Joshua Woods
CS-481
Artificial Intelligence
Dr. Farmer

Convolutional Neural Networks and Analyzing Audio Files

1. Introduction

With the growth of digital audio content in today's online world, the ability to analyze and categorize these robustly and efficiently is becoming more and more critical. Convolutional neural networks (CNNs) offer the solution we seek. They have recently emerged as a powerful tool in audio analysis. Convolutional neural networks focus on extracting features from the provided data and are widely used in image processing. The CNNs leverage their ability to capture local dependencies and hierarchal representations; this allows them to learn the audio data and extract meaningful features automatically. For those reasons, they have shown great promise in audio classification and event detection. In this paper, I will present an in-depth analysis of the architecture of CNNs, how these CNNs can be applied, and how the methods are evaluated. With these insights explored within the audio domain, I aim to understand better their effectiveness and potential use cases in the real world. There is an extensive range of applications for CNNs and analyzing audio files. Within this paper, I will examine multiple examples and go in-depth into how CNNs can perform these tasks. First, I will give an extensive overview of how CNNs work.

2. Technical Discussion

The first step in building any convolutional neural network begins with the convolutional layer. This is the most crucial step in the process as it provides the building block for the rest of the network. It works by passing a set of filters that can be learned, often called kernels, that are small but extend the depth of the input volume. The kernels systematically slide over all the data and perform multiplication and summations between their weights and the input value. The purpose of this operation is so that the local features of the data can be extracted and patterns can be found. With the local parts and designs ready, the convolutional layer can now produce feature maps. *Fig. 1* These feature maps are all specific to a different input aspect, such as a high-frequency point, high amplitude, or multiple combos. The output depth depends on the number of features applied in the convolutional layer. In most neural networks, numerous convolutional layers typically have pooling layers between them.

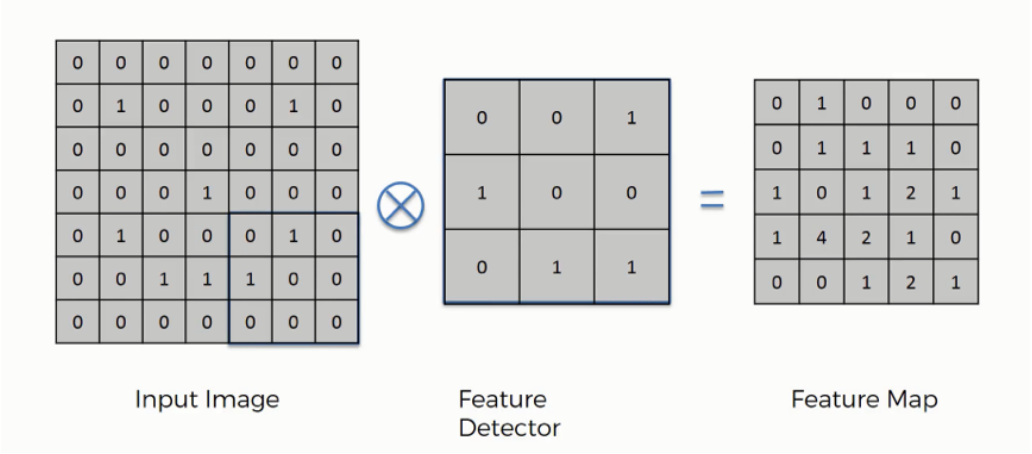

Figure 1

Following the convolutional layer building the feature maps, an activation function is applied to each element in the feature maps, introducing non-linearity. The most common function is the Rectified Linear Unit (ReLU) function, and, for the sake of this paper, it will be the only activation function being covered. However, some others are; the Leaky ReLU and the Parametric ReLU. *Fig. 2* The ReLU function introduces non-linearity so the network can learn complex patterns and relationships in the data. The formula for ReLU is relatively simple and is defined as $ReLU(x) = \max(0, x)$. [1] This formula takes the input value of X and the maximum between 0 and X. If the number is negative, it becomes 0, and positives are left unchanged. This introduces non-linearity, which is crucial if we want the network to be able to learn patterns that are not linear. As aforementioned, when the input to the function is negative, the output is zero. This allows the process to zero out any irrelevant or weak activations to the model. This representation of the model generally reduces the complexity of the model and improves efficiency. This is why the ReLU function is very efficient in computation especially compared to other tasks that require the usage of exponents or division in their calculation. Simplicity is the key here as it makes many passes forward and backward in training. Lastly, the ReLU function solves the vanishing gradient problem [1]. "The ReLu function does not trigger the problem when the number of layers grows. This function does not have an asymptotic upper and lower bound." The vanishing gradient problem diminishes the gradients' magnitude as more backpropagations occur. It provides a greater gradient flow than other activation functions, with less saturation in the positive region. There are some limitations. However, one problem that can arise is the "Dying ReLU"[2]. This is where some neurons "die" during the training and never get activated during the exercise. This means that the output will always be 0 no matter the input and stops contributing to training, which is a computational waste. This generally occurs if the learning rate is too high or the weights are set up so that too many neurons receive negative values and do not recover. Variants like the ones mentioned earlier can solve this problem, the Leaky ReLU and the Parametric ReLU.

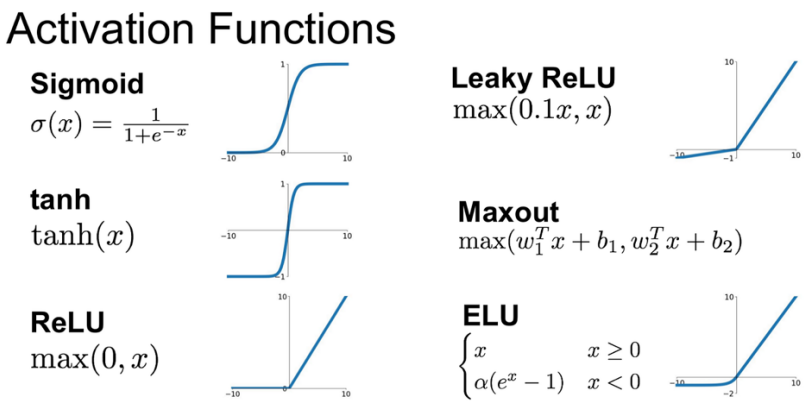

Figure 2

There are pooling layers between the convolutional layers; their two primary purposes are to reduce the spatial dimensions of the feature maps and introduce variances to slight variations from the input. The purpose of spatial reduction is only to keep the most relevant information. This reduces the number of parameters needed in the subsequent layers, which helps to mitigate overfitting. Max pooling, one pooling method, takes non-overlapping regions of the input feature map and outputs the highest value. *Fig. 3* This is to take the most dominant feature of the area, as they are the most desirable features for the model. This also helps keep the model efficient as you use considerably fewer resources once the pooling has occurred. The other purpose of the pooling layers is to introduce a degree of invariance to marginal variations on the input. This tells the network that this feature should be recognized no matter where it is in the input information. Max pooling is not the only pooling option; another commonly used method is "Average pooling," where the average is taken rather than the max. This can be useful in certain situations, and there is no actual one size fits all answer.

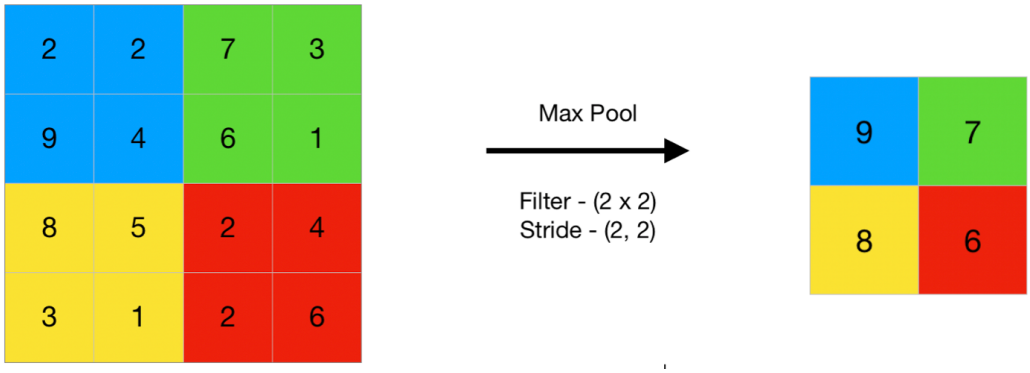

Figure 3

Following the convolutional and pooling layers, one or more "fully connected layers" are typically utilized to learn global patterns and make predictions based on the features provided. Within this layer, every neuron is connected to all the neurons in the previous layer, which is

how traditional neural networks are set up. *Fig. 4* The connections between these neurons all have weights, which are learnable parameters that determine the strength of their relationships. The weights are represented in a weight matrix with a bias vector, the dimensions of the weight matrix are represented by $(N * M)$ with N being the number of neurons in the current layer, and M is the number of neurons in the previous layer. The entries in the matrix represent the weight of the connection between a neuron in the current layer to the last layer. The bias vector is a value for each neuron of the current layer that is then added to the weight for each neuron. This allows for different activation thresholds for other neurons in each layer. An activation function is used once the weights and biases have been applied. This is the same activation function, such as the ReLU function, that went into detail earlier. This aims to add non-linearity into the network to capture complex patterns in the data. Once the activation function has been applied, the network begins parameter learning during the training process. An optimization algorithm, such as stochastic gradient descent [3], accomplishes this. The purpose is to iteratively adjust the weights and biases based on the gradients of the loss function concerning the parameters being learned. The loss function depends on the situation; the angles of the process are calculated concerning the weights and biases and are computed through backpropagation. Lastly, the output layer is connected, and the number of outputs depends on the task. In classification examples, the number of output nodes is the number of classifications possible. There is typically another activation function applied to the output; however, it is usually a softmax function, which converts the result into a probability distribution over the classes possible for predictions. The fully connected layers combine the localized features from the convolutional and pooling layers and use that information to make globalized relationships and accurate predictions.

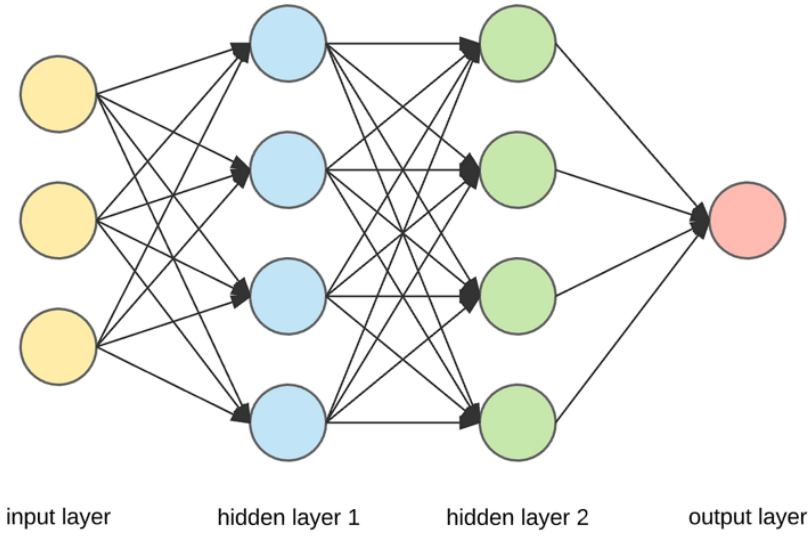

Figure 4

These neural networks also utilize backpropagation, which is the process of propagating data through each layer, and the layers apply concepts we have already discussed a weighted sum of the inputs, and an activation function. This is all during the forward pass of the input data.

Once that has occurred, the loss calculation is performed; a typical loss function is the mean squared error for the estimates. After the forward pass and the loss function are applied, the backward pass is used to compute the gradients of the loss function concerning the parameters of the network. Finally, the weights and biases of the parameters are updated with the information gained from the passes. This is all performed iteratively time after time, with each run being called an epoch. By iterating out, the network can continuously improve and build itself. Backpropagation has helped immensely in improving the feasibility of the predictions and effectiveness of the network.

Specifically, additional preprocessing steps must be taken to ensure that the network effectively processes the raw audio data when analyzing audio data. Some examples of preprocesses that can take place could be a Fast Fourier transform, sampling rate conversion, or some feature extraction techniques such as the Mel-Frequency cepstral coefficient. This helps to capture the essential features of the audio and allows it to analyze those characteristics. After the data preprocessing, the neural network begins to do its work. The convolutional layers are exceptional at picking up local dependencies within the data using the feature maps discussed earlier. *Fig. 1* The network can pick up the features of different temporal and spectral features at different scales. The pooling layers that follow the convolutional layers help to keep the network efficient. The network structure usually has more than one convolutional and pooling layer, which allows only the most relevant features, controls the model's size reasonably, and keeps the network computationally efficient. After the final pooling layer, the data is passed through a fully connected layer, which acts as a classifier. The classes to output depend on the project and will vary from project to project.

The preprocessing steps mentioned above are crucial for audio analysis, so I will review some of the more effective feature extraction techniques that can be used. I am starting with the Fast Fourier Transform (FFT). This is a familiar equation but rather old, as Gauss first proposed it in 1805. “The fast Fourier transform (FFT) is a discrete Fourier transform algorithm which reduces the number of computations needed for N points from $2\,N^2\ to\ 2\,N \lg N$, where lg is the base-2 logarithm.” [4] This algorithm reduces the number of calculations needed depending on the base of the logarithm. The Fast Fourier transform generally falls into two categories: decimation in time or frequency. For a decimation of time, the FFT algorithm developed by Cooley-Turkey rearranges the elements in a bit reversed order, then builds the output transformation. *Fig. 5* The algorithm developed by Sande-Turkey first transforms, then mixes the output values decimating by frequency. *Fig. 6* We use the FFT in audio processing for many reasons, the first and most obvious being frequency analysis. Being able to analyze the frequency patterns of the audio files is vital to the ability to detect important features. Often, the loudness or amplitude of the measured sounds will be very close to each other, and the tone, pitch, or frequency distinction needs to be considered. A Fourier transform must be applied to view the audio as a spectrogram file, which can offer more insight into anomalies or patterns within the

frequency domain. Once the audio is in the frequency domain, other filters can also be applied to it, such as a high or a low pass filter to remove either the high or low pitch sounds depending on the problem at hand. This is used for some things like noise reduction and background noise cancellation. The frequency component is fundamental in detecting pitch if the issue involves pitch detection.

$$\sum_{n=0}^{N-1} a_n e^{-2\pi i n k/N} = \sum_{n=0}^{N/2-1} a_{2n} e^{-2\pi i (2n) k/N}$$

$$+ \sum_{n=0}^{N/2-1} a_{2n+1} e^{-2\pi i (2n+1) k/N}$$

$$= \sum_{n=0}^{N/2-1} a_n^{\text{even}} e^{-2\pi i n k/(N/2)}$$

$$+ e^{-2\pi i k/N} \sum_{n=0}^{N/2-1} a_n^{\text{odd}} e^{-2\pi i n k/(N/2)},$$

Figure 5

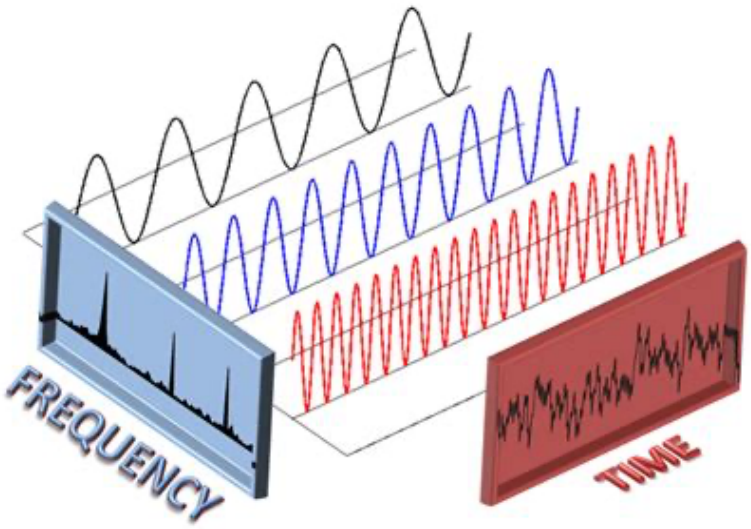

Figure 6

After the Fourier transforms, more feature extraction preprocessing techniques are typically utilized to find more information from the data. Specifically, the Mel frequency cepstral coefficients (MFCCs) are used to capture the spectral characteristics of the audio. This serves multiple purposes in the case of analyzing audio files. This is applied for dimensionality reduction, robustness to noise, is better aligned with human auditory perception, and can be used to identify information about the speaker itself. It is a powerful feature extraction technique when used correctly. The MFCC is a cepstrum in which all the frequency bands are equally spaced. This represents the perceivable sound ranges better for humans as the frequency bands are usually not distributed linearly. The number of coefficients used in the Mel filter bank should be tuned to each specific situation depending on how much granularity is desired from the transformation. Applying the MFCC transformation offers a compacted view of the audio's spectral representation, allowing for practical in-depth analysis. The MFCCs are relatively insensitive to noise variations, enabling them to be used in all uncontrolled environments. This transformation produces the spectrogram, where frequency, amplitude, and time are all represented within one graph. Fig. 7

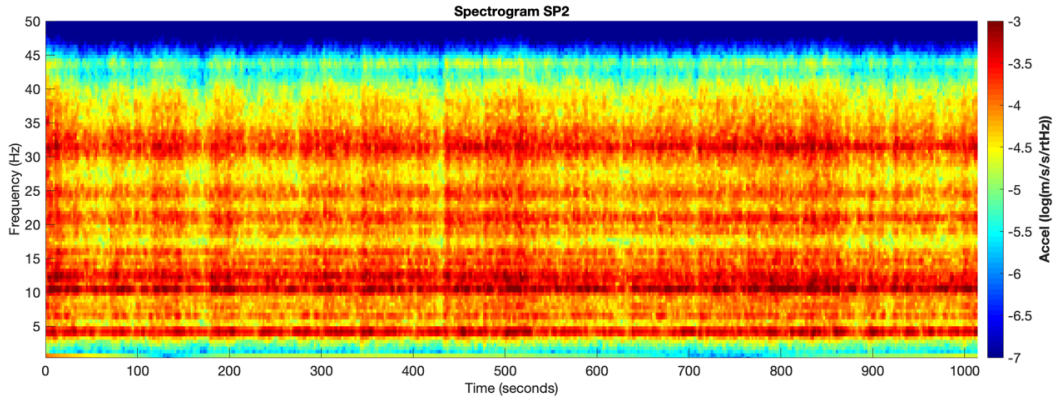

Figure 7

With some preprocessing techniques discussed and an explanation of the critical parts of Neural Networks went over; I will now go into how these networks can be applied and evaluated. Convolutional neural networks are rising in popularity due to their effectiveness at looking over data and capturing local dependencies, extracting meaningful features, and making accurate predictions. For a classification problem, the audio data needs to be acquired, typically in the form of recordings; in my experience, .wav files are better as they are less compressed and have more digital content. Once the files have been acquired, the data needs to be labeled; this is the key to the model, based on what is being labeled and when the model will treat your labels as the ground truth. The number of files and what's being labeled will differ in each situation, but it is essential to be careful when labeling the data. The labels should be named something relevant to the problem, and the number of different labels is how many classifications are involved in the problem. For a classification problem identifying traffic patterns in a large city, some relevant label names could be honking, engine idle, moving cars, person, and more. This will help you stay organized while working on the labeling. The number of files you must train on depends on

the problem. Once all the files have been labeled, the data must be preprocessed before being sent into the model and prepared.

For audio projects, the preprocessor structure is typically as follows. The windowing function is usually either a Hann or Hamming window followed by a Fast Fourier transform, then running the data through a Mel filter bank to understand the entire spectrum better and applying a logarithm function to the data. Once the preprocessor setting has been finalized, the model can be trained. Once the model has been introduced, it's time to evaluate it; typically, models will be reviewed in a confusion matrix, Fig. 8, with a couple of other metrics, such as recall and precision. Recall focuses on the completeness of optimistic predictions, and precision concentrates on the accuracy of the true positives. The confusion matrix provides an efficient way of knowing how well the model performed based on the data used to train and test it. The model can be tested with actual data if the results are optimal. In the case of a prediction model, you can feed it more files that it did not see during training to find out how it's performing against new data. If the model performs up to standards, it is ready to be used! The measures will vary from project to project as a personal project may need a different accuracy than something sold or produced on a large scale.

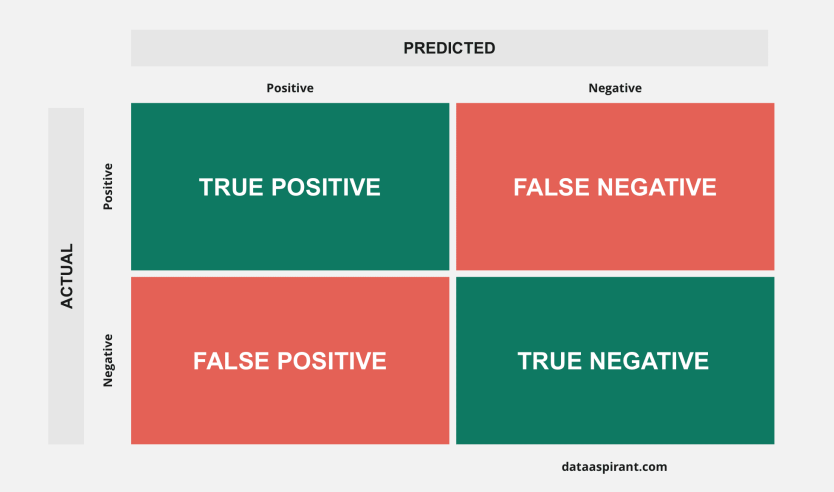

Figure 8

3. Conclusion and Future Work

In the future, I would like to develop a system that uses convolutional neural nets that would be able to detect when if something is wrong with your car while you are driving. It would be a classification problem that would need a plethora of data on how the vehicle sounds while it's going and good. My idea would be to train the model on sound and only good and look for a high value on the output. The current sound is too far from the training data if the value is too low. This would mean something is wrong; possibly, the brakes are squealing, or something is knocking in the engine.

In conclusion, I know much more about convolutional neural networks and will use this knowledge to build efficient machine-learning projects in the audio realm and more. CNNs have revolutionized how audio classification problems are solved, from their ability to learn and extract meaningful features from audio data automatically. Their versatility and

adaptability make them excellent candidates for being put into real-world situations. Further advancements in the architecture, training process, and data augmentation process will only take the field even further.

4. References

1. DeepAI. (2019, May 17). *Relu*. DeepAI. https://deepai.org/machine-learning-glossary-and-terms/relu
2. Sharma, P. (2020, April 21). *MaxPool vs avgpool*. OpenGenus IQ: Computing Expertise & Legacy. https://iq.opengenus.org/maxpool-vs-avgpool/
3. R. Ahmed, F. De Luca, S. Devkota, S. Kobourov and M. Li, "Multicriteria Scalable Graph Drawing via Stochastic Gradient Descent, $(SGD)^{2}$(SGD)2," in IEEE Transactions on Visualization and Computer Graphics, vol. 28, no. 6, pp. 2388-2399, 1 June 2022, doi: 10.1109/TVCG.2022.3155564.
4. Wolfram Alpha. (n.d.). *Fast fourier transform*. Fast Fourier Transform. https://mathworld.wolfram.com/FastFourierTransform.html