## Image.c

//Syolo

image mask\_to\_rgb(image mask)

{

int n = mask.c;

image im = make\_image(mask.w, mask.h, 3);

int i, j;

for (j = 0; j < n; ++j) {

int offset = j \* 123457 % n;

float red = get\_color(2, offset, n);

float green = get\_color(1, offset, n);

float blue = get\_color(0, offset, n);

for (i = 0; i < im.w \* im.h; ++i) {

im.data[i + 0 \* im.w \* im.h] += mask.data[j \* im.h \* im.w + i] \* red;

im.data[i + 1 \* im.w \* im.h] += mask.data[j \* im.h \* im.w + i] \* green;

im.data[i + 2 \* im.w \* im.h] += mask.data[j \* im.h \* im.w + i] \* blue;

}

}

return im;

}

//Syolo

image load\_image\_stb\_mask(char\* filename, int channels)

{

int w, h, c;

unsigned char\* data = stbi\_load(filename, &w, &h, &c, channels);

if (!data) {

char shrinked\_filename[1024];

if (*strlen*(filename) >= 1024) *sprintf*(shrinked\_filename, "name is too long");

else *sprintf*(shrinked\_filename, "%s", filename);

*fprintf*(*stderr*, "Cannot load image \"%s\"\nSTB Reason: %s\n", shrinked\_filename, stbi\_failure\_reason());

*FILE*\* fw = *fopen*("bad.list", "a");

*fwrite*(shrinked\_filename, sizeof(char), *strlen*(shrinked\_filename), fw);

char\* new\_line = "\n";

*fwrite*(new\_line, sizeof(char), *strlen*(new\_line), fw);

*fclose*(fw);

if (check\_mistakes) {

*printf*("\n Error in load\_image\_stb() \n");

*getchar*();

}

return make\_image(10, 10, 3);

//exit(EXIT\_FAILURE);

}

if (channels) c = channels;

int i, j, k;

image im = make\_image(w, h, c);

for (k = 0; k < c; ++k) {

for (j = 0; j < h; ++j) {

for (i = 0; i < w; ++i) {

int dst\_index = i + w \* j + w \* h \* k;

int src\_index = k + c \* i + c \* w \* j;

im.*data*[dst\_index] = (float)data[src\_index] / 255.;

}

}

}

*free*(data);

return im;

}

## Network.c

//Syolo

image get\_network\_Simage(network \*net)

{

int i;

for (i = net.n - 1; i >= 0; --i) {

image m = get\_network\_image\_layer(net, i);

if (m.h != 0) return m;

}

image def = { 0 };

return def;

}

## Darknet.c

//Syolo

else if (0 == *strcmp*(argv[1], "segmenter")) {

run\_segmenter(argc, argv);

}

## Darknet.h

//Syolo

typedef struct {

int w;

int h;

float scale;

float rad;

float dx;

float dy;

float aspect;

} augment\_args;

## Data.c

//Syolo

else if (a.type == SEGMENTATION\_DATA) {\*a.d = load\_data\_seg(a.n, a.paths, a.m, a.w, a.h, a.classes, a.min, a.max, a.angle, a.aspect, a.hue, a.saturation, a.exposure, a.scale); }

//Syolo

data load\_data\_seg(int n, char\*\* paths, int m, int w, int h, int classes, int min, int max, float angle, float aspect, float hue, float saturation, float exposure, int div)

{

char\*\* random\_paths = get\_random\_paths(paths, n, m);

int i;

data d = { 0 };

d.shallow = 0;

d.X.rows = n;

d.X.vals = *calloc*(d.X.rows, sizeof(float\*));

d.X.cols = h \* w \* 3;

d.y.rows = n;

d.y.cols = h \* w \* classes / div / div;

d.y.vals = *calloc*(d.X.rows, sizeof(float\*));

for (i = 0; i < n; ++i) {

image orig = load\_image\_color(random\_paths[i], 0, 0);

//augment\_args a;

//a.rad = 0;

//a.scale = 1;

//a.w = orig.w;

//a.h = orig.h;

//a.dx = 0;

//a.dy = 0;

//a.aspect = 1;

augment\_args a = random\_augment\_args(orig, angle, aspect, min, max, w, h);

image sized = rotate\_crop\_image(orig, a.rad, a.scale, a.w, a.h, a.dx, a.dy, a.aspect);

//printf("scale is %f, aspect is %f, rad is %f, a.w is %f, a.h is %f, a.dx is %f, a.dy is %f\n ", a.scale, a.aspect, a.rad, a.w, a.h, a.dx, a.dy);

int flip = rand() % 2;

if (flip) flip\_image(sized);

random\_distort\_image(sized, hue, saturation, exposure);

d.X.vals[i] = sized.*data*;

//image mask = get\_segmentation\_image(random\_paths[i], orig.w, orig.h, classes);

image mask = load\_image\_gray(random\_paths[i], orig.w, orig.h);

image sized\_m = rotate\_crop\_image\_seg(mask, a.rad, a.scale / div, a.w / div, a.h / div, a.dx / div, a.dy / div, a.aspect);

if (flip) flip\_image(sized\_m);

d.y.vals[i] = sized\_m.data;

//show\_image(sized\_m, "part", 0);

//show\_image(mask, "mask", 0);

//show\_image(sized,"size",0);

free\_image(orig);

free\_image(mask);

/\*

image rgb = mask\_to\_rgb(sized\_m, classes);

show\_image(rgb, "part");

show\_image(sized, "orig");

cvWaitKey(0);

free\_image(rgb);

\*/

}

*free*(random\_paths);

return d;

}

//Syolo

image load\_image\_gray(char\* path, int w, int h) {

char labelpath[4096];

find\_replace(labelpath, "\_leftImg8bit.png", "\_gtFine\_instanceIds.png", labelpath);

find\_replace(path, "images", "mask", labelpath);

find\_replace(labelpath, "JPEGImages", "mask", labelpath);

find\_replace(labelpath, ".jpg", ".txt", labelpath);

find\_replace(labelpath, ".JPG", ".txt", labelpath);

find\_replace(labelpath, "\_leftImg8bit.png", "\_gtFine\_instanceIds.png", labelpath);

return load\_image(labelpath, w, h, 1);

}

## Data.h

//Syolo

data load\_data\_seg(int n, char\*\* paths, int m, int w, int h, int classes, int min, int max, float angle, float aspect, float hue, float saturation, float exposure, int div);

## Image.c

#define TWO\_PI 6.2831853071795864769252866f

//Syolo

augment\_args random\_augment\_args(image im, float angle, float aspect, int low, int high, int w, int h)

{

augment\_args a = { 0 };

aspect = rand\_scale(aspect);

int r = rand\_int(low, high);

int min = (im.h < im.w\* aspect) ? im.h : im.w \* aspect;

float scale = (float)r / min;

float rad = rand\_uniform(-angle, angle) \* TWO\_PI / 360.;

float dx = (im.w \* scale / aspect - w) / 2.;

float dy = (im.h \* scale - w) / 2.;

//if(dx < 0) dx = 0;

//if(dy < 0) dy = 0;

dx = rand\_uniform(-dx, dx);

dy = rand\_uniform(-dy, dy);

a.rad = rad;

a.scale = scale;

a.w = w;

a.h = h;

a.dx = dx;

a.dy = dy;

a.aspect = aspect;

return a;

}

//Syolo

image create\_mask(image mask)

{

int n = mask.c;

image im = make\_image(mask.w, mask.h, 1);

int i, j;

for (i = 0; i < im.w \* im.h; ++i) {

if (mask.data[i] < 0.5) {

im.data[i] += 0;

}

else {

im.data[i] += 1;

}

}

}

return im;

}

## Image.h

augment\_args random\_augment\_args(image im, float angle, float aspect, int low, int high, int w, int h); //Syolo

image rotate\_crop\_image(image im, float rad, float s, int w, int h, float dx, float dy, float aspect)//Syolo

image load\_image\_stb\_mask(char\* filename, int channels);//Syolo

## parser.c

#include "logistic\_layer.h"//Syolo

if (*strcmp*(type, "[logistic]") == 0) return LOGXENT;//Syolo

//Syolo

layer parse\_logistic(list\* options, size\_params params)

{

layer l = make\_logistic\_layer(params.batch, params.inputs);

l.h = l.out\_h = params.h;

l.w = l.out\_w = params.w;

l.c = l.out\_c = params.c;

return l;

}

//Syolo

else if (lt == LOGXENT) {

l = parse\_logistic(options, params);

}

## Blas.c

//Syolo

float dice\_loss\_cpu(int n, float\* pred, float\* truth, float\* delta, float\* error)

{

int smooth = 1;

float intersection = 0;

float sum\_target = 0;

float sum\_pred = 0;

float sum\_ANB = 0;

float sum\_AUB = 0;

float dice = 0;

for (int j = 0; j < n; ++j) {

sum\_target += truth[j];

sum\_pred += pred[j];

}

sum\_AUB = sum\_target + sum\_pred + smooth;

for (int i = 0; i < n; ++i) {

float t = truth[i];

float p = pred[i];

sum\_ANB += t \* p;

delta[i] = t - p;

}

dice = 2 \* (sum\_ANB + smooth) / (sum\_AUB + smooth);

return 1 - dice;

}

## Blas.h

void dice\_loss\_cpu(int n, float\* pred, float\* truth, float\* delta, float\* error);//Syolo

void logistic\_x\_ent\_cpu(int n, float\* pred, float\* truth, float\* delta, float\* error);//Syolo