Predicting Visuals from Sound

Lingzhi Zhang[a,1], **Mia Chiquier**[a,1], **and Jianbo Shi**[a,1]

[a]University of Pennsylvania, Department of Computer Science

This manuscript was compiled on January 19, 2022

The doppler effect and interaural level difference are two sound cues humans use to localize rapidly moving sound sources. Given an initial frame and 3 seconds of sound, our multi-modal deep convolutional neural network is able to predict the speed and consequently location of the car 3 seconds later.

Embeddings also have the benefit that they are often of lower dimensionality than the original data. Put simply, each value of such a vector could describe higher level factors of a spectrum such as vowel, harshness or harmonicity. Autoencoders have multifaceted purpose – they can be used to compress data through dimensionality reduction, reveal underlying salient data features, and be sampled from to generate new data. They are usually built upon layers of convolutional neural networks, but it is their unique metastructure that results in the aforementioned characteristics.

There are roughly three components to the orthodox autoencoder: the encoder, the bottleneck, or latent vector, and the decoder. The auto-encoder looks for a compressed encoding such that when that embedding is passed into a seperate network called a "decoder", the decoder is able to reconstruct the image from just the embedding. The encoder is typically composed of some convolutional layers (and perhaps other layers too) that down-sample and encode the embedding. The decoder acts like a dilated filter and expands the input by padding every other value as zero, then runs a transpose convolution on it, which up-samples the input back to its original dimensions. Input data of high dimensionality is passed into the encoder and morphed into a lower dimensional latent vector which retains key qualities or patterns in the original data, and then the autoencoder attempts to regenerate the original content in the decoding phase. Thus, the autoencoder is trained to minimize reconstruction error between the input and the output. Since calculating this reconstruction error does not required a labelled dataset, the autoencoder qualifies as a semi-supervised learning algorithm. The bottleneck of the autoencoder with dimensionality compression roughly mimics the human cochlea's methods of frequency binning along its basilar membrane: the physiological constraints of the human auditory system are like the dimensionality reduction of the autoencoder.

The reason for why the NSynth dataset is particularly useful to use is because it allows us to encode timbre. The way this is done is through the equation P(audio)=P(timbre)P(note) where we know P(note) from a recurrent neural network that is able to predict the next note from a sequence of previous notes,and we know P(audio) from the data. P(timbre) in reality is P(audio|note) so the equation is really P(audio) = P(audio|note)P(note). However, we realized that this equation isn't the full picture because the way the NSynth model actually worked was by predicting timbre from the expectation of the note. Therefore $P(audio) = \sum P(audio|note_i * P(note_i)$ where i covers all the notes. The $P(note_i)$ does still come from the recurrent neural model and depends on the notes that prefaced the current note in the sequence, and the total distribution of notes. This is similar to the way that synthesizers work, a mixture of timbre and a particular note gives sound (a middle c on the piano is the note, and the quality of the instrument is the timbre). This works especially well for piano because piano's timbre quality does not depend on the note, but this isn't the case for all instruments.

What is timbre? The sum of waves is a wave in itself - as we know from the Fourier transform when a sum of different frequencies gives rise to

Members contributed equally to the publication

No conflicts of interest to declare

a different wave. All instruments that play the same note will share the same "fundamental frequency". So in the fourier decomposition, the frequency that has the highest power will be the same for all instruments that play the note C. Any additional frequency waves that are summed with the fundamental frequency are called the harmonics. Depending on what harmonics are added, the timbre changes. Therefore, P(audio|note), so P(timbre) is really P(summed wave|fundamental wave).

The way the NSynth architecture is built is to have an encoder figure out the embedding that represents P(audio|note), so timbre, also called the encoding vector. Our project is aimed at taking this encoding vector and seeing how well a support vector machine (SVM), a simple classifier, can classify the instrument based on this encoding vector. In NSynth, this embedding vector gets passed into a decoder architecture that reconstructs the original sound from the encoding vector - that is more complicated but it also makes use of the information of timbre to reconstruct the sound, so clearly timbre is encoded in that embedding vector.

The reason we are doing this is to better understand the latent space of the generative autoencoder and to convince ourselves that the information in the embedding truly is what we think it is - that is encodes timbre. Since timbre is so closely linked to the type of instrument the note was played on, it is easy to see if timbre is encoded in the embedding by checking if classification of the instrument is doable from the encoding vector.

Materials and Methods

To understand the NSynth autoencoder, we implemented a three step pipeline. We first created a dataset using over 1500 samples of musical notes, and generated their lower dimensional embeddings by feeding them into the NSynth encoding segment. We then wanted to see if there was enough information about timbre retained in the embeddings to actually classify the instrument, so we trained artificial classifiers on instruments. Classification accuracy is correlated to how much timbre information is imbued in the latent vector. We also visualized the latent space using PCA and t-SNE. Finally, we decided to test our spatial understanding of the latent space by taking samples of specific instruments and linearly mapping them to other instrument domains in the latent space. We then pass them through the decoder to construct the corresponding audio signal and confirm our expectations.

Autoencoder. To construct our training set, we embedded audio WAV samples from n = 11 different instruments from the NSynth Database using the NSynth encoder segment. The output embedded vector is in a tensor of shape (x,y,z) where x = batch, y = sample size/32, and z = 16 dimensions. All samples had a sample size of 40,000 and sampling rate of 16,000, resulting in an embedded matrix of shape 40,000/16,000/32 = 78 x 16. We then flattened this matrix, since we take every entry in the matrix as an individual feature in further classification stages, resulting in 1248 features. Finally, we constructed a matrix of these latent vectors (features) and their corresponding instruments (targets).

Classification. We tested how well a Support Vector Machine (SVM), and Multi-Layer Perceptron, could classify the instrument. With the assumption that an Artificial Neural Network trained on the original wav data could classify with 100% accuracy, the classification accuracy at this stage should directly correspond to how well timbre is imbued in the embedding.

Multi-Layer Perceptron. For the classification component, the first artificial classifier we used was the multi-layer perceptron. As discussed in class, the multi-layer perceptron has flexible weights with activation functions residing in the intermediary nodes. The perceptron model we used had three hidden layers of size 250, or approximately one-fifth the dimensionality of our embedded vector. Our activation function was the rectified linear unit (ReLU) function. At the end of the model, the last layer is summed in a weighted

fashion to produce a single integer corresponding to the instrument classification predicted. It was able to achieve a definitively strong classification accuracy of 84%.

We must also make sure to add a bias to the perceptron, a constant weight outside of the inputs that allows us to achieve better fit for our predictive models.

Support Vector Machine. We also implemented the support vector machine. This classifier finds a hyperplane, or multi-dimensional plane, between two classes. Since we had eleven different instruments, the SVM uses a one vs one heuristic of building n*(n-1)/2 classifiers to create as many decision boundaries between each pair of instruments. Additionally, it has a margin parameter Samples which cross this threshold, even if not completely misclassified, result in the model incurring a penalty during training. The SVM performed at a lower accuracy than the perceptron, scoring approximately 72%. Regardless, these high scores demonstrate that there is instrument indicative information embedded within our latent vector.

PCA/t-SNE. Two methods of dimensionality reduction were used to visualize the embeddings: Principle Component Analysis (PCA), and t-distributed stochastic neighbor embedding (t-SNE). Principal component analysis is a method of better understanding variation of the data, so in our case the embeddings. PCA proves useful in transforming the already reduced embedded vector into two dimensions, which is necessary for visualization purposes. We also visualized our data in 3 dimensions using t-SNE, a dimensionality reduction method that maintains local data structures by coordinating proximity in high dimensional data with its representation in the 3 dimensional visualization. Additionally, we color coded the embedding vectors according to what instruments were used in their productions, so that we may better understand how similar different instruments' timbres are to one another and see if there is consistency in the spatial location of the embeddings.

Synthesis. Finally, we were able to linearly transform embedded vectors of an original instrument into the average latent space of another instrument. The transformed embedded vector is then passed through a decoder to reconstruct the sound played by a new instrument.

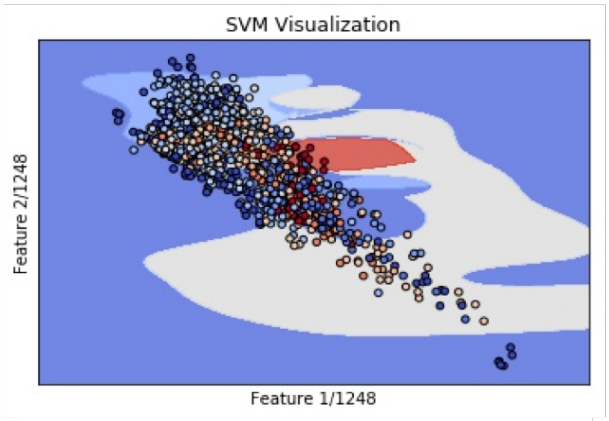
Results

Classification. Once the autoencoder was trained to deconstruct sound inputs into embedded vectors, we used the features of each vector in order to classify instruments. We first employed a Multi-Layer Perceptron which yielded an accuracy of 84%. We then employed a Support Vector Machine without any parameter tuning to classify instruments based on embedded vector features. The SVM performed slightly worse than the multi-layer perceptron, with an accuracy of 72%. Results and visualizations of both of these classification methods can be found in *Figure 2*.

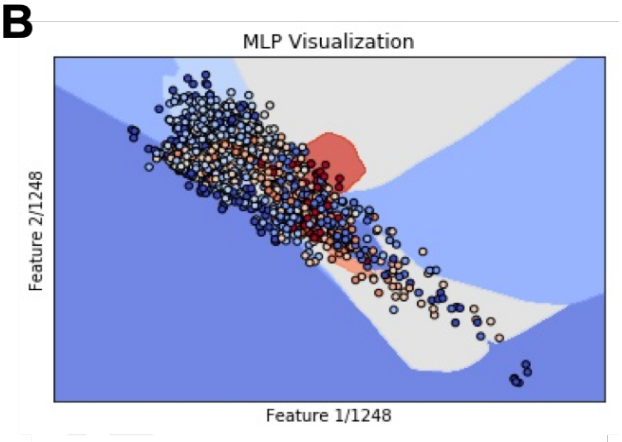
Visualizations. Representations of the 16 dimensions within the embedded vectors via PCA and t-SNE can be see in *Figure 3*. Instruments of similar timbre are clustered closer together in the latent space, leading us to believe that the embedded vectors retain timbre information despite their reduced dimensionality. Additionally, instruments that are not well segregated in the latent space have related physical properties, such as reed, vocal, and strings.

Transformation of Instruments. *Figure 4* outlines the process of linearly transforming instruments. We have had successful and unsuccessful attempts at this transformation, mostly limited by the nature of a linear transformation (see Discussion subsection: *Transformation of Instruments*). *Figure 4: A* depicts a successful transformation, where the output WAV file constructed by the decoder plays similarly to a vocal instrument even though it was perceived in the latent space as a reed instrument, while *Figure 4: B* depicts an unsuccessful transformation.

A



B



C

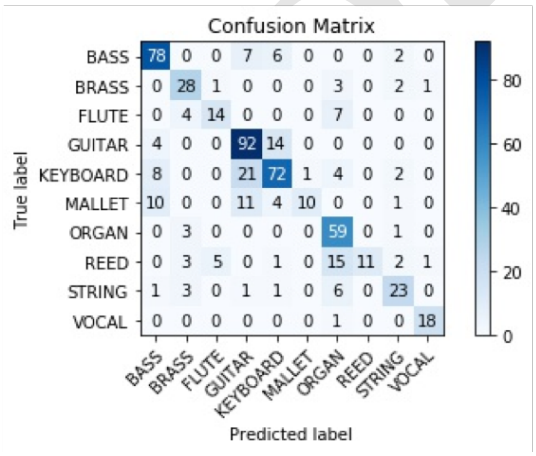
0.7206405693950177

	precision	recall	f1-score	support
0	0.77	0.84	0.80	93
1	0.68	0.80	0.74	35
2	0.70	0.56	0.62	25
3	0.70	0.84	0.76	110
4	0.73	0.67	0.70	108
5	0.91	0.28	0.43	36
6	0.62	0.94	0.75	63
7	1.00	0.29	0.45	38
8	0.70	0.66	0.68	35
10	0.90	0.95	0.92	19
micro avg	0.72	0.72	0.72	562
macro avg	0.77	0.68	0.68	562
weighted avg	0.75	0.72	0.70	562

0.8256227758007118

	precision	recall	f1-score	support
0	0.84	0.96	0.89	93
1	0.86	0.71	0.78	35
2	0.70	0.84	0.76	25
3	0.81	0.85	0.83	110
4	0.88	0.80	0.83	108
5	0.95	0.56	0.70	36
6	0.78	0.86	0.82	63
7	0.76	0.89	0.82	38
8	0.77	0.66	0.71	35
10	1.00	1.00	1.00	19
micro avg	0.83	0.83	0.83	562
macro avg	0.83	0.81	0.81	562
weighted avg	0.83	0.83	0.82	562

D



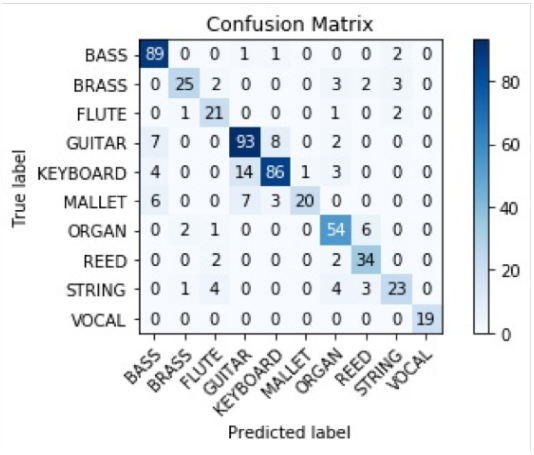


Fig. 1. Classification Algorithms Accuracies and Outputs. **(A**) Support Vector Machine Classification results. **(B)** Multi Layer Perceptron Classification results. **(C)** Accuracies and F1 scores for classification algorithms. Accuracy is of SVM is 72.06% and of MLP is 82.56%. **(D)** Confusion Matrices for classification protocols.

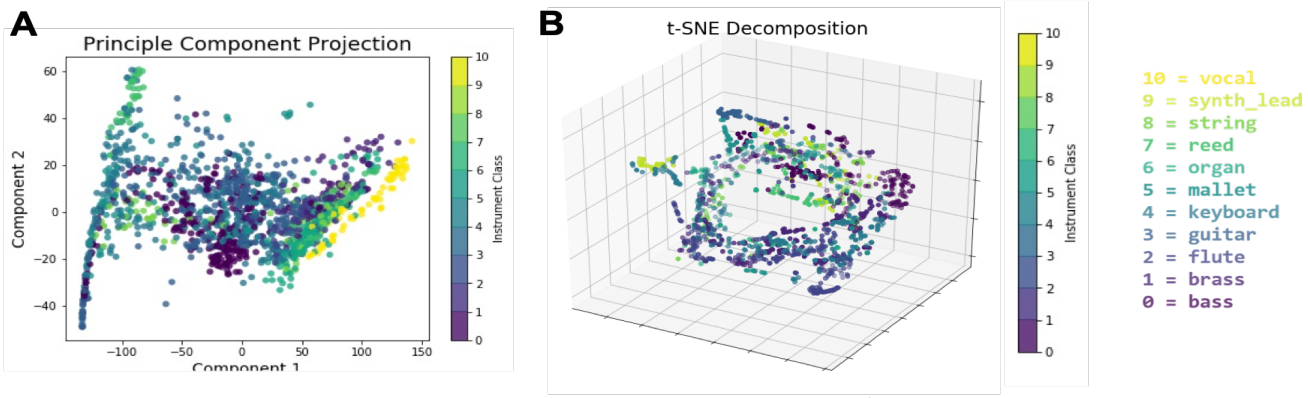


Fig. 2. Dimensionality Reduction and Visualization of Outputs. **(A)** Principle Component Analysis from the embedded Vector. Here we can see clear clusters in two dimensions. **(B)** t-SNE from the embedded vector. Three dimensions were chosen here to show a greater separation of embedded vectors.

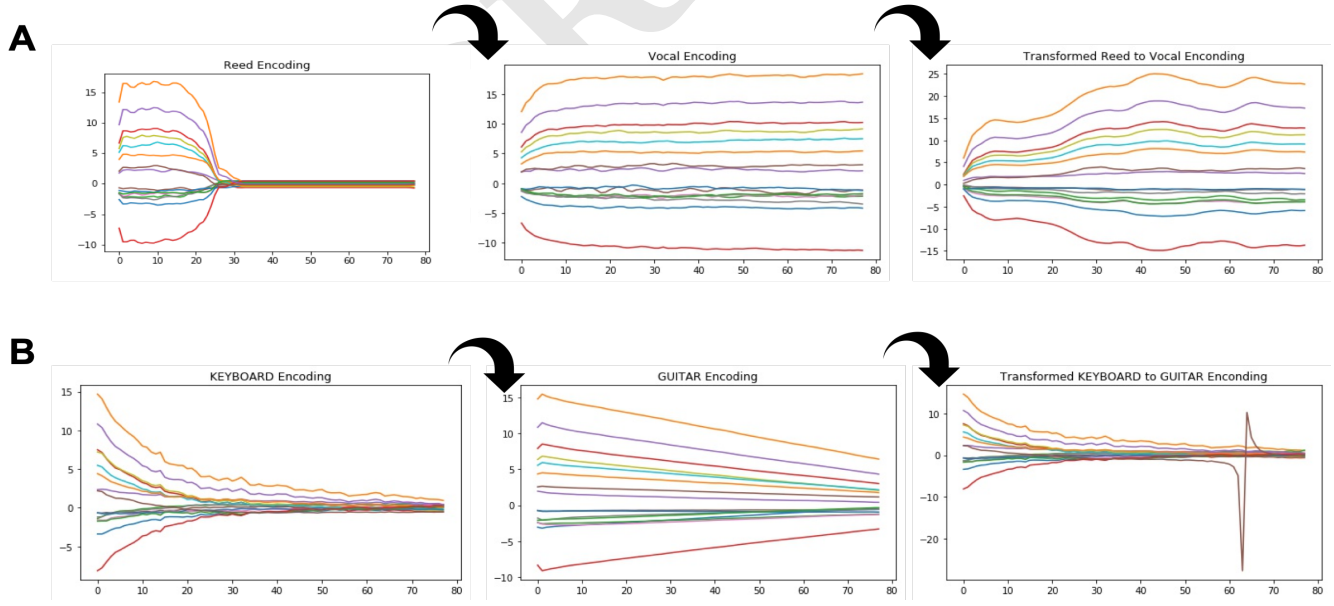


Fig. 3. Linear Transformation of Instruments. **(A)** A positive example of a linear transformation between Reed and Vocals. **(B)**. A failed linear transformation with an unexplained spike artifact. Leads to a reconstructed sound that upon auditory examination neither guitar nor keyboard

Discussion

Classification. In order to explain the lower classification accuracy of the MLP in comparison to the SVM, we also amalgamated precision, recall, and F1 scores per instrument, where precision represents the proportion of negatives correctly identified against the total negatives in the dataset, as modelled by true negatives/ (true negatives + false positives), and recall represents the proportion of positives correctly identified against the total positives in the dataset, as modelled by true positives/ (true positives + false negatives). The F1 score accounts for both precision and recall and is calculated by taking the harmonic average between the two. Interestingly, precision, recall, and F1 distributions per instrument are not mirrored between the two classifier confusion matrices. The MLP has the lowest F1 score for the brass instrument classification due to its low precision score, signifying the model misclassifies a large number of instruments as brass when they belong to other categories. However, the perceptron has the lowest F1 score for the mallet instrument classification due to its low recall score, signifying the model is unable to perceive many mallet classifications in the dataset. The macro averages across precision and recall score, or the averages with each class given equal weight in the calculation, yield more intuition into the reduced MLP performance. The MLP recall score was significantly lower than the SVM (68% vs 78%), demonstrating the model is less capable of identifying true positives in the dataset. We hypothesize that a higher classification would be possible if we optimized the size of the hidden layers with respect to classification accuracy.

Dimensionality Reduction Visualizations. The first method of dimensionality reduction is the principle component analysis, where a linear combination of features that capture the highest variance in the data are plotted on each axes. The second method is the t-SNE [7]. The t-distributed stochastic neighbor embedding method is different in two ways. First of all, it doesn't capture variance, but attempts to maintain neighbors in higher dimensions close to those same neighbors in low dimensions. While doing this, it also tries to maximize KL divergence between classes. This leads to various shapes and forms in lower dimension with the goal of maximizing distances between unlike classes, hence the interesting doughnut shape in *Figure 3*. How these dimensionality reduction techniques help us understand the brain and auto encoders is two-fold. First of all, compression of these audio files is not as simple as a max-pooling algorithm for example. Through principle component analysis, we can clearly see clustering based on input instrument, even though the audio samples are of different notes. For this reason we get dispersion between clusters, but it is still incredible that the autoencoder is able to capture these physical properties.

Transformation of Instruments. To linearly transform embedded vectors to another instrument's space, we solved the simple matrix equation $Ax = b$, where A would be the input vector, b the output vector, and x the transformation matrix. What is interesting about this approach is that to transform, say, Vocal to Reed, we took the average features of Vocals and Reeds, then calculated the transformation matrix $x = bA^{-1}$. This matrix was then multiplied by the current Vocal sample to yield a transformed Reed.

A linear transformation is one of the many ways to transform a vector to a different space.A linear transformation is one of the simplest ones, and thus may not have yielded the best instrument translations.The embedded vector is likely a very complex deconstruction of a highly dimensional audio file, so a nonlinear transformation such as a neural network will likely yield more accurate reconstructions. In order to create such a network, more training data would be needed, and a labelling protocol for properly translated instrument files would need to be established. It would be of our interest to pursue this method in future of our investigation, especially given the wide range of resources in the field of neural networks and nonlinear transformations.

Some transformations yielded better results

than others. One hypothesis for why this could be the case is related to how well the timbre of the original and transformed instruments match up. Better matching means smaller magnitude of transformation, which could in turn lead to less distortions in the final waveform, and thus a more accurate reconstruction. Another reason why we see these differences in *Figure 4* could be explained by the quality of samples provided by the NSynth database. Perhaps a keyboard file was sampled with noise and therefore linear mappings to that space will always be corrupted. Finally, it could be the case that because there are both acoustic and synthetic (digital) versions of instruments, this combination led to distortions in the creation of matrices A and b. Since we take the average of features to create these matrices, having these two versions of an instrument could completely change the nature of the average.

Future Directions

Our investigation into the latent space constructed via NSynth's autoencoder was constrained by the dataset consisting of pure musical notes we had available. Additionally, we focused on the potential of the autoencoder in timbre retention, and did not harness the full potential of the autoencoder in retaining temporal information in the embedding. Exploring the latent space constructed by feeding in note sequences, full melodies, and musical genres would be the logical next step in investigating NSynth's powers, as well as constructing more decoded compositions.

Generative neural networks in the visual domain have successfully achieved pattern mimicry of artists, for example, they have replicated Van Gogh's brushstrokes or Monet's impressionistic style. Once trained, these networks are able to take photographs of natural landscapes and reconstruct them as paintings in different artistic styles. Further, autoencoders can be used to modify a person's facial expression. The face is passed into the encoder, and then the embedding vector is modified by a little, this modified embedding is passed into the decoder, and the same face with a different expression is produced! We see potential in the modifying the generative component of NSynth's autoencoder to perform similar feats. While in the scope of our project, we were only able to investigate instrument regimes within the latent space, one could ambitiously imagine that feeding the autoencoder melodies might yield a latent space with John Coltrane vs. Kanye West regimes, and could allow synthesis of whole songs reinvented in different artistic styles.

Conclusion

An autoencoder is one architecture that mirrors how the brain captures information from the cochlea and reduces information into an invariant and less complex representation. We have built in this investigation, an autoencoder that translates sound into a vector. This vector was analyzed thoroughly by measuring its capability of discerning between instruments through classification and visualization. Finally, we attempted to transform this vector to understand whether or not the way that it is represented can be mutated, thus changing the nature of the output sound to another instrument. We hope that this study paves the way for more progress in the field of brain representation of sensory inputs.

ACKNOWLEDGMENTS. We would like to thank Clelia for her invaluable feedback on how to move forward with this project. We also want to thank everyone involved in the project, folks that helped maintain the NSynth database, and Dr. Vijay Balasubramanian

Bibliography

[1] Roche, Fanny Hueber, Thomas Limier, Samuel Girin, Laurent. (2018). Autoencoders for music sound synthesis: a comparison of linear, shallow, deep and variational models.

[2] Ramani, Dhruv Karmakar, Samarjit Panda, Anirban Ahmed, Asad Tangri, Pratham. (2018). Autoencoder Based Architecture For Fast Real Time Audio Style Transfer.

[3] Plack, C. J. (2018). The sense of hearing. Routledge.

[4] .M. Darling, "Properties and implementation of the gammatone filter: A tutorial", Speech hearing and language, University College London, 1991.

[5] S. J. Elliott and C. A. Shera, "The cochlea as a smart structure," Smart Mater. Struct., vol. 21, no. 6, p. 64001, Jun. 2012.

[6] T. P. Lillicrap et al., Learning Deep Architectures for AI, vol. 2, no. 1. 2015.

[7] L.J.P. van der Maaten and G.E. Hinton. Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research 9(Nov):2579-2605, 2008