Cuda编程

① 目录结构

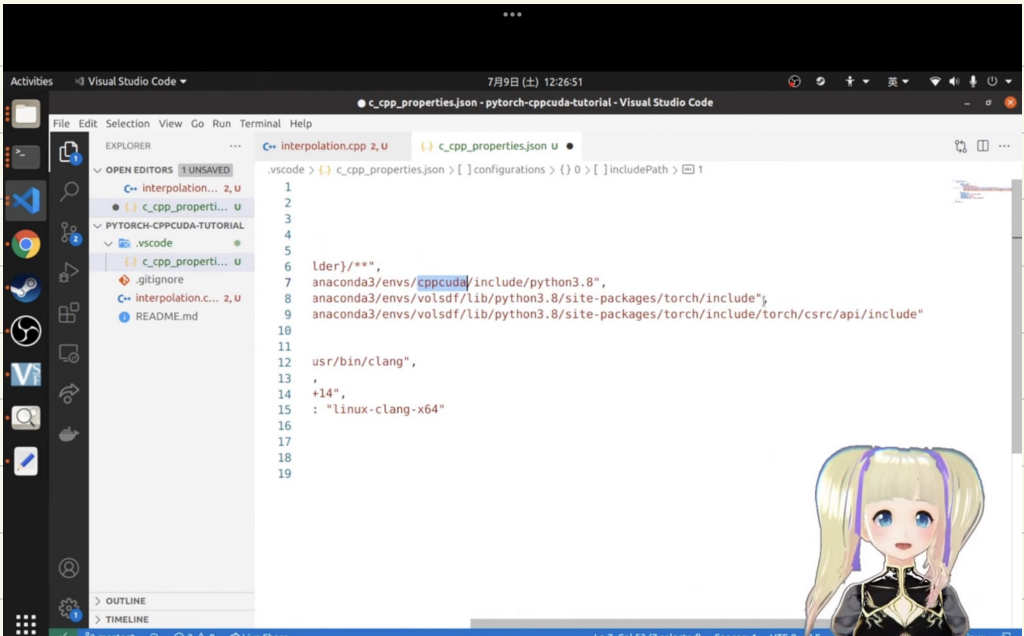
- include
-- utils.h
- interpolation_kernel.cu
- interpolation.cpp
- setup.py
- test.py

② vscode环境设置

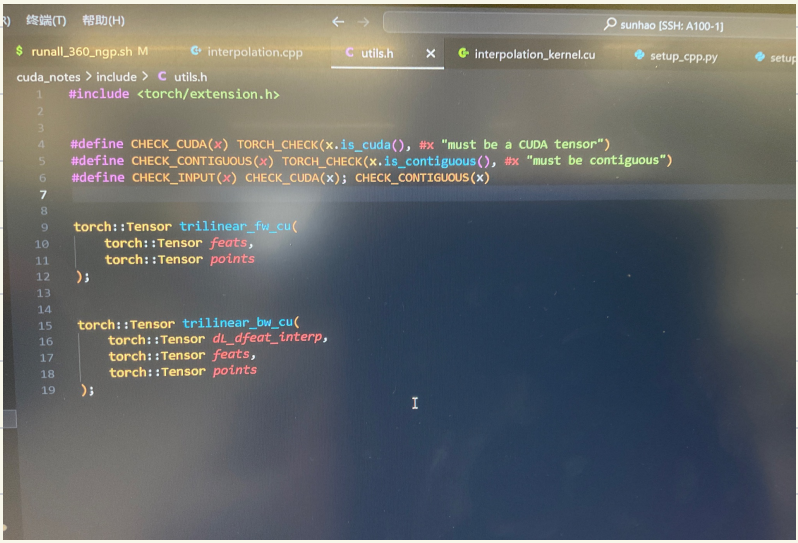
i) ctrl+shift+p

ii) 输入 edit configuration，拉出json配置文件

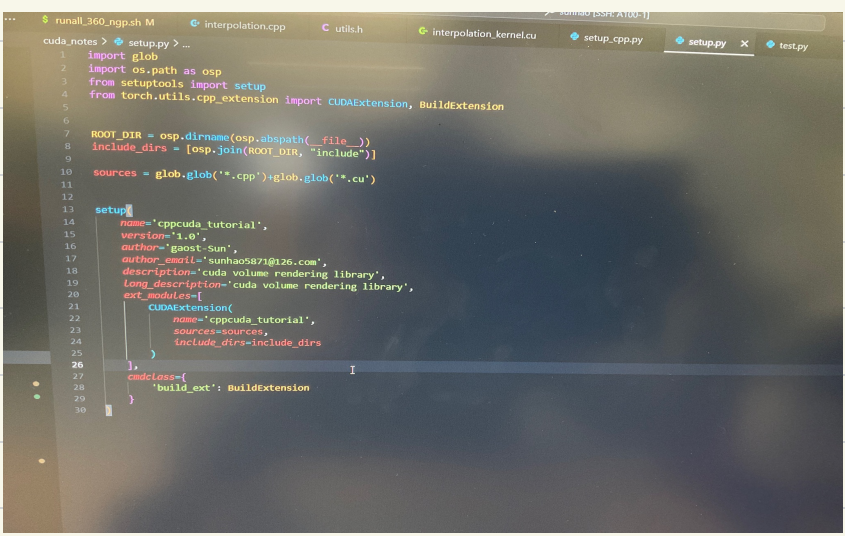
iii) 在 "includePath" 添加环境路径.



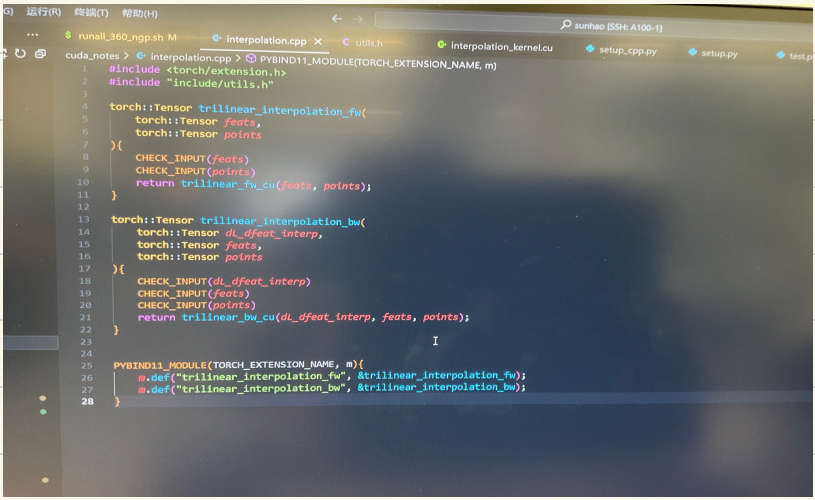
③ include/utils.h



④ setup.py

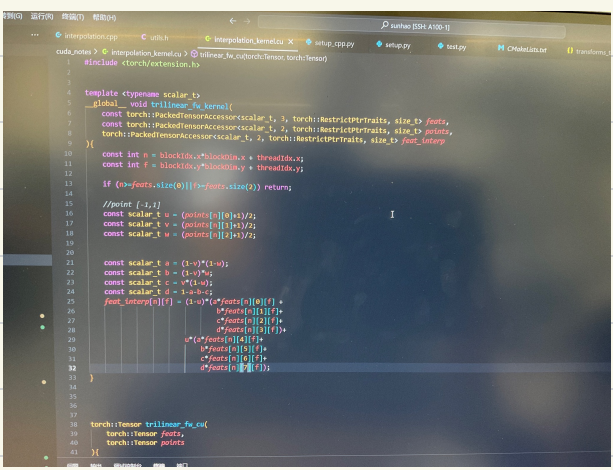


⑤ interpolation.py



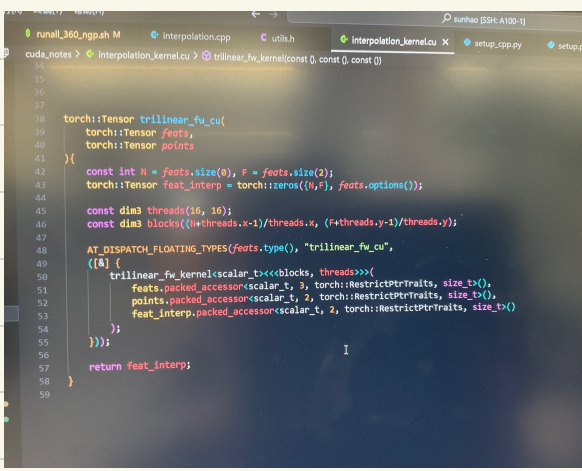
```cpp
#include <torch/extension.h>
#include "include/utils.h"

torch::Tensor trilinear_interpolation_fw(
    torch::Tensor feats,
    torch::Tensor points
){
    CHECK_INPUT(feats)
    CHECK_INPUT(points)
    return trilinear_fw_cu(feats, points);
}

torch::Tensor trilinear_interpolation_bw(
    torch::Tensor dL_dfeat_interp,
    torch::Tensor feats,
    torch::Tensor points
){
    CHECK_INPUT(dL_dfeat_interp)
    CHECK_INPUT(feats)
    CHECK_INPUT(points)
    return trilinear_bw_cu(dL_dfeat_interp, feats, points);
}

PYBIND11_MODULE(TORCH_EXTENSION_NAME, m){
    m.def("trilinear_interpolation_fw", &trilinear_interpolation_fw);
    m.def("trilinear_interpolation_bw", &trilinear_interpolation_bw);
}
```

⑥ interpolation_kernel.cu



```cpp
template <typename scalar_t>
__global__ void trilinear_fw_kernel(
    const torch::PackedTensorAccessor<scalar_t, 3, torch::RestrictPtrTraits, size_t> feats,
    const torch::PackedTensorAccessor<scalar_t, 2, torch::RestrictPtrTraits, size_t> points,
    torch::PackedTensorAccessor<scalar_t, 2, torch::RestrictPtrTraits, size_t> feat_interp
){
    const int n = blockIdx.x*blockDim.x + threadIdx.x;
    const int f = blockIdx.y*blockDim.y + threadIdx.y;

    if (n>=feats.size(0)||f>=feats.size(2)) return;

    //point [-1,1]
    const scalar_t u = (points[n][0]+1)/2;
    const scalar_t v = (points[n][1]+1)/2;
    const scalar_t w = (points[n][2]+1)/2;


    const scalar_t a = (1-v)*(1-w);
    const scalar_t b = (1-v)*w;
    const scalar_t c = v*(1-w);
    const scalar_t d = 1-a-b-c;
    feat_interp[n][f] = (1-u)*(a*feats[n][0][f] +
                               b*feats[n][1][f]+
                               c*feats[n][2][f]+
                               d*feats[n][3][f])+
                        u*(a*feats[n][4][f]+
                           b*feats[n][5][f]+
                           c*feats[n][6][f]+
                           d*feats[n][7][f]);
}


torch::Tensor trilinear_fw_cu(
    torch::Tensor feats,
    torch::Tensor points
){
```

trilinear_fw_kernel

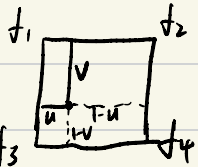


```cpp
torch::Tensor trilinear_fw_cu(
    torch::Tensor feats,
    torch::Tensor points
){
    const int N = feats.size(0), F = feats.size(2);
    torch::Tensor feat_interp = torch::zeros({N,F}, feats.options());

    const dim3 threads(16, 16);
    const dim3 blocks((N+threads.x-1)/threads.x, (F+threads.y-1)/threads.y);

    AT_DISPATCH_FLOATING_TYPES(feats.type(), "trilinear_fw_cu",
    ([&] {
        trilinear_fw_kernel<scalar_t><<<blocks, threads>>>(
            feats.packed_accessor<scalar_t, 3, torch::RestrictPtrTraits, size_t>(),
            points.packed_accessor<scalar_t, 2, torch::RestrictPtrTraits, size_t>(),
            feat_interp.packed_accessor<scalar_t, 2, torch::RestrictPtrTraits, size_t>()
        );
    }));

    return feat_interp;
}
```

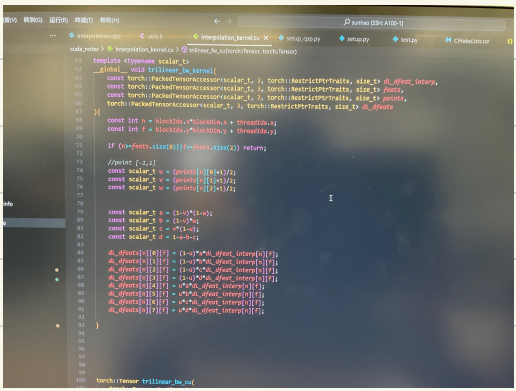
trilinear_fw_cu

- Loss 计算(梯度反传):

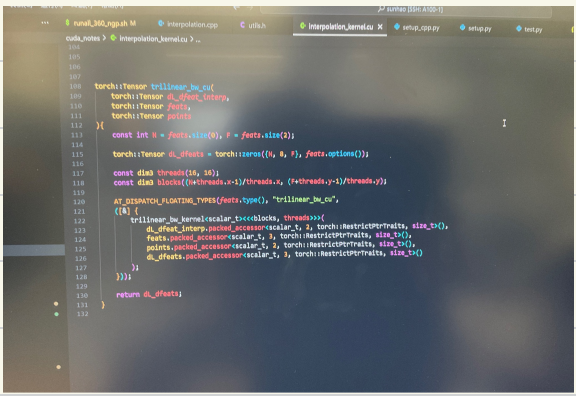
二次插值函数: $f = F(f_1, f_2, f_3, f_4) = uvf_4 + (1-u)vf_3 + u(1-v)f_2 + (1-u)(1-v)f_1$



$\frac{\partial f}{\partial f_1} = (1-u)(1-v)$ $\frac{\partial f}{\partial f_2} = u(1-v)$ $\frac{\partial f}{\partial f_3} = (1-u)v$ $\frac{\partial f}{\partial f_4} = uv$



trilinear_bw_kernel



trilinear_bw_cu

⑦ 测试结果：