Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 207 (2022) 2493–2502

Procedia Computer Science

www.elsevier.com/locate/procedia

26th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2022)

Musical Instrument Recognition with a Convolutional Neural Network and Staged Training

Dominika Szeliga[a], Paweł Tarasiuk[a], Bartłomiej Stasiak[a,], Piotr S. Szczepaniak[a]

[a]*Institute of Information Technology*
Łódź University of Technology
al. Politechniki 8, 93-590
Łódź, Poland

Abstract

Musical instrument recognition is an important task within the broad field of Music Information Retrieval. It helps to build recommendation systems, compute similarity between musical compositions and enable automatic search in music collections with regard to the instrument. The task has two variants differed by the difficulty level. The simpler one is classification based on the sound of a single instrument, while the more difficult challenge is to recognize the predominant instrument in polyphonic recordings. In this paper, we used a convolutional neural network to solve both of these problems. As the analysis of monotimbral recordings is relatively easy, we used the knowledge acquired during solving it to train a neural network to tackle the more complex predominant instrument recognition problem. Within this *staged training* scenario, we also examined the impact of introducing some intermediate stages during the training sessions. The results showed that such a training approach has a potential to improve the accuracy of classification.

Keywords: Convolutional neural networks ; CNN ; Musical instrument recognition

1. Introduction

Musical instrument recognition is one of the known and well-established Music Information Retrieval (MIR) tasks [1, 2, 3, 4, 5]. Its aim is to get information about what instrument is playing in a specific recording. This is related to a variety of problems, such as source separation, automatic music transcription, construction of music recommendation systems or music similarity evaluation. It can also be useful in other tasks, including e.g. identification of the mood or genre of a song.

There are two main challenges involved in musical instrument recognition. One of them is classification based on monotimbral, "single instrument" recordings; the other one is predominant instrument recognition. Monotimbral

E-mail address: bartlomiej.stasiak@p.lodz.pl

10.1016/j.procs.2022.09.307

recording analysis is considered simple and many researchers achieved good results here [6] (comparable to human ability to classify instruments based on a single sound, which is about 90% [1]). The second approach is more complicated because in order to recognize the predominant instrument in a music ensemble recording, the system or a human listener needs to determine which soundtrack is *dominant*, and we do not have a precise definition of this word.

1.1. Related work

In the field of single instrument recognition (based on monotimbral recordings) the authors of [6] reported the results about 90% while using a vector of 20 features as an input for a neural network (for comparison, we have achieved accuracy of 98%, by using spectrograms and convolutional neural network for the same task). Paper [2] focused mostly on recognition of multiple predominant instruments, but the authors also checked the effectiveness of their solution in single predominant instrument recognition. They achieved accuracy of 63%. The architecture of a convolutional neural network presented in [2] became an inspiration for the authors of [3]. We also took advantage of their experience with a few changes described in Sect. 3. In [4] several methods of predominant instrument recognition were used. With HSA-IMF method (Hilbert Spectrum Analysis – Intrinsic Mode Functions), the authors reported accuracy about 80%, while using spectrograms as an input for a convolutional neural network let them achieve accuracy of 75%. We have obtained the same result using less complicated network structure with new training approach.

1.2. Training approach

Following the thought, that one needs to teach a child how to keep a pen before it learns how to write and draw, we introduced a *staged training* process. The idea is to train a neural network to recognize musical instruments on the basis of monotimbral recordings – and then, in a second stage of training, to recognize the predominant instrument in complex, polyphonic audio material. We also tried training sessions with more stages, to find out if more smooth gradation of audio content complexity will improve the final classification results.

In our research, seven instruments were considered: cello, clarinet, flute, guitar, saxophone, trumpet, and violin. This choice was dictated by practical reasons, as the selected instruments were the common part of the available datasets.

1.3. Paper structure

The paper is organized as follows. In Section 2, our data preprocessing approach is explained. It is followed by presentation of the CNN architecture and the training method applied. Section 4 is structured according to the considered variants of data sets, namely: recognition based on monotimbral recordings, predominant instrument recognition, mixed data, and noise datasets. The proposed method and the results are presented and analyzed in Sections 5 and 6, respectively.

2. Data preprocessing

In order to use the data as an input for the convolutional neural network, all the recordings were transformed into log-scale spectrograms [7]. To achieve this, all the data were converted to mono .wav files with 16-bit depth and 44.1kHz sampling frequency. Such parameters allowed us to keep the complete information about frequencies from the human hearing range. Samples from the .wav files were divided into windows of size 1024 with 50% overlapping and transformed with fast Fourier transformation (FFT). Each result of the transform became one column of an image in which each gray-scale pixel represented the value of a single FFT frequency bin. The size of the resulting spectrograms was 128×96 pixels, which means that it represented an audio signal section with a length of 1.1 s. As a result of this processing, we get images that show the pattern of frequency components, corresponding to the sound timbre and its changes in time.

Log-frequency scale of the spectrograms ensures that two different pitched sounds will not differ in the distance between the horizontal lines representing the sound partials. Only a vertical shift and brightness changes may appear depending on the pitch and the temporal envelope of the analyzed sound.

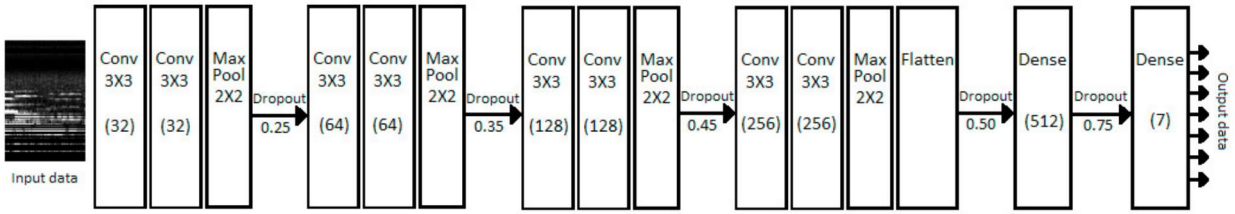

Fig. 1. *Convolutional neural network architecture*

3. Convolutional Neural Network

Convolutional neural networks (CNN) are deep neural networks consisting of several hierarchically arranged layers. The consecutive layers, starting from the input, are responsible for recognition of increasingly detailed templates. The convolutional network was chosen because of its robustness to changes of template position within the input data. A template will be recognized by CNN regardless of whether it is in the corner or in the center of the image. This is especially important in our case because in log-scale spectrograms the location of a template is dependent on pitch and time. Both these factors should not affect classification as it should be done on the basis of the sound timbre only. As the classifier output should be invariant to the translation of the template, it can be concluded that the properties of log-scale spectrograms and CNNs specifically complement each other.

The architecture of the convolutional neural network used for our experiments was inspired by models from [3, 2]. The full model is presented in Fig. 1. Convolutional layers are marked with the word "Conv". The difference between this kind of layers and the basic, fully connected (dense) layers is that in each neuron, the weighted summation of the inputs is replaced by the convolution operation. Weights (which are subject to modification in the learning process) known from traditional neural networks define the spatial *filters* used in the convolutional layers. The size and the number of the filters in each layer are presented below the word "Conv". In both convolutional and dense layers, the rectified linear unit (ReLU) activation function was used. We used Adam [8] optimization method with 0.0001 learning rate, which was chosen after the initial attempts with 0.001 value that was recommended in [8]. The final choice was sufficient to slow down the training process and thus to reduce the difference between the accuracies achieved on the training and test datasets.

MaxPooling layers help to reduce the amount of data transferred between the intermediate layers, in order to improve performance. At the same time, it nullifies the impact of minor shifts of the template position on the classification ability. The MaxPooling operation involves sliding a window of specific size across the data and returning the biggest value found in this window. The size of each MaxPooling window is presented in Fig. 1 below the word "MaxPool".

The sequence of convolutional layers and MaxPooling layers performs the input data processing with regard to the concept of the receptive fields, as it is historically inspired by [9, 10]. Each output value of the intermediate layers is computed based on the rectangular range of the network input, the size of which corresponds to the convolutional filter sizes and the MaxPooling factors. The maximum receptive field size available to the final MaxPooling layer of the model illustrated in Fig. 1 is 136×136, which exceeds the size of the inputs. In the result, each output of this layer is able to utilize the whole content of the input data.

The main difference between our CNN architecture and the model proposed in [2] are the additional dropout layers. Dropout layers [11] are responsible for switching off some randomly chosen inputs from the previous layer. The probability of switching off is called a *dropout factor*. The addition of these layers affects a dynamically changing model, which improves generalization and effectively prevents overfitting. This is especially important in the proposed training method (described in more detail in 3.1) as during each transition between stages the characteristic of input data changes. An overfitted neural network would not fit a new data. Natural drop of accuracy, observed always during transitions, would be much more significant and slow down the learning process.

3.1. Training method

Transfer learning [12] is a common training approach in the cases where the dataset is small [13]. The idea of this method is to train a neural network to solve a general problem, for which sufficiently large training data collections are available, and then train it to deal with a more specific issue. For example, we can train a network how to recognize the presence of dogs in an image and then retrain it to recognize only huskies. Accordingly, we can use a big dataset of images with and without dogs in the first stage of training, and then use a smaller dataset of husky images for a final stage. Our approach is quite similar to the transfer learning, with a few specific adjustments. The main difference is that the task in both stages is the same – musical instrument recognition – and only the source material for classification is different (single instrument recordings in the first stage and more complex musical pieces in the second stage). This is why we use more general name "staged training" instead of "transfer learning".

We also researched the concept of multiple intermediate stages of training, in attempt to make the transfer of knowledge from stage to stage smoother. To achieve that, we used mixed datasets which contained both recordings of a single instrument and fragments of songs with a predominant instrument. Another way to create datasets for the additional stages of training was addition of some "noise content" to the monotimbral recordings. For this purpose, fragments of randomly chosen songs were used. This approach enabled us to increase the similarity between the recordings from the intermediate stage and from the target stage. In this paper, both aforementioned methods are considered and included in the experiments.

4. Datasets

4.1. Instrument recognition based on monotimbral recordings

For recognition of a single instrument, we used a mixture of three collections: the University of Iowa Musical Instrument Samples [14], Philharmonia Orchestra Sound Samples [15] and RWC Music Database [16, 17]. The structure of the resulting dataset, hereinafter referred to as a *MonoDataset* is presented in Tab. 1.

4.2. Predominant instrument recognition

For predominant instrument recognition, we used IRMAS dataset [18, 19]. This dataset had already been divided into training and testing sets, but we used only the training set (and split it on our own) in order be able to follow one-label classification scheme.

4.3. Mixed datasets for the intermediate stages of training

For training and testing in the intermediate stages, we used datasets which were a mixture of collections used for monotimbral recognition and for predominant instrument recognition. Three such datasets were built, with the proportion between both source material types of 70:30, 50:50 and 30:70, respectively. These datasets are called *Dataset70-30*, *Dataset50-50*, and *Dataset30-70*, where the number at the end of a name is the proportion between the monotimbral dataset and the IRMAS dataset (the one used for predominant instrument recognition).

Table 1. Structure of the MonoDataset used for monotimbral recording recognition

Source dataset	Cell.	Clar.	Flu.	Guit.	Sax.	Trum.	Viol.	Total
University of Iowa	127	258	428	106	153	107	208	1459
Philharmonia Orchestra	824	766	763	298	667	414	1176	4908
RWC	-	-	-	466	-	-	-	466
Total	951	1024	1191	870	820	521	1456	6833

4.4. Noise datasets for the intermediate stages of training

For the intermediate training stages, we also used "noise content" datasets which were created by adding a part of a randomly chosen song from FMA dataset [20, 21] to monotimbral recordings. In this way, three datasets with different "noise content" level were built, named *Dataset05f*, *Dataset15f* and *Dataset30f*, respectively. The number at the end of each name refers to the percent of sound intensity taken from the original song which was mixed with the monotimbral recording.

5. Method

In order to evaluate the proposed staged training approach, several tests were conducted. In the first experiment, we trained the neural network to recognize a musical instrument based on a monotimbral recording, and then – to recognize the predominant instrument in a song. Several lengths of the first stage were considered and the results from each training session were compared. Thanks to this experiment we realized, that splitting the training process into two parts can improve the classification results only if the first stage of training lasts long enough. Otherwise it can decrease the classification accuracy. This experience allowed us to plan the next experiment to decide how many stages should be used to achieve the best performance and how to set the duration of stages in the training process.

From the previous experiment we know that the minimum length of the first stage in our case should be about 400 epochs. Basically, there exist three main approaches to the organization of the stage lengths: increasing – where each next stage has more epochs than the preceding one, decreasing – where each next stage has fewer epochs, and balanced – where each stage has the same number of epochs. All three approaches were tested with a different number of stages. The specific number of epochs in each stage in each training session can be seen in Tab. 2.

Another tested issue was the number of stages in the training process. We tried training sessions from one to four stages. The names of the datasets used in specific stages are presented in Tab. 3 for training with the mixed datasets and in Tab. 4 for training with the noise datasets. All of the presented datasets were divided into training and testing sets. The results shown in section 6 were obtained on the testing datasets which had not been used in the training process.

Table 2. Training stage lengths

		Number of epochs in a stage			
Configuration		**stage 1**	**stage 2**	**stage 3**	**stage 4**
	One-stage	3000	-	-	-
Increasing	Two stages	1000	2000	-	-
	Three stages	500	1000	1500	-
	Four stages	600	700	800	900
Balanced	Two stages	1500	1500	-	-
	Three stages	1000	1000	1000	-
	Four stages	750	750	750	750
Decreasing	Two stages	2000	1000	-	-
	Three stages	1500	1000	500	-
	Four stages	900	800	700	600

Table 3. Mixed datasets in training sessions

	stage 1	stage 2	stage 3	stage 4
One stage	IRMAS	-	-	-
Two stages	MonoDataset	IRMAS	-	-
Three stages	MonoDataset	Dataset50-50	IRMAS	-
Four stages	MonoDataset	Dataset70-30	Dataset30-70	IRMAS

Table 4. Noise datasets in training sessions

	stage 1	stage 2	stage 3	stage 4
One stage	IRMAS	-	-	-
Two stages	MonoDataset	IRMAS	-	-
Three stages	MonoDataset	Dataset15f	IRMAS	-
Four stages	MonoDataset	Dataset05f	Dataset30f	IRMAS

6. Results and discussion

6.1. Staged training with the mixed datsets

Table 5 shows the classification accuracy achieved in each training session, where the mixed datasets were used. The best accuracy was reached in a balanced configuration, but the differences between configuration types were not significant. The most promising number of stages was two.

Table 6 presents the average difference between the accuracy computed on the training and the testing datasets. This measure enables us to evaluate the generalization ability of the neural network (smaller value means better generalization). The best results in this field were achieved in the decreasing configuration and two-staged training. The decreasing configuration seems to be the most suitable one for staged training because the last stage is too short to result in overfitting and, at the same time, the preceding training stages are long enough to learn the general dataset used during the whole training session, thereby improving generalization.

Table 5. Best accuracy achieved in training sessions with the mixed datasets, measured on the testing dataset

	Increasing	Balanced	Decreasing	Average
Two stages	**0.7547**	0.752	0.7520	0.7529
Three stages	0.7352	0.754	0.7406	0.7432
Four stages	0.7386	0.7352	0.7359	0.7365
Average	0.7428	0.7470	0.7428	0.7442

Table 6. Average difference between accuracy measured on the training and the testing datasets, obtained for the mixed datasets. (measure of generalization)

	Increasing	Balanced	Decreasing	Average
Two stages	0.2519	0.2427	**0.2352**	0.2433
Three stages	0.2746	0.2528	0.245	0.2575
Four stages	0.2664	0.2662	0.2606	0.2664
Average	0.2643	0.2539	0.2469	0.2550

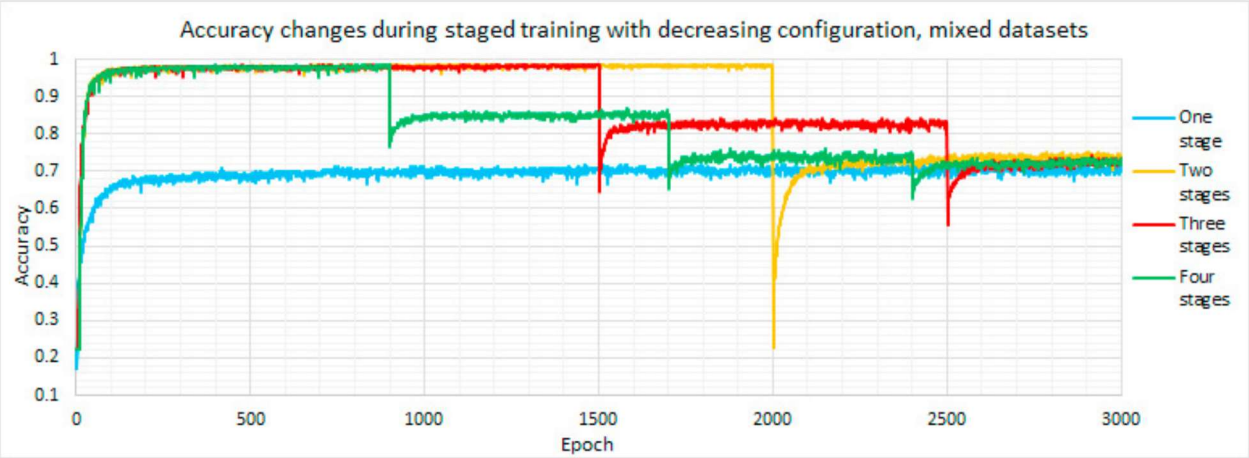

Fig. 2. *Changes in accuracy during training sessions with the mixed dataset and the decreasing configuration, measured on the testing datasets*

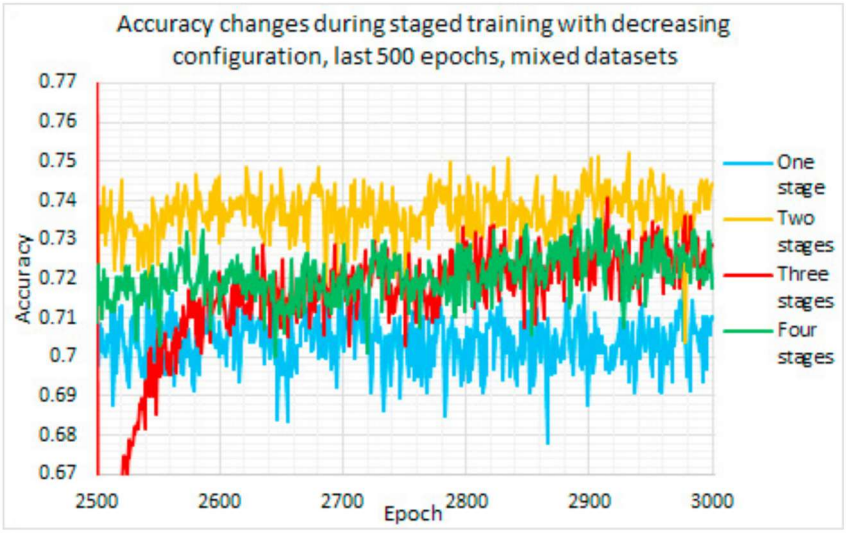

Fig. 3. *Changes in accuracy during training sessions with the mixed dataset and the decreasing configuration (the last 500 epochs), measured on the testing dataset*

Figure 2 shows accuracy changes in the training sessions with the decreasing configuration and various numbers of stages. For better readability, the last 500 epochs are shown additionally in Fig. 3.

Using intermediate training stages decreases the classification accuracy, as compared to two-staged training. This may result from the fact that using the mixed datasets makes the neural network try to solve two problems at the same time. This impedes finding similarities between the spectrograms and, as a result, it slows down the training process. We may therefore conclude, that our CNN model has problems with “multitasking”. This phenomenon can be compared to human brain behavior. Operation types apparently requiring multitasking are indeed realized by continuous switching between individual tasks in our brain [22]. This “switching” can decrease the performance of a neural network because it costs energy and resources. Achieving real multitasking is possible only if the architecture of a neural network is adjusted to the requirements of all the tasks [23].

Table 7. Best accuracy achieved in training sessions with the noise datasets, measured on the testing dataset

	Increasing	Balanced	Decreasing	Average
Two stages	**0.7547**	0.7520	0.752	0.7529
Three stages	0.7278	0.7433	0.7345	0.7352
Four stages	0.7359	0.7372	0.7419	0.7383
Average	0.7395	0.7442	0.7428	0.7421

Table 8. Average difference between accuracy measured on the training and the testing datasets, obtained for the noise datasets (measure of generalization)

	Increasing	Balanced	Decreasing	Average
Two stages	0.2519	0.2427	0.2352	0.2432
Three stages	0.2747	0.2527	**0.2344**	0.2540
Four stages	0.2602	0.2493	0.2360	0.2485
Average	0.2622	0.2483	0.2352	0.2486

6.2. *Staged training with the noise datasets*

Table 7 shows the accuracy achieved in each training session where the noise datasets were used, divided into configuration types and numbers of stages. The two-staged training leads to obtaining the best classification results, irrespective of the relative lengths of the stages.

Table 8 shows the average difference between the accuracy measured on the training and the testing datasets. It follows from this that the best generalization is achieved in the decreasing configuration which confirms the conclusion from the previous experiment and Tab. 6. It can also be observed that the lowest difference was achieved for the two-staged training sessions which were not using intermediate stages. This was surprising, taking into consideration the fact that adding noise to input data is a commonly known method of increasing the generalization performance of neural networks [24, 25]. The probable explanation of this phenomenon is that this method works only within the stage it was used in. It means that it can improve generalization "locally" but it does not cause a general improvement in this regard.

Figure 4 shows accuracy changes during the whole training process for sessions with the decreasing configuration. The last 500 epochs are additionally presented in Fig. 5 for better readability.

An interesting phenomenon may be observed with respect to the location of the starting point of each stage. When the number of stages is increased, the last training stage begins from higher accuracy level in its first epoch. In the two-stage training session, the last stage started at about 22% of the accuracy while in the three-staged training the accuracy in the first epoch of the last stage was equal to 26%. The difference is small, but it let us conclude that the intermediate stages of training have some positive impact on the training process, although it is compensated between the sessions by the differences in the length of the last stage. We cannot fully observe the final improvement in accuracy which was achieved due to the increasing number of stages, because in the sessions with more stages the last training stage was shorter.

6.3. *Comparison of staged training and traditional methods*

Tables 9 and 10 are confusion matrices for predominant instrument recognition conducted with traditionally trained and staged trained neural networks, respectively. In both cases the most difficult instrument to recognize was saxophone, the easiest one – guitar. It can be said that staged training improves classification for all classes except for the trumpet, where a slight loss was noticed (around 1%). The most significant improvement can be observed for clarinet. All the above conclusions where made based on *recall* factor (ratio of correctly classified samples from a specified class to a number of all samples from this class).

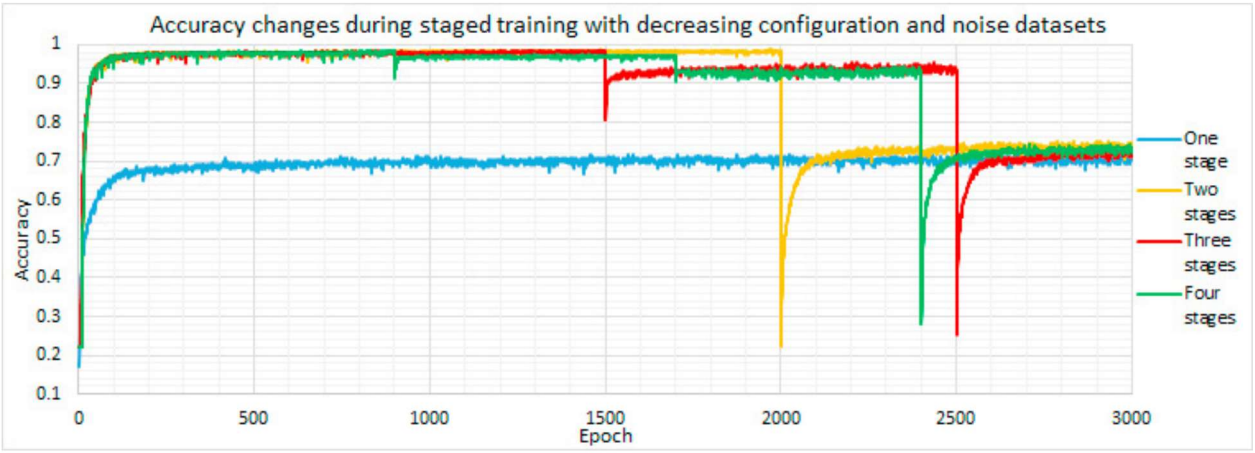

Fig. 4. *Changes in accuracy during training sessions with the noise dataset and the decreasing configuration, measured on the testing datasets*

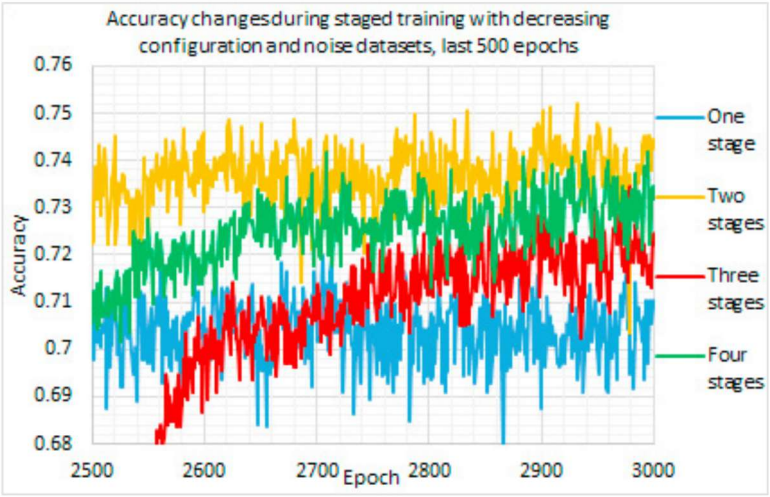

Fig. 5. *Changes in accuracy during training sessions with the noise dataset and the decreasing configuration (the last 500 epochs), measured on the testing dataset*

Table 9. Confusion matrix for neural network trained with traditional method

		Predicted						
		Cell.	Clar.	Flute	Guit.	Sax.	Trum.	Viol.
Expected	Cell.	94	4	4	14	6	2	12
	Clar.	7	152	11	1	16	8	7
	Flute	8	13	124	9	15	6	7
	Guit.	15	1	3	207	15	4	7
	Sax.	12	20	13	18	162	14	13
	Trum.	6	4	7	11	13	183	8
	Viol.	27	5	5	14	20	10	151

Table 10. Confusion matrix for neural network trained with staged training

		Predicted						
		Cell.	Clar.	Flute	Guit.	Sax.	Trum.	Viol.
Expected	Cell.	100	3	7	11	3	1	11
	Clar.	5	167	9	2	12	2	5
	Flute	5	8	134	9	10	9	7
	Guit.	6	4	6	210	13	5	8
	Sax.	12	18	11	21	164	14	12
	Trum.	4	4	7	7	23	178	9
	Viol.	16	6	11	10	23	7	159

7. Conclusions

In this paper, we tested a staged training approach in the task of musical instrument recognition. We transformed audio data into log-scale spectrograms and provided them as the input for a convolutional neural network trained to classify recordings by the predominant instrument. The results we achieved are better than those described in [2] where similar network architecture and the same dataset were used. Our results are comparable to [4]. We discovered that using the staged training approach we can improve the accuracy and generalization of the neural network. Introducing intermediate stages which use mixed datasets (datasets created as a mixture of the monotimbral and complex polyphonic recordings) can be considered as an attempt to train the neural network to solve two problems at the same time. On the other hand, using intermediate stages based on "noise content" datasets is not causing this kind of issues and leads to a slight improvement in accuracy. As a final conclusion, we recommend using two stages of training with decreasing configuration (second stage lasts less than the first one) as it leads to the best performance.

References

[1] Srinivasan A, Sullivan D, Fujinaga I. Recognition of isolated instrument tones by conservatory students. In: Proc. International Conference on Music Perception and Cognition; 2002. p. 17-21.
[2] Han Y, Kim J, Lee K. Deep convolutional neural networks for predominant instrument recognition in polyphonic music. IEEE/ACM Transactions on Audio, Speech, and Language Processing. 2016;25(1):208-21.
[3] Gómez JS, Abeßer J, Cano E. Jazz Solo Instrument Classification with Convolutional Neural Networks, Source Separation, and Transfer Learning. In: ISMIR; 2018. p. 577-84.
[4] Kim D, Sung TT, Cho S, Lee G, Sohn CB. A Single Predominant Instrument Recognition of Polyphonic Music Using CNN-based Timbre Analysis. International Journal of Engineering & Technology. 2018;7(3.34):590-3.
[5] Garcia HF, Aguilar A, Manilow E, Pardo B. Leveraging Hierarchical Structures for Few-Shot Musical Instrument Recognition. arXiv; 2021. Available from: https://arxiv.org/abs/2107.07029.
[6] Masood S, Gupta S, Khan S. Novel approach for musical instrument identification using neural network. In: 2015 Annual IEEE India Conference (INDICON). IEEE; 2015. p. 1-5.
[7] Stasiak B, Monko J. Analysis of time-frequency representations for musical onset detection with convolutional neural network. In: Proc. of the 2015 Federated Conf. on Computer Science and Information Systems, FedCSIS; 2016. p. 147 152.
[8] Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980. 2014.
[9] Hubel D, Wiesel T. Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex. Journal of Physiology. 1962;160:106-54.
[10] Fukushima K. Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. Biological Cybernetics. 1980;36:193-202.
[11] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 2014;15(56):1929-58. Available from: http://jmlr.org/papers/v15/srivastava14a.html.
[12] Pratt LY, Hanson SJ, Giles CJ, Cowan JD. Discriminability-Based Transfer between Neural Networks. In: Advances in Neural Information Processing Systems 5. Morgan Kaufmann; 1993. p. 204-11.
[13] Pan SJ, Yang Q. A survey on transfer learning. IEEE Transactions on knowledge and data engineering. 2009;22(10):1345-59.
[14] University of Iowa. Musical Instrument Samples; accessed July 08, 20019. Available at http://theremin.music.uiowa.edu/MIS.html.
[15] London Philharmonia Orch . Sound Samples; accessed July 10, 20019. Available at https://www.philharmonia.co.uk/explore/sound_sample.
[16] Goto M, Hashiguchi H, Nishimura T, Oka R. RWC music database: Music genre database and musical instrument sound database. 2003.
[17] Goto M, et al. Development of the RWC music database. In: Proceedings of the 18th International Congress on Acoustics (ICA 2004). vol. 1; 2004. p. 553-6.
[18] Bosch JJ, Janer J, Fuhrmann F, Herrera P. A Comparison of Sound Segregation Techniques for Predominant Instrument Recognition in Musical Audio Signals. In: ISMIR; 2012. p. 559-64.
[19] Universitat Pompeu Fabra Barcelona. IRMAS datset; accessed December 7, 20019.
[20] Defferrard M, Benzi K, Vandergheynst P, Bresson X. FMA: A dataset for music analysis. arXiv preprint arXiv:161201840. 2016.
[21] Defferrard M, Benzi K, Vandergheynst P, Bresson X. FMA datset; accessed January 14, 20020. Available at https://github.com/mdeff/fma.
[22] Rosen C. The myth of multitasking. The New Atlantis. 2008;(20):105-10.
[23] Caruana R. Multitask learning. Machine learning. 1997;28(1):41-75.
[24] Bishop CM. Training with noise is equivalent to Tikhonov regularization. Neural computation. 1995;7(1):108-16.
[25] Yin S, Liu C, Zhang Z, Lin Y, Wang D, Tejedor J, et al. Noisy training for deep neural networks in speech recognition. EURASIP Journal on Audio, Speech, and Music Processing. 2015;2015(1):2.