#ifndef DECONVOLUTIONAL\_LAYER\_H

#define DECONVOLUTIONAL\_LAYER\_H

#include "dark\_cuda.h"

#include "image.h"

#include "activations.h"

#include "layer.h"

#include "network.h"

typedef layer deconvolutional\_layer;

#ifdef \_\_cplusplus

extern "C" {

#endif

#ifdef GPU

void forward\_deconvolutional\_layer\_gpu(deconvolutional\_layer layer, network\_state state);

void backward\_deconvolutional\_layer\_gpu(deconvolutional\_layer layer, network\_state state);

void update\_deconvolutional\_layer\_gpu(deconvolutional\_layer layer, int skip, float learning\_rate, float momentum, float decay);

void push\_deconvolutional\_layer(deconvolutional\_layer layer);

void pull\_deconvolutional\_layer(deconvolutional\_layer layer);

#endif

deconvolutional\_layer make\_deconvolutional\_layer(int batch, int h, int w, int c, int n, int size, int stride, ACTIVATION activation);

void resize\_deconvolutional\_layer(deconvolutional\_layer \*layer, int h, int w);

void forward\_deconvolutional\_layer(const deconvolutional\_layer layer, network\_state state);

void update\_deconvolutional\_layer(deconvolutional\_layer layer, int skip, float learning\_rate, float momentum, float decay);

void backward\_deconvolutional\_layer(deconvolutional\_layer layer, network\_state state);

image get\_deconvolutional\_image(deconvolutional\_layer layer);

image get\_deconvolutional\_delta(deconvolutional\_layer layer);

image get\_deconvolutional\_filter(deconvolutional\_layer layer, int i);

int deconvolutional\_out\_height(deconvolutional\_layer layer);

int deconvolutional\_out\_width(deconvolutional\_layer layer);

#ifdef \_\_cplusplus

}

#endif

#endif