https://doi.org/10.2352/ISSN.2470-1173.2021.6.IRIACV-317

Accelerated HOG+SVM for Object Recognition

Lilong Shi[1], Chunji Wang[1], Yibing Wang[1] and Kwang Oh Kim[1]

A novel acceleration strategy is presented for computer vision and machine learning field from both algorithmic and hardware implementation perspective. With our approach, complex mathematical functions such as multiplication can be greatly simplified. As a result, an accelerated machine learning method requires no more than ADD operations, which tremendously reduces processing time, hardware complexity and power consumption. The applicability is illustrated by going through a machine learning example of HOG+SVM, where the accelerated version achieves comparable accuracy based on real datasets of human figure and digits.

1. INTRODUCTION

Feature extraction and classification are essential components in the field of computer vision. In the last decade, deep-learning based algorithms have gained popularity in contrast to the classical approaches, specifically due to their remarkable classification accuracy. On the other hand, conventional approaches, or the so-called shallow classifiers, continue to remain favorable for edge devices due to the framework simplicity and small hardware form factor, while achieving satisfactory accuracy required by the applications. For instance, the Histogram of Oriented Gradients (HOG) feature extractor, coupled with Support Vector Machine (SVM) classifier [1], is extensively used by mobile and edge devices for classification tasks that requires ultra-low power to operate in a long period of time with little to none external power source. In this paper, we focus on accelerating the existing HOG+SVM classification algorithm by dramatically reducing computational burden through replacing complex mathematical arithmetic with only ADD operations, without sacrificing classification accuracy.

2. BACKGROUND

The essential idea behind the Histogram of Oriented Gradients (HOG) descriptor [1] is that local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. First of all, the image is divided into small connected regions called *cells*, and for the pixels within each cell, a histogram of gradient directions is compiled. Both horizon and vertical gradient, g_x and g_y, can be simply computed by convolving with two 1D vectors <-1,0,1> & $<-1,0,1>^T$, and the edge magnitude and orientation are calculated by Eqn 1 and Eqn 2,

$$g = \sqrt{g_x^2 + g_y^2} \qquad \text{Eqn 1}$$

$$\theta = \arctan\frac{g_y}{g_x} \qquad \text{Eqn 2}$$

The descriptor H is the concatenation of these histograms. For improved accuracy, the local histograms can be contrast-normalized by calculating a measure of the intensity across a larger region of the image, called a *block*, and then using this value to normalize all cells within the block, ie.

$$H = \frac{H}{\|H\|_2} \qquad \text{Eqn 3}$$

Although HOG descriptors are not tied to a specific machine learning algorithm, it has worked very successfully with Support Vector Machine (SVM) classifier and therefore become widely used machine learning choice. In the simplest form, a binary-SVM can be evaluated by

$$\mathbf{W}^{\mathrm{T}}\mathbf{H} + b \qquad \text{Eqn 4}$$

where **W** is the weight vector, **H** is the normalized histogram calculated in Eqn 3, and b is the bias term. A non-negative score indicates a positive detection result, otherwise a negative detection result.

In summary, the entire framework described by Eqn 1 - 4 requires operations such as multiplication, division, square root, and arctangent, which need complex hardware design and intensive computation.

3. METHOD

In this section, a list of acceleration strategies is proposed and explained in detail.

3.1 Accelerate Multiplication and Division

If x is a positive normal floating point number represented in IEEE754 format, then $x = 2^{e_x}(1 + m_x)$ where e_x is the exponent and m_x is the mantissa of x. Let y equal to the logarithm of x with base 2, then

$$y = \log_2 x \approx e_x + m_x + \sigma \qquad \text{Eqn 5}$$

where σ is a free parameter used to tune the approximation [2]. For example, $\sigma \approx 0.0430357$ yields the optimal approximation for float32. Moreover, it is not hard to derive the approximation for $x = 2^y$ given y from reversing Eqn 5. First of all, σ is subtracted from y and then e and m components are separated. Specifically, $e_x \approx \lfloor y - \sigma \rfloor$ and $m_x \approx y - \sigma - e_x$. Again, e_x is the exponent and m_x is the mantissa of the approximated $x \approx 2^y$.

After deriving the approximation of log2 and pow2 functions, the multiplication and division functions can be approximated as [3],

$$x * y = \text{pow2}(\log2(x) + \log2(y)) \qquad \text{Eqn 6}$$
$$x/y = \text{pow2}(\log2(x) - \log2(y)) \qquad \text{Eqn 7}$$

In practice, the e and m do not need to be computed explicitly and the calculation can be simplified further with barely two integer ADD and SHIFT operations (details in Appendix).

3.2 Accelerate Square Root

Given two floating point numbers x and y represented in IEEE754 format, a family of non-linear functions can be approximated as long as they satisfy the function $y = x^p$, for $-1 \leq p \leq 1$ [2][4]. Let x_{int} be the value of x if it is interpreted as an integer rather than float, then

$$y \approx (1-p) \cdot L \cdot (B - \sigma_{int}) + p \cdot x_{int} \qquad \text{Eqn 8}$$

where L is the scale; B is base; σ a free parameter used to tune the approximation [2][4]. For instance, to calculate $y = x^{1/2}$ (ie. $p = 0.5$) in standard IEEE754 float32 format, we have $L = 2^{23}$(for 23bit mantissa), $B = 127$ (for 7 bit exponent), and $\sigma = 0.0450465$ (that is, σ_{int}=-0x4B0D2) tuned for minimal approximation error (i.e. ±3.5%), then $y \approx (1/2)(2^{23})(127-0.0450465) + x_{int}/2$. This equation can be easily computed by right-shifting x_{int} by 1 bit and adding a constant to it. Similarly, functions $y = x^{-1}$, $y = x^{-1/2}$, etc… can be derived based on Eqn 8.

3.3 Accelerate Arctangent

The last function to accelerate is arctangent of y/x for the orientation of the gradients. A linear approximation method has been explained in [5] based on first-order Lagrange interpolation, where arctan() can be approximated by scaling with a constant. Depending on the relationship between x and y in each of the four quadrants. Note that the constant scaling factor, π/4, can be removed from Eqn 9. Instead, we can scale the indices of histogram bins while creating the histogram. The division of y and x can be calculated by Eqn 7 derived in Section 3.1. The above approximation for arctan(y/x) produces a maximum absolute error of 5º and average error close to 0º (due to cancellation). Such error is considered tolerable for the HOG feature descriptor, as later demonstrated in the experiment section.

3.3 Fully Accelerate HOG+SVM required Operations

In the end we can put them all together for a complete picture about the acceleration and approximation of HOG & SVM, as illustrated in Table 1.

Table 1. Approximation Summary

	Functions	Approx. Eqn	Required Ops
HOG Func	$g = \left(g_x^2 + g_y^2\right)^{1/2}$	(5),(6),(8)	5 add, 1fadd
	$\theta = \text{atan}(g_y/g_x)$	(5),(7),(9)	2 add
	$H = H/\lVert H \rVert_2$	(5),(6),(8)	2 add, 1fadd
SVM Func	$\mathbf{W}^{\mathrm{T}}\mathbf{H} + b$	(5), (6)	1add, 1fadd

4. EXPERIMENTS AND RESULTS

We tested our accelerated algorithms with two datasets: 1) INRIA person dataset consisting of more than 2000 images of person [6]; 2) a digit dataset consisting of 5000 images of digits.

4.1 HOG + binary SVM on INRIA dataset

In this test, half of dataset (1111 images) cropped to size 64x128 are used for training and the rest for testing. Each image is divide into 8x16 cells each containing 8x8 pixels. For each cell, a 1x9 gradient histogram is computed and the histograms of every 2x2 cells are concatenated to form block histograms. After histogram normalization, all block histograms are combined into a 4608x1 feature vector, which is fed to a binary-SVM classifier. The full acceleration for all HOG-related and SVM-related operations are implemented for both training and inference. To further reduce computation, all data and operations are processed as half-precision (ie float16) with 5-bit exponent and 10-bit mantissa. Table 2 shows classification result by the accelerated approach and the standard approach.

4.2 HOG + multi-class SVM on digit dataset

In this test, half of the dataset (2500) images of size 20x20 are selected for training and the rest for testing. The classification accuracy is shown in Table 2 for both standard and the accelerated method.

Table 2. Classification result comparison

Dataset	Accuracy by Standard Method (float64)	Accuracy by Accelerated Method (float16)
INRIA	99.64%	98.11%
Digit	88.72%	86.12%

5. CONCLUSION

In this paper a simple and effective acceleration strategy for HOG feature descriptor and SVM classifier is proposed and tested on real datasets. Only ADD operations are used for training/inference to replace complex mathematical functions such as multiplications, division, square-root and arctangent. Such acceleration greatly reduces hardware complexity, therefore reduces latency and power consumption, with only small trade-off in classification accuracy, which may well acceptable for many computer vision applications.

REFERENCES

[1] Dalal, N. & Triggs, B. Histograms of Oriented Gradients for Human Detection. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. II, pages 886-893, June 200

[2] https://en.wikipedia.org/wiki/Fast_inverse_square_root, last accessed May 14, 2020

[3] Mitchell, J.N., Computer multiplication and division using binary logarithms. IRE Trans. Electron. Computers, vol. 11. Pp. 512-517, Aug. 1962

[4] Hennessey, J. & Patterson, D. A. *Computer Organization and Design (2nd ed.)* (Morgan Kaufmann Publishers, 1998)

[5] Rajan, S., Wang S., Inkol, R. and Joyal, A. Efficient approximations for the arctangent function. *IEEE Signal Processing Magazine*, vol. 23, no. 3, pp. 108-111, May 2006

[6] INRIA dataset, http://pascal.inrialpes.fr/data/human/, last accessed May 14, 2020