**Ryolo\_layer.c中**

#include "Ryolo\_layer.h"

layer make\_Ryolo\_layer

l.type = RYOLO;

l.forward = forward\_Ryolo\_layer;

l.backward = backward\_Ryolo\_layer;

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

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

void resize\_Ryolo\_layer

box get\_Ryolo\_box

ious delta\_Ryolo\_box

void averages\_Ryolo\_deltas

void delta\_Ryolo\_class

int compare\_Ryolo\_class

static int Ryolo\_entry\_index

yolo->Ryolo

entry\_index->Ryolo\_entry\_index

**darknet.h中**

RYOLO//rotate

} LAYER\_TYPE;

**Parser.c中添加**

//rotate\_yolo

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

{

int classes = option\_find\_int(options, "classes", 20);

int total = option\_find\_int(options, "num", 1);

int num = total;

char\* a = option\_find\_str(options, "mask", 0);

int\* mask = parse\_yolo\_mask(a, &num);

int max\_boxes = option\_find\_int\_quiet(options, "max", 200);

layer l = make\_Ryolo\_layer(params.batch, params.w, params.h, num, total, mask, classes, max\_boxes);

if (l.outputs != params.inputs) {

*printf*("Error: l.outputs == params.inputs \n");

*printf*("filters= in the [convolutional]-layer doesn't correspond to classes= or mask= in [yolo]-layer \n");

*exit*(*EXIT\_FAILURE*);

}

//assert(l.outputs == params.inputs);

l.max\_delta = option\_find\_float\_quiet(options, "max\_delta", *FLT\_MAX*); // set 10

char\* cpc = option\_find\_str(options, "counters\_per\_class", 0);

l.classes\_multipliers = get\_classes\_multipliers(cpc, classes, l.max\_delta);

l.label\_smooth\_eps = option\_find\_float\_quiet(options, "label\_smooth\_eps", 0.0f);

l.scale\_x\_y = option\_find\_float\_quiet(options, "scale\_x\_y", 1);

l.objectness\_smooth = option\_find\_int\_quiet(options, "objectness\_smooth", 0);

l.iou\_normalizer = option\_find\_float\_quiet(options, "iou\_normalizer", 0.75);

l.cls\_normalizer = option\_find\_float\_quiet(options, "cls\_normalizer", 1);

char\* iou\_loss = option\_find\_str\_quiet(options, "iou\_loss", "mse"); // "iou");

if (*strcmp*(iou\_loss, "mse") == 0) l.iou\_loss = MSE;

else if (*strcmp*(iou\_loss, "giou") == 0) l.iou\_loss = GIOU;

else if (*strcmp*(iou\_loss, "diou") == 0) l.iou\_loss = DIOU;

else if (*strcmp*(iou\_loss, "ciou") == 0) l.iou\_loss = CIOU;

else l.iou\_loss = IOU;

*fprintf*(*stderr*, "[Ryolo] params: iou loss: %s (%d), iou\_norm: %2.2f, cls\_norm: %2.2f, scale\_x\_y: %2.2f\n",

iou\_loss, l.iou\_loss, l.iou\_normalizer, l.cls\_normalizer, l.scale\_x\_y);

char\* iou\_thresh\_kind\_str = option\_find\_str\_quiet(options, "iou\_thresh\_kind", "iou");

if (*strcmp*(iou\_thresh\_kind\_str, "iou") == 0) l.iou\_thresh\_kind = IOU;

else if (*strcmp*(iou\_thresh\_kind\_str, "giou") == 0) l.iou\_thresh\_kind = GIOU;

else if (*strcmp*(iou\_thresh\_kind\_str, "diou") == 0) l.iou\_thresh\_kind = DIOU;

else if (*strcmp*(iou\_thresh\_kind\_str, "ciou") == 0) l.iou\_thresh\_kind = CIOU;

else {

*fprintf*(*stderr*, " Wrong iou\_thresh\_kind = %s \n", iou\_thresh\_kind\_str);

l.iou\_thresh\_kind = IOU;

}

l.beta\_nms = option\_find\_float\_quiet(options, "beta\_nms", 0.6);

char\* nms\_kind = option\_find\_str\_quiet(options, "nms\_kind", "default");

if (*strcmp*(nms\_kind, "default") == 0) l.nms\_kind = DEFAULT\_NMS;

else {

if (*strcmp*(nms\_kind, "greedynms") == 0) l.nms\_kind = GREEDY\_NMS;

else if (*strcmp*(nms\_kind, "diounms") == 0) l.nms\_kind = DIOU\_NMS;

else l.nms\_kind = DEFAULT\_NMS;

*printf*("nms\_kind: %s (%d), beta = %f \n", nms\_kind, l.nms\_kind, l.beta\_nms);

}

l.jitter = option\_find\_float(options, "jitter", .2);

l.*resize* = option\_find\_float\_quiet(options, "resize", 1.0);

l.focal\_loss = option\_find\_int\_quiet(options, "focal\_loss", 0);

l.ignore\_thresh = option\_find\_float(options, "ignore\_thresh", .5);

l.truth\_thresh = option\_find\_float(options, "truth\_thresh", 1);

l.iou\_thresh = option\_find\_float\_quiet(options, "iou\_thresh", 1); // recommended to use iou\_thresh=0.213 in [yolo]

l.random = option\_find\_float\_quiet(options, "random", 0);

l.track\_history\_size = option\_find\_int\_quiet(options, "track\_history\_size", 5);

l.sim\_thresh = option\_find\_int\_quiet(options, "sim\_thresh", 0.8);

l.dets\_for\_track = option\_find\_int\_quiet(options, "dets\_for\_track", 1);

l.dets\_for\_show = option\_find\_int\_quiet(options, "dets\_for\_show", 1);

l.track\_ciou\_norm = option\_find\_float\_quiet(options, "track\_ciou\_norm", 0.01);

int embedding\_layer\_id = option\_find\_int\_quiet(options, "embedding\_layer", 999999);

if (embedding\_layer\_id < 0) embedding\_layer\_id = params.index + embedding\_layer\_id;

if (embedding\_layer\_id != 999999) {

*printf*(" embedding\_layer\_id = %d, ", embedding\_layer\_id);

layer le = params.net.layers[embedding\_layer\_id];

l.embedding\_layer\_id = embedding\_layer\_id;

l.embedding\_output = (float\*)xcalloc(le.batch \* le.outputs, sizeof(float));

l.embedding\_size = le.n / l.n;

*printf*(" embedding\_size = %d \n", l.embedding\_size);

}

char\* map\_file = option\_find\_str(options, "map", 0);

if (map\_file) l.*map* = read\_map(map\_file);

a = option\_find\_str(options, "anchors", 0);

if (a) {

int len = *strlen*(a);

int n = 1;

int i;

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

if (a[i] == '#') break;

if (a[i] == ',') ++n;

}

for (i = 0; i < n && i < total \* 2; ++i) {

float bias = *atof*(a);

l.biases[i] = bias;

a = *strchr*(a, ',') + 1;

}

}

return l;

}

LAYER\_TYPE string\_to\_layer\_type(char \* type)

if (*strcmp*(type, "[ryolo]") == 0) return RYOLO; //rotate

**network.c中添加**

else if (l.type == RYOLO) {

resize\_Ryolo\_layer(&l, w, h);//rotate\_yolo

}

**Data.c中**

box\_label \*read\_Rboxes(char \*filename, int \*n)//rotate

int fill\_truth\_Rdetection(const char\* path, int num\_boxes, int truth\_size, float\* truth, int classes, int flip, float dx, float dy, float sx, float sy,

int net\_w, int net\_h)

data load\_data\_Rdetection(int n, char\*\* paths, int m, int w, int h, int c, int boxes, int truth\_size, int classes, int use\_flip, int gaussian\_noise, int use\_blur, int use\_mixup,

float jitter, float resize, float hue, float saturation, float exposure, int mini\_batch, int track, int augment\_speed, int letter\_box, int mosaic\_bound, int contrastive, int contrastive\_jit\_flip, int show\_imgs)

**darknet.h中**

typedef struct Rbox {

float x, y, w, h, r1, r2;//rotate\_yolo

float adv;

float jud;

} Rbox;

typedef struct Rbox\_label {

int id;

int track\_id;

//float x, y, w, h;

float x, y, w, h, r1, r2;//rotate\_yolo

int adv;//rotate\_yolo

int jud;//rotate\_yolo

float left, right, top, bottom;

} Rbox\_label;

typedef enum {

CLASSIFICATION\_DATA, RDETECTION\_DATA,DETECTION\_DATA, CAPTCHA\_DATA, REGION\_DATA, IMAGE\_DATA, COMPARE\_DATA, WRITING\_DATA, SWAG\_DATA, TAG\_DATA, OLD\_CLASSIFICATION\_DATA, STUDY\_DATA, DET\_DATA, SUPER\_DATA, LETTERBOX\_DATA, REGRESSION\_DATA, SEGMENTATION\_DATA, INSTANCE\_DATA, ISEG\_DATA, ROTATE\_DATA

} data\_type;//rotate

data load\_data\_detection(int n, char \*\*paths, int m, int w, int h, int c, int boxes, int truth\_size, int classes, int use\_flip, int gaussian\_noise, int use\_blur, int use\_mixup,

float jitter, float resize, float hue, float saturation, float exposure, int mini\_batch, int track, int augment\_speed, int letter\_box, int mosaic\_bound, int contrastive, int contrastive\_jit\_flip, int show\_imgs);

**Box.c添加**

box float\_to\_Rbox\_stride(float\* f, int stride)

{

box b = { 0 };

b.x = f[0];

b.y = f[1 \* stride];

b.w = f[2 \* stride];

b.h = f[3 \* stride];

b.r1 = f[4 \* stride];//rotate\_yolo

b.r2 = f[5 \* stride];//rotate\_yolo

b.adv = f[6 \* stride];//rotate\_yolo

b.jud = f[7 \* stride];//rotate\_yolo

return b;

}

Data.c