OUR 2016-2017 Report

Student

Ganesh S
gs401@snu.edu.in
Bachelors in Computer Science and Engineering (2013-2017)
Shiv Nadar University, India

Faculty Sponsor

N Sukumar
n.sukumar@snu.edu.in
Professor, Department of Chemistry
Director, Center for Informatics
Shiv Nadar University, India

Date of Submission

20 April 2017

Abstract

We implemented a solution to the problem of query by singing/humming (QBSH) in music information retrieval (recognizing a tune). After some preliminary exploration of the Wavelet Transform, we applied deep learning techniques to solve the problem. We created a deep neural network to do cover song recognition[1], a related task for which we had ample amounts of data. The network learns to embed sequences in a Euclidean space. Specifically, our feedforward attention-based convolutional neural network learned to produce 128-dimension embeddings using a triplet loss function. We then used the cover song network as a feature extractor for hummed queries. We then attempted to predict the song name by applying k Nearest Neighbours algorithm on the hummed queries. It was trained on the Second Hand Songs dataset, which contains 18,000 songs and covers. This model obtained a disappointing mean reciprocal rank of 0.1277 on the Roger Jang QBSH Dataset. Detailed analysis of the results is ongoing.

Introduction

Our problem was the task of identifying from audio the song that someone was singing/humming. We proposed a solution in the beginning: obtaining a sequence of Wavelet-Transform coefficients from the audio signal and then comparing a sequence of such ratios against sequences obtained from other audio signals using a string distance metric (Levenshtein distance). We experimented with the Wavelet transform to obtain a coefficient sequence and were able to also invert the transformed sequence to obtain the original audio signal. We needed to compress the transformed signal i.e., discard most of the “detail” coefficients to obtain strings of manageable length (strings which could be compared easily).

At this point, we began investigating deep learning techniques for solving this problem. Deep learning is an emerging paradigm within the discipline of machine learning. Machine learning deals with making computers do tasks without explicitly programming them to do so. In the traditional paradigm, a domain expert (say, biologist) looks at medical images and defines relevant features. For example, size of the tumor, color, shape, etc. And, then, a machine learning algorithm performs regression or classification for the given input based on its numerical features. For example: predicting whether the tumor is benign or malignant.

In deep learning, neural networks with many layers are trained on the raw input data, not requiring feature engineering performed by a domain expert. The neural network layers learn representations of the input data and learns to compose these representations in the later layers, thus learning more and more abstract representations as the depth increases. Deep learning models have achieved near-human-level performance for problems in several domains such as computer vision, speech recognition, recreational games, and also our domain of interest, music information retrieval.

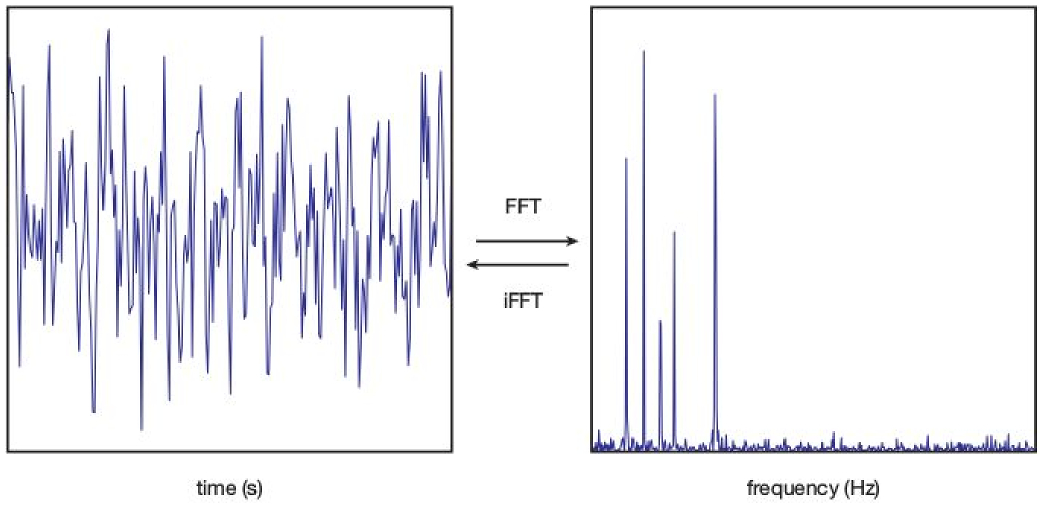
[1] A cover song is a new performance/recording of a previous recording by a different artist/composer.

Our deep learning solution is the following: we first train a deep neural network to solve the related task of cover song recognition, for which there is relatively more data available. We can then use the same network to help us in solving our ultimate task of query by singing/humming. This reuse of learned model parameters is called transfer learning, and is common when there is a paucity of data (deep networks require lots of labelled training data).

Main body

1. Original proposal: Wavelet Transform on the audio signal

To understand the Wavelet Transform, we must first understand the Fourier Transform. The Fourier Transform decomposes any signal into sines and cosines. It converts a time-domain signal into a frequency-domain signal. The key assumption in Fourier analysis is that a signal can be considered to be composed of many sinusoidal components, each having a specific frequency.



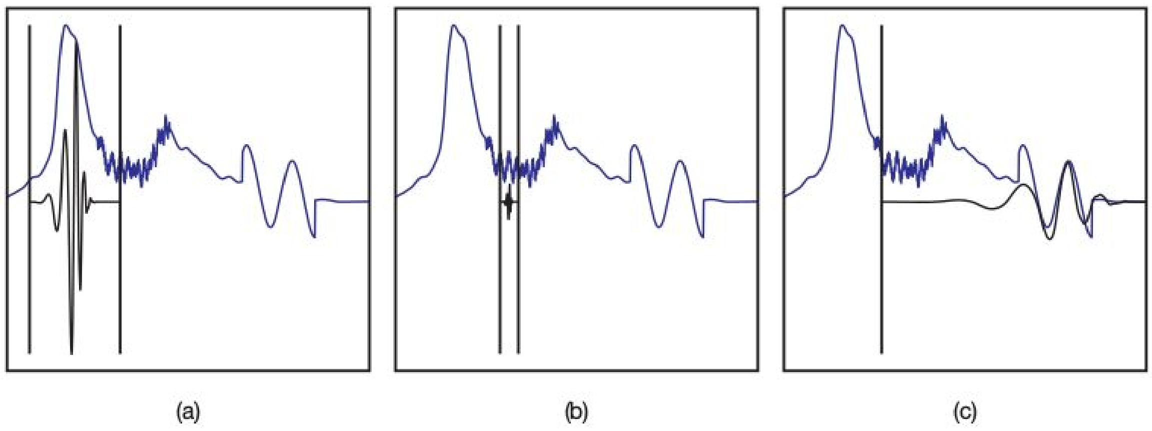
Example of the Fourier Transform [Adapted with permission from ref.1]

When we perform a Fourier Transform, we cannot get both time and frequency domain information simultaneously. Sine the basis functions are sinusoidal, a single component with frequency ω will affect the entire domain of the signal *f(t)*. This makes the Fourier Transform not entirely suitable for analyzing non-stationary (non-periodic) functions. Isolation of a given frequency component to a finite-time region is necessary when dealing with localized signal features (eg., a sharp spike) which are by nature non-stationary. The Fourier basis function lacks the ability to distinguish or isolate time-domain information. The Short Term Fourier Transform overcomes this hindrance by dividing the non-stationary signal into small windows in an attempt to achieve a locally stationary signal. The Fourier Transform is performed in each window to obtain the local frequency composition of the signal. The *window length* selection, then, becomes crucial to ensure that the local signal is sufficiently stationary. Narrowing it improves location resolution, but reduces the clarity of the frequency information.

The Wavelet Transform [1][2] uses basis functions that are simultaneously localized in both the time and the frequency domains to avoid the tradeoff. The basis function, a wavelet, should satisfy two conditions:

1. It has a small concentrated finite burst of energy in the time domain. This makes the wavelet “little” in the sense that it is well-localized in the time domain.
2. It exhibits oscillation in time. This makes it periodic, giving it some wave-like character.

The transform uses the wavelet to convert a signal from one domain to another. It breaks apart a signal using scaled and translated versions of the basis function. The advantage is that it effectively localizes information in both time and frequency domains simultaneously. This makes it ideal for analyzing non-stationary signals such as molecular property distributions, or, in our case, music.



[Shown above are three wavelet windows over a non-stationary signal that illustrate a partial wavelet scaling analysis; the wavelet function simultaneously identifies component frequency and location within the signal. Adapted with permission from ref.1]

We were able to apply the Discrete Wavelet Transform using Multi-resolution analysis scheme to obtain coefficients for an audio signal. We were also able to obtain the original signal when we applied the Inverse transform.

The software library we were using, PyWavelets, generated representation that was as long as the time domain signal. We applied it to the audio signal of some songs and obtained an equally long coefficient series. The DWT is reversible so we were able to obtain the exact same audio signal when we applied the Inverse Wavelet Transform to the coefficients. We obtained a sequence of coefficients equal to the length of the audio signal. In order for an edit distance measure to effectively separate them, we needed to compress the signal. It was possible to do this by eliminating most of the detail coefficients.

2. Deep Learning

This two-stage architecture of our proposed QBSH system (feature extraction and then pattern recognition) is the dominant paradigm in content-based music informatics. In fact, this is the dominant paradigm in *every* domain where ML algorithms are applied. Machine learning has traditionally relied greatly on domain expertise to obtain valuable features/descriptors, which are then supplied to a pattern recognition machine. In deep learning, these features are learned automatically from the raw data.

For MIR, E J Humphrey, et al. criticized this paradigm [3]. Specifically, they identified three issues: hand-crafted feature design is sub-optimal and unsustainable (better features than, say, mel-frequency cepstrum coefficients exist for speech recognition, the very task they were designed for), the power of shallow architectures is fundamentally limited, and short-term analysis cannot encode musically meaningful structure (musical events occur in the range of at least seconds and minutes and not milliseconds). The authors offered a conceptual argument for both *learning* and *depth*.

These arguments prompted us to study the advances made by deep learning techniques in the performance of artificial neural networks. The deep learning paradigm is characterized by neural networks that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction [4]. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition , machine translation and many other domains. Deep learning discovers intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional nets have brought about breakthroughs in processing images[5], video, speech and audio, whereas recurrent nets have shed light on sequential data such as text and speech.

Simple Artificial Neural networks are capable of learning functions from data: functions for classification or regression [6]. They do this by performing affine transformations [7] on the input data.

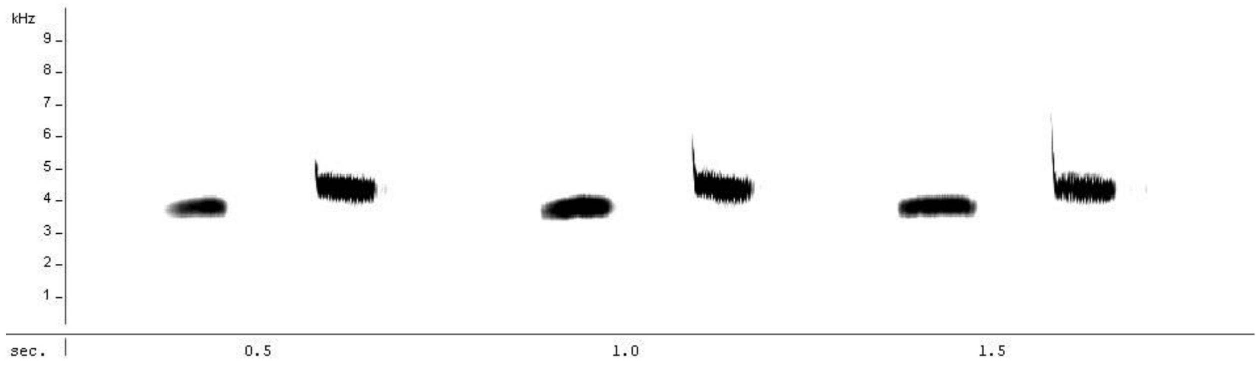
A convolutional neural network is a particular kind of neural network. It contains filters that exploit the grid nature of image/audio/text data. The filters learn weights that activate when a feature of interest is present in the input data, even if it translated.

3. Strategies for Applying Deep Learning to Music Data

In almost all learning-based applications for audio data, the raw audio signal is first converted to Fourier spectrograms. This mid-level representation is then supplied to the model as input.

Convolutional networks for audio spectrograms

A spectrogram can be likened to an image of a sound, and can be highly characteristic of the audio source (e.g., bird songs spectrograms are highly characteristic to the species in question).



[Spectrogram of the bird song of *Parus major*. Creative Commons license. [2]]

Since convolutional networks have produced state-of-the-art results for image data tasks, many deep learning implementations for audio data also use spectrogram as input, instead of the raw audio signal. One example is ref. [8] where the authors successfully train a convnet to predict latent representations for audio spectrograms. The ground truth was obtained from collaborative filtering based recommendation systems[3].

Embeddings

Our problem of interest, querying a song by humming/singing, *can* be formalized as a supervised learning classification problem. But, it would be difficult to scale such a system to many songs for the following reason: when classifying with neural nets, the network has to output vectors approximately equal to the labels, which are represented as one-hot encoding vectors[4]. The length of the one-hot encoding vector would grow linearly with the number of song cliques.

There is another way to train a convnet to recognize cover songs. A convnet can be trained to learn continuous-valued fixed-length vector representations for audio sequences. Such representations are called *embeddings*. The embedding space is amenable to learning and comparison because many simple learning algorithms, such as nearest neighbors and k-means clustering, are particularly effective given data where Euclidean distance corresponds to some notion of similarity. When a new audio input is given, identifying the (clique) label of the song is simple: supply it to the network to get an embedding, find the label of the ***k* nearest neighbors** in the embedding space.

Neural network architectures are flexible enough that one can define loss functions that adjust

[2] Image distributed under a Creative Commons 3.0 License by Maxime Metzmacher. 2011.

[3] Collaborative filtering models learn latent representations for songs by using information gleaned from many people's listening preferences.

[4] For a two class problem, the one-hot encoding vector labels for the two classes would be [1,0] and [0,1] respectively.

network parameters to produce similar embeddings for semantically similar inputs, and dissimilar embeddings for semantically dissimilar inputs.

Triplet loss function

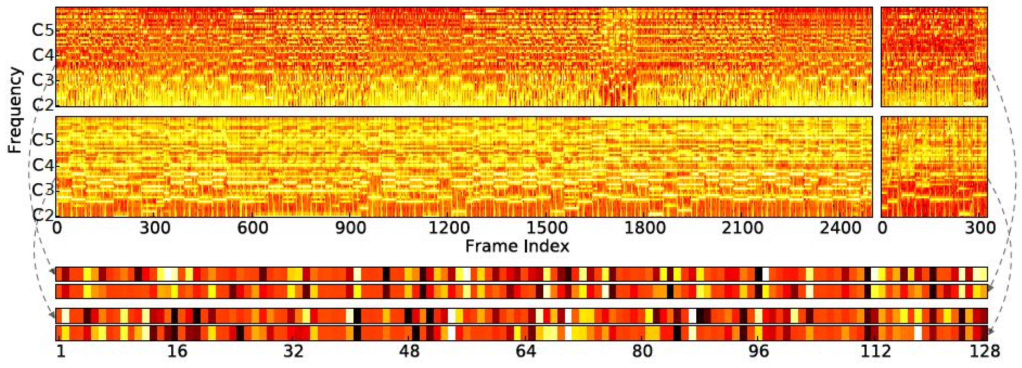
To give a concrete example, this sort of loss function, called triplet loss, was implemented in [9] to embed images of faces such that images of the same person have small pairwise distances in the embedded space and images of different people have large distances (see figure below). With this model, the authors were able to achieve state-of-the-art results in labelling faces.

The triplet loss function takes the following form:

$$\mathcal{L} = \frac{1}{|\mathcal{P}|} \sum_{(x,y)\in\mathcal{P}} \|f(x) - f(y)\|_2^2 + \frac{\alpha}{|\mathcal{N}|} \sum_{(a,b)\in\mathcal{N}} \max(0, m - \|f(a) - f(b)\|_2)^2$$

Where P and N are collections of matching and non-matching pairs of audio spectrograms. f represents the learned embedding function. α is a hyperparameter that defines the non-matching pair distance threshold. m is another hyperparameter that defines the importance of the non-matching pair distance.

Embeddings have also been used for audio data. In [10, 11], Raffel trained a multi-modal architecture of two networks (one for MIDIs and the other for MP3s since the statistics of these two data forms are different) trained jointly to produce similar embeddings for matching MP3 and MIDI sequences. The authors used this successfully to match MP3 clips from the Million Songs Dataset [12] with a very large collection of MIDI files (figure below; taken from their paper).



[Embeddings computed by a deep convolutional network from spectrograms of audio data. Adapted with permission from ref. [11]]

Attention mechanism

Audio input sequences can be of variable length. Sometimes, they can be very long and the musical motif of interest is present only in a particular region of the input (other regions can be safely ignored). This issue was prominent in recurrent neural network when trying to model long-term dependencies. When learning embeddings, this issue may result in the end of the sequence having greater impact on the embedding than the beginning. A technique for mitigating this issue has been dubbed the "attention mechanism".

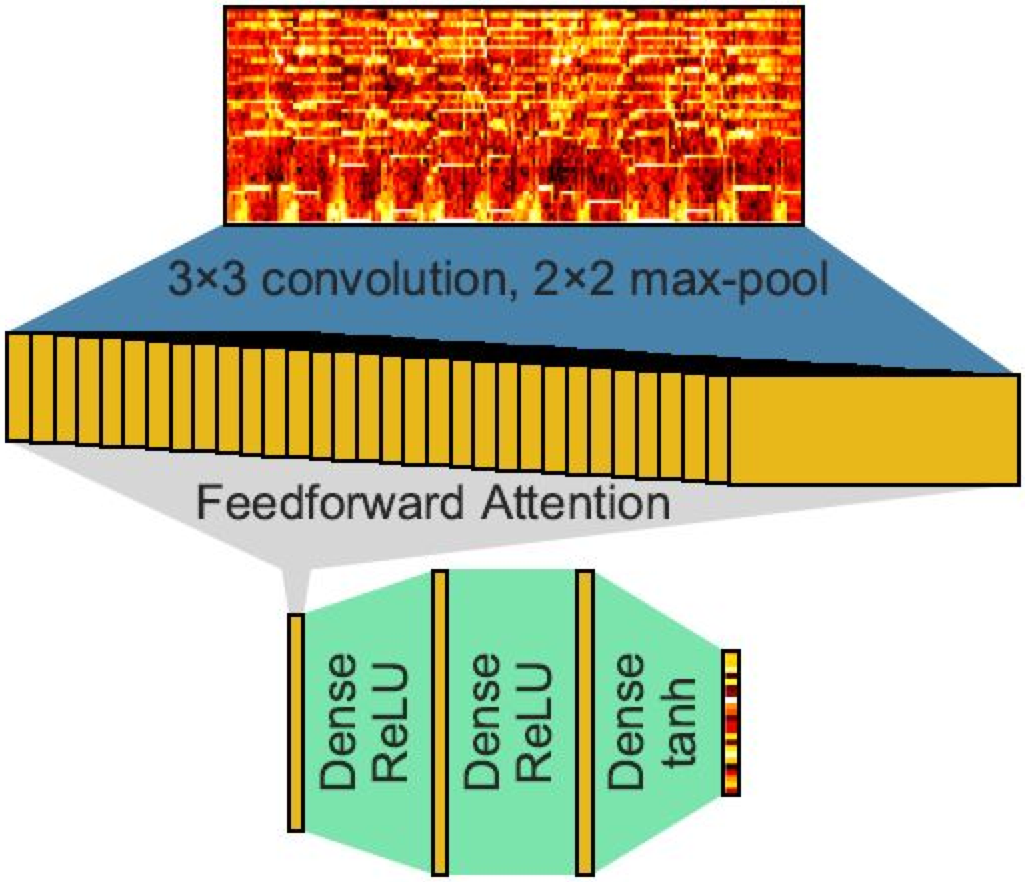
Essentially, it involves learning weights to each sequence step based on the current and previous states. When used with recurrent networks, the addition of attention has proven effective in a wide variety of domains, including machine translation, speech recognition, and image captioning. We use a feedforward attention mechanism proposed by Raffel et al. [10]

$$e[t] = a(h[t])$$
$$\omega[t] = \frac{\exp(e[t])}{\sum_{k=1}^{T} \exp(e[k])}$$
$$c = \sum_{t=1}^{T} \omega[t] h[t]$$

Conceptually, this attention mechanism can be thought of as using the learned nonlinear function *a* to compute scalar importance values for each sequence step in *h*, which are then normalized using the softmax function and used to compute a weighted mean *c* over all sequence elements in *h*. This mechanism produces a fixed length vector which is a necessary input constraint for the later fully-connected layers of the network.

Transfer learning:
Deep learning models perform well when there is a large amount of labelled data. Unfortunately, for our query by singing/humming task we have a small dataset. This is common is several domains. For niche tasks with very little labeled data, researchers use a pre-trained model as a feature extractor and run a classifier on top of that dataset. For example, in computer vision, a convnet is trained on the ImageNet dataset (which contains 1.2 million images with 1000 categories), and then use the convnet either as an initialization or a fixed feature extractor for the task of interest.

4. Implementation Details



[Illustration of our attention-based feedforward convolutional network. Adapted with permission from ref. [11]]

We implemented an attention-based feedforward convolutional neural network. It contains three 2D convolutional layers and one feedforward attention layer. The network had the following architecture:

INPUT => STANDARDIZE => CONV => RELU => POOL => CONV => RELU => POOL => CONV => RELU => => ATTENTION => POOL => FC => FC => OUTPUT

The network outputs an embedding in a 128-dimension space. The network was first trained to do cover song recognition by training it on 18,196 MP3 preview clips from the Second Hand Songs Dataset (a subset of the Million Songs Dataset). These 18,196 songs are organized into 5,854 cliques. The network was trained to produce similar embeddings for all songs from the same cover song clique and dissimilar embeddings for songs from different cliques. This was done using the a triplet loss function.

We implemented the network by building on top of Colin Raffel's open-source implementation of a pairwise sequence embedding network. The code was written in Python and used the Lasagne deep learning library and the Theano numerical computation library. The source code for our work is available on Github [13].

We used the same learning rate as the pairwise sequence embedding network: 9.72e-5.

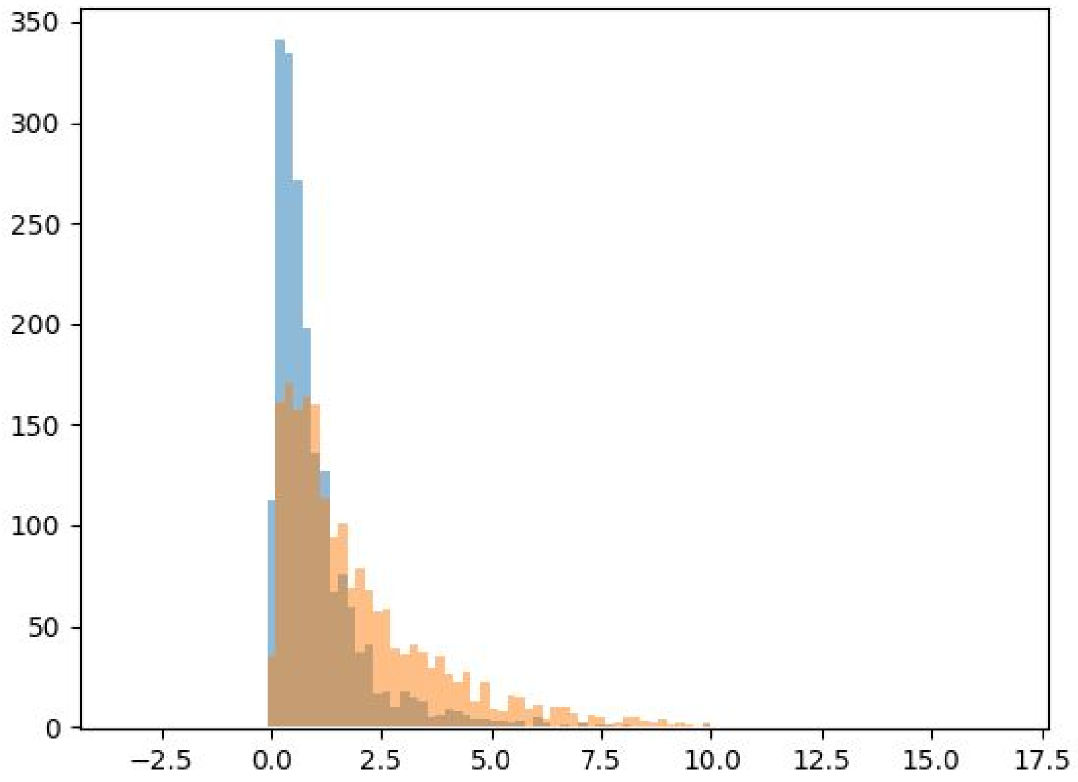
5. Results

Results on cover song dataset:

We ran our code for 112 epochs on g2.2xlarge GPU machines available as AWS EC2 instances. Actually, the code was run for 1000 epochs but it only saved that epoch's weights where it achieved the least loss on a validation set (5% of the whole dataset): a form of early stopping. The least validation set loss we obtained was 7.095.

We obtained slightly distinct distance distributions (Bhattacharyya coefficient=0.801) for matching pairs (songs and their covers) and non-matching pairs (unrelated songs) obtained from the Second Hands Song dataset. The non-matching pairs distribution had a fatter tail and a higher mean distance (0.65 versus 0.42 for the matching pairs). However, the separation between the two distributions is statistically significant (p-value<0.01)[5]. The overlap is shown in brown.

[5] We obtained a p-value of 2.56×10^{-113} when we performed a two-sided Welch's t-test for the two distance distributions. The null hypothesis was that the two distributions had identical average (expected) values.



[Distance distribution for unseen matching pairs (songs and their covers; shown in blue) and non-matching pairs (unrelated songs; shown in orange).]

We attempted changing the loss function to see if a better validation loss would achieved: we made the loss inversely proportional to the non-matching pair distance. The resulting network performed worse during training. It achieved a validation loss of 32.4752 (at epoch 164) when it was trained for a 1000 epochs.

Results on QBSH dataset:

We then tried using the cover song embedding network as a feature extractor for our humming data. The dataset we used was the Roger Jang QBSH corpus [14]. It contains 4431 hummed queries for 48 unique songs. We transformed these audio queries into 128-dimension embeddings. We used 67% of the embeddings as training data for a k Nearest Neighbors classifier (k=100).

We obtained a mean reciprocal rank of 0.1277, which is much inferior to state of the art results for this task (>0.90)[15].

One possible reason transfer learning did not work could have been this: the source task data and transfer task data were very different from each other. The source data (cover song MP3s) contained instrument sounds whereas the target data contained only vocal sounds. In follow up work, we intend to repeat the procedure using training and test examples drawn from the same distribution.

6. Nomenclature

Definitions of some of the commonly used terms in this report are provided below.

1. **Bhattacharyya coefficient:** A measure of the approximate overlap between two statistical samples. For two samples, *p* and *q*:

$$BC(\mathbf{p},\mathbf{q}) = \sum_{i=1}^{n} \sqrt{p_i q_i}$$

   The number of bins/partitions (n) chosen affects the value of the coefficient. Too many bins will cause a drop in accuracy by creating too many partitions without any members despite being in a densely-populated sample space. Too few bins will reduce accuracy by overestimating the overlap region. For our report, we have chosen the number of bins to be half of the number of number of a samples from one set (n=979).

2. **Learning rate** : A neural network updates its parameters, W, in the direction of the steepest descent of the loss, ∇L. The learning rate determines the magnitude of these updates.

$$W := W - \eta \nabla L$$

3. **ReLU:** A Linear rectifier unit. It is an activation function in a neural network. It is defined as *f(x) = max(0, x)* where *x* is the input to the neuron. It is analogous to the half-wave rectifier in electrical engineering. Networks containing linear rectifiers perform better compared to networks containing other activation functions (sigmoid function, hyperbolic tangent function, etc.).

4. **Max Pool**: This layer in a neural network performs a sample-based discretization on the input data. It effectively downsamples the input data by performing the *max* operation on its non-overlapping subregions. It serves to reduce the number of parameters in the network and hence avoids overfitting.
5. **Epochs**: The number of times a neural network algorithm sees the entire training data.

6. **K Nearest Neighbors Classifier**: A non-parametric learning algorithm. The label assigned to a point, represented as a feature vector, is the most frequent label for the k-closest training data points, also represented as feature vectors.

7. **Mean Reciprocal Rank** is an evaluation metric for any information retrieval system that produces a list of possible responses for a given query, ordered by their estimated probability of correctness. For a single query, the reciprocal rank is the reciprocal of the rank of the first correct response. The mean reciprocal rank is average of reciprocal ranks for a set of sample queries. It is used widely in evaluation of Music Information Retrieval systems.

7. Acknowledgments

I am very grateful to my faculty sponsor, Dr. N Sukumar, for all the hours he spent on discussions about the project. I am very thankful for his support and for being patient with my tendency to digress a lot. Dr. V K Jayaraman from the Center for Informatics also provided an immense amount of mentoring and support.

I would also like to thank Dr. Colin Raffel[6], a resident at Google Brain, for helping me in applying ideas from his PhD thesis. He also helped me in reusing his project source code.

8. References

1. Sundling, C. Matthew, Nagamani Sukumar, Hongmei Zhang, Mark J. Embrechts, and Curt M. Breneman. Wavelets in Chemistry and Chemoinformatics. Reviews in Computational Chemistry 22 (2006): 295.
2. Robi Polikar. The Wavelet Tutorial. Rowan University. 1996.
3. Eric J Humphrey, Juan P Bello, Yann LeCun. Feature Learning and Deep Architectures: New Directions for Content-Based Music Informatics. Journal of Intelligent Information Systems 2013. 41 (3), 461–481. 2013.
4. Yann LeCun, Yoshua Bengio, Geoffrey Bengio. Deep Learning. Nature. 521, 436–444. 28 May 2015.
5. Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton. ImageNet Classification with Deep Convolutional Neural Networks. (NIPS). 2012.
6. Michael A. Nielsen. Neural Networks and Deep Learning. Determination Press. 2015.
7. Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep Learning. MIT Press. 2016.
8. Van den Oord, Aaron, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. Advances in neural information processing systems. 2013.
9. Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. Proceedings of the IEEE Conference on Computer

[6] https://scholar.google.co.in/citations?user=I66ZBYwAAAAJ&hl=en

Vision and Pattern Recognition. 2015.
10. Colin Raffel and Daniel PW Ellis. Pruning subsequence search with attention-based embedding. Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on. IEEE, 2016.
11. Colin Raffel. Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching. Diss. COLUMBIA UNIVERSITY, 2016.
12. Thierry Bertin-Mahieux, et al. The Million Song Dataset. ISMIR. Vol. 2. No. 9. 2011.
13. Ganesh Srinivas. Deep Sing: a Deep Network for Query by Singing/Humming. Source code. 2017.
14. Roger Jang. Music Information Retrieval Query by Singing/Humming (MIR-QBSH) Dataset. National Taiwan University. 2009.
15. J. Stephen Downie. MIREX 2016 Evaluation Results. Online document. 2016.