#include "layer.h"

#include "dark\_cuda.h"

#include <stdlib.h>

void free\_sublayer(layer \*l)

{

if (l) {

free\_layer(\*l);

free(l);

}

}

void free\_layer(layer l)

{

free\_layer\_custom(l, 0);

}

void free\_layer\_custom(layer l, int keep\_cudnn\_desc)

{

if (l.share\_layer != NULL) return; // don't free shared layers

if (l.antialiasing) {

free\_sublayer(l.input\_layer);

}

if (l.type == CONV\_LSTM) {

if (l.peephole) {

free\_sublayer(l.vf);

free\_sublayer(l.vi);

free\_sublayer(l.vo);

}

else {

free(l.vf);

free(l.vi);

free(l.vo);

}

free\_sublayer(l.wf);

free\_sublayer(l.wi);

free\_sublayer(l.wg);

free\_sublayer(l.wo);

free\_sublayer(l.uf);

free\_sublayer(l.ui);

free\_sublayer(l.ug);

free\_sublayer(l.uo);

}

if (l.type == CRNN) {

free\_sublayer(l.input\_layer);

free\_sublayer(l.self\_layer);

free\_sublayer(l.output\_layer);

l.output = NULL;

l.delta = NULL;

#ifdef GPU

l.output\_gpu = NULL;

l.delta\_gpu = NULL;

#endif // GPU

}

if (l.type == DROPOUT) {

if (l.rand) free(l.rand);

#ifdef GPU

if (l.rand\_gpu) cuda\_free(l.rand\_gpu);

if (l.drop\_blocks\_scale) cuda\_free\_host(l.drop\_blocks\_scale);

if (l.drop\_blocks\_scale\_gpu) cuda\_free(l.drop\_blocks\_scale\_gpu);

#endif

return;

}

if (l.mask) free(l.mask);

if (l.classes\_multipliers)free(l.classes\_multipliers);

if (l.cweights) free(l.cweights);

if (l.indexes) free(l.indexes);

if (l.input\_layers) free(l.input\_layers);

if (l.input\_sizes) free(l.input\_sizes);

if (l.layers\_output) free(l.layers\_output);

if (l.layers\_delta) free(l.layers\_delta);

if (l.map) free(l.map);

if (l.rand) free(l.rand);

if (l.cost) free(l.cost);

if (l.state) free(l.state);

if (l.prev\_state) free(l.prev\_state);

if (l.forgot\_state) free(l.forgot\_state);

if (l.forgot\_delta) free(l.forgot\_delta);

if (l.state\_delta) free(l.state\_delta);

if (l.concat) free(l.concat);

if (l.concat\_delta) free(l.concat\_delta);

if (l.binary\_weights) free(l.binary\_weights);

if (l.biases) free(l.biases), l.biases = NULL;

if (l.bias\_updates) free(l.bias\_updates), l.bias\_updates = NULL;

if (l.scales) free(l.scales), l.scales = NULL;

if (l.scale\_updates) free(l.scale\_updates), l.scale\_updates = NULL;

if (l.weights) free(l.weights), l.weights = NULL;

if (l.weight\_updates) free(l.weight\_updates), l.weight\_updates = NULL;

if (l.align\_bit\_weights) free(l.align\_bit\_weights);

if (l.mean\_arr) free(l.mean\_arr);

#ifdef GPU

if (l.delta && l.delta\_pinned) {

cudaFreeHost(l.delta);

l.delta = NULL;

}

if (l.output && l.output\_pinned) {

cudaFreeHost(l.output);

l.output = NULL;

}

#endif // GPU

if (l.delta) free(l.delta), l.delta = NULL;

if (l.output) free(l.output), l.output = NULL;

if (l.activation\_input) free(l.activation\_input), l.activation\_input = NULL;

if (l.squared) free(l.squared);

if (l.norms) free(l.norms);

if (l.spatial\_mean) free(l.spatial\_mean);

if (l.mean) free(l.mean), l.mean = NULL;

if (l.variance) free(l.variance), l.variance = NULL;

if (l.mean\_delta) free(l.mean\_delta), l.mean\_delta = NULL;

if (l.variance\_delta) free(l.variance\_delta), l.variance\_delta = NULL;

if (l.rolling\_mean) free(l.rolling\_mean), l.rolling\_mean = NULL;

if (l.rolling\_variance) free(l.rolling\_variance), l.rolling\_variance = NULL;

if (l.x) free(l.x);

if (l.x\_norm) free(l.x\_norm);

if (l.m) free(l.m);

if (l.v) free(l.v);

if (l.z\_cpu) free(l.z\_cpu);

if (l.r\_cpu) free(l.r\_cpu);

if (l.binary\_input) free(l.binary\_input);

if (l.bin\_re\_packed\_input) free(l.bin\_re\_packed\_input);

if (l.t\_bit\_input) free(l.t\_bit\_input);

if (l.loss) free(l.loss);

// CONV-LSTM

if (l.f\_cpu) free(l.f\_cpu);

if (l.i\_cpu) free(l.i\_cpu);

if (l.g\_cpu) free(l.g\_cpu);

if (l.o\_cpu) free(l.o\_cpu);

if (l.c\_cpu) free(l.c\_cpu);

if (l.h\_cpu) free(l.h\_cpu);

if (l.temp\_cpu) free(l.temp\_cpu);

if (l.temp2\_cpu) free(l.temp2\_cpu);

if (l.temp3\_cpu) free(l.temp3\_cpu);

if (l.dc\_cpu) free(l.dc\_cpu);

if (l.dh\_cpu) free(l.dh\_cpu);

if (l.prev\_state\_cpu) free(l.prev\_state\_cpu);

if (l.prev\_cell\_cpu) free(l.prev\_cell\_cpu);

if (l.stored\_c\_cpu) free(l.stored\_c\_cpu);

if (l.stored\_h\_cpu) free(l.stored\_h\_cpu);

if (l.cell\_cpu) free(l.cell\_cpu);

#ifdef GPU

if (l.indexes\_gpu) cuda\_free((float \*)l.indexes\_gpu);

if (l.z\_gpu) cuda\_free(l.z\_gpu);

if (l.r\_gpu) cuda\_free(l.r\_gpu);

if (l.m\_gpu) cuda\_free(l.m\_gpu);

if (l.v\_gpu) cuda\_free(l.v\_gpu);

if (l.forgot\_state\_gpu) cuda\_free(l.forgot\_state\_gpu);

if (l.forgot\_delta\_gpu) cuda\_free(l.forgot\_delta\_gpu);

if (l.state\_gpu) cuda\_free(l.state\_gpu);

if (l.state\_delta\_gpu) cuda\_free(l.state\_delta\_gpu);

if (l.gate\_gpu) cuda\_free(l.gate\_gpu);

if (l.gate\_delta\_gpu) cuda\_free(l.gate\_delta\_gpu);

if (l.save\_gpu) cuda\_free(l.save\_gpu);

if (l.save\_delta\_gpu) cuda\_free(l.save\_delta\_gpu);

if (l.concat\_gpu) cuda\_free(l.concat\_gpu);

if (l.concat\_delta\_gpu) cuda\_free(l.concat\_delta\_gpu);

if (l.binary\_input\_gpu) cuda\_free(l.binary\_input\_gpu);

if (l.binary\_weights\_gpu) cuda\_free(l.binary\_weights\_gpu);

if (l.mean\_gpu) cuda\_free(l.mean\_gpu), l.mean\_gpu = NULL;

if (l.variance\_gpu) cuda\_free(l.variance\_gpu), l.variance\_gpu = NULL;

if (l.m\_cbn\_avg\_gpu) cuda\_free(l.m\_cbn\_avg\_gpu), l.m\_cbn\_avg\_gpu = NULL;

if (l.v\_cbn\_avg\_gpu) cuda\_free(l.v\_cbn\_avg\_gpu), l.v\_cbn\_avg\_gpu = NULL;

if (l.rolling\_mean\_gpu) cuda\_free(l.rolling\_mean\_gpu), l.rolling\_mean\_gpu = NULL;

if (l.rolling\_variance\_gpu) cuda\_free(l.rolling\_variance\_gpu), l.rolling\_variance\_gpu = NULL;

if (l.variance\_delta\_gpu) cuda\_free(l.variance\_delta\_gpu), l.variance\_delta\_gpu = NULL;

if (l.mean\_delta\_gpu) cuda\_free(l.mean\_delta\_gpu), l.mean\_delta\_gpu = NULL;

if (l.x\_norm\_gpu) cuda\_free(l.x\_norm\_gpu);

// assisted excitation

if (l.gt\_gpu) cuda\_free(l.gt\_gpu);

if (l.a\_avg\_gpu) cuda\_free(l.a\_avg\_gpu);

if (l.align\_bit\_weights\_gpu) cuda\_free((float \*)l.align\_bit\_weights\_gpu);

if (l.mean\_arr\_gpu) cuda\_free(l.mean\_arr\_gpu);

if (l.align\_workspace\_gpu) cuda\_free(l.align\_workspace\_gpu);

if (l.transposed\_align\_workspace\_gpu) cuda\_free(l.transposed\_align\_workspace\_gpu);

if (l.weights\_gpu) cuda\_free(l.weights\_gpu), l.weights\_gpu = NULL;

if (l.weight\_updates\_gpu) cuda\_free(l.weight\_updates\_gpu), l.weight\_updates\_gpu = NULL;

if (l.weight\_deform\_gpu) cuda\_free(l.weight\_deform\_gpu), l.weight\_deform\_gpu = NULL;

if (l.weights\_gpu16) cuda\_free(l.weights\_gpu16), l.weights\_gpu16 = NULL;

if (l.weight\_updates\_gpu16) cuda\_free(l.weight\_updates\_gpu16), l.weight\_updates\_gpu16 = NULL;

if (l.biases\_gpu) cuda\_free(l.biases\_gpu), l.biases\_gpu = NULL;

if (l.bias\_updates\_gpu) cuda\_free(l.bias\_updates\_gpu), l.bias\_updates\_gpu = NULL;

if (l.scales\_gpu) cuda\_free(l.scales\_gpu), l.scales\_gpu = NULL;

if (l.scale\_updates\_gpu) cuda\_free(l.scale\_updates\_gpu), l.scale\_updates\_gpu = NULL;

if (l.input\_antialiasing\_gpu) cuda\_free(l.input\_antialiasing\_gpu), l.input\_antialiasing\_gpu = NULL;

if (l.optimized\_memory < 2) {

if (l.x\_gpu) cuda\_free(l.x\_gpu); l.x\_gpu = NULL;

if (l.output\_gpu) cuda\_free(l.output\_gpu), l.output\_gpu = NULL;

if (l.activation\_input\_gpu) cuda\_free(l.activation\_input\_gpu), l.activation\_input\_gpu = NULL;

}

if (l.delta\_gpu && l.keep\_delta\_gpu && l.optimized\_memory < 3) cuda\_free(l.delta\_gpu), l.delta\_gpu = NULL;

if (l.rand\_gpu) cuda\_free(l.rand\_gpu);

if (l.squared\_gpu) cuda\_free(l.squared\_gpu);

if (l.norms\_gpu) cuda\_free(l.norms\_gpu);

if (l.input\_sizes\_gpu) cuda\_free((float\*)l.input\_sizes\_gpu);

if (l.layers\_output\_gpu) cuda\_free((float\*)l.layers\_output\_gpu);

if (l.layers\_delta\_gpu) cuda\_free((float\*)l.layers\_delta\_gpu);

// CONV-LSTM

if (l.f\_gpu) cuda\_free(l.f\_gpu);

if (l.i\_gpu) cuda\_free(l.i\_gpu);

if (l.g\_gpu) cuda\_free(l.g\_gpu);

if (l.o\_gpu) cuda\_free(l.o\_gpu);

if (l.c\_gpu) cuda\_free(l.c\_gpu);

if (l.h\_gpu) cuda\_free(l.h\_gpu);

if (l.temp\_gpu) cuda\_free(l.temp\_gpu);

if (l.temp2\_gpu) cuda\_free(l.temp2\_gpu);

if (l.temp3\_gpu) cuda\_free(l.temp3\_gpu);

if (l.dc\_gpu) cuda\_free(l.dc\_gpu);

if (l.dh\_gpu) cuda\_free(l.dh\_gpu);

if (l.prev\_state\_gpu) cuda\_free(l.prev\_state\_gpu);

if (l.prev\_cell\_gpu) cuda\_free(l.prev\_cell\_gpu);

if (l.stored\_c\_gpu) cuda\_free(l.stored\_c\_gpu);

if (l.stored\_h\_gpu) cuda\_free(l.stored\_h\_gpu);

if (l.last\_prev\_state\_gpu) cuda\_free(l.last\_prev\_state\_gpu);

if (l.last\_prev\_cell\_gpu) cuda\_free(l.last\_prev\_cell\_gpu);

if (l.cell\_gpu) cuda\_free(l.cell\_gpu);

#ifdef CUDNN // shouldn't be used for -map

if (!keep\_cudnn\_desc) {

if (l.srcTensorDesc) CHECK\_CUDNN(cudnnDestroyTensorDescriptor(l.srcTensorDesc));

if (l.dstTensorDesc) CHECK\_CUDNN(cudnnDestroyTensorDescriptor(l.dstTensorDesc));

if (l.srcTensorDesc16) CHECK\_CUDNN(cudnnDestroyTensorDescriptor(l.srcTensorDesc16));

if (l.dstTensorDesc16) CHECK\_CUDNN(cudnnDestroyTensorDescriptor(l.dstTensorDesc16));

if (l.dsrcTensorDesc) CHECK\_CUDNN(cudnnDestroyTensorDescriptor(l.dsrcTensorDesc));

if (l.ddstTensorDesc) CHECK\_CUDNN(cudnnDestroyTensorDescriptor(l.ddstTensorDesc));

if (l.dsrcTensorDesc16) CHECK\_CUDNN(cudnnDestroyTensorDescriptor(l.dsrcTensorDesc16));

if (l.ddstTensorDesc16) CHECK\_CUDNN(cudnnDestroyTensorDescriptor(l.ddstTensorDesc16));

if (l.normTensorDesc) CHECK\_CUDNN(cudnnDestroyTensorDescriptor(l.normTensorDesc));

if (l.normDstTensorDesc) CHECK\_CUDNN(cudnnDestroyTensorDescriptor(l.normDstTensorDesc));

if (l.normDstTensorDescF16) CHECK\_CUDNN(cudnnDestroyTensorDescriptor(l.normDstTensorDescF16));

if (l.weightDesc) CHECK\_CUDNN(cudnnDestroyFilterDescriptor(l.weightDesc));

if (l.weightDesc16) CHECK\_CUDNN(cudnnDestroyFilterDescriptor(l.weightDesc16));

if (l.dweightDesc) CHECK\_CUDNN(cudnnDestroyFilterDescriptor(l.dweightDesc));

if (l.dweightDesc16) CHECK\_CUDNN(cudnnDestroyFilterDescriptor(l.dweightDesc16));

if (l.convDesc) CHECK\_CUDNN(cudnnDestroyConvolutionDescriptor(l.convDesc));

if (l.poolingDesc) CHECK\_CUDNN(cudnnDestroyPoolingDescriptor(l.poolingDesc));

//cudnnConvolutionFwdAlgo\_t fw\_algo, fw\_algo16;

//cudnnConvolutionBwdDataAlgo\_t bd\_algo, bd\_algo16;

//cudnnConvolutionBwdFilterAlgo\_t bf\_algo, bf\_algo16;

}

#endif // CUDNN

#endif // GPU

}