Learning Embeddings for Laughter Categorization

Ganesh Srinivas - Bachelors in Computer Science and Engineering (Senior Year) - Shiv Nadar University, India

email - github/bitbucket repos - homepage - resume

Abstract

I propose to train a deep neural network to discriminate between various kinds of laughter (giggle, snicker, etc.) A convolutional neural network can be trained to produce continuous-valued vector representations (embeddings) for spectrograms of audio data. A triplet-loss function during training can constrain the network to learn an embedding space where Euclidean distance corresponds to acoustic similarity. In such a space, algorithms like k-Nearest Neighbors can be used for classification. The network weights can be visualized to glean insight about the low- and high-level features it has learned to look for (pitch, timbre, unknowns, etc.) I also propose to obtain visualizations of the embedding space of laughter sounds using dimension reduction techniques like Principal Components Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE). I will also apply these same visualization techniques *directly* on the high-dimension audio spectrograms. All techniques proposed here have been applied previously on related problems in audio and image processing.

Background and Rationale for Techniques Proposed

Laughter is a universally produced vocal signal that plays an important role in human social interaction[1]. Empirical work has explored some of the perceptual and acoustic aspects of laughter. Deep learning models, currently the state-of-the-art in speech, music, and other machine perception domains, can perhaps contribute to enriching our understanding of laughter.

Deep learning refers to a class of machine learning algorithms that learn multiple levels of representations that correspond to different levels of abstraction. Deep *convolutional* neural networks perform remarkably in detecting patterns in image data. They can be trained to learn continuous-valued fixed-length vector representations for audio sequences when the audio is supplied in the form of Fourier spectrograms. Such representations are called *embeddings*.

---

[1] Bryant, Gregory A., and C. Athena Aktipis. "The animal nature of spontaneous human laughter." Evolution and Human Behavior 35.4 (2014): 327-335.

Neural network architectures are flexible enough that one can define loss functions that adjust network parameters to produce similar embeddings for semantically similar inputs, and dissimilar embeddings for semantically dissimilar inputs.

To give a concrete example, this sort of loss function, called triplet loss, was implemented in (Schroff et al., 2015)[2] 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 taken from paper). With this model, the authors were able to achieve state-of-the-art results in labelling faces.

Figure 2. **Model structure.** Our network consists of a batch input layer and a deep CNN followed by L_2 normalization, which results in the face embedding. This is followed by the triplet loss during training.

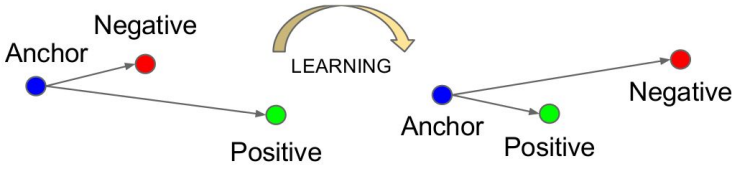


Figure 3. The **Triplet Loss** minimizes the distance between an *anchor* and a *positive*, both of which have the same identity, and maximizes the distance between the *anchor* and a *negative* of a different identity.

Embeddings have also been used for audio data. (Raffel and Ellis 2016)[3] 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 with a very large collection of MIDI files (figure below; taken from their paper).

[2] Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015..
[3] Raffel, Colin, and Daniel PW Ellis. "Pruning subsequence search with attention-based embedding." Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on. IEEE, 2016.

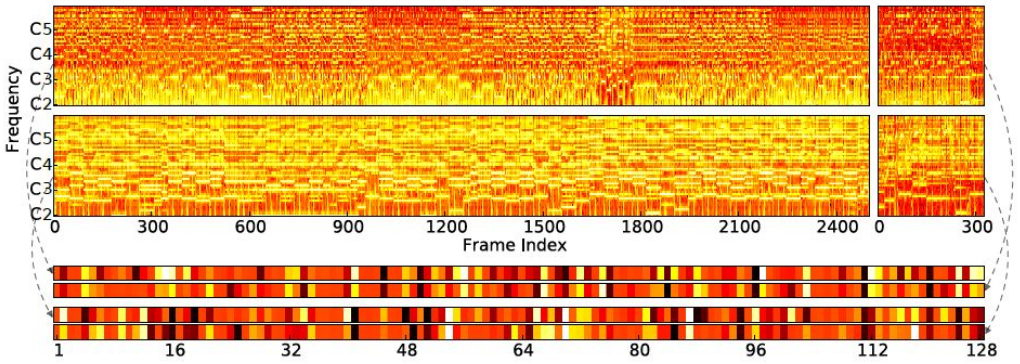


Fig. 3. Embeddings for two pairs of matching sequences from our best-performing system. The first two rows display pairs of matching constant-Q spectrograms. For both pairs, the right sequence is a subsequence of the left. The last two rows show their resulting embeddings. Dashed arrows denote which embedding corresponds to which sequence.

I am interested in learning such embeddings for variable length audio segments containing various kinds of laughter. A variant of the attention mechanism is capable of dealing with variable length sequences: essentially it teaches the network to focus on parts of the entire input - a weighted average over the input where the weights are learned during the training.

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 category of laughter is simple: supply it to the network to get an embedding, find the label of the ***k* nearest neighbors** in the embedding space.

Once the network has been trained, I will also **visualize/play the sound that the network filters have learned to activate maximally for**. This is inspired by Sander Dieleman's work, who obtained music playlists that maximally activated various CNN filters for a model trained to produce content-based music recommendations. The filters were found to look for the sound inventory of various styles of music[4].

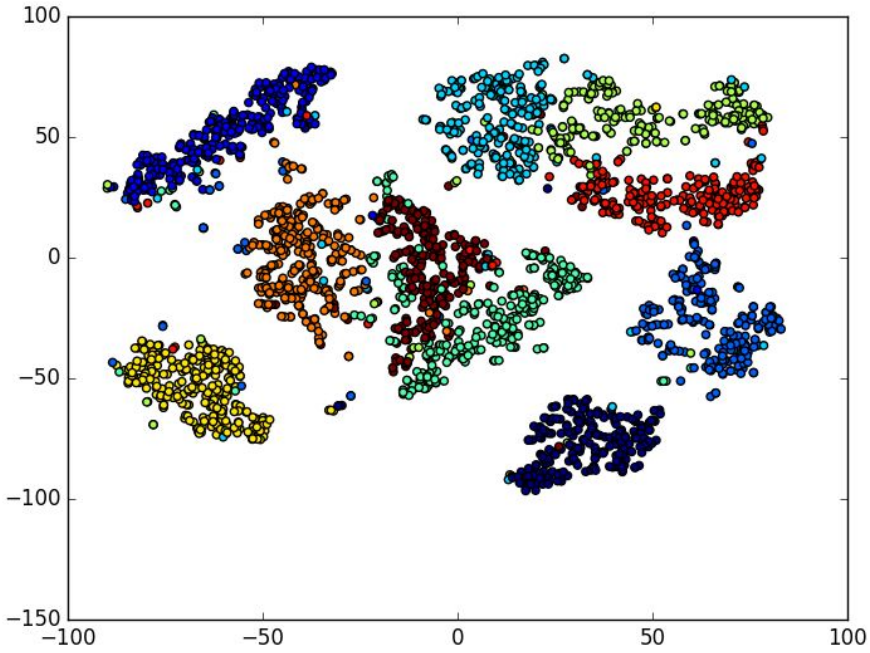
Once embeddings have been produced for a large number of audio clips, I will visualize them using **dimension reduction techniques such as PCA and t-SNE**.

My motivation comes from successful applications of t-SNE to produce meaningful visualizations of many kinds of data: music embeddings, word embeddings[5].

I will also trim a large number of audio spectrograms of laughter events to a fixed size and apply dimension reduction directly on them. My motivation of this comes from such an approach having yielded good results on visualizing the space of bird songs[6], and even visualizing 28x28

[4] http://benanne.github.io/2014/08/05/spotify-cnns.html
[5] http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/
[6] https://aiexperiments.withgoogle.com/bird-sounds

(784 dimension space!) images of handwritten digits in two dimensions. Just applying PCA produces a mush, but PCA+t-SNE produces natural clusters in the data that correspond to the various digits! (figure below; my own implementation)



Implementation Details

Learning embeddings for various kinds of laughter: The recently released Google Audio Set has a Laughter ontology[7] i.e., annotations of five kinds of laughter : giggle, snicker, baby laughter, belly laugh and chuckle/chortle. An attention-based feedforward convolutional network can be trained on this labelled data to produce fixed-length embeddings. The attention-mechanism can be described as computing a weighted average over the input whose weights are itself learnt while training. A triplet loss function will encourage it to produce similar embeddings (Euclidean distance-wise) for intra-class audio sequences and dissimilar embeddings for inter-class audio sequences. I already have code for doing this (in Lasagne-Theano syntax, though, not yet in Keras/TensorFlow).

The Red Hen Rapid Annotator[8] can be used to extract clips from its datasets that contain laughter. The trained model can now be used to generate embeddings for Red Hen clips.

Since we have access to video data as well, the system can be made multi-modal in the future. I have a vague idea of something that might work, perhaps a project for a future student: attach an

[7] https://research.google.com/audioset/ontology/laughter_1.html
[8] http://vrnewsscape.ucla.edu/elan-clips/Laughter

object detection network that also produces an embedding for laughter categorization. It could take as input a few of frames of close-up shots of the person's face.

Examining the learned model and embeddings space: I think it means finding the inputs that produce maximal activations for a particular filter (a convolutional layer filter). I can learn it in a day or two.

Dimension reduction using PCA and/or t-SNE: The inventor of t-SNE has published code in Python and CUDA. There is also an implementation available in Scikit-Learn.[9] I've used both, but only on toy datasets. Time will go into hyperparameter tuning[10]: perplexity and learning rate.

Timeline

I have a working knowledge of Python and the following ML/deep learning libraries: TensorFlow, Keras, Lasagne, Theano and Scikit-Learn.

Week 1: Assemble training, validation and test data from the Google Audio Set dataset's Laughter Ontology.

Deliverable: Annotated audio segments for five categories of laughter.

Week 2: Refining the Red Hen's preliminary training set[11] for laughter detection task. It can be done using the Rapid Annotator to clip out the non-laughter in two to three days.

Deliverable: Audio clips from Red Hen's dataset.

Week 3: Translate my code for an triplet-loss CNN written in Lasagne-Theano to Keras/TensorFlow syntax.

Deliverable: Tensorflow implementation of laughter embedding network. Specifically, a feedforward attention-based convolutional network that produces 128-dimension embeddings using a triplet-loss function.

Week 4: Train one or two implementation networks (with different hyperparameters) and get results.

Deliverable: Test set performance (precision, recall, etc.) after 100 epochs of training or until overfitting, whichever occurs earlier.

Week 5: Buffer week. In case of poor results, perform hyper-parameter search to find the best network hyperparameters (number of layers, learning rate, embedding dimension, etc.)
Deliverable: Test set performance (precision, recall, etc.) for various combinations of hyperparameters: layers (convolutional filter sizes, ReLU, dropout, etc.), optimization algorithm

---

[9] `from sklearn.manifold import tsne`
[10] Wattenberg, et al., "How to Use t-SNE Effectively", Distill, 2016. http://doi.org/10.23915/distill.00002
[11] http://vrnewsscape.ucla.edu/elan-clips/Laughter/

(SGD, Adam, RMSProp, etc.), learning rate, embedding dimension (128, 512, etc.)

Week 6: Examining the learned model.

Deliverable: In an interactive Python notebook, visualize and play the low- and high-level features that maximally activate various filters in the network. Specifically, a playlist of audio snippets that maximally activate every filter must be prepared (two examples).

Week 7: Implement PCA and/or t-SNE dimension reduction code for visualizing embeddings.

Deliverable: A script that takes 128-dimension audio embeddings as input and reduces dimensionality to 2D/3D using PCA/t-SNE using Scikit-Learn's/van der Maaten's implementation.

Week 8: Visualizing laughter embeddings in 2D/3D space to **look for insights**. Do spontaneous and volitional laughter form separate clusters? Etc. If perceivably distinct recordings of spontaneous and volitional laughter aren't available, use the small but reliable (since it is ground truth) recordings from the supplementary section of Bryant and Aktipis' paper.[12]

Deliverable: Pictures/GIFs of embedding space for various hyperparameter combinations. (two examples)

Week 9: Report of visualizations and insights gleaned, if any.

Deliverable: 2-page writeup of discussion and comments from mentors.

Week 10: Documenting results and refactoring code developed so far.

Deliverable: An IPython notebook that allows the user to reproduce the results and analysis (example)

Week 11: Writing report and wrapping up things.

Deliverable: A five-page document giving explanations for chosen approach and reporting the results of experiments and analyses.

Student Information

I'm a senior year undergraduate student majoring in Computer Science and Engineering from Shiv Nadar University, near New Delhi, India. I have over three years of experience with machine learning: internships, coursework, tutorials taught, undergraduate research, Udacity ML nanodegree (in progress). I am currently implementing a solution to the very cool problem of query by singing/humming (what tune is that?) problem in music information retrieval. It involves[13] deep neural networks, attention-mechanism, transfer learning and embeddings.

---

[12] Bryant, Gregory A., and C. Athena Aktipis. "The animal nature of spontaneous human laughter." Evolution and Human Behavior 35.4 (2014): 327-335.

[13] https://github.com/ganesh-srinivas/deep-sing