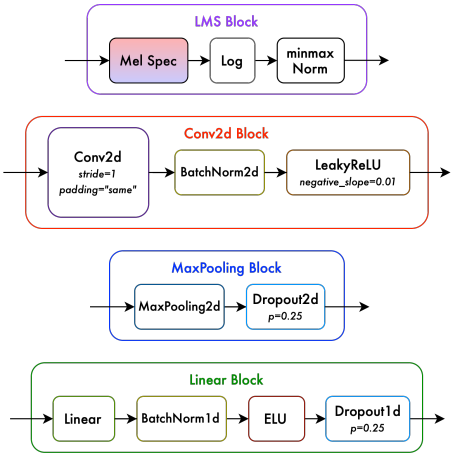

Figure 3: Custom blocks used in the proposed architecture.

4.1.4 Training Process. We train our model for 100 epochs, using a batch size of 32, considering memory use and computational speed. Random pitch shifting is applied separately to each data sample of each batch with a probability of 50% thanks to the torch-audiomentations library[8]. Cross-entropy loss is minimized using ADAM optimization with a weight decay of $1e^{-5}$. The training was performed on an A6000 GPU machine and lasted 40 minutes.

4.2 Results

4.2.1 Accuracy, Macro F1, Confusion Matrix. Measurements were taken on the model state that achieved the lowest validation loss. As mentioned in 4.1.2, our training dataset is unbalanced because of the difference in the total time duration of each class.

[7]https://github.com/khornlund/pytorch-balanced-sampler
[8]https://github.com/asteroid-team/torch-audiomentations

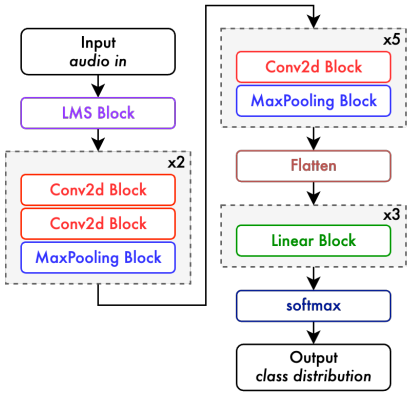

Figure 4: Entire architecture using our custom blocks.

To address this concern, we also measured the macro F1-score because it allows for a more representative measure of model performance, considering no difference between highly and poorly populated classes [12]. Table 3 reports detailed results of accuracy, macro F1-score, and standard deviation for all three experimental setups, while figure 5 shows individual class accuracy for Setup C with data augmentations.

Table 3: Comparison of accuracy, F1-score, and standard deviation results with and without adding audio augmentation during training. Results averaged on five tests (%)

Setup	Original		Augmented	
	Accuracy $\pm\sigma$	Macro F1 $\pm\sigma$	Accuracy $\pm\sigma$	Macro F1 $\pm\sigma$
A	94.48 ± 0.65	92.70 ± 1.15	97.91 ± 0.35	96.58 ± 0.59
B	98.57 ± 0.21	**97.83** ± 0.31	**99.01** ± 0.12	97.61 ± 0.27
C	51.96 ± 2.12	45.64 ± 2.29	82.58 ± 0.87	78.47 ± 1.78

Figure 5: Confusion matrix for training setup C with data augmentations, averaged on five tests (%)

4.2.2 Latency. Latency was measured by measuring the time interval between successive classification results of the first 1000 inferences in running our best-performing model. We did not use

the onset detection in this experiment. To measure the time interval in Max 9, we used the `timer` object. Mean and standard deviation were computed thanks to `array.mean` and `array.stddev` objects. The model was run on the CPU device on an Apple Silicon M1 laptop. We found an average latency of 5.8ms with a standard deviation of 0.43ms. Modifying the sampling rate and buffer size of the DAW did not affect latency.

5 Implementation of ipt∼

The `ipt_tilde` object is a new Max/MSP external designed to enable real-time classification of IPTs using a pre-trained Convolutional Neural Network (CNN). The object is implemented with a multi-threaded architecture that facilitates low-latency audio processing and classification, delivering the output in real-time to Max patches. Below, we break down the core components and functionalities of the implementation (the code for the implementation of ipt∼ is availabe on GitHub[9]):

5.1 Model Loading and Setup

Upon initialization, the `ipt_tilde` object dynamically loads a pre-trained CNN model using the `PyTorch C++ API`[10]. The model's file path and target device (CPU, CUDA, or MPS) are specified by the user. This flexibility allows the classifier to be deployed across different environments. The `IptClassifier` class manages the loading and inference tasks, selecting the computation device based on the user's input, with a fallback to CPU if the specified device is unavailable.

While this paper introduces the new EG-IPT dataset, which is specific to electric guitar, the `ipt_tilde` object can load any trained model, making it adaptable to different instruments. This generality allows the object to be used for classifying IPTs across a variety of instruments.

5.2 Audio Preprocessing

To ensure consistent input for the model, the `ipt_tilde` object integrates the `r8brain-free-src`[11] library for high-quality downsampling. This library converts any audio input to the model's trained sampling rate, preserving classification accuracy with alias-free resampling. Audio input is captured in real-time via Max MSP's audio system and enqueued in a FIFO buffer (`m_audio_fifo`). The audio thread processes each sample, while a separate thread performs inference and delivers results.

5.3 Multi-Threaded Architecture

The object uses a multi-threaded architecture to simultaneously handle audio input and classification inference, preventing blocking or delays in the audio stream. Specifically:

- **Audio Input Handling**: A dedicated thread processes the main audio input, collecting and buffering samples. Each incoming sample is queued in the FIFO buffer `m_audio_fifo`.
- **Classification Inference**: A separate thread performs inference on the buffered audio. Once sufficient data is gathered, the classifier generates results and enqueues them in the `m_event_fifo` for delivery to the Max outlets.

[9]https://github.com/nbrochec/ipt_tilde.git
[10]https://pytorch.org/cppdocs/
[11]https://github.com/avaneev/r8brain-free-src/

5.4 Signal Processing and Classification

The core of the classification process is the `IptClassifier`, which handles the prediction. Key components include:

- **Leaky Integrator**: To stabilize predictions and mitigate abrupt changes due to noise or transient events, the `LeakyIntegrator` smooths the classifier's output. The integrator applies a decay to past predictions, producing more stable and continuous output over time. The leaky integration is modeled as:

$$y_t = (1 - \alpha) \cdot y_{t-1} + \alpha \cdot x_t$$

  where:
  - y_t is the output at time t,
  - x_t is the current classification output,
  - y_{t-1} is the previous output,
  - α is the leaky factor, defined as $\alpha = \frac{\Delta t}{\tau}$, where Δt is the elapsed time and τ is the time constant.

  The integrator smooths transitions between consecutive classification outputs, reducing the impact of transient noise.
- **Circular Buffer**: To manage real-time audio processing, a circular buffer stores incoming audio samples. The buffer ensures that only a fixed number of samples are kept, discarding older data as new samples are added. The circular buffer is defined as:

$$b_t = (b_{t-1} + 1) \mod N$$

  where:
  - b_t is the current position at time t,
  - b_{t-1} is the previous position,
  - N is the buffer size.

  When the buffer reaches its limit, it wraps around, ensuring that only the most recent samples are stored.
- **Energy Threshold**: An energy threshold filters low-energy signals. The threshold is based on the RMS (Root Mean Square) value of the audio signal. If the RMS exceeds a specified threshold (in dB), the signal is considered valid for classification. The energy threshold is defined as:

$$\text{Threshold}_{\text{db}} = 20 \log_{10} \left(\frac{\text{RMS}(x)}{\text{Reference Level}} \right)$$

  where:
  - $\text{RMS}(x)$ is the root mean square of the audio signal x,
  - Reference Level is a fixed value for normalization.

  Signals below the threshold are discarded, ensuring the classifier only responds to significant audio events.

5.5 System Control and Customization

Several adjustable attributes allow users to control the behavior of the classifier in real-time through Max/MSP messages. For detailed explanations of components like the Leaky Integrator and Energy Threshold, refer to the previous section.

- **Sensitivity**: The `sensitivity` attribute controls the model's responsiveness to incoming audio. A higher sensitivity allows quicker reactions to changes in audio input, while lower sensitivity smooths the output.
- **Energy Threshold**: The `threshold` attribute defines the minimum energy level required for classification to occur. It helps filter out irrelevant signals.
- **Threshold Window Size**: The `window` attribute controls the size of the temporal window over which the classifier's

confidence is smoothed. A larger window reduces fluctuations but introduces latency, while a smaller window increases responsiveness but may cause more jitter.
- **Enabled Flag**: The `enabled` attribute allows the classifier to be manually turned on or off, controlling whether classification occurs during the processing cycle.

5.6 Error Handling and Robustness

The object is designed with robust error-handling mechanisms that ensure smooth operation during model loading, initialization, and inference. If errors occur during the loading of the model or the execution of inference, appropriate warnings and error messages are displayed to the user, but the system continues to run without interruption, ensuring that live performance is not disrupted.

5.7 Max/MSP Integration

The `ipt_tilde` object is fully integrated into Max/MSP, utilizing the Min API[12] for the creation of Max externals. This API ensures seamless communication between the audio processing components and the Max environment. Key features include:

- **Attribute Management**: Parameters such as `sensitivity`, `threshold`, and `window` can be adjusted and queried in real-time via Max messages.
- **Real-Time Output**: Classification results are delivered to Max via the following outlets:
  - `outlet_main`: Sends the index of the selected class based on the classification result. This integer value represents the predicted class, which can be used for further processing or control within Max.
  - `outlet_classname`: Sends the name of the selected class as a symbol, providing a human-readable label for the classification result. This allows users to easily interpret the results in terms of meaningful categories.
  - `outlet_distribution`: Sends a list of floating-point values representing the class probability distribution. This list indicates the confidence level for each class, providing insight into how certain the classifier is about its prediction. This output can be useful for analyzing or visualizing the classifier's decision-making process.

5.8 Compilation and Packaging

The `ipt_tilde` object is compiled using CMake[13], linking against the necessary PyTorch and Min-API dependencies. The shared library is built and packaged as a Max-compatible external, which can be used directly in Max/MSP patches. The compilation process integrates the `libtorch` library, which is managed by the `FetchContent` mechanism in the CMake script. This approach ensures that the necessary PyTorch files are downloaded and configured automatically as part of the build process. The `CMake` configuration specifies the architecture for macOS as `arm64` and sets the deployment target to macOS 10.13, ensuring compatibility with the target environment. Additionally, the inclusion of `r8brain-free-src` for audio resampling and integration with Min-API simplifies the packaging and deployment of the Max external. Post-build steps ensure that all necessary dependencies, including the model and its associated libraries, are included in the final package.

[12]https://github.com/Cycling74/min-api
[13]https://cmake.org/documentation/

6 Discussion

When tested on setups A and B (semi-heterogeneous datasets), our model shows a very strong performance, reaching 99.01% accuracy and 97.83% Macro F1-score. When tested on setup C (fully heterogeneous dataset), the model shows a strong performance, reaching 82.58% accuracy and 78.47% Macro F1-score. In any case, using augmented datasets brought overall better performance, thus validating our hypothesis that multi-source recordings enhance classifier's performance [4].

The confusion matrix Figure 5 shows that several IPTs are misclassified. We think these misclassifications are related to perceptual similarity between classes. For example, the *muted* playing technique is a short-duration technique with a limited spectral content, which is also the case for *staccato*. We think increasing the resolution of the LMS would provide the model with better insight into the slight variations of frequencies over time, allowing for accurate classification of short-duration techniques [17]. For the *vibrato* and the *legato* techniques, the model tends to classify them at 23.91% and 22.88%, respectively, as *bend* techniques. We think this misclassification is related to the span of the pitch movement over time, as the vibrato, legato, and bend imply pitch change. Our *legato* class is composed of three playing techniques, *hammer-on*, *pull-off*, and *slide*, that consists of playing a pitch next to another either above or below. We think that including larger *hammer-on* and *pull-off* intervals in our training dataset could further characterize the *legato* class, which musically also includes large spans of tied notes, which we think would improve its classification. We think the misclassification of *vibrato* and *bend* is more complex to dampen. Indeed, the production of *vibrato* on string instruments consists of bending the string to create an up-and-down alternative pitch movement. The only difference with the basic *bend* is that the string is only bent up or down once. Testing out several combinations of simultaneous analysis window length is worth exploring as it may provide the model recent and 'older' information [2] and therefore may be able to classify *vibrato* and *bend* properly.

Regarding the latency performance, the result shows that the system is trustworthy, with an average latency of 5.8ms over 1000 inferences. Compared to an existing study [10] that uses PyAudio and PythonOSC, our implementation of Max/MSP object drastically reduces the latency. The object needs testing from different users to fine-tune its parameters according to the model trained and the desired playing characteristics. Also, we hope the spread of our workflow could help improve its efficiency and generalization.

Table 4 presents a comparative overview of our dataset and architecture performance in terms of accuracy, efficiency (F1 score), and real-time latency relative to other state-of-the-art guitar classification methods. This is not intended as a formal benchmarking study, as the compared systems differ in architecture due to limited availability of datasets or implementation details in prior works. The comparisons are based on reported metrics from published studies. In some cases, standard deviations are not included because the original studies do not provide complete statistical information. Most of these studies assess only two out of the three performance aspects chosen, and only one [21] includes an evaluation of real-time latency, while the others focus exclusively on offline processing. Additionally, several of these works address multiple tasks simultaneously, which reduces the clarity and focus on the IPT recognition task specifically.

Table 4: Comparison of our EG-IPT dataset and model architecture with other state-of-the-art guitar classification approaches, based on the number of IPTs, accuracy, F1 score, and real-time system latency.

Dataset and Setup	n IPTs	Accuracy $\pm\sigma$ (%)	Macro F1 $\pm\sigma$ (%)	Latency $\pm\sigma$ (ms)
PercCNN[21]	4	-	92.92 ± 2.99	12.68 ± 1.13
IDMT-SMT-Guitar [15]	6	83 ± 13.79	-	-
Guitar Playing Techniques [37]	7	-	71.70 ± 13.40	-
Guitar Style [23]	9	84.2 ± -	83.1 ± 1.2	-
AG-PT [34]	12	-	88.0 ± -	-
EG-IPT	**14**	**99.01 ± 0.12**	**97.83 ± 0.31**	**5.80 ± 0.43**

Finally, in this study, we use a large amount of data with a relatively large model (3 million parameters), which may be a problem for users with limited data and calculation resources. To tackle this concern in the future, we will conduct an ablation study to remove layers that do not improve model performance. Furthermore, we think approaching IPTs classification task with a few-short learning method, which consists of providing a few labeled examples to enable rapid generalization [41], is worth exploring.

7 Conclusions

This paper introduced a comprehensive framework for real-time classification of Instrumental Playing Techniques (IPTs), featuring the new EG-IPT dataset and the novel ipt∼ Max/MSP object. The EG-IPT dataset significantly advances the field by offering diverse electric guitar techniques recorded across various sources and pickup configurations, providing robust training data for machine learning models. A CNN-based classifier demonstrated state-of-the-art performance, validating the dataset's relevance for further research. The ipt∼ object enables real-time classification in Max/MSP, supporting models trained on diverse datasets and accommodating various instruments and techniques. This integration establishes an end-to-end workflow for data collection, machine learning, and interactive applications in human-computer music interfaces. While this paper emphasized technical details, the contributions open significant possibilities for interactive systems in the NIME community. The ipt∼ object complements existing interactive systems in Max/MSP, enabling integration with libraries like FluCoMa [38], MuBu [32] and Rave/nn∼ [6], as well as HCI systems like Voyager [16], Somax2 [1], Dicy2 [25], and Improtek [24]. By contributing these tools, we aim to expand expressive musical interactions, reinforcing the NIME theme of entangling data collection, machine learning, and interactive human-machine performance.

8 Ethical Standards

This research was supported by funding from a European Research Council (ERC) project, which had no influence on the design, execution, analysis, or reporting of the study. The EG-IPT dataset was recorded entirely by the authors, ensuring full control over its ethical handling. The Strandberg dataset was recorded by a human subject with informed consent obtained prior to the recording sessions. All data collection adhered to ethical principles, ensuring the privacy, autonomy, and rights of the individual involved. No animals were involved in this research. This work is intended to benefit the NIME community by providing open and ethically sourced tools and datasets, promoting responsible and innovative applications in human-computer musical interaction.

Acknowledgments

This research is supported by the European Research Council (ERC) as part of the Raising Co-creativity in Cyber-Human Musicianship (REACH) Project directed by Gérard Assayag, under the European Union's Horizon 2020 research and innovation program (GA #883313). Funding support for this work was provided by a Japanese Ministry of Education, Culture, Sports, Science and Technology (MEXT) scholarship to Nicolas Brochec. The authors would like to thank Lenny Renault for recording the Strandberg Dataset.

References

[1] Gérard Assayag, Laurent Bonnasse-Gahot, and Joakim Borg. 2022. Cocreative Interaction: Somax2 and the REACH Project. *Computer Music Journal* 46, 4 (Dec. 2022), 7–25. https://doi.org/10.1162/comj_a_00662

[2] Sebastian Böck, Andreas Arzt, Florian Krebs, and Markus Schedl. 2012. Online real-time onset detection with recurrent neural networks. In *Proceedings of the 15th International Conference on Digital Audio Effects (DAFx-12), York, UK*. sn, 17–21.

[3] Nicolas Brochec and Tsubasa Tanaka. 2023. Toward Real-Time Recognition of Instrumental Playing Techniques for Mixed Music: A Preliminary Analysis. In *International Computer Music Conference (ICMC 2023)*. Shenzhen, China. https://hal.science/hal-04263718

[4] Nicolas Brochec, Tsubasa Tanaka, and Will Howie. 2024. Microphone-based Data Augmentation for Automatic Recognition of Instrumental Playing Techniques. In *International Computer Music Conference (ICMC 2024)*. Seoul, South Korea. https://hal.science/hal-04642673

[5] Judith C Brown and Paris Smaragdis. 2004. Independent component analysis for automatic note extraction from musical trills. *The Journal of the Acoustical Society of America* 115, 5 (2004), 2295–2306.

[6] Antoine Caillon and Philippe Esling. 2021. RAVE: A variational autoencoder for fast and high-quality neural audio synthesis. arXiv:2111.05011 [cs.LG] https://arxiv.org/abs/2111.05011

[7] Yu-Hua Chen, Wen-Yi Hsiao, Tsu-Kuang Hsieh, Jyh-Shing Roger Jang, and Yi-Hsuan Yang. 2022. Towards automatic transcription of polyphonic electric guitar music: A new dataset and a multi-loss transformer model. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 786–790.

[8] Jean-Francois Ducher and Philippe Esling. 2019. Apprentissage profond pour la reconnaissance en temps réel des modes de jeu instrumentaux. In *Journées d'Informatique Musicale*.

[9] Jean-Francois Ducher and Philippe Esling. 2019. Folded CQT RCNN for real-time recognition of instrument playing techniques. In *International Society for Music Information Retrieval*.

[10] Marco Fiorini and Nicolas Brochec. 2024. Guiding Co-Creative Musical Agents through Real-Time Flute Instrumental Playing Technique Recognition. In *Sound and Music Computing Conference (SMC 2024)*. Porto, Portugal. https://hal.science/hal-04635907

[11] Marco Fiorini, Nicolas Brochec, Joakim Borg, and Riccardo Pasini. 2025. *EG-IPT Dataset (Electric Guitar Instrumental Playing Techniques)*. https://doi.org/10.5281/zenodo.15205644

[12] Margherita Grandini, Enrico Bagli, and Giorgio Visani. 2020. Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756* (2020).

[13] Tung-Sheng Huang, Ping-Chung Yu, and Li Su. 2023. Note and playing technique transcription of electric guitar solos in real-world music performance. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1–5.

[14] Yu-Fen Huang, Jeng-I Liang, I-Chieh Wei, Li Su, et al. 2020. Joint analysis of mode and playing technique in Guqin performance with machine learning.. In *ISMIR*. 85–92.

[15] Christian Kehling, Jakob Abeßer, Christian Dittmar, and Gerald Schuller. 2014. Automatic Tablature Transcription of Electric Guitar Recordings by Estimation of Score-and Instrument-Related Parameters.. In *DAFx*. 219–226.

[16] George E Lewis. 2000. Too many notes: Computers, Complexity and Culture in Voyager. *Leonardo Music Journal* 10 (2000), 33–39.

[17] Dichucheng Li, Mingjin Che, Wenwu Meng, Yulun Wu, Yi Yu, Fan Xia, and Wei Li. 2023. Frame-level multi-label playing technique detection using multi-scale network and self-attention mechanism. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1–5.

[18] Beici Liang, György Fazekas, Andrew McPherson, and Mark Sandler. 2017. Piano pedaller: a measurement system for classification and visualisation of piano pedalling techniques. (2017).

[19] Davide Lionetti, Luca Turchet, Massimiliano Zanoni, and Paolo Belluco. 2024. Muscle-Guided Guitar Pedalboard: Exploring Interaction Strategies Through Surface Electromyography and Deep Learning. , Article 37 (September 2024), 11 pages. https://doi.org/10.5281/zenodo.13904842

[20] Vincent Lostanlen, Joakim Andén, and Mathieu Lagrange. 2018. Extended playing techniques: the next milestone in musical instrument recognition. In *Proceedings of the 5th international conference on digital libraries for musicology*. 1–10.

[21] Andrea Martelloni, Andrew P McPherson, and Mathieu Barthet. 2023. Real-time Percussive Technique Recognition and Embedding Learning for the Acoustic Guitar. *arXiv preprint arXiv:2307.07426* (2023).

[22] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. 2015. librosa: Audio and music signal analysis in python.. In *SciPy*. 18–24.

[23] Alexandros Mitsou, Antonia Petrogianni, Eleni Amvrosia Vakalaki, Christos Nikou, Theodoros Psallidas, and Theodoros Giannakopoulos. 2024. A multimodal dataset for electric guitar playing technique recognition. *Data in Brief* 52 (2024), 109842. https://doi.org/10.1016/j.dib.2023.109842

[24] Jérôme Nika, Marc Chemillier, and Gérard Assayag. 2017. ImproteK: introducing scenarios into human-computer music improvisation. *ACM Computers in Entertainment* (Jan. 2017). https://doi.org/10.1145/3022635

[25] Jérôme Nika, Ken Déguernel, Axel Chemla, Emmanuel Vincent, Gérard Assayag, et al. 2017. DYCI2 agents: merging the "free", "reactive", and "scenario-based" music generation paradigms. In *International Computer Music Conference*. Shangai, China.

[26] Tan Hakan Ozaslan and Josep Lluis Arcos. 2010. Legato and glissando identification in classical guitar. In *7th Sound and Music Computing Conference (SMC)*, Vol. 457.

[27] Ashis Pati and Alexander Lerch. 2017. A Dataset and Method for Guitar Solo Detection in Rock Music. https://doi.org/10.17743/aesconf.2017.978-1-942220-15-2

[28] Hegel Pedroza, Gerardo Meza, and Iran R. Roman. 2022. *EGFxSet: Electric guitar tones processed through real effects of distortion, modulation, delay and reverb*. https://doi.org/10.5281/zenodo.7044411

[29] Nicola Privato, Victor Shepardson, Giacomo Lepri, and Thor Magnusson. 2024. Stacco: Exploring the Embodied Perception of Latent Representations in Neural Synthesis. In *Proceedings of the International Conference on New Interfaces for Musical Expression*. Utrecht, Netherlands. http://nime.org/proceedings/2024/nime2024_62.pdf

[30] Loïc Reboursière, Otso Lähdeoja, Thomas Drugman, Stéphane Dupont, Cécile Picard-Limpens, and Nicolas Riche. 2012. Left and right-hand guitar playing techniques detection. In *Proceedings of the International Conference on New Interfaces for Musical Expression*. University of Michigan, Ann Arbor, Michigan. https://doi.org/10.5281/zenodo.1180575

[31] Robert Rowe. 1992. *Interactive music systems: machine listening and composing*. MIT Press, Cambridge, MA, USA.

[32] Norbert Schnell, Axel Röbel, Diemo Schwarz, Geoffroy Peeters, Riccardo Borghesi, et al. 2009. MuBu and friends–assembling tools for content based real-time interactive audio processing in Max/MSP. In *ICMC*.

[33] Hugo Scurto and Ludmila Postel. 2023. Soundwalking Deep Latent Spaces. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, Miguel Ortiz and Adnan Marquez-Borbon (Eds.). Mexico City, Mexico, Article 33, 4 pages. https://doi.org/10.5281/zenodo.11189166

[34] Domenico Stefani, Gregorio Andrea Giudici, and Luca Turchet. 2024. On the Importance of Temporally Precise Onset Annotations for Real-Time Music Information Retrieval: Findings from the AG-PT-set Dataset. In *Proceedings of the 19th International Audio Mostly Conference: Explorations in Sonic Cultures* (Milan, Italy) *(AM '24)*. https://doi.org/10.1145/3678299.3678325

[35] Domenico Stefani, Matteo Tomasetti, Filiippo Angeloni, and Luca Turchet. 2024. Esteso: Interactive AI Music Duet Based on Player-Idiosyncratic Extended Double Bass Techniques. In *Proceedings of the International Conference on New Interfaces for Musical Expression*. Utrecht, Netherlands. https://doi.org/10.5281/zenodo.13904929

[36] Li Su, Hsin-Ming Lin, and Yi-Hsuan Yang. 2014. Sparse modeling of magnitude and phase-derived spectra for playing technique classification. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22, 12 (2014), 2122–2132.

[37] Li Su, Li-Fan Yu, and Yi-Hsuan Yang. 2014. Sparse Cepstral, Phase Codes for Guitar Playing Technique Classification.. In *ISMIR*. 9–14.

[38] Pierre Alexandre Tremblay, Owen Green, Gerard Roma, James Bradbury, Ted Moore, Jacob Hart, and Alex Harker. 2022. Fluid Corpus Manipulation Toolbox. https://doi.org/10.5281/zenodo.6834643

[39] Federico Visi. 2024. In *Proceedings of the International Conference on New Interfaces for Musical Expression*. Utrecht, Netherlands. https://doi.org/10.5281/zenodo.13904810

[40] Changhong Wang, Emmanouil Benetos, and Elaine Chew. 2021. *CBFdataset: A Dataset of Chinese Bamboo Flute Performances*. https://doi.org/10.5281/zenodo.5744336

[41] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. 2020. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)* 53, 3 (2020), 1–34.

[42] Qingyang Xi, Rachel M Bittner, Johan Pauwels, Xuzhou Ye, and Juan Pablo Bello. 2018. GuitarSet: A Dataset for Guitar Transcription.. In *ISMIR*. 453–460.