Appeared as: Jan Schlüter and Sebastian Böck. Improved Musical Onset Detection with Convolutional Neural Networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6979–6983, 2014.

http://doi.org/10.1109/ICASSP.2014.6854953

IMPROVED MUSICAL ONSET DETECTION WITH CONVOLUTIONAL NEURAL NETWORKS

Jan Schlüter

Austrian Research Institute for Artificial Intelligence, Vienna
jan.schlueter@ofai.at

Sebastian Böck

Department of Computational Perception, Johannes Kepler University, Linz, Austria
sebastian.boeck@jku.at

ABSTRACT

Musical onset detection is one of the most elementary tasks in music analysis, but still only solved imperfectly for polyphonic music signals. Interpreted as a computer vision problem in spectrograms, Convolutional Neural Networks (CNNs) seem to be an ideal fit. On a dataset of about 100 minutes of music with 26k annotated onsets, we show that CNNs outperform the previous state-of-the-art while requiring less manual preprocessing. Investigating their inner workings, we find two key advantages over hand-designed methods: Using separate detectors for percussive and harmonic onsets, and combining results from many minor variations of the same scheme. The results suggest that even for well-understood signal processing tasks, machine learning can be superior to knowledge engineering.

Index Terms— Music information retrieval, Multi-layer neural network

1. INTRODUCTION

Detecting musical onsets – i.e., finding the starting points of all musically relevant events in an audio signal – is the first step for several higher-level music analysis tasks such as beat detection, tempo estimation and transcription. On a spectral representation, onset detection is closely related to edge detection in images: Onsets are characterized by a swift change of spectral content over time, and can even be accompanied by wide-band transients clearly visible in a spectrogram (Fig. 3a, 3b). Highlighting sharp oriented edges in an image requires local information only and can be accomplished by convolution with a small filter kernel, which is the basic operation of a Convolutional Neural Network (CNN). Figure 1 demonstrates this by convolving a grayscale photograph with a 5x5 patch of random values. This lead to the idea of training such a network to find onsets in spectrogram excerpts: If a randomly initialized CNN already detects edges, it should quickly learn a set of suitable filter kernels to detect onsets.

In this work, we extend our preliminary experiments presented in [**?**] to set a new state of the art for musical onset detection. Following a review of related work in Sect. 2, we will describe the network architecture and training method in Sect. 3, present quantitative experimental results in Sect. 4 and perform a qualitative analysis of a trained network in Sect. 5 to understand its inner workings.

2. RELATED WORK

First attempts at onset detection relied on traditional signal processing, exploiting changes of spectral energy [**?**, **?**, **?**, **?**], pitch [**?**, **?**] or phase [**?**, **?**, **?**] accompanying an onset.

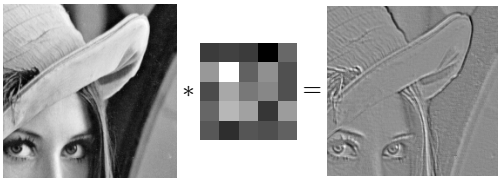


Fig. 1: Convolving an image (*left*) with a random 5x5 kernel (*center*, enlarged) finds oriented edges (*right*).

Neural networks have been successfully explored for the task as well. Marolt et al. [**?**] use neural networks to improve peak picking on a hand-crafted onset detection function, but do not learn the detection function itself and restrict their experiments to piano music. Lacoste and Eck [**?**] learn an onset detector on spectral data with neural networks, but propose convolution for future work only. Eyben et al. [**?**] train a bidirectional Recurrent Neural Network (RNN) on Mel-scaled magnitude spectrograms preprocessed with a time difference filter. Böck et al. [**?**, **?**, **?**] refine this model in several steps, defining the current state of the art in onset detection.

Convolutional learning on music audio data has been evaluated for genre and artist classification [**?**, **?**, **?**], tagging [**?**], key detection [**?**] and chord detection [**?**]. Although results are promising, CNNs have never been applied to the comparably low-level task of onset detection, apart from our initial experiments [**?**].

3. METHOD

We will first give an introduction to CNNs in general, then explain how we apply them to the task at hands, and finally discuss the chosen training methods.

3.1. Convolutional Neural Networks

CNNs are feed-forward neural networks characterized by their *convolutional* layers, in which neurons are spatially arranged to form *feature maps*. Each neuron is connected to a fixed-size local region of the input corresponding to its position in the map, and all neurons in a map share weights. The output computed by a feature map can be interpreted as a convolution of its input with a small filter kernel (the shared weights), followed by an elementwise nonlinearity. Compared to a fully connected layer, a feature map retains the spatial layout of the input data and has a much lower number of trainable parameters, both of which can help learning on large inputs. Optionally, a convolutional layer can be followed by a *pooling* layer that subsamples each feature map by retaining, e.g., only the max-

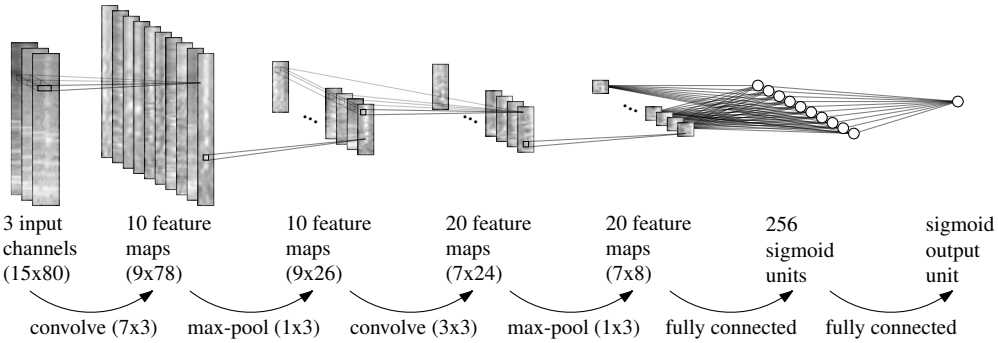


Fig. 2: One of the Convolutional Neural Network architectures used in this work. Starting from a stack of three spectrogram excerpts, convolution and max-pooling in turns compute a set of 20 feature maps classified with a fully-connected network.

imum value in non-overlapping 2x2 pixel cells. This both reduces the amount of data and introduces some translation invariance. To be used for classification, the computation chain of a CNN ends in a fully-connected network that integrates information across all locations in all feature maps of the layer below. When introduced, this type of architecture set the state-of-the-art in handwritten digit recognition [**?**], and still defines the state-of-the-art on several computer vision tasks [**?**].

3.2. Application to Onset Detection

To be used as an onset detector, we train a CNN on spectrogram excerpts centered on the frame to classify, giving binary labels to distinguish onsets from non-onsets (see Fig. 2).

Computer vision usually uses square filters, and square pooling. In spectrograms, the two dimensions represent two different modalities, though, and we found rectangular shapes to be more effective (cf. [**?**]). In particular, as the task mostly entails finding changes over time, we use filters wide in time and narrow in frequency, and as the task requires results of high time resolution, but is oblivious to frequency, we perform max-pooling over frequencies only.

Computer vision often handles color images, presenting the input such that each neuron has access to the same local region in all color channels (e.g., red, green, and blue). Here we train on a stack of spectrograms instead, with different window sizes, but the same frame rate, and reduced to the same number of frequency bands with logarithmic filter banks. This way each neuron can combine information of high temporal and high frequency accuracy for its location.

To detect onsets in a test signal, we compute the spectrograms and feed them to the network (instead of giving excerpts of the size used in training, we can apply the convolution and pooling operations to the full input at once), obtaining an onset activation function over time. This function is smoothed by convolution with a Hamming window of 5 frames, and local maxima higher than a given threshold are reported as onsets.

3.3. Training methodology

We train our networks using mini-batch gradient descent with momentum, minimizing cross-entropy error. As an extension to our experiments in [**?**], for each training case we randomly drop 50% of the inputs of the two fully-connected layers and double the remaining connection weights, to improve generalization and avoid the need for early stopping (see [**?**]). As another extension, we note that for our spectrogram frame rate (100 Hz), assigning each annotated onset to a single frame may be inappropriate – some annotations are not accurate enough, and some onsets are not that sharp –, so we assign it to three frames instead, weighting the extra frames less in training. We will investigate the effect of our extensions in the experiments.

4. EXPERIMENTAL RESULTS

Starting from the initial experiment of [**?**], we perform several modifications to both architecture and training, yielding a further improvement over the previous state of the art. We will report on these improvements in detail after describing the data and evaluation method.

4.1. Data

We evaluate our networks on a dataset of about 102 minutes of music annotated with 25,927 onsets detailed in [**?**, p. 4] and also used in [**?**]. It contains monophonic and polyphonic instrumental recordings as well as popular music excerpts. Following [**?**], we compute three magnitude spectrograms with a hop size of 10 ms and window sizes of 23 ms, 46 ms and 93 ms. We apply an 80-band Mel filter from 27.5 Hz to 16 kHz and scale magnitudes logarithmically. We normalize each frequency band to zero mean and unit variance (constants computed on a hold-out set). The network input for a single decision consists of the frame to classify plus a context of ± 70 ms (15 frames in total), from all three spectrograms, which is about the context we found the RNN of [**?**] to use.

4.2. Evaluation

As in [**?**, **?**], a reported onset is considered correct if it is not farther than 25 ms from an unmatched target annotation; any excess detections and targets are false positives and negatives, respectively. From the precision/recall curve obtained by varying the threshold, we report metrics for the point of optimal F-score only. As in [**?**, **?**], all results are obtained in 8-fold cross-validation.

4.3. Initial Architecture

Our initial architecture from [**?**] is depicted in Fig. 2: From the 3-channel spectrogram excerpts of 15 frames by 80 bands, a convolutional layer with filters of 7 frames by 3 bands (by 3 channels) computes 10 feature maps of 9 frames by 78 bands. The next layer performs max-pooling over 3 adjacent bands without overlap, reducing the maps to 26 bands. Another convolutional layer of 3×3 filters

	Precision	Recall	F-score
RNN [**?**, **?**]	0.892	0.855	0.873
CNN [**?**]	0.905	0.866	0.885
+ Dropout	0.909	0.871	0.890
+ Fuzziness	0.914	0.885	0.899
+ ReLU	**0.917**	**0.889**	**0.903**
SuperFlux [**?**]	0.883	0.793	0.836

Table 1: Performance of the state-of-the-art RNN compared to the proposed CNN and a hand-designed method. See Sections 4.3–4.6 for details on rows 2–5.

and another 3-band max-pooling layer result in 20 maps of 7 frames by 8 bands (1120 neurons in total). These are processed by a fully-connected layer of 256 units and a final fully-connected layer of a single output unit predicting onsets. Both convolutional layers use the tanh nonlinearity (with a scalar bias per feature map), and the fully-connected layers use the logistic sigmoid.

The network is trained in mini-batches of 256 examples, for 100 epochs, using a fixed learning rate of 0.05, and an initial momentum of 0.45, linearly increased to 0.9 between epochs 10 and 20.

It achieves an F-score of 88.5%, about one percent point above the state-of-the-art RNN (Table 1, rows 1–2). (Trained on single-channel spectrograms, both models lose about one percent point.)

4.4. Bagging and Dropout

Bagging is a straightforward way to improve the performance of a classifier without changing its architecture: Training four RNNs and averaging their outputs gives a slight improvement to 87.7% F-score. Similarly, bagging two of our CNNs improves results to 89.1%, but four CNNs perform the same. A single CNN with twice the number of units in each layer overfits and obtains 87.9% only. Jointly training two CNNs connected to the same output unit does not overfit, but is inferior to training them separately. We conclude that the benefit of bagging over simply enlarging the network does not stem from the fact that its constituent parts do not overfit, but that they are forced to solve the task on their own – when training two CNNs jointly, the second will not receive any learning signal when the first produces the correct answer with high confidence and vice versa.

The same holds for the hidden units *within* each CNN. An elegant way to ensure that each unit receives a learning signal and is encouraged to solve its task independently of its peers is using *dropout* [**?**]: For each training case, half of the units are omitted from the network (cheaply accomplished by masking their output), chosen at random, and remaining weights are doubled to compensate. Applying this to the inputs of the two fully-connected layers and increasing the learning rate to 1.0, multiplied with 0.995 after each epoch, yields 89.0% F-score. Note that dropout does not incur any higher costs at test time, while bagging two CNNs is twice as expensive. Another key advantage is that it prevents overfitting, allowing us to fix training time to 300 epochs and try different setups without the need for early stopping on a validation set.

4.5. Fuzzier Training Examples

Onsets are annotated as time points. For training, we associate each annotation with its closest spectrogram frame and use this frame (along with its ± 7 frames of context) as a positive example, and all others as negative examples. Some onsets have a soft attack, though, or are not annotated with 10 ms precision, resulting in actual onsets being presented to the network as negative training examples. To counter this, we would like to train on less sharply defined ground truth. One solution would be to replace the binary targets with sharp Gaussians and turn the classification problem into a regression one, but preliminary experiments on the RNN showed no improvement. Instead, we define a single frame before and after each annotated onset to be additional positive examples. To still teach the network about the most salient onset position, these examples are weighted with only 25% during training. This measure improves F-score to 89.9%, using a higher detection threshold than before. Simply excluding 1 or 2 frames around each onset from training, letting the network freely decide on those, works just slightly worse.

4.6. Rectified Linear Units

Both the hand-designed SuperFlux algorithm [**?**] and the state-of-the-art RNN [**?**] build on precomputed positive differences in spectral energy over time. Replacing the tanh activation function in the convolutional layers with the linear rectifier $y(x) = \max(0, x)$ provides a direct way for the CNN to learn to compute positive differences in its spectral input, and has been generally shown useful for supervisedly trained networks [**?**]. In our case, it improves F-score to our final result of 90.3%. Using rectified linear units for the fully-connected hidden layer as well reduces performance to 89.6%.

5. INTROSPECTION

While we have developed a state-of-the-art musical onset detector that is perfectly usable as a black box, we would like to know how it works. In particular, we hope to gain some insights on why it is better than existing hand-crafted algorithms, and possibly learn from its solution to improve these algorithms.

For this purpose, we train a CNN with the second convolutional layer and max-pooling layer removed to make it easier to interpret, and tanh units for the remaining convolutional layer (dropout and fuzziness as before). It achieves 88.8% F-score, which is still far superior to the SuperFlux algorithm (Table 1, last row), making it an interesting model to study. We will visualize both the connections learned by the model and its hidden unit states on test data to understand its computations. To guide us, we will start at the output unit and work our way backwards through the network, concentrating on the parts that contribute most to its classification decisions.

5.1. Output Unit

The output unit computes a weighted sum of the 256 hidden unit states below, then applies the logistic sigmoid function, resulting in a value between 0.0 and 1.0 interpretable as an onset probability. Fig. 3b shows this output over time for two well-chosen test signals: One rich in percussive onsets, the other in transient-free harmonic ones.[1] Except for a false positive in the latter, the network output well matches the ground truth.

To understand how the output is driven by the 256 hidden units, we visualize their states for the two signals, ordered by connection weight to the output unit (Fig. 3c). Interestingly, the most strongly connected units (near the top and bottom border) are hardly active and do not seem to be useful for these examples – they may have specialized to exotic corner cases in the training data. In contrast, a large number of units with small connection weights (near the sign

[1] `http://ofai.at/~jan.schlueter/pubs/2014_icassp/`

change prominently visible in the figure) clearly reflects the onset locations. Comparing states for the two signals, we see that a number of positively connected units (below the sign change) detect percussive onsets only, while others also detect harmonic ones.

5.2. Fully-Connected Hidden Layer

Having identified the most interesting hidden units (the ones near the sign change), we will investigate what they compute. Fig. 3d visualizes the connections of two units to the feature maps in the layer below. The second one displays a sharp wide-band off-on-off connection to the fourth map, and similarly sharp connections to other maps. It is good in detecting percussive onsets, which are short wide-band bursts. The first unit computes more long-term differences, notably in the first and ninth map, and manages to capture harmonic onsets. Other units look very similar to the two types shown, with variations in timing and covered frequency bands.

5.3. Convolutional Layer

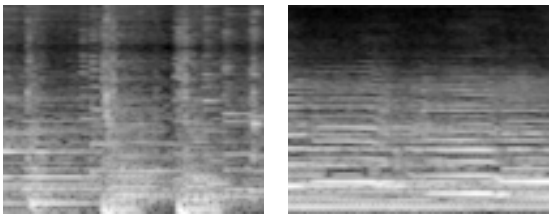
To close the remaining gap to the input, we will study the feature maps computed by the convolutional layer. From the previous investigation, maps 4 and 9 seem to play an important role. For the first signal, map 4 highlights the onsets very sharply (Fig. 3e). Looking at the corresponding filter (Fig. 3g), it seems to detect energy bursts of 1 to 3 frames in the mid-sized spectrogram, and compute a temporal difference in the long-window one. Map 9 also computes this temporal difference and contrasts it against a slightly offset difference in the short-window spectrogram (Fig. 3h). While still very fuzzy, this enhances onsets of the second signal (Fig. 3f).

5.4. Insights

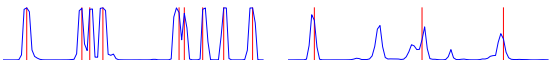
Although our inspection was highly selective, covering a small part of the network only, we formed a basic intuition of what it does. Like spectral flux based methods, the network computes spectral differences over time. In doing so, it adapts the context to the spectrogram window length, which was also found to be crucial in [**?**]. And like [**?**], the CNN separates the detection of percussive and pitched onsets. As a novel feature, the network computes the difference of short- and long-window spectrograms to find onsets. However, imitating this is not enough to build a good onset detector. In fact, the key factor seems to be that the network combines hundreds of minor variations of the same approach, something that cannot be reproduced with hand-designed algorithms.

6. DISCUSSION

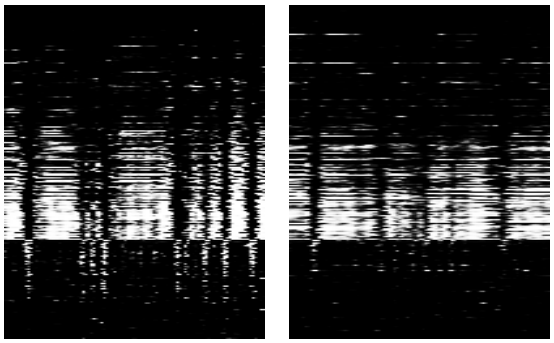
Through a combination of recent neural network training methods, we significantly advanced the state of the art in musical onset detection. Analyzing the learned model, we find that it rediscovered several ideas used in hand-designed methods, but is superior by combining results of many slightly different detectors. This shows that even for easily understandable problems, labelling data and applying machine learning may be more worthwhile than directly engineering a solution. Further improvements may be achieved by training larger networks, by trying other filter shapes, by regularizing the convolutional layers [**?**], and by including phase information. More insights might be won by recent CNN visualization techniques [**?**, **?**]. Another direction for future research is to combine ideas from CNNs with RNNs, such as local connectivity and pooling, to obtain a state-of-the-art model suitable for low-latency real-time processing.



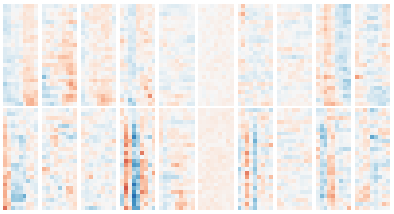
(a) input spectrograms (mid-sized window length only)



(b) network output (blue line) and ground truth (vertical red bars)



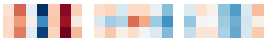
(c) penultimate layer states ordered by connection weight to output, from strongly negative (top) to strongly positive (bottom)



(d) weights of two penultimate layer units: each block shows connections to one of the ten feature maps in the layer below, with time increasing from left to right, frequency from bottom to top, red and blue denoting negative and positive weights, respectively

(e) feature map 4 for the first signal (after pooling and tanh)

(f) feature map 9 for the second signal (after pooling and tanh)



(g) filter kernel for map 4: three 7×3 blocks for the three input spectrograms (mid, short, long)

(h) filter kernel for map 9: three 7×3 blocks for the three input spectrograms (mid, short, long)

Fig. 3: Network weights and states for two test signals (see Sect. 5).

Acknowledgements: This research is supported by the Austrian Science Fund (FWF): TRP 307-N23, and by the European Union Seventh Framework Programme FP7 / 2007-2013 through the PHENICX project (grant agreement no. 601166). The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry for Transport, Innovation, and Technology.