# Convolutional Neural Networks with Batch Normalization for Classifying Hi-hat, Snare, and Bass Percussion Sound Samples

Nicolai Gajhede
Sound and Music Computing Group,
Aalborg University Copenhagen
Copenhagen, Denmark
gajser2001@yahoo.dk

Oliver Beck
Deep AI
Berlin, Germany
oliver.beck@deep-ai.com

Hendrik Purwins
Audio Analysis Lab &
Sound and Music Computing Group,
Aalborg University Copenhagen
Copenhagen, Denmark
hpu@create.aau.dk

## ABSTRACT

After having revolutionized image and speech processing, convolutional neural networks (CNN) are now starting to become more and more successful in music information retrieval as well. We compare four CNN types for classifying a dataset of more than 3000 acoustic and synthesized samples of the most prominent drum set instruments (bass, snare, hi-hat). We use the Mel scale log magnitudes (MLS) as a representation for the input of the CNN. We compare the classification results of 1) a CNN (3 conv/max-pool layers and 2 fully connected layers) without drop-out and batch normalization vs. three variants, 2) with drop-out, 3) with batch normalization (BN), and 4) with both drop-out and BN. The CNNs with BN yield the best classification results (97% accuracy).

## Categories and Subject Descriptors

I.5.1 [**Pattern Recognition**]: Neural Networks

## Keywords

musical instrument classification; convolutional neural networks

## 1. INTRODUCTION

Machine learning and in particular Big Data are becoming increasingly important to enable multisensory human-computer interaction, equipping the computer with the capability to react and adapt to the human in an intelligent way. Deep Convolutional Neural Networks (CNNs) applied to large datasets have been tremenduously successful in classifying and localizing objects in images [16, 28, 26, 12, 29, 27]. How is this approach adapted to audio? Whereas in image recognition, some labeled databases comprise more than a million images, labeled databases in music information retrieval are smaller. In this paper, we explore the impact of key architectural elements of CNNs such as dropout and batch normalization in the context of a new drum sound dataset of 3713 samples, comprising acoustic as well as synthesized sounds of bass, snare and hi-hat.



*AM '16, October 04-06, 2016, Norrköping, Sweden*



DOI: http://dx.doi.org/10.1145/2986416.2986453

### 1.1 Audio Information Retrieval with CNNs

In their overview on Deep Neural Networks (DNNs) for speech recognition, Deng et al. [4] point out a couple of important issues when applying DNNs for audio: They found that DNNs work better on filterbank outputs than on MFCCs (Mel Frequency Cepstral Coefficients) and observed that rectified linear units (ReLUs) learn much faster than logistic neurons. The higher overfitting observed when using ReLUs could be counteracted by a regularization method called "dropout". Deng et al. [5] noted that ReLUs contribute to energy normalization of features.

DNNs are trained using some form of stochastic gradient descent. However, setting the overall network architecture (number and type of layers and their order, number of units per layer) as well as an appropriate learning rate schedule and strength of regulizer requires experience and above all experimentation. In addition, according to Deng et al. [5], it is difficult to automatically learn Mel-like filter weights or delta features in an end-to-end fashion. Deng et al. [4] performed training with different sampling rates chosen as input features of the raw audio data. Therefore the choice of which features of the raw data to use as network input is also one that requires careful consideration. In the following we give an overview of feature pre-processing, network design choices and how network learning could be achieved for state-of-the art sound modeling with DNN:

For musical onset detection, Schlüter and Böck [24] use 80 Mel scale log magnitudes (MLS) for 3 different window sizes. They obtain the MLS by taking 80 weighted averages across the hearing range of the magnitude spectrum, where the weights of the average form a triangle centered at frequencies spaced according to the Mel scale (linearly for lower frequencies and logarithmically for higher frequencies). Then the lograrithm of those 80 values is taken. From the test set they calculate mean and standard deviation for every frequency band and use it to normalize the training set. Schlüter and Böck [24] used rectangular filters, wide in time and narrow in frequency for the convolutional layers, and max-pooling filters wide in frequencies and narrow in time. They compared different architectures, a fully connected network with 2 hidden layers of 256 units and an architecture of two consecutive convolutive/max-pool layers (with 10 and 20 feature maps respectively), ending in fully

connected layers with two different sets of filters ($7 \times 3, 1 \times 3$ vs. $5 \times 5, 1 \times 2$). However they do not find significant performance differences between those architectures which are comparable to the performance observed using state-of-the-art recurrent neural networks.

For audio tag prediction, Dieleman and Schrauwen [6] suggest another approach: *end-to-end learning*. The raw audio sampled at 16 kHz is processed with length 265 and stride 256 convolution, then filtered with two consecutive pairs of layers: 32 filters of length 8, alternating with max-pooling layers of size 4. These lead to two fully connected layers of 100 and 50 units respectively. They use rectified linear units except for the final layer (sigmoids). When comparing spectrogram input with raw audio and traditional features as input it turns out that the end-to-end learning does not yield the best acurracies.

Grill and Schlüter [11] use CNNs for musical boundary detection. They first calculate the 40 MLS, max pool them along the time axis. From this, they apply the Discrete Cosine Transform Type II to arrive at MFCCs, collect MFCCs in a bag of context length $m$. From this, a matrix of cosine distances between time lagged blocks is calculated. They normalize frequency bands to zero mean and unit variance. They use a convolutional layer of 32 kernels across 8 time frames and 6 frequency bands, a max-pooling layer of $3 \times 6$, followed by another convolution layer with 64 $6 \times 3$ kernels, followed by a fully connected layer of 128 units. The networks can be fused at different stages: for the output units, the two hidden dense layers, or the last convolution layer.

For instrument recognition, Li et al. [17] use filters that have the same height as the input, followed by ReLU rectification.. Their network consists of 3 pairs of convolution layers (convolution, max pooling), followed by two fully connected layers, first with ReLu and Dropout, then with sigmoid. Park and Lee [20] use multiresolution recurrence plots (MRP) derived from sample blocks of $2^n$ lengths and combine them with a spectrogram image. In order to compress variable length input sample blocks into vectors that all have the same length, variable subsampling by max pooling is used. Compression of the amplitudes is achieved via the square root function. Sounds are normalized by subtraction of the mean. They consider 8 temporal acquisition points with various offsets from the tone onset for taking the spectrogram images as well as the MRP images, for 7 layers of different resolution. Like the three colour channels, here, analogously, the different channels correspond to the different time resolutions. Also $64 \times 64$ spectrograms are used at all the offsets following the tone onset. The two representations (spectra and MRPs) are combined when reaching the final fully connected layers of a multi-column CNN. They used 10-fold cross validation for evaluation. Then they classified among 3 different instrument families and among 20 instruments, comparing three networks: spectrograms only, MRPs only and a combination.

## 1.2 Other Methods for Percussion Instrument Classification

Often, features used in audio classification are spectral methods, such as mel-frequency spectral coefficients (MFCCs), spectral moments, or band-energy ratio [25]. According to Scholler and Purwins [25], there are several approaches used for drum transcription [30, 31]. Herrera et al. [13] suggest a widely referenced method for classification on isolated drum samples. Different features are calculated for the attack and decay part of the drum signal. Features for the attack part include attack energy, temporal centroid, log-attack time, zero crossing rate. The large majority of features are extracted from the decay part, e.g. spectral flatness, spectral centroid, spectral kurtosis, zero crossing rate, zero crossing rate variance, skewness and relative energy in a number of different frequency subbands. In addition, the mean and variance of 13 MFCCs are calculated for the whole signal. FitzGerald et al. [10] applied prior subspace analysis to drum transcription. Dittmar and Uhle [8] suggested non-negative independent component analysis for the same task. In Paulus and Virtanen [21], an atomic spectral envelope and a temporal envelope of the gain was learnt for each instrument, assuming the statistical independence of the spectral and the temporal envelope. On a database of isolated instrument sounds, these atoms were learnt by non-negative matrix factorization (NMF), a method that has been dominating the field of drum transcription since. Paulus and Virtanen [21] introduced a procedure to iteratively learn the coefficients for a linear decomposition of the sound spectra for each frame, in terms of a dictionary of atoms of spectral shapes. Then, high deltas between consecutive coefficients of the same instrument spectrum indicate an onset of this instrument. Benetos et al. [2] compare local, sparse, and discriminant NMF for instrument classification. As features, they used zero-crossing rate, spectral flux, spectral roll-off, MFCCs, spectral centroid, spectral envelope, spectral spread, spectral flatness, and spectral projection coefficients. Applying a nearest neighbor classifier to the weight vectors of the NMF output, employing the cosine similarity matrix yields then the right instrument class. Lindsay-Smith et al. [18] introduced an improved gains update method to NMF. Battenberg et al. [1] computed spectral templates using agglomerative clustering and a Gamma mixture model. Dittmar and Gärtner [7] further modified the NMF update rule to enable real-time drum transcription. Röbel et al. [23] presented an NMF-based approach that models the relative level of background noise through regularization. Scholler and Purwins [25] presented a biologically inspired three-step process for audio classification: (1) Efficient atomic functions are learned in an unsupervised manner on mixtures of percussion sounds (drum phrases), optimizing the length as well as the shape of the atoms. (2) An analogue spike model is used to sparsely approximate percussion sound signals (bass drum, snare drum, hi-hat). The spike model consists of temporally shifted versions of the learned atomic functions, each having a precise temporal position and amplitude. To obtain the decomposition given a set of atomic functions, Matching Pursuit is used. [19] (3) Features are extracted from the resulting spike representation of the signal. Then a support vector machine is used for classification.

Compared to similar work [20, 17], our main contribution is the application of batch normalziation (BN) to a larger and more diverse dataset (more instances per class).

# 2. METHODS

## 2.1 Data

The dataset consists of 3713 audio clips altogether (1320 bass, 1124 hi-hat, 1269 snare) of approximately 1 second duration each. These sounds have been collected from a variety of commercial and other sound libraries.

## 2.2 Data Preprocessing

Similar to Schlüter and Böck [24], our input representation is a matrix of Mel scale log magnitudes (MLS). Starting from the beginning of the sample (the sound event onset), we analyze frames of 512 samples with a hop size corresponding to 10 ms. After applying an FFT (size 1024) the magnitude spectrum is compressed into 80 values using triangular weighted frequency band averages following the Mel frequency spacing from 28 Hz to 16 kHz. Then the logarithm of these values is taken, yielding a $80 \times 8$ MLS matrix (8 temporal offsets after the onset and 80 frequency bands). All

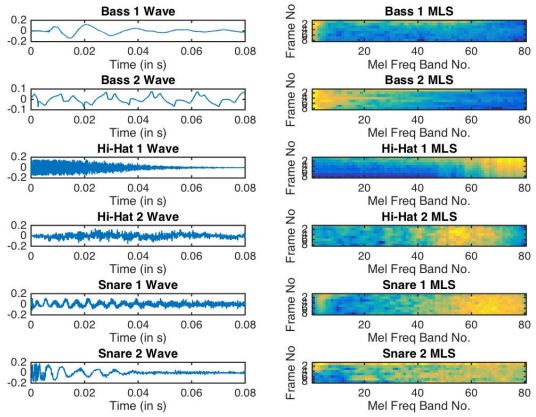


**Figure 1: Waveforms (first 0.08 s after onset) and $8 \times 80$Mel log frequency magnitude spectrum (MLS) matrices of 2 sounds of each of the drum classes are shown. The great variety of (acoustically and synthetically generated) sounds becomes apparent.**

values are normalized so that they scale between 0 and 1. Dropout as a means to combat overfitting has been introduced by Hinton et al. [14]. In the dropout method at each iteration neurons of hidden layers are randomly switched off with a certain predetermined probability (often set to 50%). This methods prevents single neurons to learn spurious dependencies and enables them to generalize better over 'the combinatorially large variety of internal contexts in which they must operate'.[14, 22] Dropout has been named one of the key ingredients to successfully learn supervised deep convolutional neural networks by the seminal study of Krizhevsky et al. [16]. Since then other methods such as batch normalization have partially replaced dropout as a means to combat network overfitting [15]. Nevertheless current state-of-the-art architectures for ImageNet classification [27] use dropout in their architectures before the fully connected layer for improved network generalization.

## 2.3 Batch Normalization

Ioffe and Szegedy [15] define the internal covariate shift 'as the change in the distribution of network activations due to the change in network parameters during training.' To reduce the internal covariate shift, Ioffe and Szegedy [15] propose to normalize the distribution of the inputs of each layer to have zero means and unite variance for each mini-batch. Batch normalization is then applied immediately before each non-linearity. For details of the approach we refer the reader to Ioffe and Szegedy [15]. As a consequence of introducing batch normalization to a DNN, they find that importantly learning rate could increased significantly without observing any ill-effects from vanishing or exploding gradients. Also the learning rate schedule could be substantially increased and weight regularization be reduced. Lastly they find that the need for dropout regularization is strongly diminished in networks employing batch normalization. In summary, since the introduction of batch normalization many of the most successfull DNNs, including the winning architecture of the ILSVRC 2015 challenge [12], are using batch normalization. For alternative approaches using different network transfer function to achieve zero-mean activations and some of the advantages of batch normalization see e.g. Clevert et al. [3].

## 2.4 Mxnet

We use Mxnet, a framework for building deep neural networks with the use of GPUs for high parallelization.

## 2.5 Deep Neural Network (DNN) with Variants

After numerous initial experiments of different settings we have chosen to use the following setup for our architecture.

Our basic DNN architecture (Figure 2) consists of 3 convolutional layers and two fully-connected layers. From the input layer 16 feature maps are generated through the use of 2x4 (frequency bands $\times$ time) kernels using ReLU activation functions for the weights, followed by a max-pooling layer of 3x1 and stride 1x1 using 'max' method in the pooling. The second layer is constructed of 32 feature maps generated by 3x3 kernels using ReLUs, followed by max-pooling of 3x1 and stride 1x1. The third and final convolutional layer consists of 64 feature maps generated by 3x3 kernels using ReLU, max-pooling of 3x1, and stride 1x1.

The first fully-connected layer consist of a flattening technique, 256 hidden layer units and sigmoid activations functions. The last fully-connected layers has 3 output units, corresponding to the three classes using a softmax function for the output.

## 2.6 DNN Variants

- We have placed the dropout technique in the end of the first fully-connected layer, before the final fully-connected layer.
- The batch normalization (BN) version of the network applied batch normalization to all convolutional layers.
- The dropout & BN version integrates both methods.

# 3. RESULTS

We compare the performance of the DNN variants with k-nearest neighbours (KNN) and random forests classification, both implemented in the Matlab PRTools 5 Toolbox [9] and used with default parameters.

The data set is randomly split into a training set of 2970 samples (80%) and a test set of 743 samples (20%). In Table 1, the test set accuracy of the 4 variants is reported. Variants that use BN perform best in this experiment.

**Accuracies of DNN Variants**

| *Variant* | *Accuracy* |
|---|---|
| DNN | 0.95 |
| DNN w. Dropout | 0.96 |
| DNN w. BN | 0.97 |
| DNN w. Dropout & BN | 0.97 |
| KNN | 0.96 |
| Random Forests | 0.91 |

**Table 1: Test set accuracies of the different DNN with and without dropout and batch normalization (BN) after** 300 **epochs using a learning rate of** 0.005**. As a baseline comparison, the results for k-nearest neighbors and random forests classification are given as well.**

Both the dropout and the batch normalization technique give a classification improvement when classifying drum sounds (hi-hat, bass, snare). All DNN variants perform better than random forest classification. DNN with batch normalization performs better than KNN.

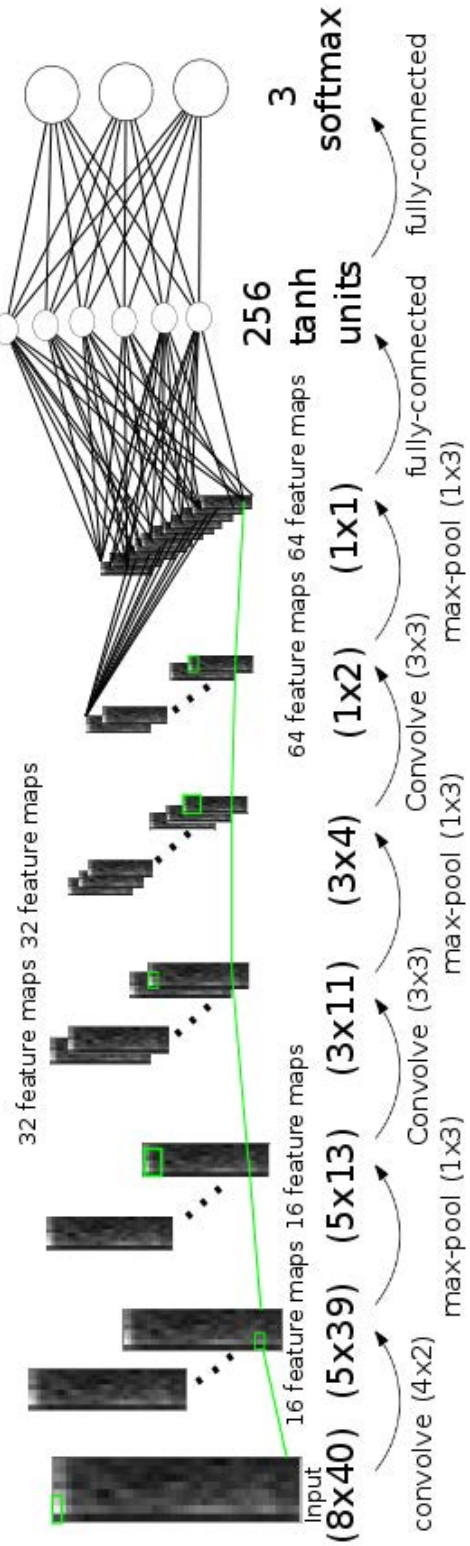


**Figure 2: DNN architecture.**

BN and Dropout & BN roughly double the computation time compared to the other variants. In addition, BN appears to have little sensitivity wrt. the learning rate.

## 4. CONCLUSION

We showed that for our data set, the use of batch normalization slightly outperforms the other deep neural network variants and other classification methods (KNN, random forests). As future work it will be interesting to analyze percussion sounds in context, e.g. in a drum pattern or a drum channel or a mixture of different instruments. For context-sensitive analysis, Deng et al. [4, 5] suggest latent semantic analysis representation from the preceeding text in Penn Treebank data. Improvement in classification opens future perspective of a more intelligent human-like multisensory human-computer interaction.

## 5. ACKNOWLEDGMENTS

We are grateful to Felix Tutzer for compiling and providing the drum sound sample dataset.

## 6. REFERENCES

[1] Eric Battenberg, Victor Huang, and David Wessel. 2012. Toward live drum separation using probabilistic spectral clustering based on the itakura-saito divergence. In *Audio Engineering Society Conference: 45th International Conference: Applications of Time-Frequency Processing in Audio*. Audio Engineering Society.

[2] Emmanouil Benetos, Margarita Kotti, and Constantine Kotropoulos. 2006. Musical instrument classification using non-negative matrix factorization algorithms. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*. IEEE, 4–pp.

[3] Djork-Arne Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *arXiv preprint arXiv:1511.07289* (2015).

[4] Li Deng, Geoffrey Hinton, and Brian Kingsbury. 2013a. New types of deep neural network learning for speech recognition and related applications: An overview. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 8599–8603.

[5] Li Deng, Jinyu Li, Jui-Ting Huang, Kaisheng Yao, Dong Yu, Frank Seide, Mike Seltzer, Geoffrey Zweig, Xiaodong He, Julia Williams, and others. 2013b. Recent advances in deep learning for speech research at Microsoft. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 8604–8608.

[6] Sander Dieleman and Benjamin Schrauwen. 2014. End-to-end learning for music audio. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 6964–6968.

[7] Christian Dittmar and Daniel Gärtner. 2014. Real-Time Transcription and Separation of Drum Recordings Based on NMF Decomposition.. In *DAFx*. 187–194.

[8] Christian Dittmar and Christian Uhle. 2004. Further steps towards drum transcription of polyphonic music. In *Audio Engineering Society Convention 116*. Audio Engineering Society.

[9] R Duin, P Juszczak, P Paclik, E Pekalska, D de Ridder, D Tax, and S Verzakov. 2010. PRTools 4.1. *A Matlab Toolbox for Pattern Recognition, Software and Documentation downloaded May* (2010).

[10] Derry FitzGerald, Robert Lawlor, and Eugene Coyle. 2003. Prior subspace analysis for drum transcription. In *Audio Engineering Society Convention 114*. Audio Engineering Society.

[11] Thomas Grill and Jan Schlüter. 2015. Music boundary detection using neural networks on spectrograms and self-similarity lag matrices. In *Signal Processing Conference (EUSIPCO), 2015 23rd European*. IEEE, 1296–1300.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *CoRR* abs/1512.03385 (2015). http://arxiv.org/abs/1512.03385

[13] P. Herrera, A. Yeterian, and F. Gouyon. 2002. Automatic classification of drum sounds: a comparison of feature selection methods and classification techniques. *Music and Artificial Intelligence* (2002), 69–80.

[14] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR* abs/1207.0580 (2012). http://arxiv.org/abs/1207.0580

[15] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[17] Peter Li, Jiyuan Qian, and Tian Wang. 2015. Automatic Instrument Recognition in Polyphonic Music Using Convolutional Neural Networks. *arXiv preprint arXiv:1511.05520* (2015).

[18] Henry Lindsay-Smith, Skot McDonald, and Mark Sandler. 2012. Drumkit transcription via convolutive NMF. In *International Conference on Digital Audio Effects (DAFx), York, UK*.

[19] S.G. Mallat and Z. Zhang. 1993. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing* 41, 12 (1993), 3397–3415.

[20] Taejin Park and Taejin Lee. 2015. Musical instrument sound classification with deep convolutional neural network using feature fusion approach. *arXiv preprint arXiv:1512.07370* (2015).

[21] J. Paulus and T. Virtanen. 2005. Drum transcription with non-negative spectrogram factorisation. In *Proc. EUSIPCO*. 4–8.

[22] Karol J Piczak. 2015. Environmental sound classification with convolutional neural networks. In *Machine Learning for Signal Processing (MLSP), 2015 IEEE 25th International Workshop on*. IEEE, 1–6.

[23] Axel Roebel, Jordi Pons, Marco Liuni, and Mathieu Lagrange. 2015. On automatic drum transcription using non-negative matrix deconvolution and itakura saito divergence. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 414–418.

[24] Jan Schlüter and Sebastian Böck. 2013. Musical onset detection with convolutional neural networks. *6th International Workshop on Machine Learning and Music (MML), Prague, Czech Republic* (2013).

[25] Simon Scholler and H. Purwins. 2011. Sparse Approximations for Drum Sound Classification. *IEEE Journal of Selected Topics in Signal Processing* 5 (2011), 933 – 940.

[26] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR* abs/1409.1556 (2014). http://arxiv.org/abs/1409.1556

[27] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. 2016. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261* (2016).

[28] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2014. Going Deeper with Convolutions. *CoRR* abs/1409.4842 (2014). http://arxiv.org/abs/1409.4842

[29] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the Inception Architecture for Computer Vision. *CoRR* abs/1512.00567 (2015). http://arxiv.org/abs/1512.00567

[30] K. Tanghe, S. Degroeve, and B. De Baets. 2005. An algorithm for detecting and labeling drum events in polyphonic music. *Proc. MIREX, London, UK* (2005).

[31] K. Yoshii, M. Goto, and H.G. Okuno. 2006. Drum sound recognition for polyphonic audio signals by adaptation and matching of spectrogram templates with harmonic structure suppression. *IEEE Transactions on Audio, Speech, and Language Processing* 15, 1 (2006), 333–345.