فصل ۸

پردازش تصویر

اهداف این جلسه

- آشنایی با دوربین های تحت شبکه و دریافت اطلاعات به صورت برخط
- پیش پردازش تصویر و افزایش آن

در پایانِ این جلسه شما قادر خواهید بود که :

- عملیات کانولوشن را بر روی تصاویر انجام دهید
- اطلاعات را از سرور RTSP دریافت و ذخیره کنید
- تجزیه و تحلیل مدل های پردازش تصویر با استفاده از ابزار PyTorch

۱.۸ آماده‌سازی

ابتدا نیاز است که دوربین تحت شبکه را به شبکه متصل کرد تا بتوان با استفاده از پروتوکل RTSP تصاویر را با استفاده از Codec مورد نظر ذخیره کرد.
با در دست داشتن آدرس `rtsp://admin:12345@172.21.36.10/MediaInput/h264` فریم های برخط را با استفاده از توابع کتابخانه OpenCV دریافت کنید. می‌توانید با گذاشتن یک وقفه مناسب تغییرات دریافت شده را با فاصله‌های زمانی ملموس دریافت کنید. برای پردازش این تصاویر در قسمت های بعدی این تصاویر را در پوشه Frames قرار دهید.

تمرین اول: داده افزایی عکس

با استفاده از یکسری تغییرات مانند Rotation می‌توان از روی دیتاست داده شده تعدادی عکس ایجاد کرد.(دقت داشته که این عملیات را با استفاده از آرایه های Numpy بدون استفاده از توابع آماده انجام دهید.)

۱.۱.۸ چرخش تصاویر

با استفاده از کتابخانه PIL عکس ها را بخوانید و با دردست داشت آرایه‌های مورد نظر عکس ها را ۱۸۰ و ۹۰ درجه پادساعتگرد بچرخانید.

۲.۱.۸ بازتاب تصاویر

با استفاده از کتابخانه‌های ذکر شده عملیات Horizontal Flip را انجام دهید.

۳.۱.۸ تغییر روشنایی تصویر

با استفاده از متغیر مقیاس factor سعی کنید که میزان روشنایی یک عکس را تغییر دهید.(توجه داشته باشید که مقدار هر پیکسل باید بین ۰ تا ۲۵۵ باشد.)

تمرین دوم: Convolution

فرض کنید آرایه $f(i,j)$ تصویر ورودی ما است. و آرایه $h(i,j)$ را به عنوان کرنل ذر نظر میگیریم، و سعی می‌کنیم عملیات کانولوشن را بر روی آرایه ورودی انجام می‌شود. با تکمیل کردن تابع $Convolution(image, kernel)$ عملیات خواسته شده را انجام دهید:

$$Output[i,j] = \sum_{m=1}^{M}\sum_{n=1}^{N} f[m,n]h[i-m, j-n] \tag{۱.۸}$$

۴.۱.۸ تشخیص لبه ها

با استفاده از فیلتر Zobel و اعمال آن بر روی تصویر دریافت شده سعی کنید لبه ها را مشخص کنید.

۵.۱.۸ تغییر اندازه تصاویر

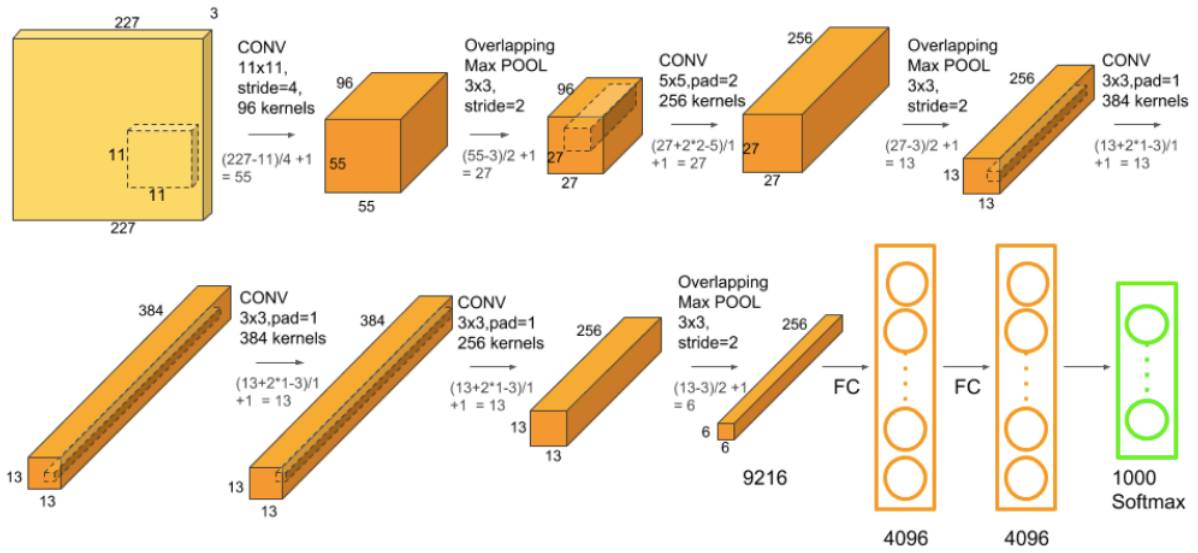
با استفاده از ابزار PyTorch و توابع آماده سعی کنید که تصویر ورودی را به اندازه $(227, 227)$ تغییر دهید.

تمرین سوم: استفاده از شبکه AlexNet

۲.۸ AlexNet

با توجه به اینکه AlexNet یک شبکه عصبی عمیق است که برای دسته بندی و تشخیص تصاویر به کار می‌رود. این شبکه عمیق تشکیل شده از ۸ لایه است : ۵ لایه کانولوشنی (برخی از آن‌ها داری لایه max-pooling است)، ۲ لایه متصل پنهان، و یک لایه متصل در قسمت آخر برای تعیین کلاس خروجی است.

در لایه اول از یک پنجره کانولوشن با سایز ۱۱×۱۱ استفاده شده است. به این دلیل است که اندازه ورودی بزرگ است، بنابراین باید از یک هسته بزرگ برای گرفتن شی استفاده کنیم. شکل پنجره کانولوشن در لایه های بعدی به تدریج به 5x5 و 3x3 کاهش می یابد، اما تعداد فیلترها به موازات آن افزایش می یابد. دو لایه کانولوشن اول و آخر با لایه‌های max-pooling دنبال می‌شوند که در آن یک پنجره جمع‌آوری به اندازه 3x3 و یک گام از ۲ مرحله اعمال می‌شود. از این رو، اندازه خروجی در سراسر این لایه های ادغام به نصف کاهش می یابد.



شکل ۱.۸: ساختار AlexNet

۱.۲.۸ استخراج یک لایه آموزش دیده

در این قسمت ابتدا برای اینکه اندازه تصویر متناسب با اندازه ورودی این شبکه باشد نیاز است که تصویر کوچک شده در قسمت قبل را به عنوان تصویر ورودی در نظر گرفت.

لایه های نخستین این شبکه وظیفه استخراج ویژگی های سطح پایین را دارد مانند تشخیص لبه‌ها و گوشه‌ها را دارد.

با استفاده از ابزار PyTorch ابتدا این مدل را به صورت آموزش دیده دریافت کنید. سپس لایه اول کانولوشن را جدا کرده و بر روی تصویر ورودی با استفاده از تابع Convolution اجرا کنید و مقادیر ساخته شده را در آرایه Result قرار بدهید. سپس با استفاده از تابع `plot_images` که در فایل کمکی `helpers.py` قرار گرفته است، عکس های کانوالو شده را در خروجی مشاهده کنید.

در عکس‌های خروجی چه چیزی مشخص شده است؟ عملکرد آن را با فیلتر Zobel مقایسه کنید.

تمرین چهارم: اجرای دسته بند بر روی عکس های موجود

هدف استفاده از شبکه AlexNet این است که بتوانیم کلاس عکس های موجود را تشخیص دهیم. برای این کار ابتدا تمامی عکس‌های بدست آمده از دوربین نظارتی که در پوشه `Frames` ذخیره شده است را با انجام پیش پردازش در یک batch ذخیره کنید. سپس این آرایه را به صورت ورودی به مدل AlexNet بدهید. در آخر هم با استفاده از کلاس های مشخص شده نام کلاس هدف را پیشبینی کنید.