Musical Tempo and Key Estimation using Convolutional Neural Networks with Directional Filters

Hendrik Schreiber
tagtraum industries incorporated
hs@tagtraum.com

Meinard Müller
International Audio Laboratories Erlangen
meinard.mueller@audiolabs-erlangen.de

ABSTRACT

In this article we explore how the different semantics of spectrograms' time and frequency axes can be exploited for musical tempo and key estimation using Convolutional Neural Networks (CNN). By addressing both tasks with the same network architectures ranging from shallow, domain-specific approaches to deep variants with directional filters, we show that axis-aligned architectures perform similarly well as common VGG-style networks developed for computer vision, while being less vulnerable to confounding factors and requiring fewer model parameters.

1. INTRODUCTION

In recent years Convolutional Neural Networks (CNN) have been employed for various Music Information Retrieval (MIR) tasks, such as key detection [1, 2], tempo estimation [3], beat and rhythm analysis [4–6], genre recognition [7, 8], and general-purpose tagging [9, 10]. Typically, a spectrogram is fed to the CNN and then classified in a way appropriate for the task. In contrast to recent computer vision approaches like Oxford's Visual Geometry Group's (VGG) deep image recognition network [17], some of the employed CNN architectures for MIR tasks use rectangular instead of square filters. The underlying idea is that, while for images the axes *width* and *height* have the same meaning, the spectrogram axes *frequency* and *time* have fundamentally different meaning. For MIR tasks mainly concerned with temporal aspects of music (e.g., tempo estimation, rhythmic patterns), rectangular filters aligned with the time axis appear suitable [3]. Correspondingly, tasks primarily concerned with frequency content (e.g., chord or key detection), may be approached with rectangular filters aligned with the frequency axis [11]. In fact, tempo and key estimation can be seen as tasks from two different ends of a spectrum of common MIR tasks, which are addressed by systems relying more or less on temporal or spectral signal properties (Figure 1). Systems for other tasks like general-purpose tagging or genre recognition are found more towards the center of this spectrum as they usually require both spectral and temporal information.

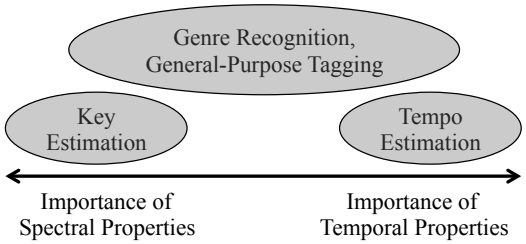

Figure 1: Several MIR tasks and their reliance on spectral or temporal signal properties.

In [12] Pons et al. explored the role of CNN filter shapes for MIR tasks. In particular, they examined using rectangular filters in shallow CNNs for automatic genre recognition of ballroom tracks. Defining *temporal* filter shapes as $1 \times n$ and *spectral* filter shapes as $m \times 1$, they showed that using temporal filters alone, 81.8% accuracy can be reached, which is in line with a Nearest Neighbour classifier (k-NN) using tempo as feature scoring 82.3% [13]. Using just spectral filters, the test network reached 59.6% accuracy, and a fusion architecture with both temporal and spectral filters performed as well as an architecture using square filters, scoring 87%. The experiments confirmed that such *directional* filters can be used to match either temporal or spectral signal properties and that both may be useful for genre recognition.

Even though directional filters did not outperform square filters, there are good arguments for using them: First, CNNs using specialized, directional filters may use fewer parameters or match musical concepts using fewer layers [14]. Second, by limiting what a filter can match, one can influence what a CNN might learn, thus better avoid "horses" [15] and improve explainability. The latter is especially interesting for genre recognition systems, given their somewhat troubled history with respect to explicit matching of musical concepts [14, 16]. To further explore how and why directional or square filters contribute to results achieved by CNN-based classification systems for MIR tasks, we believe it is beneficial to build on Pons et al.'s work and experiment with tasks that explicitly aim to recognize either high-level temporal or spectral properties, avoiding hard to define concepts like genre. Such tasks are global key and tempo estimation.

The remainder of this paper is structured as follows: In Section 2 we describe our experiments by defining both tasks, the used network variants, the training procedure,

and evaluation. The results are then presented in Section 3 and discussed in Section 4. Finally, in Section 5 we present our conclusions.

2. EXPERIMENTS

For the purpose of comparing the effects of using different filter shapes we train and evaluate different CNN architectures for the key and tempo estimation tasks using several datasets. In this section, we first describe the two tasks, then discuss the used network architectures and datasets, and finally outline the evaluation procedure.

2.1 Key Estimation

Key estimation attempts to predict the correct key for a given piece of music. Oftentimes, the problem is restricted to major and minor modes, ignoring other possible modes like Dorian or Lydian, and to pieces without modulation. Framed this way, key estimation is a classification problem with $N_{\mathrm{K}} = 24$ different classes (12 tonics, major/minor). The current state-of-the-art system is CNN-based using a VGG-style architecture with square filters [2] and a fully convolutional classification stage, as opposed to a fully connected one. This allows training on short and prediction on variable length spectrograms.

In our experiments we follow a similar approach. As input to the network (Section 2.3) we use constant-Q magnitude spectrograms with the dimensions $F_{\mathrm{K}} \times T_{\mathrm{K}} = 168 \times 60$; F_{K} being the number of frequency bins and T_{K} the number of time frames. F_{K} covers the frequency range of 7 octaves with a frequency resolution of two bins per semitone. Time resolution is $0.19\,\mathrm{s}$ per time frame, i.e. 60 frames correspond to $11.1\,\mathrm{s}$. Since all training samples are longer than $11.1\,\mathrm{s}$, we choose a random offset for each sample during each training epoch and crop the spectrogram to 60 frames. To account for class imbalances within the two modes, each spectrogram is randomly shifted along the frequency axis by $\{-4, -3, \ldots, 6, 7\}$ semitones and the ground truth labels are adjusted accordingly. We define *no shift* to correspond to a spectrogram covering the 7 octaves starting at pitch E1. In practice, we simply crop an 8 octaves spanning spectrogram that starts at C1 to 7 octaves. After cropping the spectrogram is normalized so that it has zero mean and unit variance.

2.2 Tempo Estimation

Even though tempo estimation naturally appears to be a regression task, Schreiber and Müller [3] have shown that it can also be treated as a classification task by mapping Beats Per Minute (BPM) values to distinct tempo classes. Concretely, their system maps the integer tempo values $\{30, \ldots, 285\}$ to $N_{\mathrm{T}} = 256$ classes. As input to a CNN with temporal filters and elements from [18] and [14] they use mel-magnitude-spectrograms. Even though we work with other network architectures than [3] (Section 2.3), we use the same general setup. We also treat tempo estimation as classification into 256 classes and use mel-magnitude-spectrograms with the dimensions $F_{\mathrm{T}} \times T_{\mathrm{T}} = 40 \times 256$ as input; F_{T} being the number of frequency bins and T_{T} the number of time frames. F_{T} covers the frequency range $20 - 5,000\,\mathrm{Hz}$. The time resolution is $0.46\,\mathrm{ms}$ per time frame, i.e., 256 frames correspond to $11.9\,\mathrm{s}$.

Just like the training excerpts for key estimation, the mel-spectrograms are cropped to the right size using a different randomly chosen offset during each epoch. To augment the training dataset, spectrograms are scaled along the time axis before cropping using the factors $\{0.8, 0.84, \ldots, 1.16, 1.2\}$. Ground truth labels are adjusted accordingly [3]. After cropping and scaling spectrograms are normalized ensuring zero mean and unit variance per sample.

Module
`ShallowMod`
`ClassMod`

(a) `Shallow`

Module	Size
`DeepMod`	$\ell = 0$
`DeepMod`	$\ell = 1$
`DeepMod`	$\ell = 2$
`DeepMod`	$\ell = 2$
`DeepMod`	$\ell = 3$
`DeepMod`	$\ell = 3$
`ClassMod`	

(b) `Deep`

Table 1: Used network architectures. (a) `Shallow` architecture consisting of a variant of the `ShallowMod` module and a `ClassMod` module. (b) `Deep` architecture consisting of multiple, `DeepMod` modules parameterized with ℓ to influence the filter count and a `ClassMod` module.

2.3 Network Architectures

To gain insights into how filter shapes affect performance of CNN-based key and tempo estimation systems we run experiments with two very different architectures: a relatively shallow but specialized one, and a commonly used much deeper one from the field of computer vision. Both architectures are used for both tasks.

2.3.1 Shallow Architectures

Our `Shallow` architectures, outlined in Table 1a, consists of two parts: the feature extraction module `ShallowMod` and the classification module `ClassMod`. `ShallowMod`, depicted in Table 2a, is inspired by a classic signal processing approach that first attempts to find local spectrogram peaks along one axis, averages these peaks over the other axis, and then attempts to find a global pattern, i.e., a periodicity for tempo estimation [19] and a pitch profile for key detection [20]. In terms of CNNs this means that our first convolutional layer consists of short directional filters (local peaks), followed by a one-dimensional average pooling layer that is orthogonal to the short filters, followed by a layer with long directional filters (global pattern) that stretch in the same direction as the short filters. We use ReLU as activation function for the convolutional layers and to avoid overfitting we add a dropout layer [21] after each ReLU. The parameters k and p_{D} let us scale the number of convolutional filters and dropout probabilities.

(a) `ShallowMod`

Layer	Temp	Spec	Square
Input			
Conv, ReLU	$k, 1 \times 3$	$k, 3 \times 1$	n.a.
Dropout	p_{D}	p_{D}	n.a.
AvgPool	$F_{\mathrm{T}} \times 1$	$1 \times T_{\mathrm{K}}$	n.a.
Conv, ReLU	$64k, 1 \times T_{\mathrm{T}}$	$64k, F_{\mathrm{K}} \times 1$	n.a.
Dropout	p_{D}	p_{D}	n.a.

(b) `DeepMod`

Layer	Temp	Spec	Square
Input			
Conv, ReLU	$2^\ell k, 1 \times 5$	$2^\ell k, 5 \times 1$	$2^\ell k, 5 \times 5$
BatchNorm			
Conv, ReLU	$2^\ell k, 1 \times 3$	$2^\ell k, 3 \times 1$	$2^\ell k, 3 \times 3$
BatchNorm			
MaxPool	2×2	2×2	2×2
Dropout	p_{D}	p_{D}	p_{D}

(c) `ClassMod`

Layer	Temp	Spec	Square
Input			
Conv, ReLU	$N_{\mathrm{T}}, 1 \times 1$	$N_{\mathrm{K}}, 1 \times 1$	n.a.
GlobalAvgPool			
Softmax			

Table 2: Layer definitions for the three modules `ShallowMod`, `ClassMod`, and `DeepMod`, describing number of filters (e.g., k or $64k$) and their respective shapes (e.g., 1×3 or 5×5).

`ShallowMod` is followed by a fully convolutional classification module named `ClassMod` (Table 2c), which consists of a 1×1 bottleneck layer (pointwise convolution) with as many filters as desired classes (N_{K} or N_{T}), a global average pooling layer, and the $\mathrm{softmax}$ activation function. Note, that because all directional filters are identically aligned, the model has an asymmetric, *directional capacity*, i.e., it has a much larger ability to describe complex relationships in one direction than in the other.

We use the same general architecture for both key and tempo estimation. The only differences are the filter and pooling directions and dimensions. For tempo estimation we use temporal filters with pooling along the frequency axis, and for key estimation spectral filters with pooling along the time axis. Both architectures are named after their filter directions, `ShallowTemp` and `ShallowSpec`, respectively. We also adjust the pooling and the long filters shape to the size of the input spectrogram, which is different for the two tasks.

2.3.2 Deep Architectures

The second architecture, `Deep` (Table 1b), is a common VGG-style architecture consisting of six parameterized feature extraction modules `DeepMod` (Table 2b) followed by the same classification module that we have already used in `Shallow`. Each of the feature extraction modules contains a convolutional layer with 5×5 filters followed by a convolutional layer with 3×3 filters. The convolutional layers consist of $2^\ell k$ filters each, with network parameter k and module parameter ℓ. While ℓ influences the number of filters in an instance of `DeepMod`, k lets us scale the total number of parameters in the network. As shown in Table 1b, deeper instances have more filters. The convolutional layers are followed by a 2×2 max pooling layer. Should pooling not be possible along an axis, because the output from the previous layer is only 1 wide or high, pooling is skipped along that axis. This happens for example, when a tempo spectrogram with its 40 bands passes through more than 5 max pools. Each pooling layer is followed by a dropout layer with probability p_{D}. To counter covariate shift, we add batch normalization [22] layers after each convolutional layer.

The general structure of the `Deep` architecture is customized neither for the key nor for the tempo task. However, in order to investigate how different filter shapes affect the network's performance, we modify the described architecture by replacing the square convolutional kernels with directional ones, i.e., 3×3 with 1×3 or 3×1, and 5×5 with 1×5 or 5×1. Analogous to the naming scheme used for shallow networks, we denote the directional variants `DeepTemp` and `DeepSpec`. The original variant is named `DeepSquare`.

Split	Key Datasets
Training	80% of `LMD Key` ∪ 80% of `MTG Key`
Validation	10% of `LMD Key` ∪ 20% of `MTG Key`
Testing	`GiantSteps Key`, `GTzan Key`, 10% of `LMD Key`

Split	Tempo Datasets
Training	80% of `EBall` ∪ 80% of `MTG Tempo` ∪ 80% of `LMD Tempo`
Validation	20% of `EBall` ∪ 20% of `MTG Tempo` ∪ 10% of `LMD Tempo`
Testing	`GiantSteps Tempo`, `GTzan Tempo`, 10% of `LMD Tempo`, `Ballroom`

Table 3: Dataset splits used in key (top) and tempo (bottom) estimation experiments.

2.4 Datasets

We use the following publicly available datasets from different genres for both training and evaluation (listed in alphabetical order). The used splits are randomly chosen and listed in Table 3.

`Ballroom` (698 samples): 30 s excerpts with tempo annotations [23].

`EBall` (3,826 samples): 30 s excerpts with tempo annotations, excluding tracks also occurring in the regular `Ballroom` dataset [3, 23, 24].

`GiantSteps Key` (604 samples): 2 min excerpts of electronic dance music (EDM) [25].

`GiantSteps Tempo` (661 samples): 2 min excerpts of EDM [25]. Revised tempo annotations from [26].

GTzan Key (836 samples): 30 s excerpts from 10 different genres [27]. Key annotations from [28]. [1] Most tracks with missing key annotations belong to the genres classical, jazz, and hip-hop.

GTzan Tempo (999 samples): 30 s excerpts from 10 different genres [27]. Tempo annotations from [29].

LMD Key (6,981 samples): 30 s excerpts, predominantly rock and pop [30, 31]. Due to a MIDI peculiarity, this dataset does not contain any tracks in C major. Some form of data augmentation as described above is therefore necessary.

LMD Tempo (3,611 samples): 30 s excerpts, predominantly rock and pop [3, 30].

MTG Tempo / MTG Key (1,158 samples): 2 min EDM excerpts annotated with both key and tempo [3, 32]. We used only tracks that are still publicly available, have an unambiguous key, and a high key annotation confidence. [2]

2.5 Evaluation

Since the proposed network architectures are fully convolutional, we can choose at prediction time to pass a track either in one long spectrogram or as multiple shorter windows through the network. In the latter case, predictions for all windows would have to be aggregated. Informal experiments did not show a remarkable difference. For this work we choose to predict values for whole spectrograms.

When evaluating key estimation systems either a simple accuracy or a score is used that assigns additional value to musically justifiable mistakes, like being off by a perfect fifth. [3] For this work, we choose to only report the percentage of correctly classified keys. Tempo estimation systems are typically evaluated using the metrics *Accuracy1* and *Accuracy2*. While *Accuracy1* reports the percentage of correctly estimated tempi allowing a 4% tolerance, *Accuracy2* additionally permits so-called octave errors, i.e., errors by a factor of 2 and 3 [23]. We choose to report only *Accuracy1*.

For training we use Adam [33] as optimizer with a learning rate of 0.001, a batch size of 32, and early stopping once the validation loss has not decreased any more during the last 100 epochs. In this work, one epoch is defined as having shown all training samples to the network once, regardless of augmentation. We choose k so that we can compare architectures with similar parameter counts. Shallow is trained with $k \in \{1, 2, 4, 6, 8, 12\}$ and Deep with $k \in \{2, 4, 8, 16, 24\}$. Additionally, DeepSquare is trained with $k = 1$. For both architectures we apply various dropout probabilities $p_D \in \{0.1, 0.3, 0.5\}$. Each variant is trained 5 times and mean validation accuracy along with its standard deviation is recorded for each variant. In total we train 420 models with 84 different configurations.

[1] https://github.com/alexanderlerch/gtzan_key

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

[3] https://www.music-ir.org/mirex/wiki/2018:Audio_Key_Detection

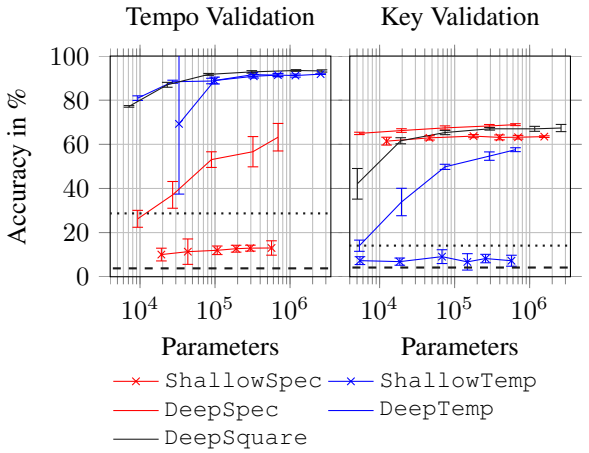

Figure 2: Mean validation accuracies for the various network configurations depending on their number of network parameters. Only the best dropout configuration is shown. Whiskers represent the standard deviation based on 5 runs.

For testing, we pick the dropout variant of each network class that performed best on the validation set and evaluate it against the test datasets. Again, we report the mean accuracies for 5 runs along with their standard deviations.

3. RESULTS

Figure 2 shows mean validation accuracies of 5 runs for each configuration, using their best performing dropout probability p_D. The dashed black line is the accuracy a random classifier achieves, and the dotted black line shows the accuracy of the algorithm that always outputs the class that most often occurs in the validation set. With accuracy values slightly above random, ShallowSpec and ShallowTemp perform worst of all architectures, when used for the task they were *not* meant for. But when used for the task they were designed for, both perform well. A higher number of parameters leads to slightly better results. When training ShallowTemp with $k = 1$ for the tempo task, the network performed very poorly in one of the five runs, which is the cause for the very large standard deviation of 32.2 shown in Figure 2. The mean accuracy for the 4 successful runs was 85.2%. When comparing with the Deep architectures, we see that DeepTemp performs just as well as ShallowTemp with $k > 1$ on the tempo task, and DeepSpec clearly outperforms ShallowSpec on the key task. Surprisingly, the DeepSpec architecture reaches fairly high accuracy values (up to 63%) on the tempo task when increasing the model capacity via k, even though it only has convolutional filters aligned with the frequency axis. We can make a similar observation for the DeepTemp architecture. It too reaches relatively high accuracy values on the key task (up to 57%) when increasing k. The unspecialized DeepSquare is by a small margin the best performing architecture for the tempo task, and comes in as a close second for key detection with $k > 1$. But for $k = 1$, DeepSquare performs considerably worse than DeepSpec with $k = 2$ (42% compared to 64%), even though both have similar parameter counts

of ca. 5 000.

We selected the dropout variant for each architecture and parameter setting with the best validation accuracy and ran predictions on the test sets. Detailed results are shown in Figure 3. The general picture is very similar to validation: `Deep` architectures tend to perform slightly better than `Shallow` architectures on the tasks they are meant for and `Shallow` architectures perform poorly on the task they were not meant for. In fact, `ShallowTemp` performs no better on `GTzan Key` and `GiantSteps Key` than the random baseline. For both key and tempo `DeepSquare` performs as well or better than any other architecture, except when drastically reducing the model capacity for the key task ($k = 1$). Then accuracy decreases well below `DeepSpec`'s performance with similar parameter count: 33% compared to 50% for `GTzan Key`, and 21% compared to 51% for `GiantSteps Key`.

To provide an absolute comparison, we chose the best performing representative from each architecture (based on validation accuracy, regardless of dropout configuration or capacity), and calculated accuracies for each test set (Table 4, incl. reference values from the literature). In 5 out of 7 test cases `DeepSquare` reaches the highest accuracy score among our architectures. The other two are reached by `DeepTemp` for `GiantSteps Tempo` and by `DeepSpec` for `LMD Key`. For both tasks we observe that the margin by which the best performing network is better than the second best for a given dataset differs considerably. `DeepSquare` reaches an accuracy of 92.4% for the `Ballroom` tempo dataset, which is 4.2 pp (percentage points) better than the second best network (`DeepTemp`, 88.2%). The differences between best and second best accuracy are considerably lower for the other datasets: 1.7 pp (`LMD Tempo`), 1.6 pp (`GTzan Tempo`), and 0.6 pp (`GiantSteps Tempo`). For the key task, `DeepSquare` reaches an accuracy of 49.9% on `GTzan Key`, which is 5.1 pp better than the second best network (`DeepSpec`, 44.8%), while the differences between best and second best for the other datasets are 3.1 pp (`GiantSteps Key`), and 2.4 pp (`LMD Key`).

4. DISCUSSION

The results show that simple shallow networks with axis-aligned, directional filters can perform well on both the key and tempo task. Conceptually, both tasks are similar enough that virtually the same architecture can be used for either one, as long as the input representation and the filter direction are appropriate. Using the wrong filter direction, e.g., `ShallowSpec` for the tempo task, leads to very poor results close to the random baseline. Together, this strongly supports the hypothesis that the `Shallow` architecture indeed learns what we want it to learn, i.e., pitch patterns for key detection or rhythmic patterns for tempo detection, but not both.

This stands in contrast to the standard VGG-style network (`DeepSquare`). Because of its square filters, we cannot be certain what it learns, just by analyzing its static architecture. It is designed to pick up on anything that could provide a hint towards correct classification, be it rhythm and pitch patterns, or timbral properties like instrumentation. And indeed our experiment shows that without being specialized for either key or tempo estimation in any way, `DeepSquare` works very well for both tasks. In Section 3 we noted that `DeepSquare` achieved the greatest tempo accuracy for `Ballroom` and the greatest key accuracy for `GTzan Key` by a considerable margin of 4.2 pp

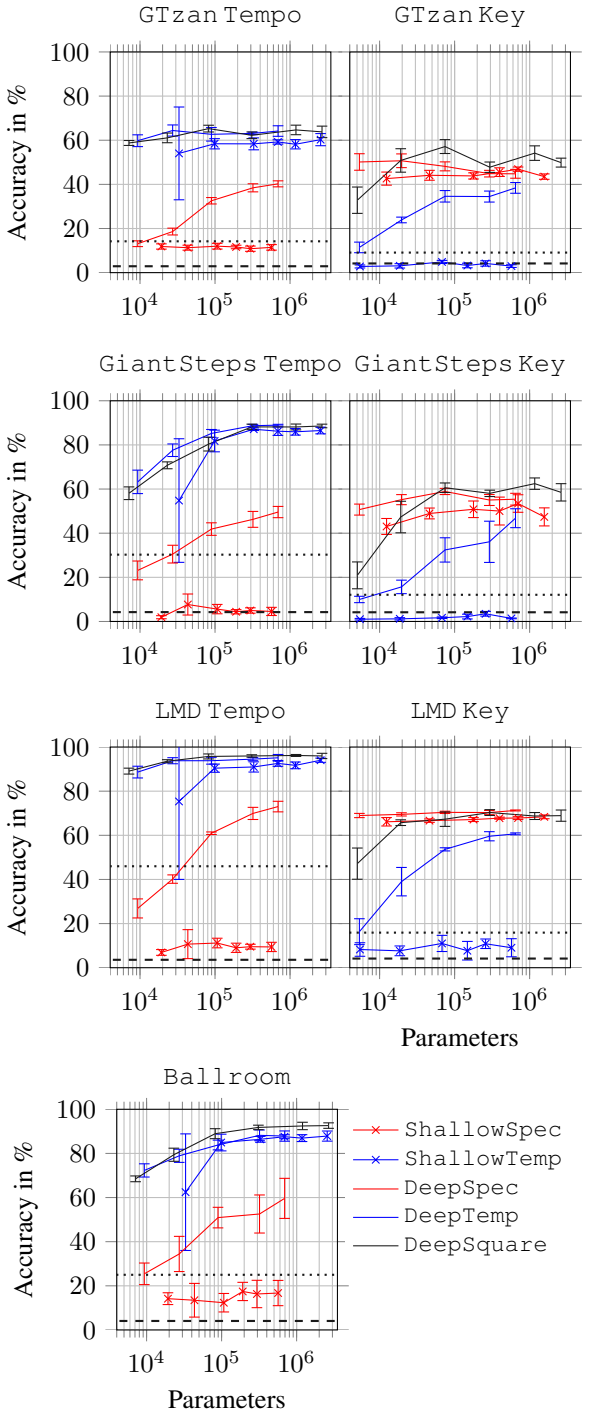

Figure 3: Mean test accuracies for various network configurations and datasets depending on their number of network parameters. Whiskers represent one standard deviation based on 5 runs. Dropout was chosen based on performance during validation.

Architecture	GS	GT	LMD	BR
ShallowTemp	86.5 1.5	60.3 2.7	94.0 1.0	87.9 2.3
DeepTemp	**88.7** 0.6	63.1 0.6	94.5 0.7	88.2 2.4
ShallowSpec	4.5 1.9	11.5 1.3	9.4 2.1	16.7 5.7
DeepSpec	49.6 2.5	40.2 1.4	73.0 2.4	59.6 9.1
DeepSquare	88.1 1.3	**64.7** 2.1	**96.2** 0.4	**92.4** 1.7
Literature	82.5 [3]	78.3 [29]	—	92.0 [3]

(a) Tempo

GS	GT	LMD
1.7 0.4	4.9 0.7	11.0 3.7
46.8 4.3	38.4 2.4	60.7 0.4
50.8 3.8	43.8 1.4	67.1 0.9
55.4 2.7	44.8 2.0	**71.3** 0.2
58.5 3.9	**49.9** 2.0	68.9 2.5
67.9 [2]	~45 [28]	—

(b) Key

Table 4: Mean estimation accuracies of 5 runs with standard deviation (small font). Best results per test are set in **bold**. Model variants chosen based on validation performance (ignoring parameter count). GS=GiantSteps, GT=GTzan, LMD=LMD, BR=Ballroom.

and 5.1 pp, respectively. This margin may be a result of the fact that key and tempo are related to genre [34–37]. Specifically, in Ballroom there is a strong correlation between genre and tempo, and GTzan Key is the key test set with the greatest genre diversity and therefore stands to benefit the most from an architecture that can distinguish genres based on *both* temporal and timbral properties. Consequently, square filters *should* improve accuracy results for these datasets. But this does not conclusively show that only the network's ability to measure specifically key or tempo is reflected by these results, as the system is by design vulnerable to confounds [15]. By using directional filters in DeepSpec and DeepTemp we intentionally limit the standard VGG-style architecture in a way that seeks to lessen this vulnerability as well as reduce the number of required parameters.

The results for DeepSpec and DeepTemp show that a VGG-style network with directional filters can perform very well on either task. For networks with a large number of parameters test results are similar to DeepSquare, with a tendency towards a slightly worse performance. Interestingly, the situation is different for low-capacity networks with $k = 2$ for DeepSpec, and $k = 1$ for DeepSquare. Here, DeepSpec clearly outperforms DeepSquare, even though the parameter count is similar. Perhaps with ca. 5 000 parameters DeepSquare simply does not have enough capacity aligned in the right direction to still perform well on the task.

The fact that DeepSpec and DeepTemp with $k = 2$ perform very poorly on the tasks they are not meant for, supports the hypothesis that they only learn the intended features for the tasks they are meant for. For $k > 2$ we cannot be quite as certain, as both architectures reach higher accuracy scores on the tasks they were not meant for for greater values of k. We believe this effect may be a result of the 2×2 max pooling in the DeepMod modules.

5. CONCLUSIONS

We have shown that shallow, signal processing-inspired CNN architectures using directional filters can be used successfully for both tempo and key detection. By using shallow networks designed for key detection on the tempo task and vice versa, we were able to experimentally support the hypothesis that these networks are incapable of matching information from the domain they were not meant for, which would make them less susceptible to confounds.

We further demonstrated that a standard VGG-style architecture can be used for tempo estimation, as it has been shown before for key detection [2]. By replacing square filters with directional filters, we derived a musically motivated, directional VGG-variant that performs similarly well as the original one, but is less vulnerable to confounds, especially when used for key detection with low capacity models. In such scenarios we were also able to observe efficiency gains, i.e., better performance than the standard VGG-style network with similar parameter counts.

Additional Material

Code to recreate models and reproduce the reported results can be found at https://github.com/hendriks73/directional_cnns.

Acknowledgments

The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institute for Integrated Circuits IIS. Meinard Müller is supported by the German Research Foundation (DFG MU 2686/11-1).

6. REFERENCES

[1] F. Korzeniowski and G. Widmer, "End-to-end musical key estimation using a convolutional neural network," in *Proceedings of the 25th European Signal Processing Conference (EUSIPCO)*, Kos, Greece, 2017.

[2] ——, "Genre-agnostic key classification with convolutional neural networks," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, September 2018.

[3] H. Schreiber and M. Müller, "A single-step approach to musical tempo estimation using a convolutional neural network," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, September 2018.

[4] A. Holzapfel and T. Grill, "Bayesian meter tracking on learned signal representations," in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, New York, NY, USA, August 2016, pp. 262–268.

[5] S. Durand, J. P. Bello, B. David, G. Richard, S. Durand, J. P. Bello, B. David, G. Richard, B. David, G. Richard *et al.*, "Robust downbeat tracking using an ensemble of convolutional networks," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 25, no. 1, pp. 76–89, 2017.

[6] A. Gkiokas and V. Katsouros, "Convolutional neural networks for real-time beat tracking: A dancing robot application," in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, October 2017, pp. 286–293.

[7] S. Dieleman, P. Brakel, and B. Schrauwen, "Audio-based music classification with a pretrained convolutional network." in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, 2011, pp. 669–674.

[8] A. Schindler, T. Lidy, and A. Rauber, "Comparing shallow versus deep neural network architectures for automatic music genre classification," in *Proceedings of the 9th Forum Media Technology (FMT2016)*, vol. 1734, St. Pölten, Austria, November 2016, pp. 17–21.

[9] M. Dorfer and G. Widmer, "Training general-purpose audio tagging networks with noisy labels and iterative self-verification," DCASE2018 Challenge, Tech. Rep., September 2018.

[10] K. Choi, G. Fazekas, and M. B. Sandler, "Automatic tagging using deep convolutional neural networks," in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, New York, NY, USA, August 2016, pp. 805–811.

[11] B. McFee and J. P. Bello, "Structured training for large-vocabulary chord recognition," in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, October 2017, pp. 188–194.

[12] J. Pons, T. Lidy, and X. Serra, "Experimenting with musically motivated convolutional neural networks," in *Proceedings of 14th International Workshop on Content-Based Multimedia Indexing (CBMI)*. Bucharest, Romania: IEEE, June 2016, pp. 1–6.

[13] F. Gouyon, S. Dixon, E. Pampalk, and G. Widmer, "Evaluating rhythmic descriptors for musical genre classification," in *AES 25th International Conference*, London, UK, 2004.

[14] J. Pons and X. Serra, "Designing efficient architectures for modeling temporal features with convolutional neural networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. New Orleans, USA: IEEE, March 2017, pp. 2472–2476.

[15] B. L. Sturm, "A simple method to determine if a music information retrieval system is a "horse"," *IEEE Transactions on Multimedia*, vol. 16, no. 6, pp. 1636–1644, October 2014.

[16] ——, "Classification accuracy is not enough," *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 371–406, 2013.

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

[18] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich *et al.*, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, June 2015, pp. 1–9.

[19] A. P. Klapuri, "Sound onset detection by applying psychoacoustic knowledge," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Washington, DC, USA, 1999, pp. 3089–3092.

[20] C. L. Krumhansl, *Cognitive foundations of musical pitch*, ser. Oxford Psychology Series. Oxford University Press, 1990, no. 17.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[22] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[23] F. Gouyon, A. P. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano, "An experimental comparison of audio tempo induction algorithms," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1832–1844, 2006.

[24] U. Marchand and G. Peeters, "The extended ballroom dataset," in *Late Breaking Demo of the International Conference on Music Information Retrieval (ISMIR)*, New York, NY, USA, 2016.

[25] P. Knees, Á. Faraldo, P. Herrera, R. Vogl, S. Böck, F. Hörschläger, and M. Le Goff, "Two data sets for tempo estimation and key detection in electronic dance music annotated from user corrections," in *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, Málaga, Spain, October 2015, pp. 364–370.

[26] H. Schreiber and M. Müller, "A crowdsourced experiment for tempo estimation of electronic dance music," in *Proceedings of the 19th International Society*

for Music Information Retrieval Conference (ISMIR), Paris, France, September 2018.

[27] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[28] S. Kraft, A. Lerch, and U. Zölzer, "The tonalness spectrum: feature-based estimation of tonal components," in *Proceedings of the 16th International Conference on Digital Audio Effects (DAFx)*, Maynooth, Ireland, September 2013, p. 8.

[29] G. Percival and G. Tzanetakis, "Streamlined tempo estimation based on autocorrelation and cross-correlation with pulses," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 22, no. 12, pp. 1765–1776, 2014.

[30] C. Raffel, "Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching," Ph.D. dissertation, Columbia University, 2016.

[31] H. Schreiber, "CNN-based automatic musical key detection," in *Music Information Retrieval Evaluation eXchange (MIREX)*, Suzhou, China, October 2017.

[32] Á. Faraldo, S. Jordà, and P. Herrera, "A multi-profile method for key estimation in EDM," in *Proceedings of the AES International Conference on Semantic Audio*. Erlangen, Germany: Audio Engineering Society, June 2017.

[33] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[34] F. Hörschläger, R. Vogl, S. Böck, and P. Knees, "Addressing tempo estimation octave errors in electronic music by incorporating style information extracted from wikipedia," in *Proceedings of the Sound and Music Computing Conference (SMC)*, Maynooth, Ireland, 2015.

[35] B. Schuller, F. Eyben, and G. Rigoll, "Tango or waltz?: Putting ballroom dance style into tempo detection," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2008, p. 12, 2008.

[36] L. Xiao, A. Tian, W. Li, and J. Zhou, "Using statistic model to capture the association between timbre and perceived tempo." in *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR)*, Philadelphia, USA, 2008.

[37] Á. Faraldo, E. Gómez, S. Jordà, and P. Herrera, "Key estimation in electronic dance music," in *European Conference on Information Retrieval*. Springer, 2016, pp. 335–347.