#include "utils.h"

#include "crop\_layer.h"

#include "dark\_cuda.h"

#include <stdio.h>

image get\_crop\_image(crop\_layer l)

{

int h = l.out\_h;

int w = l.out\_w;

int c = l.out\_c;

return float\_to\_image(w,h,c,l.output);

}

void backward\_crop\_layer(const crop\_layer l, network\_state state){}

void backward\_crop\_layer\_gpu(const crop\_layer l, network\_state state){}

crop\_layer make\_crop\_layer(int batch, int h, int w, int c, int crop\_height, int crop\_width, int flip, float angle, float saturation, float exposure)

{

fprintf(stderr, "Crop Layer: %d x %d -> %d x %d x %d image\n", h,w,crop\_height,crop\_width,c);

crop\_layer l = { (LAYER\_TYPE)0 };

l.type = CROP;

l.batch = batch;

l.h = h;

l.w = w;

l.c = c;

l.scale = (float)crop\_height / h;

l.flip = flip;

l.angle = angle;

l.saturation = saturation;

l.exposure = exposure;

l.out\_w = crop\_width;

l.out\_h = crop\_height;

l.out\_c = c;

l.inputs = l.w \* l.h \* l.c;

l.outputs = l.out\_w \* l.out\_h \* l.out\_c;

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

l.forward = forward\_crop\_layer;

l.backward = backward\_crop\_layer;

#ifdef GPU

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

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

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

l.rand\_gpu = cuda\_make\_array(0, l.batch\*8);

#endif

return l;

}

void resize\_crop\_layer(layer \*l, int w, int h)

{

l->w = w;

l->h = h;

l->out\_w = l->scale\*w;

l->out\_h = l->scale\*h;

l->inputs = l->w \* l->h \* l->c;

l->outputs = l->out\_h \* l->out\_w \* l->out\_c;

l->output = (float\*)xrealloc(l->output, l->batch \* l->outputs \* sizeof(float));

#ifdef GPU

cuda\_free(l->output\_gpu);

l->output\_gpu = cuda\_make\_array(l->output, l->outputs\*l->batch);

#endif

}

void forward\_crop\_layer(const crop\_layer l, network\_state state)

{

int i,j,c,b,row,col;

int index;

int count = 0;

int flip = (l.flip && rand()%2);

int dh = rand()%(l.h - l.out\_h + 1);

int dw = rand()%(l.w - l.out\_w + 1);

float scale = 2;

float trans = -1;

if(l.noadjust){

scale = 1;

trans = 0;

}

if(!state.train){

flip = 0;

dh = (l.h - l.out\_h)/2;

dw = (l.w - l.out\_w)/2;

}

for(b = 0; b < l.batch; ++b){

for(c = 0; c < l.c; ++c){

for(i = 0; i < l.out\_h; ++i){

for(j = 0; j < l.out\_w; ++j){

if(flip){

col = l.w - dw - j - 1;

}else{

col = j + dw;

}

row = i + dh;

index = col+l.w\*(row+l.h\*(c + l.c\*b));

l.output[count++] = state.input[index]\*scale + trans;

}

}

}

}

}