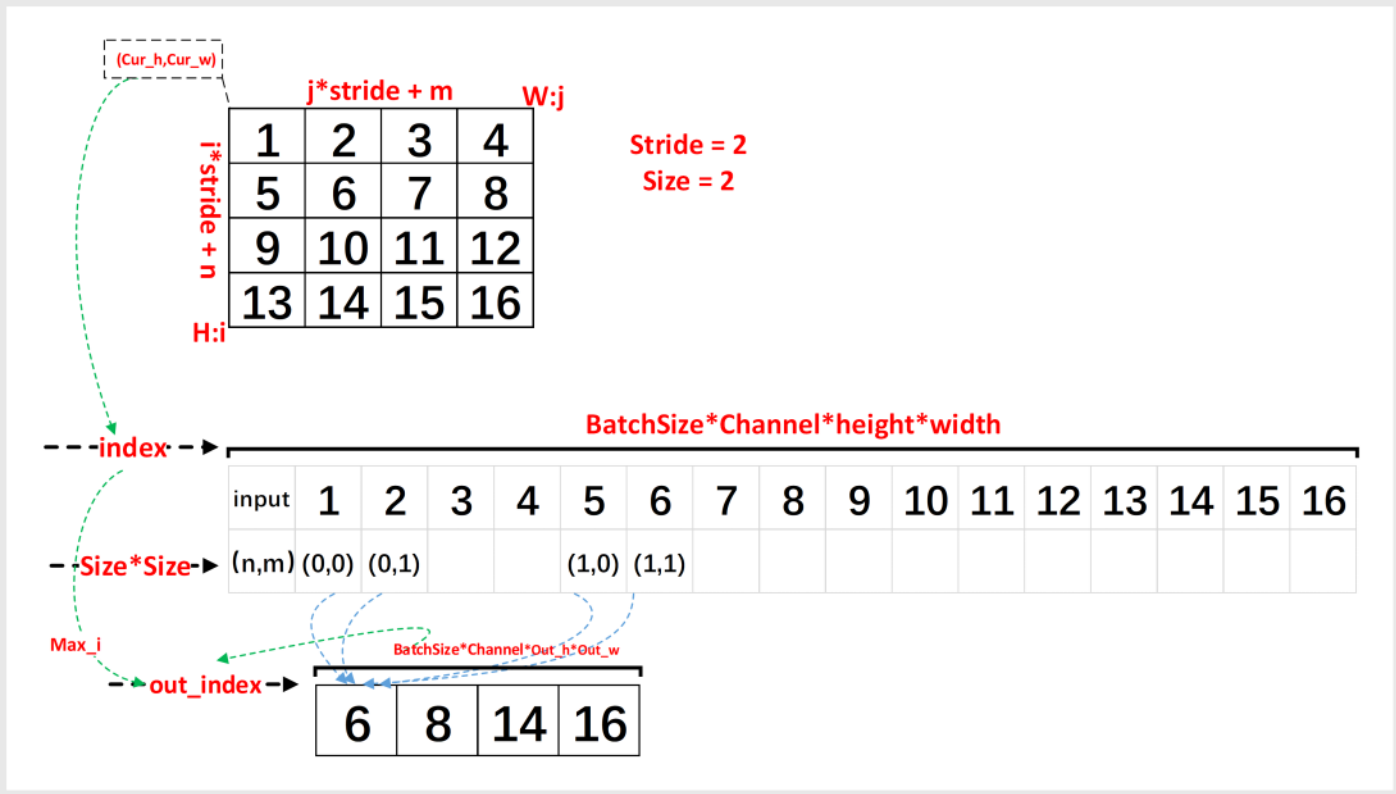
Maxpool_layer

2018年11月29日 16:23




```
maxpool_layer make_maxpool_layer(int batch, int h, int w, int c, int size, int stride, int padding)
```

batch-batchsize

h-height

w-width

size-kernelSize

Stride-stride

Padding-padding

//计算池化操作后的输出尺寸

//w_out=(w_in + 2*pad - kernel_size) / stride +1;

///h_out =(h_in + 2*pad - kernel_size) / stride +1;

```
l.out_w = (w + 2*padding)/stride;
l.out_h = (h + 2*padding)/stride;
```

```
void forward_maxpool_layer(const maxpool_layer l, network net)
{
    int b,i,j,k,m,n;
    int w_offset = -l.pad/2;
    int h_offset = -l.pad/2;
```

```
    int h = l.out_h;
    int w = l.out_w;
    int c = l.c;
    //遍历输出元素
    for(b = 0; b < l.batch; ++b){
        for(k = 0; k < c; ++k){
            for(i = 0; i < h; ++i){
                for(j = 0; j < w; ++j){
                    int out_index = j + w*(i + h*(k + c*b));//计算输出索引
                    float max = -FLT_MAX;
                    int max_i = -1;
                     //依次取出kernelSize*kernelSize个元素
                    for(n = 0; n < l.size; ++n){
                        for(m = 0; m < l.size; ++m){
                                //依次计算当前kernel内部元素的相对坐标
                            int cur_h = h_offset + i*l.stride + n;
                            int cur_w = w_offset + j*l.stride + m;
                              //根据相对坐标计算出一维数组的真实坐标
                            int index = cur_w + l.w*(cur_h + l.h*(k + b*l.c));
                              //边界检查
                            int valid = (cur_h >= 0 && cur_h < l.h &&
                                         cur_w >= 0 && cur_w < l.w);
                              //取出元素
                            float val = (valid != 0) ? net.input[index] : -FLT_MAX;
                              //确定最大值元素及其角标
                            max_i = (val > max) ? index : max_i;
                            max   = (val > max) ? val   : max;
                        }
                    }
                      //输出kernel内最大元素及其角标
                    l.output[out_index] = max;
                    l.indexes[out_index] = max_i;
                }
            }
        }
    }
}

void backward_maxpool_layer(const maxpool_layer l, network net)
{
    int i;
    int h = l.out_h;
    int w = l.out_w;
    int c = l.c;
```

```
    for(i = 0; i < h*w*c*l.batch; ++i){
        //梯度只传给上一层最大值位置的神经元，其余神经元的梯度都是0
        int index = l.indexes[i];
        net.delta[index] += l.delta[i];
    }
}
```