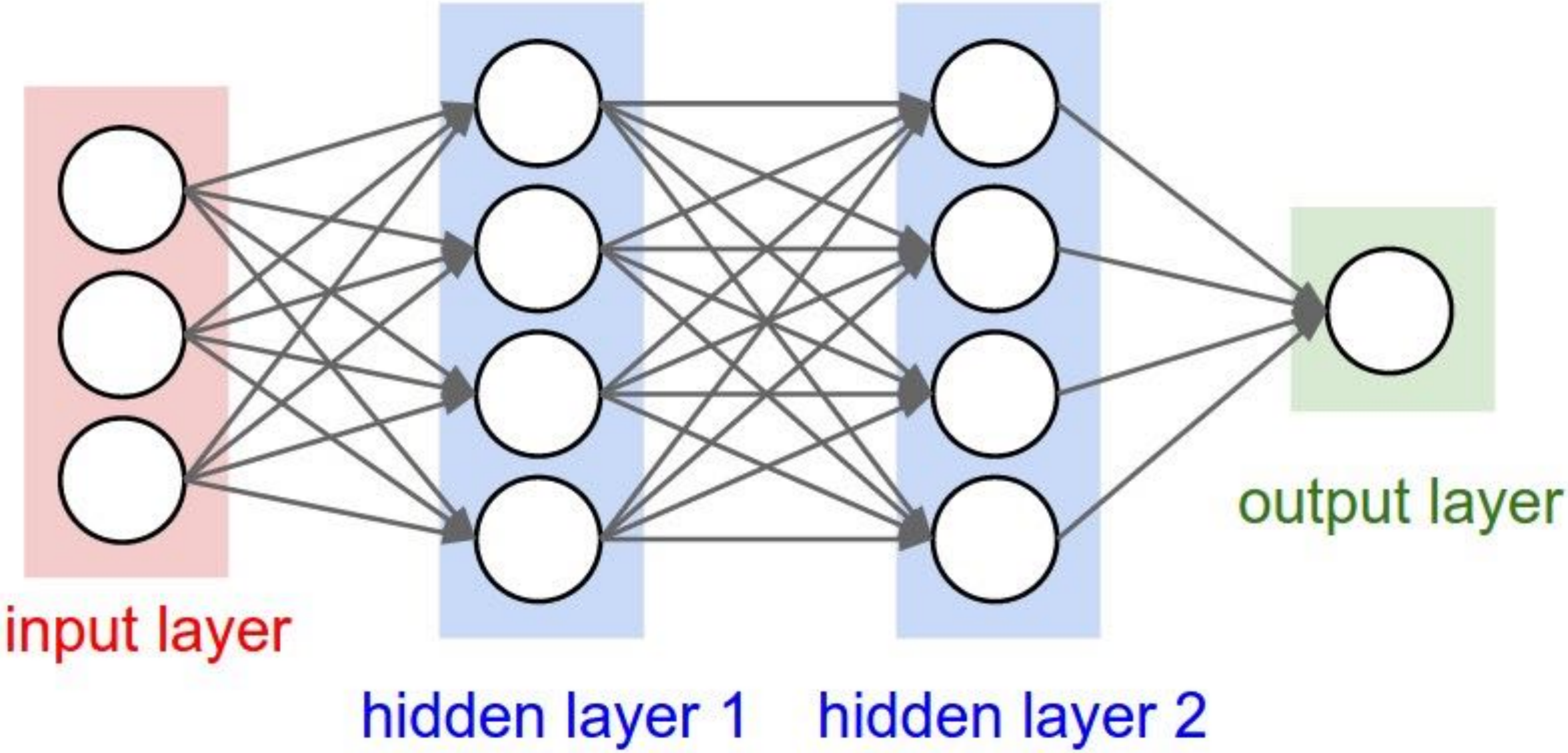
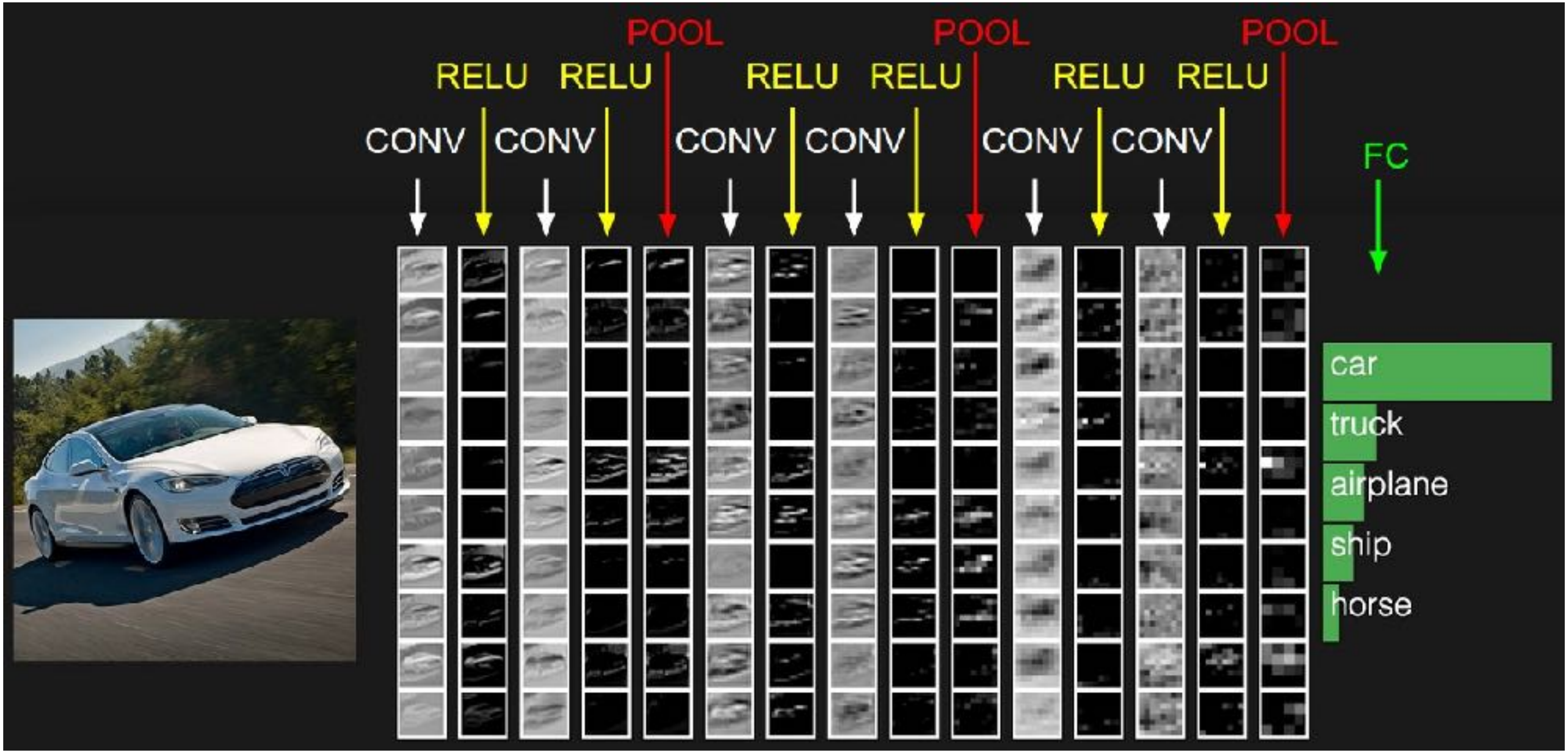
Lab Project

- Optimize a convolution for execution either on CPU or on GPU
- A convolution is implemented as a matrix multiplication

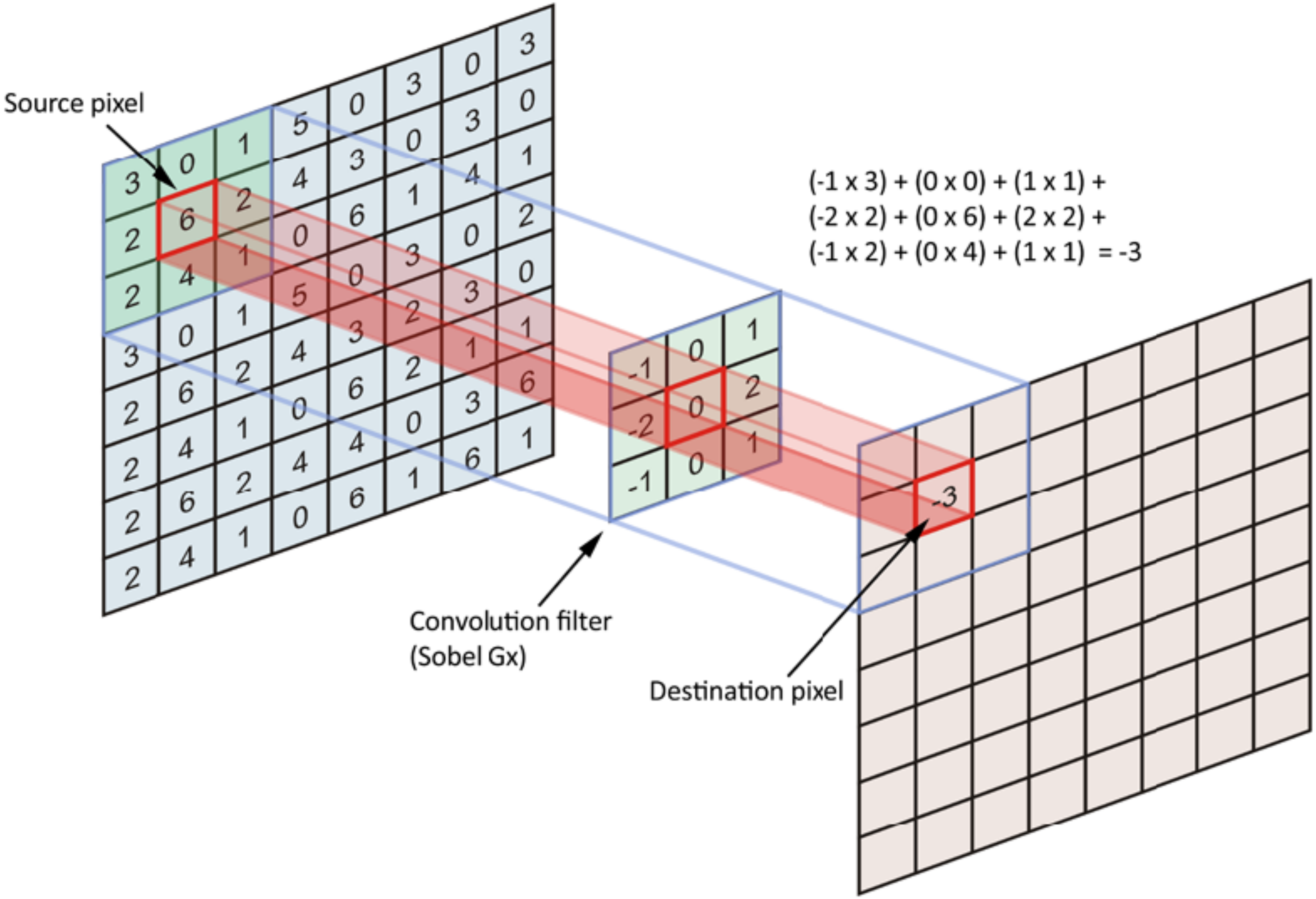
Neural Networks



Convolutions



Convolutions



3	0	1	5	0	3	0	3
2	6	2	4	3	0	3	0
2	4	1	0	6	1	4	1
3	0	1	5	0	3	0	2
2	6	2	4	3	2	3	0
2	4	1	0	6	2	1	1
2	6	2	4	4	0	3	6
2	4	1	0	6	1	6	1

-1	0	1
-2	0	2
-1	0	1

(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3

Implementing Convolution

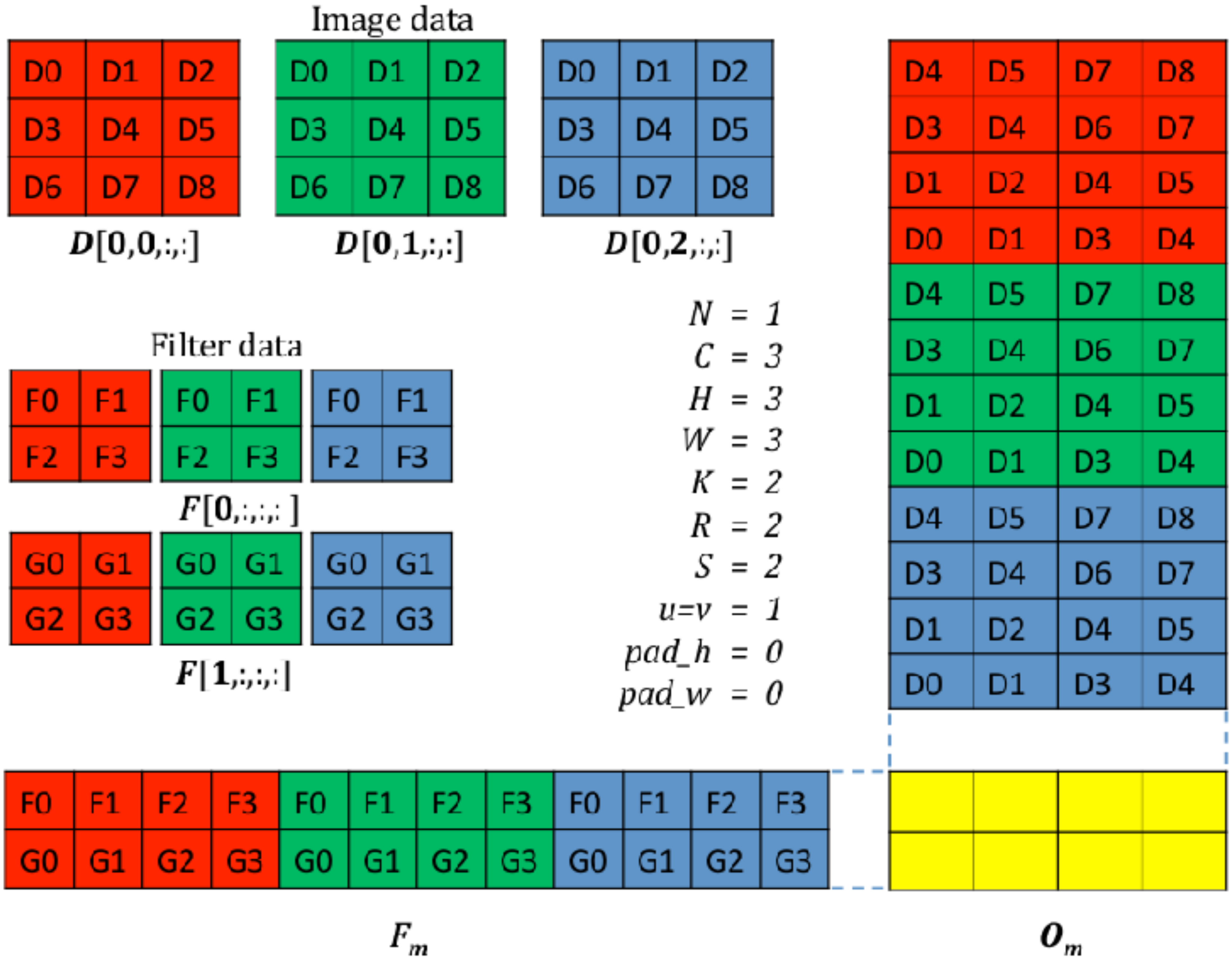


Image data

D0	D1	D2
D3	D4	D5
D6	D7	D8

$D[0,0,:,:]$

D0	D1	D2
D3	D4	D5
D6	D7	D8

$D[0,1,:,:]$

D0	D1	D2
D3	D4	D5
D6	D7	D8

$D[0,2,:,:]$

Filter data

F0	F1	F0	F1	F0	F1
F2	F3	F2	F3	F2	F3

$F[0,:,:,:]$

G0	G1	G0	G1	G0	G1
G2	G3	G2	G3	G2	G3

$F[1,:,:,:]$

$N = 1$
$C = 3$
$H = 3$
$W = 3$
$K = 2$
$R = 2$
$S = 2$
$u{=}v = 1$
$pad\_h = 0$
$pad\_w = 0$

D4	D5	D7	D8
D3	D4	D6	D7
D1	D2	D4	D5
D0	D1	D3	D4
D4	D5	D7	D8
D3	D4	D6	D7
D1	D2	D4	D5
D0	D1	D3	D4
D4	D5	D7	D8
D3	D4	D6	D7
D1	D2	D4	D5
D0	D1	D3	D4

F0	F1	F2	F3	F0	F1	F2	F3	F0	F1	F2	F3
G0	G1	G2	G3	G0	G1	G2	G3	G0	G1	G2	G3

F_m

O_m

Matrix Multiplication

C

```
for (i = 0; i < N; i++)
    for (j = 0; j < N; j++)
   {
      C[i][j] = 0;
      for (k = 0; k < N; k++)
          C[i][j] += A[i][k] * B[k][j];
    }
```

Time Measurements

C++

```
#include <chrono>

…
auto start = std::chrono::high_resolution_clock::now();
…
auto end = std::chrono::high_resolution_clock::now();
std::chrono::duration<double,std::milli> duration = end - start;
std::cout << "time: “ << duration << std::endl;
```

C

```
#include <time.h>

static double rtclock()
{
  struct timeval Tp;
  gettimeofday (&Tp, NULL);
  return (Tp.tv_sec + Tp.tv_usec * 1.0e-6);
}
```

perf

```
# Record the execution time of functions
$ perf record ./test

# Print the recording report
$ perf report

# Print hardware counters
$ perf stat ./test

# perf stat [-e <EVENT>] <command>
$  perf stat -e cache-misses ./test

# Show all events
$ perf list
```

Tiling

Original

```
for (i=0; i<= N-1; i++)
 for (j=0; j<= N-1; j++)
  C(i, j) = 0
  for (k=0; k<N; k++)
   C(i, j) += A(i, k) * B(k, j)
```

Tiled

```
for (i0=0; i0 < N/32; i0++)
   for (j0=0; j0 < N/32; j0++)
      for (i1=32*i0; i1<32*i0+32; i1++)
         for (j1=32*j0; j1<32*j0+32; j1++)
        C(i1, j1) = 0
            for k in 0 … N
                C(i1, j1) += A(i1, k) * B(k, j1)
```