Understanding Convolution Neural Networks From A Signal Processing Perspective

Edwin Chacko
edwin.chacko@mail.utoronto.ca

December 5, 2024

Abstract

This report examines Convolutional Neural Networks (CNNs) through the lens of signal processing, demonstrating how fundamental signal processing principles underpin CNN architecture and operation. We analyze the relationship between traditional signal processing concepts and their CNN counterparts, including the distinction between convolution and cross-correlation operations, the interpretation of CNN layers as adaptive filter banks, and the parallels between sampling theory and CNN downsampling operations. Through Fourier analysis of CNN filters, we show how the hierarchical organization of CNN layers naturally aligns with frequency-based signal decomposition, with early layers capturing high-frequency components and deeper layers detecting more complex, low-frequency patterns. This signal processing perspective provides theoretical insights into CNN design choices and suggests approaches for improving CNN architectures. By bridging these two domains, we enhance our understanding of CNNs' effectiveness and provide a framework for developing more efficient and interpretable neural networks.

1 Introduction

The Convolution Neural Network (CNN) was first developed in 1988 for the MNIST handwritten digits task [1]. Since then, CNNs have revolutionized fields like image recognition, medicine, and computer vision [2]. While its success is often attributed to advanced Machine Learning (ML) algorithms and techniques, CNNs are fundamentally rooted in signal processing principles. Concepts such as convolution, filtering, sampling, and Fourier Transforms, which are foundational to signal processing, are integral to CNNs and their operations [3].

This report examines CNNs through a signal processing lens, exploring the deep connections between these two domains. We analyze how CNNs adapt and extend traditional signal processing concepts like filter banks, sampling theory, and frequency analysis to create powerful learning architectures. By studying these relationships, we gain insights into CNN design choices, interpretability, and potential improvements. The report particularly focuses on:

- The relationship between convolution and cross-correlation in CNNs

- Frequency domain analysis of CNN filters and their learning process
- The parallel between CNNs and adaptive filter banks
- Sampling and downsampling operations in CNNs and their signal processing implications

Through this analysis, we demonstrate how understanding CNNs from a signal processing perspective can inform better architecture design and provide theoretical justification for various CNN techniques.

2 Background

2.1 Signal Analysis

2.1.1 Convolution

In signal processing, the convolution of a discrete time signal $x[n]$ with another $y[n]$ is given by [4]:

$$(x * y)[n] = \sum_{k=-\infty}^{\infty} x[k]y[n-k] \tag{1}$$

This equation defines the convolution operation as the integral of the product of one function with a time-reversed and shifted version of another function. CNNs use cross-correlation which is similar to convolution, except there is no time-reversal of k in the second function. The equation for cross-correlation is given by [4],

$$\phi_{xy}[n] = \sum_{m=-\infty}^{\infty} x[m]y[m+n] \tag{2}$$

2.1.2 Fourier Transform

The Fourier Transform of a continuous-time signal $x[n]$ is defined as [4]:

$$X[k] = \sum_{n=-\infty}^{\infty} x[n]e^{-j2\pi kn/N} \tag{3}$$

The Inverse Fourier Transform to recover $x[n]$ from its frequency-domain representation $X[k]$ is given by [4]:

$$x[n] = \frac{1}{N}\sum_{k=-\infty}^{\infty} X[k]e^{j2\pi kn/N} \tag{4}$$

2.1.3 Filter Banks

In signal processing, filter banks decompose signals into multiple frequency components, enabling analysis across different spectral bands. A filter bank consists of multiple bandpass

filters $H_k(\omega)$, each isolating specific frequency components of the input signal. The response $Y_k(\omega)$ for each filter is given by:

$$Y_k(\omega) = H_k(\omega) \cdot X(\omega) \tag{5}$$

In the discrete-time domain, this becomes:

$$y_k[n] = x[n] * h_k[n] \tag{6}$$

where $h_k[n]$ is the impulse response of the kth filter and $*$ denotes convolution. For more information on filter banks, see [5]

2.1.4 Convolution Theorem

Let $x[n]$ and $h[n]$ be two discrete-time signals with respective Discrete-Time Fourier Transforms (DTFTs) $X(\omega)$ and $H(\omega)$. The convolution theorem states:

1. If $y[n]$ is the convolution of $x[n]$ and $h[n]$. Then the DTFT of $y[n]$ is given by,

$$Y(\omega) = X(\omega) \cdot H(\omega), \tag{7}$$

   where $X(\omega) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}$ and $H(\omega) = \sum_{n=-\infty}^{\infty} h[n]e^{-j\omega n}$.

2. If

$$Y(\omega) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\omega')H(\omega - \omega')\, d\omega',$$

   then the corresponding time-domain signal $y[n]$ is given by:

$$y[n] = x[n] \cdot h[n]. \tag{8}$$

2.1.5 The Sampling Theorem

In signal processing, sampling converts continuous-time signals to discrete-time signals through periodic measurements. The Sampling theorem states that to accurately reconstruct a bandlimited signal, the sampling frequency must be at least twice the highest frequency component in the signal [4]:

$$\omega_s \geq 2\omega_{\max} \tag{9}$$

2.2 Deep Learning

This section provides an introduction to the Deep Learning concepts relevant to this paper. For a more detailed description on any of the concepts introduced here, see [3].

2.2.1 Neural Networks

A neural network is a directed acyclic graph that takes an input vector $\mathbf{x}$ and returns and output vector $\mathbf{y}$. It consists of interconnected layers of nodes (neurons), which process input data and produce an output through a series of affine transformations. Mathematically, a neural network can be represented as a composition of functions [3].

The output $\mathbf{y}$ of a feedforward neural network with L layers is given by:

$$\mathbf{y} = f(\mathbf{x}) = f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(1)}(\mathbf{x}), \tag{10}$$

With the transformation of the l^{th} layer is given by,

$$f^{(l)}(\mathbf{z}) = \phi^{(l)}(\mathbf{W}^{(l)}\mathbf{z} + \mathbf{b}^{(l)}) \tag{11}$$

where:

- $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$: Weights and biases for layer l, respectively.
- ϕ: Non-linear activation function (e.g., ReLU, sigmoid).

The activation function is used to introduce non-linearity in the function. It can be shown that without using non-linear activations, any L layer network can be represented as a single layer one [6]. Thus, we require non-linear activation.

2.2.2 CNNs

CNNs are an adaptation of neural networks that share the same structure defined in Equation 10, but differ in the nature of the transformation applied at the l^{th} layer. A CNN is a neural network with at least one convolution layer [3]. The convolution layer consists of three stages; convolution, detection, and pooling.

Convolution Stage

This stage processes the input feature map using the learnable filters to produce an output feature map. For an input feature map $\mathbf{z} \in \mathcal{R}^{H \times W \times C}$ and a convolutional kernel $\mathbf{W} \in \mathcal{R}^{K_h \times K_w \times C \times K}$, the output feature map $\mathbf{z}'$ is given by [3]:

$$\text{Conv}(\mathbf{z}) = \mathbf{z}'_{i,j,k} = \sum_{p=1}^{K_h} \sum_{q=1}^{K_w} \sum_{c=1}^{C} \mathbf{z}_{i+p,j+q,c} \cdot \mathbf{W}_{p,q,c,k} + \mathbf{b}_k, \tag{12}$$

where:

- K_h, K_w: Height and width of the kernel.
- i, j: Spatial indices of the output feature map.
- p, q: Spatial indices of the kernel.

This stage extracts local patterns in the kernel window of the input feature map using the shared weight for that local location. It is important to note that Equation 12 differs from the definition of convolution in Equation 1 and is a variant known as cross–correlation.

Detection Stage

In the detection stage, we apply the non-linear activation function ϕ onto the output feature map $\mathbf{z}'$ [3]. The transformation is given by,

$$\mathbf{a}_{i,j,k} = \phi(\mathbf{z}'_{i,j,k}), \tag{13}$$

where $\mathbf{z}'_{i,j,k}$ is the output of the convolution stage, and $\mathbf{a}_{i,j,k}$ is the activation value.

Pooling Stage

The pooling stage performs dimension reduction of the feature map to retain the most important information and improve computational efficiency. In CNNs, max pooling and average pooling are the most commonly used pooling methods. For a pooling window of size $K_p \times K_p$, the pooling operation is [3]:

- **Max Pooling:**

$$\text{Pooling}(\mathbf{a}) = \mathbf{z}''_{i,j,k} = \max_{p,q \in \{0,\ldots,K_p-1\}} \mathbf{a}_{i+p,j+q,k}. \tag{14}$$

  This selects the maximum value in the pooling window.

- **Average Pooling:**

$$\text{Pooling}(\mathbf{a}) = \mathbf{z}''_{i,j,k} = \frac{1}{K_p^2} \sum_{p=0}^{K_p-1} \sum_{q=0}^{K_p-1} \mathbf{a}_{i+p,j+q,k}. \tag{15}$$

  This computes the average value in the pooling window.

Pooling reduces the spatial resolution and introduces translation invariance, making the feature maps more robust.

The three stages together form the convolution layer of a CNN. The same structure defined in Equation 10 applies in CNNs, however, the transformation of the l^{th} layer is computed differently,

$$f^{(l)}(\mathbf{z}) = (\text{Pooling} \circ \phi \circ \text{Conv})(\mathbf{z}) \tag{16}$$

2.2.3 Hierarchical Feature Extraction

CNNs leverage multiple convolutional layers to build hierarchical feature representations of the input signal. Early layers capture low-level features such as edges and textures, allowing deeper layers to recognize more complex patterns like shapes and objects.

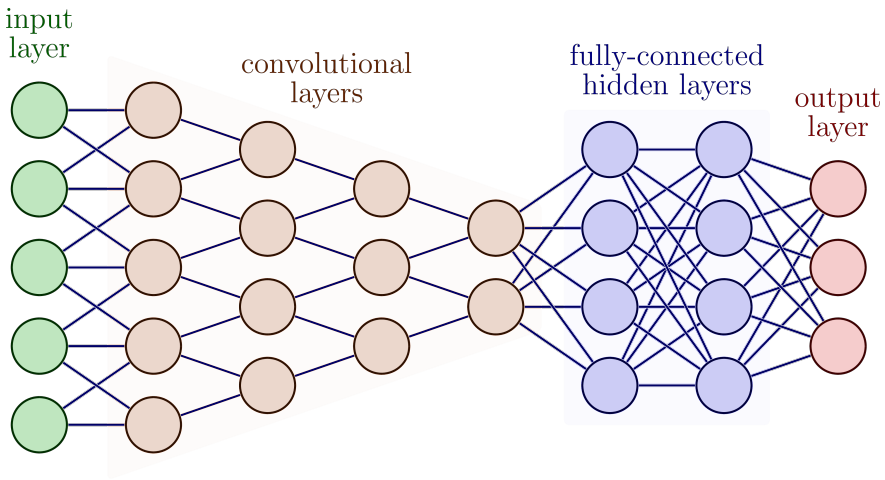

Figure 1: CNN architecture with feed-forward. [7]

Figure 1 above shows the CNN architecture with a feed-forward after then convolutional layers. The convolution layers lose a dimension each proceeding layer, demonstrating the pooling layers that perform dimensionality reduction.

3 Signal Processing Perspective of CNNs

3.1 Convolution in CNNs vs. Signal Processing

3.1.1 Convolution and Cross-correlation

In signal processing, Equation 1, restated below for convenience, is used to modify a signal by convolving it with another one, typically representing a system's impulse response. CNNs use cross-correlation which is slightly different as it does not time-reverse the summation variable k. Cross-correlation is given by Equation 2, again restated below for convenience, and is used to measure the similarity between two signals. Cross-correlation slides the filter signal over the other one and measures overlap at each step.

$$(1) \quad (x * y)[n] = \sum_{k=-\infty}^{\infty} x[k]y[n-k] \qquad (2) \quad \phi_{xy}[n] = \sum_{m=-\infty}^{\infty} x[m]y[m+n]$$

Cross-correlation is preferred to convolution in CNNs due to its simpler computational efficiency. In CNNs, the flipping of the summation variable k has the effect of flipping the kernel both spatial dimensions. Since CNNs learn the kernel itself, using the convolution would lead to the model learning the flipped kernel, which unnecessarily increases computation at each step by having to flip it. Using cross-correlation foregoes this inefficiency and results in exact same learned kernel [8].

3.1.2 Implications for Filter Learning

In signal analysis, filters are predefined and serve a specified function. To contrast, the goal in CNNs is to learn the filters themselves. Convolving a signal with a given filter applies

that filter to the signal (e.g., a low pass filter). Additionally, filters are symmetric as they must work in the same way across the entire signal. In CNNs, symmetry is not always desired as asymmetric filters are useful for learning specific features. For example, in image recognition tasks, where the filter is attempting to learn horizontal or vertical edges, here imposing symmetry is detrimental to the task.

3.2 Frequency Domain Analysis of CNN Filters

3.2.1 Fourier Transform and Convolution Theorem

The convolution theorem is a fundamental principle in signal processing, stating that convolution in the time (or spatial) domain is equivalent to multiplication in the frequency domain. For discrete signals $x[n]$ and $h[n]$ the theorem is expressed as:

$$(x * h)[n] \leftrightarrow X(\omega) \cdot H(\omega),$$

where $X(\omega)$ and $H(\omega)$ are the Fourier Transforms of $x[n]$ and $h[n]$, respectively.

This property is often utilized in signal processing to perform filtering efficiently in the frequency domain using the Fast Fourier Transform (FFT).

In Convolutional Neural Networks (CNNs), while convolution is performed in the spatial domain, the learned filters implicitly shape the frequency response of the network. Applying the Fourier Transform to these filters reveals their spectral characteristics, providing insights into their role in feature extraction.

3.2.2 Frequency Analysis of Learned Filters

Visualization of Filter Spectra The frequency response of a CNN filter can be analyzed by taking its 2D Fourier Transform. This visualization helps identify the frequency components that the filter is most sensitive to:

- High-frequency components in a filter correspond to features like edges and fine textures.
- Low-frequency components detect smoother, more generalized patterns, such as shapes or gradients.

For example, a filter with strong energy in high frequencies is effective at detecting sharp edges, while a filter with dominant low-frequency components captures broader, smoother features. Visualizing these spectra using heatmaps or surface plots provides an intuitive understanding of the filter's behavior.

Bandwidth and Generalization of Filters The bandwidth of a filter, or the range of frequencies it responds to, significantly affects its performance:

- Narrowband filters specialize in detecting specific patterns or localized features.
- Wideband filters are more generalized, capturing diverse features across a broader range of inputs.

In CNNs, the ability to generalize is closely tied to the balance between low- and high-frequency characteristics of the filters. Networks trained on diverse datasets often learn filters with a mix of narrow and wide bandwidths, enabling them to robustly detect patterns across varying data distributions. Overemphasis on high-frequency components, however, can lead to overfitting to noise or dataset-specific details, reducing generalization performance.

3.3 Filter Bank Analysis of CNNs

3.3.1 CNNs as Adaptive Filter Banks

Filter banks are a fundamental concept in signal processing, where they decompose signals into multiple frequency components using a set of bandpass filters. Each filter isolates specific spectral bands, enabling detailed analysis across frequencies (as introduced in 2.1.3).

CNNs extend this concept by implementing adaptive filter banks through their convolutional layers. While traditional filter banks use predetermined filters, CNNs learn their filter coefficients from data, making them remarkably flexible for various signal processing tasks. CNNs excel in signal processing tasks and can directly process raw signals and achieve high performance in applications such as ECG classification, structural health monitoring, and motor-fault detection [9]. The number and arrangement of these filters can be optimized for specific applications, allowing CNNs to automatically discover relevant features in the data.

Recall Equation 1 which defines the output for a CNN layer with K filters, each of size $K_h \times K_w \times C$. This parallel with filter banks explains why careful selection of kernel sizes and layer architecture significantly impacts a CNN's ability to capture different frequency components.

3.3.2 Frequency Response and Feature Extraction

The frequency response of CNN filters reveals a hierarchical structure that mirrors classical signal processing theory [10]. This structure has important implications for CNN architecture design [11]:

- Low-frequency filters capture broad patterns and gradual variations, suggesting deeper layers should have larger receptive fields
- High-frequency filters detect edges and fine details, explaining why early layers typically use smaller kernels
- Mid-frequency filters identify textures and intermediate-scale patterns, often requiring a balanced distribution of filter sizes

This spectral hierarchy explains why CNNs achieve effective feature extraction: the training process optimizes each layer's filters to capture relevant frequency components at different scales. Early layers learn high-frequency edge detectors, while deeper layers develop into complex, lower-frequency pattern recognizers. This organization aligns with the signal processing theory covered in ECE355, demonstrating how fundamental principles of filter design inform modern deep learning architectures.

The learning process itself can be understood as optimizing a complex filter bank, where each training iteration refines the filters' frequency responses. This perspective provides insight into why CNNs excel at tasks like image recognition and signal classification—they automatically learn optimal filter responses across different frequency bands, adapting to the specific characteristics of the input data.

This connection to filter bank theory not only explains the success of CNNs in various signal processing applications but also provides a theoretical framework for improving their design and performance. By understanding CNNs through this lens, we can make more informed decisions about architecture choices and better predict how modifications will affect the network's ability to process different types of signals.

3.4 Sampling and Downsampling in CNNs

3.4.1 Sampling in Signal Processing vs CNNs

In signal processing, sampling converts continuous-time signals to discrete-time signals through periodic measurements. Recall the sampling theorem 9.

CNNs employ analogous operations through striding and pooling layers. These operations reduce spatial dimensions while attempting to preserve essential information, similar to downsampling in signal processing.

3.4.2 Striding and Pooling Operations

The convolution operation in CNNs can be modified by introducing stride, which determines how the kernel moves across the input. For a stride S, the output dimension Y of a convolution layer with input dimension X and kernel size K is given by:

$$Y = \left\lfloor \frac{X - K}{S} \right\rfloor + 1 \tag{17}$$

This relationship shows how increasing stride reduces output dimensions, similar to decreasing sampling rate in signal processing. The pooling operation, introduced in Section 2.2.2, performs a similar dimension reduction but uses operations like max or average to summarize local regions.

3.4.3 Aliasing Effects in CNNs

Just as undersampling in signal processing leads to aliasing, aggressive striding or pooling in CNNs can cause information loss and artifacts. The frequency interpretation helps explain this phenomenon:

1. When stride exceeds the spatial frequency of features in the input, high-frequency components can be misrepresented as lower frequencies

2. This aliasing manifests as:

    - Loss of fine texture details

- Misidentification of patterns
- Reduced accuracy in feature detection

3.4.4 Anti-aliasing in CNNs

To mitigate aliasing effects, CNNs can incorporate techniques inspired by signal processing:

1. Progressive downsampling: Using multiple layers with smaller strides instead of a single layer with large stride
2. Blur pooling: Applying a low-pass filter before downsampling operations
3. Learned downsampling: Allowing the network to adapt its downsampling behavior through training

These approaches help maintain the balance between computational efficiency and information preservation, similar to anti-aliasing filters in traditional signal processing.

3.4.5 Frequency Analysis of Downsampling

The effect of striding and pooling can be understood through the convolution theorem discussed in Section 3.2. When downsampling by a factor of S, the frequency spectrum of the signal is scaled by S in the frequency domain:

$$Y(\omega) = X(S\omega) \tag{18}$$

This relationship helps explain why gradual downsampling through multiple layers often performs better than aggressive downsampling in a single layer, as it allows for better control of the frequency content at each stage.

3.4.6 Implications for CNN Design

Understanding the link between sampling theory and CNN downsampling highlights the trade-off between computational efficiency and information retention. Techniques like progressive downsampling, blur pooling, and learned downsampling mitigate aliasing, preserving critical features while reducing spatial dimensions. These strategies are crucial for applications requiring fine detail, such as medical imaging, or real-time efficiency, like autonomous vehicles. Incorporating signal processing-inspired methods into CNNs improves robustness, accuracy, and task-specific performance.

4 Conclusion

This report has examined Convolutional Neural Networks (CNNs) through the lens of signal processing, revealing the deep theoretical connections between these two domains. Our analysis demonstrates how CNNs naturally extend and adapt fundamental signal processing principles in their architecture and operation. Through exploring key concepts like convolution, Fourier analysis, filter banks, and sampling theory, we've shown how CNNs can

be understood as implementing learnable, adaptive filter banks where different layers naturally organize to capture various frequency components of the input signal. The choice of cross-correlation over convolution represents a practical optimization that preserves learning capabilities while reducing computational overhead, while the hierarchical structure of CNN feature extraction mirrors classical signal processing theory, with early layers detecting high-frequency components and deeper layers capturing more complex, lower-frequency patterns.

These connections between signal processing and CNNs not only enhance our theoretical understanding but also provide practical insights for improving CNN architectures. The analysis of sampling operations in CNNs through traditional sampling theory helps us better understand and mitigate potential information loss, while Fourier analysis of learned filters provides insights into their role in feature extraction. Looking forward, these fundamental connections could guide the development of more efficient and interpretable neural network architectures, suggesting that continued exploration of signal processing principles may lead to further advances in deep learning design.

References

[1] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] S. Müksch, T. Olausson, J. Wilhelm, and P. Andreadis, "Quantitative analysis of image classification techniques for memory-constrained devices," *CoRR*, vol. abs/2005.04968, 2020. [Online]. Available: https://arxiv.org/abs/2005.04968

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[4] A. Oppenheim, A. Willsky, and S. Nawab, *Signals & Systems*, ser. Prentice-Hall signal processing series. Prentice-Hall International, 1997. [Online]. Available: https://books.google.ca/books?id=O9ZHSAAACAAJ

[5] S. Sarangi, M. Sahidullah, and G. Saha, "Optimization of data-driven filterbank for automatic speaker verification," *Digital Signal Processing*, vol. 104, p. 102795, Sep. 2020.

[6] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0893608089900208

[7] I. Neutelings, "Neural networks," 2021, accessed: 2024-11-20. [Online]. Available: https://tikz.net/neural_networks/

[8] L. Mao, "Convolution vs. cross-correlation," 2021, accessed: 2024-11-20. [Online]. Available: https://leimao.github.io/blog/Cross-Correlation-VS-Convolution/

[9] S. Kiranyaz, T. Ince, O. Abdeljaber, O. Avci, and M. Gabbouj, "1-d convolutional neural networks for signal processing applications," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 8360–8364.

[10] S. Fujieda, K. Takayama, and T. Hachisuka, "Wavelet convolutional neural networks for texture classification," 2017. [Online]. Available: https://arxiv.org/abs/1707.07394

[11] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," 2013. [Online]. Available: https://arxiv.org/abs/1311.2901