3D Crossbar Algorithms

1 Original

INPUT: N×M Crossbar (NN-chip), AM, IB, P_{th} and S_{th}
OUTPUT: Weighted NN-chip

$i \leftarrow [0\ to\ M-1]$
$j \leftarrow [1\ to\ K]$
$inp \leftarrow [1\ to\ p]$
$AM_c \leftarrow column\ of\ AM$
for each i in IB do
  $C_i \leftarrow i^{th}$ column of NN-chip
  for each j in IB[i] do
    for each inp in IB[i][j] do
      AM← Weight_Update(AM, inp, S_{th}, AM_c)
    end for
    $W_{AM} \leftarrow$ weights of AM
    NN-chip← Weight_Update(NN-chip, W_{AM}, P_{th}, C_i)
    RESET AM
  end for
end for
return Weighted NN-chip

2 Approach: Distribution Of Features

Following the 6th section of the previous paper, each one of the k layers of the 3D crossbar will be of i×M size such that i×k=N. We can use an AM crossbar of dimeansions i×1. Complexity: p×

INPUT: i×k×M Crossbar (NN-chip), i×M AM_1 Crossbar , AM_2 Crossbar (i×1), IB, P_{th} and S_{th}
OUTPUT: Weighted NN-chip

$inp \leftarrow [1\ to\ p]$
$AM_c \leftarrow column\ of\ AM$
for b from 0 to k-1 do
  $L_b \leftarrow b^{th}$ layer/plane of NN-chip
  $C_b \leftarrow b^{th}$ column of AM_1
  for each q in IB do
    for each j in IB[q] do
      for each inp in IB[q][j] do
        AM ← Weight_Update(AM, inp[i×b:i×(b+1)], S_{th}, AM_2)
      end for
      $W_{AM2} \leftarrow$ weights of AM_2
      $AM_1 \leftarrow$ Weight_Update(AM_1, W_{AM2}, P_{th}, C_b)
      RESET AM_2
    end for
  end for
  $W_{AM1} \leftarrow$ weights of AM_1
  NN-Chip ← Weight_Update2D(NN-Chip,W_{AM1}, P_{th}, L_b)
end for
return Weighted NN-chip

3 Approach: Distribution Of Classes

We can do the same distribution with classes instead of features. each one of the k layers of the 3D Crossbar will be of i×N size such that i×k=M. Here the dimensions of the AM Crossbar remain the same i.e. N×1.
Note: 'K' is number of sub-batches and 'k' is the distribution factor.

INPUT : i×k×N Crossbar (NN-chip), AM Crossbar, IB, P_{th} and S_{th}
OUTPUT : Weighted NN-chip

$j \leftarrow$ [1 *to* K]
$AM_c \leftarrow$ *column of AM*
for `b from 0 to k-1` **do**
  for `a from 1 to i` **do**
    c← b*i + a
    $C_c \leftarrow c^{th}$ column of NN-chip
    for `each j in IB[c]` **do**
      for `each inp in IB[c][j]` **do**
        AM← Weight_Update(AM, inp, S_{th}, AM_c)
      end for
      $W_{AM} \leftarrow$ weights of AM
      NN-chip ← Weight_Update(NN-chip, W_{AM}, P_{th}, C_c)
      RESET AM
    end for
  end for
end for
return Weighted NN-chip

Algorithm 1 Weight_Update2D(Crossbar, Weight, threshold, layer)

r ← number of rows in Crossbar
c ← number of columns in Crossbar
for `i ← 0 to r-1` **do**
  for `j ← 0 to c-1` **do**
    if `Weight[i][j] ≥ threshold` **then**
      $\text{Crossbar}_{layer(x)}$.append(i,j)
    end if
  end for
end for
$\text{Crossbar}_{layer(x)} \leftarrow V_{sc}$
layer(y) ← GND
return Crossbar