RANDOMLY WEIGHTED CNNs FOR (MUSIC) AUDIO CLASSIFICATION

Jordi Pons and Xavier Serra

Music Technology Group, Universitat Pompeu Fabra, Barcelona

ABSTRACT

The computer vision literature shows that randomly weighted neural networks perform reasonably as feature extractors. Following this idea, we study how non-trained (randomly weighted) convolutional neural networks perform as feature extractors for (music) audio classification tasks. We use features extracted from the embeddings of deep architectures as input to a classifier – with the goal to compare classification accuracies when using different randomly weighted architectures. By following this methodology, we run a comprehensive evaluation of the current architectures for audio classification, and provide evidence that the architectures alone are an important piece for resolving (music) audio problems using deep neural networks.

Index Terms— *random, neural networks, audio, ELM, SVM.*

1. INTRODUCTION

Previous computer vision works show that the classification performance delivered by random CNN features is correlated with the results of their end-to-end trained counterparts [1–3]. Thus meaning, in practice, that one can bypass the time-consuming process of learning to evaluate a given architecture [1]. We build on top of this result to evaluate current CNN architectures for audio classification: we use the CNNs' embeddings to construct feature vectors for a classifier, with the goal to compare classification performances when different non-trained CNN architectures are used to extract features. To the best of our knowledge, this is the first comprehensive evaluation of randomly weighted CNNs for (music) audio classification [4–6].

Additionally, extreme learning machines (ELMs) [7–9] are also closely related to our work. In short, ELMs are classification/regression models[1] that are based on a single-layer feed-forward neural network with random weights. They work as follows: first, ELMs randomly project the input into a latent space; and then, learn how to predict the output via a least-square fit. More formally, we aim to predict:

$$\hat{Y} = W_2 \; \sigma(W_1 X),$$

where W_1 is the (randomly weighted) matrix of input-to-hidden-layer weights, σ is the non-linearity, W_2 is the matrix of hidden-to-output-layer weights, and X represents the input. Training proceeds as follows: *1)* set W_1 with random values; *2)* estimate W_2 via a least-squares fit:

$$W_2 = \sigma(W_1 X)^{+} Y$$

where $^{+}$ denotes the Moore-Penrose inverse. Since no iterative process is required for learning the weights, training is faster than stochastic gradient descent [9]. Provided that we process audio signals with randomly weighted CNNs, ELM-based classifiers are a natural choice for our study – so that all the pipeline (except the last layer) is based on random projections that are only constrained by the structure of the neural network.

In this work we evaluate the most used deep architectures for (music) audio classification. In order to facilitate the discussion around these architectures, we divide the deep learning pipeline into two parts: front-end and back-end. The front-end is the part that interacts with the input signal in order to map it into a latent-space, and the back-end predicts the output given the representation obtained by the front-end. Note that one can interpret the front-end as a "feature extractor" and the back-end as a "classifier". Given that we compare how several non-trained (random) CNNs perform as feature extractors and we will use out-of-the-box classifiers to predict the classes, this literature review focuses in introducing the main deep learning front-ends for audio classification. Front-ends are generally conformed by CNNs [10–14], since these can encode efficient representations by sharing weights along the signal. Figure 1 depicts the main CNN front-end paradigms, which can be divided into two groups depending on the used input signal: waveforms [10, 12, 15] or spectrograms [11, 13, 14]. Further, the design of the filters can be either based on domain knowledge or not. For example, one leverages domain knowledge when the *frame-level single-shape*[2] front-end for waveforms is designed so that the length of the filter is set to be the same as the window length in a STFT [10]. Or for a spectrogram front-end, it is used *vertical* filters to learn spectral representations [16], *or horizontal* filters to learn longer temporal cues [17]. Generally, a single filter shape is used in the first CNN layer [10, 11, 16, 17], but some recent work reported performance gains when using several filter shapes in the first layer [12–14, 18]. Using many filters promotes a more rich feature extraction in the first layer, and facilitates leveraging domain knowledge for designing the filters' shape. For example: a *frame-level many-shapes* front-end for waveforms can be motivated from a multi-resolution time-frequency transform[3] perspective – with window sizes varying inversely with frequency [12]; or, since it is known that some patterns in spectrograms are occurring at different time-frequency scales, one can intuitively incorporate *many vertical and/or horizontal* filters to efficiently capture those in a spectrogram front-end [13, 14, 18]. On the other hand, when domain knowledge is not used, it is common to employ a deep stack of small filters, e.g.: 3×1 in the *sample-level* front-end used for waveforms [15], or 3×3 in the *small-rectangular filters* front-end used for spectrograms [11]. These models make minimal assumptions over the local stationarities of the signal, so that any structure can be learnt via hierarchically combining small-context representations.

2. METHOD

Our goal is to study which CNN front-ends work best via evaluating how non-trained models perform as feature extractors. Our evaluation is based on the traditional pipeline of *features extraction + classifier*. We use the embeddings of non-trained (random) CNNs as features: for every audio clip, we compute the average of each feature map (in every layer) and concatenate these values to construct a feature vector [4]. The baseline feature vector is constructed from 20 MFCCs, their Δs and $\Delta\Delta$s. We compute their mean and standard deviation through time, and the resulting feature vector is of size 120. We set the widely used MFCCs + SVM setup as baseline. To allow

[1] Support Vector Machines are also classification/regression models.

[2] Italicized names correspond to the front-end types in Figure 1.

[3] The Constant-Q Transform [19] is an example of such transform.

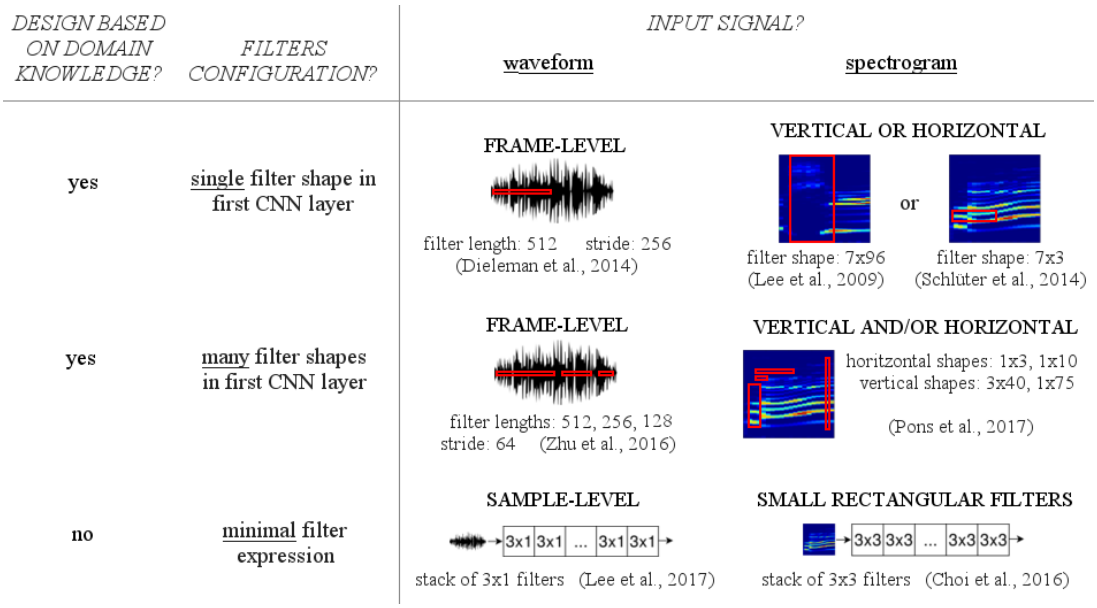

Fig. 1. CNN front-ends for audio classification tasks – with examples of possible configurations for every design paradigm.

a fair comparison with the baseline, CNN models have ≈ 120 feature maps – so that the resulting feature vectors have a similar size as the MFCCs vector. Further, we evaluate an alternative configuration with more feature maps (≈3500) to show the potential of this approach. Model's description omit the number of filters per layer for simplicity – full implementation details are accessible online.[4]

2.1. Features: randomly weighted CNNs

Except for the *VGG* model that uses ELUs as non-linearities [11], the rest use ReLUs [20] – and we do not use batch normalization. We use waveforms and spectrograms as input to our CNNs:

Waveform inputs — are of ≈ 29sec (350,000 samples at 12kHz):

· *Sample-level:* is based on a stack of 7 blocks that are composed by a 1D-CNN layer (filter length: 3, stride: 1), and a max-pool layer (size: 3, stride: 3) – with the exception of the input block which has no max-pooling and its 1D-CNN has a stride of 3 [15]. Averages to construct the feature vector are computed after every pooling layer, except for the first layer that are computed after the CNN.

· *Frame-level many-shapes:* consists of a 1D-CNN layer with 5 filter lengths: 512, 256, 128, 64, 32 [12]. Note that out of this single 1D-CNN layer, 5 feature maps (resulting of the different filter length convolutions) are concatenated. Due to that, every feature map needs to have the same (temporal) size. For that reason, every filter's stride is of 32 and we use *same* padding to easily concatenate the feature maps. Since this model is single-layered and it might be in disadvantage, we increase its depth via adding 3 more 1D-CNN layers (length: 7, stride: 1) – where the last 2 layers have residual connections, and the penultimate layer's feature map is down-sampled by two (MP x2), see Figure 2. Averages to construct the feature vector are calculated for each feature map after every CNN layer.

· *Frame-level:* consists of a 1D-CNN layer with a filter of length 512 [10]. Stride is set to be 32 to allow a fair comparison with the *frame-level many-shapes* architecture. As in *frame-level many-shapes*, we increase the depth of the model via adding three more 1D-CNN layers – as in Figure 2. Averages to construct the feature vector are calculated for each feature map after every 1D-CNN layer.

[4]Reproduce our study: `github.com/jordipons/elmarc`

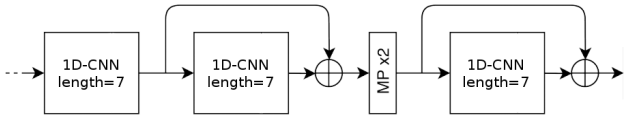

Fig. 2. Additional layers for the *frame-level* & *frame-level many-shapes* architectures, similar to [10,21] (MP stands for max pooling).

Spectrogram inputs — are set to be log-mel spectrograms (spectrograms size: 1376×96[5], being ≈ 29sec of signal). Differently from waveform models, spectrogram models use no additional layers to deepen single-layered CNNs because these already deliver a good performance – and spectrogram-based CNN layers have a stride of 1. As for the *frame-level many-shapes* model, we use *same* padding when many filter shapes are used in the same layer:

· *7×96*: consists of a single 1D-CNN layer with filters of length 7 that convolve through the time axis [10]. As a result: CNN filters are vertical and of shape 7×96. Therefore, these filters can encode spectral (timbral) representations. Averages to construct the feature vector are calculated for each feature map after the 1D-CNN layer.

· *7×86*: consists of a single 2D-CNN layer with vertical filters of shape 7×86 [14,18]. Since its vertical shape is smaller than the input (86<96), filters can also convolve through the frequency axis – what can be seen as "pitch shifting" the filter. Consequently, 7×86 filters can encode pitch-invariant timbral representations [14, 18]. Further, since the resulting activations can carry pitch-related information, we max-pool the frequency axis to get pitch-invariant features (max-pool shape: 1×11). Averages to construct the feature vector are calculated for each feature map after the max-pool layer.

· *VGG*: is based on a stack of 5 blocks combining 2D-CNN layers (with small rectangular filters of 3×3) and max-pooling (of shapes: 4×2, 4×3, 5×2, 4×2, 4×4, respectively) [11]. Averages to construct the feature vector are computed after every pooling layer.

· *Timbral*: consists of a single 2D-CNN layer with many vertical filters of shapes: 7×86, 3×86, 1×86, 7×38, 3×38, 1×38, see Figure 3 (top) [14, 22, 23]. These filters can also convolve through the frequency axis and, therefore, these can encode pitch-invariant representations. Several filter shapes are used to efficiently cap-

[5]STFT parameters: *window_size = 512*, *hop_size=256*, and *fs=12kHz*.

ture different timbrically relevant time-frequency patterns. Further, since the resulting activations can carry pitch-related information, we max-pool the frequency axis to get pitch-invariant features (max-pool shapes: 1×11 or 1×59). Averages to construct the feature vector are calculated for each feature map after the max-pool layer.

· *Temporal*: several 1D-CNN filters (of lengths: 165, 128, 64, 32) operate over an energy envelope obtained via mean-pooling the frequency-axis of the spectrogram, see Figure 3 (bottom). By computing the energy envelope in that way, we are considering high and low frequencies together while minimizing the computations of the model. Observe that this single-layered 1D-CNN is not operating over a 2D spectrogram, but over a 1D energy envelope – therefore no vertical convolutions are performed, only 1D (temporal) convolutions are computed. Averages to construct the feature vector are calculated for each feature map after the CNN layer.

· *Timbral+temporal*: combines both *timbral* and *temporal* CNNs in a single (but wide) layer, see Figure 3 [21]. Averages to construct the feature vector are calculated in the same way as for *timbral* and *temporal* architectures.

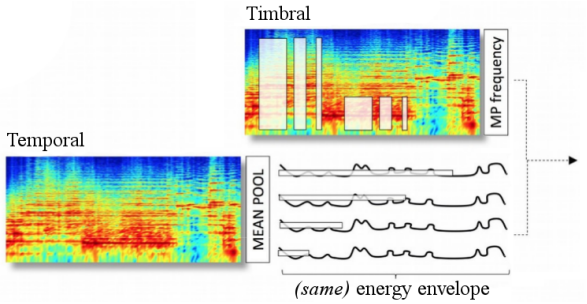

Fig. 3. *Timbral+temporal* architecture (MP stands for max-pool).

As seen, the studied architectures are representative of the audio classification state-of-the-art. For further details about the models under study: the code is accessible online[4], and a graphical conceptualization of the models is available in Figures 1, 2 and 3.

2.2. Classifiers: SVM and ELM

We study how several feature vectors (computed considering different CNNs) perform for a given set of classifiers: SVMs and ELMs. We discarded the use of other classifiers since their performance was not competitive when compared to those. SVMs and ELMs are hyper-parameter sensitive, for that reason we perform a grid search:

· *SVM* hyper-parameters: we consider both *linear* and *rbf* kernels. For the *rbf* kernel, we set γ to: 2^{-3}, 2^{-5}, 2^{-7}, 2^{-9}, 2^{-11}, 2^{-13}, $\#features^{-1}$; and for every kernel configuration, we try several C's (penalty parameter): 0.1, 2, 8, 32.

· *ELM*'s[6] main hyper-parameter is the number of hidden units: 100, 250, 500, 1200, 1800, 2500. We use ReLUs as non-linearity.

2.3. Datasets: music and acoustic events

· *GTZAN* fault-filtered version [24, 25]. Training songs: 443, validation songs: 197, and test songs: 290 – divided in 10 classes. We use this dataset to study the task of music genre classification.

· *Extended Ballroom* [26,27] – 4,180 songs divided in 13 classes. 10 stratified folds are randomly generated for cross-validation. This dataset is to study how these models classify rhythm/tempo classes.

· *Urban Sound 8K* [28] – 8732 acoustic events divided in 10 classes. 10 folds were already defined for cross-validation. Since urban sounds are $<$ 4sec and our models accepts $\approx$ 29sec inputs, the signal is repeated to create inputs of the same length. This dataset is to study how these models classify natural (non-music) sounds.

[6]ELM's implementation: `github.com/zygmuntz/Python-ELM`

3. RESULTS

Figures show average accuracies across 3 runs for every feature type, listed on the right with the length of the feature vector. A *t-test* reveals which models are performing the best (H_0: averages are equal).

3.1. GTZAN: music genre recognition

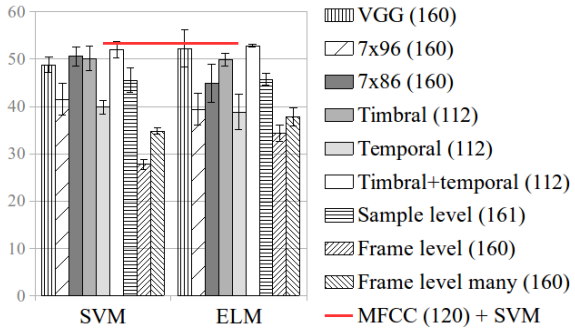

Fig. 4. Accuracy (%) results for the GTZAN dataset with random CNN feature vectors of length $\approx$ 120.

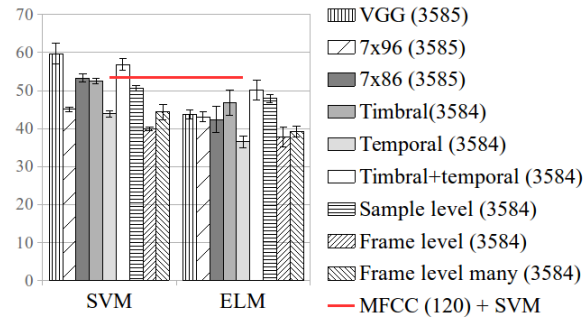

Fig. 5. Accuracy (%) results for the GTZAN dataset with random CNN feature vectors of length $\approx$ 3500.

The *sample-level* waveform model always performs better than *frame-level many-shapes* (t-test: p-value$\ll$0.05). The two best performing spectrogram-based models are: *timbral+temporal* and *VGG* – with a remarkable performance of the *timbral* model alone. The *timbral+temporal* CNN performs better than *VGG* when using the ELM ($\approx$3500) classifier (t-test: p-value=0.017); but in other cases, both models perform equivalently (t-test: p-value$>$0.05). Moreover, the *7x86* model performs better than *7x96* when using SVMs (t-test: p-value$<$0.05), but when using ELMs: *7x96* and *7x86* perform equivalently (t-test: p-value$\gg$0.05). The best *VGG* and *timbral+temporal* models achieve the following (average) accuracies: 59.65% and 56.89%, respectively – both with an SVM ($\approx$3500) classifier. Both models outperform the MFCCs baseline: 53.44% (t-test: p-value$<$0.05), but these random CNNs perform worse than a CNN pre-trained with the Million Song Dataset: 82.1% [15]. Finally, note that although *timbral* and *timbral+temporal* models are single-layered, these are able to achieve remarkable performances – showing that single-layered spectrogram front-ends (attending to musically relevant contexts) can do a reasonable job without paying the cost of going deep [14, 18]. Thus meaning, e.g., that the saved capacity can now be employed to learn additional representations.

3.2. Extended Ballroom: rhythm/tempo classification

The *sample-level* waveform model always performs better than *frame-level many-shapes* (t-test: p-value$\ll$0.05). The two best performing spectrogram-based models are: *temporal* and *timbral+temporal*, but the *temporal* model performs better than *timbral+temporal* in all cases (t-test: p-value$\ll$0.05) – denoting that

spectral cues can be a confounding factor for this dataset. Moreover, the *7x86* model performs better than *7x96* in all cases (t-test: p-value$<$0.05). The best (average) accuracy score is obtained using *temporal* models and SVMs ($\approx$3500): 89.82%. Note that the *temporal* model clearly outperforms the MFCCs baseline: 63.25% (t-test: p-value$\ll$0.05) and, interestingly, it performs slightly worse than a trained CNN: 93.7% [29].

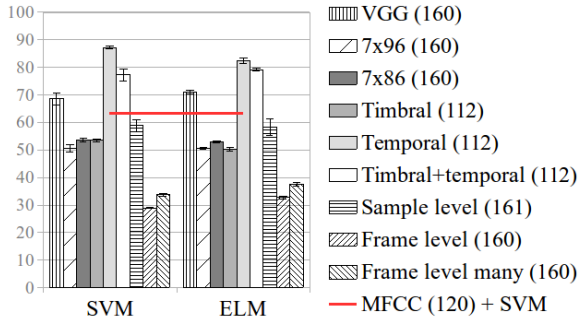

Fig. 6. Accuracy (%) results for the Extended Ballroom dataset with random CNN feature vectors of length $\approx$ 120.

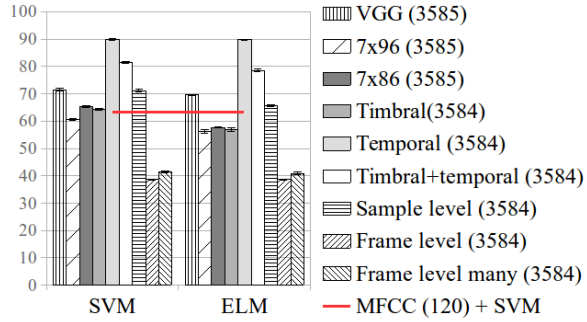

Fig. 7. Accuracy (%) results for the Extended Ballroom dataset with random CNN feature vectors of length $\approx$ 3500

This result confirms that the architectures (alone) introduce a strong prior which can significantly affect the performance of an audio model. Thus meaning that, besides learning, designing effective architectures might be key for resolving (music) audio tasks with deep learning. In line with that, note that the *temporal* architecture is designed considering musical domain knowledge – in this case: how tempo & rhythm are expressed in spectrograms. Hence, its good performance validates the design strategy of using musically motivated architectures as an intuitive way to navigate through the network parameters space [13, 14, 18].

3.3. Urban Sounds 8K: acoustic event detection

For these experiments we do not use the *temporal* model (with 1D-CNNs of length 165, 128, 64, 32). Instead, we study the *timbral+time* model – where *time* follows the same design as *temporal* but with filters of length: 64, 32, 16, 8. This change is motivated by the fact that temporal cues in (natural) sounds are shorter than temporal cues in music.

The *sample-level* waveform model always performs better than *frame-level many-shapes* (t-test: p-value$\ll$0.05). The two best performing spectrogram-based models are: *VGG* and *timbral+time* – but *VGG* performs better than *timbral+time* in all cases (t-test: p-value$\ll$0.05). Also, the *7x86* model performs better than *7x96* in all cases (t-test: p-value$<$0.075). The best (average) accuracy score is obtained using *VGG* and SVMs ($\approx$3500): 70.74% – outperforming the MFCCs baseline: 65.49% (t-test: p-value$<$0.05), and performing slightly worse than a trained CNN: 73% (without data augmentation [30]). Finally, note that *VGG*s achieved remarkable results when recognizing genres and detecting acoustic events – tasks where timbre is an important cue. Therefore, one could argue that *VGG*s are good at representing spectral features.

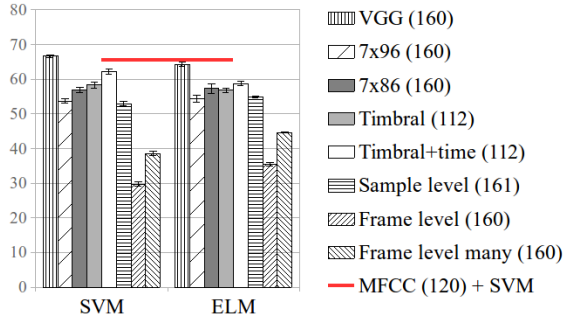

Fig. 8. Accuracy (%) results for the Urban Sounds 8k dataset with random CNN feature vectors of length $\approx$ 120.

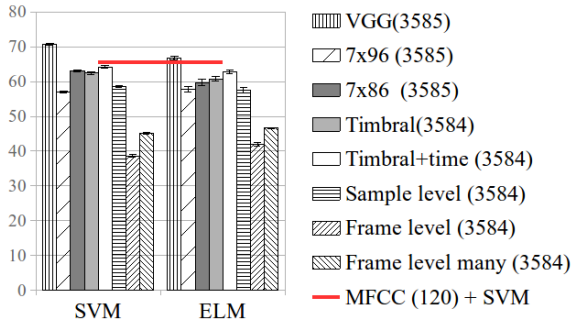

Fig. 9. Accuracy (%) results for the Urban Sounds 8k dataset with random CNN feature vectors of length $\approx$ 3500

4. CONCLUSIONS

This study proposes a cheap way to evaluate CNN architectures via comparing the obtained classification accuracies when using different randomly weighted CNN architectures as feature extractors. The results we obtain are far from *random*, since: *(i)* randomly weighted CNNs are, in some cases, close to match the accuracies obtained by trained CNNs; and *(ii)* these are able to outperform MFCCs. This result denotes that the architectures alone are an important piece of the deep learning solution and therefore, searching for efficient architectures capable to encode the specificities of (music) audio signals might help advancing the state of our field. In line with that, we have shown that (musical) priors embedded in the structure of the model can facilitate capturing useful (temporal) cues for classifying rhythm/tempo classes. Besides, we show that for waveform front-ends: *sample-level* $\gg$ *frame-level many-shapes* $>$ *frame-level*, as noted in the (trained) literature [12, 15]. Further, we show that for spectrogram front-ends: *7x96*$<$*7x86*, as shown in previous (trained) works [18,31] – possibly because via allowing the filters to convolve through the frequency axis the model captures pitch-invariant timbral representations. Finally: *timbral* (+*temporal/time*) and *VGG* spectrogram front-ends achieve remarkable results for tasks where timbre is important – as previously noted in the (trained) literature [14]. Although our main conclusions are backed by external results reported in the (trained) literature, we leave for future work consolidating those via doing a similar study considering trained models.

5. ACKNOWLEDGMENTS

This work is supported by the Maria de Maeztu Programme (MDM-2015-0502), and we are grateful for the GPUs donated by NVidia.

6. REFERENCES

[1] Andrew M Saxe, Pang Wei Koh, Zhenghao Chen, Maneesh Bhand, Bipin Suresh, and Andrew Y Ng, "On random weights and unsupervised feature learning.," in *International Conference on Machine Learning (ICML)*, 2011, pp. 1089–1096.

[2] Amir Rosenfeld and John K Tsotsos, "Intriguing properties of randomly weighted networks: Generalizing while learning next to nothing," *arXiv preprint arXiv:1802.00844*, 2018.

[3] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky, "Deep image prior," *arXiv preprint arXiv:1711.10925*, 2017.

[4] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho, "Transfer learning for music classification and regression tasks," *International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[5] Jaehun Kim, Julián Urbano, Cynthia Liem, and Alan Hanjalic, "One deep music representation to rule them all?: A comparative analysis of different representation learning strategies," *arXiv preprint arXiv:1802.04051*, 2018.

[6] Relja Arandjelovic and Andrew Zisserman, "Look, listen and learn," in *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 609–617.

[7] Wouter F Schmidt, Martin A Kraaijveld, and Robert PW Duin, "Feedforward neural networks with random weights," in *International Conference on Pattern Recognition*. IEEE, 1992, pp. 1–4.

[8] Yoh-Han Pao, Gwang-Hoon Park, and Dejan J Sobajic, "Learning and generalization characteristics of the random vector functional-link net," *Neurocomputing*, vol. 6, no. 2, pp. 163–180, 1994.

[9] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[10] Sander Dieleman and Benjamin Schrauwen, "End-to-end learning for music audio," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.

[11] Keunwoo Choi, George Fazekas, and Mark Sandler, "Automatic tagging using deep convolutional neural networks," *International Society for Music Information Retrieval Conference (ISMIR)*, 2016.

[12] Zhenyao Zhu, Jesse H Engel, and Awni Hannun, "Learning multiscale features directly from waveforms," *arXiv preprint arXiv:1603.09509*, 2016.

[13] Jordi Pons and Xavier Serra, "Designing efficient architectures for modeling temporal features with convolutional neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[14] Jordi Pons, Olga Slizovskaia, Rong Gong, Emilia Gómez, and Xavier Serra, "Timbre analysis of music audio signals with convolutional neural networks," *European Signal Processing Conference (EUSIPCO)*, 2017.

[15] Jongpil Lee, Jiyoung Park, Keunhyoung Luke Kim, and Juhan Nam, "Samplecnn: End-to-end deep convolutional neural networks using very small filters for music classification," *Applied Sciences*, vol. 8, no. 1, pp. 150, 2018.

[16] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2009.

[17] Jan Schluter and Sebastian Bock, "Improved musical onset detection with convolutional neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.

[18] Jordi Pons, Thomas Lidy, and Xavier Serra, "Experimenting with musically motivated convolutional neural networks," in *International Workshop on Content-Based Multimedia Indexing (CBMI)*. IEEE, 2016, pp. 1–6.

[19] Judith C Brown, "Calculation of a constant q spectral transform," *The Journal of the Acoustical Society of America*, vol. 89, no. 1, pp. 425–434, 1991.

[20] Xavier Glorot, Antoine Bordes, and Yoshua Bengio, "Deep sparse rectifier neural networks," in *International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.

[21] Jordi Pons, Oriol Nieto, Matthew Prockup, Erik M Schmidt, Andreas F Ehmann, and Xavier Serra, "End-to-end learning for music audio tagging at scale," *Workshop on Machine Learning for Audio Signal Processing (ML4Audio) at NIPS*, 2017.

[22] Jordi Pons, Rong Gong, and Xavier Serra, "Score-informed syllable segmentation for a cappella singing voice with convolutional neural networks," *International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[23] Rong Gong, Jordi Pons, and Xavier Serra, "Audio to score matching by combining phonetic and duration information," *International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[24] George Tzanetakis and Perry Cook, "Musical genre classification of audio signals," *IEEE Transactions on speech and audio processing*, vol. 10, no. 5, pp. 293–302, 2002.

[25] Corey Kereliuk, Bob L Sturm, and Jan Larsen, "Deep learning and music adversaries," *IEEE Transactions on Multimedia*, vol. 17, no. 11, pp. 2059–2071, 2015.

[26] Ugo Marchand and Geoffroy Peeters, "Scale and shift invariant time/frequency representation using auditory statistics: Application to rhythm description," in *International Workshop on Machine Learning for Signal Processing*. IEEE, 2016, pp. 1–6.

[27] Pedro Cano, Emilia Gómez, Fabien Gouyon, Perfecto Herrera, Markus Koppenberger, Beesuan Ong, Xavier Serra, Sebastian Streich, and Nicolas Wack, "ISMIR 2004 audio description contest," *Music Technology Group of the Universitat Pompeu Fabra, Technical Report*, 2006.

[28] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *ACM International Conference on Multimedia*, Orlando, FL, USA, Nov. 2014.

[29] Yeonwoo Jeong, Keunwoo Choi, and Hosan Jeong, "Dlr: Toward a deep learned rhythmic representation for music content analysis," *arXiv preprint arXiv:1712.05119*, 2017.

[30] Justin Salamon and Juan Pablo Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.

[31] Sergio Oramas, Oriol Nieto, Francesco Barbieri, and Xavier Serra, "Multi-label music genre classification from audio, text, and images using deep features," *International Society for Music Information Retrieval Conference (ISMIR)*, 2017.