FAST CONVOLUTION ALGORITHM FOR REAL-VALUED FINITE LENGTH SEQUENCES

Weiwei Wang, Victor DeBrunner, Linda S. DeBrunner

ww20br@my.fsu.edu, victor.debrunner@eng.famu.fsu.edu, linda.debrunner@eng.famu.fsu.edu
Florida State University
Department of Electrical and Computer Engineering,

ABSTRACT

The Fast Fourier Transform (FFT)-based convolution is the most popular fast convolution algorithm. In past work, we developed the Discrete Hirschman Transform (DHT)-based convolution. When compared to the FFT-based convolution, our DHT-based convolution can reduce the computational complexity by a third. Recently, we developed a comprehensive DFT algorithm where every calculation is natively real-valued (RV) dot products. In this paper, we first apply the natively real-valued DFT to linear convolution. We call this method the RV-based convolution. The arithmetic analysis reveals that it efficiently reduces the operation counts. The algorithm is fast regardless of length.

Index Terms— BF convolution, FFT-based convolution, DHT-based convolution, RV-based convolution

1. INTRODUCTION

The Brute Force (BF) convolution is defined as the sum of the dot product of two sequences after one is time reversed and shifted. The FFT-based convolution is much more computationally attractive than the BF convolution for two long sequences. It lowers the computational complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N\log_2 N)$ [1], where N is the convolution length. However, when the length of x and h differ significantly, the supremacy of FFT-based convolution suffers. The DHT-based convolution [2] that combines overlap-add and the Hirschman Optimal Transform (HOT) [3] does not suffer from this drawback. The HOT is based on periodic shifts of the FFT, partitioning the long sequence x into many sub-sequences. Using the small sub-sequences can reduce the computational load. In [4], the Optimal DHT-based convolution was proposed with an approach for determining the best parameter choices for the largest reduction in computational load. In [5], the author presents a new implementation of the real-valued split-radix FFT, an algorithm that uses fewer operations than any other real-valued power-of-2-length FFT.

But, these FFT-based convolutions and DHT-based convolution require multiplications and additions with complex numbers. A complex number multiplication requires 4 real-valued multiplications and 2 real-valued additions (4/2 algorithm), as well as the requisite added memory for temporary storage. This produces an "irregular" construct in any linear algebra computation. In [6] the authors presented a natively real-valued DFT where every calculation uses only vector products and vector-scalar products. The natively real-valued FFT [7] combines the natively real-valued DFT and the prime factor FFT (PFA) [8]. It requires the same number of computations as the Cooley-Tukey algorithm for data lengths $N = 2^n$. For other lengths, it is substantially superior to other FFT methods.

This research is very helpful to hardware implementation. In [9], the author proved fast convolution can reduce the processing time for VLSI and ASIC implementation. The HOT based OFDM [10] is efficient for FPGA implementation.

In this paper, we consider only real-valued sequences. The operation counts are the non-trivial real multiplications and additions. In Section 2, we review the DHT-based convolution. Section 3 is our proposed RV-based convolution algorithm. Section 4 is an example. Section 5 shows the computation complexity analysis. Conclusions and our future work are discussed in Section 6.

2. THE DHT-BASED CONVOLUTION

The Brute Force convolution: $y(k) = \sum_{n=0}^{N-1} h(n)x(k-n), k = 1, ..., N$ We represent the operation of convolution symbolically as $y = h * x$. In general, the FFT converts convolution to element-wise multiplication ($\odot$): $y = \mathrm{FFT}^{-1}\{\mathrm{FFT}\{h\} \odot \mathrm{FFT}\{x\}\}$. Of course, it needs to pad the sequences with zeros to ensure cyclic convolution is equal to the desired linear convolution.

We repeat a few of the salient results from [3, 4] to aid the reader in understanding the DHT-based convolutions. L_x and L_h denote the length of sequence x and h, respectively. Parameter J ($J > L_h$) is the length of the sub-sequences of x. K_{DHT} is the length of the FFTs for the DHT computation. It is normally chosen as a power of 2 and larger than $J + L_h - 1$. Parameter L is the number of segments as $L = \lceil \frac{L_x}{J} \rceil$. Between every J elements of x and at the end of the sequence insert $L_h - 1$ zeros to generate an LK_{DHT}-point sequence $\hat{x}$. This sequence $\hat{x}$ is multiplied by a permutation matrix P of

size $LK_{\mathrm{DHT}} \times LK_{\mathrm{DHT}}$ to generate a reordered sequence

$$\begin{aligned} \widetilde{x}(b+1+(a-1)L) &= \hat{x}(a+K_{\mathrm{DHT}}b), \\ a &= 1,\ldots,K_{\mathrm{DHT}}, \quad b = 0,\ldots,L-1. \end{aligned} \tag{1}$$

So, matrix P has ones in the coordinate pairs $(b+1+(a-1)L, a+K_{\mathrm{DHT}}b)$. Then, do an LK_{DHT}-point DHT transform $\widetilde{x}$ to $\widetilde{X}$. Note that H is the K_{DHT} point DFT of h, and that H should be pre-computed. Where each $H(i), i = 1, 2, \ldots, K$, is repeated L times as:

$$\widetilde{H} = [H(1)\cdots H(1)\cdots H(K_{\mathrm{DHT}})\cdots H(K_{\mathrm{DHT}})]^{\mathrm{T}} \tag{2}$$

Next, we compute the IDHT of the dot product $\widetilde{H} \odot \widetilde{X}$ to get $\widetilde{y}$. Then $\widetilde{y}$ is rearranged according to the inverse of the permutation matrix P^{T} to get $\hat{y}$. Then, the last $L_h - 1$ numbers and the first $L_h - 1$ numbers of two contiguous K_{DHT}-point segments are overlap-added. The result is the linear convolution of x and h.

3. THE RV-BASED CONVOLUTION

RV-based convolution is derived from DHT-based convolution. If we use the natively real-valued FFTs instead of the short FFTs found in the DHT-based convolution, then we get our natively RV-based convolution.

If x is a sequence of length L_x, the DFT of x is defined by $X(k) = \frac{1}{\sqrt{L_x}} \sum_{n=0}^{L_x-1} x(n) e^{-j\frac{2\pi}{L_x}nk} = Fx$. where F is the Fourier transform matrix. The matrix F can be diagonalized to $F = V\Lambda V^{-1}$ using the Jordan normal representation, where V is the eigenvector matrix of F. If we choose the orthonormal eigenvectors with real values, then $V^{-1} = V^{\mathrm{T}}$ and $F = V\Lambda V^{\mathrm{T}}$. Now, the only complex component is contained in Λ, which is the diagonal matrix containing the eigenvalues of F. If X is the DFT of x, then

$$X = Fx = V\Lambda V^{\mathrm{T}}x \tag{3}$$

Define $\mathrm{diag}(\cdot)$ used to convert a vector to a diagonal matrix. Assume $\hat{X} = \mathrm{diag}(V^{\mathrm{T}}x)$ and $\Lambda = \mathrm{diag}(\lambda)$, $\lambda = [1\cdots 1\ -1\cdots -1\ j\cdots j\ -j\cdots -j]^{\mathrm{T}}$ is a column vector containing the eigenvalues of F. Note that $\Lambda V^{\mathrm{T}}x = \hat{X}\lambda$, then Eq (3) becomes

$$X = V\hat{X}\lambda = V\hat{X}uz \tag{4}$$

where

$$u = \begin{bmatrix} 1 & \cdots & 1 & -1 & \cdots & -1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 0 & 1 & \cdots & 1 & -1 & \cdots & -1 \end{bmatrix}^{\mathrm{T}}$$

The two columns of product $V\hat{X}u$ are the real and imaginary part of the DFT result. The vector $z = [1\ \ j]^{\mathrm{T}}$ combines these into one complex and column vector.

The natively real-valued FFT [7] is a comprehensive algorithm to calculate the DFT using only real-valued computations. We combine it with the appropriable prime factor algorithm (PFA) or Cooly-Tukey algorithm to reduce the computational counts for fast convolution. Consider a $N = (L_x + L_h - 1) = N_1N_2$ point natively real-valued FFT. This results in an N_1 point DFT followed by an N_2 point DFT. N_1 and N_2 are co-prime factors of N.

In the natively RV-based convolution, J is the length of the sub-sequences split from x and $J > L_h$. Parameter K_{RV} is the length of the RV-based FFT. Unlike the DHT-based convolution, we do not need to pad zeros to make K_{RV} equal to power of 2. K_{RV} is equal to $J + L_h - 1$. Parameter L is the number of segments, i.e. $L = \lceil \frac{L_x}{J} \rceil$. If the last sub-sequence is shorter than J, we should pad some zeros so that it is length J. Insert $L_h - 1$ zeros between every J elements of x and use the matrix P to generate the new indexed sequences. Then, compute the FFTs and do the dot product of $(\widetilde{H} \odot \widetilde{X})$ and its inverse Fourier transform. The last step is to do the overlap-add to get the convolution result.

4. EXAMPLE

To see the differences between the RV-based convolution to other convolution methods, we compute an example. Consider the 10-point convolution of a 9-point x and a 2-point h, where $x = [1, 2, 3, 4, 5, 6, 7, 8, 9]$ and $h = [1, 2]$. The linear convolution of sequence x and h is $y = x * h = [1, 4, 7, 10, 13, 16, 19, 22, 25, 18]$. Due to the length constraint of FFT, we need to pad zeros to h, x, and do 16-point FFTs to compute H and X. Then we compute the element-wise product, following its inverse FFT to obtain y. For DHT-based convolution, $J = 5$ and $L = \lceil \frac{L_x}{J} \rceil = \lceil \frac{9}{5} \rceil = 2$, K_{DHT} should be the power of 2; here we choose $K_{\mathrm{DHT}} = 8$ then $LK_{\mathrm{DHT}} = 16$. We need to pad x with 7 zeros to obtain the new sequence $\hat{x} = [1, 2, 3, 4, 5, 0, 0, 0, 0, 6, 7, 8, 9, 0, 0, 0, 0]$ and use the bigger matrix $P_{16\times 16}$ to produce $\widetilde{x} = P\hat{x}^{\mathrm{T}} = [1, 6, 2, 7, 3, 8, 4, 9, 5, 0, 0, 0, 0, 0, 0, 0]^{\mathrm{T}}$. Next, compute the 16-point DHT $\widetilde{X}$, $\widetilde{H} \odot \widetilde{X}$ and the 16-point IDHT $\widetilde{y} = [1, 6, 4, 19, 7, 22, 10, 25, 13, 18, 10, 0, 0, 0, 0, 0]^{\mathrm{T}}$. We form $\hat{y} = P^{\mathrm{T}}\widetilde{y} = [1, 4, 7, 10, 13, \mathbf{10}, \mathbf{0}, \mathbf{0}, \mid \mathbf{6}, \mathbf{19}, \mathbf{22}, 25, 18, \mathbf{0}, \mathbf{0}, \mathbf{0} \mid]$. Finally, the result is $y = [1, 4, 7, 10, 13, 16, 19, 22, 25, 18]$.

4.1. Computation of RV-based convolution

The natively real-valued FFT does not have any length constraint. In this example, $K_{\mathrm{RV}} = J + L_h - 1 = 5 + 2 - 1 = 6$, $LK_{\mathrm{RV}} = 12$. So, the new input sequence $\hat{x} = [1, 2, 3, 4, 5, 0, 6, 7, 8, 9, 0, 0]$ and according to (1), we have the 12×12 matrix P.

$$P = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1 \end{bmatrix}$$

Here, use P to produce $\widetilde{x} = P\hat{x}^{\mathrm{T}} = [1, 6, 2, 7, 3, 8, 4, 9, 5, 0, 0, 0]^T$ The 12-point RV-FFT of $\widetilde{X}$ and the pre-computed $\widetilde{H}$ are shown as following. Note there is a scaling by $\frac{1}{\sqrt{6}}$.

$$\widetilde{X} = \begin{bmatrix} 6.124 \\ 12.247 \\ -2.450 \\ -1.428 - 5.303j \\ -1.414j \\ 3.062 + 0.354j \\ 1.225 \\ -0.817 \\ 1.414j \\ 3.062 - 0.354j \\ -2.450 \\ -1.428 + 5.303j \end{bmatrix}, \widetilde{H} = \begin{bmatrix} 1.225 \\ 1.225 \\ 0.817 - 0.707j \\ 0.817 - 0.707j \\ -0.707j \\ -0.707j \\ -0.408 \\ -0.408 \\ 0.707j \\ 0.707j \\ 0.817 + 0.707j \\ 0.817 + 0.707j \end{bmatrix}$$

Then, multiply $\widetilde{H}$ term by term with $\widetilde{X}$ and compute the 12-point inverse RV-FFT of $(\widetilde{H} \odot \widetilde{X})$

$$\widetilde{y} = [1, 6, 4, 19, 7, 22, 10, 25, 13, 18, 10, 0]^T.$$

We form $\hat{y} = P^{\mathrm{T}}\widetilde{y} = [1, 4, 7, 10, 13, \mathbf{10}, | \mathbf{6}, 19, 22, 25, 18, \mathbf{0} |]$. At last, do the overlap-add to get the convolution of h with x is:

$$y = h * x = [1, 4, 7, 10, 13, 16, 19, 22, 25, 18].$$

5. COMPUTATIONAL COMPLEXITY

In this section, we want to consider the application of the convolution methods in a block-processing implementation of FIR filtering. In that case, we may assume that the DFT of h is known since it is the fixed filter. Thus, we need only count the computations involving signal x. Suppose x is the real-valued input sequence. We count the number of real multiplications M, and the number of real additions A.

First, let us see the computation counts of different Fourie transform algorithms because they are the base layer of convolutions. Tab. 1 shows the 6-point computation counts. Fig. 1 depicts the number of multiplications of length 4 to 65 points for different DFT methods. The DHT can be computed by shorter FFTs, such as 64-point DHT can be calculated by 4 groups of 16-point FFTs. So, it is already combined in Fig. 1. Both Tab. 1 and Fig. 1 show that the natively real-valued FFT can do any length of DFT. It has a lower computational load for the sequences that have lengths that are not a power of 2. If we use RV-FFT instead of the Cooley-Tukey FFTs, we do not need to add so many zeros to the convolution sequences. This will reduce the trivial computations caused by those zeros. When the convolution length is long, it is then does the more computationally attractive.

For RV-based convolution, the computations in $\widetilde{H} \odot \widetilde{X}$ are complex multiplications. A complex multiplication can be calculated using 4 real multiplications and 2 real additions

Table 1. The Comparison for different 6-point DFTs

Method	Muls	Adds	Tiwddle	Regularity
BF DFT	32	42	6	irregular
6_p FFT	28	30	6	irregular
8_p FFT	8	32	4	irregular
6_p RV-DFT	28	36	-	regular
6_p RV-FFT	20	24	-	irregular

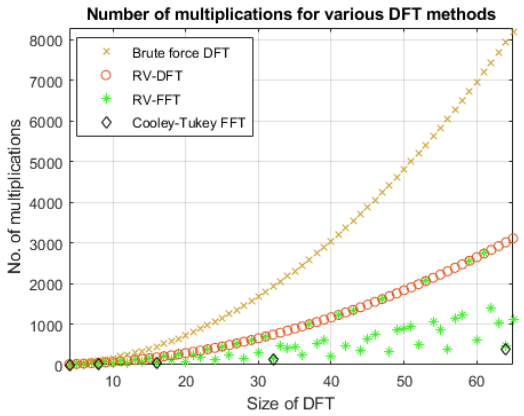

Fig. 1. Number of multiplications

(4/2 algorithm). The number of multiplications and additions, respectively, are:

$$\begin{aligned} M_{\mathrm{RV}_{\mathrm{conv}}} &= 2L(M_{\mathrm{RV}_{\mathrm{FFT}}}) + 4LK_{\mathrm{RV}} \\ A_{\mathrm{RV}_{\mathrm{conv}}} &= 2L(A_{\mathrm{RV}_{\mathrm{FFT}}}) + 2LK_{\mathrm{RV}} \end{aligned} \tag{5}$$

Fig. 1 also reveals that we can always find points where the RV-FFT based convolution is the best result. For example, the 20-point and 24-point RV-FFT requires 80 and 116 multiplications, respectively, whereas a 32-point FFT needs 136 multiplications. If we assume the short FFTs in the convolution algorithm are length 24, L_h is 12 and L_x is varied (its sub-sequence has the length of $J = 24 + 1 - 12 = 13$), use the RV-FFT based convolution will always have lower multiplication counts compared to the DHT-based Convolution. It reduces the multiplications by 39.0% and additions by 110%. But when compared to Cooley-Tukey FFT-based convolution, the results change somewhat. We still need to consider the convolution length N_{FFT} and N_{RV}.

For the convolution size from 37-point to 1161-point, consider the $L_h = 12$ and $J = 13$ to generate Tab. 2. Tab. 2 indicates that, except for some cases that the RV-FFT based convolution requires more computation counts than FFT-based convolution. Those cases have a common feature: the real convolution length $N = L_x + L_h - 1$ is equal to or close to the FFT-based convolution length. Consider when $N = 126$, we only need to pad 2 zeros to make the convolution length of the FFT-based convolution a power of 2 (128). The RV-based convolution with the length 216 requires more oper-

ations than the FFT-based convolution. This superiority of FFT-based convolution will shrink, or even disappear, when the convolution length is large, for example 1024-point convolution. Here, there is no need to zero-pad for the FFT-based convolution, but it still requires more operations than RV-FFT based convolution. Fig. 2 and Fig. 3 show the reduction in computations for those convolution algorithms where their differences are visible.

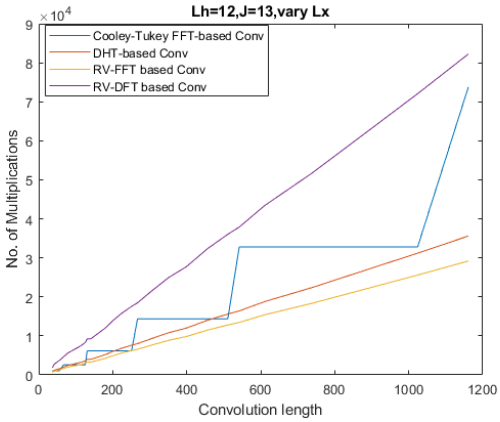

Fig. 2. Number of multiplications for N-point real-valued convolution($L_h = 12$).

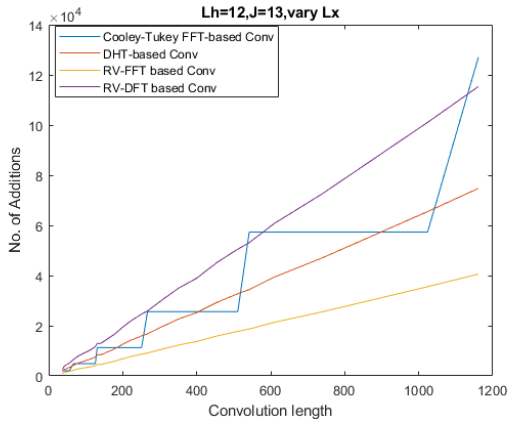

Fig. 3. Number of additions for N-point real-valued convolution($L_h = 12$).

6. CONCLUSION

The computational complexity analysis in Section 5 shows that the computation complexity of the convolution algorithm follows the computation complexity of the lower layer FFTs. When the convolution length is equal to or longer than 1024-point, the best overall computational choice is the RV-based convolution algorithm. Additionally, this method has no length constraint. It can efficiently reduce the real multiplications and the real additions required as compared to both FFT-based and DHT-based convolution. When the input sequence is long, the RV-FFT based convolution is more computationally attractive. Additionally, this implementation does not require any extra memory to store twiddle factors (which it does not possess!). Finally, if we use the RV-DFT to realize the convolution, the process is "regular" in that all computed dot products involve only real-valued vectors. This is very favorable in both hardware and software to the "irregular" construct when the vectors forming the dot product contain complex data, so that one complex multiplication and addition becomes many real computations. We will examine the superiority of real-valued dot product to the complex number multiplication in our future work.

Table 2. Number of multiplications for real-valued convolution($L_h = 12$, $J = 13$)

L_x	N	N_{FFT}	LK_{DHT}	LK_{RV}	RV-DFT Conv		Radix2 FFT Conv		DHT Conv		RV-FFT Conv	
					MULs	ADDs	MULs	ADDs	MULs	ADDs	MULs	ADDs
26	37	64	64	48	1848	2592	1040	2056	800	1680	**656**	912
45	56	64	128	96	3696	5184	**1040**	2056	1600	3360	1312	1824
65	76	128	160	120	4620	6480	2576	4872	2000	4200	**1640**	2280
102	113	128	256	192	7392	10368	**2576**	4872	3200	6720	2624	3648
115	126	128	288	216	8316	11664	**2576**	4872	3600	7560	2952	4104
130	141	256	320	240	9240	12960	6160	11272	4000	8400	**3280**	4560
189	200	256	480	360	13860	19440	6160	11272	6000	12600	**4920**	6840
240	251	256	608	456	17556	24624	**6160**	11272	7600	15960	6232	8664
256	267	512	640	480	18480	25920	14352	25608	8000	16800	**6560**	9120
500	511	512	1248	936	36036	50544	14352	25608	15600	32760	**12792**	17784
728	739	1024	1792	1344	51744	72576	32784	57352	22400	47040	**18368**	25536
1013	1024	1024	2496	1874	72072	101088	32784	57352	31200	65520	**25584**	35568
1150	1161	2048	2848	2136	82236	115344	**73744**	**126984**	35600	74760	**29192**	**40584**

7. REFERENCES

[1] H. J. Nussbaumer, "The fast Fourier transform," in *Fast Fourier Transform and Convolution Algorithms*. Springer, 1981, pp. 80–111.

[2] T. Przebinda, V. DeBrunner, and M. Ozaydın, "The optimal transform for the discrete hirschman uncertainty principle," *IEEE Transactions on Information Theory*, vol. 47, no. 5, pp. 2086–2090, 2001.

[3] V. DeBrunner and E. Matusiak, "An algorithm to reduce the complexity required to convolve finite length sequences using the hirschman optimal transform (HOT)," in *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, vol. 2. IEEE, 2003, pp. II–577.

[4] D. Xue, L. S. DeBrunner, and V. DeBrunner, "Reduced complexity optimal convolution based on the Discrete Hirschman Transform," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 5, pp. 2051–2059, 2021.

[5] H. V. Sorensen, D. Jones, M. Heideman, and C. Burrus, "Real-valued fast Fourier transform algorithms," *IEEE Transactions on acoustics, speech, and signal processing*, vol. 35, no. 6, pp. 849–863, 1987.

[6] R. Thomas, V. DeBrunner, and L. S. DeBrunner, "A Sparse Algorithm for Computing the DFT Using Its Real Eigenvectors," *Signals*, vol. 2, no. 4, pp. 688–705, 2021.

[7] R. Thomas, V. DeBrunner, and L. DeBrunner, "A natively real-valued FFT algorithm," in *2021 55th Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2021, pp. 908–912.

[8] I. J. Good, "The interaction algorithm and practical fourier analysis," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 20, no. 2, pp. 361–372, 1958.

[9] P. Katkar, T. Sridhar, G. Sharath, S. Sivanantham, and K. Sivasankaran, "VLSI implementation of fast convolution," in *2015 Online International Conference on Green Engineering and Technologies (IC-GET)*. IEEE, 2015, pp. 1–5.

[10] E. H. Krishna, K. Sivani, and K. A. Reddy, "Hardware implementation of OFDM transceiver using FPGA," in *2015 International Conference on Computer and Computational Sciences (ICCCS)*. IEEE, 2015, pp. 3–7.