TRANSFER LEARNING USING MUSICAL/NON-MUSICAL MIXTURES FOR MULTI-INSTRUMENT CLASSIFICATION

Author(s) Name(s)

Author Affiliation(s)

ABSTRACT

Datasets for most music information retrieval (MIR) tasks tend to be relatively small. However, in deep learning, insufficient training data often leads to overfitting. One way to overcome this problem is the use of transfer learning (TL). In this work, we compare various TL methods for the task of multi-instrument classification. The classifier, a convolutional neural network (CNN) operating on mel-spectrogram representations of short audio chunks, is able to identify eight instrument families and seven specific instruments from polyphonic music recordings. Training was conducted in two phases: After pre-training with a music tagging dataset, the CNN is retrained using three multi-track datasets. For this reason, we experimented with different TL methods. The experiment suggests that training the fully connected layers from scratch and fine-tuning the convolutional layers yield the best performance. Training examples are produced on-the-fly by mixing of single-instrument tracks from the multi-track datasets. Therefore, two different mixing strategies – musical and non-musical mixing – are investigated. It turns out that a blend of both mixing strategies works best for multi-instrument classification. The final classifier shows state-of-the-art F1-scores for classes with sufficient training data.

***Index Terms*—** One, two, three, four, five

1. INTRODUCTION

Multi-instrument classification, also known as multi-instrument recognition, strives to identify all instruments in a musical recording. Recently, deep learning became the preferred way to tackle this problem – and nearly all other problems in MIR. Training of deep neural networks (DNNs) requires huge amounts of data. However, compared to other audio signal processing tasks, data for multi-instrument recognition is scarce. TL is a good option to boost generalization of DNNs for tasks with insufficient data. For pre-training there are various large-scale audio and music datasets available [1, 2]. Promising results have been obtained with transfer of DNNs to audio analysis tasks, after pre-training on a huge audio dataset [3]. For TL to be effective, source and target task have to be similar. In this paper, we focus on multi-instrument recognition as a target task. Since music tagging is highly related to multi-instrument recognition, we use a major music tagging dataset [4] for pre-training.

Fortunately, three major multi-track datasets have been released lately [5, 6, 7]. These multi-track datasets, which are mainly developed for music source separation, allow extensive data augmentation by mixing of the single-instrument tracks. In music source separation, good results have been obtained by randomly combining sources from different songs [8]. However, at least for humans, source separation and multi-instrument recognition are significantly easier in case of such non-musical mixtures, i.e. each instrument plays in a different key and tempo. In this work, we want to investigate if extending these non-musical mixtures with musical mixtures while training DNNs can bring performance gains for multi-instrument recognition.

2. METHOD

blabla

2.1. Architecture

The classifier's architecture was adopted from [9]. The proposed network is very similar to a VGG-Net [10] commonly used for image classification. A stack of convolutional layers, hereafter referred to as *backbone*, is followed by fully connected layers. Before we continue with the classifier, we should briefly talk about the backbone at this point. In an initial step, the time domain signal is converted to a mel-spectrogram with 128 frequency bins. A window length of 512 and a hop size of 256 are used to compute the spectrogram. After that, seven convolutional blocks try to find patterns in this time-frequency representation. Each block consists of a 2D convolutional layer with a kernel size of 3×3 followed by batch normalization, ReLU activation and 2×2 max pooling. After seven max pooling layers, the frequency dimension is reduced to one. Finally, 1D max pooling is applied over the time axis to shrink the time dimension to one as well. Since the last convolutional layer has 512 kernels, the resulting output representation of the backbone also has a size of 512. After the backbone extracted suitable features

Thanks to XYZ agency for funding.

from the mel-spectrograms, subsequent fully connected layers learn non-linear combinations of theses features and finally make a decision whether a certain class is active or not. Between the fully connected layers, batchnorm and dropout are applied. In the last layer of the classifier, a sigmoid activation function maps the model's predictions to values between zero and one, hence the outputs can be interpreted as probabilities.

2.2. Data

For TL, a combination of three multi-track datasets is used – *MedleyDB* [5], *Mixing Secrets* [6] and *Slakh* [7]. The first two datasets both feature approximately 130 songs and the last one contains 2100 songs. Note that, although the Slakh dataset is superior when it comes to size, several instruments sound quite unrealistic, because all audio was synthesized from MIDI files using virtual instruments. In addition to the three multi-track datasets, *MTG-Jamendo* [4], a large music tagging dataset is utilized for pre-training of all models. This music tagging dataset already comes with a predefined split. However, the multi-track datasets were randomly divided into training, validation and test sets containing $65\,\%$, $17.5\,\%$ and $17.5\,\%$ of all songs, respectively. We made the decision to split the multi-track datasets at random for the sake of simplicity. Be aware that this practice can potentially lead to overoptimistic results if songs from the same artist or album are contained in multiple splits.

Since the three multi-track datasets used in this work exhibit different class structures, it is necessary to construct a unified instrument taxonomy. Moreover, we want this taxonomy to be hierarchical, hoping that our classifier can leverage the additional information to learn a broader concept of musical instruments. Similar approaches were already taken several times in the deep learning literature and were successfully implemented for IR applications [11]. As a general rule for training DNNs, a sufficient number of examples for every class is required. In other words, classes with insufficient examples have to be discarded, which basically means that some information gets lost. However, our multi-track datasets are highly imbalanced and for certain instruments only few examples are available. To still utilize these underrepresented classes for training, a two-level hierarchy was defined. It is inspired by the *Hornbostel-Sachs* [12] system, which classifies musical instruments based on their underlying sound production mechanisms. As the physical principle of an instrument strongly correlates with its sound, such a taxonomy seems appropriate for our use case. The first level of our proposed taxonomy represents eight instrument families: voice, percussion, bowed string, plucked string, woodwind, brass, key and synth. The second level embodies seven specific instruments for which abundant data was available: singer, drums, violin, electric guitar, acoustic guitar, electric bass and piano.

2.3. Data Loading

Our main data augmentation approach is to create new unique mixtures out of the single-instrument tracks (sources) from the multi-tracks. Therefore, we propose two mixing strategies – *musical mixing* and *non-musical mixing*. Musical mixing means that only sources from the same song are combined to generate a new mix, whereas non-musical mixing signifies that each source in a mix originates from a different song. The latter, which was successfully used in music source separation [8], obviously results in "non-musical", strange sounding mixes, as each instrument plays in a different key and tempo. Nevertheless, the number of training examples can be increased massively by this mixing strategy, which in turn improves generalization. Musical mixing, on the other hand, naturally produces well-sounding results, since all sources originate from the same song and are therefore uniform in tempo and key. In order to investigate the performance of the two proposed mixing strategies, an experiment is conducted in Section **??**. Note that for validation and testing, musical mixing was used exclusively, because in real-world recordings, instruments will almost always play in the same key and tempo.

Initially, one of the three multi-track datasets is selected – each one with the same probability of $1/3$. This step is necessary, because the Slakh dataset is much larger than the other two multi-track datasets. Simply merging them and sampling from the combined dataset would lead to a domination of Slakh samples. However, this should be avoided, because, as already mentioned, the Slakh dataset is synthesized from MIDI files and is therefore quite unrelated to real-world music recordings. In the next step, one of the two proposed mixing strategies is selected. Musical mixing is applied with probability $p_{musical}$ and non-musical mixing is used with probability $1 - p_{musical}$. Then we choose the number of sources N in the mix. As our models should be able to cope with solo-instrument recordings as well, we load single sources ($N = 1$) with a certain probability $p_{single-source}$. After that, the following steps are performed N times for the non-musical mixing method: select a song, select a source from this song, select a four-second chunk from this source, apply some digital effects on the audio and add it to the mix. Four audio effects, which have been tried and tested for data augmentation in MIR, are utilized (each with a certain probability): amplitude scaling (gain), three types of filters (high shelf, low shelf and peaking filter), pitch shifting and time stretching. The order of these effects as well as all their parameters are random. Note that, in the musical mixing case, all sources originate from the same song, hence a song is selected only once and not N times (at every loop iteration). Furthermore, the time position of all chunks has to be identical in order to obtain musical sounding results. Finally, pitch shifting and time stretching are omitted for the same reason when musical mixing is applied.

2.4. Transfer Learning (TL)

Transfer Learning is the procedure of applying knowledge gained from solving one task to another, related task [13]. CNNs typically learn general concepts, like detecting edges or simple shapes, in the earlier layers and increasingly more complex, task-specific concepts in the later layers. Therefore, the first convolutional layers can easily be reused for a similar task. If source and target task are very related, such as music tagging and multi-instrument recognition, even a complete transfer of all convolutional layers can be considered [14]. In order to train our classifier, we experimented with two transfer learning methods. For both approaches, we first replaced the fully connected layers of the pre-trained models with new, randomly initialized ones. Furthermore, the size of the output layer was adjusted to 15 to suit the target task of predicting eight instrument families and seven explicit instruments, according to the taxonomy proposed in Section **??**. During training with the multi-track data, we either (1) froze the whole backbone network and only retrained the fully connected layers or (2) allowed the backbone to be learnable as well, but with a smaller learning rate than the fully connected layers. As a third method (3) the entire model is trained with the multi-track data from scratch, i.e. not utilizing the pre-trained weights at all. Strictly speaking, method (3) has nothing to do with transfer learning, nevertheless we use it as a reference to show the benefits of methods (1) and (2). The results of these three experiments are discussed in Section **??**. For all experiments, the ADAM optimizer in combination with a stepwise learning rate scheduler was employed. Binary cross-entropy served as a loss function.

3. EXPERIMENTS

blabla

3.1. Transfer Learning (TL)

In order to investigate which one of the TL methods proposed in Section 2.4 works best for our use case, an experiment was conducted. Therefore, multiple models were trained on the multi-tracks, applying the three different strategies as presented above. Each model was evaluated on the testing data and ROC-AUC, PR-AUC and test loss were computed. For all three transfer learning methods, 0.01 was chosen as an initial learning rate for the fully connected layers. When training from scratch, the same learning rate was used for the backbone as well. For the fine-tuning approach, the learning rate of the backbone was initialized with 0.0001. Keep in mind that a scheduler was applied, which reduces the learning rate every 20 epochs by a constant factor.

Table 1 contains the results of the three TL experiments. Unsurprisingly, the performance is worst when no pre-training is exploited and the model is trained from scratch on the multi-track data. Utilizing a pre-trained backbone with frozen parameters significantly increases the performance, indicating that pre-training on a large-scale dataset is beneficial indeed. However, fine-tuning the weights of the backbone to fit the needs of the target task yields additional performance gains.

	ROC-AUC	PR-AUC	Test Loss
From Scratch	0.8488	0.6812	0.4459
Frozen Backbone	0.8908	0.7623	0.4160
Fine-tuned	0.9429	0.8334	0.3618

Table 1. ROC-AUC, PR-AUC and test loss for different TL approaches.

learning-curves.png

Fig. 1. ROC-AUC, PR-AUC and test loss for different values of $p_{musical}$.

3.2. Mixing Strategy

As explained in Section **??**, our main data augmentation technique is based on mixing of individual sources to create new examples for training. For this reason, the two mixing strategies musical mixing with probability $p_{musical}$ and non-musical mixing with probability $1 - p_{musical}$ are used. To determine the best ratio between these two methods, we experimented with different values for the hyperparameter $p_{musical}$. The experiment was repeated with four distinct seeds and the evaluation metrics were averaged. As depicted in Fig. 2, the best performance was obtained for $p_{musical} = 0.4$. The result of this investigation suggests, that neither of the two mixing strategies is superior, but rather a combination of both works best. Using only the non-musical mixing approach yields a larger number of training examples. However, the model does not get the chance to learn to identify sources in "real" music where all instruments play in the same key and tempo. In this case, IR is usually more difficult though, at least for us humans, since masking of in-

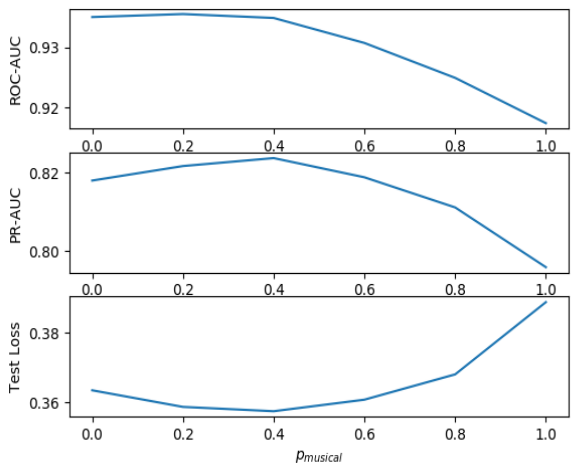

Fig. 2. ROC-AUC, PR-AUC and test loss for different values of $p_{musical}$.

dividual sources is more pronounced due to synchronous note onsets and overlapping partials. Although overlapping of partials results in consonant sounds, it is substantially harder to distinguish individual sources in this case. On the other hand, if only the musical mixing approach is utilized, fewer training examples can be created, as the number of possible combinations of sources from a single song is limited. Fewer training examples in turn can lead to poorer generalization of the model. We infer from this experiment, that a blend of both mixing strategies yields best results for multi-instrument recognition.

3.3. Performance of the Classifier

Fig. 3 shows the performance of our model per class. Unsurprisingly, classes with abundant training data, such as percussion, drums or plucked strings, exhibit the best F1-scores, while identification of instruments or families with insufficient data, like brass, woodwind or violin, does not work well. Note that the F1-score for the brass class is zero because the number of true positives is zero – no brass instrument was identified correctly. Instead, the model always predicts the absence of the brass family, which is true most of the time, since the brass class is very poorly represented in the data. Apart from that, we obtained state-of-the-art performance for the majority of classes. With F1-scores above $95\,\%$ for some classes, our classifier can easily compete with other recent multi-instrument recognition systems from the literature [15, 16, 17].

4. CONCLUSION

blabla ...

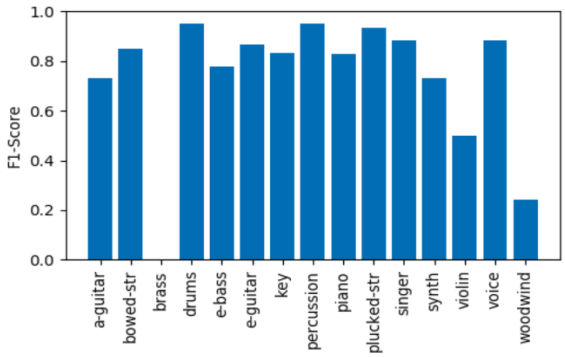

Fig. 3. Class-wise F1-scores of the classifier.

5. REFERENCES

[1] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2017, pp. 776–780.

[2] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere, "The million song dataset," 2011.

[3] Qiuqiang Kong, Yin Cao, Turab Iqbal, Yuxuan Wang, Wenwu Wang, and Mark D Plumbley, "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.

[4] Dmitry Bogdanov, Minz Won, Philip Tovstogan, Alastair Porter, and Xavier Serra, "The mtg-jamendo dataset for automatic music tagging," 2019.

[5] 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 *ISMIR*, 2014, vol. 14, pp. 155–160.

[6] Siddharth Gururani and Alexander Lerch, "Mixing secrets: A multi-track dataset for instrument recognition in polyphonic music," *Proc. ISMIR-LBD*, pp. 1–2, 2017.

[7] Ethan Manilow, Gordon Wichern, Prem Seetharaman, and Jonathan Le Roux, "Cutting music source separation some slakh: A dataset to study the impact of training data quality and quantity," in *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019, pp. 45–49.

[8] Stefan Uhlich, Marcello Porcu, Franck Giron, Michael Enenkl, Thomas Kemp, Naoya Takahashi, and Yuki Mitsufuji, "Improving music source separation based on deep neural networks through data augmentation and network blending," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 261–265.

[9] Minz Won, Andres Ferraro, Dmitry Bogdanov, and Xavier Serra, "Evaluation of cnn-based automatic music tagging models," *arXiv preprint arXiv:2006.00751*, 2020.

[10] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[11] Hugo Flores Garcia, Aldo Aguilar, Ethan Manilow, and Bryan Pardo, "Leveraging hierarchical structures for few-shot musical instrument recognition," *arXiv preprint arXiv:2107.07029*, 2021.

[12] Erich M Von Hornbostel and Curt Sachs, "Systematik der musikinstrumente. ein versuch," *Zeitschrift für Ethnologie*, vol. 46, no. H. 4/5, pp. 553–590, 1914.

[13] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu, "A survey on deep transfer learning," in *International conference on artificial neural networks*. Springer, 2018, pp. 270–279.

[14] Ricardo Ribani and Mauricio Marengoni, "A survey of transfer learning for convolutional neural networks," in *2019 32nd SIBGRAPI conference on graphics, patterns and images tutorials (SIBGRAPI-T)*. IEEE, 2019, pp. 47–57.

[15] Siddharth Gururani, Mohit Sharma, and Alexander Lerch, "An attention mechanism for musical instrument recognition," *arXiv preprint arXiv:1907.04294*, 2019.

[16] Fabian Seipel, "Music instrument identification using convolutional neural networks," *TU Berlin, Institut für Kommunikation und Sprache, Fachgebiet Audiotechnologie, Masterarbeit*, 2018.

[17] Venkatesh Shenoy Kadandale, "Musical instrument recognition in multi-instrument audio contexts," M.S. thesis, MSc thesis, Universitat Pompeu Fabra, 2018.