DSDL winter seminar

Week 2

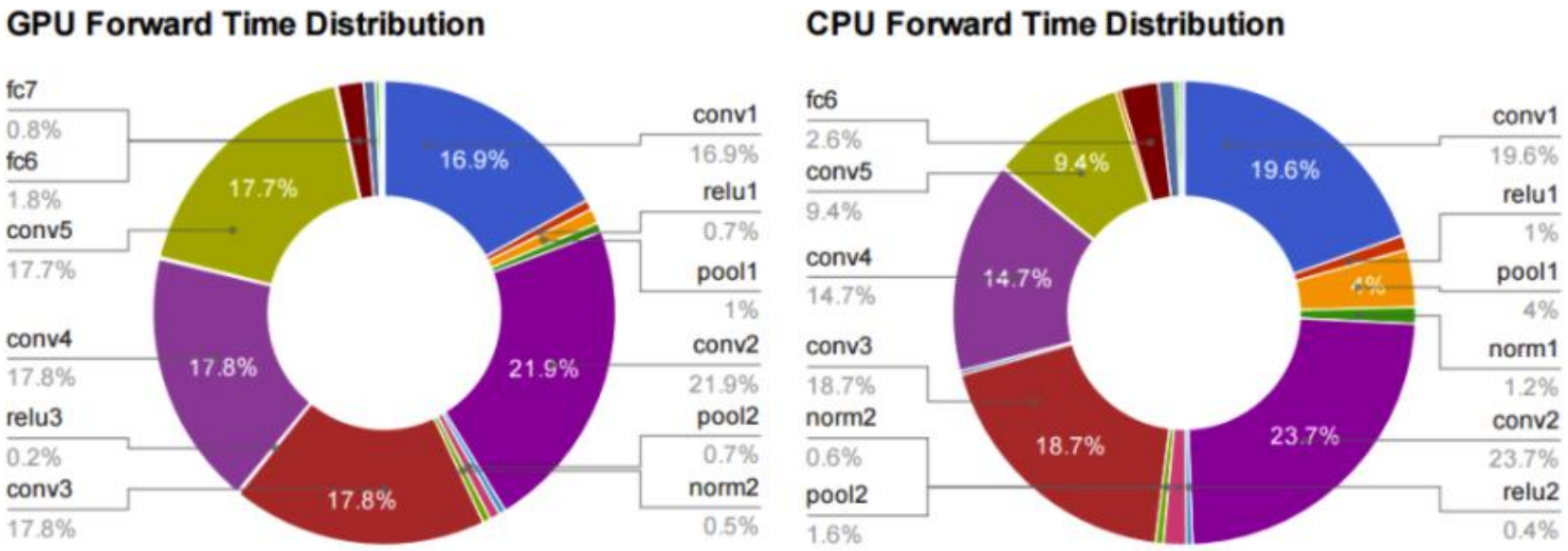
안권환

분석 코드 list

- gemm.h
- gemm.c
- matrix.c
- matrix.h
- layer.h
- layer.c
- network.c
- network.h
- sg_dsdl.h

gemm.c – gemm_bin

- All of the layers that start with fc (for fully-connected) or conv (for convolution) are implemented using GEMM, and almost all the time (95% of the GPU version, and 89% on CPU) is spent on those layers.

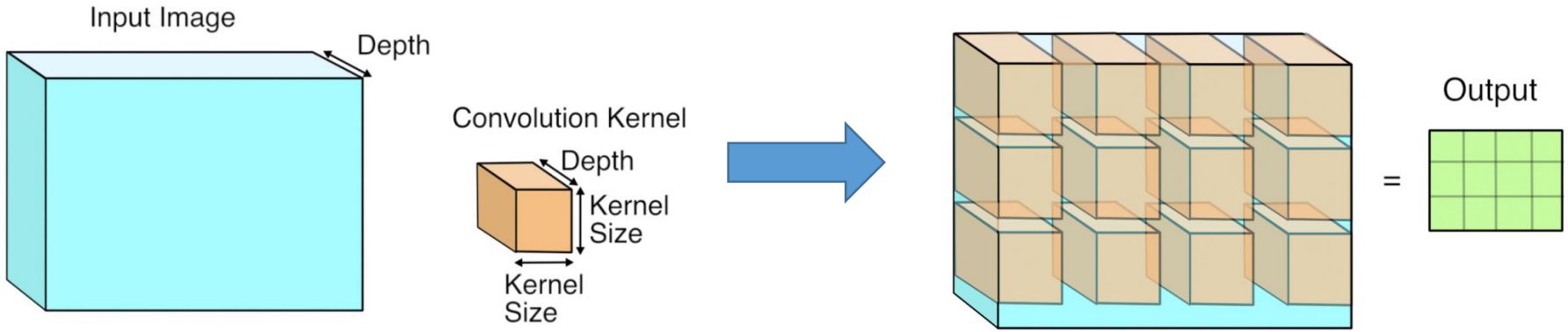


Jia, Y. (2014). *Learning semantic image representations at a large scale*. University of California, Berkeley.

gemm.c – gemm_bin

- What is GEMM? General Matrix to Matrix Multiplication

일반적인 개념

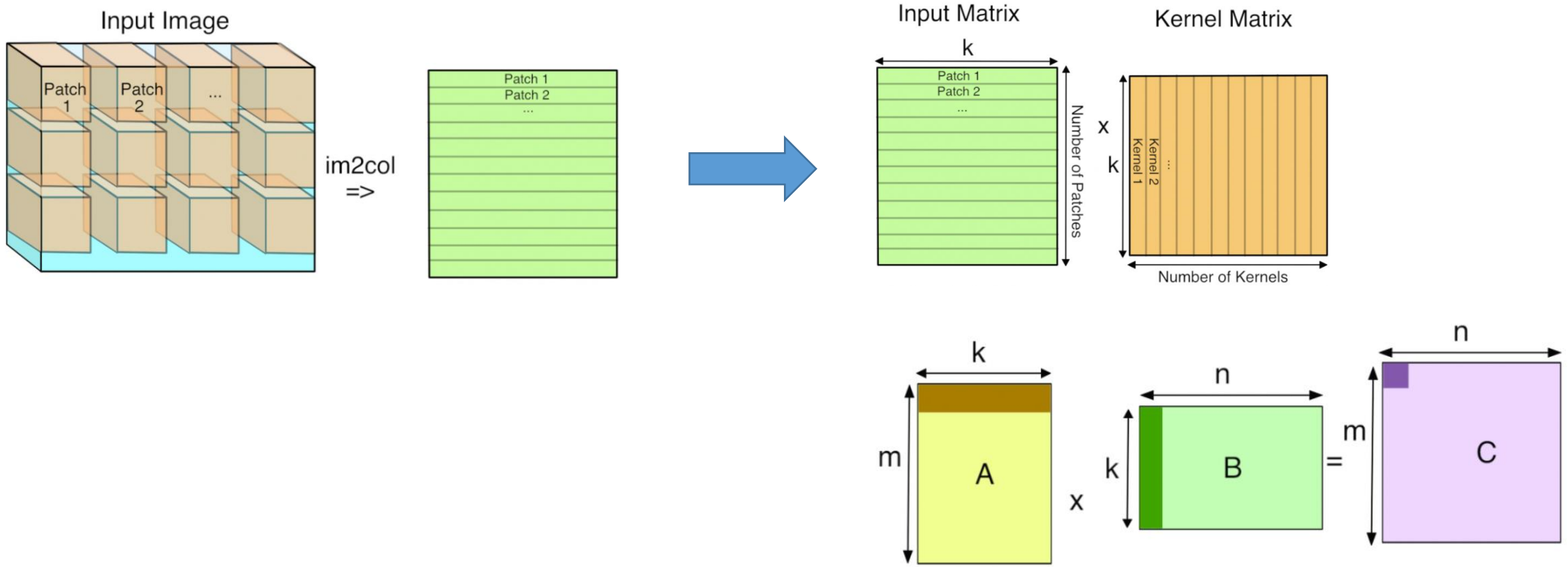


- Why uses GEMM? Need billlion FLOPs to calculate a 3D single frame(image)

Jia, Y. (2014). *Learning semantic image representations at a large scale*. University of California, Berkeley.

gemm.c – gemm_bin

- So? Im2col을 활용하여 matrix 형태로 바꿔서 연산



Jia, Y. (2014). *Learning semantic image representations at a large scale*. University of California, Berkeley.

matrix.c – free_matrix

- free(m.vals[i])를 하고, 추가적으로 free(m.vals)를 하는 이유?

```
// free matrix
void free_matrix(matrix m)
{
    int i;
    // vals : variable-length array
    for(i = 0; i < m.rows; ++i) free(m.vals[i]);
    // ??????????
    free(m.vals);
}
```

network.c

- network.c의 역할을 알기 위해 코드의 모든 variable을 알아야함
- 코드의 모든 variable들을 알기 위해, 다른 src 파일에 대해 더 파악할 필요가 있음

network.c – reset_network_state

- Need to insert line??

```
// need to insert line???
void reset_network_state(network *net, int b)
{
    int i;
    for (i = 0; i < net->n; ++i) {
//          #ifdef GPU
//          layer l = net->layers[i];
//          if(l.state_gpu){
//              fill_gpu(l.outputs, 0, l.state_gpu + l.outputs*b, 1);
//          }
//          if(l.h_gpu){
//              fill_gpu(l.outputs, 0, l.h_gpu + l.outputs*b, 1);
//          }
//          #endif
    }
}
```

network.c – forward_network

- How does this work?

```
// ???????
void forward_network(network *netp)
{
//#ifdef GPU
//    if(netp->gpu_index >= 0){
//        forward_network_gpu(netp);
//        return;
//    }
//#endif
    network net = *netp;
    int i;
    for(i = 0; i < net.n; ++i){
        net.index = i;
        layer l = net.layers[i];
        if(l.delta){
            fill_cpu(l.outputs * l.batch, 0, l.delta, 1);
        }
        l.forward(l, net);
        net.input = l.output;
        if(l.truth) {
            net.truth = l.output;
        }
    }
    calc_network_cost(netp);
}
```