# GENRE-AGNOSTIC KEY CLASSIFICATION WITH CONVOLUTIONAL NEURAL NETWORKS

**Filip Korzeniowski and Gerhard Widmer**
Institute of Computational Perception,
Johannes Kepler University, Linz, Austria
filip.korzeniowski@jku.at

## ABSTRACT

We propose modifications to the model structure and training procedure to a recently introduced Convolutional Neural Network for musical key classification. These modifications enable the network to learn a genre-independent model that performs better than models trained for specific music styles, which has not been the case in existing work. We analyse this generalisation capability on three datasets comprising distinct genres. We then evaluate the model on a number of unseen data sets, and show its superior performance compared to the state of the art. Finally, we investigate the model's performance on short excerpts of audio. From these experiments, we conclude that models need to consider the harmonic coherence of the whole piece when classifying the local key of short segments of audio.

## 1. INTRODUCTION

The musical key is the highest-level harmonic representation in Western tonal music. It thus plays a central role in understanding the semantic content of a piece. Such understanding drives not only theoretical analyses of music, but is also relevant for modern music creators, who mix samples from various different pieces that fit well harmonically into a new composition. However, deriving the key of a musical piece is a demanding task that only experts can perform. It is thus impractical to annotate large music collections by hand. Therefore, we need computational key classification systems.

Most key classification systems (e.g. [9, 16, 17, 21]) conform to the same principle: they extract a time-frequency representation of the audio, filter out nuisances, map this representation to chroma vectors, and accumulate them over time. The resulting feature vector is then compared to template vectors for each key. The drawbacks of such approaches include that key templates differ for different musical genres [9] and favour one key mode over another [1]. This leads to key classification systems that perform well only on the musical styles they were designed for. Although there are attempts to address these issues [2], ideally, we would want a model that handles different kinds of input autonomously, and does not need human intervention to e.g. balance mode probabilities.

Data-driven methods bear the potential to meet this requirement. Recently, an end-to-end neural-network-based key classification model was introduced [14]. Although it generalised better across musical genres than hand-crafted approaches, it still achieved the best results when tuned specifically for a musical style. In this paper, we present modifications to the model structure and its training procedure that enable the model to learn a key classifier that is agnostic to genre. Not only does it perform better than the model proposed in [14] on all genres the latter is optimised for; it does so not despite, but *because* it is trained on various musical styles, instead of a specific one (see Sec. 3.4).

## 2. METHOD

We build upon the same audio processing pipeline used in [14], and input to the network a log-magnitude log-frequency spectrogram (5 frames per second, frame size 8 192, sample rate 44 100 Hz, 24 bins per octave). We limit the frequency range to the harmonically most relevant 65 Hz to 2 100 Hz, as found in [12].

The network structure proposed in [14] was modelled after typical processing pipelines used for key classification. It features five convolutional layers of $5 \times 5$ kernels for spectrogram processing, followed by a dense projection into a frame-wise embedding space, which is then averaged over time and classified using a softmax layer. All layers except the last use the exponential-linear activation function [4] (ELU). The architecture, which we name *KeyNet*, is summarised in Table 1a.

During training, the model is shown the complete spectrogram of a piece. Its weights are then adapted using stochastic gradient descent to minimise the categorical cross-entropy between the predicted key distribution and the ground truth. We will refer to the KeyNet architecture, when trained using full spectrograms, as *KeyNet/F*.

### 2.1 Adaptations of the Training Procedure

The outlined training scheme has two drawbacks. First, the computation of a single update is expensive; the network has to process the full spectrogram (e.g. $600 \times 105$ values for

a two-minute piece), and keep intermediate results for back-propagating the error. Training is thus slow and requires much memory. Second, it keeps the variety of the data lower than necessary, as the network sees the same spectrograms at every epoch.

To circumvent these drawbacks, we show the network only short snippets instead of the whole piece at training time (similar to random cropping in computer vision). These snippets should be as short as possible to reduce computation time, but have to be long enough to contain the relevant information to determine the key of a piece. From our datasets, we found 20 s to be sufficient (with the exception of classical music, which we need to treat differently, due to the possibility of extended periods of modulation—see Sec. 3.1 below). Each time the network is presented a song, we cut a random 20 s snippet from the spectrogram. The network thus sees a different variation of each song every epoch.

During testing, the network processes the whole piece. This gives better results than when using only a snippet. Since we do not have to store intermediate results and process each piece many times as in training, memory space and run time are not an issue. We will refer to KeyNet models trained using spectrogram snippets as *KeyNet/S*.

We expect this modification to have the following effects. a) Back-propagation will be faster and require less memory, because the network sees shorter snippets; we can thus train faster, and process larger models. b) The network will be less prone to over-fitting, since it almost never sees the same training input; we expect the model to generalise better. c) The network will be forced to find evidence for a key in each excerpt of the training pieces, instead of relying on parts where the key is more obvious; by asking more of the model, we expect it to pick up more subtle relationships between the audio and its key.

### 2.2 Adaptations of the Model Structure

The KeyNet architecture uses a dense layer to project the processed spectrogram into a key embedding space. In its original formulation, which uses an embedding space with 48 dimensions and 8 feature maps in the convolutional layers, this projection accounts for 65 % of the network's parameters. Dense layers are also more prone to over-fitting than convolutional layers.

We thus propose to use a network architecture that does away with dense layers, and relies on convolutions and pooling only. At the same time, we move away from modelling the network based on traditional key classification methods—recall that the components of KeyNet were designed to correspond to components in typical key classification pipelines—and instead use a general network architecture for classification, based on the all-convolutional net [19]. The new architecture is summarised in Table 1b, and will be referred to as *AllConv*. As with KeyNet/S, we will train this architecture only with the snippet method.

We expect this change to improve results and generalisation because a) convolutional layers over-fit less than dense layers; b) given the same number of parameters, deeper networks are more expressive than shallower ones [8, 15]; c) comparable architectures have shown to perform well in other audio-related tasks [7, 13].

(a) KeyNet Architecture

| Layer Type | FMaps | Params |
|---|---|---|
| Input | | |
| Conv-ELU | $N_f$ | $5 \times 5$ |
| Conv-ELU | $N_f$ | $5 \times 5$ |
| Conv-ELU | $N_f$ | $5 \times 5$ |
| Conv-ELU | $N_f$ | $5 \times 5$ |
| Conv-ELU | $N_f$ | $5 \times 5$ |
| Dense-ELU | | $2 \cdot N_f$ |
| Pool-Time Avg. | | |
| Dense-Softmax | | 24 |

(b) AllConv Architecture

| Layer Type | FMaps | Params |
|---|---|---|
| Input | | |
| Conv-ELU | $N_f$ | $5 \times 5$ |
| Conv-ELU | $N_f$ | $3 \times 3$ |
| Pool-Max | | $2 \times 2$ |
| Conv-ELU | $2N_f$ | $3 \times 3$ |
| Conv-ELU | $2N_f$ | $3 \times 3$ |
| Pool-Max | | $2 \times 2$ |
| Conv-ELU | $4N_f$ | $3 \times 3$ |
| Conv-ELU | $4N_f$ | $3 \times 3$ |
| Pool-Max | | $2 \times 2$ |
| Conv-ELU | $8N_f$ | $3 \times 3$ |
| Conv-ELU | $8N_f$ | $3 \times 3$ |
| Conv-ELU | 24 | $1 \times 1$ |
| Pool-Global Avg. | | |
| Softmax | | |

**Table 1**. Neural Network architectures. $N_f$ is a parameter that controls the model complexity. Horizontal lines denote dropout layers [20]. Here, dropout is applied on complete feature maps, not individual units. Each convolution is followed by batch normalisation [11]. *FMaps* indicates the number of feature maps, while *Params* the parameters of the layer (kernel size, pool size, or number of units).

## 3. EXPERIMENTS

We first evaluate how the proposed modifications affect the key classification performance in Sec. 3.3. Then, we analyse how the number and genre of training data influence results in Sec. 3.4.

### 3.1 Data

Since we are interested in how well the models generalise across different genres, we use datasets that encompass three distinct musical styles. As in [14], we apply pitch shifting in the range of -4 to +7 semitones to increase the amount of training data.

**Electronic Dance Music:** Here, we use songs from the GiantSteps MTG Key dataset [1], collected by Ángel Faraldo. It comprises 1486 distinct two-minute audio previews from www.beatport.com, with key ground truth for each excerpt. We only use excerpts labelled with a single key and a high confidence (1077 pieces), and split them into 80 % training and 20 % validation. For testing, we use the GiantSteps Key Dataset [2]. It comprises 604 two-minute audio previews from the same source (but distinct from the training set).

[1] https://github.com/GiantSteps/giantsteps-mtg-key-dataset
[2] https://github.com/GiantSteps/giantsteps-key-dataset

**Pop/Rock Music:** For this genre, we use the McGill Billboard dataset [3] [3]. It consists of 742 unique songs sampled from the American Billboard charts between 1958 and 1991. We split these songs into subsets of 62.5% for training, 12.5% for validation, and 25% for testing. We determine the global key for each song using the procedure described in [14], which leaves us with 625 songs with key annotations in total. The exact division and key ground truths are available online [4].

**Classical Music:** To cover this genre, we collected 1504 (mostly piano) pieces from our internal database for which we could derive the key from the piece's title. Classical pieces often modulate their key, but usually start in the key denoted in the title. We thus only use the first 30 s of each recording. Tracking key modulations is left for future work. We then select 81 % for training, 9 % for validation, and 10 % for testing.

### 3.2 Metrics

We adopt the standard evaluation score for Key Classification as defined in the MIREX evaluation campaign [5]. It goes beyond simple accuracy, as it considers harmonic similarities between key classes. A prediction can fall into one of the following categories:

**Correct:** if the tonic and the mode (major/minor) of prediction and target correspond.

**Fifth:** if the tonic of the prediction is the fifth of the target (or vice versa), and modes correspond.

**Relative Minor/Major:** if modes differ and either a) the predicted mode is minor and the predicted tonic is 3 semitones below the target, or b) the predicted mode is major and the predicted tonic is 3 semitones above the target.

**Parallel Minor/Major:** if modes differ but the predicted tonic matches the target.

**Other:** Prediction errors not caught by any category, i.e. the most severe errors.

Then, a weighted score can be computed as $w = r_c + 0.5 \cdot r_f + 0.3 \cdot r_r + 0.2 \cdot r_p$, where $r_c$, $r_f$, $r_r$, and $r_p$ are the ratios of the correct, fifth, relative minor/major, and parallel minor/major, respectively. We will use this weighted score for our comparisons.

### 3.3 Evaluation of the Adaptations

To evaluate the effect of our proposed adaptations, we train the three setups (KeyNet/F, KeyNet/S, AllConv) with the combined data of all datasets. We will consider *validation* results in the first sets of experiments, and show results on the testing sets only for our analyses and final evaluations. This way, we ensure that the final results are unbiased.

[3] http://ddmal.music.mcgill.ca/research/billboard
[4] http://www.cp.jku.at/people/korzeniowski/bb.zip
[5] http://www.music-ir.org/mirex

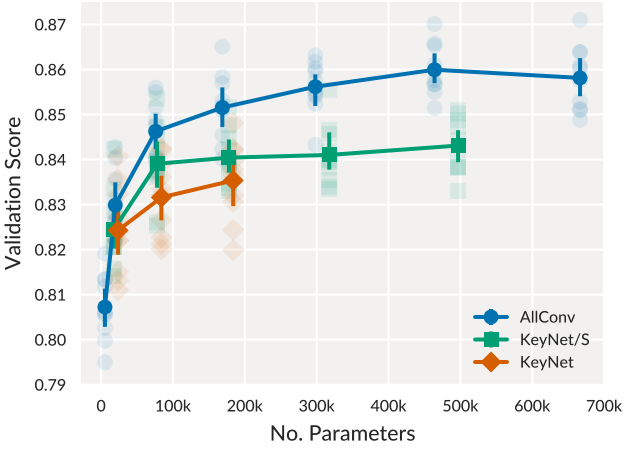


**Figure 1**. Average validation score over 10 runs for the different model setups. Whiskers represent 95 % confidence intervals computed by bootstrapping. Transparent dots show results of the individual runs. We see that given similar network sizes, the AllConv model performs best. Also, using snippet training (KeyNet/S) improves results compared to full spectrogram training (KeyNet/F), and enables training larger networks.

The capacity of a neural network depends not only on the architecture, but also on its size. For a fair comparison, we evaluate each architecture with varying network sizes. For the AllConv architecture, we select the number of feature maps $N_f \in \{2, 4, 8, 12, 16, 20, 24\}$. For the KeyNet architecture, the network size depends on the number of feature maps in the convolutional layers and the size of the embedding space. For practical reasons, we set the size of the embedding space to be $2N_f$, and select $N_f \in \{8, 16, 24, 32, 40\}$. Note that if we train on full spectrograms (KeyNet/F), we could not train networks with $N_f > 24$ due to memory constraints. For each model, we tried dropout probabilities of $p \in \{0.0, 0.1, 0.2\}$.

Figure 1 presents the results of the three model configurations. For each model and model capacity, we select the best dropout probability based on the validation results. The experiments show that both adaptations are beneficial. Training with snippets instead of full spectrograms gives better results at smaller network capacities and enables training of larger networks. The AllConv architecture achieves even better results, regardless of its size.

We can quantify two reasons for this, which are consequences of the expected benefits of the adaptations: *better generalization* through increased data variety and the absence of dense layers, and *better expressivity* through deeper architectures and by training the network on a more difficult task. For the first, better generalisation, we compare the average ratio of validation accuracy to training accuracy for each of the models (higher indicates less over-fitting): 0.945, 0.969, and 0.982 for KeyNet/F, KeyNet/S, and AllConv, respectively. For the second, *model expressiveness*, we compare the model's capability to fit the training data in terms of accuracy: 0.837, 0.858, and 0.907 for KeyNet/F,

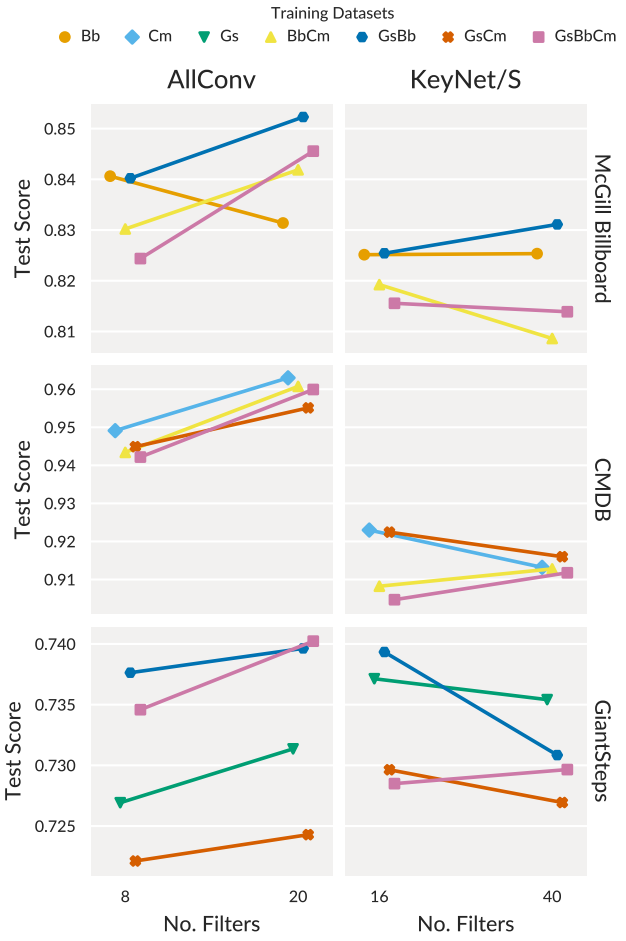


**Figure 2**. Average test scores over 10 runs for each architecture (columns), split by dataset (rows). The smaller models are on the left of each column. Colors indicate the training data used: *Bb* stands for the Billboard dataset, *Cm* for the classical music dataset, and *Gs* for the GiantSteps dataset. Each row shows the results of runs where the training set also contained the training data of the respective test set genre (e.g. in the first row, we only see runs where McGill Billboard data was included in training).

KeyNet/S, and AllConv, respectively. Stronger models that generalise better achieve better results.

### 3.4 Influence of Training Data

We then want to see how the number and genre of the datasets used for training affects results. To this end, we select the hyper-parameter settings for AllConv and KeyNet/S that achieved the best average results in the previous experiment: $N_f = 20, p = 0.1$ for AllConv, $N_f = 40, p = 0.1$ for KeyNet/S. Additionally, we consider smaller models of each type, i.e. $N_f = 8$ for AllConv and $N_f = 16$ for KeyNet/S, both without dropout. Under these settings, both architectures have a comparable number of parameters. We train these models using all possible 1, 2, and 3-combinations of the datasets, and evaluate them on all data. The results are shown in Fig. 2.

The main observations are: a) increasing model capacity is more beneficial to the AllConv model than KeyNet/S, regardless of dataset; b) adding capacity to the AllConv model enables it to better deal with diverse data—the biggest gains of additional parameters are achieved if the model is trained on a combined dataset (pink line)—while this is not always the case for KeyNet/S (see the Billboard results, where it seems that adding classical music to the training set impairs the performance of this model); c) given enough capacity in the AllConv model, training using the complete data performs better than (or almost equal to) fitting a specific genre, while the opposite is the case for KeyNet/S, where specialised models outperform the general ones. We thus argue that the AllConv model not only copes better with diverse training data, but that it leverages the diversity in the training data to perform as well as it does.

## 4. EVALUATION

Motivated by the results above, the remainder of our analysis focuses on the AllConv model. To thoroughly investigate its performance and compare it to the state of the art, we evaluate it on the following unseen datasets:

**KeyFinder:** 1 000 songs from a variety of popular music genres [6]. Unfortunately, we have only the audio for 998 of the songs available.

**Isophonics:** 180 songs by The Beatles, 19 songs by Queen, and 18 songs by Zweieck [7]. Since these songs contain key modulations, we split them into single key segments and retain only segments annotated as major or minor keys, as was done for the 2017 MIREX evaluation campaign [8].

**Robbie Williams:** 65 songs by Robbie Williams, which we also split into single key segments as outlined above [6].

**Rock:** 200 songs taken from Rolling Stone's "500 Greatest Songs of All Time" list [9] [5]. As with the McGill Billboard dataset, only the tonics are annotated. We first split the songs according to the annotated tonics, and then follow a similar procedure as described in [14]: if more than 80 % of the tonic chords are in either major or minor, the mode is set accordingly; if there are no tonic chords in a segment, we consider dominant chords in the same way.

We select the best AllConv model based on the validation score over the compound data of Electronic, Pop/Rock and Classical music. On average, models with $N_f = 20$ and dropout probability of 0.1 performed best. However, the best single model used $N_f = 24$ (see Fig. 1), and was consequently chosen as final model.

In Table 2, we compare this model to other models proposed in the academic literature. For each dataset, we show the results of the best competing system, if available. For

[6] http://www.ibrahimshaath.co.uk/keyfinder/
[7] http://isophonics.net/datasets
[8] http://www.music-ir.org/mirex/wiki/2017:Audio_Key_Detection_Results
[9] http://rockcorpus.midside.com/

| Dataset | Model | Weighted | Correct | Fifth | Relative | Parallel | Other |
|---|---|---|---|---|---|---|---|
| GiantSteps | AllConv | **74.6** | **67.9** | 7.0 | 8.1 | 4.1 | **12.9** |
| | CK1 [14] | 74.3 | **67.9** | 6.8 | 7.1 | 4.3 | 13.9 |
| Billboard | AllConv | **85.1** | **79.9** | 5.6 | 4.2 | 6.2 | **4.2** |
| | CK2 [14] | 83.9 | 77.1 | 9.0 | 4.9 | 4.2 | 4.9 |
| Classical | AllConv | 96.6 | 95.2 | 1.4 | 1.4 | 1.4 | 0.7 |
| | - | - | - | - | - | - | - |
| KeyFinder | AllConv | **76.1** | **70.0** | 5.7 | 7.4 | 4.7 | **12.1** |
| | bgate [10] | 72.4 | 65.0 | 8.6 | 6.5 | 5.4 | 14.4 |
| Isophonics | AllConv | **82.5** | **76.3** | 7.6 | 5.4 | 3.7 | **7.1** |
| | BD1 [2] | 75.1 | 66.0 | 13.6 | 5.1 | 3.9 | 9.2 |
| R. Williams | AllConv | **81.2** | **72.4** | 10.8 | 10.3 | 1.3 | **5.2** |
| | HS1 [18] | 77.1 | 68.8 | 10.1 | 9.0 | 3.2 | 9.0 |
| Rock | AllConv | 74.3 | 69.3 | 6.5 | 1.7 | 6.0 | 16.5 |
| | - | - | - | - | - | - | - |

**Table 2**. Evaluation results. Best results are in boldface.

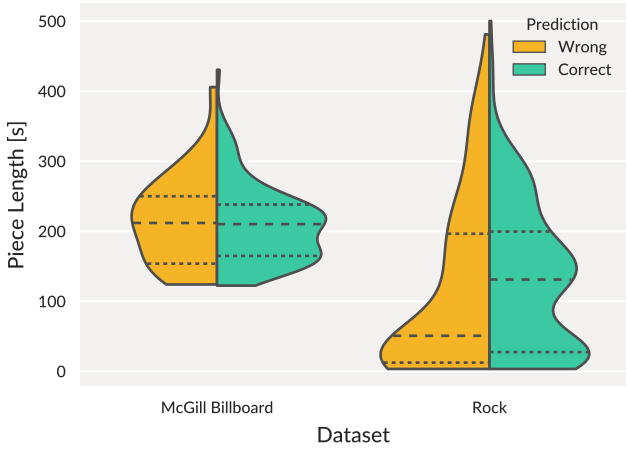


**Figure 3**. Distributions of the length of correctly and incorrectly classified excerpts depending on the dataset they come from. Densities are estimated using kernel density estimation. Horizontal lines with long dashes indicate the median, those with short dashes the quartiles. The densities are normalised, i.e. they do not indicate how many instances were classified correctly (or incorrectly), but only the distribution of except lengths within each group.

the GiantSteps and Billboard datasets, the best competing systems were variants of the neural-network-based model from [14]. For the pre-segmented Isophonics and Robbie Williams datasets, we use the results available on the MIREX 2017 website. For the KeyFinder dataset, we report the best results achieved using the open-source reference implementation [10] of the algorithms from [10].

As we can see, the proposed model *performs best for all datasets* for which comparisons were possible. Keep in mind that the systems we compare to are often specifically tuned for a genre (CK1, CK2, HS1, bgate) or set up to favour certain key modes prevalent in a dataset (BD1), while we use the same, general model for all datasets. For example, CK1 performs badly on the Billboard dataset ($w = 72.8$), BD1 on the GiantSteps ($w = 59.6$), and HS1 on the Isophonics dataset ($w = 64.1$). In this light, it is remarkable that the proposed model consistently out-performs the others.

However, the results also point us to a deficiency of the model. Recall that for some datasets (e.g. Rock), we split the files according to key annotations, and process each excerpt individually. If we compare the results on the Rock dataset with those on the Billboard dataset, we see a large discrepancy, although both sets comprise similar musical styles. As Fig. 3 demonstrates, the duration of a classified excerpt plays a major role here: for the Billboard set, the median length of excerpts classified correctly matches the one of incorrect classifications; for the Rock set, however, the median lengths differ greatly: 131 s vs. 51 s, for correctly and incorrectly classified excerpts, respectively. The distribution of excerpt lengths that are classified correctly is thus very different from the one of incorrectly classified excerpts in the Rock set. The shorter an excerpt, the more likely it is classified incorrectly.

This is not surprising per se. Determining the key of a piece requires a certain amount of musical context. However, it shows that in order to move beyond global key classification, and towards recognising key modulations, it will not suffice to detect key boundaries and apply known methods within these boundaries. To recognise key modulations, classifying short excerpts individually will reach a glass ceiling. Instead, we will need models that consider the hierarchical harmonic coherence of the whole piece.

[10] https://github.com/angelfaraldo/edmkey

## 5. CONCLUSION

We have presented a genre-agnostic key classification model based on the system developed in [14], with improvements of the training procedure and network structure. These improvements enable faster training, better generalisation, and training larger and thus more powerful models, which can leverage diverse training data instead of being impaired by it. The resulting key classifier generalises well over datasets of different musical styles, and out-performs systems that are specialised for specific genres (see Table 2).

## 6. ACKNOWLEDGEMENTS

This work is supported by the European Research Council (ERC) under the EU's Horizon 2020 Framework Programme (ERC Grant Agreement number 670035, project "Con Espressione").

## 7. REFERENCES

[1] Joshua Albrecht and Daniel Shanahan. The Use of Large Corpora to Train a New Type of Key-Finding Algorithm: An Improved Treatment of the Minor Mode. *Music Perception: An Interdisciplinary Journal*, 31(1):59–67, 2013.

[2] Gilberto Bernardes, Matthew E. P. Davies, and Carlos Guedes. Automatic musical key estimation with adaptive mode bias. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, USA, March 2017.

[3] John Ashley Burgoyne, Jonathan Wild, and Ichiro Fujinaga. An Expert Ground Truth Set for Audio Chord Recognition and Music Analysis. In *12th International Society for Music Information Retrieval Conference (ISMIR)*, Miami, USA, October 2011.

[4] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). In *International Conference on Learning Representations (ICLR), arXiv:1511.07289*, San Juan, Puerto Rico, February 2016.

[5] Trevor de Clercq and David Temperley. A corpus analysis of rock harmony. *Popular Music*, 30(01):47–70, January 2011.

[6] Bruno Di Giorgi, Massimiliano Zanoni, Augusto Sarti, and Stefano Tubaro. Automatic chord recognition based on the probabilistic modeling of diatonic modal harmony. In *Proceedings of the 8th International Workshop on Multidimensional Systems*, Erlangen, Germany, September 2013.

[7] Hamid Eghbal-Zadeh, Bernhard Lehner, Matthias Dorfer, and Gerhard Widmer. CP-JKU Submissions for DCASE-2016: A Hybrid Approach Using Binaural I-Vectors and Deep Convolutional Neural Networks. In *Detection and Classification of Acoustic Scenes and Events (DCASE)*, September 2016.

[8] Ronen Eldan and Ohad Shamir. The Power of Depth for Feedforward Neural Networks. In *Proceedings of Machine Learning Research (COLT 2016)*, New York, USA, June 2016.

[9] Ángel Faraldo, Emilia Gómez, Sergi Jordà, and Perfecto Herrera. Key Estimation in Electronic Dance Music. In Nicola Ferro, Fabio Crestani, Marie-Francine Moens, Josiane Mothe, Fabrizio Silvestri, Giorgio Maria Di Nunzio, Claudia Hauff, and Gianmaria Silvello, editors, *Advances in Information Retrieval*, volume 9626, pages 335–347. Springer International Publishing, Cham, 2016.

[10] Ángel Faraldo, Sergi Jordà, and Perfecto Herrera. A Multi-Profile Method for Key Estimation in EDM. In *Proceedings of the AES International Conference on Semantic Audio*, Erlangen, Germany, June 2017.

[11] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167*, March 2015.

[12] Filip Korzeniowski and Gerhard Widmer. Feature Learning for Chord Recognition: The Deep Chroma Extractor. In *17th International Society for Music Information Retrieval Conference (ISMIR)*, New York, USA, August 2016.

[13] Filip Korzeniowski and Gerhard Widmer. A Fully Convolutional Deep Auditory Model for Musical Chord Recognition. In *26th IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, Salerno, Italy, September 2016.

[14] Filip Korzeniowski and Gerhard Widmer. End-to-End Musical Key Estimation Using a Convolutional Neural Network. In *25th European Signal Processing Conference (EUSIPCO)*, Kos, Greece, August 2017.

[15] Shiyu Liang and R. Srikant. Why Deep Neural Networks for Function Approximation? In *International Conference on Learning Representations (ICLR), arXiv:1610.04161*, Toulon, France, April 2017.

[16] Katy Noland and Mark Sandler. Signal Processing Parameters for Tonality Estimation. In *Audio Engineering Society Convention 122*. Audio Engineering Society, May 2007.

[17] Steffen Pauws. Musical Key Extraction From Audio. In *Proceedings of the 5th International Society for Music Information Retrieval Conference (ISMIR)*, Barcelona, Spain, October 2004.

[18] Hendrik Schreiber. MIREX 2017: CNN-Based Automatic Musical Key Detection Submissions HS1/HS2/HS3. Technical report, MIREX, 2017.

[19] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for Simplicity: The All Convolutional Net. In *International Conference on Learning Representations (ICLR), Workshop Track, arXiv:1412.6806*, May 2014.

[20] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[21] David Temperley. What's Key for Key? The Krumhansl-Schmuckler Key-Finding Algorithm Reconsidered. *Music Perception: An Interdisciplinary Journal*, 17(1):65–100, October 1999.