RELATIVE MUSIC LOUDNESS ESTIMATION USING TEMPORAL CONVOLUTIONAL NETWORKS AND A CNN FEATURE EXTRACTION FRONT-END

Blai Meléndez-Catalán
Music Technology Group
Universitat Pompeu Fabra
BMAT Licensing S.L.
Barcelona, Spain
bmelendez@bmat.com

Emilio Molina
BMAT Licensing S.L.
Barcelona, Spain
emolina@bmat.com

Emilia Gómez
Music Technology Group
Universitat Pompeu Fabra
Joint Research Center, EC
Barcelona/Seville, Spain
emilia.gomez@upf.edu

ABSTRACT

Relative music loudness estimation is a MIR task that consists in dividing audio in segments of three classes: *Foreground Music*, *Background Music* and *No Music*. Given the temporal correlation of music, in this work we approach the task using a type of network with the ability to model temporal context: the Temporal Convolutional Network (TCN). We propose two architectures: a TCN, and a novel architecture resulting from the combination of a TCN with a Convolutional Neural Network (CNN) front-end. We name this new architecture CNN-TCN. We expect the CNN front-end to work as a feature extraction strategy to achieve a more efficient usage of the network's parameters. We use the OpenBMAT dataset to train and test 40 TCN and 80 CNN-TCN models with two grid searches over a set of hyper-parameters. We compare our models with the two best algorithms submitted to the tasks of music detection and relative music loudness estimation in MIREX 2019. All our models outperform the MIREX algorithms even when using a lower number of parameters. The CNN-TCN emerges as the best architecture as all its models outperform all TCN models. We show that adding a CNN front-end to a TCN can actually reduce the number of parameters of the network while improving performance. The CNN front-end effectively works as a feature extractor producing consistent patterns that identify different combinations of music and non-music sounds and also helps in producing a smoother output in comparison to the TCN models.

1. INTRODUCTION

One of the main applications of music detection algorithms is the monitoring of music for copyright management [1, 2, 3, 4]. In the copyright management business, collective management organizations tax broadcasters for the music they broadcast. In some cases, the tax is different depending on whether this music is played in the foreground or the background[1]. In the context of broadcast audio, music is used many times in the background, for instance, as a means to create a certain atmosphere. In this scenario, music detection algorithms fall short as we need to estimate the loudness of music in relation to other simultaneous non-music sounds, i.e., its relative loudness.

Motivated by this industrial need, we proposed relative music loudness estimation as a sub-task of the music detection task in MIREX[2] 2018 and 2019. This sub-task is simplified to the segmentation of an audio stream into three classes: *Foreground Music*, *Background Music* and *No Music*. In addition, we published the Open Broadcast Media Audio from TV (OpenBMAT) dataset [5] for training and evaluation, which we use in this work.

In this paper, we study the usefulness of Temporal Convolutional Networks (TCN) for the task of relative music loudness estimation. TCNs are a type of architecture with the ability to model temporal context, which we consider a fundamental characteristic when analyzing temporally correlated signals such as music. We then introduce a CNN front-end to the TCN architecture producing a novel type of network that we name CNN-TCN. We expect the CNN front-end to work as a feature extraction strategy to achieve a more efficient usage of the network's parameters. We train 40 TCN and 80 CNN-TCN models with two grid searches over several hyper-parameters, and compare them among themselves and with the two best algorithms submitted to the tasks of music detection and relative music loudness estimation in MIREX 2019.

2. SCIENTIFIC BACKGROUND

Music detection is the closest task to relative music loudness estimation that we can find in the literature. In the case of music detection, foreground and background music are not separated into two different classes. However, many authors differentiate between foreground and background music in their works: Seyerlehner et al. [2] already mentioned these two concepts while stating that background music is harder to detect. Several other authors [1, 3, 4] agree in the fact that music detection is often applied to scenarios characterized by the presence of background music, effectively differentiating them from scenarios where music has the main role.

Besides, the literature also addresses the task of music detection combined, primarily, with the detection of speech, but also other types of sounds such as noise or environmental sounds. In 1996, Saunders [6] became the first author to publish a paper describing a speech and music segmentation algorithm already achieving outstanding results. His approach, though, assumes that there is no overlap between both classes, which is very frequent in broadcast audio. One year later, Scheirer et al. [7] introduced the overlaps between music and speech as an extra class obtaining an error rate of 35%, which reveals the complexity of the task even when the non-music part includes only speech and not other type of non-

[1] https://createurs-editeurs.sacem.fr/brochures-documents/regles-de-repartition-2017

[2] https://www.music-ir.org/mirex/wiki/MIREX_HOME

music sounds. Richard et al. [8] also followed the segmentation approach using three classes: *Music*, *Speech* and *Mixed*. Their evaluation showed that their algorithm clearly detects when *Speech* is alone, but has more difficulties to differentiate between *Music* and *Mixed*. Lu et al. [9], in 2001, and Panagiotakis et al. [10], in 2005, added other classes such as *Environmental Sounds* and *Silence* to the taxonomies they used, but failed to consider overlaps between classes.

In 2012, Schlüter et al. [11] proposed the first approach to music and speech detection using deep learning architectures. They designed two identical networks, one for music detection and another one for speech detection, based on mean-covariance Restricted Boltzmann Machines. Lidy [12] also used a neural network for his submission to the task of music/speech classification and detection in MIREX 2015. The model he presented is a shallow CNN with only one 2D-convolutional layer; nevertheless, he achieved second place in the competition out of 7 participants. Doukhan et al. [13] used a slightly deeper architecture, but followed the same design pattern, to create a model for music and speech segmentation. Gfeller et al. [14] created a CNN for music detection already including six consecutive separable 2D-convolutional layers. Jang et al. [15] proposed a new type of filter for music detection called melCL filter. It mimics the filters used in the Mel filter bank with the advantage of having weights that can be optimized through backpropagation.

Gimeno et al. [16] proposed a Recurrent Neural Network (RNN) for the task of music and speech segmentation. RNNs can model temporal context, which is a desirable characteristic when working with temporally correlated signals such as speech or music. They presented a model consisting of two stacked Bidirectional Long Short-Term Memory layers (BiLSTM). BiLSTM layers allow the training sequence to be read forward and backwards including both past and future information into the decision for each time step. Moreover, LSTM layers help palliating the vanishing and exploding gradient effects [17]. de Benito-Gorrón et al. [18] evaluated several architectures including CNNs and LSTM networks. The combination of both produced the best performance.

TCNs are a type of deep learning architecture that can also model temporal context. Actually, Bai et al. [19] showed that they offer equal or even better performance than RNNs. Additionally, they do not suffer from exploding/vanishing gradients. We offer a thorough description of these type of networks in Section 3.2. Lemaire et al. [20] used this kind of architecture with non-causal filters for the task of music and speech detection.

The relative music loudness estimation task appeared for the first time in MIREX 2018. Meléndez-Catalán et al. [21] submitted a regression algorithm based on a CNN to this competition. The output of the network is transformed into a class label by means of a set of thresholds and the result smoothed using several heuristic rules. The algorithm reached 86.15% accuracy in MIREX 2018 dataset 1, an early version of the later published OpenBMAT. In MIREX 2019, Meléndez-Catalán et al. presented this same algorithm along with a very similar CNN, and a CNN-TCN prototype developed during the elaboration of the present work [22]. Both new algorithms outperformed the CNN from 2018 with the CNN-TCN obtaining first place.

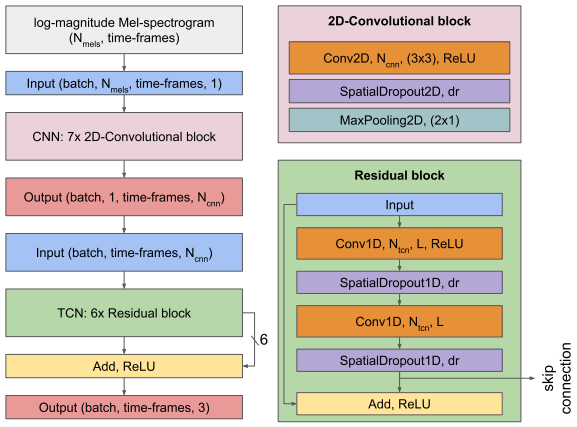

Figure 1: *(left) CNN-TCN architecture. (right-top) Convolutional block. (right-bottom) Residual block.*

3. PROPOSED MODELS

In this section, we detail the models that we propose for the task of relative music loudness estimation: the TCN and the CNN-TCN, which is the combination of a CNN front-end with a TCN. We include an explanation of the input features and the architecture of the models.

3.1. Feature generation

We use audio at 8000 samples per second with 16 bits per sample and normalized to have a maximum amplitude value of 1. From the audio data, we compute the power spectrogram with a Hanning window of length 512 samples (64 ms) and a hop size of 128 samples (16 ms). We then apply a Mel filter bank with $N_{mels} = 128$ filters to obtain the Mel-spectrogram. Then, we change its magnitude scale to logarithmic to produce the log-magnitude Mel-spectrogram. The resulting training instances have 625 time-frames, which is equivalent to 10 seconds, making the input to the network a matrix with shape 128x625. We apply min-max normalization independently to each input to ensure that its values range from 0 to 1.

3.2. Architectures

The TCN model is formed by a stack of 6 residual blocks. A residual block, as defined by He et al. [23], applies a certain function (F) to an input (x) that depends on the weights ($\{W_n\}$) and biases ($\{b_n\}$) of the N layers contained in the residual block. The output of this function is then added back to the input and passed to an activation function to obtain the output of the residual block (y). This way, the layers inside the residual block learn modifications to the input instead of a complete transformation. This has proven to ease their optimization [23]. Eq. 1 presents the formal definition of a residual block.

$$y = Activation(F(x, \{W_n\}, \{b_n\}) + x) \tag{1}$$

In the right-bottom part of Fig. 1, we show the structure of the residual blocks that we use in this paper. Our residual blocks contain two 1D-convolutional layers as proposed by Bai et al. [19].

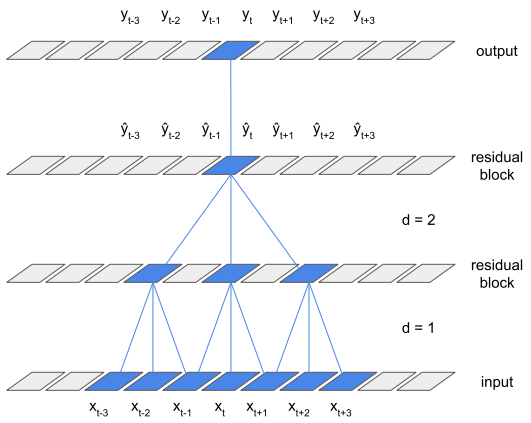

Figure 2: *An example of a TCN's receptive field used to classify a single time-frame. The architecture includes 2 residual blocks with dilations* $d = [1, 2]$ *and non-causal filters of length* $L = 3$*. This network's receptive field is equal to 7 time-frames.*

However, we have removed the activation function of the second 1D-convolutional layer so that the modifications learned by the residual block may include negative values as originally designed by He et al. [23].

All 1D-convolutional layers have N_{tcn} non-causal filters [20] of length L and a spatial dropout rate dr. The 1D-convolutional layers of each subsequent residual block have a higher dilation rate starting at one and increasing by a factor of two for each block. The first 1D-convolutional layer of the TCN reads the log-magnitude Mel-spectrogram as N_{mels} scalar temporal sequences by interpreting the frequency axis as channels.

The temporal context used to classify each time-frame is called receptive field [19]. Eq. 2 shows how the receptive field RF of the TCN depends on the filter length L and the vector of dilation rates d of length D. Using non-causal filters implies that the receptive field is divided equally between past and future time-frames.

$$RF = 1 + \sum_{i=0}^{D} d[i] * (L - 1) \tag{2}$$

Fig. 2 is a simplified representation of a TCN model where $x_t \in \mathbb{R}^{\mathrm{N_{mels}}}$ is the time-frame t of the input features, $\hat{y}_t \in \mathbb{R}^{\mathrm{N_{tcn}}}$ is the output of the last residual block for that time-frame and $y_t \in \mathbb{R}^3$ is the vector that carries the probability of the 3 relative music loudness estimation classes for that time-frame. The network includes 2 residual blocks with $d = [1, 2]$ and non-causal filters of length $L = 3$. Following Eq. 2, we obtain a receptive field of 7 time-frames.

In the left part of Fig. 1, we show the CNN-TCN model, which is a combination of a CNN front-end and the TCN described above. The CNN consists in a stack of 7 blocks that comprehend: (1) a 2D-convolutional layer with N_{cnn} 3x3 filters and a ReLU activation function, (2) a spatial dropout layer with a dropout rate dr and (3) a max-pooling layer. We apply the max-pooling only to the frequency axis reducing its dimensionality by a factor of two for each block until it is equal to one. The TCN reads the output of the CNN as N_{cnn} scalar temporal sequences and models their evolution.

Concatenating a CNN and a TCN does not necessarily produce a model with more parameters than the TCN model alone. Eq. 3 shows how the number of parameters (P) in a 1D-convolutional layer of the TCN depends on the number of input channels (N_{ch}), the number of filters (N_{tcn}), their length (L) and the bias vector (b). Adding the CNN front-end transforms the number of channels that we input to the TCN from $N_{ch} = N_{mels}$ to $N_{ch} = N_{cnn}$. If we reduce the number of channels by using a low N_{cnn} value, we can significantly decrease the number of parameters of the first 1D-convolutional layer of the TCN compensating the number of parameters added by the CNN itself.

$$P = N_{ch} * N_{tcn} * L + b \tag{3}$$

The output layer for both TCN and CNN-TCN architectures has 3 neurons, each of them corresponding to one of the three classes of the relative music loudness estimation task. Using a softmax activation function, the networks outputs the probability of each class for each time-frame. This means they offer predictions with a frame-level time resolution regardless of the input size in the temporal dimension. We assign the class with the highest probability to each time-frame. We refer to a set of contiguous time-frames of the same class as a segment of that class. The segment starts at the beginning of the first time-frame and ends at the end of the last time-frame, in chronological order.

3.3. Smoothing

Classifying at a frame-level allows an algorithm to be very precise in detecting a change of class; however, it also makes it prone to produce short erroneous segments that makes the classification noisy. To solve this issue, we apply a smoothing strategy to the output of our models: we use a sliding window that assigns the most represented class across all time-frames covered by the window to its central time-frame.

4. EXPERIMENTAL SETUP

In this experiment, we carry out two grid searches over a total of four hyper-parameters to find the configuration that produces the best possible TCN (TCN^{best}) and CNN-TCN ($CNNTCN^{best}$) models. We compare these models with the two best algorithms presented to the tasks of music detection and relative music loudness estimation of MIREX 2019. In what follows, we group them as MIREX algorithms. We impose two restrictions: (1) TCN^{best} must have a lower number of parameters than the MIREX algorithms; and (2) $CNNTCN^{best}$ must have a lower number of parameters than TCN^{best}. This way we make sure that any improvement comes from a more appropriate architecture and not just from an increase of the networks learning capacity. In this section, we detail the dataset, describe the MIREX algorithms, and explain the training methodology and evaluation metrics that we use in the experiment.

4.1. Dataset

In this work, we use OpenBMAT [5][3], an open dataset annotated for the tasks of music detection and relative music loudness estimation that contains over 27 hours of TV broadcast audio from *France*, *Germany*, *Spain* and the *United Kingdom* distributed over

[3] https://zenodo.org/record/3381249

1647 one-minute long excerpts. It is the first dataset to include annotations about the loudness of music in relation to other simultaneous non-music sounds and it is designed to encompass two essential features: (1) contain music both isolated and mixed with other type of non-music sounds; and (2) including a significant number of multi-class audio files, i.e., audio files that contain class changes. OpenBMAT has been cross-annotated by 3 annotators obtaining high inter-annotator agreement percentages both for the music detection and the relative music loudness estimation tasks. The annotators were told to ignore any segment shorted than 1 second. The annotation tool that we used for the annotation is BAT [24].[4]

This dataset comes with 10 predefined splits containing approximately 15% *Foreground Music*, 35% *Background Music* and 50% *No Music*. During training, we use nine of them: eight for the training set and one for the development set. The tenth split constitutes the testing set. From each split we only use the audio excerpts that have at least partial agreement, i.e., the parts where at least two annotators agree. For the classes used in the relative music loudness estimation task this supposes 99.79% of the content. We always pick the classes with the most agreement as ground truth.

4.2. MIREX algorithms

We choose two algorithms to compare our models with. They are two of the algorithms that Meléndez-Catalán et al. presented to the tasks of music detection and relative music loudness estimation of MIREX 2019: *M1* [21] and *M2* [22]. *M1* was already submitted to MIREX 2018, where it obtained first place out of 5 participants in the music detection task, and was the first algorithm to participate in the relative music loudness estimation task. In 2019, *M2* and *M1* obtained second and third place, respectively, in both tasks. The winner was a CNN-TCN prototype that Meléndez-Catalán et al. produced during the elaboration of this paper.

M1 and *M2* consist in a CNN with 3 convolutional blocks and 2 dense layers. Each of the convolutional blocks is composed by a 2D-convolutional layer with a ReLU activation, and a max-pooling layer. The two algorithms differ in several hyper-parameters such as the number of 2D-convolutional filters and their size. *M1* and *M2* have a total of 97,779 and 453,763 parameters, respectively. The input to both networks is the log-magnitude Mel-spectrogram, with 128 frequency bins, of approximately 2 seconds of audio, which translates to 128 time-frames. The difference in accuracy between *M2* and *M1* in MIREX 2019 was approximately 2 percentage points (pp) for the task of music detection and 3 pp for the task of relative music loudness estimation. Given that *M2* has approximately 4.5 times the amount of parameters of *M1*, we find it difficult to judge what architecture is more appropriate for these two tasks. That is why we include both of them in the comparison.

Originally, *M1* and *M2* are two-neuron output regression algorithms that adapt to classification through a set of thresholds. Unfortunately, the annotations in OpenBMAT are designed for classification and not for regression. This is why we replace the two-neuron output layers by three-neuron output layers and train the networks for classification. These two algorithms include several rules that aim to smooth their predictions by modifying the class of a particular segment based on its class and duration, and the class and duration of the contiguous segments.

[4] https://github.com/BlaiMelendezCatalan/BAT

We train them for 100 epochs using the ADAM optimizer with learning rate $lr = 0.001$, and the categorical cross-entropy loss function, which we weight to compensate the imbalance in terms of instances per class of the training and development sets. Both algorithms tend to overfit the training set, so we apply a range of dropout rates to the 2D-convolutional layers. We save the models that produces the lowest loss for the development set.

4.3. Training

The training process has two steps: first, we train a set of TCNs through a grid search over the hyper-parameters described in Section 3.2: (1) the number of filters N_{tcn}, (2) the filter length L and (3) the dropout rate dr of the 1D-convolutional layers. (2) allows us to modify the receptive field of the TCN without affecting the model's architecture. With (1) and (3) we experiment with the learning capacity of the network and its regularization, respectively. We train 40 models using the following set of values for each hyper-parameter:

- $N_{tcn} \in [16, 32]$
- $L \in [3, 5, 7, 9, 11]$
- $dr \in [0.0, 0.05, 0.1, 0.15]$

In the second step, we combine these TCN models with two CNNs to generate 80 CNN-TCN models. The 2D-convolutional layers of these CNNs have N_{cnn} 3x3 filters where:

- $N_{cnn} \in [16, 32]$

We choose sets of hyper-parameter values that produce a majority of networks with a number of parameters lower than the number of parameters of *M1*. Given that these networks require regularization through the usage of dropout layers, we considered that they have enough capacity to absorb the training dataset.

We fix the overlap between instances to make sure that we train our models and the MIREX algorithms using approximately the same amount of seconds of audio. The downside of this strategy is that the number of instances differs significantly due to the difference in input size. We fix the overlap to 50%, which results in 12,892 training instances and 1,616 development instances to train our models, and 74,585 training instances and 9,286 development instances to train the MIREX algorithms.

We train all our models for 100 epochs using the ADAM optimizer with learning rate $lr = 0.001$, and the categorical cross-entropy loss function, which we weight to compensate the imbalance in terms of instances per class of the training and development sets. We keep the model that produces the lowest loss for the development set. We shuffle the training data every epoch and present it to the network in batches of 128 instances. We use keras 2.2.4 and tensorflow-gpu 1.12.

4.4. Metrics

To evaluate a model, we run it on the testing set and compute its confusion matrix with ground truth as rows and the model's classification as columns. The values in this confusion matrix are the number of seconds per class classified as each class. By weighting each row by the total number of seconds per class, we obtain a balanced confusion matrix. A balanced confusion matrix shows what percentage of the number of seconds per class is classified as each class, and thus, is independent of the actual balance of the testing set. The statistics that we extract from this balanced confusion

Table 1: *Statistics of M1, M2, TCN^{best} and $CNNTCN^{best}$ with and without smoothing (S). The metrics are described in Section 4.4.*

Model	$bAcc$	bP_{Fg}	bR_{Fg}	bP_{Bg}	bR_{Bg}	bP_{No}	bR_{No}	RS	params
$M1$	81.36%	86.98%	84.22%	73.12%	75.9%	84.49%	83.97%	1.76	97,779
$M1$ (S)	81.7%	88.35%	79.57%	71.98%	79.79%	86.54%	85.74%	0.83	97,779
$M2$	82.57%	85.79%	86.88%	75.42%	75.47%	86.51%	85.34%	1.62	453,763
$M2$ (S)	83.04%	86.56%	83.92%	74.83%	77.86%	88.21%	87.33%	0.81	453,763
TCN^{best}	85.76%	90.08%	89.21%	79.52%	80.39%	87.79%	87.69%	39.27	85,635
TCN^{best} (S)	86.27%	90.68%	89.14%	79.85%	81.63%	88.52%	88.05%	1.54	85,635
$CNNTCN^{best}$	90.05%	91.43%	93.38%	86.18%	84.58%	92.43%	92.18%	11.41	80,963
$CNNTCN^{best}$ (S)	**90.06%**	91.81%	92.79%	85.73%	85.2%	92.61%	92.19%	**1.14**	80,963

Table 2: *Weighted confusion matrices for the $M1$ and $M2$ algorithms with smoothing.*

Ground Truth	*M1* Classified As			*M2* Classified As		
	Fg	Bg	No	Fg	Bg	No
Fg	**79.54%**	18.65%	1.77%	**83.92%**	15.22%	0.86%
Bg	8.64%	**79.79%**	11.56%	11.33%	**77.86%**	10.8%
No	1.85%	12.41%	**85.74%**	1.7%	10.97%	**87.33%**

Table 3: *Weighted confusion matrices for the TCN^{best} and $CNNTCN^{best}$ models with smoothing.*

Ground Truth	TCN^{best} Classified As			$CNNTCN^{best}$ Classified As		
	Fg	Bg	No	Fg	Bg	No
Fg	**89.14%**	10.16%	0.7%	**92.79%**	6.89%	0.32%
Bg	7.66%	**81.63%**	10.72%	7.76%	**85.2%**	7.03%
No	1.51%	10.44%	**88.05%**	0.51%	7.29%	**92.19%**

matrix share this property. We consider: the balanced accuracy ($bAcc$), precision (bP_{class}) and recall (bR_{class}). We also propose a new metric that we name ratio of segments (RS): the ratio between the number of predicted segments (S_{pr}) and the average number of ground truth segments for all annotators (S_{gt}) across (N) files. We formally define RS in Eq. 4. This metric provides relevant information about how noisy a model is regardless of how correct its predictions are, which is another characteristic of its performance. The optimal value of RS is 1.

$$RS = \frac{\sum_{n=0}^{N} S_{pr_n}}{\sum_{n=0}^{N} S_{gt_n}} \tag{4}$$

5. RESULTS AND DISCUSSION

As shown in Table 1, both $CNNTCN^{best}$ and TCN^{best} models outperform the MIREX algorithms in terms of $bAcc$ despite using less parameters. TCN^{best} uses the following hyper-parameters:

- $N_{tcn} = 32$
- $L = 5$
- $dr = 0.15$

The receptive field of this model is 253 time-frames, which is equivalent to approximately 4 seconds. After the smoothing, TCN^{best} obtains a $bAcc$ 4.6 pp higher than $M1$ using 12.4% less parameters and 3.2 pp higher than $M2$ using 81% less parameters. $CNNTCN^{best}$ uses the following hyper-parameters:

- $N_{cnn} = 32$
- $N_{tcn} = 16$
- $L = 7$
- $dr = 0.15$

The receptive field of this model is 379 time-frames, which is equivalent to approximately 6.1 seconds. There is an improvement in $bAcc$, after the smoothing, of 8.4 pp with respect to $M1$ using 17% less parameters and of 7 pp with respect to $M2$ using 82% less parameters. The improvement with respect to TCN^{best} is of 3.8 pp while using 5.5% less parameters. Obtaining better results with less parameters implies that the architecture makes a more efficient usage of its parameters, and thus, is more appropriate for the task. Note that TCN^{best} and $CNNTCN^{best}$ consider, respectively, twice and thrice more temporal context to classify than the MIREX algorithms.

As shown in Table 2 and Table 3, $CNNTCN^{best}$ provides a strong improvement with respect to $M1$ and $M2$ in the detection of background music, which is one of the most challenging types of content due to the low volume of the music [5]. $CNNTCN^{best}$ correctly classifies 34.9% of the background music that $M2$ cannot detect and misclassifies as *No Music*. This percentage rises to 39.2% in the case of $M1$. However, the statistics for the *Background Music* class show that there is still room for improvement for the relative music loudness estimation and music detection tasks.

In the left part of Fig. 3, we observe that all CNN-TCN models achieve better $bAcc$ than any TCN model. This figure includes all TCN and CNN-TCN models. Table 4 shows a comparison between TCN^{best} and a CNN-TCN model that shares the same hyper-parameters. Note that the CNN-TCN model has less parameters and that it outperforms TCN^{best} in terms of $bAcc$. This further proves that using a CNN front-end improves performance.

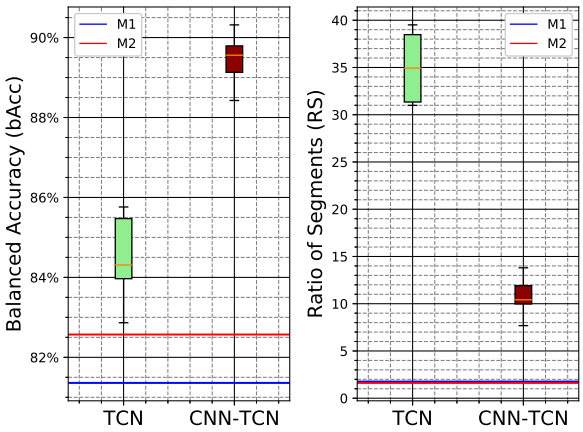

Figure 3: *Comparison between M1, M2 and all the TCN and CNN-TCN models in terms of bAcc and RS without smoothing. The horizontal lines at the bottom correspond to M1 and M2.*

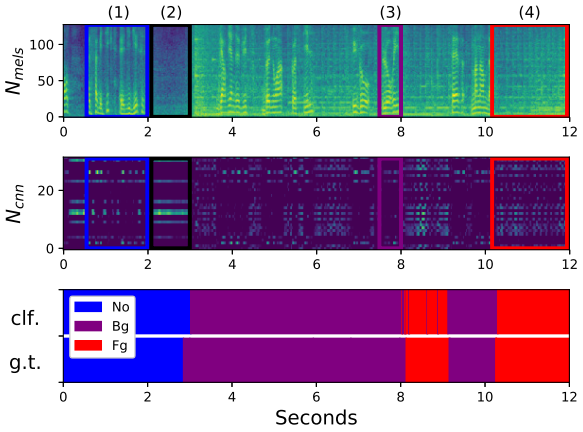

Figure 4: *(top) Example of the log-magnitude Mel-spectrogram, which we use as input features for both the TCN and CNN-TCN architectures. (mid) Output of the CNN in the CNN-TCN architecture for these features. (bottom-top) $CNNTCN^{best}$ classification for these features without smoothing. (bottom-bottom) Ground truth of these features.*

Table 4: *Comparison between TCN^{best} and a CNN-TCN model with the same hyper-parameters except for the dropout rate (dr). We pick the best dropout rate for each model. We do not apply any smoothing.*

Arch	*bAcc*	N_{cnn}	N_{tcn}	*L*	params
TCN^{best}	85.76%	-	32	5	85,635
$CNNTCN$	**89.15%**	16	32	5	**78,211**

The ratio of segments RS in Table 1 shows that both TCN^{best} and $CNNTCN^{best}$ predict a number of segments much higher than the number of segments in the ground truth. In this particular aspect, $M1$ and $M2$ are superior to our models, which are prone to generate noise in the form of short erroneous segments, especially near class changes. The bottom part of Fig. 4 shows an example of this noise around second 8. To remove this noise, we apply the smoothing strategy described in Section 3.3. Using the development set, we evaluate the impact on RS and $bAcc$ of 6 window sizes ranging from 0.5 to 3 seconds with steps of 0.5 seconds. We find an optimal window size of 2 seconds for both models. This window size produces a strong decrease of RS and a light improvement of $bAcc$. Smaller window sizes do not decrease RS enough neither increase $bAcc$ significantly. Larger window sizes start decreasing $bAcc$ as they can remove correct segment of approximately 1 second, which is the minimum segment length in OpenBMAT. As shown in Table 1, after the smoothing, TCN^{best} and $CNNTCN^{best}$ predict 54% and 14% more segments with respect to the segments in the ground truth, respectively. Note in Fig. 3 that all CNN-TCN models produce significantly less noise than any TCN model. This indicates that the CNN front-end also helps in reducing this phenomenon.

Analyzing the duration of the $CNNTCN^{best}$ errors when we apply no smoothing shows that almost 90% of the misclassified segments have a duration lower or equal to 0.2 seconds. These errors amount to approximately 16% of the misclassified time and come mainly from a noisy classification and precision errors in class changes during the annotation or the classification. Listening to the errors with duration equal or higher than 3 seconds, which represent approximately 50% of the misclassified time, we discover several patterns. $CNNTCN^{best}$ misclassifies:

- Loud and mixed non-music sound effects as in action films (*No Music*) classified as *Background Music*.
- Speech mixed with background non-music noises with an identifiable pitch such as engine sounds (*No Music*) classified as *Background Music*
- Low volume background music (*Background Music*) classified as *No Music*. Especially percussive music, live music and tones.
- Loud live music mixed with applauses, cheering and other audience sounds (*Foreground Music*) classified as *Background Music*.

We have analyzed the features extracted by the CNN front-end. The top and mid parts of Fig. 4 show how the CNN front-end works as a feature extractor transforming and reducing the dimensionality of the input log-magnitude Mel-sectrogram. In the example, we observe 4 consistent patterns in the generated features corresponding to 4 sound combinations: (4) isolated music, (1) isolated speech, (3) mixed music and speech, (2) silence. The bottom part of Fig. 4 presents the $CNNTCN^{best}$ classification and the ground truth for the input at the top of the figure. We observe an annotation precision error around second 3, which is shorter than 0.2 seconds and an example of noisy classification in the *Foreground Music* segment between seconds 8 and 9.

6. CONCLUSIONS

In this paper, we have evaluated two architectures for the task of relative music loudness estimation: the TCN and the CNN-TCN, a novel architecture that consists in the combination of a TCN with a CNN front-end. We have run two grid searches over several hyper-parameters training 40 TCN models and 80 CNN-TCN models using the OpenBMAT dataset [5]. We have compared these models

with the two best algorithms submitted to the tasks of music detection and relative music loudness estimation in MIREX 2019: $M1$ and $M2$. We have obtained TCN and CNN-TCN models that outperform these MIREX algorithms while using less parameters. Producing better results with a lower number of parameters means that our architectures make a more efficient usage of its parameters, and thus, are more appropriate for the task. We have named TCN^{best} the TCN that performs the best while using less parameters than $M1$ and $M2$. We have named $CNNTCN^{best}$ the CNN-TCN that performs the best while using less parameters than TCN^{best}.

The results of the evaluation after the smoothing have shown that, in terms of $bAcc$, TCN^{best} outperforms $M1$ by 4.6 pp using 12.4% less parameters and $M2$ by 3.2 pp using 81% less parameters. $CNNTCN^{best}$ beats $M1$ by 8.4 pp using 17% less parameters and $M2$ by 7 pp using 82% less parameters. It also outperforms TCN^{best} by 3.8 pp using 5.5% less parameters. Additionally, our models provide a better classification of background music, which is a challeging type of content due to the low volume of the music [5]. $CNNTCN^{best}$ correctly classifies 39.2% and 34.9% of the background music that $M2$ and $M1$, respectively, misclassify as *No Music*. We have also observed that both TCN^{best} and $CNNTCN^{best}$ use a much larger temporal context for classification than the MIREX algorithms.

The ratio of segments RS has revealed that adding a CNN front-end helps in smoothing the classification in comparison to the isolated TCN. We have also proven that the CNN front-end can reduce the number of parameters of the network with respect to an isolated TCN with the same hyper-parameters while improving its performance. Finally, we have observed that the CNN front-end effectively works as a feature extractor that reduces the dimensionality of the input features and transforms them into consistent patterns that identify different combination of music and non-music sounds.

7. ACKNOWLEDGMENTS

We thank the Catalan Industrial Doctorates Plan for funding this research, especially Jesús Ruiz de la Torre. We also thank Alex Ciurana and Furkan Yesiler for the essential advice that they provided. Finally, we thank Cristina Garrido for helping with the management of the funding.

8. REFERENCES

[1] Yongwei Zhu, Qibin Sun, and Susanto Rahardja, "Detecting musical sounds in broadcast audio based on pitch tuning analysis," in *IEEE International Conference on Multimedia and Expo (ICME)*, 2006, pp. 13–16.

[2] Klaus Seyerlehner, Tim Pohle, Markus Schedl, and Gerhard Widmer, "Automatic music detection in television productions," in *Proceedings of the 10th International Conference on Digital Audio Effects (DAFx-07)*, 2007.

[3] Tomonori Izumitani, Ryo Mukai, and Kunio Kashino, "A background music detection method based on robust feature extraction," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2008, pp. 13–16.

[4] Theodoros Giannakopoulos, Aggelos Pikrakis, and Sergios Theodoridis, "Music tracking in audio streams from movies," in *Proceedings of the IEEE 10th Workshop on Multimedia Signal Processing (MMSP)*, 2008, pp. 950–955.

[5] Blai Meléndez-Catalán, Emilio Molina, and Emilia Gómez, "Open broadcast media audio from tv: A dataset of tv broadcast audio with relative music loudness annotations," *Transactions of the International Society for Music Information Retrieval*, vol. 2, no. 1, pp. 43–51, 2019.

[6] John Saunders, "Real-time discrimination of broadcast speech/music," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1996, vol. 2, pp. 993–996.

[7] Eric Scheirer and Malcolm Slaney, "Construction and evaluation of a robust multifeature speech/music discriminator," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1997, vol. 2, pp. 1331–1334.

[8] Gaël Richard, Mathieu Ramona, and Slim Essid, "Combined supervised and unsupervised approaches for automatic segmentation of radiophonic audio streams," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2007, vol. 2, pp. 461–464.

[9] Lie Lu, Hao Jiang, and HongJiang Zhang, "A robust audio classification and segmentation method," in *Proceedings of the ninth ACM international conference on Multimedia*, 2001, pp. 203–211.

[10] Costas Panagiotakis and George Tziritas, "A speech/music discriminator based on RMS and zero-crossings," in *IEEE Transactions on Multimedia*, 2005, vol. 7, pp. 155–166.

[11] Jan Schlüter and Reinhard Sonnleitner, "Unsupervised feature learning for speech and music detection in radio broadcasts," in *Proceedings of the 15th International Conference on Digital Audio Effects (DAFx-12)*, 2012.

[12] Thomas Lidy, "Spectral convolutional neural network for music classification," *Music Information Retrieval Evaluation eX-change (MIREX)*, 2015.

[13] David Doukhan and Jean Carrive, "Investigating the use of semi-supervised convolutional neural network models for speech/music classification and segmentation," in *The Ninth International Conferences on Advances in Multimedia (MMEDIA)*, 2017.

[14] Beat Gfeller, Ruiqi Guo, Kevin Kilgour, Sanjiv Kumar, James Lyon, Julian Odell, Marvin Ritter, Dominik Roblek, Matthew Sharifi, Mihajlo Velimirović, et al., "Now playing: Continuous low-power music recognition," in *NIPS 2017 Workshop: Machine Learning on the Phone (NIPS)*, 2017.

[15] Byeong-Yong Jang, Woon-Haeng Heo, Jung-Hyun Kim, and Oh-Wook Kwon, "Music detection from broadcast contents using convolutional neural networks with a mel-scale kernel," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2019, no. 1, pp. 11, 2019.

[16] Pablo Gimeno, Ignacio Viñals, Alfonso Ortega, Antonio Miguel, and Eduardo Lleida, "A recurrent neural network approach to audio segmentation for broadcast domain data.," in *IberSPEECH*, 2018, pp. 87–91.

[17] Yoshua Bengio, Patrice Simard, Paolo Frasconi, et al., "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[18] Diego de Benito-Gorron, Alicia Lozano-Diez, Doroteo T Toledano, and Joaquin Gonzalez-Rodriguez, "Exploring convolutional, recurrent, and hybrid deep neural networks for speech and music detection in a large audio dataset," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2019, no. 1, pp. 9, 2019.

[19] Shaojie Bai, J Zico Kolter, and Vladlen Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[20] Quentin Lemaire and Andre Holzapfel, "Temporal convolutional networks for speech and music detection in radio broadcast," in *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, 2019, pp. 229–236.

[21] Blai Meléndez-Catalán, Emillio Molina, and Emilia Gomez, "Music and/or speech detection mirex 2018 submission," 2018.

[22] Blai Meléndez-Catalán, Emillio Molina, and Emilia Gomez, "Music detection mirex 2019 submission," 2019.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[24] Blai Meléndez-Catalán, Emilio Molina, and Emilia Gomez, "BAT: an open-source, web-based audio events annotation tool," in *3rd Web Audio Conference*, 2017.