#include "network.h"

#include "utils.h"

#include "parser.h"

char \*dice\_labels[] = {"face1","face2","face3","face4","face5","face6"};

void train\_dice(char \*cfgfile, char \*weightfile)

{

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);

}

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

int imgs = 1024;

int i = \*net.seen/imgs;

char \*\*labels = dice\_labels;

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

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

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

clock\_t time;

while(1){

++i;

time=clock();

data train = load\_data\_old(paths, imgs, plist->size, labels, 6, net.w, net.h);

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: %f, %f avg, %lf seconds, %ld images\n", i, loss, avg\_loss, sec(clock()-time), \*net.seen);

free\_data(train);

if((i % 100) == 0) net.learning\_rate \*= .1;

if(i%100==0){

char buff[256];

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

save\_weights(net, buff);

}

}

}

void validate\_dice(char \*filename, char \*weightfile)

{

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

if(weightfile){

load\_weights(&net, weightfile);

}

srand(time(0));

char \*\*labels = dice\_labels;

list \*plist = get\_paths("data/dice/dice.val.list");

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

int m = plist->size;

free\_list(plist);

data val = load\_data\_old(paths, m, 0, labels, 6, net.w, net.h);

float \*acc = network\_accuracies(net, val, 2);

printf("Validation Accuracy: %f, %d images\n", acc[0], m);

free\_data(val);

}

void test\_dice(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 = dice\_labels;

char buff[256];

char \*input = buff;

int indexes[6];

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, net.w, net.h);

float \*X = im.data;

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

top\_predictions(net, 6, indexes);

for(i = 0; i < 6; ++i){

int index = indexes[i];

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

}

free\_image(im);

if (filename) break;

}

}

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

{

if(argc < 4){

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

return;

}

char \*cfg = argv[3];

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

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

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

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

else if(0==strcmp(argv[2], "valid")) validate\_dice(cfg, weights);

}