#include "activation\_layer.h"

#include "utils.h"

#include "dark\_cuda.h"

#include "blas.h"

#include "gemm.h"

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

layer make\_activation\_layer(int batch, int inputs, ACTIVATION activation)

{

layer l = { (LAYER\_TYPE)0 };

l.type = ACTIVE;

l.inputs = inputs;

l.outputs = inputs;

l.batch=batch;

l.output = (float\*)xcalloc(batch \* inputs, sizeof(float));

l.delta = (float\*)xcalloc(batch \* inputs, sizeof(float));

l.forward = forward\_activation\_layer;

l.backward = backward\_activation\_layer;

#ifdef GPU

l.forward\_gpu = forward\_activation\_layer\_gpu;

l.backward\_gpu = backward\_activation\_layer\_gpu;

l.output\_gpu = cuda\_make\_array(l.output, inputs\*batch);

l.delta\_gpu = cuda\_make\_array(l.delta, inputs\*batch);

#endif

l.activation = activation;

fprintf(stderr, "Activation Layer: %d inputs\n", inputs);

return l;

}

void forward\_activation\_layer(layer l, network\_state state)

{

copy\_cpu(l.outputs\*l.batch, state.input, 1, l.output, 1);

activate\_array(l.output, l.outputs\*l.batch, l.activation);

}

void backward\_activation\_layer(layer l, network\_state state)

{

gradient\_array(l.output, l.outputs\*l.batch, l.activation, l.delta);

copy\_cpu(l.outputs\*l.batch, l.delta, 1, state.delta, 1);

}

#ifdef GPU

void forward\_activation\_layer\_gpu(layer l, network\_state state)

{

copy\_ongpu(l.outputs\*l.batch, state.input, 1, l.output\_gpu, 1);

activate\_array\_ongpu(l.output\_gpu, l.outputs\*l.batch, l.activation);

}

void backward\_activation\_layer\_gpu(layer l, network\_state state)

{

gradient\_array\_ongpu(l.output\_gpu, l.outputs\*l.batch, l.activation, l.delta\_gpu);

copy\_ongpu(l.outputs\*l.batch, l.delta\_gpu, 1, state.delta, 1);

}

#endif