AAC ENCODING DETECTION AND BITRATE ESTIMATION USING A CONVOLUTIONAL NEURAL NETWORK

Daniel Seichter [#⋆] *Luca Cuccovillo* [#] *Patrick Aichroth* [#]

[#] Fraunhofer Institute for Digital Media Technology, 98693 Ilmenau, Germany
[⋆] Ilmenau University of Technology, 98684 Ilmenau, Germany

ABSTRACT

In this paper, we propose a new method for AAC encoding detection and bitrate estimation from PCM material. The algorithm is based on a Convolutional Neural Network that can distinguish between eight different bitrates. It achieves an average accuracy of 94.65% by analysis of only 116.10 ms of content.

***Index Terms*—** AAC encoding detection, bitrate estimation, audio quality control, Convolutional Neural Network

1. INTRODUCTION

Considering today's complex media production workflows, and the exchange of content among providers, archives, broadcasters etc., it is not uncommon to have unintended or undocumented encoding steps within the content lifetime. This can lead to quality issues or redundant storage. In order to address this problem, it is useful to have algorithms for detecting *previous* encoding steps and estimate the applied bitrates (including higher bitrates, as applied for archival purposes) based on an analysis of the decoded content. Such information can be used e.g. to avoid unintended double encoding or further processing of material in production workflows, to select superior quality items among perceptual duplicates, or to reencode material to save storage space without compromising quality.

AAC [1] is one of the most popular audio codecs, being used e.g. by Apple iTunes, Youtube, and within DVB and DAB standards, and hence an important candidate for respective algorithms. However, while several approaches for single and double MP3 encoding detection exist [2–5], only few methods address AAC: Jin et al. [6] investigated double AAC encoding detection within AAC content, which is significantly different from detection in decoded material. Gaertner at al [7] provided a framing grid analysis approach including AAC, however focusing on tampering detection and hence not addressing bitrate estimation. Finally, Luo et al. [8] presented a method for AAC single encoding detection from uncompressed PCM files, albeit evaluated only for lower bitrates.

In this paper, we propose a new method for AAC encoding detection and bitrate estimation from decoded content, based on a Convolutional Neural Network (CNN). The method achieves an accuracy of 94.65% for only 116.10 ms of analyzed material, and 97.90% for 2 s of analyzed material.

The paper is organized as follows: Section 2 introduces Convolutional Neural Networks (CNNs) and their relevance for AAC encoding detection. Section 3 describes the overall approach, including topology and parameters of the CNN. The evaluation approach and results are presented in Section 4, and Section 5 closes with a summary and ideas for further improvements and work.

2. CONVOLUTIONAL NEURAL NETWORKS

According to [9], GPU-based *feed-forward CNNs* are one of the current de-facto standards for Neural Networks (NNs): Such networks, first presented in [10], were successfully applied e.g. to image classification [11], multi-digit number recognition [12] and video classification [13].

Feed-forward NNs can be seen as a complex structure, where the l-th layer receives an input $o^{(l-1)}$ from the previous layer, computes an activation $z^{(l)} = g_l(o^{(l-1)})$, and produces an output $o^{(l)} = f_l(z^{(l)})$. The specific type of layer is uniquely determined by both the input function $g_l(\cdot)$ and the output function $f_l(\cdot)$. While in standard feed-forward NNs, each neuron of a specific layer is connected to *every* neuron of the previous layer. *CNNs*, in contrast, have neurons being locally connected to overlapping *regions* of the input by means of weight sharing [10].

2.1. Relevance of CNN for AAC encoding detection

Modified Discrete Cosine Transform (MDCT) coefficients are a key information carrier for AAC encoding detection, as evident from previous work on AAC and MP3 encoding detection [2–6, 8]: All approaches rely on the analysis of quantization traces left on MDCT coefficients during encoding.

Luo et al. [8] use pre-existing MP3 detection features based on band-pass filtering of MDCT coefficients [5], and

This work has been conducted within *AudioTrust+*, a research project partially funded by the Thuringian Ministry for Economic Affairs, Science and Digital Society (TMWWDG)

extended it for AAC detection, thereby also using Mel-Frequency Cepstral Coefficients (MFCCs) and their first and second derivatives. As a consequence, the respective classification is based on a complex description of local interconnections between *adjacent frequency-bins*.

Jin et al. [6] address double AAC encoding detection with an algorithm that operates directly in the compressed domain, based on the analysis of transitions of Huffman Codebooks for consecutive frames. Their findings suggest that the *evolution in time* of quantized frames may be related to the applied encoding.

If we consider the matrix of MDCT coefficients as an image, with the y-axis corresponding to the *time* domain, and the x-axis corresponding to the *frequency* domain, the intrinsic local connectivity of the CNNs is a very useful property: It allows us to address local interconnections in the frequency domain and in the time domain, thus exploiting the findings in both [8] and [6]. Moreover, the existence of several hidden layers provides robustness with respect to content variability.

3. PROPOSED ALGORITHM

The proposed algorithm consists of two consecutive phases:

1. Extraction of *quantized* MDCT coefficients, as described in Section 3.1; without this step, the input MDCT coefficients would not expose quantization traces, thus invalidating the whole procedure.

2. Application of a CNN to determine whether the input file was previously encoded with AAC, as described in Section 3.2, also estimating the bitrate that was used.

3.1. Extraction of quantized MDCT coefficients

The AAC standard [1] requires the use of *sine* or *kbd* (Kaiser-Bessel derived) window shapes and 4 possible sequences (long, long-start, 8-short, long-stop), for a total number of 16 possible combinations (2 shapes $\times$ 4 sequences $\times$ 2 sides of each window).

As a consequence, the correct retrieval of *quantized* MDCT coefficients from decoded AAC files is not trivial. Several conditions need to be met, namely:

1. The extraction must happen on the same framing grid used during the encoding.

2. The left and the right shape of the applied window function must be correct.

3. The window sequence of the applied window function must be correct.

Let us denote with x the input file and with $X \in \mathbb{R}^{L\times 1024}$ its quantized MDCT coefficients. In order to compute the L frames of X the following operations are performed:

1. Identify the framing grid offset o_{grid} by applying the approach from Gärtner et al. [7], and erase the first o_{grid} samples of x.

2. Compute the MDCT coefficients of every possible combination c, obtaining the set of MDCT matrices $\mathcal{X} = \{X^c\}, X^c \in \mathbb{R}^{L\times 1024}$.

3. Compute the amount of zero coefficients z_l^c and the number of unique values u_l^c for the l-th row X_l^c of every X^c.

4. Build X incrementally, by selecting for each frame l the correct combination c_l^*, i.e.,

$$X_l := X_l^{c_l^*}, \forall l \in [1, \ldots, L]\,.$$

c_l^* is chosen by analyzing the ratios $rz_{c_1,c_2} = (z_l^{c_1}/z_l^{c_2})$ and $ru_{c_1,c_2} = (u_l^{c_1}/u_l^{c_2})$ for every pair $c_1 \neq c_2$. Intuitively, if $c_1 = c_l^*$ then rz_{c_1,c_2} is high and ru_{c_1,c_2} is low for every c_2.

3.2. Network structure

The input of the proposed CNN is the matrix X of the quantized MDCT coefficients of the input file x, computed as described in Section 3.1. Each element $y(i)$ of the vector y returned by the CNN can be interpreted as the probability of x belonging to the i-th class.

Before feeding any input data to the CNN a preprocessing is applied, i.e.,

1. Log-scaling: $\quad X_{l,j} \longleftarrow \log_{10}(|X_{l,j}| + 1)$,

2. Standardization: $\quad X_{l,j} \longleftarrow (X_{l,j} - \bar{\mu}_j)\,/\bar{\sigma}_j$,

with $\bar{\mu}_j$ and $\bar{\sigma}_j$ being the sample mean and sample standard deviation of the j-th column of all matrices used by the network during the training. $X_{l,j}$ represents the j-th MDCT coefficient of the l-th frame of the input matrix. After the preprocessing, the matrix $X \in \mathbb{R}^{L\times 1024}$ can finally enter the CNN shown in Figure 1.

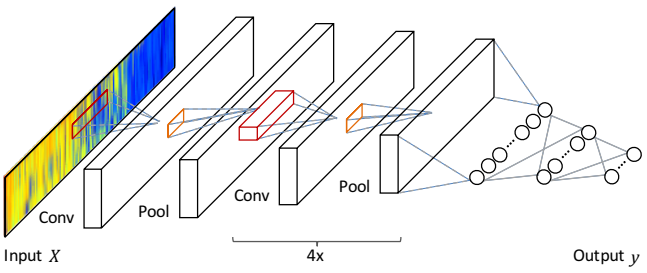

Fig. 1. Structure of the proposed CNN

The network consists of a first convolutional stage, and a second full-connected stage: In the first stage, *convolutional*

layers ($h^{(1)}, h^{(3)}, h^{(5)}, h^{(7)}, h^{(9)}$) with Rectified Linear Units (ReLUs) and *max-pooling* layers ($h^{(2)}, h^{(4)}, h^{(6)}, h^{(8)}, h^{(10)}$) alternate between each others; in the second stage, a *full-connected* layer $h^{(11)}$ of ReLUs connects the output of $h^{(10)}$ to the input of the last *softmax* layer $h^{(12)}$. In order to avoid over-fitting, dropout [14] was applied to the second stage of the network during the training phase.

Details and parameters of the network, including the size of shapes and strides used by each CNN layer, are reported in Table 1 with the format [size_y × size_x]. As evident by analyzing the dimensions of the shapes, the first stage is addressing the local interconnections in the frequency domain (x-axis), while the evolution in time (y-axis) is handled by the second stage.

Table 1. Parameters of the proposed network

Layer	CNN Stage			FC stage
	Channels	Shape	Stride	Neurons
$h^{(1)}$	48	$[1 \times 9]$	$[1 \times 1]$	
$h^{(2)}$		$[1 \times 4]$	$[1 \times 2]$	
$h^{(3)}$	64	$[1 \times 9]$	$[1 \times 1]$	
$h^{(4)}$		$[1 \times 4]$	$[1 \times 2]$	
$h^{(5)}$	80	$[1 \times 9]$	$[1 \times 1]$	
$h^{(6)}$		$[1 \times 4]$	$[1 \times 2]$	
$h^{(7)}$	96	$[1 \times 9]$	$[1 \times 1]$	
$h^{(8)}$		$[1 \times 4]$	$[1 \times 2]$	
$h^{(9)}$	96	$[1 \times 9]$	$[1 \times 1]$	
$h^{(10)}$		$[1 \times 4]$	$[1 \times 2]$	
$h^{(11)}$				3600
$h^{(12)}$				9

The model identification is performed after fixing the dimension of the input matrix, i.e., $X \in \mathbb{R}^{4 \times 1024}$. These dimensions reflect the choice of training a network able to recognize the encoding by using only 4 frames, i.e., 116.10 ms of an audio recording sampled at 44.1 kHz.

4. EVALUATION AND RESULTS

The system was evaluated using AAC files encoded with neroAacEnc v1.5.4[1]. Differences with other encoders were not included for conciseness: As long as quantized MDCTs are retrieved, the CNN is able to generalize correctly. Nine classes were considered, as shown in Table 2.

The initial content consisted of 50 PCM files, each of them being subsequently encoded with all bitrates, then annotated and decoded. From each channel of all files, 23 non-overlapping excerpts of 2 s were extracted from each channel, leading to a total amount of 2300 segments (23 excerpts × 2 channels × 50 files). Each segment produced 84 half-overlapping MDCT frames, i.e., 48300 examples *per class* were produced for the system. The division of content into training, validation and test sets is shown in Table 3 and led to a training time of 5.3 min/epoch.

[1] http://www.nero.com/eng/company/about-nero/nero-aac-codec.php

Table 2. Labels of known classes $\mathcal{C} = \{\mathcal{C}_i\}$

	PCM	AAC (kbps)							
		32	48	64	96	128	192	256	320
$\mathcal{C}_i$	1	2	3	4	5	6	7	8	9

Table 3. Content setup

Target Set	Amount per class (#)			
	Files	Segments	Frames	Examples
Training	20	920	77280	19320
Validation	10	460	38640	9660
Test	20	920	77280	19320

The three sets are completely disjoint: Every original file and all of its encoded versions belong to the same set, in order to avoid any evaluation bias. All results reported in Section 4.1 and Section 4.2 refer to previously unused samples of the test set.

4.1. Quantized MDCT coefficients extraction

A preliminary evaluation addressed the quality of the MDCT coefficient extraction in Section 3.1, by comparing the detected sequence with the ground truth information embedded in the bitstream of the encoded AAC files. It is important to note that neroAacEnc v1.5.4 is not using all possible combinations, but only the four combinations listed in Table 4.

Table 4. Window combinations used by neroAacEnc

Combination c	Sequence	Shape	
		left	right
1	long	kbd	kbd
2	long-start	kbd	sine
3	8-short	sine	sine
4	long-stop	sine	kbd

As shown in Table 5, the algorithm achieved an overall accuracy higher than 95%, independently from the investigated bitrate, and was thus considered suitable for our purposes.

4.2. AAC encoding detection and bitrate estimation

A second evaluation addressed the AAC encoding detection and bitrate estimation at example level, i.e., by considering

Table 5. Evaluation of the MDCT coefficients extraction

Bitrate (kbps)	Window detection accuracy (%)				
	Long	Start	Short	Stop	Overall
32	97.3	93.4	93.0	91.9	96.8
48	97.0	95.2	95.4	93.2	96.8
64	97.3	96.1	96.4	94.3	97.1
96	97.7	95.4	97.5	95.0	97.5
128	98.5	97.9	98.9	96.5	98.4
192	97.2	97.8	99.6	96.7	97.5
256	97.3	96.6	98.9	96.9	97.4
320	98.6	94.7	98.6	96.9	98.2

only 116.10 ms (4 MDCT frames at 44.1 kHz) of content:

Let y be the output of the proposed CNN referring to the input example. The detected class $\bar{\mathcal{C}}$ was determined as described in the following equation:

$$\bar{\mathcal{C}} = \mathcal{C}_i, \quad i = \arg\max_i \Big(y(i)\Big). \tag{1}$$

The resulting Confusion Matrix between the real class $\mathcal{G}$ and the detected class $\bar{\mathcal{C}}$ on the test set is shown in Table 6. The percentages relate to 19320 examples per class, and correspond to a overall accuracy of 94.65%.

Table 6. Confusion Matrix (input of length 116.10 ms)

$\mathcal{G}$	$\bar{\mathcal{C}}$ as in equation (1)								
	1	2	3	4	5	6	7	8	9
1	94.7	0.1	0.1	0.9	2.4	0.7	0.4	0.5	0.2
2	0.0	96.9	3.0	0.1	0.0	0.0	0.0	0.0	0.0
3	0.0	5.9	91.0	3.1	0.0	0.0	0.0	0.0	0.0
4	0.0	0.1	1.2	97.7	1.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.6	98.8	0.5	0.1	0.0	0.0
6	0.1	0.0	0.0	0.1	3.4	95.7	0.6	0.1	0.0
7	0.0	0.0	0.0	0.0	0.6	0.8	94.5	3.9	0.2
8	0.2	0.1	0.0	0.1	0.3	0.1	8.3	90.6	0.3
9	0.7	0.1	0.0	0.2	0.3	0.1	1.2	5.4	92.0

A third evaluation addressed AAC encoding detection and bitrate estimation at segment level:

Let $\mathcal{X} = \{X_j\}, X_j \in \mathbb{R}^{4\times 1024}$ be the set of input MDCT matrices and $\mathcal{Y} = \{y_j\}, y_j \in \mathbb{R}^9\}$ the set of output vectors of the CNN obtained by one input segment x – i.e., the amount $N := \|\mathcal{X}\| = \|\mathcal{Y}\|$ of elements in $\mathcal{X}$ and $\mathcal{Y}$ is equal to 21, and corresponds to approximately 2 s of content.

The detected class $\bar{\mathcal{C}}$, based on fusion, was determined as follows:

$$\bar{\mathcal{C}} = \mathcal{C}_i, \quad i = \arg\max_i \left(\frac{1}{N}\cdot\sum_{j=1}^{N} y_j\,(i)\right). \tag{2}$$

The complete Confusion Matrix is reported in Table 7, where percentages refer to 920 segments per class. Every element on the main diagonal, corresponding to the accuracy of the relative class, increased due to the higher amount of available information. The average accuracy of this segment-based approach consequently raised to 97.9%, confirming the overall high reliability of the system.

Table 7. Confusion Matrix (input of length 2 s)

$\mathcal{G}$	$\bar{\mathcal{C}}$ as in equation (2)								
	1	2	3	4	5	6	7	8	9
1	96.9	0.0	0.0	0.2	2.6	0.3	0.0	0.0	0.0
2	0.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	1.5	98.1	0.4	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	100.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	100.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	2.2	97.8	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	6.1	93.9	0.0
9	0.1	0.0	0.0	0.0	0.1	0.0	0.8	4.5	94.5

The work by Luo et al. [8] reached an average accuracy of 92.33% on a private dataset and of 94.74% on the GTZAN dataset. However, these results were achieved by analyzing 5 s of mono content, encoded with lower AAC bitrates (32, 64, 96 and 128 kbps), and 22.05 kHz sampling rate, while our approach aims at CD-quality stereo content sampled at 44.1 kHz, high bitrates, and short analysis segments. Considering the different goals, and the fact that [8] is based on features that are highly sensitive to sampling frequency or analysis window length changes, a comparison would have been misleading, and was therefore not performed.

5. CONCLUSIONS AND OUTLOOK

To the best of our knowledge, the method proposed in this paper represents the first attempt to perform AAC encoding detection and bitrate estimation by means of Deep Learning. The system was tested for 8 different bitrates, spanning the most common configurations used by AAC encoders, thereby achieving an average accuracy of 94.65% by analyzing only 116.10 ms of content. The possibility of performing a fusion led to an even higher accuracy of 97.9%.

The application of a CNN was based on an assessment of previous works in this domain [6, 8], and it reflects the choice of addressing both the evolution in time, and the interconnections between adjacent frequency bins, which proved to be the key carriers of information in the aforementioned approaches.

For the near future, it is planned to use a CNN for double AAC encoding detection from PCM files. Moreover, we would like to apply such a network to narrow-band content like speech, to support forensics analysis for content authentication purposes.

6. REFERENCES

[1] ISO/IEC, "International standard ISO/IEC 13818-7:1997 - information technology – generic coding of moving pictures and associated audio information – part 7: Advanced audio coding (AAC)," 1997.

[2] Tiziano Bianchi, Alessia De Rosa, Marco Fontani, Giovanni Rocciolo, and Alessandro Piva, "Detection and classification of double compressed MP3 audio tracks," in *Proceedings of the First ACM Workshop on Information Hiding and Multimedia Security*, New York, NY, USA, 2013, MMSec '13, pp. 159–164, ACM.

[3] Mengyu Qiao, A.H. Sung, and Qingzhong Liu, "Improved detection of MP3 double compression using content-independent features," in *Signal Processing, Communication and Computing (ICSPCC), 2013 IEEE International Conference on*, Aug 2013, pp. 1–4.

[4] S. Hicsonmez, E. Uzun, and H.T. Sencar, "Methods for identifying traces of compression in audio," in *Communications, Signal Processing, and their Applications (ICCSPA), 2013 1st International Conference on*, Feb 2013, pp. 1–6.

[5] Da Luo, Weiqi Luo, Rui Yang, and Jiwu Huang, "Compression history identification for digital audio signal," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, March 2012, pp. 1733–1736.

[6] Chao Jin, Rangding Wang, Diqun Yan, Pengfei Ma, and Jinglei Zhou, "An efficient algorithm for double compressed AAC audio detection," *Multimedia Tools and Applications*, pp. 1–18, 2015.

[7] Daniel Gärtner, Christian Dittmar, Patrick Aichroth, Luca Cuccovillo, Sebastian Mann, and Gerald Schuller, "Efficient cross-codec framing grid analysis for audio tampering detection," in *Audio Engineering Society Convention 136*, Apr 2014.

[8] Da Luo, Weiqi Luo, Rui Yang, and Jiwu Huang, "Identifying compression history of wave audio and its applications," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 10, no. 3, pp. 30:1–30:19, April 2014.

[9] Jürgen Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015, Published online 2014; based on TR arXiv:1404.7828 [cs.NE].

[10] Kumar Chellapilla, Sidd Puri, and Patrice Simard, "High performance convolutional neural networks for document processing," in *Tenth International Workshop on Frontiers in Handwriting Recognition*, 2006.

[11] Matthew D. Zeiler and Rob Fergus, "Visualizing and understanding convolutional networks," *CoRR*, vol. abs/1311.2901, 2013.

[12] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet, "Multi-digit number recognition from street view imagery using deep convolutional neural networks," *CoRR*, vol. abs/1312.6082, 2013.

[13] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and Li Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, June 2014, pp. 1725–1732.

[14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.