Now Playing: Continuous low-power music recognition

Blaise Agüera y Arcas*, **Beat Gfeller, Ruiqi Guo, Kevin Kilgour, Sanjiv Kumar, James Lyon, Julian Odell, Marvin Ritter, Dominik Roblek, Matthew Sharifi, Mihajlo Velimirović**
Google Research
{blaisea,beatg,guorq,kkilgour,sanjivk,jameslyon,
juliano,marvinritter,droblek,mns,mvelimirovic}@google.com

Abstract

Existing music recognition applications require a connection to a server that performs the actual recognition. In this paper we present a low-power music recognizer that runs entirely on a mobile device and automatically recognizes music without user interaction. To reduce battery consumption, a small music detector runs continuously on the mobile device's digital signal processor (DSP) chip and wakes up the main application processor only when it is confident that music is present. Once woken, the recognizer on the application processor is provided with a few seconds of audio which is fingerprinted and compared to the stored fingerprints in the on-device fingerprint database of tens of thousands of songs. Our presented system, *Now Playing*, has a daily battery usage of less than 1 % on average, respects user privacy by running entirely on-device and can passively recognize a wide range of music.

1 Introduction

When someone is curious about an unknown song that is being played, they currently launch a music recognition application that captures several seconds of audio and uses a server for recognition. Ideally all a curious user would have to do is glance at their mobile device to find out which song is being played without even having to unlock it. There are multiple challenges that have to be tackled in order to make this possible. In this paper we present a first of its kind, continuous low-power music recognizer.

An overview of this setup can be seen in figure 1. First, an audio fingerprinter is required that can run on a mobile device. We developed a neural network fingerprinter to produce compact but discriminative fingerprints from short segments of audio. A small database (DB) of fingerprint sequences generated from selected songs is used by a matching algorithm that can quickly match the short sequence of fingerprints from the audio segment.

Because the fingerprinter is relatively computationally expensive and would have a significant impact on the battery if left running all day, a gatekeeper music detector is necessary to avoid triggering it when no music is present. This music detector runs on a separate DSP chip and only wakes the main processor when music is present, see Figure 1. Since everything runs locally on the device without sending either audio or fingerprints to a server, the privacy of the user is respected and the whole system can run in airplane mode.

2 Related Work

To solve the task of audio content identification, Ouali et al. [12] developed a spectrogram-based audio fingerprint which uses multiple binary spectrograms created with various thresholds. Each of the computed fingerprints is then compared against known references. Baluja and Covell [3]

*Authors are listed in alphabetical order by last name.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

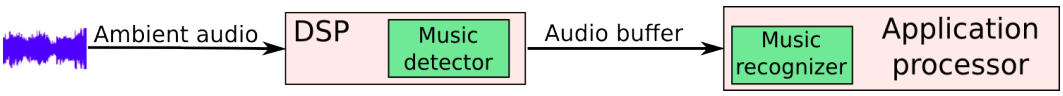


Figure 1: *Setup of the continuous music recognition system. The neural network fingerprinter and the fingerprint matcher run on the application processor which is woken up and provided a buffer of audio when the music detector is confident that music is present.*

applied computer vision techniques to extract a fingerprint which can be indexed at large scale using locality-sensitive hashing (LSH) [7].

Another approach that uses binary fingerprints was developed by Tsai et al. [16]. Their features are derived by applying the constant-Q transform to the audio which is a transformation where the spacing and width of its filters match the pitches of the musical scale. The fingerprints are then binarized using a hamming embedding.

Malekesmaeili and Ward [11] designed an audio fingerprint which is invariant to time or frequency scaling. Their fingerprint is based on the time-chroma representation of the audio signal which groups all pitches that differ by an octave together. Unlike the logarithmic Mel scale, pitch shifting a song by a full octave will result in the same time-chroma representation and pitch shifts by fractions of an octave only shift the values along the chroma axis. They show that their fingerprints can outperform scale invariant feature transform (SIFT) based fingerprints.

An audio fingerprinting technique explicitly designed for embedded applications is proposed by Plapous et al. [13] where they forgo the fast Fourier transform (FFT) that most other fingerprint techniques use in favor of using optimized infinite impulse response (IIR) filter banks. They were able to improve both the computational speed of the audio fingerprints as well as reduce their size while maintaining the accuracy of their baseline Robust Audio Hashing technique [9].

Chandrasekhar et al. [4] present an overview of music identification techniques and compare their usefulness for mobile devices based on a set of noisy audio clips.

A number of popular music identification applications are available for mobile devices such as Shazam [18, 17], SoundHound [1] and Google Sound Search [2]. These applications are typically manually triggered by users and require a connection to a server for recognition.

3 Neural Network Fingerprinter

At the heart of our music recognizer is the neural network fingerprinter (NNFP) which analyzes a few seconds of audio and emits a single fingerprint embedding at a rate of one per second. A detailed structure of the NNFP can be seen in Figure 2a. A stack of convolutional layers is followed by a two-level divide-and-encode block [10] which splits the intermediate representation into multiple branches. All layers except for the final divide-and-encode layer use the ELU [5] activation function and batch normalization.

We trained the network using the triplet loss function [15], which for every audio segment minimizes the distance to examples of the same audio segment, while ensuring that their distances to all other audio segments are larger. In this context, audio segments are only considered the same if their starting positions differ by less than a few hundred milliseconds and are from the same song.

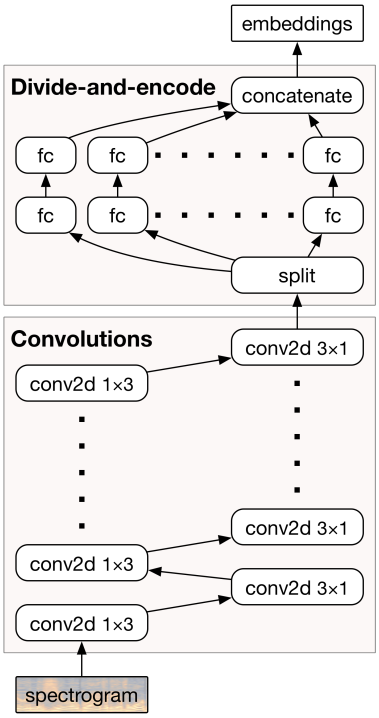
The NNFP model is trained on a dataset of noisy music segments which are aligned to the corresponding segment in their reference song.

4 Fingerprint Matching

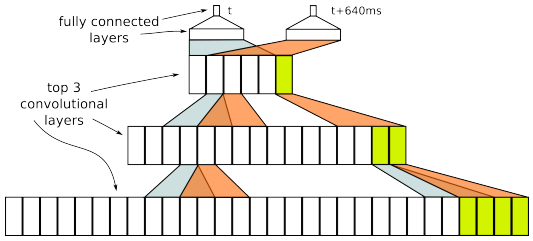
During matching, we need to identify the fingerprints sequence in our song DB which most closely matches the sequence of fingerprints generated by the query audio. We perform sequence search in two stages. First, each fingerprint of the query sequence is searched using an approximate nearest neighbor method to find the top-K nearest fingerprints in the DB. Then, we perform fine-grained scoring with the most promising candidate sequences.

Due to constraints on energy consumption and storage, we had to compress the DB while supporting fast on-device retrieval. We adopted a strategy similar to Guo et al. [8], Wu et al. [19], which uses a hybrid solution of trees and quantization where parameters are trained in a data-dependent fashion.

At indexing time, we minimize the quantization error $\|x - \tilde{x}\|_2$, where x is a fingerprint vector and $\tilde{x}$ is its compressed approximation, as a proxy for minimizing the distance approximation error when searching for query fingerprint q: $|\|q - x\|_2 - \|q - \tilde{x}\|_2|$.



(a) *Topology of the music recognizer.*



(b) *Music detector inference. Highlighted layer vectors are required for the next output at* $t + 640$ ms.

layer	size-in	kernel
separable conv2d	446x1x32	4x1, 2
separable conv2d	222x1x32	4x1, 2
separable conv2d	110x1x32	4x1, 2
separable conv2d	54x1x32	4x1, 2
separable conv2d	26x1x32	4x1, 2
separable conv2d	12x1x32	4x1, 2
flatten	5x1x32	
fully connected	160	
fully connected	8	

(c) *Music detector topology. All layers but the last use the ReLU activation function and batch normalization.*

Figure 2: *Architectures of the music detector and recognizer neural networks.*

At search time, vector quantizers are used to identify a set of partitions close to the query. An exhaustive search of those partitions only touches approximately 2 % of the DB. The stored quantized encoding of the fingerprints is approximately 32 times smaller than the original floating point representation.

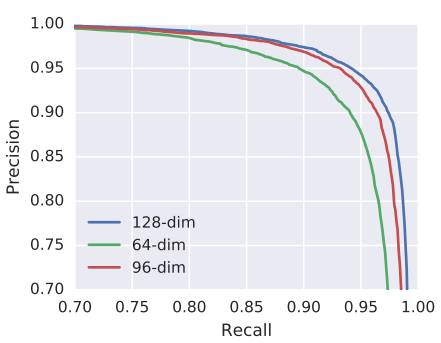
Most errors from searching the subset of compressed fingerprints can be recovered during the sequence matching stage. This stage reuses promising partial matches found through an approximate search, and retrieves full sequences of fingerprints from the DB. Our sequence similarity metric accounts for search errors and uses a method inspired by Qin et al. [14], where the similarity between two fingerprints is determined by the local density of fingerprints in our song DB and adjusts the final acceptance threshold based upon the set of similar sequences found. The effectiveness of this approach is shown in Figure 3b.

5 Music Detector

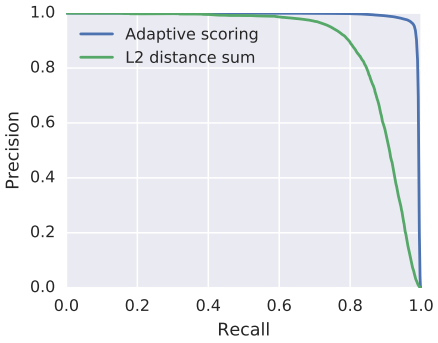
The music detector runs continuously on a separate DSP chip and acts as a gatekeeper so the computationally expensive fingerprints are only computed when ambient music is playing. Memory and operations per second are extremely limited on the DSP chip so as to avoid draining the device's battery. The music detector runs in 3 stages. In the first stage, log Mel features are extracted from the audio stream.

In the next stage, a neural network computes the probability of music playing using a window of the computed log Mel feature vectors. The network, described in detail in table 2c consists of a stack of convolutional layers, each of which reduces the dimensionality of its input by a factor of two. In total the model has 8 k parameters and occupies less than 10 KB of memory. The kernel stride of 2 means that each convolutional layer requires 2 new input vectors in order to produce a new output vector. This can be seen in figure 2b which depicts the network's top 3 convolutional layers. The network containing a stack of 6 convolutional layers therefore outputs a new prediction every 2^6 frames or 640 ms.

The music detector neural network is trained on a dataset containing a subset of AudioSet [6] and an additional set of noise-added audio clips which were labeled with music present or not present.



(a) Performance of embeddings with different numbers of dimensions. The models generating these embeddings are identical except for the final divide-and-encode block.



(b) Comparison of adaptive scoring similarity metric against a naïve implementation using L^2 distance.

Figure 3: *Evaluation of the neural network fingerprinter and fingerprint matcher.*

During training we take a random sub-clip as the input for the network. We simulate quantized activations during training.

In the final stage the stream of predictions from the neural network is averaged over a sliding window of a few seconds. After c consecutive music predictions with a confidence over a threshold t, a detection is registered and an audio buffer is sent to the fingerprinter on the application processor. This smoothing and aggregation helps to filter out some of the neural network's errors and ensures that the audio buffer contains a sufficient amount of music to be recognized.

6 Evaluation

Given the limited space available on a mobile device for the fingerprint DB, we had to find a compromise fingerprint size. Smaller fingerprints allow for more songs in the DB while larger fingerprints result in a higher recognition accuracy. We evaluated the performance of the NNFP and the matching algorithm using 64, 96 and 128 dimensional fingerprints on a set of 20 k 8 s long audio segments from 10 k different songs. Figure 3a shows the precision/recall curves of these three models. The 96 and 128 dimensional models clearly outperform the 64 dimensional model with the 128 dimensional model being only slightly better than the 96 dimensional model. We decided to use the 96 dimensional model so that each song occupies on average less than 3 KB in our DB, roughly 30 % less than the 128 dimensional model.

6.1 Music detector performance

In order to evaluate the performance of the music detector, we tested it on 12 k short (16 s - 40 s) music regions within a 450 h audio test set. The test set contains various background noises such as speech, office noise, street sounds, walking, and vehicle (train, bike, car) noises. The music varied in loudness from imperceivable by humans to very loud. For the music detector we had to make a trade off between wanting a high recall (always triggering with music regions) and avoiding false positives. By accepting a false positive rate of roughly once every 20 minutes on non-silent audio, we were able to still maintain a recall of 75.5 %.

6.2 Energy consumption evaluation

On both the Pixel 2 and Pixel 2 XL devices, activating the DSP music detector resulted in a small increase of stand-by power consumption of under 0.4 mA. Further power is only consumed when the application processor is woken up to identify a song, with each wakeup costing around 2.0 J or 0.139 mAh which on average occurs about 100 times a day. In total this results in *Now Playing* consuming on average 23.5 mAh or about 0.9 % of the Pixel 2's 2700 mAh battery per day.

7 Conclusion

We presented a continuous music recognition system that uses a low-power music detector to only wake up the main application processor only when music is present, thereby keeping the

total daily power consumption under 1 % of the battery for an average user. Using our neural network fingerprinter we are able to generate compact and discriminative fingerprints at a rate of one 96 dimensional embedding per second, which means that each song occupies on average less than 3 KB in our on-device DB, alleviating the need to query a server and thereby preserving the user's privacy. The overall system we outlined allows us to detect, recognize and inform users which song is playing without them having to touch their Pixel 2 phone and without any data leaving the device.

Acknowledgments

The authors would like to thank Gabriel Taubman, Katsiaryna Naliuka, Chris Thornton, Jan Althaus, Brandon Barbello and Tom Hume, all with Google, for their help with the implementation of *Now Playing*. The authors would also like to thank David Petrou, Felix Yu and Corinna Cortes for their thoughtful comments on the paper.

References

[1] Soundhound. `http://soundhound.com`, .

[2] Google Sound Search. `http://support.google.com/googleplaymusic/answer/2913276?hl=en`, .

[3] S. Baluja and M. Covell. Audio fingerprinting: Combining computer vision & data stream processing. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 2, pages II–213. IEEE, 2007.

[4] V. Chandrasekhar, M. Sharifi, and D. A. Ross. Survey and evaluation of audio fingerprinting schemes for mobile query-by-example applications. In *ISMIR*, volume 20, pages 801–806, 2011.

[5] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[6] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

[7] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529, 1999.

[8] R. Guo, S. Kumar, K. Choromanski, and D. Simcha. Quantization based fast inner product search. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 482–490, 2016.

[9] J. Haitsma, T. Kalker, and J. Oostveen. Robust audio hashing for content identification. In *International Workshop on Content-Based Multimedia Indexing*, volume 4, pages 117–124. University of Brescia, 2001.

[10] H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[11] M. Malekesmaeili and R. K. Ward. A local fingerprinting approach for audio copy detection. *Signal Processing*, 98:308–321, 2014.

[12] C. Ouali, P. Dumouchel, and V. Gupta. A spectrogram-based audio fingerprinting system for content-based copy detection. *Multimedia Tools and Applications*, 75(15):9145–9165, 2016.

[13] C. Plapous, S.-A. Berrani, B. Besset, and J.-B. Rault. A low-complexity audio fingerprinting technique for embedded applications. *Multimedia Tools and Applications*, pages 1–20, 2017.

[14] D. Qin, C. Wengert, and L. V. Gool. Query adaptive similarity for large scale object retrieval. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1610–1617, June 2013. doi: 10.1109/CVPR.2013.211.

[15] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[16] T. Tsai, T. Praetzlich, and M. Muller. Known-artist live song identification using audio hashprints. *IEEE Transactions on Multimedia*, 2017.

[17] A. Wang. The shazam music recognition service. *Communications of the ACM*, 49(8):44–48, 2006.

[18] A. Wang et al. An industrial strength audio search algorithm. In *Ismir*, volume 2003, pages 7–13. Washington, DC, 2003.

[19] X. Wu, R. Guo, A. T. Suresh, D. Simcha, F. Yu, and S. Kumar. Multiscale quantization for fast similarity search. In *Advances in Neural Information Processing Systems*, 2017.