
Immersions - How Does Music Sound to Artificial Ears?

Vincent Herrmann

Institute for Musicology and Music Informatics
University of Music Karlsruhe
vincent.herrmann@web.de

Abstract

Immersions is a system that visualizes and sonifies the inner workings of a sound processing neural network in real-time. An optimization procedure generates sounds that activate certain regions in the network. This renders audible the way music sounds to this artificial ear. In addition the activations of the neurons at each point in time are visualized, creating the experience of immersing oneself into the depths of an artificial intelligence.

1 Introduction

Although neural networks are often described as black boxes, there exist methods to make us see and hear what is going on inside a neural networks. A well known example of this is the DeepDream technique [1] that generates bizarre psychedelic but strangely familiar pictures. We follow a related approach to make audible the innards of sound processing neural nets.

The basic idea is the following: You start with an arbitrary sound clip (e.g. a drum loop). This clip will be modified by optimization procedure in way that stimulates a certain region in the neural net. The changed clip can then be used as a basis for a new optimization. This way you obtain sounds that are generated by the neural network in a freely associative way.

One of the most important properties of neural networks is that they process information on different levels of abstraction. An activation in some region of the net thus may for example be associated with short and simple sounds or noises, a different region with a more complex musical phenomenon, such as rhythm, phrase or key. For what exactly a neural networks listens depends on the data it was trained on, as well as on the task it is specialized to accomplish. This means to different nets music will sound differently. All this can be directly experienced with the Immersions project. For this I developed a setup that allows - not unlike a DJ mixing console - the generation, visualization and control of sound in real-time.

2 Technical Concepts

2.1 Contrastive Predictive Coding

In the center of this project stands an "artificial ear", meaning a sound processing neural network. It has been shown that analogies exist between convolutional neural networks (ConvNets) and the human auditory cortex [2]. Of course the human auditory system is not a classification network, as the used used in [2], we usually don't learn with the help of explicit labels. The question of the actual learning mechanisms in the brain is highly contested, but a promising candidate, especially for perceptual learning, might be predictive coding [3, 4]. Here future stimuli are predicted and then compared to the actual occurred stimuli. The learning entails reducing the discrepancy between prediction and reality.

One method of using the idea of predictive coding for unsupervised learning in artificial neural networks is presented in [5] and [6]. For this so-called contrastive predictive coding you need two nets: The encoder net builds a compressed encoding \mathbf{z}_t of the signal for each time step t and the autoregressive net summarizes multiple sequential encodings into a vector \mathbf{c} . The encoding k time steps in the future, \mathbf{z}_{t+k} , is predicted by a different learnt linear transformation \mathbf{M}_k of \mathbf{c} for each time step. The loss for each mini-batch is calculated as follows:

$$\mathcal{L}_{CPC} = - \sum_{n,k} \log \frac{\exp(\mathbf{z}_{n,k}^T \mathbf{M}_k \mathbf{c}_n)}{\sum_m \exp(\mathbf{z}_{m,k}^T \mathbf{M}_k \mathbf{c}_n)}$$

Both n and m index the mini-batch dimension. For details on the model architecture used here, please see Figure 2 and appendix A.

For contrastive predictive coding no expensively labeled datasets are required, the model learns to make sense of any inputs it is given. The models for this project were trained on datasets consisting of about 40 hours of house music mixes or 200 hours of classical piano music.

2.2 Visualization

To illustrate what is going on inside the model at each moment we visualize the activation of the artificial neurons. The time dimension is retained, resulting in animated pictures. The neurons have to be arranged in a 2D layout according to their position in the network (given by layer, channel and pitch). The neurons are connected with one another according to different patterns depending on the type of layer (e.g. fully connected or convolutional). This leads to a graph layout problem.

To find a layout for a graph of this size (tens of thousands vertices and many millions connections) a multilevel force layout method [7, 8] is suited. Here the graph is modeled as a physical system, where the vertices repel each other as if electrically charged with the same polarity, and edges act like rubber bands pulling connected vertices together. To avoid local energy minima we start with few vertices and gradually add more until the layout of the complete graph is calculated. Details on the procedure can be found in appendix B.

Once the layout is worked out the current state of the net can be depicted by lighting up strongly activated neurons and letting others stay dark (see Figure 1).

2.3 Input Optimization

Today's neural networks usually are completely differentiable. This means we can generate inputs for a trained model that maximize the activations of certain neurons in the network using gradient based optimization [9, 1, 10]. These inputs can then be directly experienced by humans and show which particular stimuli the selected neurons respond to.

Neural nets, especially if they did not receive adversarial training [11], are susceptible to small changes in the input. These can lead to local optima that are not perceptible by humans but still have the required properties. To prevent this we apply several types of regularization: temporal shifting of the input, small pitch changes, masking of random regions in the scalogram, noisy gradients and de-noising of the input. All these methods make the input optimization more difficult in certain ways and thus enforce more robust and distinct results.

3 Live Performance

The generated audio clips have a duration of about four seconds. With that they lend themselves for a loop based live performance. One loop then corresponds for example to two 4/4 measures at 120 bpm. The optimization procedure described above constantly generates new audio clips. As soon as one clip has finished playing, the latest newly calculated clip is started, resulting in an acoustic morphing.

All aspects of the procedure can be adjusted in real-time. For intuitive control I developed a GUI and made the most important parameters controllable by a MIDI controller. The setup described here is very flexible, other networks trained on different data or with different architectures can easily be employed.

4 Ethical Implication

Neural networks might be the artificial systems that come yet closest to our own ways of thinking and perceiving. This work tries to explore this area between familiarity and strangeness. The visualization in a way anthropomorphizes, or at least biologizes, the results of calculations (the emerging shapes, although completely determined by the network architecture, could remind one of a micro-organism or a brain in a jar). You might argue that this paints a wrong and maybe even dangerous picture, buying into the current hype or promoting overconfidence in these kind of novel systems. My hope however is that this work rather invites people to think critically about the results they see and hear. We are all fascinated by neural networks not without a reason. I don't think we have even begun exhaust the many ways to examine them and gain a deeper understanding.

For further material, demos and code, please see <https://vincentherrmann.github.io/demos/immersions>

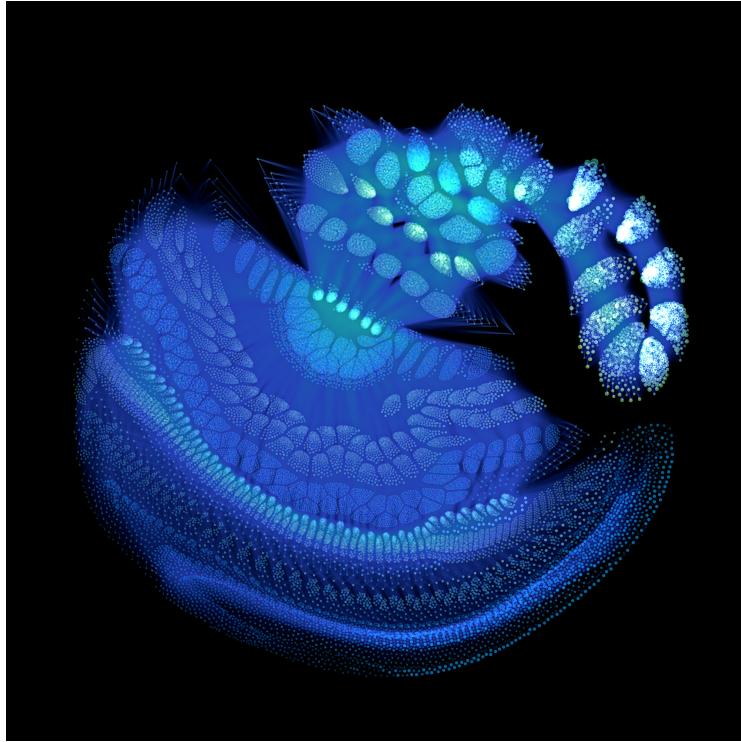


Figure 1: Activation visualization of the model trained on house music

References

- [1] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. 2015.
- [2] Alexander JE Kell, Daniel LK Yamins, Erica N Shook, Sam V Norman-Haignere, and Josh H McDermott. A task-optimized neural network replicates human auditory behavior, predicts brain responses, and reveals a cortical processing hierarchy. *Neuron*, 98(3):630–644, 2018.
- [3] Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79, 1999.
- [4] Karl Friston. A theory of cortical responses. *Philosophical transactions of the Royal Society B: Biological sciences*, 360(1456):815–836, 2005.
- [5] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

- [6] Sherjil Ozair, Corey Lynch, Yoshua Bengio, Aaron van den Oord, Sergey Levine, and Pierre Sermanet. Wasserstein dependency measure for representation learning. *arXiv preprint arXiv:1903.11780*, 2019.
- [7] Chris Walshaw. A multilevel algorithm for force-directed graph-drawing. *Journal of Graph Algorithms and Applications*, 7(3):253–285, 2006.
- [8] Yifan Hu. Efficient, high-quality force-directed graph drawing. *Mathematica Journal*, 10(1):37–71, 2005.
- [9] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.
- [10] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11):e7, 2017.
- [11] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Brandon Tran, and Aleksander Madry. Learning perceptually-aligned representations via adversarial robustness. *arXiv preprint arXiv:1906.00945*, 2019.
- [12] Einar Kjartansson. Constant q-wave propagation and attenuation. *Journal of Geophysical Research: Solid Earth*, 84(B9):4737–4748, 1979.
- [13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [14] Josh Barnes and Piet Hut. A hierarchical $O(n \log n)$ force-calculation algorithm. *nature*, 324(6096):446, 1986.
- [15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

A Network Architecture

The encoder network receives as input the scalogram of a roughly four second long audio clip. A scalogram is the result of a constant-Q or a wavelet transformation [12] of the audio signal and comes quite close to the representation that the cochlea passes on to the brain (see [2]). Since it is a 2D representation (with the dimensions time and frequency) we can use a conventional 2D-ConvNet of the kind common in the computer vision and image processing.

As encoder we use a modified ResNet [13] with ReLu nonlinearities, max pooling and batch normalization. In addition to the usual quadratic kernels we also apply kernels that only span the pitch dimension and are supposed to detect overtones and harmonies (see Table 1). The autoregressive model is a ResNet as well, but with 1D convolutions (Table 2).

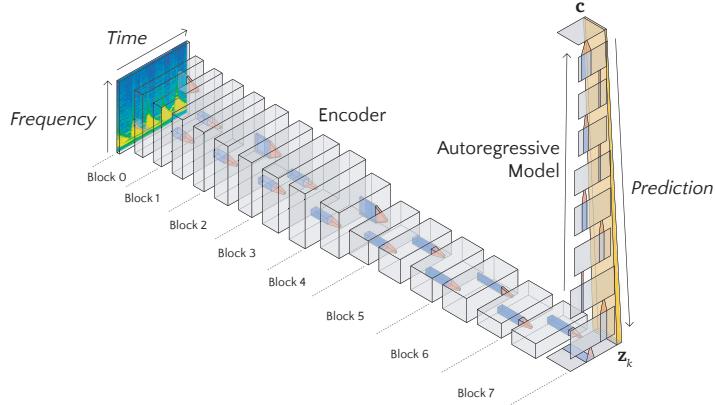


Figure 2: Model architecture

Block	Channels	Conv 1	Conv 2
0	8	3x3 pool 2	1x25
1	16	3x3	3x3
2	32	3x3	1x25
3	64	3x3	3x3
4	128	3x3 pool 2	1x25
5	256	3x3	3x3
6	512	3x3	1x4
7	512	3x3	3x3

Table 1: Encoder

Block	Channels	Conv
0	512	5
1	512	4
2	512	1 pool 2
3	512	3
4	256	3
5	256	1 pool 2
6	256	3
7	256	1
8	256	6

Table 2: Autoregressive Model

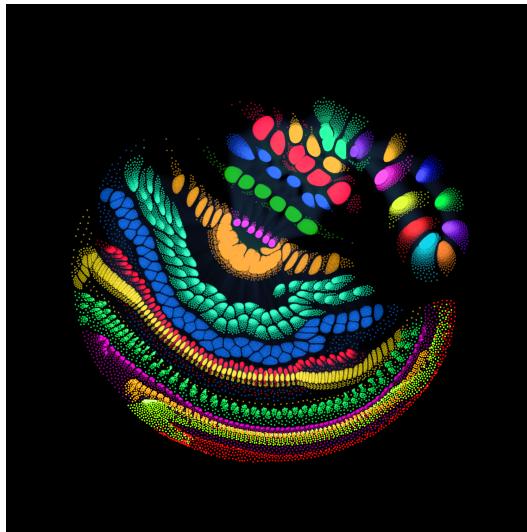


Figure 3: Activation visualization of the model with each layer in a different color

B Neural Layouts

The coarse graph levels for the layout calculation presented in 2.2 are achieved by iteratively merging two vertices into one with the combined weight. First we merge neighbouring vertices in the channel dimension, then in the pitch dimension. During the layout calculation the higher resolution graphs are reconstructed in the reverse order. When calculating the layout for a trained neural network, I used the variance of each activation over the validation set as the vertex weight.

Since in a force layout each vertex interacts with every other one, to keep the computational load acceptable a Barnes-Hut tree approximation [14] has to be used in order to reduce the computational complexity. For GPU acceleration and easy integration into the software existing ecosystem I implemented the graph layout algorithm in PyTorch. The code is available at <https://vincentherrmann.github.io/demos/immersions>.

This visualization method is very flexible and can be used for various neural network architectures. While it is possible to calculate a layout for a complete 2D convolutional network, in a 2D visualization setting the results are overcrowded and thus not very insightful. Because of that we stick with cuts through one of the spatial dimensions for the VGG [15] and ResNet [13] architectures shown in Figures 5 and 6.

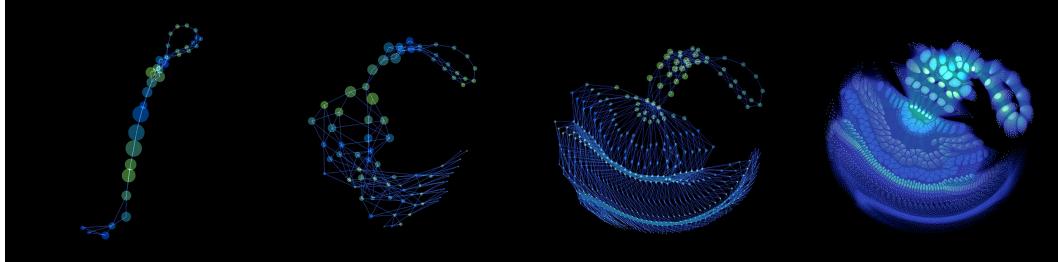


Figure 4: Four steps of the layout calculation for the model architecture used in this paper

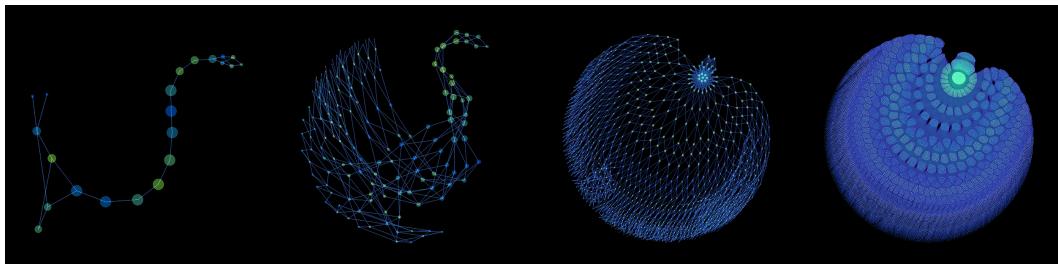


Figure 5: Four steps of the layout calculation for a VGG16 architecture

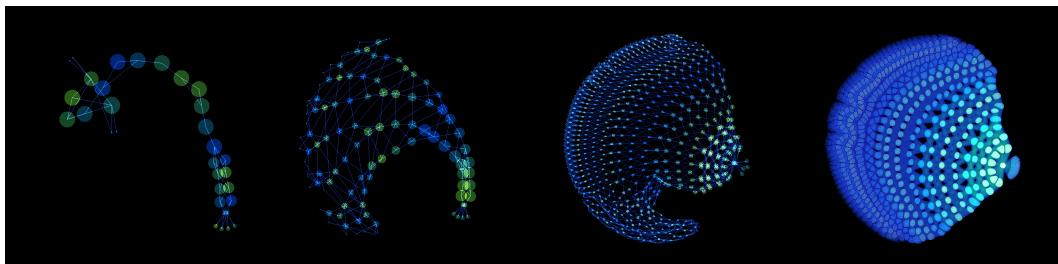


Figure 6: Four steps of the layout calculation for a ResNet18