INVESTIGATING CNN-BASED INSTRUMENT FAMILY RECOGNITION FOR WESTERN CLASSICAL MUSIC RECORDINGS

Michael Taenzer[1], Jakob Abeßer[1], Stylianos I. Mimilakis[1], Christof Weiß[2], Meinard Müller[2], and Hanna Lukashevich[1]

[1] Fraunhofer IDMT, Ilmenau, Germany
[2] International Audio Laboratories Erlangen, Germany
michael.taenzer@idmt.fraunhofer.de

ABSTRACT

Western classical music comprises a rich repertoire composed for different ensembles. Often, these ensembles consist of instruments from one or two of the families woodwinds, brass, piano, vocals, and strings. In this paper, we consider the task of automatically recognizing instrument families from music recordings. As one main contribution, we investigate the influence of data normalization, pre-processing, and augmentation techniques on the generalization capability of the models. We report on experiments using three datasets of monotimbral recordings covering different levels of timbral complexity: isolated notes, isolated melodies, and polyphonic pieces. While data augmentation and the normalization of spectral patches turned out to be beneficial, pre-processing strategies such as logarithmic compression and channel-energy normalization did not lead to substantial improvements. Furthermore, our cross-dataset experiments indicate the necessity of further optimization routines such as domain adaptation.

1. INTRODUCTION

In classical music, there are compositions for a variety of distinct instrumentations. Chamber music, for example, comprises ensembles of different size, which often consist of instruments from a specific instrument family such as woodwinds, brass, piano, vocal, or strings. Typical examples are brass quintets, piano duos, choirs, or string quartets. While the automatic classification of such *monotimbral* recordings w.r.t. the instrument family is a simplification of general instrument recognition, it still constitutes a challenging task. Successfully tackling this problem could help to organize and browse classical music collections. Furthermore, recognizing instrument families from monotimbral recordings constitutes the first step towards handling more complicated scenarios such as orchestra recordings, where we often find passages featured by specific instrument families.

In this paper, we approach the task of music instrument family recognition from classical music recordings. For our experiments, we consider three scenarios using datasets of monotimbral recordings with different levels of timbral complexity. We start with analyzing recordings of isolated notes (`IN`) played by an individual instrument. Furthermore, we test on isolated, monophonic melodies (`IM`) with a natural variety of note durations. For the third scenario, we consider monotimbral, mostly polyphonic music recordings (`MP`), where one or more instruments of the same instrument family are playing simultaneously. In Section 3, we describe these datasets in detail.

To approach the instrument family recognition task, we make use of a state-of-the-art instrument recognition algorithm [8] based on convolutional neural networks using spectrogram segments as input. As our main contributions, we apply this method to the instrument family scenario. For the `MP` scenario, we investigate how the model performance can be improved using strategies for data augmentation and pre-processing. We systematically test the generalization capability of the trained models to previously unseen datasets in a sequence of cross-dataset experiments.

2. RELATED WORK

Traditional algorithms for automatic instrument recognition (AIR) rely on audio features measuring instrument-specific timbral properties of music signals. Fuhrmann [4] provides a comprehensive overview of such techniques. As an example with a focus on classical music, Eggink & Brown [3] propose a system to recognize five wind and string instruments based on partial frequency and magnitude features combined with a Gaussian classifier.

Due to the rapid proliferation of deep-learning techniques, most recent publications mainly focus on data-driven algorithms, which are the focus of this literature review. These algorithms are trained to learn a direct mapping from low-level signal representations such as mel-spectrograms to higher-level attributes such as instrument labels. While data-driven approaches require less domain knowledge, they usually need large amounts of training data in order to learn models that generalize well to unseen

datasets. However, popular AIR datasets such as MedleyDB [1], IRMAS [2], or MusicNet [21] are still of limited size. As a consequence, authors often apply data augmentation techniques such as pitch shifting [7] to virtually enlarge the number of audio files.

Concerning the generalization capability, AIR approaches based on deep neural networks (DNNs) show good performance on particular datasets, but cross-dataset experiments (as we present in Section 5.2) are rarely performed. Such experiments are crucial for better understanding to which extent DNN models generalize to unseen datasets with different characteristics. In the related field of audio event recognition, researchers often observe this limitation of data-driven algorithms and suggest additional domain adaptation steps [5]. Another challenge is the entanglement between perceptual attributes such as pitch and timbre in spectrogram representations. Lostanlen et al. [14] propose weight-sharing strategies for DNN models in order to derive pitch-invariant representations, which still maintain good timbre discriminability.

Typical model architectures used in recently proposed AIR systems are convolutional neural networks (CNNs) [6–9, 12, 17, 20] and hybrid convolutional-recurrent neural networks (CRNNs) [7]. Most of the CNN architectures comprise several convolutional layers for feature learning and a set of dense layers for classification. Han et al. [8] proposed such a CNN architecture to recognize the predominant instrument in polyphonic and multitimbral recordings. The authors evaluate different late-fusion techniques to aggregate frame-level model predictions in order to obtain song-level instrument labels. This model has been used and extended in recent AIR literature [6, 20]. Takahashi et al. [20] show in a comparative experiment that using horizontal and vertical filter shapes instead of symmetrical ones improves recognition performance, but requires more training time. Hung & Yang [9] tested an alternative CNN model, which includes residual blocks with additional skip connections to allow for reducing the vanishing-gradient problem during training.

Concerning the input representation, most DNN-based AIR systems process mel-spectrogram segments (*patches*). As alternative, Hung & Yang test constant-Q spectrograms and harmonic-series features as input to the models [9]. Li et al. [12] propose an end-to-end-learning approach using a CNN architecture that directly processes raw audio data. Hung & Yang [9] show that using score information as an additional cue leads to small improvements in the frame-level recognition of seven classical instruments.

3. DATASETS

In this section, we describe three datasets that we use for our instrument family recognition experiments. Regarding the level of difficulty, the isolated-note scenario (`IN`) constitutes the simplest task represented by the Studio On Line Dataset (`DB-SOL`) presented in Section 3.1. A scenario with increased level of difficulty comprises isolated melodies (`IM`), represented by the University of Rochester Multi-modal Music Performance Dataset (`DB-URMP`) described in Section 3.2. As our most complicated scenario, we consider monotimbral polyphonic recordings of classical music realized in the Monotimbral Classical Dataset (`DB-MTC`), which comprises recordings of monotimbral, mostly polyphonic classical pieces (`MP`, see Section 3.3). Table 1 summarizes the properties of the three datasets.

Table 1: Number of audio files, spectral patches, and average patches per file for each dataset and experiment.

Dataset	Files	Patches	Patches / file (avg)
Original Datasets (with silent patches)			
`DB-MTC`	50	38078	762
`DB-MTC`$^+$	400	304624	762
`DB-URMP`	149	33693	226
`DB-SOL`	20604	225273	11
Experiment 1 (silent patches removed)			
`DB-MTC`	50	34163	683
`DB-MTC`$^+$	400	281841	705
Experiment 2 (3 classes, silent patches removed)			
`DB-MTC`	30	20900	697
`DB-URMP`	149	31236	210
`DB-SOL`	20604	202486	10
`DB-M/U/S`	20783	254622	12

3.1 Studio On Line Dataset (`DB-SOL`)

The Studio On Line dataset [1], recorded in 2002 at IRCAM (Paris), comprises over 25000 isolated note recordings from 16 different instruments covering the instrument families woodwinds, brass, and strings. Recognizing the instrument family of such isolated note recordings (`IN`) constitutes a relatively simple scenario since there is no spectral overlap of multiple notes. However, the large variety of instrument playing techniques—in particular, frequency modulation techniques such as vibrato and trill—makes the recognition task more complex. For our experiments, we discarded recordings from `DB-SOL` that only comprise mechanical instrument sounds without a clear pitch as well as breathing and speaking sounds.

3.2 University of Rochester Multi-modal Music Performance Dataset (`DB-URMP`)

The University of Rochester Multi-modal Music Performance (URMP) Dataset [11] was originally published to study audio-visual music performance analysis. The dataset comprises 44 ensemble pieces including duets, trios, quartets, and quintets, most of which are arrangements of popular classical pieces. For all pieces, multitrack recordings are available with a total of 149 isolated instruments tracks. Within each track, one melody instrument from the families woodwinds, brass, and strings is recorded in isolation. We use these individual tracks as the basis for our isolated-melodies (`IM`) scenario.

[1] Freely available as part of the Orchids software at `http://forumnet.ircam.fr/product/orchids-en/`. In [13], the dataset was used for evaluating an instrument recognition system.

3.3 Monotimbral Classical Dataset (DB-MTC)

To test the instrument family recognition task on a realistic scenario, we compiled a dataset consisting of 50 tracks from commercial recordings. The data comprises mostly polyphonic classical pieces composed for instruments of one family. For each of the five families, we included ten audio files, each from a different CD. The total duration in minutes for each instrument family is 63.2 (woodwinds), 37.6 (brass), 75.1 (piano), 51.4 (vocal), and 82.8 (strings).

The woodwind class mainly comprises chamber music works such as wind quintets by Cambini, Danzi, Hindemith, Nielsen, and Reicha, as well as a quartet by Rossini and a sextet by Janacek. Furthermore, we consider a serenade by W. A. Mozart, an excerpt from Dvořak's Ninth symphony (*New World*), and a partita for wind ensemble by Krommer. We are aware of the problem that wind ensembles often include a french horn, which is a brass instrument. While this is a typical situation in classical music, it might influence our recognition experiments. For the brass selection, we use brass ensemble music for five to ten players. We consider pieces by ten different composers, played by Canadian Brass, German Brass, Mnozil Brass, and other ensembles. Into the piano class we placed solo sonatas and fugues by Beethoven, Berg, C. P. E. Bach, and others, played by different pianists. Regarding vocal music, we include music for solo voice—such as Berio's *Sequenza III*—and choirs. The choir pieces comprise a renaissance composition by Allegri, romantic pieces by Bruckner and Janacek, modern pieces by Ligeti and Scelsi, and more. The strings class consists of several string orchestra pieces by Barber, Hindemith, Lutosławski, Penderecki, and Rawsthorne. Additionally, we include chamber music such as string quartets, a quintet by Schubert and a sextet by Brahms. [2]

4. SYSTEM OVERVIEW

In the following, we present our system for instrument family recognition, which consists of three main components. The first component (Section 4.1) transforms the audio signal of a music recording into a mel-based time–frequency representation. The second component (Sections 4.2 and 4.3) applies pre-processing techniques such as normalization or compression to the time–frequency representation. The third component (Section 4.4) consists of a CNN that outputs class probabilities and is trained in a supervised fashion. Figure 1 summarizes the main processing steps together with additional details regarding the network architecture (second and third columns of the figure).

4.1 Mel-spectogram Representation

For computing the time–frequency representation of the recordings, we follow the work by Han et al. [8]. We resample the audio signals to a sample rate of $f_s = 22050$ Hz and compute the mel-spectrogram [3] using 128 mel-bands, a hop size of 512 samples, and a window size of 1024 samples. Then, we normalize the magnitude in each frequency band of the mel-spectogram via dividing by the number of mel-bands. For each recording, we further segment the resulting mel-spectrogram representation into time–frequency patches with a length of 43 frames (approx. one second) with an overlap of 21 frames (approx. 0.5 seconds). This results in a tensor $X \in \mathbb{R}^{N \times 43 \times 128}$, where N indicates the total number of computed patches. In order to remove potential silent parts in the recordings, we discard a patch as soon as the mean of its magnitude values is below 5% of the entire file's maximal magnitude.

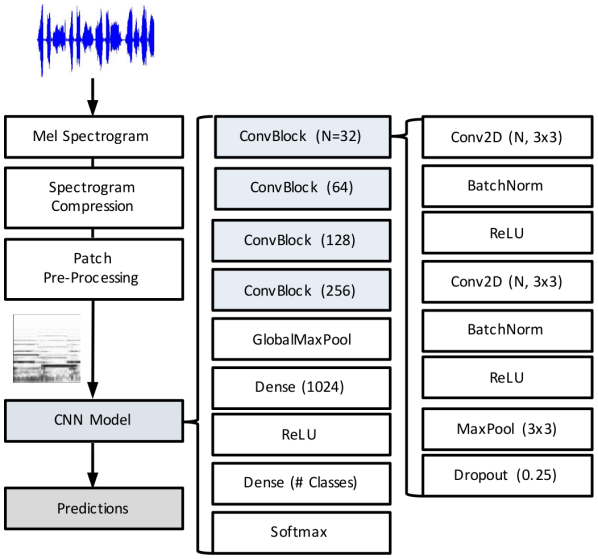

Figure 1: Reference model proposed by Han et al. [8] with slight modifications as discussed in Section 4.4. Spectrogram patches are processed by successive pairs of convolutional layers followed by batch normalization and ReLU activation function, max pooling (MaxPool), and global max-pooling (GlobMaxPool).

4.2 Spectrogram Dynamic Range Compression

Classical music recordings commonly exhibit a large dynamic range. To account for this, we investigate the effect of pre-processing strategies for compressing the dynamic range of the mel-spectrograms. In the following, we compare four different approaches for dynamic compression.

The first strategy, denoted as `NO`, does not apply any dynamic range compression. In this case, we directly use the mel-spectrogram as input to the model. The second strategy applies logarithmic compression defined by $X \leftarrow \log(1 + \gamma X)$. For our experiments, we consider two settings with $\gamma = 1$ (denoted as `LC 1`) and $\gamma = 10000$ (denoted as `LC 10000`), respectively. As the fourth strategy, we employ Per-Channel Energy Normalization (`PCEN`) proposed in [22] and further studied in [15]. `PCEN` ap-

[2] Due to copyright issues, we cannot publish the audio files. Instead, we publish the spectrogram patch tensors and corresponding targets to allow for reproducibility of our experiments under `https://doi.org/10.5281/zenodo.3258829`.

[3] We use the implementation from `librosa` (`https://librosa.github.io/librosa/`), version 0.6.2.

plies a first-order Infinite Impulse Response (IIR) filter, which controls the gain of the spectral representation, followed by dynamic range compression [22]. In principle, PCEN enhances prominent spectral characteristics (such as onsets), while attenuating low-energy frequency bands that are correlated with reverberation or corrupted by noise [22]. As the main benefit, the resulting spectral representation is robust against effects of reverberation and additive noise. For our experiments, we use PCEN as implemented in the librosa[3] Python library with default parameters (gain=0.98, bias=2, power=0.5, time_constant=0.4).

4.3 Patch Pre-processing

After the dynamic range compression step of the mel-spectrogram, we apply another pre-processing technique in order to normalize the spectral patches before we feed them to the CNN model. For this *patch pre-processing* step (not to be confused with the batch normalization within the CNN), we compare four different approaches. Let mean$(\cdot)$ and std$(\cdot)$ denote the computation of the average and standard deviation, respectively. $X_{:,:,f}$ denotes a slice of the given tensor $X \in \mathbb{R}^{N\times 43\times 128}$ for a fixed frequency index $f \in \{1,\ldots,128\}$. Similarly, $X_{p,:,:}$ denotes a slice of the tensor X for a fixed patch index $p \in \{1,\ldots,N\}$. Based on this, we define the four approaches as follows:

The first approach (A) performs frequency-based Zero-Mean and Unit-Variance (ZMUV) normalization following early approaches for efficiently training DNNs [10]. For each $f \in \{1,\ldots,128\}$, it is computed via:

$$X_{:,:,f} \leftarrow \frac{X_{:,:,f} - \text{mean}(X_{:,:,f})}{\text{std}(X_{:,:,f}) + \epsilon}. \quad (1)$$

The second approach (B) applies global ZMUV normalization to the tensor X, following the work presented in [18]:

$$X \leftarrow \frac{X - \text{mean}(X)}{\text{std}(X) + \epsilon}. \quad (2)$$

The third approach (C) employs local patch pre-processing, where each patch $p \in \{1,\ldots,N\}$ is normalized individually in the following way:

$$X_{p,:,:} \leftarrow \frac{X_{p,:,:} - \text{mean}(X_{p,:,:})}{\text{std}(X_{p,:,:}) + \epsilon}. \quad (3)$$

The fourth approach (denoted as "–") does not apply any pre-processing: The mel-spectral representation is provided directly to the CNN model.

For the approaches A and B, we apply ZMUV normalization to the validation and test set using mean and standard deviation as computed from the training set.

4.4 CNN Model

For our experiments, we adopt a CNN architecture proposed by Han et al. [8], illustrated in Figure 1. The model is based on a VGG-type architecture [19] and consists of four blocks that perform convolution operations. Each block contains a pair of 2D convolutional layers, each comprising K kernels of size 3×3. After each convolutional layer, we apply batch normalization (BatchNorm) followed by the Rectified Linear Unit (ReLU) activation function. At the end of each convolutional block, we use 3×3 max-pooling and dropout (with probability 0.25). Between subsequent convolutional blocks, we increase the number of channels K by a factor of two.

After the fourth convolutional block, we use a global max-pooling layer in order to flatten the latent representation. We give the flattened representation to a fully-connected feed-forward layer with 1024 units, followed by the final feed-forward layer that uses a soft-max activation function. We extend the architecture presented by Han et al. [8] using additional batch normalization layers after each convolutional layer. The batch normalization layers perform ZMUV normalization across each batch of mel-spectrogram patches. We train the model using categorical cross-entropy loss, the Adam optimizer with a learning rate of 10^{-4}, and a batch size of 128. In order to reduce overfitting, we implement early stopping during model training with a patience of 20 epochs. Since the audio files in DB-MTC substantially differ in length, we use a class-weighting scheme during training to compensate for class imbalance, which is computed as an inverse proportion of the number of training items per class.

5. EXPERIMENTS

In this section, we present our experiments on instrument family recognition. For Experiment 1 (Section 5.1), we consider polyphonic, monotimbral recordings using the DB-MTC dataset and test the improvement strategies discussed in Section 4. In Experiment 2 (Section 5.2), we investigate the generalization capabilities of the trained models in a cross-dataset experiment using the three datasets described in Section 3. As evaluation measure, we report the micro-average F-score. The F-score is computed as the harmonic mean between precision and recall on a patch-level and is not affected by potential class imbalance.

5.1 Experiment 1: Instrument Family Classification in Monotimbral Classical Music Recordings

In this experiment, we evaluate whether data augmentation, spectrogram compression, and patch pre-processing techniques lead to an improved classification performance. We focus on the MP scenario using the DB-MTC dataset.

5.1.1 Data Augmentation

Due to the small number of 10 audio files per instrument family in DB-MTC (see Section 3.3), we enlarge the dataset using the data augmentation techniques shown in Table 2. We apply algorithms taken from the Audio Degradation Toolbox [16] that implement brown and white noise, two room impulse responses, dynamic range compression, and two kinds of signal attenuation. In total, we create seven augmented versions of each audio file in the dataset, thus increasing the size of the dataset from 50 to 400 files. We refer to this augmented dataset as DB-MTC$^+$.

Table 2: Overview of applied data augmentation methods. *(def)* indicates where default presets from [16] are used.

Abbr.	Augmentation Type	Approach
Noi	Noise	Brown noise, SNR -6 dB
		White noise, SNR +22 dB
Imp	Impulse Response	Great Hall *(def)*
		Classroom *(def)*
Dyn	Dynamic Range Compression	*(def)*
Att	Attenuation	-3 dB
		-6 dB

Table 3: Confusion matrix for the best parameter configuration in Experiment 1 including all augmentations (DB-MTC$^+$), averaged over three folds. The overall F-score is 0.89.

True \ Predicted	woo	bra	pia	voc	str
woodwinds (woo)	**0.92**	0.06	0.02	0.00	0.00
brass (bra)	0.20	**0.70**	0.07	0.00	0.03
piano (pia)	0.01	0.01	**0.97**	0.01	0.01
vocal (voc)	0.09	0.00	0.01	**0.82**	0.08
strings (str)	0.00	0.02	0.02	0.02	**0.94**

5.1.2 Evaluation Procedure

We systematically evaluate 96 combinations of data augmentation, spectrogram compression, and patch pre-processing methods as listed in Table 4. For each configuration, we perform three validation runs and report the mean F-score. In each run, we randomly split the dataset on file level into training (40%), validation (30%), and test set (30%). We use the additional augmented versions of the files for the training and validation sets and test only on the clean, non-augmented signals.

5.1.3 Results

In Table 4, we show the results of Experiment 1. We observe the highest F-score of 0.89 for a system using no compression of the mel-spectrogram (NO), frequency-based patch normalization (A), and a fully augmented training set. Independent of the applied data augmentation, the results show that a normalization of spectral patches before model training is very beneficial if no (NO) or only mild spectrogram compression (LC 1) is applied. In contrast, strong compression (LC 10000) elevates the results for systems without patch pre-processing (−) much closer to the regions with pre-processing: Strong spectrogram compression and the patch normalization techniques have a similar effect, with the latter tending to have an even greater impact. At least one of the methods should be considered for usage. The simple logarithmic compression strategies outperform the PCEN strategy, which we apply using default parameters. We assume that using a trainable PCEN front-end instead seems to be more promising for future work.

In Table 3, we show a confusion matrix for this experiment using the ideal parameter combination. While the piano class is recognized best, confusions mainly occur between vocal and woodwinds or strings, and between woodwinds and brass. Concerning the latter confusion, both families are wind instruments and, therefore, exhibit certain timbral similarity. Furthermore, the presence of french horns in both classes (as discussed in Section 3.3) might be problematic.

5.1.4 Baseline System

To compare the CNN-based results to a simple baseline system relying on standard audio features, we extract 20 mel-frequency cepstral coefficients (MFCC) per spectral patch $X_{p,:,:}$ and average them over the patch duration. This way, each spectral patch is represented by a 20-dimensional MFCC feature vector. We train a random forest classifier with 50 estimators obtaining an F-score of 0.75. This result—which could be further improved by using data augmentation and more diverse audio features—indicates that the benefit of our deep learning strategy over standard approaches is only weak when using datasets of limited size such as DB-MTC.

5.2 Experiment 2: Cross-Dataset Evaluation

In this experiment, we evaluate how well the CNN model generalizes to unseen datasets that represent different levels of timbral complexity. Ideally, we expect the model to learn spectro-temporal patterns that are unique to particular instrument families so that these patterns are recognized independent of a dataset's acoustic characteristics.

5.2.1 Evaluation Procedure

We split all three datasets DB-MTC, DB-SOL, DB-URMP and, additionally, a combination of them called DB-M/U/S, into individual training, validation, and test sets and perform cross-dataset evaluations. Concretely speaking, we train a model using training and validation sets taken from one dataset and evaluate using the test set of another dataset. Due to differences between the datasets, we restrict ourselves in this experiment to the three instrument families woodwinds, brass, and strings which are consistently present over all datasets. Consequently, we discard piano and vocal recordings from DB-MTC. Data augmentation and a comparison with the MFCC-based baseline system are not part of this experiment as we solely focus on the cross-dataset performance of the models.

We compare two approaches for splitting datasets into training, validation, and test sets. We either randomly select patches (patch-based) or split patches based on files (file-based) in order to avoid overfitting due to patches from the same file ending up in both the training and test sets. For the patch-based approach, we identify the smallest amount of available patches per class among the datasets. The smallest class is the brass class in DB-MTC with 4397 patches. Therefore, for the patch-based evaluation, we sample the same amount of patches from all other classes and datasets. We then use a split ratio of 40%–30%–30% to create training, validation, and test sets. Hence, each dataset finally consists of

Table 4: Mean F-scores for the parameter optimization on the DB-MTC dataset (Experiment 1) described in Section 5.1. The abbreviations for data augmentation methods (first four columns) are introduced in Table 2. The spectrogram compression methods LC 1, LC 10000, and PCEN, as well as the patch pre-processing methods –, A, B, C in the remaining columns are described in Section 4.2 and Section 4.3, respectively. The optimal parameter configuration (NO, A, full data augmentation) is highlighted using gray background color. The average standard deviation between F-scores over all three validation runs is 0.03 (min: 0.003, max: 0.15).

Augmentation Types				NO				LC 1				LC 10000				PCEN			
Noi	Imp	Dyn	Att	–	A	B	C	–	A	B	C	–	A	B	C	–	A	B	C
-	-	-	-	0.40	0.78	0.82	**0.85**	0.34	0.76	0.84	0.84	0.66	0.76	0.82	0.75	0.70	0.67	0.67	0.65
✓	✓	✓	✓	0.46	**0.89**	0.86	0.86	0.49	0.88	0.87	0.85	0.86	0.86	0.85	0.83	0.79	0.80	0.79	0.80
✓	-	-	-	0.49	**0.87**	0.86	0.85	0.49	**0.87**	0.84	0.86	0.81	0.82	0.82	0.82	0.75	0.76	0.75	0.75
-	✓	-	-	0.42	**0.86**	0.85	0.85	0.42	0.84	0.85	0.85	0.84	0.83	0.81	0.83	0.79	0.77	0.79	0.78
-	-	✓	-	0.55	0.81	0.83	0.83	0.45	0.83	0.83	**0.85**	0.78	0.83	0.79	0.81	0.75	0.74	0.72	0.76
-	-	-	✓	0.35	0.81	0.84	**0.87**	0.36	0.81	0.83	0.82	0.77	0.78	0.78	0.81	0.74	0.72	0.71	0.74

13191 patches. We create the fourth dataset DB-M/U/S by equally sampling patches from the other datasets.

For the file-based approach, we split each dataset using the same ratio of 40%–30%–30% on a file level, i. e., patches from one file will exclusively end up in one of the subsets. Here, no further steps are taken to balance out the amount of patches. Since this procedure leads to class imbalance, we use class weights as discussed in Section 4.4. The file-based version of DB-M/U/S is generated by accumulating all subsets over all datasets. Table 1 summarizes all datasets used in this experiment. For model training, we pick the parameters that lead to the best result in Experiment 1, namely no compression (NO) and patch pre-processing method C. We do not include any augmentations in this experiment.

5.2.2 Results

Table 5 shows the results for the cross-dataset evaluation on the patch-based data split. Due to the split strategy, the classifier overfits to the training set and naturally achieves high F-scores on the corresponding test set when patches are randomly mixed. This overfitting effect is supported by the fact that adding additional training data from a different dataset in DB-M/U/S even degrades the performance for testing on DB-MTC and DB-SOL.

Table 6 shows the results for the file-based data split. Due to its large size, DB-SOL has the highest impact on the model performance in the mixed dataset DB-M/U/S. When comparing the performance for training and testing on DB-MTC, the F-score drops by 0.12 for the file-based split strategy. This confirms our expectations since the DB-MTC dataset has a small number of files per class and a large variance of file durations.

As a general observation for both split strategies, the CNN approach for instrument family recognition shows only a limited capability to generalize well towards unseen data, which becomes apparent for all cross-dataset combinations in both tables (high values on the diagonal).

Table 5: Resulting F-scores for cross-dataset evaluation using patch-based dataset split. The rows and columns of the table indicate the training and test sets for each configuration, respectively.

Training \ Test	DB-MTC	DB-URMP	DB-SOL	DB-M/U/S
DB-MTC	**0.96**	0.70	0.62	0.77
DB-URMP	0.49	**0.96**	0.61	0.70
DB-SOL	0.64	0.78	**0.95**	0.79
DB-M/U/S	0.95	**0.97**	0.91	0.94

Table 6: Resulting F-scores for cross-dataset evaluation using file-based dataset split.

Training \ Test	DB-MTC	DB-URMP	DB-SOL	DB-M/U/S
DB-MTC	**0.84**	0.60	0.68	0.68
DB-URMP	0.51	**0.92**	0.69	0.70
DB-SOL	0.69	0.74	**0.99**	0.94
DB-M/U/S	0.89	0.95	**0.99**	0.98

6. CONCLUSIONS

In this paper, we investigated a state-of-the-art convolutional neural network model for automatic instrument family recognition in Western classical music recordings. Focusing on monotimbral, polyphonic recordings, we showed that increasing the amount of training data via augmentation techniques leads to improved classification performance. We also found that pre-processing is of central importance for achieving a good system. Combining patch normalization with dynamic compression or per-channel energy normalization does not further improve the results, but these techniques may compensate the effect of patch normalization to some degree. Given that a simple MFCC-based baseline system already achieves good performance in Experiment 1, the possible superiority of more complex data-driven methods such as CNNs needs to be assessed carefully. In a cross-dataset experiment, we further tested how well the CNN model generalizes towards unseen data. Our results indicate that current CNN models lack generalization capability across different datasets, thus indicating the need for applying further optimization methods such as domain adaptation [5].

Acknowledgments: This work has been supported by the German Research Foundation (AB 675/2-1, MU 2686/11-1). The International Audio Laboratories

Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institut für Integrierte Schaltungen IIS.

7. REFERENCES

[1] Rachel M. Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello. MedleyDB: A Multitrack Dataset for Annotation-Intensive MIR Research. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, pages 155–160, Taipei, Taiwan, 2014.

[2] Juan Bosch, Jordi Janer, Ferdinand Fuhrmann, and Perfecto Herrera. A comparison of sound segregation techniques for predominant instrument recognition in musical audio signals. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, pages 559–564, Porto, Portugal, 2012.

[3] Jana Eggink and Guy J. Brown. Instrument recognition in accompanied sonatas and concertos. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 217–220, 2004.

[4] Ferdinand Fuhrmann. *Automatic musical instrument recognition from polyphonic music audio signals*. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2012.

[5] Shayan Gharib, Konstantinos Drossos, Emre Cakir, Dmitriy Serdyuk, and Tuomas Virtanen. Unsupervised adversarial domain adaptation for acoustic scene classification. In *Proceedings of the 3rd Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, pages 138–142, Surrey, UK, 2018.

[6] Juan Gómez, Jakob Abeßer, and Estefanía Cano. Jazz solo instrument classification with convolutional neural networks, source separation, and transfer learning. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pages 577–584, Paris, France, 2018.

[7] Siddharth Gururani, Cameron Summers, and Alexander Lerch. Instrument activity detection in polyphonic music using deep neural networks. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pages 569–576, Paris, France, 2018.

[8] Yoonchang Han, Jae-Hun Kim, and Kyogu Lee. Deep convolutional neural networks for predominant instrument recognition in polyphonic music. *CoRR*, abs/1605.09507, 2016.

[9] Yun-Ning Hung and Yi-Hsuan Yang. Frame-level instrument recognition by timbre and pitch. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pages 135–142, Paris, France, 2018.

[10] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade, this book is an outgrowth of a 1996 NIPS Workshop*, pages 9–50, London, UK, 1998. Springer-Verlag.

[11] Bochen Li, Xinzhao Liu, Karthik Dinesh, Zhiyao Duan, and Gaurav Sharma. Creating a musical performance dataset for multimodal music analysis: Challenges, insights, and applications. *CoRR*, abs/1612.08727, 2016.

[12] Peter Li, Jiyuan Qian, and Tian Wang. Automatic instrument recognition in polyphonic music using convolutional neural networks. *CoRR*, abs/1511.05520, 2015.

[13] Vincent Lostanlen, Joakim Andén, and Mathieu Lagrange. Extended playing techniques: the next milestone in musical instrument recognition. In *Proceedings of the 5th International Conference on Digital Libraries for Musicology (DLfM)*, pages 1–10, Paris, France, 2018.

[14] Vincent Lostanlen and Carmine-Emanuele Cella. Deep convolutional networks on the pitch spiral for music instrument recognition. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, pages 612–618, New York, NY, USA, 2016.

[15] Vincent Lostanlen, Justin Salamon, Mark Cartwright, Brian McFee, Andrew Farnsworth, Steve Kelling, and Juan Pablo Bello. Per-Channel Energy Normalization: Why and How. *IEEE Signal Processing Letters*, 26(1):39–43, 2019.

[16] Matthias Mauch and Sebastian Ewert. The audio degradation toolbox and its application to robustness evaluation. In *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, Curitiba, Brazil, 2013.

[17] Taejin Park and Taejin Lee. Musical instrument sound classification with deep convolutional neural network using feature fusion approach. *CoRR*, abs/1512.07370, 2015.

[18] Jordi Pons and Xavier Serra. Designing efficient architectures for modeling temporal features with convolutional neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2472–2476, New Orleans, USA, 2017.

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[20] Takumi Takahashi, Satoru Fukayama, and Masataka Goto. Instrudive: A music visualization system based on automatically recognized instrumentation. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pages 561–568, Paris, France, 2018.

[21] John Thickstun, Zaid Harchaoui, and Sham M. Kakade. Learning features of music from scratch. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, pages 1–14, Palais des Congrès Neptune, Toulon, France, 2017.

[22] Yuxuan Wang, Pascal Getreuer, Thad Hughes, Richard F. Lyon, and Rif A. Saurous. Trainable frontend for robust and far-field keyword spotting. *CoRR*, abs/1607.05666, 2016.