# An explanation of the SCD kernel algorithm

11 Dec. 2020

**Algorithm 1** SCD matrix and alpha profile calculation.

```
1:  procedure SCD($data[256][32], Alpha\_Profile[4096]$)
2:      for ($i = 0; i < 256; i++$){ do
3:          for ($j = 0; j <\quad ; j++$){ do
4:              Step 4.1: $x \leftarrow data[i] * Conjugate(data[j])$
5:              Step 4.2: $y \leftarrow FFT\_32(x)$
6:              $SCD\_Matrix \leftarrow \{y[31:24]\}\{y[7:0]\}$
7:              Step 4.3.1: $Calculate\ partial\ Alpha\_Profile$
8:              Step 4.3.2: $Merge\ partial\ Alpha\_Profiles$
9:          }# $end\ of\ inner\ loop$
10:         Step 4.3.3: $Update\ Alpha\_Profile$
11:     }# $end\ of\ outer\ loop$
12:     return $Alpha\_Profile$
```

Python code for the above algorithm (changed a few parts to match with the paper):

```
Sx = np.zeros((2*Np, 2*N), dtype=complex)
Mp = N // Np

for k in range(Np):
   for l in range(Np):
      XF2 = np.fft.fft(XD[:,k] * np.conjugate(XD[:,l]))
      XF2 = np.fft.fftshift(XF2)

      i = k+l
      a = int(((k-l)/float(Np) + 1.) * N)
      Sx[i,a-Mp: a+Mp] = XF2[(P-Mp): (P+Mp)]  # assuming we are using XF2[8:15]
return Sx
```

In this example, XD is of size Np*P which is 256*32 = 8,192, while Sx is of size 2*Np*2*N which is 2*256*2*2048 = 2,097,152. If we choose to output the whole Sx matrix, then the output bandwidth is much larger than input bandwidth. This is where I found the I/O is unbalanced. Instead, we can use the internal Sx to continue the calculation of alpha profile and finally output alphas only. Then the number of outputs is reduced to 4,096.

Those figures are copied from the paper to have a more virtualized understanding.
In Figure 1, XD and its conjugate are coming from the following matrix of size Np*P. Each row will do a element-wise multiplication with other rows (including itself). After the complex multiplication and the P-point FFT, half of the FFT outputs (can be either middle half, or top/bottom half) are used to form a tiny block of the SCD matrix in Figure 2. E.g., row 0 and conjugate of row 0 will form a block of index (0, 0).

Our idea is to map different rows (diagonal view) in Figure 2 to different PEs (i.e. Row 0 maps to PE 0, Row 1 maps to PE 1, vice versa). However, it is not that straightforward to calculate the alpha profile (max values) among different PEs because of the location of different blocks. Therefore, I proposed to calculate the partial alphas (8 values) in each iteration and forward them to the next PE till the end of one row (horizontal view). We only need to do the calculation of the black-color marked triangle part because of the SCD symmetry. Take the kth iteration for example, partial alpha = max(bottom 8 elements of kth iteration, top 8 elements of (k-1)th iteration, partial alpha from previous PE). I have shown the instruction scheduling in the other document and it proved to be efficient.

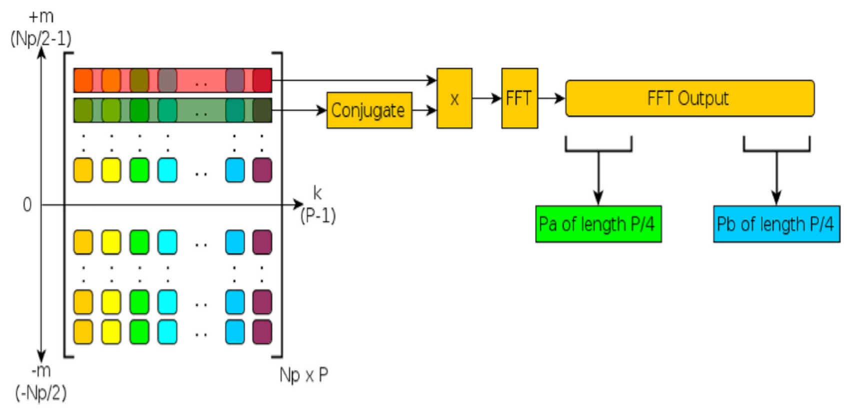

Figure 1

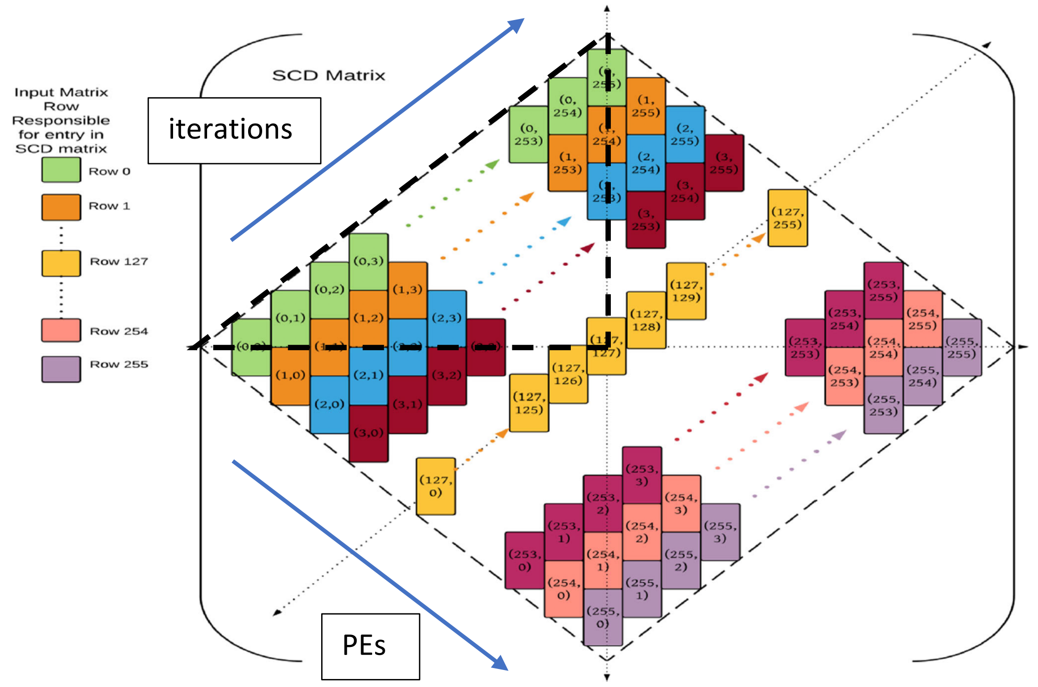

**Figure 7** Alpha profile generation for classification.

Figure 2