آرین تشکر - 40023494

تکلیف دوم درس سیستم های چند رسانه ای پیشرفته

ساختار فایل تحویلی:

پوشه ی اصلی فایل تکلیف شامل یک فایل Report.pdf (فایلی که هم اکنون در حال خواندن آن هستید) و یک پوشه به نام Deliverables می باشد. در پوشه ی Deliverables چندین فایل سورس متلب قرار گرفته است که هر کدام از آن ها برای قسمتی از تکلیف مورد استفاده قرار می گیرند (به خصوص فایل main.m شامل کد تست بخش های مختلف می باشد). همچنین در این پوشه، دو پوشه ی دیگر به نام های Inputs و Results قرار دارند که به ترتیب فایل های ورودی و نتایج را در خود جای می دهند. به علاوه در این پوشه، یک پوشه به نام NNP (مخفف Neural Network Predictor) وجود دارد. که در برگیرنده تلاش ناموفق برای ایجاد یک Predictor با استفاده از یک شبکه عصبی برای سوال دوم می باشد.

تمامی فایل به صورت کامل کامنت گذاری شده اند، در صورت نیاز می توانید به اصل کد ها برای نکات پیاده سازی و توضیحات تکمیلی مراجعه کنید.

سوال 1) پیاده سازی پیش گوی MED:

سورس کد مربوط: Deliverables\MED.m

کد ها:

MED::encPredict:

```
function obj = encPredict(obj)
            % Calculates an error map from a given input image using MED.
            [H, W] = size(obj.image);
            pred = int16(zeros([H, W]));
            obj.error = int16(zeros([H, W]));
            % Pad one above and one to the left with 0
            padded = int16(padarray(obj.image, [1, 1], 0, 'pre'));
            % A = W, B = N, C = NW
            for i = 2:H+1
                for j = 2:W+1
                    % MED
                    A = padded(i, j-1);
                    B = padded(i-1, j);
                    C = padded(i-1, j-1);
                    if C >= max(A, B)
                        pred(i-1, j-1) = min(A, B);
                    elseif C <= min(A,B)
                        pred(i-1, j-1) = max(A, B);
                    else
                        pred(i-1, j-1) = A + B - C;
                    end
                    obj.error(i-1, j-1)= int16(obj.image(i-1, j-1)) - pred(i-1, j-1);
                end
            end
        end
```

تنها نکته ی حائز اهمیت در این تابع بخش padding است که در طی آن برای انجام prediction روی لبه های بالا و سمت چپ تصویر نیاز به پیکسل های اضافه داریم که آن ها را با 0-padding به تصویر می افزاییم. بقیه ی کد مستقیماً از الگوریتم MED توضیح داده شده در کلاس اقتباس شده است. در نهایت تصویر error به عنوان یک آرایه int16 در شئ کلاس ذخیره می شود. (فرض می کنیم که در نهایت با انجام error-remapping این آرایه را می توان به int8 تبدیل کرد، بنابراین عناصر این آرایه هنوز هم عملاً 8 بیتی هستند.)

MED:decPredict:

```
function rec = decPredict(obj)
            % Counterpart of encPredict. Performs MED prediction on
            % using only the error map as reference. Operations are
            % symmetric to encPredict. Returns the reconstructed image.
            [H, W] = size(obj.error);
            pred = int16(zeros([H+1, W+1]));
            rec = uint8(zeros([H, W]));
            for i = 2:H+1
                for j = 2:W+1
                    A = pred(i, j-1);
                    B = pred(i-1, j);
                    C = pred(i-1, j-1);
                    if C >= max(A, B)
                        pred(i, j) = min(A, B);
                    elseif C <= min(A,B)
                        pred(i, j) = max(A, B);
                    else
                        pred(i, j) = A + B - C;
                    end
                    pred(i, j) = pred(i, j) + obj.error(i-1, j-1);
                    rec(i-1, j-1) = uint8(pred(i, j));
                end
            end
        end
```

این تابع نیز دقیقاً همانند encPredict عمل می کند با این تفاوت که در نهایت برای تشکیل تصویر اصلی مقدار پیشگویی را با مقدار خطای متناظر با همان پیکسل جمع می کند تا به وضعیت lossless برسیم.

سوال 2) ارائه ی یک پیشگوی بهتر از MED:

سورس کد مربوط: Deliverables\AdaGAP.m

نتایج مربوط: Deliverables\Results\predictor_comparison.mat

ایده ی اولیه ی من استفاده از یک شبکه ی عصبی با استفاده از 7 همسایه ی علّی برای پیشگویی مقدارپیکسل فعلی بود که این شبکه به دلیل تعداد بسیار کم Feature ها، نمی توانست به خوبی آموزش داده شود. در واقع نیاز به استفاده از یک شبکه ی عصبی dynamic است که وزن های آن وابسته به ورودی های مدل باشد. این تلاش نا موفق را می توانید به همراه دیتاست استفاده شده برای آموزش در پوشه ی Deliverables/NNP مشاهده کنید.

ایده ی بعدی ترکیب دو روش GAP و ALCM برای بهبود کارایی MED بود. این روش (که آن را AdaGAP نامیده ام)، از همان ایده ی اصلی استفاده شده در GAP و از همان مقادیر ثابت هیوریستکی استفاده می کند با این تفاوت که مقدار ارث بری پیکسل در حال پیشگویی از پیکسل بالایی (N) و پیکسل سمت چپ (W) بسته به مقدار خطای پیشگویی پیکسل قبلی، متغیر است. در واقع در صورتی که مقدار خطا مثبت باشد به ضریب پیکسل پیشگوی غالب (N یا W) مقدار ثابت ROC (به صورت پیش فرض برابر 1/128) افزوده می شود و در غیر این صورت همین مقدار از ضریب همان پیکسل کاسته می شود و در صورتی که خطا صفر باشد مقدار ضریب تغییری نمی کند. همچنین از آنجایی که ممکن است مقدار ضریب بزرگ تر از 1 باشد، مقدار پیشگویی نهایی همیشه با 255 مینیمم گرفته می شود. همچنین به دلیل این که بر خلاف ALCM مقدار هیچ ضریب دیگری غیر از ضریب غالب تغییر نمی کند (در ALCM هم ضریب غالب و هم ضریب مغلوب تغییر پیدا می کردند. این امر به تعادل مجموع ضرایب کمک می کرد)، مجبوریم برای این که در صورت بروز خطا در ضرایب مدت زیادی را برای بازگشتن به ضرایب صحیح صرف نکنیم، روی مقدار خطایی که برای آن ضریب اعمال می شود یک آستانه تعریف کنیم که در این آزمایش این مقدار به صورت تجربی برای 3 شد. این بدان معناست که ضرایب در صورتی که قدر مطلق خطای پیکسل جاری کمتر از 3 باشد به اندازه ی ROC تغییر می کنند و در غیر این صورت به مقادیر اولیه شان reset می شوند. نتیجه ی این عملیات این است که قله ی نمودار هیستوگرام خطا های تولید شده peak تیز تری تولید می کند و علاوه بد آن دامنه ی کوچک تری نیز خواهد داشت. نقطه ی تمایز دیگر AdaGAP با روش GAP معمولی این است که AdaGAP از padding استفاده نمی کند. به جای استفاده از padding برای سطر اول از پیکسل سمت چپ و برای ستون اول از پیکسل بالا به عنوان prediction استفاده میکنیم و اولین پیکسل تصویر را نیز به عنوان overhead در نظر می گیریم. همچنین edge case های سطر دوم و ستون های دوم و آخر نیز همانطور که درکد مشخص شده است، handle شده اند. دلیل این کار عملکرد بهتر آن در برابر روش padding بود (در ازای اضافه شدن یک پیکسل overhead).

به علت طولانی بودن کد ها و نبودن امکان نمایش کد به صورت منظم در این سند، لطفاً به اصل سورس ها مراجعه کنید.

در جدول صفحه ی بعد می توانید نتایج مقایسه ی AdaGAP را با MED مشاهده کنید.

Image Name	Image Entropy	Error Entropy (MED)	Error BPP (AdaGAP)	Run Time (AdaGAP)
Baboon	7.2925	5.2340	5.2185	11.6827
Bridge	7.6819	5.6689	5.6332	12.2162
Cells	4.8106	3.6835	3.5331	6.7018
Livingroom	7.2952	4.8392	4.8353	10.8798
MRI_1	6.4273	3.7660	3.8093	8.3437
MRI_2	6.6198	3.6123	3.6590	7.5327
Peppers	7.5715	4.8437	4.6516	9.4028
Retina	5.6204	3.0184	2.9832	5.5869
Average	6.6649	4.3333	4.2904	9.0433

سوال 3) Quantized RLE:

سورس کد های مربوط: Deliverables\GRLOQ.m, QRLE.m, RLE.m

ابتدا باید تصویر با توجه به محدودیت near-lossless تعیین شده است، طوری بهینه شود که RLE بتواند بهترین run های ممکن را ارائه کند. برای این کار از یک الگوریتم حریصانه (و نه لزوماً بهینه) استفاده شده است.

کد:

GRLOQ.m:

```
% Greedy Run Length Optimized Quantizer
function [R, C, CF_Vector, RF_Vector] = GRLOQ(img, d)
    % return R and C so that the original image is reconstructable
    [R, C] = size(img);
    % return both the row-first and column-first vectors
    CF_Vector = int16(reshape(img, 1, []));
    RF_Vector = int16(reshape(img', 1, []));
    % greedily append any pixel to the previous one if it doesn't violate
    % the "d" near-lossless constraint.
    for i=2:length(CF_Vector)
        if abs(CF_Vector(i-1) - CF_Vector(i)) <= d
            CF_Vector(i) = CF_Vector(i-1);
        end
        if abs(RF_Vector(i-1) - RF_Vector(i)) <= d
            RF_Vector(i) = RF_Vector(i-1);
        end
    end
    CF_Vector = uint8(CF_Vector);
    RF_Vector = uint8(RF_Vector);
end
```

در این کد با دریافت یک تصویر grayscale به عنوان ورودی ابتدا آن را به دو بردار اول-سطر و اول-ستون تبدیل می کنیم. سپس با شروع از دومین مولفه ی بردار، اگر اندازه ی اختلاف عنصر فعلی و عنصر قبلی در محدوده ی d مشخص شده باشد، این مولفه را برابر

با مولفه ی قبلی قرار می دهیم تا موقع انجام RLE به run طولانی تری برسیم. ضمناً هر دو بردار quantize شده ی اول-سطر و اول-ستون را به عنوان خروجی بر می گردانیم تا بتوانیم پس از انجام RLE آنی که آنتروپی کمتری دارد را به عنوان خروجی نهایی گزارش کنیم.

کد:

QRLE.m

```
function [output_img, hx, psnr] = QRLE(img, d)
    if isstring(img)
        img = imread(img);
    end
    [R, C, CFV, RFV] = GRLOQ(img, d);
    % run RLE on both vectors from GRLOQ.
    % output the one with the least entropy.
    CFRLE = RLE(CFV);
    RFRLE = RLE(RFV);
    hx_c = (numel(CFRLE{1})*Entropy_Array(CFRLE{1}) + ...
        numel(CFRLE{2})*Entropy_Array(CFRLE{2}))/numel(img);
    hx_r = (numel(RFRLE{1})*Entropy_Array(RFRLE{1}) + ...
        numel(RFRLE{2})*Entropy_Array(RFRLE{2}))/numel(img);
    if hx_c >= hx_r
        output_img = uint8(reshape(RFV, R, C)');
        hx = hx_r;
    else
        output_img = uint8(reshape(CFV, R, C));
        hx = hx_c;
    end
    psnr = PSNR(img, output_img);
end
```

در این تابع به عنوان ورودی یک تصویر (یا فایل بارگزاری شده ی آن و یا مسیر آن) را به همراه محدودیت d دریافت می کنیم و به عنوان خروجی تصویر quantize شده ی نهایی، آنتروپی بهترین RLE ممکن بدست آمده از GRLOQ و PSNR تصویر نهایی را بر می گردانیم. در این تابع ابتدا پس از دریافت خروجی های اول-سطر و اول-ستون GRLOQ آن ها به تابع RLE می دهیم و سپس آنتروپی هر کدام را با استفاده از فرمول زیر محاسبه می کنیم:

$$Total\ Entropy\ (BPP) = \frac{len(run\_lengths) \times Entropy(run\_lengths) + len(values) \times Entropy(values)}{number\ of\ symbols}$$

و در نهایت RLE با آنتروپی کمتر را به عنوان خروجی بر می گردانیم.

نتایج نهایی این آزمایش را می توانید در صفحه ی بعد مشاهده کنید.

Image Name	d = 0		d = 1		d = 3		d = 5	
	Entropy	PSNR	Entropy	PSNR	Entropy	PSNR	Entropy	PSNR
Baboon	7.2236	Inf	6.8614	61.3485	6.0206	50.7402	5.3065	45.5649
Bridge	7.4914	Inf	7.0336	61.4512	6.2667	51.3202	5.4056	45.4223
Cells	4.7867	Inf	3.5314	55.8372	2.3836	49.8549	2.0429	46.8549
Livingroom	7.1309	Inf	6.5615	60.2337	5.2998	49.5251	4.1297	44.1321
MRI_1	5.6298	Inf	4.9226	59.8589	3.9566	50.3184	3.3229	45.7476
MRI_2	6.2122	Inf	5.0192	57.0360	3.5692	48.5777	2.8131	44.6495
Peppers	7.3710	Inf	6.6063	59.3733	4.9804	48.8668	3.6757	44.0310
Retina	5.0520	Inf	3.3081	55.5461	1.6866	47.7839	1.1309	44.0993
Average	6.3622	Inf	5.4805	58.8356	4.2704	49.5562	3.4784	45.0627