Fault Tolerant CNN

Error Prone CNN Application

- Recent literature indicates that soft errors are inevitable in modern systems, from edge computing devices to supercomputers
- high-energy cosmic radiation
- Aging and wear of devices
- CNN inference applications often call for power-efficient and cost-efficient machine learning accelerators, which may adopt overclocking with voltage underscaling, incurring more soft errors than common hardware incurs.

- Resilient convolutional neural networks are essential for guaranteeing the correctness of inference applications.
- A single bit flip happened during CNN image classification could result in as much as 40% and 70% SDC rate in datapath and memory, respectively.
- Such high SDC rates would downgrade the CNN prediction accuracy dramatically.

[**SDC Rate: Silent Data Corruption (SDC) Rate**]

- Existing resilient solutions (ECC) are insufficient for protecting CNN inference applications against multiple soft errors.

CNN Basics

Notation	Description
D	Feature map, dimension is $4D$
W	Kernels, also called filters, dimension is $4D$
O	Output, dimension is $4D$
B	Bias, dimension is $1D$
C	Checksums
S	Block summations of O, corresponding to checksums
$\otimes$	Convolution operation
N	First dimension of D and O
M	First dimension of W and second dimension of O
Ch	Second dimension of D and W, also called channels
H	Third and fourth dimension of D
R	Third and fourth dimension of W
E	Third and fourth dimension of O
U	Stride size

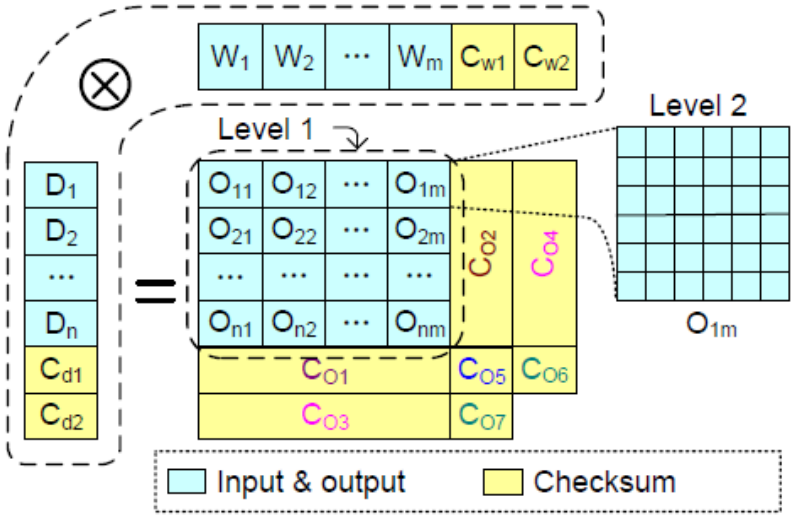
$$\mathbf{O}[n][m][x][y] = \mathbf{B}[m] + \sum_{k=0}^{Ch-1} \sum_{i=0}^{R-1} \sum_{j=0}^{R-1} \mathbf{D}[n][k][Ux+i][Uy+j] \times \mathbf{W}[m][k][i][j]$$

$$0 \leq n < N, 0 \leq m < M, 0 \leq x, y < E, E = \frac{H-R+U}{U}$$

The convolution operation involves two significant inputs: the feature map (fmap) **D**, $\mathbf{D} \in \mathbb{R}^{N \times Ch \times H \times H}$, and the convolutional kernels **W**, $\mathbf{W} \in \mathbb{R}^{M \times Ch \times R \times R}$.

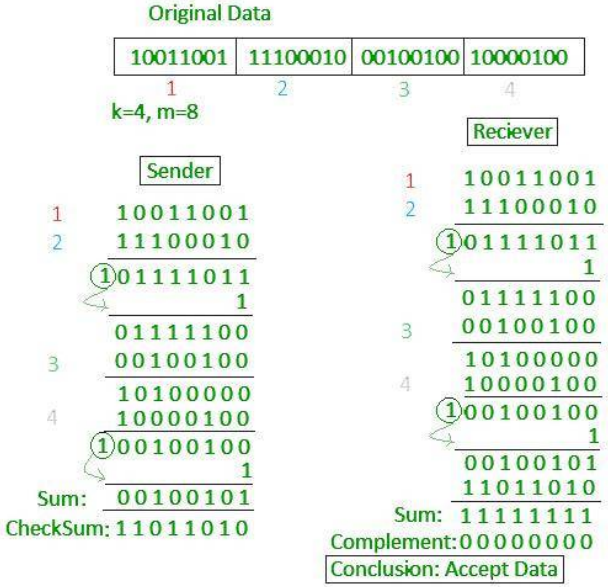
B, is applied to the output after convolution, and the final result is denoted as **O**, $\mathbf{O} \in \mathbb{R}^{N \times M \times E \times E}$.

Notation	Description
D	Feature map, dimension is $4D$
W	Kernels, also called filters, dimension is $4D$
O	Output, dimension is $4D$
B	Bias, dimension is $1D$
C	Checksums
S	Block summations of **O**, corresponding to checksums
$\otimes$	Convolution operation
N	First dimension of **D** and **O**
M	First dimension of **W** and second dimension of **O**
Ch	Second dimension of **D** and **W**, also called channels
H	Third and fourth dimension of **D**
R	Third and fourth dimension of **W**
E	Third and fourth dimension of **O**
U	Stride size

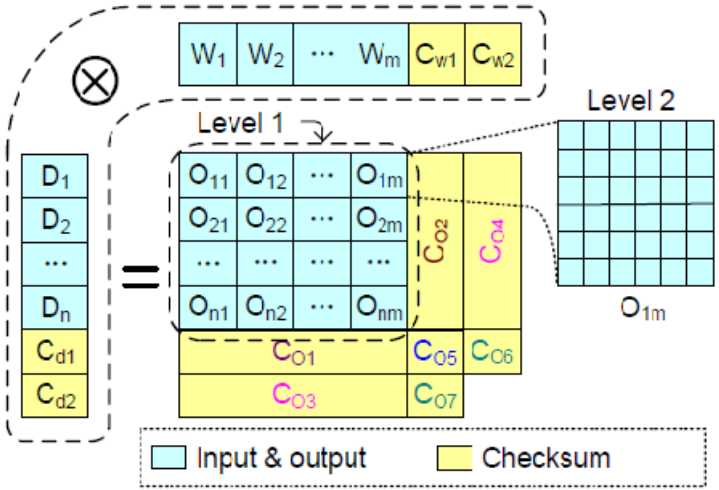


Checksum

- In checksum error detection scheme, the data is divided into k segments each of m bits.
- In the sender's end the segments are added using 1's complement arithmetic to get the sum. The sum is complemented to get the checksum.
- The checksum segment is sent along with the data segments.
- At the receiver's end, all received segments are added using 1's complement arithmetic to get the sum. The sum is complemented.
- If the result is zero, the received data is accepted; otherwise discarded.



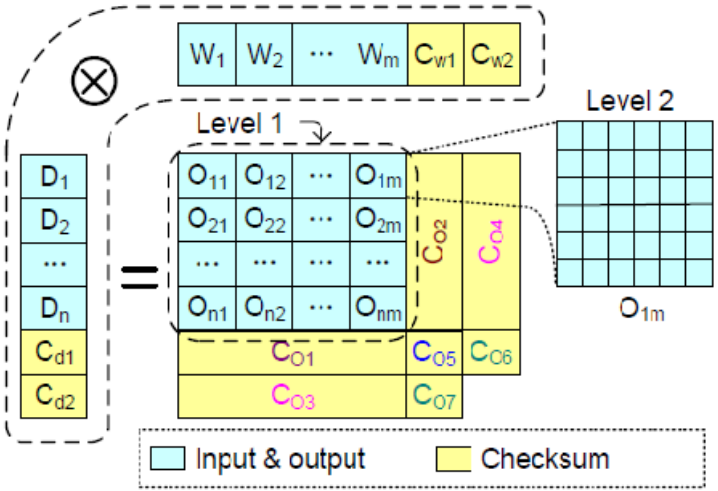
CNN Checksums



$$\mathbf{C}_{d1} = \sum_{n=0}^{N-1} \mathbf{D}_n$$
$$\mathbf{C}_{d2} = \sum_{n=0}^{N-1} n\mathbf{D}_n$$
$$\mathbf{C}_{w1} = \sum_{m=0}^{M-1} \mathbf{W}_m$$
$$\mathbf{C}_{w2} = \sum_{m=0}^{M-1} m\mathbf{W}_m$$

- Cd1 and Cw1 are calculated before the convolution operation,
- Any memory error striking D or W during the convolution would not affect Cd1 or Cw1.

CNN Checksums



$$
\begin{aligned}
\mathbf{C}_{o1} &= \mathbf{C}_{d1} \otimes \mathbf{W} \\
\mathbf{C}_{o2} &= \mathbf{D} \otimes \mathbf{C}_{w1} \\
\mathbf{C}_{o3} &= \mathbf{C}_{d2} \otimes \mathbf{W} \\
\mathbf{C}_{o4} &= \mathbf{D} \otimes \mathbf{C}_{w2} \\
\mathbf{C}_{o5} &= \mathbf{C}_{d1} \otimes \mathbf{C}_{w1} \\
\mathbf{C}_{o6} &= \mathbf{C}_{d1} \otimes \mathbf{C}_{w2} \\
\mathbf{C}_{o7} &= \mathbf{C}_{d2} \otimes \mathbf{C}_{w1}
\end{aligned}
$$

- The output O is represented in the form of blocks.
- Elements inside the same block are independent with respect to checksums.
- perform the checksum comparison independently for each element across blocks.
- Multiple soft errors in the same block can be detected and corrected independently.

Distributive property of Convolution

$$\mathbf{D}_1 \otimes \mathbf{W}_1 + \mathbf{D}_2 \otimes \mathbf{W}_1 =$$
$$\sum_{k=0}^{Ch-1} \sum_{i=0}^{R-1} \sum_{j=0}^{R-1} \mathbf{D}_1[k][Ux+i][Uy+j] \times \mathbf{W}_1[k][i][j]$$
$$+ \sum_{k=0}^{Ch-1} \sum_{i=0}^{R-1} \sum_{j=0}^{R-1} \mathbf{D}_2[k][Ux+i][Uy+j] \times \mathbf{W}_1[k][i][j]$$
$$= \sum_{k=0}^{Ch-1} \sum_{i=0}^{R-1} \sum_{j=0}^{R-1} (\mathbf{D}_1 + \mathbf{D}_2)[k][Ux+i][Uy+j] \times \mathbf{W}_1[k][i][j]$$
$$= (\mathbf{D}_1 + \mathbf{D}_2) \otimes \mathbf{W}_1$$

Similarly, we can get the following equation.

$$\mathbf{D}_1 \otimes \mathbf{W}_1 + \mathbf{D}_1 \otimes \mathbf{W}_2 = \mathbf{D}_1 \otimes (\mathbf{W}_1 + \mathbf{W}_2)$$

Notation	Description
D	Feature map, dimension is $4D$
W	Kernels, also called filters, dimension is $4D$
O	Output, dimension is $4D$
B	Bias, dimension is $1D$
C	Checksums
S	Block summations of O, corresponding to checksums
$\otimes$	Convolution operation
N	First dimension of D and O
M	First dimension of W and second dimension of O
Ch	Second dimension of D and W, also called channels
H	Third and fourth dimension of D
R	Third and fourth dimension of W
E	Third and fourth dimension of O
U	Stride size

$$\mathbf{C}_{o1}[m] = (\sum_{n=0}^{N-1} \mathbf{D}_n) \otimes \mathbf{W}_m = \sum_{n=0}^{N-1} (\mathbf{D}_n \otimes \mathbf{W}_m) = \sum_{n=0}^{N-1} \mathbf{O}_{nm}$$

$$\mathbf{C}_{o2}[n] = \mathbf{D}_n \otimes (\sum_{m=0}^{M-1} \mathbf{W}_m) = \sum_{m=0}^{M-1} (\mathbf{D}_n \otimes \mathbf{W}_m) = \sum_{m=0}^{M-1} \mathbf{O}_{nm}$$

These equations show the equality between the sum of output and the output checksums. Let $\mathbf{S}_{o1}$ and $\mathbf{S}_{o2}$ be the summation of the output, where $\mathbf{S}_{o1}[m] = \sum_{n=0}^{N-1} \mathbf{O}_{nm}$, $\mathbf{S}_{o2}[n] = \sum_{m=0}^{M-1} \mathbf{O}_{nm}$. We can compare $\mathbf{C}_{o1}$, $\mathbf{C}_{o2}$ with $\mathbf{S}_{o1}$, $\mathbf{S}_{o2}$ to detect, locate, and correct soft errors if they exist.

Row Checksum Scheme (RC)

$$\mathbf{C}_{o1} = \mathbf{C}_{d1} \otimes \mathbf{W}$$
$$\mathbf{C}_{o2} = \mathbf{D} \otimes \mathbf{C}_{w1}$$
$$\mathbf{C}_{o3} = \mathbf{C}_{d2} \otimes \mathbf{W}$$
$$\mathbf{C}_{o4} = \mathbf{D} \otimes \mathbf{C}_{w2}$$
$$\mathbf{C}_{o5} = \mathbf{C}_{d1} \otimes \mathbf{C}_{w1}$$
$$\mathbf{C}_{o6} = \mathbf{C}_{d1} \otimes \mathbf{C}_{w2}$$
$$\mathbf{C}_{o7} = \mathbf{C}_{d2} \otimes \mathbf{C}_{w1}$$

The row checksums used in this scheme are $\mathbf{C}_{o1}$ and $\mathbf{C}_{o3}$. $\mathbf{C}_{o3}$ is computed from convolution operation between $\mathbf{C}_{d2}$ and $\mathbf{W}$, and the related output summation is defined by $\mathbf{S}_{o3}[m] = \sum_{n=0}^{N-1} n \times \mathbf{O}_{nm}$.

For the detection of soft errors, we need to compare $\mathbf{C}_{o1}$ with $\mathbf{S}_{o1}$. If they are not equal to each other at location j, the error can be located by $i = \frac{\mathbf{C}_{o3}[j] - \mathbf{S}_{o3}[j]}{\mathbf{C}_{o1}[j] - \mathbf{S}_{o1}[j]}$ and j, and it can be corrected by adding $\mathbf{C}_{o1}[j] - \mathbf{S}_{o1}[j]$ to the block (i, j).

Column Checksum Scheme (ClC)

$$
\begin{aligned}
\mathbf{C}_{o1} &= \mathbf{C}_{d1} \otimes \mathbf{W} \\
\mathbf{C}_{o2} &= \mathbf{D} \otimes \mathbf{C}_{w1} \\
\mathbf{C}_{o3} &= \mathbf{C}_{d2} \otimes \mathbf{W} \\
\mathbf{C}_{o4} &= \mathbf{D} \otimes \mathbf{C}_{w2} \\
\mathbf{C}_{o5} &= \mathbf{C}_{d1} \otimes \mathbf{C}_{w1} \\
\mathbf{C}_{o6} &= \mathbf{C}_{d1} \otimes \mathbf{C}_{w2} \\
\mathbf{C}_{o7} &= \mathbf{C}_{d2} \otimes \mathbf{C}_{w1}
\end{aligned}
$$

The column checksums used in this scheme are $\mathbf{C}_{o2}$ and $\mathbf{C}_{o4}$. $\mathbf{C}_{o4}$ is defined by performing convolution operation between $\mathbf{D}$ and $\mathbf{C}_{w2}$, and the related output summation is defined as $\mathbf{S}_{o4}[n] = \sum_{m=0}^{M-1} m \times \mathbf{O}_{nm}$. To detect soft errors, we compare $\mathbf{C}_{o2}$ with $\mathbf{S}_{o2}$ first. If they are not equal to each other at location i, the error can be located by i and j $(= \frac{\mathbf{C}_{o4}[i] - \mathbf{S}_{o4}[i]}{\mathbf{C}_{o2}[i] - \mathbf{S}_{o2}[i]})$, and it can be recovered by adding $\mathbf{C}_{o2}[i] - \mathbf{S}_{o2}[i]$ to the block (i, j).

Checksum-of-Checksum Scheme (CoC/CoC-D)

$$\mathbf{C}_{o1} = \mathbf{C}_{d1} \otimes \mathbf{W}$$
$$\mathbf{C}_{o2} = \mathbf{D} \otimes \mathbf{C}_{w1}$$
$$\mathbf{C}_{o3} = \mathbf{C}_{d2} \otimes \mathbf{W}$$
$$\mathbf{C}_{o4} = \mathbf{D} \otimes \mathbf{C}_{w2}$$
$$\mathbf{C}_{o5} = \mathbf{C}_{d1} \otimes \mathbf{C}_{w1}$$
$$\mathbf{C}_{o6} = \mathbf{C}_{d1} \otimes \mathbf{C}_{w2}$$
$$\mathbf{C}_{o7} = \mathbf{C}_{d2} \otimes \mathbf{C}_{w1}$$

scheme involves neither D nor W but only their checksums, so it is named checksum-of-checksum scheme (or CoC scheme for short).

$$\mathbf{C}_{o5} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \mathbf{O}_{nm} = \mathbf{S}_{o5}$$
$$\mathbf{C}_{o6} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} m \times \mathbf{O}_{nm} = \mathbf{S}_{o6}$$
$$\mathbf{C}_{o7} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} n \times \mathbf{O}_{nm} = \mathbf{S}_{o7}$$

Checksum-of-Checksum Scheme (CoC/CoC-D)

$$\mathbf{C}_{o5} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \mathbf{O}_{nm} = \mathbf{S}_{o5}$$

$$\mathbf{C}_{o6} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} m \times \mathbf{O}_{nm} = \mathbf{S}_{o6}$$

$$\mathbf{C}_{o7} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} n \times \mathbf{O}_{nm} = \mathbf{S}_{o7}$$

$$\mathbf{C}_{o5} - \mathbf{S}_{o5} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \mathbf{O}'_{nm} - \mathbf{O}_{nm} = \delta$$

$$\mathbf{C}_{o6} - \mathbf{S}_{o6} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} m \times (\mathbf{O}'_{nm} - \mathbf{O}_{nm}) = j \times \delta$$

$$\mathbf{C}_{o7} - \mathbf{S}_{o7} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} n \times (\mathbf{O}'_{nm} - \mathbf{O}_{nm}) = i \times \delta$$

The location i, j can be obtained by $i = (C_{o7} - S_{o7})/\delta$ and $j = (C_{o6} - S_{o6})/\delta$. Then the soft error can be fixed by adding δ to $\mathbf{O}_{ij}$.

Analysis of Protection Ability for Convolution Checksum Schemes

Fault Model

Assumptions:

- transient faults in computational units
- data corruption faults (both transient and persistent) in memory (including cache).

- Fault represents a malfunction event, and we denote its corresponding symptom as soft error.

- Soft error protection includes error detection and error correction.
- Error detection means that the scheme can detect soft errors without knowing the exact location. Error correction means that the scheme can locate the soft error locations and recover the incorrect result.

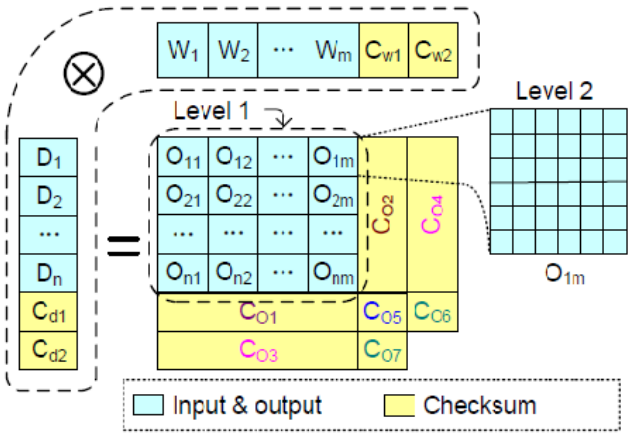
Limitation

- Without loss of generality, the following analysis considers at most one fault per convolution.
- One convolutional neural network contains several or even tens of convolutional layers, and the total forward execution time of a CNN model is usually within seconds.
- Thus, we can reasonably assume that at most one fault may strike to one convolutional layer, considering the short executing time of a single layer.
- Multiple faults per convolution can also be detected by our schemes and recovered by recomputing the corrupted convolutional layer.

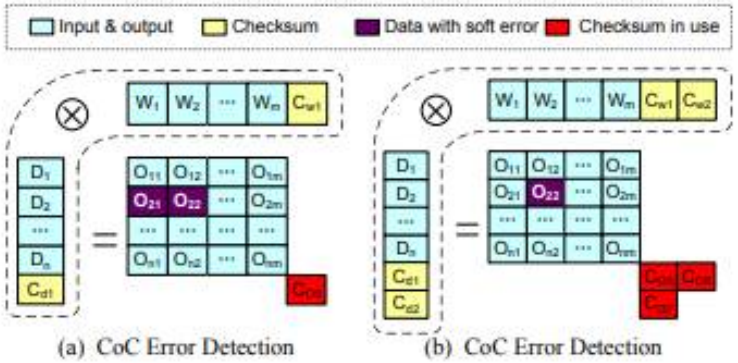
Analysis of Soft Error in D and W

- One fault occurring during the convolution execution can result in multiple soft errors in W and D.
- **The soft errors in W can be detected by comparing the checksum of W with Cw1 and corrected by reloading weights from the CNN model.**
- **The soft errors in D do not need correction because D will be discarded after convolution computation**.
- The resulting errors in the output can be detected and corrected by the checksums of the output, as demonstrated below.

- One fault during the convolution execution can result in corruption of one block row or column of O.
- By definition, the row i of O is computed by the ith block of D with W. Thus, one fault in D would result in at most one corrupted row.
- The column j of O is computed by D with the jth block of W. Thus, one fault in W would result in at most one corrupted column.
- Moreover, the intermediate result will be reused only by the same row or column, such that one fault in the computational units would corrupt only values in the same row or column.
- Accordingly, the soft error protection ability in the context of at most one corrupted row or column of O will be decided.



Soft Error Protection Ability of CoC Scheme



$$
\begin{aligned}
\mathbf{C}_{o1} &= \mathbf{C}_{d1} \otimes \mathbf{W} \\
\mathbf{C}_{o2} &= \mathbf{D} \otimes \mathbf{C}_{w1} \\
\mathbf{C}_{o3} &= \mathbf{C}_{d2} \otimes \mathbf{W} \\
\mathbf{C}_{o4} &= \mathbf{D} \otimes \mathbf{C}_{w2} \\
\mathbf{C}_{o5} &= \mathbf{C}_{d1} \otimes \mathbf{C}_{w1} \\
\mathbf{C}_{o6} &= \mathbf{C}_{d1} \otimes \mathbf{C}_{w2} \\
\mathbf{C}_{o7} &= \mathbf{C}_{d2} \otimes \mathbf{C}_{w1}
\end{aligned}
$$

When soft errors strike the input or output data:

Multiple soft errors can be detected by using only Co5. A single soft error in O can be corrected by CoC using all checksums including Co5, Co6, and Co7, as shown in Figure 2 (b). However, CoC cannot correct soft errors across multiple blocks in O.

Soft Error Protection Ability of CoC Scheme

$$\mathbf{C}_{o5} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \mathbf{O}_{nm} = \mathbf{S}_{o5}$$

$$\mathbf{C}_{o6} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} m \times \mathbf{O}_{nm} = \mathbf{S}_{o6}$$

$$\mathbf{C}_{o7} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} n \times \mathbf{O}_{nm} = \mathbf{S}_{o7}$$

$$\mathbf{C}_{o5} - \mathbf{S}_{o5} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \mathbf{O}'_{nm} - \mathbf{O}_{nm} = \delta$$

$$\mathbf{C}_{o6} - \mathbf{S}_{o6} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} m \times (\mathbf{O}'_{nm} - \mathbf{O}_{nm}) = j \times \delta$$

$$\mathbf{C}_{o7} - \mathbf{S}_{o7} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} n \times (\mathbf{O}'_{nm} - \mathbf{O}_{nm}) = i \times \delta$$

The location i, j can be obtained by $i = (C_{o7} - S_{o7})/\delta$ and $j = (C_{o6} - S_{o6})/\delta$. Then the soft error can be fixed by adding δ to $\mathbf{O}_{ij}$.

When soft errors strike the input or output data:

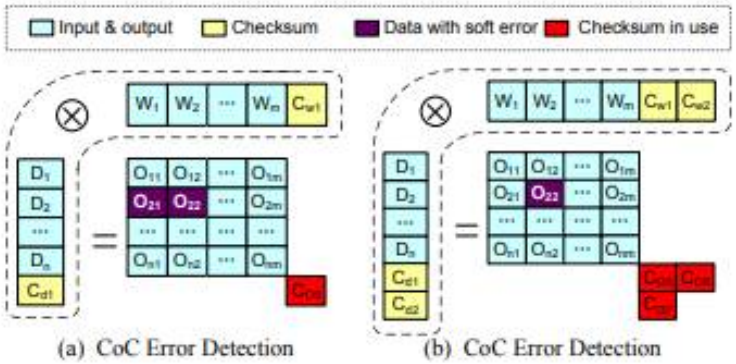
A single soft error in O can be corrected by CoC using all checksums including Co5, Co6, and Co7.

[CoC cannot correct soft errors across multiple blocks in O.]

when checksums are corrupted

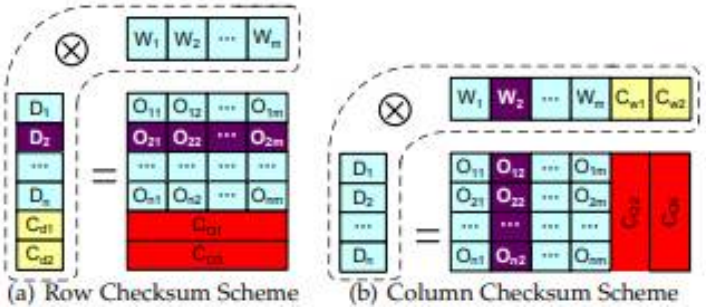
- the protection ability of the CoC scheme when soft errors happen inside the checksums
- . Such soft errors can cause inconsistency among the output checksums of CoC, which can be used for error detection.
- For example, if Cd1 is corrupted, leading to corrupted Co5 and Co6 with correct Co7.
- We can detect this abnormal pattern when comparing checksums with the summation of O to detect the input checksum corruption.
- The input D, W, and output O are clean and without soft errors since fault frequency is at most once per convolution. Thus, we can safely discard all the checksums and finish this convolution computation

$$
\begin{aligned}
\mathbf{C}_{o1} &= \mathbf{C}_{d1} \otimes \mathbf{W} \\
\mathbf{C}_{o2} &= \mathbf{D} \otimes \mathbf{C}_{w1} \\
\mathbf{C}_{o3} &= \mathbf{C}_{d2} \otimes \mathbf{W} \\
\mathbf{C}_{o4} &= \mathbf{D} \otimes \mathbf{C}_{w2} \\
\mathbf{C}_{o5} &= \mathbf{C}_{d1} \otimes \mathbf{C}_{w1} \\
\mathbf{C}_{o6} &= \mathbf{C}_{d1} \otimes \mathbf{C}_{w2} \\
\mathbf{C}_{o7} &= \mathbf{C}_{d2} \otimes \mathbf{C}_{w1}
\end{aligned}
$$



Ability of Row and Column Checksum Scheme

- the row checksum scheme and column checksum scheme are symmetric.
- the row checksum scheme can detect and correct soft errors if they are in the same row.
- If the soft errors are in the same column, the row checksum scheme can only detect soft errors; it has no correction ability.
- The column checksum scheme, on the contrary, can detect and correct errors located in the same column but fail to correct those appearing in the same row.



Revisit Row Checksum Scheme (RC)

$$\mathbf{C}_{o1} = \mathbf{C}_{d1} \otimes \mathbf{W}$$
$$\mathbf{C}_{o2} = \mathbf{D} \otimes \mathbf{C}_{w1}$$
$$\mathbf{C}_{o3} = \mathbf{C}_{d2} \otimes \mathbf{W}$$
$$\mathbf{C}_{o4} = \mathbf{D} \otimes \mathbf{C}_{w2}$$
$$\mathbf{C}_{o5} = \mathbf{C}_{d1} \otimes \mathbf{C}_{w1}$$
$$\mathbf{C}_{o6} = \mathbf{C}_{d1} \otimes \mathbf{C}_{w2}$$
$$\mathbf{C}_{o7} = \mathbf{C}_{d2} \otimes \mathbf{C}_{w1}$$

The row checksums used in this scheme are $\mathbf{C}_{o1}$ and $\mathbf{C}_{o3}$. $\mathbf{C}_{o3}$ is computed from convolution operation between $\mathbf{C}_{d2}$ and $\mathbf{W}$, and the related output summation is defined by $\mathbf{S}_{o3}[m] = \sum_{n=0}^{N-1} n \times \mathbf{O}_{nm}$.

For the detection of soft errors, we need to compare $\mathbf{C}_{o1}$ with $\mathbf{S}_{o1}$. If they are not equal to each other at location j, the error can be located by $i = \frac{\mathbf{C}_{o3}[j] - \mathbf{S}_{o3}[j]}{\mathbf{C}_{o1}[j] - \mathbf{S}_{o1}[j]}$ and j, and it can be corrected by adding $\mathbf{C}_{o1}[j] - \mathbf{S}_{o1}[j]$ to the block (i, j).

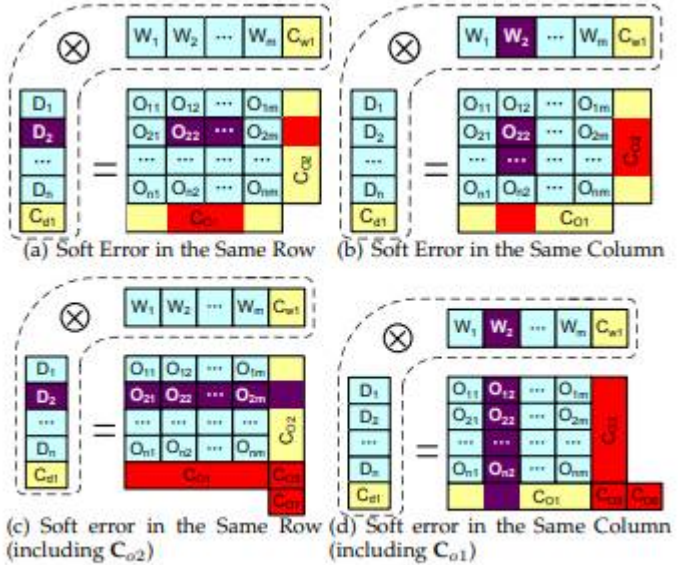
Revisit Column Checksum Scheme (ClC)

$$
\begin{aligned}
\mathbf{C}_{o1} &= \mathbf{C}_{d1} \otimes \mathbf{W} \\
\mathbf{C}_{o2} &= \mathbf{D} \otimes \mathbf{C}_{w1} \\
\mathbf{C}_{o3} &= \mathbf{C}_{d2} \otimes \mathbf{W} \\
\mathbf{C}_{o4} &= \mathbf{D} \otimes \mathbf{C}_{w2} \\
\mathbf{C}_{o5} &= \mathbf{C}_{d1} \otimes \mathbf{C}_{w1} \\
\mathbf{C}_{o6} &= \mathbf{C}_{d1} \otimes \mathbf{C}_{w2} \\
\mathbf{C}_{o7} &= \mathbf{C}_{d2} \otimes \mathbf{C}_{w1}
\end{aligned}
$$

The column checksums used in this scheme are $\mathbf{C}_{o2}$ and $\mathbf{C}_{o4}$. $\mathbf{C}_{o4}$ is defined by performing convolution operation between $\mathbf{D}$ and $\mathbf{C}_{w2}$, and the related output summation is defined as $\mathbf{S}_{o4}[n] = \sum_{m=0}^{M-1} m \times \mathbf{O}_{nm}$. To detect soft errors, we compare $\mathbf{C}_{o2}$ with $\mathbf{S}_{o2}$ first. If they are not equal to each other at location i, the error can be located by i and j $(= \frac{\mathbf{C}_{o4}[i] - \mathbf{S}_{o4}[i]}{\mathbf{C}_{o2}[i] - \mathbf{S}_{o2}[i]})$, and it can be recovered by adding $\mathbf{C}_{o2}[i] - \mathbf{S}_{o2}[i]$ to the block (i, j).

Ability of Full Checksum Scheme

- Full checksum scheme has the highest ability to correct soft errors.
- The scheme uses both the row checksum Co1 and column checksum Co2 so that it can correct soft errors in both directions.
- If soft errors exist in Co1, however, Co1 can no longer be used to locate or correct soft errors.
- To support error correction in this situation, we use checksum Co5 and Co6 from the CoC scheme to locate the corrupted column, and we then use Co2 to correct the soft errors.
- If soft errors exist in Co2, Co5 and Co7 are used to locate the corrupted row, and Co1 is used to correct the soft errors.



Soft Error Protection Ability of Full Checksum Scheme

Soft Error Protection Ability of CoC Scheme

$$\mathbf{C}_{o5} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \mathbf{O}_{nm} = \mathbf{S}_{o5}$$

$$\mathbf{C}_{o6} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} m \times \mathbf{O}_{nm} = \mathbf{S}_{o6}$$

$$\mathbf{C}_{o7} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} n \times \mathbf{O}_{nm} = \mathbf{S}_{o7}$$

$$\mathbf{C}_{o5} - \mathbf{S}_{o5} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \mathbf{O}'_{nm} - \mathbf{O}_{nm} = \delta$$

$$\mathbf{C}_{o6} - \mathbf{S}_{o6} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} m \times (\mathbf{O}'_{nm} - \mathbf{O}_{nm}) = j \times \delta$$

$$\mathbf{C}_{o7} - \mathbf{S}_{o7} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} n \times (\mathbf{O}'_{nm} - \mathbf{O}_{nm}) = i \times \delta$$

The location i, j can be obtained by $i = (C_{o7} - S_{o7})/\delta$ and $j = (C_{o6} - S_{o6})/\delta$. Then the soft error can be fixed by adding δ to $\mathbf{O}_{ij}$.

When soft errors strike the input or output data:

A single soft error in O can be corrected by CoC using all checksums including Co5, Co6, and Co7.

[CoC cannot correct soft errors across multiple blocks in O.]

Class assignment

Show with some example index values :

If soft errors exist in Co2, Co5 and Co7 are used to locate the corrupted row, and Co1 is used to correct the soft errors.

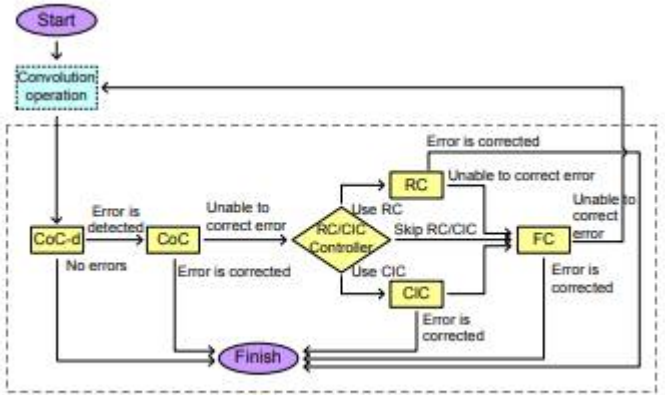
Ability of four schemes

- CoC scheme has the lowest error correction ability and that the full checksum scheme has the best error correction ability.
- The abilities of the row checksum scheme and column checksum scheme are higher than that of the CoC scheme but lower than that of the full checksum scheme.
- CoC-D can detect multiple soft errors but without correction ability.

The analysis here serves as the fundamental basis of our low-overhead high protection design.

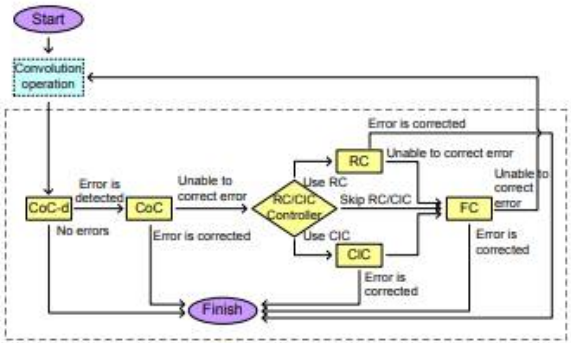
Multischeme Workflow for Soft Error Protection

- To achieve the highest protection ability and lowest overhead: multischeme workflow by integrating the four schemes.
- use CoC-D to detect errors because it has the lowest overhead.
- For the error correction, put CoC in the beginning because it is the most lightweight method.
- By comparison, FC has highest correction ability but also highest time overhead, so put it at the end of the workflow.



Multischeme Workflow Designed to Detect/Correct Soft Errors

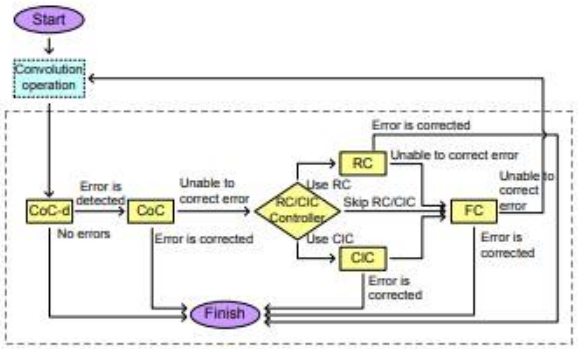
Multischeme Workflow for Soft Error Protection



Multischeme Workflow Designed to Detect/Correct Soft Errors

- The error detection modules will be executed for every execution whether there is a soft error or not. Thus, any unnecessary computations should be avoided in order to reduce the overall overhead.
- For instance, both CoC-D and FC are able to detect all the soft errors, but we adopt only CoC-D in the workflow for error detection because FC has a much higher overhead. RC and CIC cannot detect soft errors correctly if the checksum is corrupted.

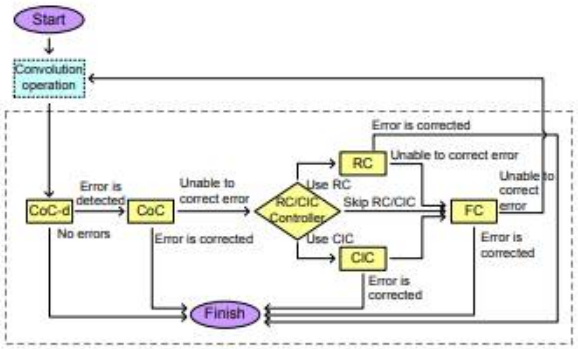
Multischeme Workflow for Soft Error Protection



Multischeme Workflow Designed to Detect/Correct Soft Errors

- The error correction module will not be executed until some soft errors are detected. The schemes in this module will be invoked to fix soft errors according to the workflow. If it fails to correct the errors due to inconsistency of checksum blocks or illegal error locations, the next-level scheme will be invoked.

Multischeme Workflow for Soft Error Protection

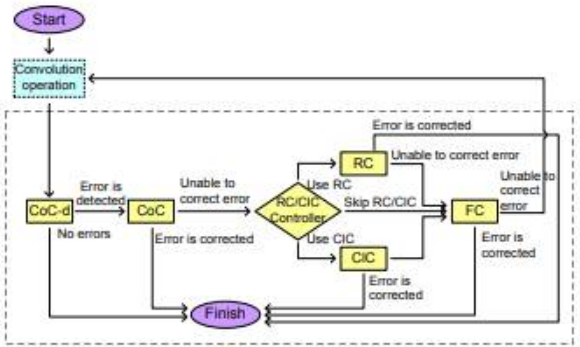


Multischeme Workflow Designed to Detect/Correct Soft Errors

- Checksums can be reused among different CNN schemes in the workflow, the runtime of the workflow is actually lower than the sum of all schemes' runtimes.
- For example, both CoC-D and CoC use Co5; if CoC-D detects soft errors and CoC is invoked to correct soft errors, CoC can save the time of computing Co5 and its corresponding summation So5, since they have been computed by CoC-D.

Multischeme Workflow for Soft Error Protection

- Consider, runtime of the workflow CoC+FC as t0
- the runtime of workflow CoC+RC as t1
- the runtime of workflow CoC+RC+FC as t2.



Multischeme Workflow Designed to Detect/Correct Soft Errors

- Enabling RC can fix some soft errors before FC, thus changing the runtime from t0 to t1.
- When RC fails to correct soft errors, however, FC still needs to be invoked; and the runtime will increase from t0 to t2.
- Denoting the probability of row soft errors by pr and the probability of column soft errors by pc, we can derive the average time saved by RC as ty = pr(t0 − t1) and the average time increase by RC as tn = pc(t2 − t0).
- In order to minimize the total runtime, RC should be enabled when ty > tn

Ref:

https://www.researchgate.net/publication/348144782_FT-CNN_Algorithm-Based_Fault_Tolerance_for_Convolutional_Neural_Networks

Code Ref

https://oneapi-src.github.io/oneDNN/v0/index.html

https://github.com/oneapi-src/oneDNN/blob/master/examples/cnn_inference_f32.cpp

https://github.com/oneapi-src/oneDNN/blob/master/examples/cnn_inference_int8.cpp