Boxlets: a Fast Convolution Algorithm for Signal Processing and Neural Networks

Patrice Y. Simard, Léon Bottou, Patrick Haffner and Yann LeCun
AT&T Labs-Research
100 Schultz Drive, Red Bank, NJ 07701-7033
{patrice,leon,b.haffner,yann}@research.att.com

Abstract

Signal processing and pattern recognition algorithms make extensive use of convolution. In many cases, computational accuracy is not as important as computational speed. In feature extraction, for instance, the features of interest in a signal are usually quite distorted. This form of noise justifies some level of quantization in order to achieve faster feature extraction. Our approach consists of approximating regions of the signal with low degree polynomials, and then differentiating the resulting signals in order to obtain impulse functions (or derivatives of impulse functions). With this representation, convolution becomes extremely simple and can be implemented quite effectively. The true convolution can be recovered by integrating the result of the convolution. This method yields substantial speed up in feature extraction and is applicable to convolutional neural networks.

1 Introduction

In pattern recognition, convolution is an important tool because of its translation invariance properties. Feature extraction is a typical example: The distance between a small pattern (i.e. feature) is computed at all positions (i.e. translations) inside a larger one. The resulting “distance image” is typically obtained by convolving the feature template with the larger pattern. In the remainder of this paper we will use the terms image and pattern interchangeably (because of the topology implied by translation invariance).

There are many ways to convolve images efficiently. For instance, a multiplication of images of the same size in the Fourier domain corresponds to a convolution of the two images in the original space. Of course this requires $K N \log N$ operations (where $N$ is the number of pixels of the image and $K$ is a constant) just to go in and out of the Fourier domain. These methods are usually not appropriate for feature extraction because the feature to be extracted is small with respect to the image.

For instance, if the image and the feature have respectively $32 \times 32$ and $5 \times 5$ pixels, the full convolution can be done in $25 \times 1024$ multiply-adds. In contrast, it would require $2 \times K \times 1024 \times 10$ to go in and out of the Fourier domain.

Fortunately, in most pattern recognition applications, the interesting features are already quite distorted when they appear in real images. Because of this inherent noise, the feature extraction process can usually be approximated (to a certain degree) without affecting the performance. For example, the result of the convolution
is often quantized or thresholded to yield the presence and location of distinctive features [1]. Because precision is typically not critical at this stage (features are rarely optimal, thresholding is a crude operation), it is often possible to quantize the signals before the convolution with negligible degradation of performance.

The subtlety lies in choosing a quantisation scheme which can speed up the convolution while maintaining the same level of performance. We now introduce the convolution algorithm, from which we will deduce the constraints it imposes on quantization.

The main algorithm introduced in this paper is based on a fundamental property of convolutions. Assuming that \( f \) and \( g \) have finite support and that \( f^n \) denotes the \( n \)-th integral of \( f \) (or the \( n \)-th derivative if \( n \) is negative), we can write the following convolution identity:

\[
(f * g)^n = f^n * g = f * g^n
\]

(1)

where \( * \) denotes the convolution operator. Note that \( f \) or \( g \) are not necessarily differentiable. For instance, the impulse function\(^1\), noted \( \delta \), verifies the identity:

\[
\delta_a^n * \delta_b^m = \delta_{a+b}^{m+n}
\]

(2)

where \( \delta_a^n \) denotes the \( n \)-th integral of the delta function, translated by \( a \) (\( \delta_a(x) = \delta(x-a) \)). Equations 1 and 2 are new to signal processing. Heckbert has developed an effective filtering algorithm [2] where the filter \( g \) is a simple combination of polynomial of degree \( n - 1 \). Convolution between a signal \( f \) and the filter \( g \) can be written as

\[
f * g = f^n * g^{-n}
\]

(3)

where \( f^n \) is the \( n \)-th integral of the signal, and the \( n \)-th derivative of the filter \( g \) can be written exclusively with delta functions (resulting from differentiating \( n - 1 \) degree polynomials \( n \) times). Since convolving with an impulse function is a trivial operation, the computation of Equation 3 can be carried out effectively. Unfortunately, Heckbert’s algorithm is limited to simple polynomial filters and is only interesting when the filter is wide and when the Fourier transform is unavailable (such as in variable length filters).

In contrast, in feature extraction, we are interested in small and arbitrary filters (the features). Under these conditions, the key to fast convolution is to quantize the images to combinations of low degree polynomials, which are differentiated, convolved and then integrated. The algorithm is summarized by equation:

\[
f * g \approx F * G = (F^{-n} * G^{-m})^{m+n}
\]

(4)

where \( F \) and \( G \) are polynomial approximation of \( f \) and \( g \), such that \( F^{-n} \) and \( G^{-m} \) can be written as sums of impulse functions and their derivatives. Since the convolution \( F^{-n} * G^{-m} \) only involves applying Equation 2, it can be computed quite effectively. The computation of the convolution is illustrated in Figure 1. Let \( f \) and \( g \) be two arbitrary 1-dimensional signals (top of the figure). Let’s assume that \( f \) and \( g \) can both be approximated by partitions of polynomials, \( F \) and \( G \). On the figure, the polynomials are of degree 0 (they are constant), and are depicted in the second line. The details on how to compute \( F \) and \( G \) will be explained in the next section. In the next step, \( F \) and \( G \) are differentiated once, yielding successions of impulse functions (third line in the figure). The impulse representation has the advantage of having a finite support, and of being easy to convolve. Indeed two impulse functions can be convolved using Equation 2 (4 \( \times \) 3 = 12 multiply-adds on the figure). Finally the result of the convolution must be integrated twice to yield

\[
F * G = (F^{-1} * G^{-1})^2
\]

(5)

\(^{1}\)also called Dirac delta function