#include "network.h"

#include "utils.h"

#include "parser.h"

void train\_tag(char \*cfgfile, char \*weightfile, int clear)

{

srand(time(0));

float avg\_loss = -1;

char \*base = basecfg(cfgfile);

char\* backup\_directory = "backup/";

printf("%s\n", base);

network net = parse\_network\_cfg(cfgfile);

if(weightfile){

load\_weights(&net, weightfile);

}

if (clear) {

\*net.seen = 0;

\*net.cur\_iteration = 0;

}

printf("Learning Rate: %g, Momentum: %g, Decay: %g\n", net.learning\_rate, net.momentum, net.decay);

int imgs = 1024;

list\* plist = get\_paths("tag/train.list");

char \*\*paths = (char \*\*)list\_to\_array(plist);

printf("%d\n", plist->size);

int N = plist->size;

clock\_t time;

pthread\_t load\_thread;

data train;

data buffer;

load\_args args = {0};

args.w = net.w;

args.h = net.h;

args.min = net.w;

args.max = net.max\_crop;

args.size = net.w;

args.paths = paths;

args.classes = net.outputs;

args.n = imgs;

args.m = N;

args.d = &buffer;

args.type = TAG\_DATA;

args.angle = net.angle;

args.exposure = net.exposure;

args.saturation = net.saturation;

args.hue = net.hue;

fprintf(stderr, "%d classes\n", net.outputs);

load\_thread = load\_data\_in\_thread(args);

int epoch = (\*net.seen)/N;

while(get\_current\_batch(net) < net.max\_batches || net.max\_batches == 0){

time=clock();

pthread\_join(load\_thread, 0);

train = buffer;

load\_thread = load\_data\_in\_thread(args);

printf("Loaded: %lf seconds\n", sec(clock()-time));

time=clock();

float loss = train\_network(net, train);

if(avg\_loss == -1) avg\_loss = loss;

avg\_loss = avg\_loss\*.9 + loss\*.1;

printf("%d, %.3f: %f, %f avg, %f rate, %lf seconds, %ld images\n", get\_current\_batch(net), (float)(\*net.seen)/N, loss, avg\_loss, get\_current\_rate(net), sec(clock()-time), \*net.seen);

free\_data(train);

if(\*net.seen/N > epoch){

epoch = \*net.seen/N;

char buff[256];

sprintf(buff, "%s/%s\_%d.weights",backup\_directory,base, epoch);

save\_weights(net, buff);

}

if(get\_current\_batch(net)%100 == 0){

char buff[256];

sprintf(buff, "%s/%s.backup",backup\_directory,base);

save\_weights(net, buff);

}

}

char buff[256];

sprintf(buff, "%s/%s.weights", backup\_directory, base);

save\_weights(net, buff);

pthread\_join(load\_thread, 0);

free\_data(buffer);

free\_network(net);

free\_ptrs((void\*\*)paths, plist->size);

free\_list(plist);

free(base);

}

void test\_tag(char \*cfgfile, char \*weightfile, char \*filename)

{

network net = parse\_network\_cfg(cfgfile);

if(weightfile){

load\_weights(&net, weightfile);

}

set\_batch\_network(&net, 1);

srand(2222222);

int i = 0;

char \*\*names = get\_labels("data/tags.txt");

clock\_t time;

int indexes[10];

char buff[256];

char \*input = buff;

int size = net.w;

while(1){

if(filename){

strncpy(input, filename, 256);

}else{

printf("Enter Image Path: ");

fflush(stdout);

input = fgets(input, 256, stdin);

if(!input) return;

strtok(input, "\n");

}

image im = load\_image\_color(input, 0, 0);

image r = resize\_min(im, size);

resize\_network(&net, r.w, r.h);

printf("%d %d\n", r.w, r.h);

float \*X = r.data;

time=clock();

float \*predictions = network\_predict(net, X);

top\_predictions(net, 10, indexes);

printf("%s: Predicted in %f seconds.\n", input, sec(clock()-time));

for(i = 0; i < 10; ++i){

int index = indexes[i];

printf("%.1f%%: %s\n", predictions[index]\*100, names[index]);

}

if(r.data != im.data) free\_image(r);

free\_image(im);

if (filename) break;

}

}

void run\_tag(int argc, char \*\*argv)

{

if(argc < 4){

fprintf(stderr, "usage: %s %s [train/test/valid] [cfg] [weights (optional)]\n", argv[0], argv[1]);

return;

}

int clear = find\_arg(argc, argv, "-clear");

char \*cfg = argv[3];

char \*weights = (argc > 4) ? argv[4] : 0;

char \*filename = (argc > 5) ? argv[5] : 0;

if(0==strcmp(argv[2], "train")) train\_tag(cfg, weights, clear);

else if(0==strcmp(argv[2], "test")) test\_tag(cfg, weights, filename);

}