Audio equalization is the process of boosting and lowering the volume of different frequency bands (ranges) within an audio signal. It is traditionally done manually through software or hardware (older methodology) (also, an equalizer will be abbreviated as EQ for the rest of the paper). There are different types of EQs, but the most common are graphic (less control over bands) and parametric (newer and most control over bands). It is important because it makes audio sound better, but also very necessary for many sound engineering-related tasks such as ensuring that a song doesn't have different instruments that clash, creating what is known as a "muddy signal", shaping a sound to the user's preference, fixing a song to fit its environment or target device (e.g., studio speakers sound much different than phone speakers, so it is important to adjust for the difference in sound), ensuring that the vocals of a song or a speech sound crystal clear, and fixing instruments or microphones that sound off. Now, AI plays a role in audio equalization through multiple different areas. For instance, it provides many benefits that can aid engineers, as well as help novices perform sound engineering that would normally require much more experience. Also, it can analyze an audio signal and make adjustments that would normally require the knowledge of an experienced engineer. Additionally, through neural networks, it is able to learn optimal EQ settings, allowing adjustments fit for specific genres, venues, or user preferences.

In order to better understand how AI is used in equalization, it is important to first understand how equalization itself works. Equalization uses three main processes in order to shape sound: sampling, Fourier transformation, and finally a digital filter (which is what actually performs equalization). The first part, sampling, is necessary because an audio signal that is sent to and played from speakers is a continuous analog signal, something that computers cannot easily work with, so it must first be converted to a different format. Sampling achieves this conversion through the process of capturing a continuous analog signal at various time intervals. Ideally, it produces samples that do not differentiate from the continuous signal at various intervals. The two main components that make up sampling are frequency and bit depth. Frequency (also known as sampling rate) is the average number of samples captured in a second and is measured in hertz, so for instance, 8kHz = 48,000 samples per second. The other component, bit depth, is the number of bits (the computer bit) per sample, i.e., the resolution of each sample. One important thing to note about bit depth is that it influences the dynamic range (range from smallest to loudest noise) of each sample. There are multiple different frequencies and bit depths that are used today for sampling, but frequency is the important value to consider when ensuring that sampling can accurately capture a continuous analog audio signal. The reasoning behind this is highlighted by the Nyquist-Shannon sampling theorem: if a function x(t) contains no frequencies higher than B hertz, then it can be completely

determined from its ordinates at a sequence of points spaced less than 1/(2B) seconds apart. In essence, this means that the sample rate has to be at least double the highest frequency component of the original signal. How sampling actually uses frequency and bit depth to transform an analog signal into a format for computers is through two processes called discretization and quantization. Firstly, discretization is the process of breaking down a continuous-time signal into discrete-time intervals (i.e., actually taking samples from a signal) and introduces discretization error. Secondly, quantization is the process of mapping the amplitude of each sampled value to a finite set of levels and is often done through rounding and truncation. Additionally, quantization is necessary because computers can only work with binary data, which continuous audio signals are not, so the signal needs to be converted into data that a computer can work with.

After sampling is performed on a continuous analog audio signal, the next main step is to extract frequency information from the sampled signal that is used to perform equalization. The process of extracting this frequency information is done through a discrete Fourier transform, which transforms the signal from a time-domain signal to a frequency-domain signal. Apart from being able to extract frequency information, a discrete Fourier transform is necessary to get frequency information out of the signal, and to put the signal in a form that can be worked with for equalization processing. Essentially, the discrete Fourier transform is the continuous Fourier transform that works over a finite set of samples produced from sampling. The continuous Fourier transform decomposes a continuous signal into a sum of sinusoidal signals (a periodic sine wave) that produces the magnitude (the strength of the signal, from $|X(f)|$) and the phase (the position of a wave at a given time interval, from $X(f)$). Magnitude is needed because it is the actual frequency value of each sample, and is used in digital filters (explained later) to actually do equalization. Phase is needed to ensure that any modifications to the magnitude, or frequency, through equalization does not modify the time of the signal. It is important to point out that the discrete Fourier transform by itself is far too slow to be used in signal processing, as well as other computing applications. To solve this issue, there have been multiple algorithms created to compute the discrete Fourier transform much more efficiently. The most well-known and used version is the fast Fourier transform, which significantly reduces the number of computations required. More specifically, it reduces the complexity of computing the DFT from $O(n^2)$ to $O(n \log n)$. Multiple different algorithms exist, but the most common is the Cooley-Tukey that recursively breaks down a DFT into smaller sizes and uses roots of unity multiplications.

To actually perform equalization, the logarithmic decibel-scaled fast Fourier transform is then fed into a digital filter. A digital filter is a mathematical algorithm that modifies certain frequency components of the signal based on inputs (e.g., logarithmically-scaled FFT samples) and outputs (the modified signal after being modified from the algorithm). The output of a digital filter can be modeled using the transfer function, where B(z) are the inputs and A(z) are the outputs. Digital filters use impulse response to refer to the output of the algorithm when an impulse, or brief input signal, is fed in as the input, and there are two main types of digital filters that center around impulse response: finite and infinite. Finite impulse response (FIR) filter does not use feedback, meaning the output depends only on current and past input values, whereas infinite impulse response (IIR) filter uses feedback, meaning that past output values influence future outputs (e.g., it recursively goes back to use previous output values).

In order to utilize AI for equalization, machine learning is used to train on and output to a digital filter (or multiple digital filters). Often, this is done through a neural network, with the convolutional neural network (or CNN) being the most commonly used variant. A CNN builds upon a regular neural network that includes input layers, hidden layers, and an output layer, and is often used for signal processing because they excel with data that has patterns. Because of this, it is the most common ML used for image processing. The main part of what sets CNNs apart from traditional neural networks is in the hidden layers, of which the most important are: convolution, which applies a set of filters (or kernels) to an input that each only act on a small set of the input, resulting in what is known as a feature or activation map; a Rectified Linear Unit (ReLU), which helps speed up training by making all negative values non-negative; pooling, which performs down sampling on the feature map, which reduces the total number of parameters the CNN is learning on; a fully connected layer, which uses the previous layer to calculate the probabilities for each output classification; and finally, the final layer, which is the output that provides a classification output. It is worth pointing out that each layer uses shared weights and bias values for all neurons. CNNs are used for equalization by being trained on and outputting to a digital filter.

References

Amirivojdan, A., Nasiri, A., Zhou, S., Zhao, Y., & Gan, H. (2024). Chicken Sense: A Low-Cost Deep Learning-Based Solution for Poultry Feed Consumption Monitoring Using Sound Technology. AgriEngineering, 6(3), 2115-2129. https://doi.org/10.3390/agriengineering6030124

GeeksforGeeks. (n.d.). Introduction to Convolutional Neural Network. From https://www.geeksforgeeks.org/introduction-convolution-neural-network/

Glasswing Ventures. (2024, February 21). AI Atlas #16: Convolutional Neural Networks (CNNs). Glasswing Ventures Blog. https://glasswing.vc/blog/ai-atlas-16-convolutional-neural-networks-cnns/

Ling, Q., Chen, Y., Feng, Z., Pei, H., Wang, C., Yin, Z., & Qiu, Z. (2025). Monitoring Canopy Height in the Hainan Tropical Rainforest Using Machine Learning and Multi-Modal Data Fusion. Remote Sensing, 17(6), 966. https://doi.org/10.3390/rs17060966

MathWorks. (n.d.). Convolutional neural network (CNN). MathWorks. Retrieved from https://www.mathworks.com/discovery/convolutional-neural-network.html

Mockenhaupt, F., Rieber, J. S., & Nercessian, S. (2024). Automatic equalization for individual instrument tracks using convolutional neural networks. arXiv. https://arxiv.org/abs/2407.16691

Pepe, G., Gabrielli, L., Squartini, S., Tripodi, C., & Strozzi, N. (2022). Deep optimization of parametric IIR filters for audio equalization. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 30, 1136-1149. https://doi.org/10.1109/TASLP.2022.3155289

Prince, S. J. D. (n.d.). Lecture 7. University of Oxford. https://www.robots.ox.ac.uk/~sjrob/Teaching/SP/17.pdf

Taboga, Marco (2021). "Discrete Fourier Transform - Frequencies", Lectures on matrix algebra. https://www.statlect.com/matrix-algebra/discrete-Fourier-transform-frequencies.

Välimäki, V., & Reiss, J. D. (2016). All About Audio Equalization: Solutions and Frontiers. Applied Sciences, 6(5), 129. https://doi.org/10.3390/app6050129

Wang, J., Li, Z., Zhou, L., Ma, C., & Sun, W. (2025). Ensemble Streamflow Simulations in a Qinghai-Tibet Plateau Basin Using a Deep Learning Method with Remote Sensing Precipitation Data as Input. Remote Sensing, 17(6), 967. https://doi.org/10.3390/rs17060967