

Received 7 June 2022, accepted 13 July 2022, date of publication 19 July 2022, date of current version 26 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3192387

RESEARCH ARTICLE

Aliasing-Free Nonlinear Signal Processing Using Implicitly Defined Functions

EMMY S. WEI 

Evergreen Valley High School, San Jose, CA 95148, USA

e-mail: emmy.s.wei@gmail.com

ABSTRACT Digital signal processing relies on the Nyquist-Shannon sampling theorem that applies to and requires a continuous signal with limited bandwidth. However, many systems or networks of signal processing involve nonlinear functions, which could generate new frequency components beyond the original bandwidth and lead to aliasing. Indeed, aliasing-induced shift variance has long been a nuisance and unsolved problem in convolutional neural networks and has recently been found to severely impair the performance of machine learning applications. The same problem exists in other fields such as computational lithography. In this paper, a new method and algorithms are introduced to solve the problem of aliasing induced by nonlinear functions involving operations other than linear convolutions and pointwise multiplications. Said new method and algorithms employ implicitly defined functions that are implemented via iterations of polynomial operations so that aliasing is completely avoided by upsampling signals before polynomial operations and limiting signal spectra before downsampling. Theoretical analyses and exemplary algorithms are presented to implement nonlinear functions commonly used in signal processing networks. In particular, exemplary embodiments and numerical experiments are reported to illustrate and verify aliasing-free operations of Wiener-Padé approximants, which are already universal in their ability to approximate any continuous activation functions to the desired accuracy.


INDEX TERMS Convolutional nonlinear networks, aliasing, translation invariance, shift-invariance, implicitly defined functions, iterative multiplications, spectrum control.

I. THE PROBLEM OF ALIASING IN NONLINEAR SIGNAL PROCESSING

The advent and wide applications of modern digital computers have made computing or signal processing ubiquitous and indispensable in industries, scientific researches, and daily lives. Many computational or signal processing systems involve a number of functional steps or function applications, interchangeably called signal processing modules or stages, each of which applies a predetermined function on an input signal or image and transforms it into an output signal or image. Such functional steps or signal processing modules are interconnected to constitute a signal processing network, or just a network in short. In this paper, the terms signal and image are used interchangeably and refer to a physical

quantity, or its digitized representation on a computer, that is distributed in a region of space, time, or spacetime.

In most applications, such a physical quantity of concern is a continuous and smooth distribution in spacetime, and in many cases, has its spatial or temporal spectral contents supported by a compact set in the corresponding frequency domain, thus being called a band-limited signal or image, for which, the largest separation between two spectral points in the compact frequency support along a specific frequency axis is called the bandwidth of the image or signal along the specified frequency axis. Thanks to the celebrated Nyquist-Shannon sampling theorem, a band-limited signal can be represented by and fully reconstructed from a discretely sampled version of the original signal, so long as the sampling rate is beyond the so-called Nyquist rate [1]. Moreover, any linear translation-invariant (LTI) function or system [1] transforms a band-limited input signal into a band-limited output signal with a compact support that is the

The associate editor coordinating the review of this manuscript and approving it for publication was Cesar Vargas-Rosales .

same as or within the compact support of the input signal. Therefore, an LTI network consisting of only LTI signal processing modules can have all of the input, intermediate, and output signals discretely sampled at the same rate, which can represent and reconstruct the corresponding continuous and smooth distributions of physical quantities without loss of information or accuracy. This is the fundamental principle behind the wide applications of digital signal or image processing in modeling and simulating real physical, chemical, or engineering processes on computers.

Unfortunately, not all signal processing tasks can be done by linear functions. Many signal transformations involve nonlinear operations. In fact, a certain nonlinearity is necessary in order for a signal processing network to be a universal approximator, able to approximate any continuous or (Lebesgue) integrable function arbitrarily well as the width or depth of the network grows. Without loss of generality, this paper is focused on the so-called *convolutional nonlinear networks* (CNNs), which comprise interconnected stages of LTI signal transformations (*i.e.*, convolutions) and spatiotemporally on-site nonlinear operations. Archetypical CNNs include the classical Volterra-Wiener systems [2]–[4] as well as the recently fashionable convolutional neural networks [5], [6]. In such CNNs, a typical nonlinear operation on an originally band-limited signal will generally generate new spectral components beyond the presumed compact frequency support of the original signal, so that the original sampling rate sufficient for the input signal becomes insufficient for the output signal. In terms of digital signal processing (DSP) on a computer, applying a nonlinear function pointwise to each pixel of an image array representing a discretely sampled input signal produces an output image array that no longer represents and is no longer able to reconstruct the would-be continuous and smooth distribution as the ideal result of nonlinearly transforming the continuous and smooth input signal. Rather, when the Nyquist-Shannon sampling theorem is invoked, such an output image array reconstructs and represents a distorted version of the ideal continuous and smooth output signal, leading to the so-called aliasing problem due to spectrum rollback and overlapping [1].

One detrimental effect of nonlinearity-induced aliasing is a loss of symmetry that is inherent in the physical signal and system. Translation invariance, or called shift-invariance, is one important symmetry but is easily lost due to aliasing. The convolutional neural networks, widely and successfully applied in machine learning, especially deep learning applications, are designed to embed the property of shift-invariance through its explicit incorporation of convolutions. Unfortunately, they also employ activation functions such as rational functions, sigmoid, hyperbolic tangent, ReLU, Swish, max pooling, Softmax, and SoftPool, *etc.* [7]–[12], which are highly nonlinear and create high-frequency contents beyond the sampling rates of the signals or images, resulting in aliasing and shift variance. Recent studies have shown that image classification and speech recognition systems based on

convolutional neural networks are not really shift-invariant, and a small coordinate shift to input images or signals can induce significant degradation and fluctuations in the classification or recognition performances. In refs. [13]–[15], it has been found that a slight spatial shift of input images, as small as a single pixel and indiscernible to human eyes, can lead to significantly different classification outputs in pattern recognition applications using multi-layered convolutional neural networks. Generally, it is understood that such shift variance is a result of aliasing due to the nonlinear activation functions in the neural networks. Ref. [16] demonstrated the impact of aliasing on audio and speech processing via deep convolutional neural networks. Countermeasures such as low-pass filtering before nonlinear operations have been proposed and tested in applications [17], [18], which may mitigate the problems of aliasing but not remove them completely. A consensus is that “the problem of insuring invariance to small image transformations in neural networks while preserving high accuracy remains unsolved” [15], [19].

Other striking examples come from the field of computational lithography, where the Volterra-Wiener type of models [20]–[22] and convolutional neural networks [23]–[28] have been employed to simulate the diffusive and nonlinear process of photochemistry, which turns an optical image of photoexposure in a photoresist material into a 3D topography of a developed photoresist. There, the optical image is sampled at near or slightly over the Nyquist rate determined by a bandwidth limit of the optical imaging system, with pixels sized on the order of tens of nanometers. While the process of photochemistry is highly nonlinear to induce seemingly unavoidable aliasing, such that a minuscule coordinate shift by a small fraction of the pixel size could induce a significant change in the results, manifesting themselves in lithographic patterns having size or position errors approaching even exceeding a nanometer, becoming unacceptable in advanced high-density lithography processes. It is now widely recognized that aliasing-induced shift variance poses a fundamental challenge to computational lithography using CNNs.

To summarize, aliasing in nonlinear signal processing affects many practical applications and is a fundamental problem in pressing need of a solution. In the past, methods such as upsampling an input signal at a higher rate than its frequency bandwidth before applying a nonlinear operation and then downsampling the output signal, have been applied and proven useful in combating aliasing-induced signal distortions. Indeed, for a nonlinear operation that is associated with a polynomial function, such as those incorporated in a Volterra-Wiener model truncated to a finite order, upsampling input signals by a factor equal to or above the degree of the polynomial could have aliasing completely avoided. However, many CNNs involve nonlinear functions that are non-polynomial, for which, although upsampling does help to reduce aliasing-induced problems, no amount of upsampling could have aliasing completely avoided, so there is always a vestigial distortion. Worse yet, in a large CNN having multiple nonlinear operations cascaded in series, there is a

risk of multiple stages of aliasing distortions compounding and snowballing exponentially into a large and unacceptable error. Such an exponential growth of aliasing-induced errors is a new form of numerical instability, which could severely limit the sustainable depth of a useful nonlinear network.

In the absence of a known theory or method to unravel signal distortions after several aliasing stages, the only possibility of saving a CNN from serious aliasing-induced signal distortions seems to be making each nonlinear operation aliasing-free or aliasing-reduced very efficiently at an affordable increase in computational cost. A greedy objective is to have aliasing-induced errors reduced to exponentially small at a linearly increased computational cost. Therefore, it is a theoretically interesting and practically important question whether non-polynomial signal operations can be implemented or well approximated in such a manner that aliasing-induced signal distortions are avoided or exponentially suppressed, while the incurred computational cost increases only linearly. In this paper, I propose, test, and demonstrate a new method of realizing a nonlinear operation via an implicitly defined function, whose algorithmic implementation involves an iterative solver to compute an output signal through numerical iterations, where each iteration involves only polynomial functions. It also employs upsampling, spectral limiting, and downsampling to ensure aliasing-free operations, such that the overall iterative solver and the produced output signal are aliasing-free. A CNN that incorporates such implicitly defined functions and operates in an aliasing-free manner is called an implicit convolutional nonlinear network (iCNN).

II. CONTINUOUS AND DISCRETE SIGNALS, THE NYQUIST-SHANNON SAMPLING THEOREM, ALIASING AND SHIFT VARIANCE

Let an interested region of spacetime be coordinated by a variable vector $x \in \mathbb{R}^d$, $d \in \mathbb{N}$ and $I(x)$ denote an interested image or signal over the spacetime region. For any given coordinate shift $\delta x \in \mathbb{R}^d$, let $T_{\delta x}$ denote a spacetime *translation operator* such that $[T_{\delta x}I](x) \stackrel{\text{def}}{=} I(x - \delta x)$, with $T_{\delta x}I$ representing the coordinate-shifted version of I . It is easy to see that for any $\delta x \in \mathbb{R}^d$, the operator $T_{\delta x}$ is invertible and $T_{\delta x}^{-1} = T_{-\delta x}$.

Definition 1: A function f on continuous signals is called *translation-invariant*, or *shift-invariant*, if $f(T_{\delta x}I) = T_{\delta x}[f(I)]$ holds for any legitimate input signal I and for any coordinate shift δx .

For a given $\delta x \in \mathbb{R}^d$, let $f_{\delta x} \stackrel{\text{def}}{=} T_{\delta x}^{-1}fT_{\delta x} \stackrel{\text{def}}{=} T_{\delta x}^{-1} \circ f \circ T_{\delta x}$ denote a composite function that shifts an input signal forward by δx , then applies the function f , and finally shifts the resulted signal backward by $-\delta x$. Equivalently, a function f is translation-invariant if and only if $f_{\delta x} = f$ holds for any $\delta x \in \mathbb{R}^d$. An LTI function or system, as mentioned above, is both linear and translation-invariant. A single step of signal processing, or a system having many steps of signal processing as a whole, being represented by an input-output function f , is not translation-invariant, or said to violate the

symmetry of translation-invariance, when there exists one legitimate input image I and one coordinate shift vector $\delta x \in \mathbb{R}^d$, such that $f_{\delta x}(I) \neq f(I)$.

The celebrated Nyquist-Shannon sampling theorem [1] states that if a continuous signal in spacetime is band-limited, then it can be sampled into a discrete signal at a sampling rate faster than the so-called Nyquist rate, and such continuous-to-discrete (also called analog-to-digital) signal conversion is information lossless, in the sense that, whenever needed, the original continuous signal can be fully recovered from the sampled discrete signal using the Whittaker-Shannon interpolation formula [1]. The Nyquist-Shannon sampling theorem laid down the foundation for DSP and the subsequent information revolution and information age. In practice, a DSP algorithm or device constitutes a *DSP system*, which processes a discrete/digital signal sampled from a continuous/analog signal at a predetermined sampling rate, referred to as the *DSP sampling rate* hereafter.

To analyze a DSP system, let \mathcal{S} denote the operation of sampling a continuous, smooth, band-limited signal I into a discrete image array $J = \mathcal{S}(I)$, and conversely, let \mathcal{S}^* represent the reverse operation of reconstructing a continuous, smooth, band-limited signal $I' = \mathcal{S}^*(J')$ from a discrete image array J' using the Whittaker-Shannon interpolation formula. Note that the operator composition $\mathcal{S}\mathcal{S}^* \stackrel{\text{def}}{=} \mathcal{S} \circ \mathcal{S}^*$ always reduces to an identity operator $\mathbb{1}$, transforming any discrete image array back to itself, while $\mathcal{S}^*\mathcal{S} \stackrel{\text{def}}{=} \mathcal{S}^* \circ \mathcal{S}$ maps identically only for band-limited continuous and smooth signals and when the \mathcal{S} and \mathcal{S}^* operations are done at a sufficient DSP sampling rate. It is another manifest of aliasing that $\mathcal{S}^*\mathcal{S}$ does not reduce to the identity operator $\mathbb{1}$ when acting on signals having excessive spectral contents.

A DSP system represented by a function g on digital signals takes as input a discrete image array $\mathcal{S}(I)$ that is sampled from a continuous, smooth, band-limited signal I , and transforms $\mathcal{S}(I)$ into an output discrete image array $g(\mathcal{S}(I))$, which would reconstruct and represent a continuous, smooth, band-limited signal $\mathcal{S}^*(g(\mathcal{S}(I)))$ per the Nyquist-Shannon sampling theorem, where the \mathcal{S} and \mathcal{S}^* operations are performed at the same DSP sampling rate. In other words, such a DSP algorithm or system represented by a function g on digital signals implements an effective composite function $f = \mathcal{S}^*g\mathcal{S} \stackrel{\text{def}}{=} \mathcal{S}^* \circ g \circ \mathcal{S}$, whose domain and range are two spaces of continuous signals or images that may or may not be the same.

Definition 2: A DSP system represented by a function g on discrete signals is called *translation-invariant*, or *shift-invariant*, if and only if the composite function $\mathcal{S}^*g\mathcal{S}$ is translation-invariant over the space of band-limited signals, whose bandwidth is below the DSP sampling rate.

Specifically, the Nyquist-Shannon sampling theorem requires that a continuous signal in time (using a signal in time just for example) containing a maximum frequency component at F_{\max} Hz must be sampled at a rate of F_s samples/second, with $F_s > 2F_{\max}$, in order for the sampled discrete signal to be aliasing-free and able to reconstruct

the original continuous signal. Otherwise, if a continuous signal contains a spectral component that oscillates faster than $F_s/2$ Hz, or a discrete signal has gone through a nonlinear operation creating new spectral components with a frequency exceeding $F_s/2$ Hz, then the discrete signal could suffer from aliasing and experience signal distortions such as shift variance. A shift variance detector would be useful to monitor and measure such an aliasing-induced problem. The following algorithm provides an exemplary embodiment.

Algorithm 1 (Shift Variance Detection)

- 1.1 Receive a function to be tested, a test signal as input, and a spacetime shift;
 - 2.2 Apply said function to the test signal and obtain the first output;
 - 3.3 Shift the test signal by the spacetime shift and obtain a shifted input;
 - 4.4 Apply said function to said shifted input and obtain a shifted output;
 - 5.5 Shift the shifted output backward by the spacetime shift and obtain a second output;
 - 6.6 Compute and report a difference between said first output and said second output.
-

The example above is a straightforward algorithm for detecting violations of shift-invariance, which literally follows the definition of shift-invariance. It should be noted that the spacetime shift is often a fraction of a spacetime pixel, so the operation of shifting the signals or images in spacetime is implemented using fast Fourier transforms (FFTs) by virtue of the “(space)time shifting” property of the Fourier transform [1], namely, a spacetime signal or image is first transformed to the spectral domain via FFT, then each frequency component is phase-modulated by a phasor depending on the frequency and the spacetime shift, and finally an inverse FFT is applied to transform the spectral signal or image into a shifted signal or image in spacetime. Another useful note is on the number to report for the difference between the first and second outputs in Algorithm 1, whose customary choice is an L^1 , L^2 , or L^∞ norm of the difference signal or image between the two outputs. In the following, the L^∞ norm of the difference signal or image will be used as an indicator of shift variance, which is simply the maximum of the absolute values among all of the spacetime sampling points of the difference signal or image. As a definitive measure of shift variance, a number called the *relative shift variance* (RSV) will be used, which is defined as the ratio between the L^∞ norm of the difference signal or image and the L^∞ norm of the average of the first and second outputs.

An optical image from a computational lithography simulator is chosen as a test image, which is a 128×128 two-dimensional (2D) array of discrete samples from a band-limited optical intensity distribution, with the spacing between adjacent pixels, known as the *grid size*, being 50 nm in both the horizontal and the vertical directions. The test image is shown in Fig. 1 (a), where the colormap has

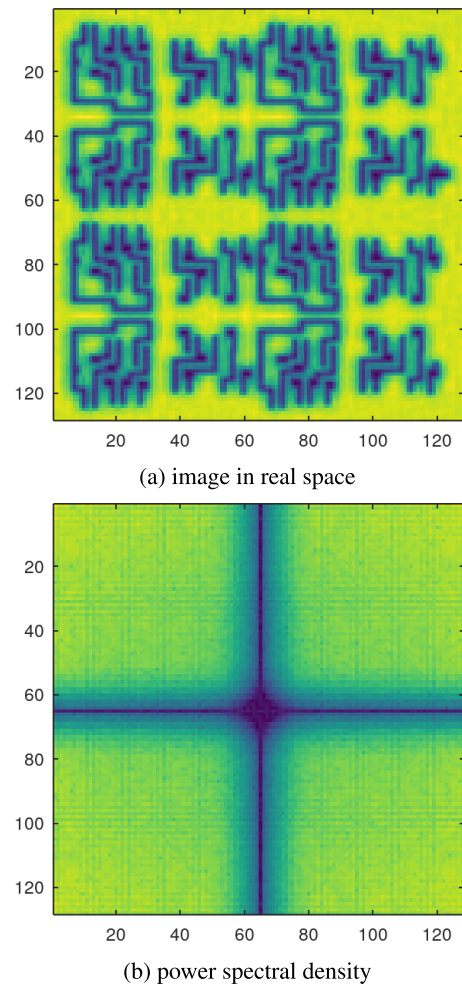


FIGURE 1. The original test image in real space and its power spectral density.

bright yellow representing high-intensity values close to 1 (in arbitrary unit) and dark blue indicating low-intensity values near 0. Fig. 1 (b) shows the power spectral density of the test image in log scale, displaying the logarithm of the absolute value squared of the amplitudes of the spectral components, where the colormap has bright yellow representing large positive values indicating a relatively high power density and dark blue representing large negative values indicating a vanishingly small power density. From Fig. 1 (b), it can be clearly seen that the test image has been sampled at a DSP sampling rate slightly higher than the Nyquist rate, such that its spectral image has a small margin with vanishing amplitudes for the highest frequency components. The same colormaps will be used in the following for images in the 2D real space and the spectral domain respectively, without repeating how they should be interpreted.

Said computational lithography simulator models a lithography process where a photomask containing designed patterns is illuminated by a light source, the light transmitted from the photomask is focused by a projection lens and forms an optical image on a silicon wafer coated with a photoresist,

triggering complicated photochemical reactions in the photoresist, which later develops into desired topography that selectively covers or exposes areas in the underlying silicon wafer. The optical image is band-limited by fundamental physics because the light wave has a finite wavelength and the projection lens can only receive and transmit light beams at a finite angle. Therefore, the optical image can be properly sampled into a discrete 2D array without aliasing. By contrast, a highly nonlinear function is needed to model the photochemical process in the photoresist, which generates a lot of high-frequency signal components beyond the Nyquist frequency, even though the input optical image is band-limited and moderately upsampled. In particular, a so-called Padé approximant [29] as a rational function, namely, a quotient between two polynomial functions, is particularly suitable for modeling the sigmoid curve-like response of a photoresist, which has the property of suppressing low-intensity image values further down toward 0 while saturating toward the maximum value 1 for high-intensity image values.

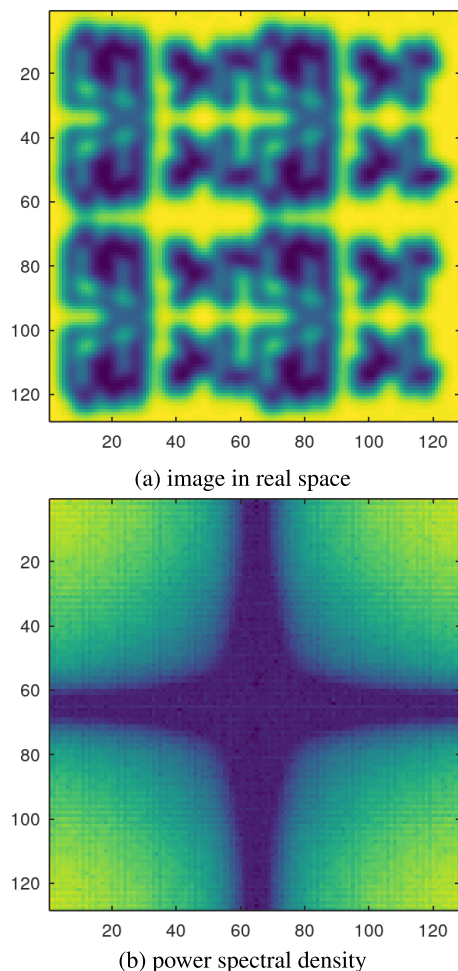


FIGURE 2. The Gauss-blurred image in real space and its power spectral density.

For a smoke test to check the methodology and utilities coded in Octave using double precision floating point

numerics, a shift variance detector is applied to an LTI function that simply blurs an input image by convolving it with a Gaussian kernel $(2\pi\sigma^2)^{-1} \exp[-(x^2 + y^2)/2\sigma^2]$, $\sigma = 2$ pixels, namely, 100 nm in the real space. Using the test image of Fig. 1 as input, the Gauss-blurred image and its power spectral density are shown in Fig. 2 (a) and (b) respectively, with Fig. 2 (b) showing that the margins around the high-frequency spectral regions with near-zero amplitudes are significantly widened in comparison to Fig. 1 (b), as a result of the Gaussian filter heavily attenuating the high-frequency signal components. Theoretically, such a Gauss-blurring function should induce no aliasing and suffer no shift variance. Indeed, when a typical shift vector $(\delta x, \delta y) = (0.3, 0.4)$ pixel, namely, $(\delta x, \delta y) = (15, 20)$ nm in the real space is used to forward-shift the input image and backward-shift the output image, an RSV value of 1.35×10^{-15} is obtained, which is on the order of numerical rounding errors. Repeated experiments with many other randomly chosen shift vectors $(\delta x, \delta y)$ all produce similar, vanishingly small RSV values on the order of 10^{-15} , indicating the absence of shift variance associated with the example Gauss-blurring function, just as shift-invariance is theoretically expected for any LTI function.

III. CLASSICAL METHODS OF ALIASING AVOIDANCE BY UPSAMPLING

Given a $(d \in \mathbb{N})$ -dimensional continuous image I' having a finite bandwidth $W - \epsilon$, with $W > 0$, $\epsilon > 0$, $\epsilon \ll W$ being an arbitrarily small number, suffice it to sample I' at the Nyquist rate W into a discrete image I , whose domain in the frequency space, denoted by $\text{FDom}(I) \subseteq \mathbb{Z}^d$, has a width W in each of the d frequency axes. The set of frequency points on which I assumes a nonzero spectral amplitude is called the support of I in the frequency space, denoted by $\text{FSupp}(I)$, which is a subset of $\text{FDom}(I)$. Many applications choose to oversample I' at a higher rate uW , $u > 1$, yielding a so-called upsampled discrete image J with an enlarged domain $\text{FDom}(J) = u \text{FDom}(I)$ in the frequency space, even though the spectral amplitudes of J vanish in the set $\text{FDom}(J) \setminus \text{FDom}(I)$. Such oversampling serves the purpose of either better spatial resolution or anti-aliasing in a subsequent operation.

Definition 3: A discrete image I is said to be upsampled into a discrete image J , conversely, J is said to be downsampled into I , when $\text{FDom}(J) = u \text{FDom}(I)$, I and J correspond to the same continuous signal, that is, $S^*(J) = S^*(I)$. The number $u > 1$ is called an upsampling factor (USF).

A simple application of Fourier analysis asserts that the discrete I is upsampled into the discrete J if and only if $\text{FSupp}(J) = \text{FSupp}(I) \subseteq \text{FDom}(I)$ and J and I assume identically the same spectral amplitudes on $\text{FSupp}(J)$. Operationally, to upsample a discrete image I by a USF $u > 1$, firstly apply an FFT to I if it is not already in the frequency space, then create an all-zero image J in the frequency space with $\text{FDom}(J) = u \text{FDom}(I)$, then copy the spectral amplitudes of I in $\text{FSupp}(I)$ to the corresponding

frequency points in $\text{FDom}(J)$, finally apply an inverse FFT to J if it is desired to be in the spacetime representation. Conversely, to downsample a discrete J into a discrete I , with $\text{FDom}(I) = \text{FDom}(J)/u$, just copy the spectral amplitudes within the set $\text{FDom}(I)$ from J to I when both are in the frequency space, and discard the high-frequency components beyond $\text{FDom}(I)$. Alternatively, when u is an integer, J is *spectrally clipped or limited* such that $\text{FSupp}(J) \subseteq \text{FDom}(I)$ and brought back in the spacetime representation, then the operation is simply spacetime downsampling which copies pixels of J that are indexed by multiples of u into I [1].

When a band-limited signal goes through a nonlinear function, there are usually new and higher frequency components generated, which if exceeding the sampling rate of the input and output discrete signals, will lead to aliasing and shift variance. Nevertheless, it follows from the Nyquist-Shannon sampling theorem [1] that, when the nonlinear function is a polynomial of degree $k \in \mathbb{N}$, the signal bandwidth is broadened by no more than k fold, and aliasing can be completely avoided by having the input signal upsampled with a $\text{USF} \geq k$ before undergoing the polynomial transformation, so that the upsampled larger image array has a sufficient bandwidth to accommodate newly generated high-frequency signal components by the polynomial function without any spectrum overlapping or aliasing taking place.

Fig. 3 (a) and (b) show the output image in real space and its power spectral density respectively, when the test image of Fig. 1 without upsampling is transformed by a square function, where in Fig. 3 (b) it is seen that the spectral margin in the high-frequency region has been closed, and it is reasonable to suspect that spectrum overlapping and aliasing have taken place. Indeed, when the square function is subject to the above-described shift variance detector, using the usual test image as input and the typical shift vector $(\delta x, \delta y) = (0.3, 0.4)$ pixel in the real space, a large value 3.68×10^{-2} is reported for the RSV. By contrast, when the test image is upsampled by a $\text{USF} = 2$ before being square-transformed, the output image is shown in Fig. 4 (a), whose log-scaled power spectral density is shown in Fig. 4 (b), where it is apparent that the spectral content is basically doubled by the square function, which nonetheless is still well within the enlarged bandwidth and still leaves a clear margin in the high-frequency regions, so to have spectrum overlapping and aliasing completely avoided. Indeed, when testing with the shift vector $(\delta x, \delta y) = (0.3, 0.4)$ pixel in the real space, the shift variance detector reports an $\text{RSV} = 2.24 \times 10^{-15}$, still on the order of numerical rounding errors for double precision floating point numerics.

For another experiment, a cubic function is also tested, which transforms the test image of Fig. 1 without upsampling into an output image as shown in Fig. 5 (a), whose log-scaled power spectral density is shown in Fig. 5 (b), which displays no spectral margin in the high-frequency regions and indicates risks of spectrum overlapping and aliasing. Indeed, under the same $(\delta x, \delta y) = (0.3, 0.4)$ pixel in the real space as a typical shift vector, the shift variance

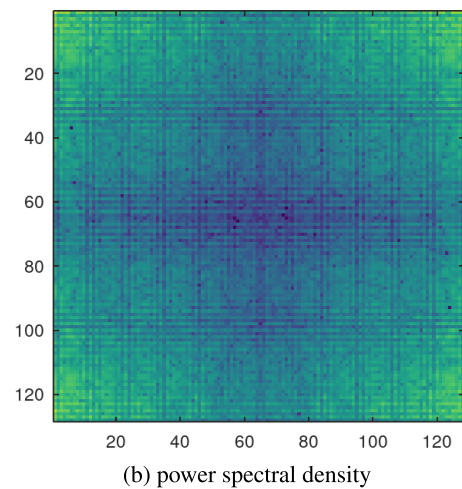
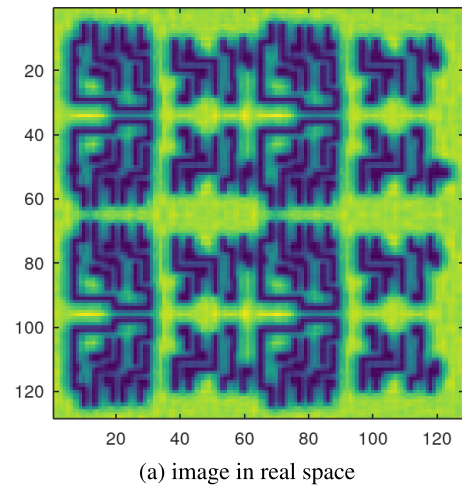


FIGURE 3. The squared image in real space and its power spectral density.

detector reports an even larger value of $\text{RSV} = 5.54 \times 10^{-2}$, indicating a severer problem of aliasing than the square function. By contrast, when the test image is firstly upsampled with a $\text{USF} = 3$ before undergoing the cubic transformation, then spectrum overlapping and aliasing can be completely avoided, and the cubically transformed output image is shown in Fig. 6 (a), whose log-scaled power spectral density is shown in Fig. 6 (b), which displays a clear spectral margin in the high-frequency regions, indicating the absence of spectrum overlapping and aliasing. Indeed, with the same arbitrarily chosen shift vector $(\delta x, \delta y) = (0.3, 0.4)$ pixel, the shift variance detector reports an $\text{RSV} = 4.00 \times 10^{-15}$, still on the order of numerical rounding errors.

Depending on applications, the output signal from a polynomial function may need to be spectrum-limited and downsampled back to a lower sampling rate. Such downsampling after upsampling is especially necessary when multiple nonlinear functions are cascaded in series, in order to stop the image arrays from snowballing into an unwieldy size. Note that the downsampling operation is linear and aliasing-free.

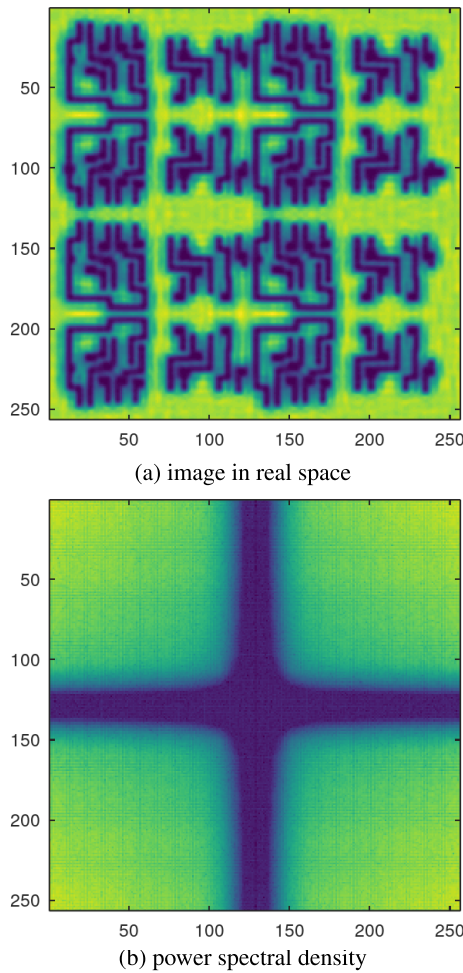


FIGURE 4. The squared upsampled image in real space and its power spectral density.

IV. ALIASING-FREE SIGNAL PROCESSING VIA IMPLICITLY DEFINED FUNCTIONS

Most convolutional neural networks employ nonlinear activation functions. Moreover, they have mostly involved non-polynomial activation functions, until the recent introduction of quadratic functions to neural networks [30]–[32]. Commonly used activation functions include rational functions, sigmoid, hyperbolic tangent, ReLU, Swish, max pooling, Softmax, and SoftPool, *etc.* [7]–[12], all of which induce spectral broadening without a definitive frequency cutoff, so that aliasing cannot be completely avoided no matter how many times the input signal is upsampled, although a larger USF does reduce the severity of aliasing-induced problems. ReLU and max pooling involve the absolute value function, which is not just non-polynomial but actually non-analytic. The non-polynomial characteristics of other named activation functions are attributable to the exponential function and the division operation, which are at least analytic so long as division by zero is averted. The absolute value function must be avoided whenever possible, and new methods should be developed to implement the exponential function and the division operation in an aliasing-free fashion.

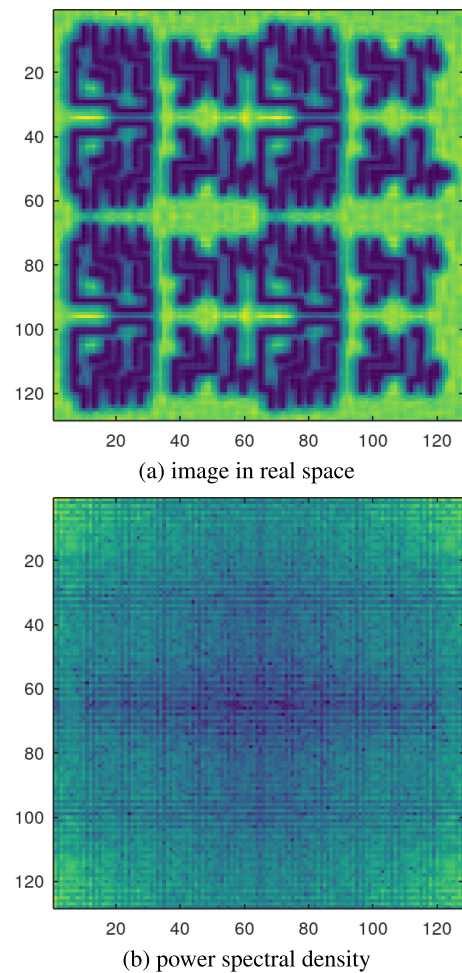


FIGURE 5. The cubed image in real space and its power spectral density.

For a concrete example, consider a so-called Wiener-Padé function WP employed in computational lithography to model the photochemical response of a photoresist, transforming an optical image $I(x, y)$ into a resist image $J(x, y)$, $(x, y) \in \mathbb{R}^2$, such that

$$J(x, y) = WP[I(x, y)] \stackrel{\text{def}}{=} \frac{(1+a)\{[I * h](x, y)\}^2}{I_m^2 + a\{[I * h](x, y)\}^2}, \quad (1)$$

where $h(x, y) \stackrel{\text{def}}{=} (2\pi\sigma^2)^{-1} \exp[-(x^2 + y^2)/2\sigma^2]$ is a two-dimensional Gaussian kernel meant to simulate the spatial blurring effects due to chemical diffusion, $\sigma > 0$ and $a > 0$ are constant parameters, $I_m > 0$ is the maximum value of the image $I(x, y)$, and the $*$ operator represents a two-dimensional convolution. It is easily seen from equation (1) that the WP function, being rational and a Padé approximant, characterizes the sigmoid curve-like response of a photoresist, which suppresses small $[I * h](x, y)$ values further down toward 0 and saturates toward the maximum value 1 for large $[I * h](x, y)$ values.

Unfortunately, the division operation for the WP function makes it difficult to avoid aliasing, even if the input image

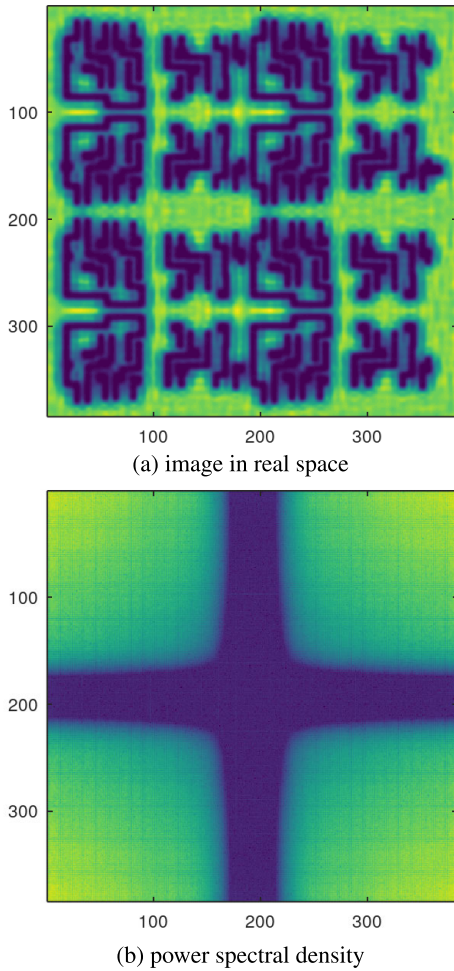


FIGURE 6. The cubed upsampled image in real space and its power spectral density.

$I(x, y)$ is upsampled. For example, even if the test image of Fig. 1 is $2\times$ upsampled as an input image to the WP function of equation (1), with $\sigma = 7.5 \text{ nm}$, $a = 2.5^2$, $I_m = 1.0$, the output image as shown in Fig. 7 (a) and (b) still suffers from spectrum overlapping and aliasing, and the shift variance detector reports an $RSV = 2.45 \times 10^{-3}$ under a typical shift vector $(\delta x, \delta y) = (0.5, 0.3)$ pixel, which may not seem to be very large but already goes beyond tolerance, due to the extremely stringent accuracy requirements in computational lithography. Next, even if the test image of Fig. 1 is $3\times$ upsampled as an input image to the same WP function, the output image as shown in Fig. 8 (a) and (b) still suffers from spectrum overlapping and aliasing, and the shift variance detector reports an $RSV = 2.54 \times 10^{-4}$ under the same shift vector $(\delta x, \delta y) = (0.5, 0.3)$ pixel, which has been much reduced but is still a significant source of error for computational lithography, not to mention that such result is only obtained at the heavy costs of $9\times$ memory usage and $9\times$ runtime.

In a large CNN or an iterative optimization loop where a signal or image goes through multiple non-polynomial

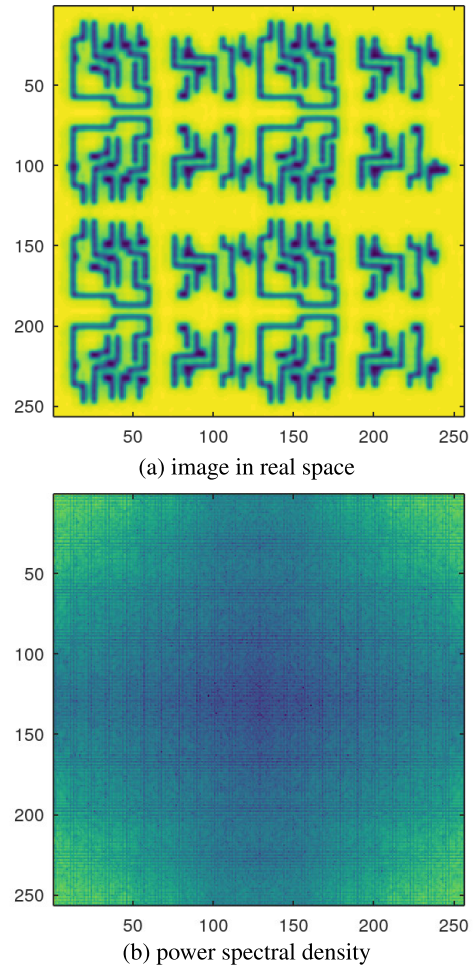


FIGURE 7. The output image in real space and its power spectral density from the Wiener-Padé function, when the input image is $2\times$ upsampled.

functions in cascade, a more serious and fundamental problem is that such aliasing-induced shift variance, though not very large per each non-polynomial transformation, will accumulate and snowball into a substantially large random variance and eventually exceed the error tolerance of a specific application. Worse still, some initial experiments seem to indicate that such aliasing-induced errors are compounded and grow at a rate faster than a polynomial function (e.g., N or N^α for a fixed $\alpha > 0$) of the number N of non-polynomial stages. Rather, the growth of such compounded aliasing-induced errors is likely to follow an exponential function of N or N^α , $\alpha > 0$, indicating the possibility of a new form of numerical instability when aliasing-induced errors are compounded after a series of non-polynomial functions. At any rate, there is a need to solve the problem of aliasing due to non-polynomial functions.

As a general method to enforce aliasing-free signal processing in CNNs that seemingly require non-polynomial functions, I propose to avoid using any explicitly defined non-polynomial function f to transform an input signal or image I into an output signal or image $J = f(I)$, instead,

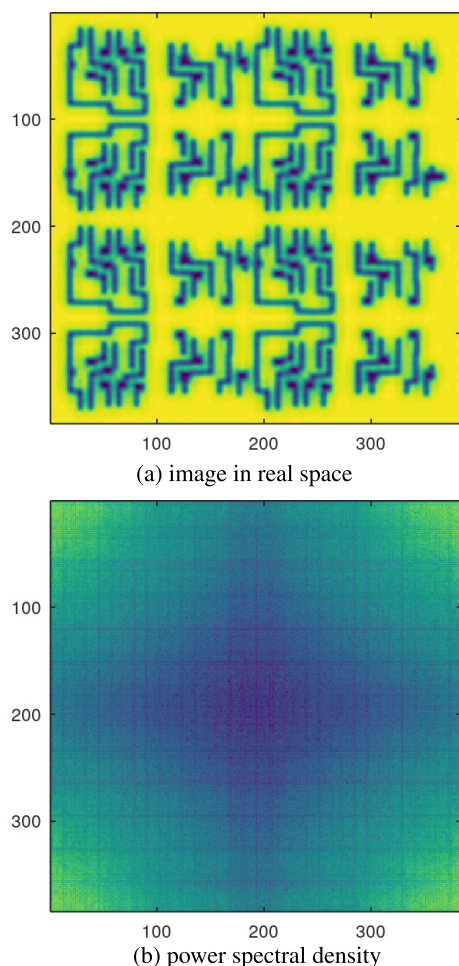


FIGURE 8. The output image in real space and its power spectral density from the Wiener-Padé function, when the input image is 3× upsampled.

to employ a numerical procedure that involves only polynomial operations on signals or images, to iteratively compute or approximate the desired output J as an implicitly defined function of the input I , either by solving an algebraic equation $p(I, J, J^{-1}) = 0$, or by minimizing the norm of a polynomial $p(I, J, J^{-1})$ of signals or images, with p being a polynomial in I and J or $J^{-1} \stackrel{\text{def}}{=} 1/J$ as variables. Normally, it is either J or J^{-1} appearing in the polynomial $p(I, J, J^{-1})$ as variables, although it is not forbidden for J and J^{-1} to appear simultaneously. To substitute an explicitly defined non-polynomial function $J = f(I)$, one could construct a polynomial $p(I, J, J^{-1})$, such that, either the algebraic equation $p(I, J, J^{-1}) = 0$ or the minimization of $\|p(I, J, J^{-1})\|$ defines an implicit function $J = g(I)$, with the function g well approximating the non-polynomial function f . Here $\|\cdot\|$ is a general norm of signals or images, with the usual choices being L^1, L^2, L^∞ , etc. The well known Newton-Raphson methods [33] solve an algebraic equation $p(I, J, J^{-1}) = 0$ iteratively and produce an approximant to the non-polynomial function $J = f(I)$. There are many gradient-based optimization methods, such as the method

of gradient descent and the conjugate gradient method [33], [34], which use numerical iterations to find an approximant for $J = f(I)$ to minimize $\|p(I, J, J^{-1})\|$.

When implemented in numerical simulations or signal processing systems, the numerical iterations for solving an algebraic equation $p(I, J, J^{-1}) = 0$ or minimizing the norm of a polynomial $p(I, J, J^{-1})$ of signals or images should only involve aliasing-free operations on signals or images, such as global amplitude scalings, additions and subtractions, linear convolutions, and polynomial operations with proper upsampling and subsequent downsampling if needed. Said polynomial functions are usually either $p(I, J, J^{-1})$ itself or its partial derivatives. At the n -th iteration, $n \in \mathbb{N}$, the iterative algorithm produces a signal or image J_n as the n -th implicit approximant to the desired output $J = g(I)$. With a good iterative algorithm, the n -th implicit approximant $J_n(\cdot)$ converges to the function $g(\cdot)$ as $n \in \mathbb{N}$ increases. In practice, an iterative algorithm is always terminated at a finite n -th iteration with a predetermined $n \in \mathbb{N}$, and outputs the signal or image $J_n(I)$ as a function of the input I , so that the algorithm realizes a function $J_n(\cdot)$ as the n -th implicit approximant to the implicitly defined function $g(\cdot)$. Clearly, a larger $n \in \mathbb{N}$ is able to produce a closer approximant to the original explicitly defined non-polynomial function but entails a higher computational cost. A practical choice of $n \in \mathbb{N}$ should be a reasonable tradeoff that achieves a satisfiable approximant within an affordable computational cost. It is important to note that a resulting n -th implicit approximant $J_n(\cdot), \forall n \in \mathbb{N}$ is guaranteed to be aliasing-free regardless of at which stage the iteration algorithm terminates, since each iteration step is made aliasing-free.

The following is a general algorithm for implementing a non-polynomial function without aliasing by solving an implicitly defined function iteratively using polynomial operations.

Algorithm 2 (Aliasing-Free Implementation of Non-Polynomial Functions)

- 2.1 Construct either an algebraic equation or an optimization problem in terms of polynomials of the input and the output or its reciprocal, whose solution corresponds to a desired non-polynomial function;
 - 2.2 Construct an iterative procedure that employs polynomial operations to produce increasingly accurate approximants to the solution of said algebraic equation or optimization problem;
 - 2.3 Execute said iterative procedure numerically in terms of linear shift-invariant and polynomial operations on signals or images, avoiding aliasing by upsampling, downsampling, and spectrum-limiting.
-

For an exemplary embodiment, consider the (pointwise) square root function, which takes a pointwise non-negative-valued image $I(x), x \in \mathbb{R}^d, d \in \mathbb{N}$ as input and returns an image J such that $J(x) = \sqrt{I(x)}, \forall x \in \mathbb{R}^d$. The functional relationship is equivalently and implicitly defined through

the rational equation $p(I, J, J^{-1}) \stackrel{\text{def}}{=} J^2 - I = 0$, or the problem of minimizing $\|p(I, J, J^{-1})\| \stackrel{\text{def}}{=} \|J^2 - I\|$. Such equivalence is exact when the images I and J are continuous and have an unlimited bandwidth. However, when the input I is a Shannon-Nyquist sampled discrete image with $BW(I) = F_{\max} > 0$, and the desired output J should also be a discrete image with a bandwidth allowance $BW(J) = uF_{\max}$, $u > 0$, where $BW(I)$ and $BW(J)$ denote the bandwidths of the images I and J respectively, then an explicitly defined square root function cannot avoid the problem of aliasing, while an implicitly defined square root function can only be approximate, promising to output the best possible approximant image J within the bandwidth allowance. Usually, the bandwidth allowance uF_{\max} should be wider than the input bandwidth F_{\max} , with $u > 1$ being an input-output USF. An aliasing-free function is implicitly defined by $J = \arg \min \{\|J^2 - I\| : BW(J) \leq uF_{\max}\}$, namely, for each image array I as an input, finding and outputting an image array J , among all of the possible choices subject to the bandwidth allowance uF_{\max} , which minimizes $\|J^2 - I\|$.

For another exemplary embodiment, consider a Wiener-Padé function $WP(I) = p(I)/q(I)$ such as the example of equation (1), with both $p(I)$ and $q(I)$ being Wiener polynomial functions of the input image I , each Wiener polynomial (or called a Wiener model) involves linear operations that convolve the input image I with a number of so-called Wiener kernels, and polynomial operations that cross-multiply the convolution results in a pointwise manner [22]. Both the $p(\cdot)$ and the $q(\cdot)$ functions are polynomial and amenable to aliasing-free implementations, while the $WP(\cdot)$ function involves division whose explicit implementation induces aliasing. By contrast, once a bandwidth allowance $uBW(I)$, $u > 0$ is chosen, the optimization problem

$$J = \arg \min \{ \|q(I)J - p(I)\|_2^2 : BW(J) \leq uBW(I) \}, \quad (2)$$

with $\|\cdot\|_2$ being the L^2 norm of images implicitly defines a function $J(\cdot)$ that best approximates the desired $WP(\cdot)$ function subject to the bandwidth allowance. Since $\|q(I)J - p(I)\|_2^2$ is a perfect-square polynomial, thus a convex function, with the individual pixels of the spatial image array J as variables, the conjugate gradient method is perfectly suited to solve the optimization problem through iterative multiplications implemented in an aliasing-free manner, since each iteration of the conjugate gradient algorithm involves only multiplications and scaling of images, namely, each iteration applies a polynomial operation to images [33], [34]. Specifically, in the beginning of an n -th conjugate gradient iteration, $n \in \mathbb{N}$, the bandwidth of J_{n-1} as a result from a previous iteration is guaranteed to satisfy $BW(J_{n-1}) \leq uBW(I)$, then both $q(I)$ and J_{n-1} are upsampled to have both $\text{FDom}[q(J_{n-1})]$ and $\text{FDom}(I)$ at least as wide as $2BW[q(I)] + uBW(I)$, so that the pointwise product $q^2(I)J_{n-1}$ is computed without aliasing, which is then spectrally clipped or limited to the bandwidth allowance $BW(J)$, and possibly downsampled if necessary, to become an increment δJ_n , that is finally added to J_{n-1} to

obtain an n -th implicit approximant $J_n = J_{n-1} + \delta J_n$, which is guaranteed to be aliasing-free.

Alternatively, the famous iterative algorithm of Newton-Raphson division can be employed to compute the reciprocal $J(I)$ of a polynomial denominator $q(I)$, namely, $J(I) = 1/q(I)$, via iterative multiplications. Then a Wiener-Padé function $WP(I) = p(I)/q(I)$ can be evaluated by one more step of multiplication $WP(I) = p(I)J(I)$. Specifically, the ingenuity of Newton-Raphson division is to rewrite the explicit operation of division $J = 1/q$ into an implicitly defined function through an algebraic equation $f(J) = J^{-1} - q = 0$, which involves a polynomial of I and J^{-1} as variables. The J solution for $J^{-1} - q(I) = 0$ can be approximated using Newton's method through iterations $J_{n+1} = J_n - f(J_n)/f'(J_n) = J_n + J_n[1 - q(I)J_n]$, $n \geq 0$, which involve only multiplications. The entire procedure of Newton-Raphson iterations can be made completely aliasing-free by properly upsampling images before multiplications and spectrum-limiting product images after multiplications. It turns out that such a Newton-Raphson implementation not only is cheaper per iteration than a conjugate-gradient counterpart, but also converges faster and requires a fewer number of iterations. Both contribute to improving the numerical efficiency. One drawback of the Newton-Raphson division algorithm is that it is only conditionally stable, and a bad initial condition could lead to an exponential blowup.

The Newton-Raphson method can be adapted for aliasing-free implementations of radical finding, or root finding for a general algebraic function, with an archetypical example of inverse square rooting, $J = 1/\sqrt{q}$. One starts with rewriting the explicit operation $J = 1/\sqrt{q}$ into an implicitly defined function through an algebraic equation $f(J) = J^{-2} - q = 0$, then applies Newton's method of root-finding to derive an iterative formula $J_{n+1} = J_n + f(J_n)/f'(J_n) = J_n + \frac{1}{2}J_n(1 - qJ_n^2)$, $n \geq 0$, which involves only iterative multiplications and can be made entirely aliasing-free. Once an aliasing-free $1/\sqrt{q}$ is obtained, an aliasing-free $\sqrt{q} = (1/\sqrt{q}) \times q$ is easily computed with just one more step of aliasing-free multiplication. These Newton's method-inspired implementations are just a couple of examples among a myriad of efficient algorithms for computing power roots or inverse power roots, some of which involve only iterative multiplications, while others may involve divisions as well. Regardless, when combined with the Newton-Raphson division algorithm, it is obvious that such efficient algorithms for power rooting or inverse power rooting can all be implemented through iterative multiplications and made aliasing-free as desired.

For a specific example, the Wiener-Padé function of equation (1) is approximately implemented using the method of Newton-Raphson division, with $J(I) = p(I)K(I)$, $K(I) = 1/q(I)$, and

$$p(I)(x, y) = (1 + a) \{ [I * h](x, y) \}^2, \quad (3)$$

$$q(I)(x, y) = I_m^2 + a \{ [I * h](x, y) \}^2, \quad (4)$$

for any $(x, y) \in \mathbb{Z}^2$, where $h(x, y) = (2\pi\sigma^2)^{-1} \exp[-(x^2 + y^2)/2\sigma^2]$ is a two-dimensional Gaussian kernel. Then a concrete numerical experiment is carried out using the test image of Fig. 1 for an input I , the bandwidth allowance for the output image being set to $BW(J) = 2BW(I)$, other parameters being $\sigma = 7.5$ nm, $a = 2.5^2$, $I_m = 1.0$, and the Newton-Raphson division being terminated at the 6th iteration, such that the 6th implicit approximant K_6 is obtained and output, which approximates the reciprocal of $q(I)$ very well, with $\|\mathbb{1} - q(I)K_6\|_\infty < 5 \times 10^{-3}$. The output image and its log-scaled power spectral density are shown in Fig. 9 (a) and (b) respectively. Remarkably, under any shift vector $(\delta x, \delta y)$, the shift variance detector always reports an $RSV < 5 \times 10^{-15}$, on a par with numerical rounding errors, indicating an absence of aliasing-induced shift variance. Also importantly, preliminary experiments having a number of such implicitly defined Wiener-Padé functions in cascade indicate no compounding of aliasing-induced errors, apart from a slow accumulation of numerical rounding errors as an image going through more and more numerical operations.

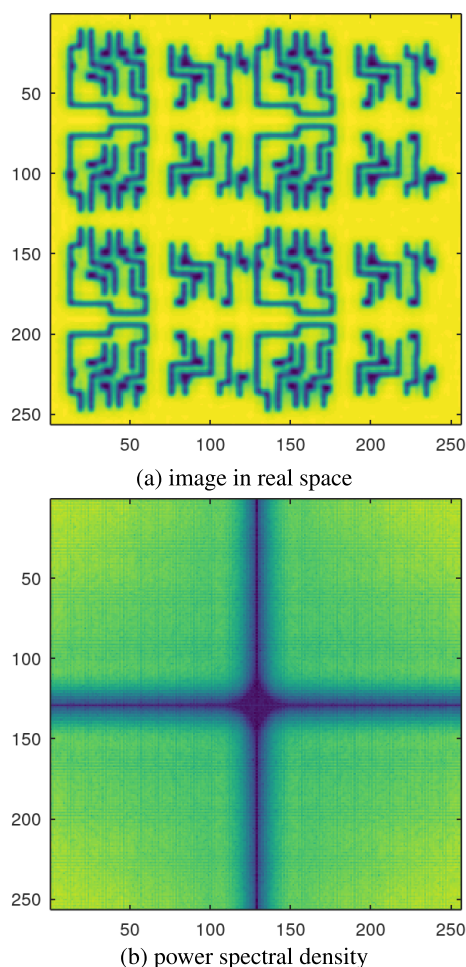


FIGURE 9. The output image in real space and its power spectral density from the implicitly defined Wiener-Padé function.

For a more comprehensive test and comparison, the above-described explicit Wiener-Padé (eWP) and implicit

Wiener-Padé (iWP) functions have been run repeatedly under a combinatorial set of spatial shift conditions, with $\delta x = 0.3, 0.5, 0.7$ and $\delta y = 0.3, 0.4$ in unit of image grid. The eWP and the iWP functions have the same bandwidth allowance $BW(J) = 2BW(I)$ for their output images. Two variants of the explicit Wiener-Padé, labeled as eWP1 and eWP2 respectively, are tested, with eWP1 having the allowed $2BW(I)$ bandwidth fully populated without any margin, and eWP2 removing about 15% of the output spectrum near the Nyquist frequencies, which is done by the iWP implementation as well. Octave functions and test scripts have been executed on a Windows 11-operated Dell Precision 7540 computer equipped with an Intel(R) Core(TM) i7-9850 CPU @2.60GHz and 23.4 GB worth of available physical memory. Table 1 records the RSV values and CPU times associated with the eWP1, eWP2, and iWP functions under the combinations of $(\delta x, \delta y)$ spatial shifts, where the eWP1 and eWP2 times refer to the CPU time spent by a shift variance detector testing the explicit Wiener-Padé functions, and the iWP time is the CPU time spent by the same shift variance detector testing the implicit Wiener-Padé function. It has been observed that eWP1 consumes the least CPU time but suffers from significant RSVs that average 3.82×10^{-3} . With just a little bit of spectrum removed around the Nyquist frequency, eWP2 manages to reduce the RSVs to an almost halved average of 2.07×10^{-3} but spends about 2.69 folds of the CPU time. Under any of the tested $(\delta x, \delta y)$ spatial shifts, the iWP function always converges within 6 Newton-Raphson iterations to produce a satisfying approximant with an L_∞ error less than 5×10^{-3} . Because of the 6 Newton-Raphson iterations, each of which involves spectrum control by upsampling and downsampling images, the iWP function costs a longer runtime that is about $8.05 \times$ and $2.99 \times$ of the eWP1 and eWP2 CPU times respectively. What this runtime overhead buys back is the complete removal of aliasing-induced shift variance, manifested by iWP RSV values being always $< 5 \times 10^{-15}$, still on the order of numerical rounding errors. By contrast, even the eWP2 RSV values mostly hover above the level of 1×10^{-3} , averaging into 2.07×10^{-3} , which is a colossal number in comparison, and beyond tolerance for many applications.

V. PERSPECTIVES OF ALIASING-FREE NONLINEAR SIGNAL PROCESSING FOR CONVOLUTIONAL NEURAL NETWORKS

Having understood the general theory and seen the aliasing-free operation of Wiener-Padé functions, one could apply the same principle and method to implement other non-polynomial operations as an implicitly defined function solved by an iterative algorithm involving polynomial operations. Important and frequently used functions include rational functions, sigmoid, hyperbolic tangent, ReLU, Swish, max pooling, Softmax, and SoftPool, *etc.* [7]–[12], all of which are non-polynomial and induce signal aliasing. Notwithstanding specifically designed implicit functions and iterative algorithms that are particularly suitable for specific

TABLE 1. RSV values and runtime costs of the eWP1, eWP2, and iWP functions under combinations of spatial shifts $(\delta x, \delta y)$ in unit of image grid.

$(\delta x, \delta y)$	(0.3, 0.3)	(0.5, 0.3)	(0.7, 0.3)	(0.3, 0.4)	(0.5, 0.4)	(0.7, 0.4)
eWP1 RSV	5.16×10^{-3}	2.45×10^{-3}	4.58×10^{-3}	4.91×10^{-3}	1.46×10^{-3}	4.34×10^{-3}
eWP1 time	1.57 sec	1.58 sec	1.58 sec	1.58 sec	1.57 sec	1.58 sec
eWP2 RSV	2.54×10^{-3}	1.39×10^{-3}	2.64×10^{-3}	2.41×10^{-3}	0.82×10^{-3}	2.63×10^{-3}
eWP2 time	4.25 sec	4.22 sec	4.19 sec	4.22 sec	4.25 sec	4.36 sec
iWP RSV	4.03×10^{-15}	4.17×10^{-15}	3.98×10^{-15}	4.06×10^{-15}	4.62×10^{-15}	3.69×10^{-15}
iWP time	12.62 sec	12.74 sec	12.74 sec	12.80 sec	12.69 sec	12.71 sec

non-polynomial functions, there exists a unified and systematic strategy to realize a general nonlinear system of signal processing, and any convolutional neural network in particular, as an iCNN free of signal aliasing. Such strategy combines the present method for aliasing-free implementations of Wiener-Padé functions with the recent developments of convolutional neural networks using Padé or rational activation units [9], [10].

Theoretically, it has been established that convolutional neural networks, particularly ReLU networks, can efficiently approximate and be approximated efficiently by rational functions, especially rational networks [7]. Practically, Padé or rational approximants as activation units have recently been introduced and tested in deep convolutional neural networks, showing rather competitive performances [9], [10]. It is straightforward to combine the construct of neural networks using Padé or rational activation units with the present invention of aliasing-free nonlinear signal processing and come up with a novel method that provides a complete solution for the aliasing-induced problems in convolutional neural networks. Specifically, a convolutional neural network can be made aliasing-free when all of the activation functions therein are replaced by Padé or rational approximants, which can be implemented aliasing-free as implicitly defined functions through iterations of polynomial operations.

Such a combined method provides a solution to the dreaded problem of aliasing and shift variance, or any other loss of signal symmetry due to aliasing [13]–[18], although it may incur an increased computational complexity in comparison to a traditional convolutional neural network using conventional non-polynomial functions that are explicitly defined and directly computed. A rule of thumb estimate is that the combined method might have to bear an $N_{iter} \times N_{usf}^d$ factor in computational complexity, where $N_{iter} \in \mathbb{N}$ is due to an average number of iterations in the iterative algorithms solving for the implicitly defined functions, N_{usf} is an average USF to avoid spectral overlap and aliasing during the polynomial operations, while $d \in \mathbb{N}$ is the dimensionality of the concerned signals or images, e.g., $d = 1$ for sound signals in audio applications, and $d = 2$ for optical images in video or vision applications. It is interesting to note that the conventional and explicit non-polynomial functions are also expensive to compute, which are usually

implemented through iterations involving multiplications as well, and embodied in hardware circuitry that costs space of computer chips instead of computational time [35]–[38]. Indeed, it was exactly those iterative algorithms for conventional and explicit non-polynomial functions that had inspired the present invention of implementing implicitly defined functions via iterations of polynomial operations. Although, it may be reminded that those iterative multiplications implementing conventional and explicit non-polynomial functions do not solve the aliasing problems because they are performed locally per each point or pixel in spacetime and lack the capability of global spectrum control.

It is conceivable that similar hardware acceleration techniques can be implemented in graphics processing units (GPUs), DSP chips, or the so-called tensor processing units (TPUs), which can be employed to compute implicitly defined non-polynomial functions for aliasing-free operations, to absorb the cost of N_{iter} iterative multiplications into hardware circuitry. To minimize the factor N_{usf}^d , the smallest possible USF $N_{usf} = 2$ can be used, with all polynomial operations in the iterative algorithms being implemented through multi-stage pairwise multiplications using the well-known Horner’s method, where the multiplication between each pair of signals or images is realized by firstly $2 \times$ upsampling each of the two multiplying signals or images, then computing the pointwise pixel-to-pixel products to produce a product signal or image at the $2 \times$ sampling rate, finally spectrum-limiting the product signal or image back to the $1 \times$ bandwidth and downsampling it if desired. If one really wishes to get rid of the N_{usf}^d computational overhead, then he/she could elect to halve the spectral contents before multiplying each pair of signals or images, provided that the reduced spectra could still represent the concerned signals or images reasonably well.

For an exemplary embodiment, consider the exponential function e^{-z} , $z \in \mathbb{R}$, whose Padé approximants are well known [7], [29], with the [4/4]-th approximant reading

$$e^{-z} \approx \frac{P_4(z)}{Q_4(z)} \stackrel{\text{def}}{=} \frac{1680 - 840z + 180z^2 - 20z^3 + z^4}{1680 + 840z + 180z^2 + 20z^3 + z^4}, \quad (5)$$

which guarantees a small relative error below 5.66×10^{-6} for all $x \in [-2, 2]$. It is obvious that $Q_4(z) \stackrel{\text{def}}{=} 1680 + 840z + 180z^2 + 20z^3 + z^4$, and $P_4(z) \equiv Q_4(-z)$. When a wider dynamic range of the input values is desired, an exponential

function e^{-z} with $|z| \leq 2^m$, $m \geq 1$ can be realized by cascading a Newton-Raphson implementation of the [4/4] Padé approximant in equation (5) for the exponential function $E_0(z) \stackrel{\text{def}}{=} \exp(-z/2^{m-1})$ with $|z| \leq 2^m$, and m stages of repeated squaring operations to obtain $E_i(z) = E_{i-1}(z)^2$ in sequence, $\forall i \in [1, m]$. This scheme inspires a *repeated squaring method* for another iterative-multiplication-only implementation of the exponential function e^{-z} , $|z| \leq 2^m$, $m \in \mathbb{Z}$, without using the Padé approximation and the Newton-Raphson algorithm. Said repeated squaring method is based on an identity $\lim_{n \rightarrow \infty} (1 - z/2^n)^{2^n} = e^{-z}$ well known in calculus, and does repeated squaring for an $n > \max(m, 0)$ times iteratively to compute a finite sequence $E_0(z) \stackrel{\text{def}}{=} 1 - z/2^n$, $E_i(z) = E_{i-1}(z)^2$, $\forall i \in [1, n]$, obtaining the n -th repeated squaring approximant $E_n(z) = (1 - z/2^n)^{2^n}$ at the end, which yields an excellent approximant to the exponential function e^{-z} , $|z| \leq 2^m$, when the value of $n - m$ is sufficiently large.

Similarly, another *repeated squaring method* can be employed, in conjunction with the Goldschmidt iteration [35] and the binomial theorem, for an alternative and efficient implementation of division via iterative multiplications. Let $z \in \mathbb{R}$ be suitably biased and scaled such that $z \in [\epsilon, 2 - \epsilon]$ for a fixed constant $\epsilon > 0$. To compute the reciprocal $z^{-1} = (1 - \zeta)^{-1}$, $\zeta \stackrel{\text{def}}{=} 1 - z \in [\epsilon - 1, 1 - \epsilon]$, one uses the identity

$$\begin{aligned} \frac{1}{1 - \zeta} &= \frac{1 + \zeta}{1 - \zeta^2} = \frac{(1 + \zeta)(1 + \zeta^2)}{1 - \zeta^4} = \dots \\ &= \frac{\prod_{k=0}^{n-1} (1 + \zeta^{2^k})}{1 - \zeta^{2^n}}, \quad \forall n \in \mathbb{N}, \end{aligned} \quad (6)$$

to approximate $(1 - \zeta)^{-1}$ as $\prod_{k=0}^{n-1} (1 + \zeta^{2^k})$ by neglecting ζ^{2^n} in the denominator, which diminishes rapidly once n becomes larger than $|\log_2 \epsilon|$. Now the multiplication-only approximation $(1 - \zeta)^{-1} \approx \prod_{k=0}^{n-1} (1 + \zeta^{2^k})$ can be efficiently computed in a prescribed logarithmic number $n \in \mathbb{N}$ of computational steps: allocate computer memory for two variables ξ and Z and initialize them as $Z \leftarrow 1$ and $\xi \leftarrow \zeta$, then repeat for n times the computations and refreshments of data in memory as $Z \leftarrow Z(1 + \xi)$ followed by $\xi \leftarrow \xi^2$, finally output the result in Z as an approximant for $(1 - \zeta)^{-1}$.

Such iterative-multiplication-only algorithms lend themselves straightforwardly to aliasing-free implementations for the exponential function $\exp[-I(x)]$ and the multiplicative inverse function $1/I(x)$ on images $I(x)$, $x \in \mathbb{R}^d$, $d \geq 1$, with suitable upsampling, spectrum-limiting, and downsampling before and after multiplications to avoid aliasing. Consequently, an (a, b) two-parameter sigmoid function (also known as a logistic function)

$$\text{sigm}[I(x)] \stackrel{\text{def}}{=} \frac{a e^{bI(x)}}{1 + a e^{bI(x)}} = \frac{a}{a + e^{-bI(x)}}, \quad (7)$$

with $a > 0$, $b > 0$, can be realized in general by cascading an aliasing-free exponential operation and an aliasing-free rational function using either Newton-Raphson division or

the Goldschmidt method with repeated squaring. The parameter a is called a numerical conditioner, which is chosen to ensure that $\|\exp[-bI(x)]\|_\infty/a$ is never excessively large, so that the numerical iterations of Newton-Raphson or Goldschmidt repeated squaring are always properly conditioned to converge quickly. Once an aliasing-free sigmoid transformation is implemented to produce a sigmoid output $\text{sigm}(I)$, an approximate ReLU function, called Swish [8], can be implemented by multiplying the input I with the sigmoid output $\text{sigm}(I)$ to obtain $\text{Swish}(I) = I \times \text{sigm}(I)$, for any input signal or image $I = I(x)$, $x \in \mathbb{R}^d$, $d \geq 1$. Similarly, aliasing-free versions of the Softmax and SoftPool, as well as many other non-polynomial functions, can be realized as compositions of aliasing-free exponential, polynomial, and rational functions.

In the special case of $\exp[-bI(x)]$, $\|bI(x)\|_\infty \leq 2$ being realized via the [4/4] Padé approximant $\exp(-bI) \approx P_4(bI)/Q_4(bI)$ as in equation (5) with no repeated squaring subsequently to enlarge the range of signal amplitudes, an aliasing-free implementation of an (a, b) two-parameter sigmoid function with a bandwidth allowance $BW[\text{sigm}(I)] = uBW(I)$, $u \geq 1$ can start with polynomial operations to generate intermediate images $P_4(bI)$ and $Q_4(bI)$ with a bandwidth allowance $\min(4, u) \times BW(I)$, then compute an intermediate image $J(x) \approx [P_4(bI) + aQ_4(bI)]^{-1}$, with a bandwidth allowance $BW(J) = uBW(I)$, using one of the numerical iterations of Newton-Raphson or Goldschmidt repeated squaring, finally multiply J and $aQ_4(bI)$ to obtain $\text{sigm}[I(x)] \approx J(x) \times aQ_4[bI(x)]$ with a bandwidth allowance $BW[\text{sigm}(I)] = uBW(I)$. Said multiplications or polynomial operations involve pointwise multiplications or additions in the x -coordinate space, $x \in \mathbb{R}^d$, which are always implemented in an aliasing-free manner, by suitably upsampling the operands and creating sufficient spectral margins before multiplying them and possibly spectrum-limiting and downsampling the product images afterward.

Alternatively, there is a so-called (matrix) sign function $\text{sign}(\cdot)$ [39] applicable to matrices as linear operators, or to signals and images in a point-wise manner, which is closely related to the sigmoid function through $\text{sigm} = (\mathbb{1} + \text{sign})/2$. Many iterative algorithms exist to compute $\text{sign}(\cdot)$ as an implicit function satisfying a characteristic equation $(\mathbb{1} - \text{sign})(\mathbb{1} + \text{sign}) = 0$ [39]. In particular, the Newton iteration $M_0 = A$, $M_{k+1} = (M_k + M_k^{-1})/2$, $k > 0$ for the sign function can be combined with the Newton-Raphson division (also called the Newton-Schulz iteration for matrices) and turned into a division-free algorithm [39], [40],

$$M_0 \stackrel{\text{def}}{=} A, \quad M_{k+1} \stackrel{\text{def}}{=} \frac{1}{2} M_k \left(3\mathbb{1} - M_k^2 \right), \quad k \geq 0, \quad (8)$$

which converges to $\lim_{k \rightarrow \infty} M_k = \text{sign}(A)$ so long as $\|\mathbb{1} - A^2\| < 1$, where A is a given matrix or signal or image, $\|\cdot\|$ denotes the operator norm when A is a matrix or the L^∞ norm when A is a signal or image. A great advantage of the Newton-Schulz iteration is the avoidance of matrix inversion or signal division, which is rightly suitable

for aliasing-free implementations of the $\text{sign}(\cdot)$ and $\text{sigm}(\cdot)$ functions in CNNs. Specifically, the Newton-Schulz iteration of equation (8) can be adapted to approximate the sign function of a signal or image A by executing each iteration step with $k > 0$ in a spectrum-controlled and aliasing-free fashion, which has M_k $3\times$ upsampled with respect to the original bandwidth BW , then computes the spacetime point-wise cubic power of the upsampled signal or image, finally spectrum-limits or downsamples the cubic-powered signal or image to produce an M_{k+1} , whose frequency components are non-vanishing only within the original bandwidth BW . The upsampling, spectrum-limiting, and downsampling operations may or may not use FFTs. As already specified in the above, once an aliasing-free approximation of $\text{sign}(A)$ has been obtained for a given signal or image A , then sigmoid and ReLU transformed results can be obtained in turn as $\text{sigm}(A) = (\mathbb{1} + \text{sign}(A))/2$ and $\text{ReLU}(A) = A \times \text{sigm}(A)$, both being completely aliasing-free approximations.

Continuing research is needed to implement these mathematical algorithms into computer codes, test and quantify their performances, and incorporate such aliasing-free nonlinear signal processing units into a large-scale neural network as activation functions. While guaranteeing an aliasing-free operation, an implicitly defined and iteratively implemented function is not an exact realization of the corresponding explicit non-polynomial function as prescribed originally. Rather, it produces an approximant within a specific accuracy to the prescribed non-polynomial activation. Therefore, it is important to investigate the impact of approximated activation functions on the performances of a large-scale network. In particular, for machine learning applications employing an iCNN, much work is needed to investigate if and how aliasing-free activations impact the network trainability, inference accuracy, and computer runtime.

VI. SUMMARY AND OPEN QUESTIONS

In summary, a new method is proposed and demonstrated, which implements or approximates non-polynomial operations using implicitly defined functions, that are realized through iterative algorithms involving only polynomial operations, so to achieve completely aliasing-free signal processing. An iCNN, which is a convolutional nonlinear network incorporating no non-polynomials other than such implicitly defined functions should remain aliasing-free as a whole, and provide the first example of an aliasing-free network that incorporates non-polynomial functions. Such an iCNN is expected to find applications where it is important to avoid aliasing-induced errors such as shift variance.

This research has opened more questions than it has managed to answer. Some interesting directions for further investigations include:

1) *In traditional, aliasing-prone networks having a number of nonlinear stages in cascade, how and to what extent does accumulated aliasing lead to numerical instability?*

- 2) *Using an implicitly defined function, how close is an n -th implicit approximant to the desired response? How does an approximation improve as its bandwidth allowance increases?*
- 3) *When applied to real convolutional neural networks, how and to what extent does the method of iCNN impact the learning and classification performances?*

ACKNOWLEDGMENT

The author thanks Prof. G. Wang of the Rensselaer Polytechnic Institute for mentoring and guiding her learning and research in image processing and convolutional neural networks and G. Redillas of the Evergreen Valley High School for teaching Newton's method. The author also acknowledges the help and supports from Prof. Wang and her parents on technical writing and LaTeX typesetting.

REFERENCES

- [1] A. V. Oppenheim and A. S. Willsky, *Signals Systems*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1997.
- [2] M. Schetzen, *The Volterra and Wiener Theories of Nonlinear Systems*. Hoboken, NJ, USA: Wiley, 1980.
- [3] M. Schetzen, "Nonlinear system modeling based on the Wiener theory," *Proc. IEEE*, vol. 69, no. 12, pp. 1557–1573, Dec. 1981.
- [4] W. J. Rugh, *Nonlinear System Theory: The Volterra/Wiener Approach*. Baltimore, MD, USA: Johns Hopkins University Press, 1981.
- [5] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," in *The Handbook of Brain Theory and Neural Networks*, 2nd ed., M. A. Arbib, Ed. Cambridge, MA, USA: MIT Press, 1995.
- [6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
- [7] M. Telgarsky, "Neural networks and rational functions," in *Proc. 34th Int. Conf. Mach. Learn.* Sydney, NSW, Australia, 2017, pp. 3387–3393.
- [8] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," 2017, *arXiv:1710.05941*.
- [9] A. Molina, P. Schramowski, and K. Kersting, "Padé activation units: End-to-end learning of flexible activation functions in deep networks," 2019, *arXiv:1907.06732*.
- [10] Q. Delfosse, P. Schramowski, M. Mundt, A. Molina, and K. Kersting, "Adaptive rational activations to boost deep reinforcement learning," 2021, *arXiv:2102.09407*.
- [11] A. Stergiou, R. Poppe, and G. Kalliatakis, "Refining activation downsampling with SoftPool," 2021, *arXiv:2101.00440*.
- [12] M. Gustineli, "A survey on recently proposed activation functions for deep learning," 2022, *arXiv:2204.02921*.
- [13] A. Fawzi and P. Frossard, "Manitest: Are classifiers really invariant?" 2015, *arXiv:1507.06535*.
- [14] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry, "Exploring the landscape of spatial robustness," 2017, *arXiv:1712.02779*.
- [15] A. Azulay and Y. Weiss, "Why do deep convolutional networks generalize so poorly to small image transformations?" *J. Mach. Learn. Res.*, vol. 20, no. 184, pp. 1–25, 2019.
- [16] Y. Gong and C. Poellabauer, "Impact of aliasing on deep CNN-based end-to-end acoustic models," in *Proc. Interspeech*, Sep. 2018, pp. 2698–2702.
- [17] R. Zhang, "Making convolutional networks shift-invariant again," 2019, *arXiv:1904.11486*.
- [18] J. Singh, A. Tripathy, P. Garg, and A. Kumar, "Lung tuberculosis detection using anti-aliased convolutional networks," *Proc. Comput. Sci.*, vol. 173, pp. 281–290, Jan. 2020.
- [19] P. Gutierrez and A. Cordier. (Jun. 10, 2020). *Robustness and Repeatability of Modern Deep Neural Networks: A Review*. [Blog Post]. [Online]. Available: <https://scortex.io/robustness-and-repeatability-of-modern-deep-neural-networks-a-review>
- [20] Y. Granik, N. Cobb, and D. Medvedev, "Application of CM0 resist model to OPC and verification," *Proc. SPIE*, vol. 6154, Mar. 2006, Art. no. 61543E.

- [21] Y. Granik, D. Medvedev, and N. Cobb, "Towards standard process models for OPC," *Proc. SPIE*, vol. 6520, Mar. 2007, Art. no. 652043.
- [22] H. Wei, "Computer simulation of photolithographic processing," U.S. Patent 8 532 964 B2, Sep. 10, 2013.
- [23] F. Zack, "Neural network based approach to resist modeling and OPC," *Proc. SPIE*, vol. 5377, pp. 670–679, May 2004.
- [24] Y. Watanabe, T. Kimura, T. Matsunawa, and S. Nojima, "Accurate lithography simulation model based on convolutional neural networks," *Proc. SPIE*, vol. 10147, Mar. 2017, Art. no. 101470K.
- [25] J. Shiley, "Machine learning for compact lithographic process models," in *Machine Learning in VLSI Computer-Aided Design*, I. M. Elfadel, D. S. Boning, and X. Li, Eds. Cham, Switzerland: Springer, 2019.
- [26] S. Shim, S. Choi, and Y. Shin, "Machine learning for mask synthesis," in *Machine Learning in VLSI Computer-Aided Design*, I. M. Elfadel, D. S. Boning, and X. Li, Eds. Cham, Switzerland: Springer, 2019.
- [27] C. Kim, S. Lee, S. Park, N. Y. Chung, J. Kim, N. Bang, S. Lee, S. Lee, S. Boone, P. Li, and J. Chang, "Machine learning techniques for OPC improvement at the sub-5 nm node," *Proc. SPIE*, vol. 11323, Mar. 2020, Art. no. 1132317.
- [28] Y. Shin, "Computational lithography using machine learning models," *IPSI Trans. Syst. LSI Des. Methodol.*, vol. 14, pp. 2–10, Jan. 2021.
- [29] G. A. Baker, *Essentials of Padé Approximants*. New York, NY, USA: Academic, 1975.
- [30] F. Fan, W. Cong, and G. Wang, "A new type of neurons for machine learning," *Int. J. Numer. Methods Biomed. Eng.*, vol. 34, no. 2, Feb. 2018, Art. no. e2920.
- [31] F. Fan, W. Cong, and G. Wang, "Generalized backpropagation algorithm for training second-order neural networks," *Int. J. Numer. Methods Biomed. Eng.*, vol. 34, no. 5, May 2018, Art. no. e2956.
- [32] F. Fan, J. Xiong, and G. Wang, "Universal approximation with quadratic deep networks," *Neural Netw.*, vol. 124, pp. 383–392, Apr. 2020.
- [33] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes*, 3rd ed. Cambridge, U.K.: Cambridge Univ. Press, 2007.
- [34] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *J. Res. Nat. Bureau Standards*, vol. 49, no. 6, pp. 409–436, Dec. 1952.
- [35] R. E. Goldschmidt, "Applications of division by convergence," M.S. thesis, Dept. Elect. Eng., MIT, Cambridge, MA, USA, 1964.
- [36] S. Oberman and M. J. Flynn, "Implementing division and other floating-point operations: A system perspective," in *Proc. Scientific Computing and Validated Numerics (SCAN)*, G. Alefeld, A. Frommer, and B. Lang, Eds. Berlin, Germany: Akademie Verlag, 1996, pp. 18–24.
- [37] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, 2nd ed. London, U.K.: Oxford Univ. Press, 2010.
- [38] Z. Hajduk, "Hardware implementation of hyperbolic tangent and sigmoid activation functions," *Bull. Polish Acad. Sci., Tech. Sci.*, vol. 66, no. 5, pp. 563–577, 2018.
- [39] N. J. Higham, *Functions of Matrices: Theory and Computation*. Philadelphia, PA, USA: SIAM, 2008.
- [40] J. Chen and E. Chow, *A Stable Scaling of Newton-Schulz for Improving the Sign Function Computation of a Hermitian Matrix*, document Preprint ANL/MCS-P5059-0114, 2014. [Online]. Available: <https://www.mcs.anl.gov/papers/P5059-0114.pdf>



EMMY S. WEI is currently a student with the Evergreen Valley High School, San Jose. Her research interest includes STEM studies.

...