دانشکده فنی دانشگاه تهران

دانشکده برق و کامپیوتر

پروژه ۴ پردازش سیگنال‌های زمان گسسته

Introduction to Image Processing

طراح:

سجاد پاکدامن ساوجی

رایانامه

sj.pakdaman@ut.ac.ir

نیم سال دوم ۹۸-۹۹

دانشجویان عزیز، قبل از پاسخ‌گوئی به سوالات به نکات زیر توجه کنید:

۱. شما باید کدها و گزارش خود را با الگو DSP_CA4_StudentNumber.zip در محل تعیین شده آپلود کنید

۲. گزارش کار شما نیر از معیار های ارزیابی خواهد بود، در نتیجه زمان کافی برای تکمیل آن اختصاص دهید

۳. شما میتوانید سوالات خود را از طریق ایمیل sj.pakdaman@ut.ac.ir بپرسید

در این تمرین با مفاهیم پایه پردازش سیگنال های چند بعدی همانند تصاویر آشنا می‌شویم و مشاهده می‌کنیم که همچون سیگنال های ۱ بعدی، امکان پردازش در حوزه های مختلف همانند زمان-گسسته و فرکانس وجود دارد.

پردازش تصویر

پردازش تصویر به مجموعه روش هایی می‌گویند که در آن به افزایش کیفیت یک تصویر پرداخته می‌شود و یا ویژگی های مورد نظر از تصویر استخراج می‌شوند. در حالت پایه پردازش تصویر به یک نگاشت از فضایی با $m \times n$ بعد (یا ۳ $\times m \times n$ بعد بسته به این که تصویر رنگی است یا خیر) به فضایی دیگر تعبیر می‌شود. برد این نگشات در صورتی که خروجی پردازش همچنان تصویر باشد، $m \times n$ و در صورتی که ویژگی باشد می‌تواند ابعاد مختلفی داشته باشد،

در این حوزه نگاشت‌ها (فیلترها) به دو دسته کلی spatial domain filters و frequency domain filters تقسیم بندی می‌شوند.

spatial domain filtering

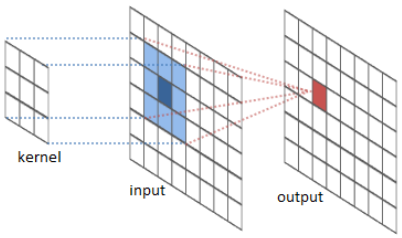
در این گروه از پردازش ها مقدار هر پیکسل در خروجی پردازش به مقدار دیگر پیکسل‌ها و مقدار خود پیکسل بستگی دارد، از این منظر این نوع پردازش یک رفتار محلی از خود نشان می‌دهد. این پردازش های محلی می‌توانند به صورت خطی و غیر خطی انجام شوند. گروهی از این فیلتر های خطی بر اساس عملیات convolution کار می‌کنند. که در زیر به توضیح بیشتر این عملیات می‌پردازیم.

شما با عملیات کانولوشن در حوزه زمان آشنایی دارید، این عملیات ، که با فرمول زیر معرفی می‌شود ، معادل برگرداندن یک سیگنال و لغزاندن آن بر روی سیگنال دیگر است، در حوزه زمان گسسته هم این عملیات به سادگی تعمیم پیدا می‌کند و در حقیقت پایه عملکرد فیلتر های FIR و پنجره گذاری قرار می‌گیرد.

$$(x * y)(t) = \int_{k \epsilon D} x(k) \times y(t-k)\, dk \qquad (x * y)[n] = \sum_{k \epsilon D} x[k] \times y[n-k]$$

$$g(x,y) = \sum_{s \epsilon D_1} \sum_{t \epsilon D_2} w(s,t) f(x-s, y-t)$$

حالت چند بعدی این عملیات هم کاملا به همین صورت است که یک سیگنال وارون می‌شود و در جهاتی که سیگنال درجه آزادی دارد بر روی سیگنال دیگری لغزانده می‌شود. در مورد تصاویر به فیلتر دوبعدی که باید بر روی تصاویر لغزانده شود kernel می‌گویند.برای این که با این مفهوم بیشتر آشنا شوید می‌توانید به اینجا مراجعه کنید.



شکل ۱: عملیات کانولوشن ۲ بعدی

در ادامه قصد داریم تاثیر تعدادی از kernel های معروف را بر روی یک تصویر مشاهده کنیم. ابتدا تصویر house.jpeg را با استفاده از دستور imread() بار گیری کنید. و کرنل های زیر را با استفاده از تابع conv2() بر روی آن اعمال کنید و توصیفی ازعمکرد هر کرنل گزارش

دهید. بدلیل وجود وزن های منفی در برخی از kernel ها ممکن است مقدار پیکسل منفی شود که قابل نمایش نیست در این حالت از قدر مطلق استفاده کنید.

0	0	0	-1	2	-1	-1	-1	-1	0.1111	0.1111	0.1111	0.0113	0.0838	0.0113	-1	-1	-1	0.0625	0.125	0.0625	0	-1	0
0	1	0	-1	2	-1	2	2	2	0.1111	0.1111	0.1111	0.0838	0.6193	0.0838	-1	8	-1	0.125	0.25	0.125	-1	5	-1
0	0	0	-1	2	-1	-1	-1	-1	0.1111	0.1111	0.1111	0.0113	0.1111	0.0113	-1	-1	-1	0.0625	0.125	0.0625	0	-1	0

identity (ح) line V (ز) line H (و) avg moving (ه) gauss (د) outline (ج) blur (ب) sharpen (آ)

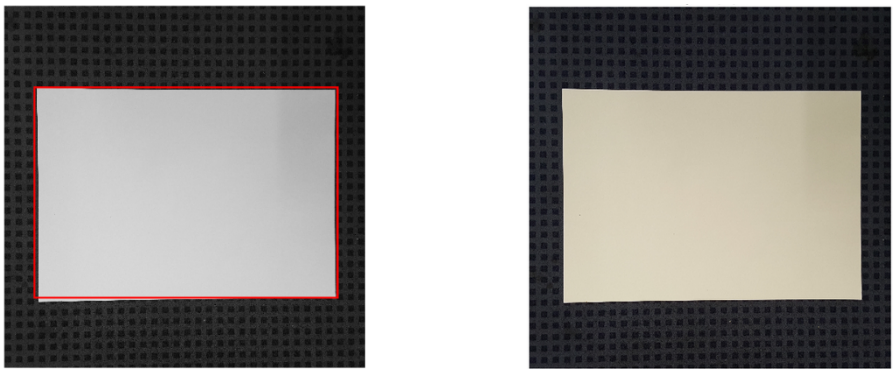
شکل ۲: تعدادی از فیلتر های معروف

در بسیاری از مواقع اندازه تصویری که در اختیار داریم مطلوب ما نیست، اگر تصویری با اندازه کوچک نیاز داشته باشیم مشکل چندانی وجود ندارد و به سادگی می‌توان از downsampling استفاده کرد، اما در صورتی که نیاز به بزرگنمایی یک تصویر داشته باشیم انتخاب های متفاوتی برای interpolation kernel خواهیم داشت.

تصویر kobe.jpeg را بارگیری کنید و با استفاده از دستور imresize() اندازه آن را با اسکیل $\frac{1}{5}$ تغییر دهید. حال مجددا تصویر را با همان دستور و اسکیل ۵ و method = nearest بازیابی کنید. علت تفاوت تصویر بازیابی شده و تصویر اصلی را توضیح دهید. سپس با استفاده از کرنل های گوسی و moving avg کیفیت تصویر را بهبود ببخشید و نتیجه را گزارش کنید.

در ادامه قصد داریم با استفاده از روشی که تا کنون آموختیم برنامه ای بنویسیم که لبه های یک کاغذ را از روی تصویر مشخص کند. عملکرد این برنامه همانند برنامه Camp Scanner است که لبه های کاغذ را پیدا کرده و با یک مستطیل رنگی آن را در تصویر مشخص می‌کند. برای پیدا کردن لبه های افقی می‌توانید از کرنل H line و برای شناسایی لبه های عمودی از کرنل V line استفاده کنید. همچنین برای رسم مستطیل می توانید از تابع rectangle() استفاده کنید. پیش نهاد می‌شود که پس از بار گیری تصویر ، آن را به حوزه gray scale ببرید ،زیرا رنگ هامختلف کمک چندانی در پیدا کردن لبه هانخواهند کرد. برنامه توصیف شده را بنویسید و صحت عملکرد آن را با استفاده از تصویر page.png بسنجید.

نکاتی در مورد گزارش : در قسمت های پیشین پس از آزمودن هر کرنل باید نتیجه را نمایش دهید و در گزارش کار خود اضافه کنید. برای نمایش تصویر می‌توانید از تابع imshow() استفاده کنید. همچنین دقت کنید که برای پیدا کردن لبه های کاغذ باید بنا بر خلاقیت و نظر خود الگوریتمی پیش نهاد دهید و استفاده از اعداد ثابت مجاز نیست.الگوریتم مورد نظر خود را در گزارش کار به صورت کامل توضیح دهید.



(آ) عکس اصلی (ب) لبه ها با قرمز مشخص شده‌است

شکل ۳: object localization

frequency domain filtering

قسمت دیگری از نگاشت های یاد شده در حوزه فرکانس اعمال می‌شوند. در این قسمت قصد داریم دامنه و فاز تصویر را در حوزه فرکانس بدست آوریم و با استفاده از noise analysis تاثیر نویز گوسی را برروی سیگنال بازیابی شده تعیین نماییم. به این منظور از مجموعه داده AT&T faces استفاده می‌کنیم. برای بدست آوردن اطلاعات بیشتری در مورد این مجموعه داده به اینجا مراجعه کنید.

شکل ۴: قسمتی از مجموعه داده AT&T faces

تصویر یکی از افراد را از مجموعه داده انتخاب کنید ، سپس اندازه و فاز تصویر را با استفاده از دستور mesh() رسم نمایید. برای بدست آوردن تبدیل فوریه تصویر می‌توانید از fft2() به همراه fftshift() استفاده کنید. حال جداگانه اندازه و فاز تبدیل فوریه را با نویز سفید گوسی مخشوش کنید. برای تولید نویز می‌توانید از دستور awgn() استفاده کنید. سپس قسمت های خواسته شده را برای $snr\epsilon\{۱, ۰/۵, ۰/۲۵\}$ انجام دهید

۱. وارون تبدیل فوریه را بر روی اندازه سالم و فاز مخشوش شده اعمال کنید ونتیجه را گزارش کنید.

۲. وارون تبدیل فوریه را بر روی اندازه مخشوش شده و فاز سالم اعمال کنید و نتیجه را گزارش کنید.

حال از مجوعه داده، دو فرد را انتخاب کنید و اندازه و فاز تبدیل فوریه هر یک را بدست آورید. در مرحله بازیابی سیگنال، فاز تبدیل فوریه تصاویر را تعویض کنید و نتیجه بدست آمده را گزارش کنید.

با توجه به نتایج بدست آمده کدام قسمت تبدیل فوریه حاوی اطلاعات بیشتری در مورد تصویر چهره می‌باشد؟

face recognition

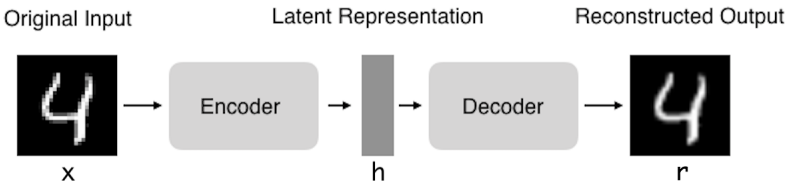
شناسایی افراد یکی از مهم ترین کاربرد های پردازش تصویر تلقی می‌شود. این کار چالش های فراوانی به‌همراه دارد، برای مثال یک سیستم تشخیص چهره باید چهره را از زوایای متفاوت شناسایی کند، باید بتواند صورت را به همراه پوشش های متفاوت مانند کلاه یا عینک تشخیص دهد و مواردی از این قبیل. در این قسمت می‌خواهیم نوع ساده‌ای از یک سیستم تشخیص چهره را پیاده سازی کنیم.

ابتدا تصاویر مجموعه داده faces را به دو دسته train و test با نسبت ٪۸۰ به ٪۲۰ تقسیم کنید و ادامه توضیحات را برای قسمت train انجام دهید.

در ابتدا باید مشکل بعد بالای تصویر را حل کنیم، زیرا به نظر نمی‌رسدکه مقدار هر پیکسل برای تشخیص چهره حیاتی باشد. برای اثبات این ادعا می‌توان به این نکته اشاره کرد که اگر از تصویر نمونه برداری شود شما همچنان قادر به تمییز افراد خواهید بود. در اصطلاح فضای ویژگی در این مسئله دارای پوچی است و بعد ویژگی حیاتی برای این مسئله کم‌تر از تعداد پیکسل های تصویر می‌باشد. برای حل این مشکل از ابزار های استخراج ویژگی همانند autoencoders استفاده می‌شود. با استفاده از تابع trainAutoencoder(data) یک auto encoder طراحی کنید و پارامتر های hiddensize = 100 و MaxEpochs = 1000 برای یکسانی نتایج به مدل ورودی بدهید.

برای ارزیابی نحوه عملکرد این مدل ، به انتخاب خودتان فردی از مجموعه داده انتخاب کنید و تصویر بازیابی شده آن را بهمراه تصویر اصلی نمایش دهید. بسته به سخت افزار مورد استفاده، آموزش auto encoder ممکن است زمان زیادی نیاز داشته باشد.

پس از آن که عملکرد این مدل از نظر بازیابی مناسب بود ، با استفاده از تابع encode() بازنمایی تصاویر را در بعد پایین‌تر (۱۰۰) بدست آورید. در ادامه برای شناسایی افراد از این داده ها استفاده می‌کنیم. برای بدست اوردن اطلاعات بیشتر در مورد auto encoders به اینجا مراجعه کنید.



شکل ۵: نحوه عملکرد auto encoder

حال که بعد داده ها را کاهش دادیم، با استفاده از یک شبکه عصبی تک لایه افراد را از یک دیگر جدا می‌کنیم. برای این که یک شبکه عصبی آموزش داده شود باید برای هر دسته (فرد) یک target معین کنیم، این کار را با استفاده از one-hot coding انجام دهید. سپس با استفاده از تابع trainSoftmaxLayer(encoded-data , targets) یک لایه softmax آموزش دهید تا قسمت طبقه بندی را انجام دهید. در مرحله قبل، بعد خروجی لایه softmax با توجه به نحوه کد کردن target ها برابر با تعداد کلاس‌ها (۴۰) خواهد بود.

برای این که بتوان از auto encoder به همراه softmax layer استفاده کرد می‌توان از تابع stack(net1 , net2) استفاده کرد.

تابع هدف شبکه عصبی فوق به گونه‌ای است که پس از پایان آموزش تلاش می‌کند خروجی شبکه عصبی برای داده های یک دسته (فرد) تا جای ممکن به target معین شده نزدیک باشد، بنابر این برای تشخیص دسته (فرد) از رابطه زیر استفاده می‌کنیم که در آن y خروجی شبکه برای داده جدید است و t_i ها target های از پیش تعیین شده هستند/

$$c^* = argmax_{i=۱:۴۰} ||y - t_i||_۲$$

در این مرحله ساخت و آموزش مدل پایان یافته است و نوبت به ارزیابی عمکرد مدل می‌رسد. برای این کار از دقت مدل استفاده می کنیم.

$$\text{acc} = \frac{\text{correct classified}}{\text{all instances}}$$

دقت مدل را برای داده های train و test بدست آورید و گزارش کنید. حداقل دقت قابل قبول روی داده های test برابر ۸۵ است.

موفق باشید