Chapter 4

Musically Motivated CNNs for Instrument Recognition

4.1 Introduction

This chapter provides a detailed description of the research, along with experiments and novel contributions, with respect to the topic of instrument recognition. Following the same strategy used when building the DNNs for music source separation in Chapter 3, all the models proposed in this chapter are also focused on efficiency (low number of trainable parameters) via the application of domain knowledge to build musically motivated CNN architectures. Hence, similar architectures are herein proposed and adapted for this new task of classification rather than separation.

The research can be divided into two related but distinct tasks. In Section 4.2, the task of Instrument Activity Detection (IAD), where instrument activations are estimated on a frame-level basis, is studied. Special interest is given in the fact that transient and stationary parts of sounds affect our perception of timbre in different ways and a method that first decomposes a music spectrogram into its constituent transient, stationary and noisy-like sounds is proposed. In Section 4.3 the pitch streaming (also known as instrument assignment) task is addressed as note-level instrument classification and a method utilising an auxiliary input carrying the pitch, onset and offset information of note-events alongside the time-frequency representation of the audio signal is proposed. The initial findings of the experiments in Section 4.2 were presented in Lordelo et al. [2020b] while most of the research done in Section 4.3 was published in Lordelo et al. [2021a]. Lastly, Section 4.4 concludes the chapter by discussing our findings and proposals as well as suggesting future potential

work.

The contributions of this chapter can be summarised as follows:

- Instrument Activity Detection: Proposal of a novel DNN for frame-level instrument recognition, i.e., each frame of a music spectrogram is associated to one or multiple instrument classes.
- Transient, Steady-State and Noisy-like sound features: Utilisation of a Transient-Stationary-Noise Decomposition (TSND) method [Driedger et al., 2014a] as pre-processing step in order to extract features related to transient, steady-state and residual sounds.
- Pitch Streaming: Proposal of a novel DNN that associates each note from a music signal to its instrumental source.
- Modular Framework for Pitch Streaming: Pitch streaming approach works with any MPE method. The streaming performance is evaluated when using ground-truth note labels as well as 2 state-of-the-art MPE algorithms [Thomé and Ahlbäck, 2017; Wu et al., 2019] to estimate note-events (pitch, onset and offset).
- Input Representation: Comparison of different representations for the input audio and auxiliary information for the task of pitch streaming.
- Musically Motivated CNN Architectures: Proposal of CNN architectures for frame-level instrument activity detection and for pitch streaming tasks that use musically motivated kernel shapes for the convolutions, facilitating learning representations for different instruments and note sound states. It is shown that their use improves the performance in both tasks.

4.2 Instrument Activity Detection

In this section, the task of instrument activity detection, where multi-label instrument classes are predicted for every frame of an input audio spectrogram, is addressed. Such problem is addressed as a frame-level instrument classification and a novel classifier architecture is proposed following the same methodology utilised in Chapter 3: the utilisation of domain knowledge to build more efficient musically motivated CNN architectures.

Studies on music cognition [McAdams et al., 1995; Clark et al., 1964] show that information regarding the production mode of the sound, i.e., if the sound is generated by bowing, plucking or hammering of a string, or by impelling air into a wind instrument, is essentially located at the beginning and at the end of the notes (transient parts). Some features taken from the attack transients

of sound, such as onset duration and crest factor, have also proved to be helpful for instrument family identification on isolated notes. On the other hand, the relation between the energy of the harmonics (stationary part) also has a fundamental impact on our perception of timbre.

For instance, Essid et al. [2005] performed experiments evaluating the usefulness of a combination of transient-based features and steady-state features for instrument identification. The authors showed that an SVM classifier achieves better performance in solo instrument recognition when using only attack-transient features compared to only steady-state features. However, this is valid only when the decision window for classification is short (48 ms and 80 ms). When larger decision windows (496 ms and 1936 ms) were used, the authors verified that methods not considering the distinction between transients and steady-state features performed better, indicating that musical context also has a central role in automatic recognition of instruments. They concluded their study proposing that a technique mixing both transient and steady-state segments and specialised classifiers would be the ideal approach to achieve a better overall performance.

Motivated by those works and the fact that one of the fundamental ideas of the research performed in the thesis is the exploitation of harmonic and percussive filters and features (see also Chapter 3 for experiments related to harmonic-percussive source separation), the experiments in this section also investigate whether the task of instrument classification can be improved by explicitly providing both types of sound features to the framework. The final proposal is to split a music spectrogram into a *transient-enhanced spectrogram*, a *steady-state-enhanced spectrogram* and a *noisy-like residual spectrogram*, which are estimated by a Transient-Stationary-Noise Decomposition (TSND) method [Driedger et al., 2014a]. Those 3 spectrograms are jointly used as inputs to the DNN classifier.

A preliminary version of the work presented in this section was presented in Lordelo et al. [2020b].

4.2.1 Proposed Method

The proposed approach is similar to other data-driven instrument recognition methods in the sense that instrument activity detection is addressed as a multi-label classification task, in which the objective is to pinpoint all the instruments that are active in each frame across time, and a DNN is used as the classifier [Hung and Yang, 2018; Gururani et al., 2018; Hung et al., 2019]. However, the proposed method is primarily different from previous work in the following ways:

1. Instead of utilising raw audio spectrograms as inputs to the neural network, transient and steady-state sound information is explicitly provided as different inputs to the framework;

2. The classifier architecture is musically motivated, including vertical, square and horizontal kernel shapes to help learning harmonic and percussive features directly from the time-frequency representation, which helps in recognising instruments more efficiently;

3. The classifier also contains a dense arrangement of skip-connections in order to avoid learning redundant feature-maps, reducing the number of trainable parameters;

4. Instead of performing the classification for every frame of the input at once, only the central frame of the input spectrogram is classified, and musical contexts varying from 50 ms to 800 ms around the central frame are evaluated.

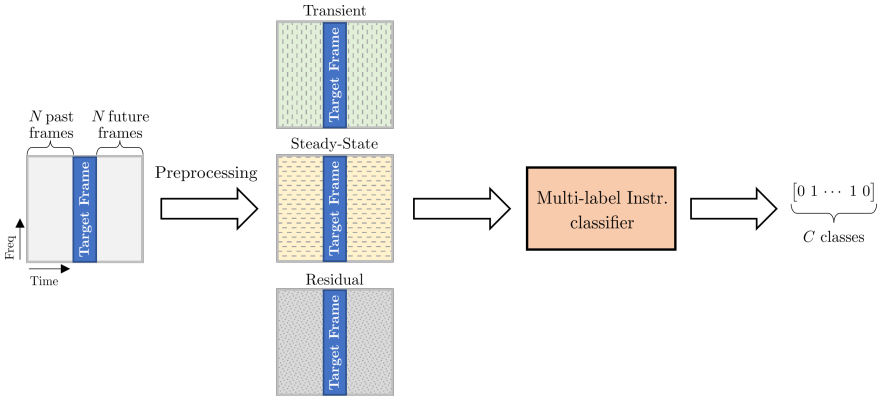


Figure 4.1: Overview of the proposed framework for IAD. During the preprocessing stage, transient-enhanced, steady-state-enhanced and residual spectrograms are estimated and are later used as inputs to a classifier. The different colours represent a different type of sound-enhanced spectrogram

Figure 4.1 depicts an overview of the proposed frame-level instrument recognition approach. The main motivation of the method is to facilitate the learning of transient-related and steady-state-related features by explicitly providing separated sources to the classifier as different input channels. More specifically, transient-enhanced, steady-state-enhanced and residual magnitude spectrograms are estimated from the original spectrogram of the music signal during a preprocessing step and later used as joint inputs to a deep-learning-based multi-label instrument classifier.

Furthermore, since musical context is also a fundamental aspect for discriminating the timbre of sounds, instead of classifying the information contained in just a single frame of the spectrogram, a context of N future frames and N past frames is used as a **decision window** for the classifier. In other words, time-frequency representations of shape $(F \times (2N+1))$ are used as inputs to the classifier, where F is the number of frequency bins and $2N+1$ is the number of frames per input (decision window). The classifier is trained to recognise the active instruments in the $(N+1)$-th frame, i.e., the instrument activation labels associated with the middle frame of the decision window is used as ground-truth targets for the classification task. In the experiments, values of N varying from 2 to 40 frames, which are equivalent of using decision windows from 50 ms to 810 ms respectively, were tested and the value of $N = 35$ obtained the best overall performance. See Subsection 4.2.5.1 for details.

4.2.1.1 Preprocessing Step

The main preprocessing step of the proposed framework is the application of a method to estimate transient-sound-enhanced and stationary-sound-enhanced spectrograms.

Even though in Chapter 3 data-driven methods for performing HPSS were proposed, it is important to note that they are not considered the best approach to be used in this case. The HPSS methods estimate a source with all unpitched-percussive instrument sounds and another with all pitched-harmonic instrumental (or vocal) sounds, but both of those sources yield sounds containing transient and stationary parts. A good example of this is that the attack part of sounds generated by harmonic instruments is essentially transient, but it is still associated with the estimated harmonic source. The Transient-Stationary Separation (TSS) is not performed by HPSS. The reader is referred to Subsection 2.2.2 where differences between harmonic-percussive source separation and transient-stationary separation tasks are discussed.

Consequently, the primary choice for the preprocessing step is a different method taken directly from the literature that essentially performs Transient-Stationary-Noise Decomposition (TSND) [Driedger et al., 2014a]. This method is an extension of the TSS method of Fitzgerald [2010], where a third residual component is added to the separation, which captures sounds that lie in between the clearly harmonic and percussive sounds of the music signal. This type of decomposition of music signals is inspired by the Sines + Transients + Noise (STN) audio model [Levine and Smith III, 1998; Petrovski et al., 2011], whose traditional application lies in low bitrate audio coding. In those works, the goal is to represent a given input audio signal in terms of a parameterised set of sine

waves, transient sounds, and noise that can be processed and later resynthesised back.

Note that instead of denoting it a "*Separation*" method, the term "*Decomposition*" is chosen here instead. The reason for this is the fact that the preprocessing stage is performing a **decomposition** of a single magnitude spectrogram into 3 disjoint magnitude spectrograms. The resulting signals are still time-frequency representations, i.e., there is no waveform estimation via an application of inverse-STFT and the phase is ignored. Therefore, the method is not performing source separation, strictly speaking.

In summary, the main preprocessing step is the application of a TSND method [Driedger et al., 2014a] for estimating 3 time-frequency representations for an input music signal:

- Transient-Enhanced Representation: the representation of stationary parts of the sounds should be suppressed while the representation of the transient parts of the sounds should be salient;
- Stationary-Enhanced Representation: the representation of transient parts of the sounds should be suppressed while the representation of the stationary parts of the sounds should be salient;
- Residual (Noise) Representation: the representation of parts of the sounds that are neither transient nor stationary.

The main idea of this method [Driedger et al., 2014a] is to perform the decomposition by exploiting the anisotropic smoothness of music spectrograms, which is based on the fact that the representation of transient-percussive sounds tend to form straight vertical lines while stationary-harmonic (steady-state) sounds are represented as horizontal structures, considering the vertical axis associated to frequency and horizontal axis to time. For more details, the reader is referred to Subsection 2.1.8, where a brief discussion about this phenomenon is exposed.

Hence transient-percussive parts of sounds can be removed from the spectrogram by a process which emphasises horizontal lines and suppresses vertical ones. On the other hand, a process that emphasises the vertical lines while suppressing the horizontal ones should result in a spectrogram which has most of the pitched-stationary sounds removed.

Such processes can be achieved using a combination of two median filters. Median filters operate by replacing the central sample of a windowed signal by the median value of the window. Mathematically, given a median filter of an odd length $L_{\mathcal{M}}$, the output $y[n]$ of the application of a median filter to a digital

input signal $x[n]$ can be defined as

$$y[n] = \mathcal{M}\left(\{x[n-k], x[n-k+1], \cdots, x[n+k-1], x[n+k]\}\right), \tag{4.1}$$

where $\mathcal{M}$ represents the median operation and $k = \frac{L_{\mathcal{M}}-1}{2}$. In effect, by performing a horizontal median operation across a fixed frequency bin of the spectrogram, fast variations with respect to time, i.e., transient-percussive events will be smoothed out, resulting in a harmonic or stationary-enhanced spectrogram. Similarly, by applying a vertical median filtering operation across a fixed frame of the spectrogram, pitched-stationary sounds will be considered outliers and will be removed from the spectrogram, generating a transient-enhanced spectrogram.

The whole method to perform the decomposition can be defined as follows. Consider an input magnitude spectrogram $X[f, n]$. Initial transient-enhanced $X'_{\mathrm{T}}[f, n]$ and stationary-enhanced $X'_{\mathrm{S}}[f, n]$ decompositions can be computed using a median filter of odd length L_{T} applied vertically (frequency-wise) and a median filter of odd length L_{S} applied horizontally (time-wise) as

$$X'_{\mathrm{T}}[f, n] = \mathcal{M}\left(\{X[f - k_{\mathrm{T}},\, n], \cdots, X[f + k_{\mathrm{T}},\, n]\}\right), \quad k_{\mathrm{T}} = \frac{L_{\mathrm{T}} - 1}{2}, \tag{4.2}$$

$$X'_{\mathrm{S}}[f, n] = \mathcal{M}\left(\{X[f,\, n - k_{\mathrm{S}}], \cdots, X[f,\, n + k_{\mathrm{S}}]\}\right), \quad k_{\mathrm{S}} = \frac{L_{\mathrm{S}} - 1}{2}. \tag{4.3}$$

As pointed out in previous work [Driedger et al., 2014a; Fitzgerald, 2010], the resulting decomposition is not deeply affected by the values of L_{T} and L_{S} as long as they are not extreme. In the experiments the values suggested by Driedger et al. [2014a] were used, which are $L_{\mathrm{T}} = 21$ frequency bins (equivalent to approximately 475 Hz) and $L_{\mathrm{S}} = 21$ frames (equivalent to 210 ms).

Since we are interested in decomposing the original spectrogram, two binary masks are then computed by comparing the values of the initial estimations $X'_{\mathrm{T}}[f, n]$ and $X'_{\mathrm{S}}[f, n]$. A transient-sound mask $M_{\mathrm{T}}[f, n]$ and a stationary-sound mask $M_{\mathrm{S}}[f, n]$ according to

$$M_{\mathrm{T}}[f, n] = \begin{cases} 1, & \text{if} \quad \dfrac{X'_{\mathrm{T}}[f, n]}{X'_{\mathrm{S}}[f, n]} > \beta, \\ 0, & \text{if} \quad \dfrac{X'_{\mathrm{T}}[f, n]}{X'_{\mathrm{S}}[f, n]} \leq \beta; \end{cases} \tag{4.4}$$

$$M_{\mathrm{S}}[f, n] = \begin{cases} 1, & \text{if} \quad \dfrac{X'_{\mathrm{S}}[f, n]}{X'_{\mathrm{T}}[f, n]} > \beta, \\ 0, & \text{if} \quad \dfrac{X'_{\mathrm{S}}[f, n]}{X'_{\mathrm{T}}[f, n]} \leq \beta, \end{cases} \tag{4.5}$$

where $\beta \in \mathbb{R}, \beta \geq 1$ is a separation factor. Intuitively, we can think that for a time-frequency bin to be included in the transient component, it is required that its value in $X'_{\mathrm{T}}[f, n]$ stands out from the stationary portion $X'_{\mathrm{S}}[f, n]$ by at least a factor of β and vice versa. In the experiments, we tested multiple values of β varying from 1.0 to 3.5. See Subsection 4.2.5.3 for details.

When $\beta = 1$ the method reduces to the transient-stationary separation method proposed by Fitzgerald [2010], but a higher value of β allows us to control how strict the decomposition should be when deciding if sounds should be considered transient or stationary. Note that both masks are disjoint, i.e., it is not possible to have both $M_{\mathrm{T}}[f, n] = M_{\mathrm{S}}[f, n] = 1$ given that $\beta \geq 1$. Therefore, in the cases where both masks are zero, the sounds can be considered as part of a residual component (noise) of the decomposition, whose corresponding binary mask $M_{\mathrm{N}}[f, n]$ can be computed as

$$M_{\mathrm{N}}[f, n] = 1 - M_{\mathrm{T}}[f, n] - M_{\mathrm{S}}[f, n] \tag{4.6}$$

In the end, the three masks are applied to the original magnitude spectrogram in order to obtain the final decomposition

$$X_{\mathrm{T}}[f, n] = M_{\mathrm{T}}[f, n] \odot X[f, n]; \tag{4.7}$$

$$X_{\mathrm{S}}[f, n] = M_{\mathrm{S}}[f, n] \odot X[f, n]; \tag{4.8}$$

$$X_{\mathrm{N}}[f, n] = M_{\mathrm{N}}[f, n] \odot X[f, n]; \tag{4.9}$$

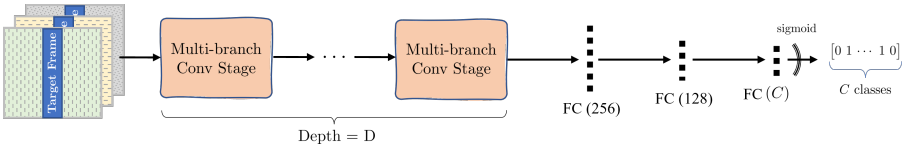
where $\odot$ represents point-wise multiplication.

The estimated spectrograms $X_{\mathrm{T}}[f, n]$, $X_{\mathrm{S}}[f, n]$ and $X_{\mathrm{N}}[f, n]$ are then concatenated in the channel dimension and used as a multi-channel input for the classifier.

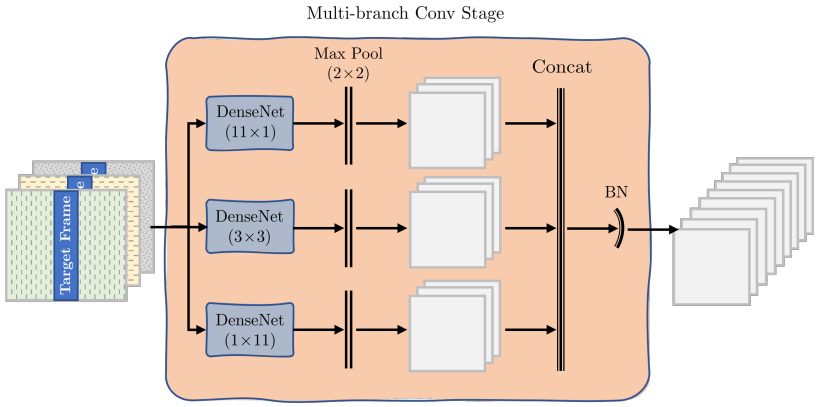
4.2.1.2 Proposed Musically Motivated Classifier Architecture

Briefly speaking, the classifier architecture is a variation of the 3W-MDenseNet, which is another contribution of this thesis (see Subsection 3.2.6 for details), but adapted to perform classification rather than source separation. The main idea behind the methodology is kept: proposal of a musically motivated CNN, whose convolution kernels are a combination of vertical, square and horizontal filters, and utilisation of DenseNets [Huang et al., 2017] instead of regular stacks of convolutions in order to avoid learning redundant feature-maps and have more representative capacity with fewer number of trainable parameters. The proposed architecture is depicted in Figure 4.2. It consists of a sequence of D multi-branch convolutional stages and three fully connected layers. When using $D = 5$ and $N = 35$, the number of trainable parameters of the proposed network

is approximately 2 million.



(a) Overview of the proposed instrument classifier. The transient-enhanced, stationary-enhanced and noise spectrograms are concatenated as a single multi-channel input. The model consists of D multi-branch convolutional stages and 2 fully-connected hidden layers before a final fully connected layer classifies them into one or more of C instrument classes.



(b) Internal structure of a multi-branch convolutional stage. Observe that 3 DenseNets with unique filter shapes run in parallel but their final feature-maps are concatenated at every stage.

Figure 4.2: Proposed architecture for instrument activity detection using TSND as preprocessing stage. The different colours of the input spectrograms represent different types of sounds: transient-enhanced, stationary-enhanced and residual spectrograms.

In the original 3W-MDenseNet, three encoder-decoders run in parallel in separate branches, each with a unique kernel shape: vertical, square or horizontal (see Figure 3.7 for details). The feature-maps generated by each encoder-decoder are later concatenated at a final layer. In the instrument classification task, a similar methodology is adopted by taking only the encoder layers from the 3W-MDenseNet and complementing with fully connected layers at the end in order to perform classification rather than source separation. In addition, a few modifications to the original encoder layers, which here are called a *multi-branch convolutional stage* are also proposed.

The main difference from the regular 3W-MDenseNet encoder layers is that instead of only concatenating the feature-maps computed using different choices of kernel shapes (vertical, square and horizontal) at a later layer in the network,

the encoder (multi-branch convolutional stage) now concatenates the feature-maps of each branch at every stage, right after each down-sampling stage, which is still done using a (2×2) max-pooling layer. By doing so, the feature-maps of every convolutional layer are given access to feature-maps computed using all different choices of kernel shapes from a previous stage.

In Figure 4.2b the internal structure of a multi-branch convolutional stage is shown. Internally, each multi-branch convolutional stage contains 3 separate branches whose convolutions have unique kernel shapes. A branch with horizontal (1×11), a branch with square (3×3), and a branch with vertical (11×1) convolutions are used. In each path, a Densely connected convolutional Network (DenseNet) [Huang et al., 2017] with growth rate $k = 24$ and number of layers $L = 4$ is used. In short, a DenseNet is a stack of L k-channel convolutional layers — each with its own activation function — with a dense pattern of skip-connections, where each layer receives the concatenation of all previous layers' outputs as input. The reader is referred to Subsection 3.2.3 for the detailed internal structure of a DenseNet. After the DenseNet, a (2×2) max-pooling layer is applied in order to down-sample the feature-maps by reducing their dimensions and increase the receptive field at each branch. Afterwards, the three branches are concatenated and the batch is normalised. The final feature-maps are used as input for the next multi-branch convolutional stage. Since we need to concatenate feature-maps that are originated by multiple kernel shapes, padding is necessary to be applied on the convolutions and on the max-pooling layers to ensure the feature-maps maintain the same dimensions across branches.

The ReLU function is used as activation for all layers apart from the last, for which a sigmoid function activation is used (final activation in the interval of $[0, 1]$) in order to perform multi-label classification among C classes. The Binary Cross-Entropy (BCE) loss is used to train the framework.

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{B}\frac{1}{C}\sum_{i=1}^{B}\sum_{j=1}^{C} y_{ij}\log(\hat{y}_{ij}) + (1 - y_{ij})\log(1 - \hat{y}_{ij}), \tag{4.10}$$

where B represents the number of samples in a training batch, the value of $j \in \{1, \cdots, C\}$ represents one of the C different instrument classes, y_{ij} is the ground-truth label with respect to sample i and instrument j, i.e., y_{ij} is 1 if input i has instrument j active or 0 otherwise. Similarly, $\hat{y}_{ij}$ is the prediction of the model with respect to sample i and instrument j (after the application of the final sigmoid activation function).

Furthermore, threshold values V_j, with $j \in \{1, \cdots, C\}$ are used in order to consider the j-th instrument class as positive (instrument is active) or nega-

tive (instrument is not active). In other words, if the final activation of the j-th instrument class is greater than or equal to V_j, that instrument is considered active, otherwise, it is inactive on the target frame. The values used for V_j can vary across the instrument classes and are learned after the model is trained using a validation set. See Subsection 4.2.4 for more details on how the experiments are set up.

4.2.2 Dataset

Despite existing large-scale publicly available datasets, such as IRMAS [Bosch et al., 2012] or OpenMIC-2018 [Humphrey et al., 2018], that could potentially be used for the general instrument recognition task, they are often weakly labelled, meaning that they contain only annotations for the instrument on a clip-level basis. There is no information regarding the time stamps when each instrument is active. In order to be able to train the proposed method in a supervised manner, frame-level instrument annotations are necessary.

Datasets that contain such type of annotations are called strongly labelled datasets and they include Bach10 [Duan et al., 2010], TRIOS [Fritsch and Plumbley, 2013], Slakh2100 [Manilow et al., 2019] and MusicNet [Thickstun et al., 2017]. Bach10 and TRIOS are small scale datasets with only 10 and 5 recordings respectively. On the other hand, Slakh2100 is a large-scale dataset containing 2100 recordings, but it consists of only synthesised data and not real-world recordings. Since the goal of the research is mainly work with real world data, avoiding the analysis of artificial sounds, the MusicNet dataset [Thickstun et al., 2017] was chosen as the ideal dataset for the experiments of this chapter.

The MusicNet dataset [Thickstun et al., 2017] is the largest publicly available dataset with non-synthesised data that is strongly labelled for the task of instrument recognition. This means that we know the exact frames where the instruments are active in the signal, which permits the training of supervised models to perform instrument recognition at the frame-level, note-level, and clip-level. The dataset contains 330 freely-licensed classical music recordings by 10 composers, written for 11 instruments, along with over 1 million annotated labels indicating the onset, offset and pitch of each note in the recordings and the instruments that play them.

The instrument taxonomy included in MusicNet is: *piano, violin, viola, cello, french horn, bassoon, clarinet, harpsichord, bass, oboe* and *flute*. However, the last 4 instruments (harpsichord, bass, oboe and flute) do not appear in the original test set provided by the authors. Therefore, in all the experiments using this dataset the labels related to those instruments were ignored and a 7-class instrument classification using the following classes: **piano, violin,**

viola, cello, french horn, bassoon and **clarinet** was performed. It is worth noting that even though the sounds of those 4 non-classified instruments are not recognised, they are not removed from the training data. The reason for this is not only to have as many recordings as possible for training, but also to allow the model to be more robust and recognise the target instruments in the potential presence of extra unwanted sounds.

4.2.3 Evaluation Metrics

The classification performance is evaluated by computing the frame-level F-score (F_s), which is directly related to the precision (P) and recall (R) according to:

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}},\ R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}},\ F_\mathrm{s} = \frac{2PR}{P + R}, \tag{4.11}$$

where TP is the number of true positives, FP the false positives and FN the false negatives.

Given a music track of the test set of MusicNet, an STFT magnitude spectogram with a time resolution of 10 ms is computed and each frame is classified. The numbers of true positives, true negatives, false positives and false negatives are computed on a per-class (per-instrument) basis and then those values are aggregated across all the tracks in the test set.

In some tables in this section, the per-class F-scores are directly provided, however, in order to obtain a single metric value that represents the overall performance of an evaluated method, the per-instrument F-score values can be averaged. In the literature, there are 3 most common ways of computing an average value of F-score and all of them are used here.

Macro-Averaged F-score

The *macro-averaged F-score* (or just *macro F-score*) is computed using the arithmetic mean of all the per-class F-scores. This method treats all classes equally regardless of the number of occurrences of each class in the dataset.

This is probably the most straightforward way of averaging the methods and is also the most common metric used in the literature.

Weighted-Averaged F-score

Instead of taking the arithmetic mean of all the per-class F-scores, the *weighted-averaged F-score* (or just *weighted F-score*) is calculated by taking a weighted average. The weight of a particular class is set to the number of examples of that class.

Differently than the macro F-score, the weighted F-score takes into account the proportion of data in each class and is a "fairer" metric when working with imbalanced datasets.

Micro-Averaged F-score

The *micro-averaged F-score* (or just *micro F-score*) is a global computation of an overall F-score by counting the true positives, false negatives, and false positives across all the different classes and then using Equation (4.11).

Essentially, it is computing the proportion of correctly classified observations out of all observations. This definition matches the overall accuracy definition in cases when we are working with a single-label method, i.e., when it is guaranteed that each test example should be assigned to exactly one class. In the general multi-label classification case the two metrics are not equivalent.

4.2.4 Experiment Setup

In all experiments the original train/test split provided in the original MusicNet dataset is used. The sampling frequency of the audio input is set to the original 44100 Hz and an STFT using Blackman-Harris windows of 1024 samples (23 ms) with a hop size of 441 samples (10 ms) is computed. As mentioned in Subsection 4.2.1, I set a decision window of N past frames and N future frames of the target frame to be classified. This means that each input is a magnitude spectrogram with shape $(F \times (2N + 1))$, where $F = 513$ and $2N + 1 = 71$.

Since MusicNet provides ground-truth instrument-activation labels for every audio sample in the resolution of 44100 Hz, the target associated with any input can be computed by taking the instrument labels associated to the middle audio sample of the middle $(N + 1) = 36$-th frame of the input. Mathematically, this means that if the first frame of the i-th input has $x[i]$ as its middle sample, the target instrument for that particular input is the label associated with the sample $x[441 \cdot N + i] = x[441 \cdot 35 + i]$. The labels associated to the N first (and last) frames of every track in the train set were ignored because the associated input to the model would require padding. However, during evaluation on test set those frames were not ignored.

It is important to mention that a lot of different inputs can be generated by treating each frame of the STFT of a music recording as a potential central frame to be classified. Therefore, if the STFT of a single music recording has a total of T frames, $T - 2N$ different inputs can be generated by moving a $2N + 1$ decision window accross the original spectrogram. For instance, despite having only 10 recordings in the MusicNet test set, the experiments are evaluated in a total of effectively different 73259 input samples.

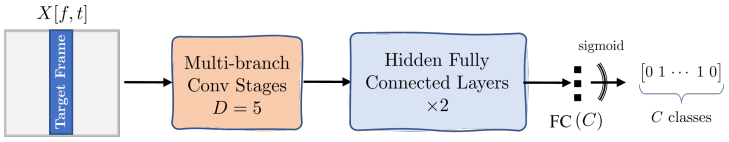
From the training set 15% of inputs and corresponding labels of each class are picked and put into a validation set. The models were trained using the Adam optimiser with an initial learning rate of 0.001, which was reduced by a factor of 0.2 if the binary cross-entropy loss stopped improving for 2 consecutive epochs on the validation set. If no improvement happens after 5 consecutive epochs, the training was stopped early. The experiments were performed using the Tensorflow/Keras Python package.

As shown in Figure 4.3, three model variations are implemented and compared:

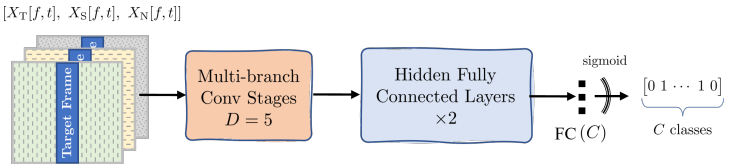
- Baseline Model: There is no TSND preprocessing stage and the original spectrogram of the music clip is used as input to the model. Block diagram is depicted in Figure 4.3a.
- Multi-Channel Model: TSND is performed and the three resulting spectrograms are concatenated, yielding a multi-channel spectrogram that is used as input to the model. Block diagram is depicted in in Figure 4.3b.
- Multi-Input Model: TSND is performed and three single-channel input heads, each with a type of estimated spectrogram, are used in the model. Block diagram is depicted in in Figure 4.3c.

In order to keep the number of trainable parameters of each type of model close to 2 million, the growth rate of every the DenseNets in each multi-branch convolutional stage is set to 24 for the baseline and the TSND-multi-channel models, and to 12 for the TSND-multi-input model. In addition, the experiments involving models with only the set of square (3×3) filters, had their DenseNet's growth rate increased to 62 at each multi-branch convolutional stage.

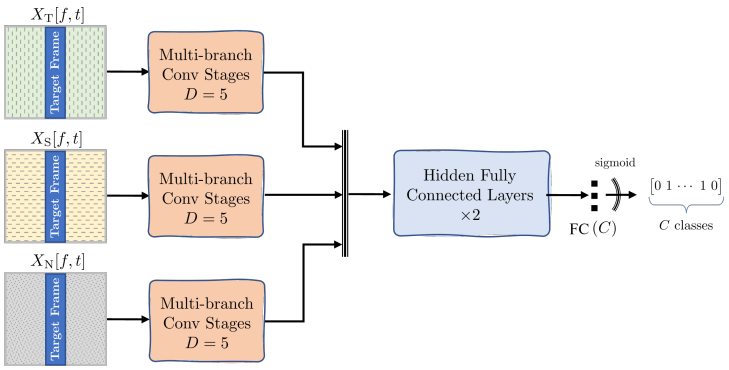
Regarding the threshold values V_j, with $j \in \{1 \cdots C\}$, that are used to verify if j-th instrument class should be considered active or inactive, for every experiment in this section, an exhaustive search is done using data from the validation set and the threshold values that obtain the best performance are later used for evaluating the models on the test set. More specifically, the per-instrument F-score is computed on the validation set using values of V_j ranging from 0.10 to 0.95 with a step size of 0.05 and the values that obtain the best per-class metric are then picked for performing evaluation on the test set.



(a) Baseline: This model has no TSND preprocessing step. Its input is generated directly taking a decision window of N future frames and N past frames from the original magnitude spectrogram of the recording.



(b) Multi-channel: This model uses TSND as preprocessing step and the transient-enhanced, stationary-enhanced and residual spectrograms are concatenated and used as a single 3-channel input for the model.



(c) Multi-Input: This model uses TSND as preprocessing step but the transient-enhanced, stationary-enhanced and residual spectrograms are used as unique input heads for the model.

Figure 4.3: Different models evaluated in the experiments. Different input colours represent different types of estimated sources: transient, stationary and residual.

4.2.5 Results

The results of experiments related to frame-level instrument classification using the previously explained methods are provided in this Subsection. It is started by showing preliminary studies performed using the baseline model only with the objective of finding the best choice of length for the decision window. Then experiments to verify whether the utilisation of multiple kernel shapes in the architecture is beneficial for the task is shown, comparing the proposed architecture with other DNNs that utilise higher number of trainable parameters.

Afterwards, experiments using TSND as preprocessing step are analysed and compared with baselines and an exhaustive test to verify how the separation factor β impacts the performance is discussed.

4.2.5.1 Length of Decision Window

This initial set of experiments was implemented in order to determine the best decision window to be used under this classification scenario. Different decision windows varying from 50 ms to 810 ms were evaluated and the results are shown on Figure 4.4. Since the STFT has a step size of exactly 441 samples, the frame-wise resolution is 10 ms and consequently, the values shown in the legend can be mapped directly to $2N + 1$ by dividing by 10.

Furthermore, note that for models with lower dimension, slightly shallower versions of the model using $D = 4$ (when $N = 15$ or $N = 10$) and $D = 3$ (when $N = 5$) had to be used instead of the regular $D = 5$ of other methods. This is due to the fact that the length of the input $(2N + 1)$ has to be at least 2^D to take into account the D stages of down-sampling by 2 via max-pooling.

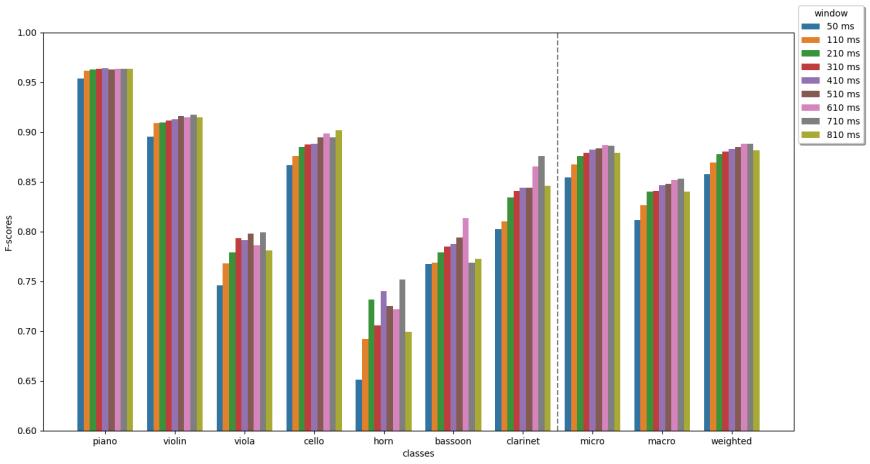


Figure 4.4: Comparison of the performance of the baseline approach (without TSND) using different duration of the input's musical context (length of decision window).

Analysing Figure 4.4, it is possible to verify that the length of decision window has more impact on the detection of woodwind instruments and horn than string instruments. It is believed that this is due to the fact that, in general, not only the average duration of the notes played by string instruments is shorter than the average duration of the notes played by woodwind and brass instruments in musical pieces, but also, their attacks are more prominent and, consequently, most of the information required to recognise string-based instru-

ment sounds is in the frequency envelope of the note attacks [McAdams, 2013; Essid et al., 2004]. Analysing shorter or longer musical contexts does not bring much benefit.

However, analysing longer windows increases the bassoon F-score from 0.7674, at 50 ms, to 0.8134 at 610 ms and can bring improvements of 7.5% in the clarinet F-score and 10.5% for horn F-score. The overall performance of the model using a context window of 710 ms (equivalent to an input length of $2N + 1 = 71$ frames) obtained the best macro F-score of 0.8529. Therefore, the value of $N = 35$ was kept for the subsequent experiments.

4.2.5.2 Effects of Kernel Shapes and Skip-Connections

This experiment has the purpose of validating the hypothesis that the addition of vertical and horizontal kernel shapes to the convolutional layers of the network is beneficial for the instrument recognition task. There is also interest in verifying whether or not using the dense arrangement of skip-connections in DenseNets can also avoid learning of redundant feature-maps and obtain high performance with a reduced number of trainable parameters.

In this case the baseline model (without TSND) is still used because it is faster to train and evaluate due to the absence of the preprocessing stage and because the effects of the application of TSND are not important to the current evaluation.

Two variations of model architecture were evaluated: an architecture using regular stacks of convolutional layers + max-pooling instead of DenseNets (no internal skip-connection) and an architecture using regular DenseNets of depth 4 at each convolutional stage (with dense arrangement of skip-connections). For each of the two types of architectures, a model using only square (3×3) kernel shapes is compared with a model that uses not only square, but also horizontal and vertical kernel shapes. Table 4.1 provides the choice of parameters for each model and the total number of trainable parameters in each case.

The results can be seen in Table 4.2, which shows the F-score metric for every instrument class and the micro, macro and weighted average metric. It is possible to note that the models that do not use DenseNets have double the number of trainable parameters, but perform similarly or, in some cases, worse than their corresponding model using DenseNets. This allow us to conclude that, indeed, the dense arrangement of skip-connections of the DenseNets facilitates the learning of more discriminative feature-maps and effectively helps in reducing the number of trainable parameters in the model.

Moreover, when comparing models using only square filters to models using a combination of square, horizontal and vertical, we see that the macro-averaged

Table 4.1: Choice of the parameters for the convolutional stages for the the different tested models. The models using multiple kernel shapes have 3 branches using (1×11), (3×3), (11×1) kernel shapes respectively. The values that appear with a $(3\times)$ prefix indicate that each of the 3 branches use that value for the respective parameter. The fully connected layers of all models are the same and follow Figure 4.2a

	Parameter	Without DenseNets		With DenseNets	
		Square	Multiple	Square	Multiple
Conv. Stage 1	Growth Rate k	—	—	60	$(3\times)$ 24
	# of Channels	64	$(3\times)$ 8	—	—
	# of Layers L	1	$(3\times)$ 1	4	$(3\times)$ 4
Conv. Stage 2	Growth Rate k	—	—	60	$(3\times)$ 24
	# of Channels	128	$(3\times)$ 16	—	—
	# of Layers L	1	$(3\times)$ 1	4	$(3\times)$ 4
Conv. Stage 3	Growth Rate k	—	—	60	$(3\times)$ 24
	# of Channels	256	$(3\times)$ 32	—	—
	# of Layers L	1	$(3\times)$ 1	4	$(3\times)$ 4
Conv. Stage 4	Growth Rate k	—	—	60	$(3\times)$ 24
	# of Channels	512	$(3\times)$ 64	—	—
	# of Layers L	1	$(3\times)$ 1	4	$(3\times)$ 4
Conv. Stage 5	Growth Rate k	—	—	60	$(3\times)$ 24
	# of Channels	512	$(3\times)$ 128	—	—
	# of Layers L	1	$(3\times)$ 1	4	$(3\times)$ 4
# of trainable parameters		5.8 M	4.2 M	2.0 M	2.0 M

F-scores are between 2 to 3 percentage points lower for the former models. This confirms the initial claim that using domain knowledge to build musically motivated CNN architectures is also beneficial for the task of instrument recognition. The findings in this experiment conform to the findings of the thesis regarding HPSS discussed in Chapter 3.

4.2.5.3 Varying Separation Factor β

In this set of experiments the models that perform classification using TSND as preprocessing stage are evaluated and compared. While the lengths of the vertical and horizontal median filtering are fixed as mentioned in Subsection 4.2.1.1, different values of the separation factor β were tested and the results are summarised in Table 4.3. In the left half of the table, metrics for multi-channel models, whose architectures follow Figure 4.3b, are shown, while in the right half, the results of the evaluation of multi-input models following Figure 4.3c are shown.

Table 4.2: Evaluation of models with different kernel shapes in the architecture as well as models whose internal convolutional stages are composed of DenseNets (dense arrangement of skip-connections) or regular convolutional layers (no skip-connections). The baseline model was used in both cases, i.e., no TSND is performed as preprocessing step.

F-score	Without DenseNets		With DenseNets	
	Square	Multiple	Square	Multiple
Piano	0.9605	0.9647	0.9615	0.9637
Violin	0.9154	0.9184	0.9063	0.9171
Viola	0.7915	0.7918	0.7686	0.7990
Cello	0.8868	0.8909	0.8814	0.8946
Horn	0.7379	0.7411	0.7038	0.7520
Bassoon	0.7518	0.8188	0.7596	0.7684
Clarinet	0.8169	0.8615	0.8191	0.8758
Micro	0.8737	0.8885	0.8669	0.8863
Macro	0.8372	0.8553	0.8287	0.8529
Weighted	0.8765	0.8894	0.8706	0.8881
# params	5.8 M	4.2 M	2.0 M	2.0 M

It is easy to realise that the multi-channel models have better performance regardless of the value of β. This should be due to the fact that using a multi-channel input allows the model to learn feature-maps directly using the information of every decomposed sound feature. When a multi-input framework is used, the convolutional stages for each type of input are isolated and therefore it is harder for the network to learn relations between feature-maps of distinct input heads during backpropagation training.

The separation factor that obtained the best overall performance was $\beta = 2.0$, but just for a marginal difference when compared to other multi-channel models. This was the value chosen for the rest of the experiments involving TSND, but using a different value should not considerably affect the results.

4.2.5.4 Smoothing Instrument Activations

A common problem of most frame-level classification models is that they are susceptible to estimating erroneous classes for a short sequence of frames. This means that it is common to find short duration segments where particular instrument activations are missing or that the instrument activations change from positive to negative during a few consecutive frames (high variance).

In order to smooth the time series composed of the binary instrument activations across time, a postprocessing step that consists of the application of a fixed-length median filtering was tested. A preliminary study testing odd-length

Table 4.3: Comparison of overall performance of the models using TSND with different values for the separation factor β. Multi-Channel are models following Figure 4.3b while Multi-Input are models following Figure 4.3c. The vertical median filtering was set to 11, which is approximately 475 Hz and the horizontal median filtering length was set to 21, which is 210 ms. The decision window is set to 710 ms. Only the averaged F-score values are shown.

Models	Multi-Channel			Multi-Input		
	micro	macro	weighted	micro	macro	weighted
$\beta = 1.0$	0.8724	0.8323	0.8755	0.8596	0.8115	0.8615
$\beta = 1.5$	0.8710	0.8325	0.8748	0.8496	0.8017	0.8559
$\beta = 2.0$	0.8803	0.8448	0.8825	0.8589	0.8163	0.8613
$\beta = 2.5$	0.8729	0.8327	0.8760	0.8481	0.8067	0.8516
$\beta = 3.0$	0.8726	0.8319	0.8749	0.8471	0.8009	0.8538
$\beta = 3.5$	0.8761	0.8362	0.8780	0.8117	0.7521	0.8022

filter lengths ranging from 3 to 49 was done for every evaluated model and the value of a fixed length of 19 samples was verified to consistently provide the most gain in performance on average across different models. Therefore, this value was then used when performing the smoothing technique.

Table 4.4 includes the F-scores of the top two proposed methods: one using the raw music spectrogram as input according to Figure 4.3a and the other using TSND and following the framework in Figure 4.3b. The table provides the F-scores when using the raw predictions (after thresholding) and when using smoothed predictions via a median filtering of length 19 across time for each instrument.

We can easily see that the smoothing technique improves the metrics of both models. As mentioned before, this is due to the fact that the median filtering postprocessing stage reduces quick variations in the instrument predictions across time. As a way of visualising the effects, Figure 4.5 shows the instrument activations for each of the 10 tracks contained in the MusicNet test set.

4.2.5.5 Usefulness of TSND for Instrument Classification

After performing the experiment to determine best choices of TSND and model parameters, it is possible to compare the performance of the instrument classification task performed by DNN classifiers when applied directly to the music spectrogram or to the decomposed sound estimates.

In addition to using the proposed musically motivated CNN classifier, which was inspired by the 3W-MdenseNet, two other traditional CNN architectures that are commonly used as classifiers due to their high performance in computer

Table 4.4: Comparison between using the raw instrument activations (direct output of the models after thresholding) and using smoothed activations, which is done through the application of a median filtering of a fixed length of 19 frames (the equivalent of 190 ms)

F-score	Best Overall Methods			
	Raw Predictions		Smoothed Predictions	
	Baseline	With TSND	Baseline	With TSND
Piano	0.9637	0.9635	0.9636	0.9631
Violin	0.9171	0.9138	0.9174	0.9147
Viola	0.7990	0.8011	0.8075	0.8044
Cello	0.8946	0.8869	0.8968	0.8914
Horn	0.7520	0.7174	0.7534	0.7270
Bassoon	0.7684	0.7897	0.7618	0.7884
Clarinet	0.8758	0.8413	0.8776	0.8442
Micro	0.8863	0.8803	0.8872	0.8821
Macro	0.8529	0.8448	0.8540	0.8476
Weighted	0.8881	0.8825	0.8891	0.8844

vision tasks. were also included in the comparison

One of the models is the VGG16 [Simonyan and Zisserman, 2015], which consists of stacks of multiple convolutional layers and max-pooling layers (down-sampling by a factor of two a total of 5 times) and two fully connected layers, with 4096 units and another with 10 (number of instrument classes). In summary, the VGG16 has 16 layers and a 14.7M trainable parameters after being adapted to perform the instrument classification task of this section. This architecture uses only square (3×3) filters and no skip-connections.

The other architecture is the ResNet50 [He et al., 2016], which is inspired by the VGG16, but uses more stacks of convolutional layers reaching a total of 50 layers internally. Also, the ResNet50 architecture has a path called the *residual path*, which consists of a path of forward skip-connections that skips the following 2 convolutional layers in the original path. Those residual connections (skip-connections) prevent vanishing of gradients during training. The ResNet50 has a total of 23.5 M trainable parameters after being adapted to perform the instrument classification task of this section. Explaining those architectures in detail is out of the scope of this thesis and the reader is referred to previous work [Simonyan and Zisserman, 2015; He et al., 2016] in order to know more about them.

Table 4.5 shows the F-scores of each evaluated model. First of all it is important to note that the proposed architecture obtains much higher performance for each instrument when compared to the other deeper classifiers. While using

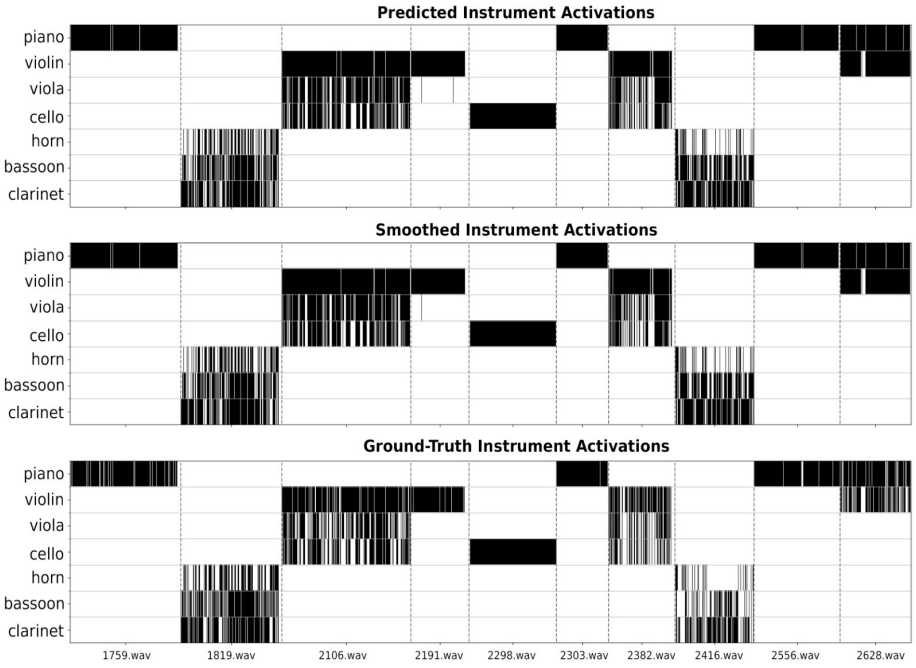


Figure 4.5: Visualisation of instrument activations across time. The activations are smoothed by applying a median filter to the binary instrument activation (after thresholding). Observe that the postprocessed models have short-time spurious activations eliminated; a good example is verifying that for track `2191.wav` the wrong short-duration viola activations are reduced after the smoothing approach.

a tenth of number of trainable parameters of a ResNet50, it surpasses the performance of the latter. This corroborates the methodology that it is essential to focus on building more efficient networks, whose architectures are adapted to a desired task using domain knowledge. However, it is important to note that performing the instrument classification using only data from MusicNet might be a simple problem to be directly addressed using deeper models, such as ResNet50 or $VGG16$. Those models need more data to not overfit to the training data. This might have happened in this case; more experiments using larger datasets and other types of instruments are necessary in order to draw a better conclusion.

Regarding TSND, it seems that it was not effective in helping the network to learn better timbre-discriminative features. While our perception of timbre is affected differently by transient and stationary parts of the sounds, having isolated transients and steady-state sounds as distinct and isolated inputs does not provide any benefit for training a neural network framework. When think-

ing more critically about this proposed methodology, it is possible to conclude that the chosen TSND procedure does not provide any new information to the framework, all the information used by the model was already in the original music spectrogram, and what the method essentially does is assigning each bin to one of the three inputs. While this could provide benefit to downstream tasks if we process each type of source separately, this type of decomposition is a transformation that essentially is transparent to the model since all the estimated sources are given back to the model as a multi-channel input. Providing a single music spectrogram as input is equivalent to providing disjoint parts of it as a multi-channel inputs. Therefore, there is neither significant improvement nor reductions of the performance.

Table 4.5: Evaluation of different CNN classifiers performing instrument activity detection on MusicNet dataset. The classifier architecture proposed in Subsection 4.2.1.2 is compared with VGG16 and ResNet50 architectures.

F-score	Classifier Architecture					
	Without TSND			With TSND		
	Proposed	VGG16	ResNet50	Proposed	VGG16	ResNet50
Piano	0.9637	0.9623	0.9431	0.9635	0.9600	0.9594
Violin	0.9171	0.9156	0.9065	0.9138	0.9086	0.9003
Viola	0.7990	0.7773	0.7377	0.8011	0.7699	0.7387
Cello	0.8946	0.8908	0.8674	0.8869	0.8825	0.8792
Horn	0.7520	0.7472	0.6885	0.7174	0.7349	0.7521
Bassoon	0.7684	0.7983	0.7456	0.7897	0.7722	0.7649
Clarinet	0.8758	0.8347	0.7640	0.8413	0.8062	0.8038
Micro	0.8863	0.8804	0.8491	0.8803	0.8691	0.8638
Macro	0.8529	0.8466	0.8076	0.8448	0.8335	0.8284
Weighted	0.8881	0.8825	0.8526	0.8825	0.8723	0.8665

4.3 Pitch-informed Instrument Assignment

While in Section 4.2 the task of instrument recognition was addressed with the goal of detecting instrument activations across time, in this section, the work in this section is focused towards processing polyphonic multi-instrumental recordings with the objective of assigning note-events to their corresponding instrumental source.

This task is known as **instrument assignment** (or **pitch streaming**) and when used along with an MPE method, allows multi-instrument transcription. Each note can not only have its pitch, onset and offset properly estimated, but the information regarding the instrumental sources that produce them can also

be recognised. This task was discussed in Section 2.3, where more information regarding this task along with a literature review is provided.

In summary, the proposal here is to address this task as a note-level instrument classification, where the main objective is to associate each note-event of a music signal to an instrument class. This method contrasts to other state-of-the-art instrument recognition approaches since it analyses each note-event individually while typical instrument recognition approaches usually address the instrument classification task on a frame-level basis, as the method proposed in Section 4.2, or on a clip-level basis [Han et al., 2017; Gururani et al., 2019].

Previous work has shown that the use of pitch information can help frame-level instrument recognition [Hung and Yang, 2018]. Inspired by this method, a framework that uses a main input carrying information about the music signal along with an auxiliary input encoding information of single note-events is proposed here. It is also shown that this approach can obtain good performance when the note-event information is predicted using state-of-the-art MPE algorithms [Thomé and Ahlbäck, 2017; Wu et al., 2019] rather than using ground-truth note labels. Therefore, the proposed method is designed as a modular framework, which can be combined with any MPE algorithm in order to obtain multi-instrumental pitch predictions.

The proposed method here was published in Lordelo et al. [2021a] but a more detailed explanation and discussion can be found in the next subsections.

Furthermore, throughout the thesis the strategy of building musically motivated CNNs by including kernels with multiple shapes in the network has proven to be an efficient way of applying domain knowledge for the tasks of source separation and instrument classification. In this section, the methodology is maintained by adapting from the proposed architecture for a frame-level classifier explained in Section 4.2, which, in turn, was a variation of the 3W-MDenseNet (explained in Subsection 3.2.6) for performing classification instead of source separation. The experiments in this section verify that this strategy can also improve instrument assignment performance.

4.3.1 Differences from Previous Work

One of the closest work to the proposed approach is Hung and Yang [2018], where a frame-level instrument recogniser is proposed using the CQT spectrogram of the music signal allied with the pitch information of the note-events as inputs. Here, the pitch annotations are also used to guide the instrument classifier, but the proposal in this thesis differs from Hung and Yang [2018] in the fact that classification is performed for each note-event individually, while Hung and Yang [2018] use the whole piano-roll at once to guide frame-level

instrument recognition. While they are able to obtain instrument activations leveraging from the pitch information, they do not stream the note events into their corresponding instruments. Similarly, the proposed approach explained in Section 4.2 also cannot associate specific notes-events to instrument classes. In fact, it predicts all instruments that are active in a target frame.

Moreover, in the task of score-informed source separation the main idea is to use the score information, usually in the form of a MIDI piano-roll, of the target instrument as a guide for the separation of that particular source. Also inspired by this task, the proposed method uses note-event information to address the problem of note-level instrument recognition as a "pitch-informed instrument assignment" task. Here, each note is considered as having constant pitch, onset and offset values and this information is used to guide the network for performing instrument recognition.

In order to do this, the whole pitchgram of a music clip is broken down into its constituent note-events, which are used separately as an auxiliary signal to condition the network to predict the correct instrument. Differently than other instrument recognition approaches, instead of just using the spectrogram of a clip to try to recognise the instruments that are active, the spectrogram along with the information of a specific note-event is used to recognise the unique instrument that is related to that note.

4.3.2 Proposed Method

In the proposed method, the same definition of note-events as in the MIREX MPE task[1] is used. Each note N is considered an event with a constant pitch f_0, an onset time T_{on} and an offset time T_{off}. Therefore, if a music signal has a total of M notes, any note N_i, with $i \in \{1, \cdots, M\}$, can be uniquely defined by the tuple $(f_0^i, T_{\text{on}}^i, T_{\text{off}}^i)$. In the experiments, two ways of obtaining this note information were used. The first is utilising ground-truth pitch labels provided by the employed dataset (MusicNet) [Thickstun et al., 2017] and the second using pitch estimates predicted by state-of-the-art MPE algorithms [Thomé and Ahlbäck, 2017; Wu et al., 2019]. The f_0 granularity is considered to follow the semitone scale, ranging from $A0$ to $G\sharp 7$ (MIDI $\#21 - 104$).

In this framework polyphony is allowed, so, most of the time more than a single note will be active, but the objective is to analyse each note of the audio signal separately in order to be able to assign an instrument class to it. This is done by using two inputs to the model: the main input $X(f, t)$, with f representing frequency and t representing time, is a time-frequency representation of a segment of the audio signal around the value of T_{on}, and an auxiliary input

[1] `https://www.music-ir.org/mirex/`

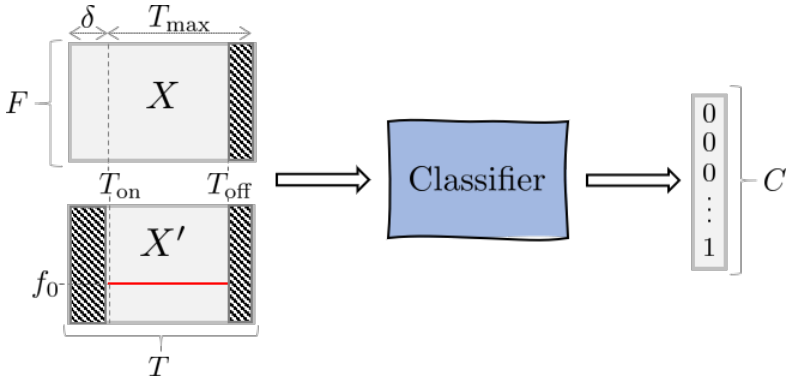


Figure 4.6: Overview of the proposed framework for instrument assignment. See Section 4.3.2 for the detailed explanation of the variables in the figure.

$X'(f,t)$, which carries information regarding a note-event (f_0, T_{on} and T_{off}). The two inputs are concatenated into a two-channel input $\mathbf{X}(f,t,c)$, where $c \in \{1,2\}$ represents the channel dimension, that is fed to the model. In Figure 4.6 an overview of the proposed framework is shown.

4.3.2.1 Main Audio Input

The main input is a time-frequency representation $X(f,t) \in \mathbb{R}^{F \times T}$ of a small clip of the music signal, where F is the number of frequency bins and T is the number of time frames. The clip is generated by first setting a maximum duration $T_{\max}$ for the note. Values of $T_{\max}$ ranging from 400 ms to 1 s were tested (see Subsection 4.3.6 for details) and 400 ms obtained the best results, so this value was kept for all of the other experiments. If any note N_i has a duration D_i greater than $T_{\max}$, i.e., $D_i = T_{\text{off}} - T_{\text{on}} > T_{\max}$, only its initial time span of $T_{\max}$ seconds is considered.

Next, for every note N_i, $X_i(f,t)$ is constructed by picking a segment of duration $T = T_{\max} + \delta$ from the original music signal starting from $T^i_{\text{on}} - \delta$, where δ is a small interval to take into account deviations between the true onset value and the annotated (or estimated) value. The inclusion of the extra window of δ from the music signal also helps the convolutional layers since it brings some context of the signal before the note onset time. The value of δ was fixed to 30 ms after initial tests. Lastly, if the note duration D_i is less than $T_{\max}$, we set the values of $X_i(f, t > D_i + \delta)$ to zero, where $D_i = T^i_{\text{off}} - T^i_{\text{on}}$.

4.3.2.2 Auxiliary Note-Related Input

In order to provide the note-event information to guide the model to recognise its corresponding instrument, an auxiliary input is necessary. It serves as a

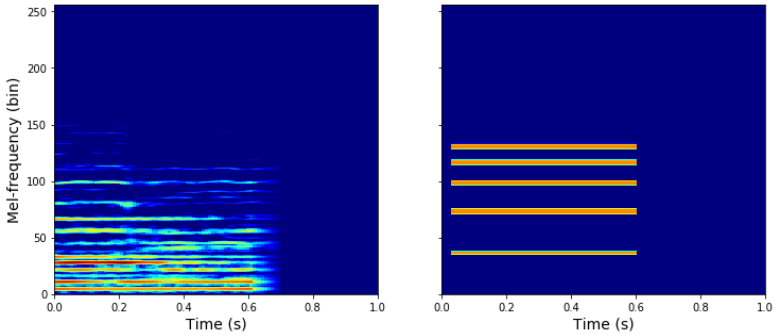


Figure 4.7: Pair of inputs using 256 Mel-frequency spectrogram. On the left is depicted $X(f,t)$, where three pitches are simultaneously active (MIDI # 58, 62 and 74) and on the right $X'(f,t)$, where the note-event with pitch # 74 is represented using a harmonic comb of $H = 5$. In this example, $D_i = 600\,\text{ms}$ and $T_{\text{max}} = 1\,\text{s}$.

"salience function" to the model, where only the frequency bins related to the fundamental frequency of a note and its harmonics will be 'highlighted'.

Effectively, the auxiliary input $X'(f,t) \in \mathbb{R}^{F \times T}$ is defined as a harmonic comb representation using the pitch value f_0 as the first harmonic[2], such that,

$$X'(f,t) = \begin{cases} 1, & \text{if } 2^{-1/24} h f_0 < f \leq 2^{1/24} h f_0 \quad \text{and} \quad T_{\text{on}} \leq t \leq T_{\text{off}} \\ 0, & \text{otherwise} \end{cases}, \quad (4.12)$$

where $h = \{1, 2, 3, \cdots, H\}$ with H being the total number of harmonics in the representation. Multiple values for H were tested (see Section 4.3.5). Note that, a tolerance of half a semitone for each harmonic value is used when constructing $X'(f,t)$ from the magnitude STFT. Furthermore, even though this representation starts as binary, the linear frequency is converted to Mel-frequency, which can result in having non-binary values due to the linear-to-Mel transformation. Also, it is important to note that the values of X' before T_{on} and after T_{off} are set to zero, indicating that the note-event that should be classified starts at T_{on} and ends at T_{off}. In Figure 4.7 we show an example of a pair of inputs for the framework.

4.3.2.3 Output

The note-level instrument assignment task is addressed as a multi-class single-label classification task. Given a pair of inputs $\mathbf{X}$, the objective is to classify them as belonging to one of C instrument classes. Therefore, a deep neural network receives $\mathbf{X}$ as input and outputs a C-dimensional vector $\hat{y}$ with a final

[2]Considering the definition that f_0 corresponds to the first harmonic.

softmax activation function, ensuring the values of $\hat{y}$ represent probabilities that sum up to 1. The model is trained using the cross-entropy loss. At inference time, the class corresponding to the dimension with the highest value in $\hat{y}$ is predicted. See Section 4.3.3 for details regarding the network architecture.

It is worth noting that in the cases where two or more instruments are playing the same pitch simultaneously, the small differences between the notes' onset and offset values can generate different inputs $\mathbf{X}$. Thus, it would still allow the instrument assignment task to be properly executed as a single-label classification scenario. However, when the pitch, onset and offset values of notes from different instruments exactly match, this system will consider them as a single note and only a single instrument will be estimated. This case rarely happens in real-world scenarios for many musical styles. For instance, from a total of 1089540 labelled note-events in the MusicNet dataset, only 10140 had the same pitch, onset and offset values, which is around 0.9%. For the experiments, we have considered note-events that were performed by a single instrument, and discarded the events that were concurrently produced (in terms of the same pitch, onset, and offset times) by multiple instruments. As a proof of concept, this is not considered a severe limitation for the framework and multi-labelled approaches are left as future work.

Also, two choices for time-frequency representations were compared: the Mel-frequency Short-Time Fourier Transforms (Mel-STFT) and the CQT. The mel-STFT is constructed by first forming X' according to Equation (4.12) and then applying the mel-frequency transformation while the CQT is also straightforward to compute by using corresponding CQT-bins based on the corresponding MIDI value of f_0.

We consider two ways of getting note-event information in the experiments. One is using human-labelled ground-truth note labels provided by MusicNet, but since in real-word applications, it is not possible to have direct access to this type of label, we also evaluated the performance of instrument assignment task by using note-events estimated by two state-of-the-art MPE algorithms: Thomé and Ahlbäck [2017] and Wu et al. [2019].

4.3.3 Musically Motivated Classifier Architecture

The architecture used in the experiments for instrument assignment is the same architecture used for the frame-level instrument recognition with few hyperparameter modifications. The architecture is based on the utilisation of vertical, horizontal and square kernel shapes and dense arrangements of skip-connections via the usage of DenseNets rather than regular stacks of convolutional layers. For a detailed explanation of the CNN architecture the reader is referred to

4.2.1.2.

Regarding the small changes performed in the architecture, in Section 4.2 raw linear-frequency scaled magnitude spectrograms were used as input to the model. Here, experiments using either Mel-STFT or CQT are implemented. This fact decreases the frequency resolution of the input representation and reduces the corresponding input dimension. In comparison, in Section 4.2 513 frequency bins are used and here either 256 Mel-bins or 115 CQT bins are used. Due to this fact initial tests suggested that smaller vertical kernels were getting slightly better results. Therefore, the vertical kernel shapes were changed from (1×11) to (1×9) and a similar follow-up change was performed on the horizontal kernels, i.e., changed from (11×1) to (9×1). The reason for the latter was because there was no considerable effect on performance and by doing so, each branch of the multi-branch convolutional stages would have the same number of trainable parameters. The square kernels kept the shape of (3×3)

Other hyperparameter choices that are different is the fact that this model uses a softmax activation layer as the final layer and uses $D = 4$ multi-branch convolutional stages and a single hidden fully connected layer of 64 neurons. A summary of the architecture adopted for the experiments in this section appear in Figure 4.8.

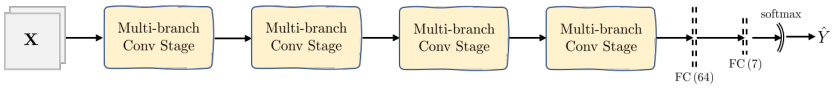


Figure 4.8: Classifier architecture. Each of the 3 DenseNets in a multi-branch convolutional stage has growth rate $k = 25$ and $L = 4$ layers. Relu is used as activation function after each convolutional layer. The number of trainable parameters is 1.2 million his time.

4.3.4 Dataset

The dataset utilised in the experiments is the MusicNet dataset [Thickstun et al., 2017], which is the largest publicly available dataset with non-synthesised data that is strongly labelled (with frame-level instrument annotations and note onset, offsets and pitch annotations). This dataset was also utilised in the experiments related to the frame-level recognition in Section 4.2, so the reader is pointed to Subsection 4.2.2 for more details about this dataset.

In the experiments of this section the same procedure of performing a 7-class instrument classification using the following classes: *piano, violin, viola, cello, french horn, bassoon* and *clarinet* was kept since the other 4 instrument classes are not in the official test set provided by the authors.

In Table 4.6 it is possible to see the statistics of the note-event information in

MusicNet. Observe that the dataset is heavily biased towards piano and violin given their frequent presence in Western classical music recordings.

Set	Piano	Violin	Viola	Cello	Horn	Bassoon	Clarinet	Harps.	Bass	Oboe	Flute	Total
Train	628549	197229	88446	89356	10770	13874	22873	4914	3006	8624	8310	1075951
	58.4%	18.3%	8.2%	8.3%	1.0%	1.2%	2.1%	0.5%	0.3%	0.8%	0.8%	100%
Test	5049	3238	842	1753	557	873	1277	0	0	0	0	13589
	37.2%	23.8%	6.2%	12.9%	4.1%	6.4%	9.4%	0	0	0	0	100%

Table 4.6: Statistics of the note-events information in MusicNet across train and test sets.

4.3.5 Experimental Setup

In all experiments the original train/test split provided by MusicNet was used with the original sampling frequency of 44100 Hz. For experiments that involved the computation of Short Time Fourier Transform (STFT) we used Blackman-Harris windows of 4096 samples to compute the Discrete Fourier Transform (DFT). The hop size was always set to 10 ms in every experiment. The hop size was kept the same as in the experiments in Section 4.2, but the STFT window was now changed to a higher number since it is beneficial to have a higher frequency resolution before converting to the mel-frequency scale.

From the training set, 5% of the notes of each class were picked and a validation set was created. The models were trained using the Adam optimiser with an initial learning rate of 0.001, which was reduced by a factor of 0.2 if the cross-entropy loss stopped improving for 2 consecutive epochs on the validation set. If no improvement happened after 10 epochs, the training was stopped early. The experiments were performed using the Tensorflow/Keras Python package.

The classification performance was evaluated by computing the note-level F-score (F_s), which was already explained in Section 4.2.3 with Equation (4.11).

For the cases when the instrument assignment is done on top of MPE algorithms, 2 groups of metrics that are generated following the MIREX evaluation protocol for the music transcription task are provided. In the first group, an estimated note is assumed correct if its onset time is within 50 ms of a reference note and its pitch is within a quarter tone of the corresponding reference note. The offset values are ignored. In the second group, on top of those requirements, the offsets are also taken into consideration. An estimated note is only considered correct if it also has an offset value within 50 ms or within 20% of the reference note's duration around the original note's offset, whichever is largest. After all notes are verified, the F-score is computed note-wise across time and the average value (micro F-score) is provided here. This evaluation method was

computed using the `mir_eval.transcription`[3] toolbox.

4.3.6 Results

4.3.6.1 Effects of the Kernel Shapes

First, the effects of the inclusion of multiple kernel shapes in the architecture of the CNN are analysed. Table 4.7 compares 3 versions of the model: one that uses only square filters in a single branch; a version using the branched structure, but with (3×3) kernels in each; and another model with the proposed multi-branch structure with horizontal, square, and vertical kernels. For the single-branched case the growth rate of the DenseNets was increased to 57 channels in order to keep the number of trainable parameters of the network close to 1.2 million.

Analysing the results it is possible to realise that the addition of new kernel shapes improved the average F-score across all classes. Regarding each instrument class, one can say that for string instruments (piano, violin, viola and cello) there is a gain in performance, while for non-string instruments (horn, bassoon, clarinet) the performance either drops or remains with a negligible gain if compared to the models that used only square filters. This suggests that the inclusion of vertical the kernel helped the model in learning the percussive characteristics of the timbre of string musical instruments. Similarly, horizontal filters, helped the model in learning harmonic-related timbre-discriminative features.

Table 4.7: Instrument assignment performance based on the kernel shapes used in network. The metrics shown are the F-scores achieved by each instrument class and the average value (macro F-score) across all instruments.

Kernel	Piano	Violin	Viola	Cello	Horn	Bassoon	Clarinet	**Macro**
(3×3)	0.994	0.936	0.757	0.954	**0.826**	0.864	0.954	0.898
$3 \times (3 \times 3)$	0.995	0.939	0.764	0.945	0.819	**0.896**	0.965	0.903
Multiple	**0.997**	**0.944**	**0.775**	**0.958**	0.810	0.879	**0.967**	**0.904**

4.3.6.2 Evaluation of Different Input Sizes

Different values for the input size (decision window) were also evaluated. More specifically, multiple values for $T_{\max}$, which is the maximum valid window of analysis for a note event, were compared. The results are shown in Table 4.8. It is possible to see that the shortest input size of 400 ms obtained the best results. It is believed that this is due to the fact that the average duration of a note-event in the test set of MusicNet is 260 ms and the 90th percentile is 0.464 ms.

[3] `https://craffel.github.io/mir_eval/`

So, the value of 400 ms is already good enough to represent the vast majority of the notes. Moreover, when the analysed note event is longer than 400 ms, the 400 ms initial window contains most of the important timbre-discriminative features for the model. This seems to go in line with human perception of timbre, which according to music psychology studies [McAdams et al., 1995], the most important information for humans to recognise instruments is around the onset of the notes.

Table 4.8: Comparison between different values for the input size. The values shown in first column represent the maximum valid note duration $T_{\max}$. The metrics are the F-scores achieved by each class and the average value (macro F-score) across all instruments .

$T_{\max}$	Piano	Violin	Viola	Cello	Horn	Bassoon	Clarinet	**Macro**
400 ms	**0.997**	**0.944**	**0.775**	**0.958**	0.810	0.879	**0.967**	**0.9043**
600 ms	0.996	0.942	0.771	0.954	**0.826**	**0.881**	0.959	0.9043
800 ms	0.996	0.944	0.772	0.957	0.814	0.868	0.965	0.9022
1 s	0.997	0.931	0.740	0.954	0.742	0.871	0.948	0.8832

4.3.6.3 Auxiliary Input and Types of Representations

To test the importance of the auxiliary input and how its modification would affect the performance of the model, a version of the model using only the main mel spectrogram input and versions using different numbers of harmonics H in the auxiliary input (from $H = 1$ to $H = 5$) were compared. Also two types of input representation for the model were tested: the Constant-Q Transform (CQT) and the mel-frequency spectrogram. The CQT was computed using 12 bins per octave and a total of 115 bins starting from $G\sharp 0$ (MIDI #20). The mel-frequency spectrogram was computed by a linear transformation of an STFT onto a mel-scaled frequency axis, using 256 mel-bins. The results are provided in Table 4.9.

Analysing the results, it is possible to say that the auxiliary input is extremely necessary for the framework. Without it, the average F-score only reaches 60.9%, while with it the performance improves up to 90.4%. Apart from piano, all other classes have a large decrease in performance when the auxiliary input is excluded from $\mathbf{X}$. It is believed that the results for the piano class continue to be high not only because of the MusicNet bias towards piano, but also because some recordings of the test set are solo piano recordings, which facilitates the classification of piano notes when analysing only the main input signal due to the absence of other classes. Regarding the number of harmonics used in the auxiliary input, in general, the CQT seems to work best with few

Table 4.9: Evaluation of instrument assignment task when using CQT or Mel spectrograms as input representation for the network as well as a comparison between models trained with no auxiliary input and models trained with different number of harmonics in the auxiliary input. This experiment was performed using $T_{max} = 400\,\text{ms}$

Main Input	Aux Input	Piano	Violin	Viola	Cello	Horn	Bassoon	Clarinet	**Macro**
CQT	—	0.960	0.732	0.116	0.725	0.512	0.296	0.681	0.575
	$H = 1$	**0.994**	0.934	0.763	0.955	0.783	**0.888**	0.950	**0.895**
	$H = 2$	0.993	0.939	0.771	**0.960**	**0.785**	0.858	0.929	0.891
	$H = 3$	0.992	**0.946**	**0.772**	0.957	0.754	0.884	**0.952**	0.894
	$H = 4$	0.993	0.938	0.766	0.958	0.784	0.869	0.950	0.894
	$H = 5$	0.993	0.939	0.767	0.952	0.769	0.874	0.949	0.892
Mel STFT	—	0.967	0.742	0.222	0.730	0.607	0.306	0.690	0.609
	$H = 1$	0.996	0.939	0.759	0.958	0.780	0.867	0.958	0.895
	$H = 2$	0.994	0.945	0.779	0.956	0.809	0.864	0.946	0.899
	$H = 3$	**0.997**	0.944	0.775	**0.958**	0.8104	0.879	**0.967**	**0.904**
	$H = 4$	0.996	0.935	0.747	0.945	**0.839**	**0.891**	0.960	0.902
	$H = 5$	0.996	**0.947**	**0.783**	0.954	0.801	0.876	0.954	0.902

harmonics, while the Mel-STFT prefers higher values. A possible explanation for this is the fact that it is harder to represent higher order odd harmonics on the CQT using a log-frequency resolution of 12 bins per octave. However, more experiments are needed in order to better investigate this assumption.

4.3.6.4 Streaming of Multi-Pitch Estimations

Once it was verified that the model obtains impressive performance when original ground-truth note-event information is used, the classifier was evaluated in a more realistic environment, where no note-event labels were readily available. Frame-level pitch values were estimated using two third-party MPE algorithms [Thomé and Ahlbäck, 2017; Wu et al., 2019]. For the algorithm in Thomé and Ahlbäck [2017], an implementation from the original authors was obtained, while an implementation of Wu et al. [2019] is available via the project Omnizart[4]. Both MPE algorithms were performed on the multi-instrumental music recordings to obtain note-event information such as pitch, onset and offset. The pair of inputs of the classifier were generated based on the estimated data.

It is important to observe that errors in the MPE estimation will be carried over to the instrument assignment task. If a note is wrongly estimated, no ground-truth class for the instrument assignment task exists, so it is hard to evaluate the results in the same way done for the other experiments. Therefore, in this experiment, transcription metrics explained in the last paragraph of

[4] `https://github.com/Music-and-Culture-Technology-Lab/omnizart`

Table 4.10: Transcription results when considering only the estimated pitches, onset times and instrument classified by the proposed model using Ground-Truth (GT) labels for note-event information and when using two different MPE methods instead. MPE-1 is Thomé and Ahlbäck [2017] and MPE-2 is Wu et al. [2019]. In the row "MPE-only", no instrument assignment is done, the multi-pitch estimates are evaluated using the reference ground-truth notes ignoring the instrument annotations, i.e., only taking into account pitches and onsets.

Instr.	Onset Only		
	GT	MPE-1	MPE-2
MPE-only	1	0.633	0.480
piano	0.997	0.745	0.451
violin	0.942	0.529	0.499
viola	0.775	0.366	0.308
cello	0.954	0.596	0.570
horn	0.804	0.460	0.429
bassoon	0.874	0.473	0.373
clarinet	0.967	0.616	0.456

Section 4.3.5 are used. The results appear in Table 4.10 ignoring note offset values and in Table 4.11 considering them.

In the first row of each table (MPE_only) the metrics are computed for each MPE algorithm individually, i.e., f-scores of all the estimated note-event information (pitch, onset and offset (Table 4.11)) with respect to the ground-truth note-event values provided by MusicNet without considering any instrumental source since MPE algorithms do not handle instrument information. Then, the evaluation on each instrumental source after using the proposed instrument assignment method on top of the raw estimated pitches and onsets is provided. Given the limitations of each MPE method used, it is possible to see that the approach can generate good multi-instrument transcriptions.

4.4 Conclusions

Research related to instrument recognition performed during the course of the PhD project was presented in this chapter.

In Section 4.2 a deep learning-based frame-level instrument recognition approach was proposed while in Section 4.3 the model was adapted to perform note-wise instrument assignment instead. In both cases the proposed methods were based on a CNN classifier, whose architecture was built to foster efficiency (high performance and low number of trainable parameters) by applying do-

Table 4.11: Transcription results considering only the estimated pitches, onset, offset times, and instrument classified by the proposed model when using Ground-Truth (GT) labels for note-event information and when using two different MPE methods instead. MPE-1 is Thomé and Ahlbäck [2017] and MPE-2 is Wu et al. [2019]. In the row "MPE-only", no instrument assignment is done, the multi-pitch estimates are evaluated using the reference ground-truth notes ignoring the instrument annotations, i.e., only taking into account pitches, onsets and offset times.

Instr.	Onset + Offset		
	GT	MPE-1	MPE-2
MPE-only	1	0.423	0.200
piano	0.997	0.497	0.196
violin	0.942	0.381	0.225
viola	0.775	0.227	0.116
cello	0.954	0.507	0.258
horn	0.804	0.232	0.166
bassoon	0.874	0.193	0.130
clarinet	0.967	0.344	0.165

main knowledge to choose better kernel shapes for the convolutions and a dense arrangement of skip-connections to avoid learning redundant feature-maps. It was shown that the proposed musically motivated CNN architecture is beneficial for both tasks.

The investigation of whether providing separated transient, stationary and residual sounds to a deep-learning model is beneficial for performing instrument recognition, has shown to bring no overall improvement to the framework. The chosen approach to estimate such type of sources based in a TSND preprocessing step estimates disjoint spectrogram masks that are used to generate 3 disjoint spectrograms, which are then used as joint inputs to the model. The disjoint separation along the channel dimension of the input seems to be ignored by the model and have no effect in performance. Using other type of transient-stationary separation seems to be necessary ideal in order to draw better conclusions.

Regarding the instrument assignment task, a proposal of addressing it as by a deep learning-based note-informed instrument recogniser was detailed in Section 4.3. The proposed framework uses the pitch, onset and offset information of the note-events as a way of guiding the classification. It was verified that the proposed approach is able to assign note-events into 7 different classes of instruments with a macro-averaged F-score of 90.4%. Furthermore, the proposed

approach could also successfully classify notes provided by an MPE algorithm, which permits generating multi-instrument transcriptions.

As future work a suggestion is to investigate more deeply the interaction of the instrument assignment method with other MPE algorithms, as well as, how to leverage the two proposed models of this section in order to build better frame and clip-level instrument recognisers.