عنوان : تمرین ششم یادگیری ماشین (CNN)

نگارنده : سحر داستانی اوغانی

شماره دانشجویی : ۹۹۱۱۲۱۰۸

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دانشکده ریاضی و علوم کامپیوتر

پاسخ قسمت تشریحی

۱) چهار مدل ژرف معروف که از پیش آموزش دیده‌اند را نام ببرید. هر یک برای چه نوع داده‌ی ورودی مناسب است.

مدل‌های از پیش آموزش دیده، منبع خوبی برای افرادی است که به دنبال یادگیری یک الگوریتم یا اجرای یک فریم ورک هستند. با توجه به محدودیت‌های زمانی و محاسباتی، همواره ساختن یک مدل از ابتدا ممکن نیست. بنابراین افراد می‌توانند از مدل‌های از پیش آموزش دیده به عنوان معیاری برای تقویت یا آزمایش مدل خود، استفاده کنند. لازم به ذکر است که از شبکه‌های آموزش دیده، می‌توان در مواردی از جمله‌ی classification, feature extraction, transfer learning استفاده کرد.

در زیر به بررسی برخی از این مدل‌های می‌پردازیم و ویژگی هر کدام را بیان می‌کنیم:

Mask R-CNN: یک فریم ورک انعطاف پذیر است که برای اهدافی مانند object instance segmentation بکار می‌رود. توانایی تشخیص ۸۰ کلاس مختلف را دارد. به دور هر شی در تصویر، یک باکس و یک قطعه‌ی جدا کننده مانند ماسک، در نظر می‌گیرد و با این روش آن را از سایر قسمت‌های تصویر، جدا می‌کند.

ورودی و تصاویر ارسالی به این مدل، محدودیتی از لحاظ اندازه و سایز ندارند.

YOLOv2: یک مدل CNN است که در زمان واقعی، قادر به تشخیص بیش از ۹۰۰۰ دسته‌بندی مختلف شی در تصویر است. این مدل ورودی را از نوع تصویر دریافت می‌کند و آن را به یک grid در اندازه‌ی $S \times S$ تقسیم می‌کند. هر سلول grid، فقط قادر به تشخیص یک شی است. در سایز و اندازه نیز محدودیتی ندارد.

برای مثال، grid زرد رنگ در تصویر مقابل، در تلاش برای تشخیص فرد است. که به صورت نقطه‌ی آبی رنگ در تصویر سمت راست نمایش داده شده است.

MobileNet: یک طراحی معماری برای دستگاه‌های موبایل است. این مدل قادر به تشخیص ۸۰ کلاس مختلف از شی است و حداکثر ۱۰ شی را در یک عکس پیدا می‌کند.

ورودی این مدل از نوع single shot detector(SSD) است و انواع تصاویر مختلف از صفحه‌ی موبایل را می‌پذیرد.

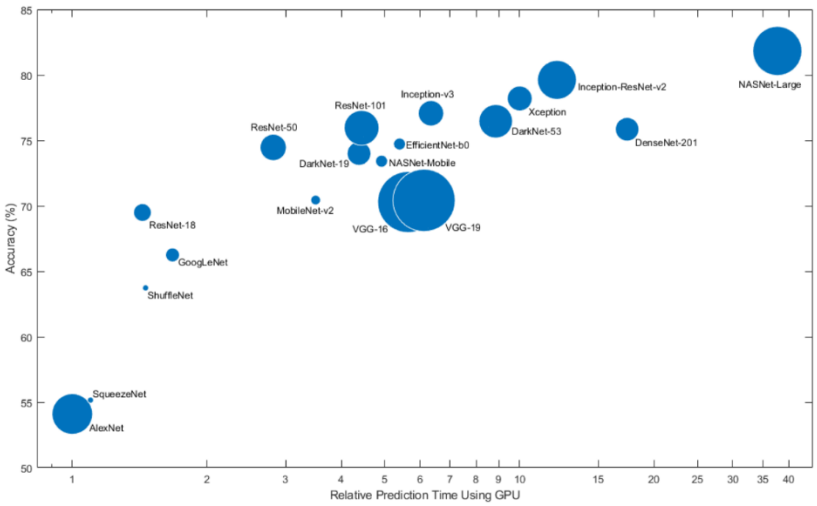
ArcFace: این مدل برای تشخیص حالت چهره‌ی انسان طراحی شده است. که ویژگی های افتراقی چهره را می آموزد و تعبیه هایی را برای ورود تصاویر چهره تولید می کند.

ورودی آن باید تصویری باشد که حتما شامل صورت انسان باشد تا بتواند آن را تشخیص دهد. در سایز و اندازه نیز محدودیتی ندارد.

AlexNet: مدلی است برای classification و object detection. این مدل قادر به تشخیص بیش از ۱۰۰۰ کلاس مختلف از اشیا است.

ورودی آن تصاویری است با اندازه‌ی 227×227.

در تصویر زیر می‌توان دقت لازم مدل‌های ژرف را با زمان لازم برای پیش‌بینی پاسخی توسط آن مدل‌ها سنجید.



۲) یک مدل (CNN شبکه عصبی کانولوشنال با یادگیری عمیق) را در نظر گرفته و سپس به سوالات زیر پاسخ دهید.

الف. ویژگی عمده این روش چیست و چه چیز موجب برتری و مقاومت بالاتر آن نسبت به سایر مدل های شبکه عصبی مانند MLP میشود؟

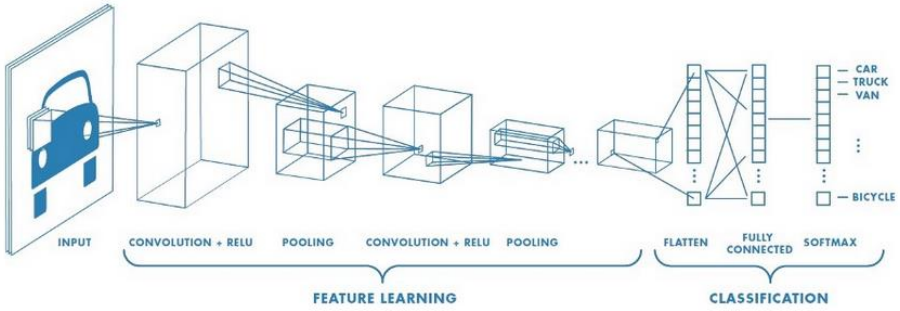
MLP(Multilayer Perceptron)ها برای هر ورودی(برای مثال: هر پیکسل یک عکس)، یک پرسپترون استفاده می‌کنند در این صورت، به هر ورودی یک سری وزن اختصاص داده می‌شود که به دلیل زیاد بودن تعداد پیکسل‌ها در هر عکس، مقادیر وزن به صورت نمایی زیاد شده و از یک حدی به بعد غیرقابل کنترل می‌گردنند. از طرفی، به دلیل fully connected بودن شبکه، پارامترهای زیادی برای هر عکس تولید می‌شود. زیرا هر نورون کاملا به نورون‌های لایه‌ی بعدی و قبلی خود متصل است. این عملیات و زیاد بودن وزن‌ها و پارامترهای شبکه، عملا شبکه را ناکارامد کرده است.

یکی دیگر از مشکلات MLPها برخورد متفاوت آن‌ها نسبت به هر ورودی است. برای مثال اگر تصویر یک گربه در یک عکس یکبار در قسمت بالایی و سمت چپ تصویر قرار گیرد و بار دیگر در سمت پایین و سمت راست تصویر قرار گیرد، MLP سعی می‌کند فرضیه خود را درست کند و همواره گربه را در قسمت ذکر شده تصور کند.

پس بزرگترین مشکل MLPها هنگامی رخ می‌دهد که می‌خواهیم یک عکس را flattened کنیم یعنی آن را از حالت ماتریسی به حالت برداری تبدیل کنیم. در این صورت، حجم وسیعی از اطلاعات فضایی گم می‌شود.

راهکار مشکلات بالا استفاده از مدلی است که ویژگی‌های منفی ذکر شده را نداشته باشد و راه حل مناسبی برای آن‌ها ارائه دهد. این مدل، شبکه عصبی کانولوشن (CNN) نام دارد. CNN با داده‌هایی که دارای رابطه‌های فضایی با یک‌دیگر هستند، بهتر کار می‌کند. برتری CNN در توانایی آن در ایجاد یک نمایش داخلی از یک تصویر دو بعدی است. این ویژگی به مدل اجازه می‌دهد که مکان‌ها و مقیاس‌ها را در ساختارهای متفاوت داده‌ها یاد گیرد. این خود یک ویژگی کاراست، زمانی که با داده‌های تصویری کار می‌کنیم.

CNN قادر به دسته‌بندی داده‌ها در ابعاد بسیار بالاست و این کار را با استفاده از استخراج ویژگی‌های مهم تصویر انجام می‌دهد. CNN این ویژگی‌ها را به عنوان ورودی به شبکه می‌دهد و از این طریق بعد و وزن‌های بی‌شمار در مدل‌های قبلی را کاهش می‌دهد. CNN هر تصویر ورودی را از یک سری لایه‌های کانولوشن عبور می‌دهد. در زیر تصویری از یک شبکه عصبی با لایه‌های کانولوشن را مشاهده می‌کنید.



در این شبکه، از ورودی‌های یک سری ویژگی استخراج شده است که این ویژگی‌ها برای learning مورد استفاده قرار گرفتند. بنابراین از تمامی نورون‌های ورودی استفاده نشده است و تنها از ویژگی‌های استخراج شده برای یادگیری استفاده شده است. همین امر سبب کاهش اتصالات اضافی و کم شدن وزن‌ها و پارامترهایی می‌شود که در MLPها مشکل ایجاد می‌کردند.

برای بررسی دلیل مقاوم بودن CNNها لطفا به تصویر بالا توجه فرمایید. این مدل‌ها از لایه‌های مختلف کانولوشنی و pooling تشکیل شده‌اند. لایه‌های pooling خود باعث تقویت مدل می‌گردند زیرا وظیفه‌ی کاهش تعداد پارامترها(وقتی که اندازه‌ی تصویر بزرگ است) را بر عهده دارند. این لایه‌ها به subsampling یا downsampling نیز معروفند که وظیفه‌ی کاهش بعد و نگهداری اطلاعات مهم را بر عهده دارند. انواع مختلف subsamplingها به شرح زیر است:

- Max Pooling

- Average Pooling

- Sum Pooling

بر اساس نوع آن‌ها خرورجی‌های متفاوتی را دریافت می‌کنیم. برای مثال، Max Pooling، ماکسیمم مقدار subsampleهای خود را به عنوان خروجی می‌دهد.

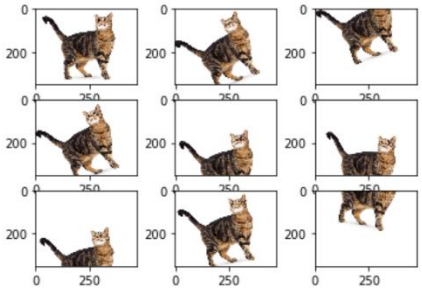
ب. چهار چالشی که این مدل بطور ویژه در قبال آن ایجاد مقاومت میکند چیست؟ این مقاومت چگونه در مدل ایجاد میشود؟

۱. درمقابل نویز مقاوم هست. زیرا subsampling تنها برخی از featureها را دریافت می‌کند و ورودی‌هایی مانند نویز یا امثال آن را نادیده می‌گیرد.

۲. در مقابل screwing مقاوم هست.

۳. در مقابل scaling مقاوم است. مدل CNN سعی می‌کند از تصویری که به عنوان ورودی به آن داده شده است، تصاویر دیگری با اندازه‌های کوچکتر بسازد. این کار را با استفاده از image augmentation انجام می‌دهد. به همین دلیل، از این به بعد می‌تواند با اندازه‌های کوچکتر تصویر اولیه ولی در مکان‌های مختلف آن کار کند.

در مقابل تصویری را مشاهده می‌کنید که توسط image augmentation از تصویر اولیه در اندازه‌ها و طرق‌های گوناگون ساخته شده است.



۴. در مقابل displacement مقاوم است. زیرا با پیمایش و جابه‌جایی یک شی در تصویر، لایه‌های Pooling به کار می‌افتند و این موضوع را قابل فهم می‌کنند.

۵. اورفیت در مدل کم است و تعداد داده‌ها روی آن تاثیری ندارد.

۶. در مواردی که عکس blurred یا shadowy باشد، استخراج ویژگی از عکس به خوبی انجام می‌گیرد و در مقابل مواردی که جلوگیری می‌کنند از این اتفاق، مقاوم است..

ج. این مدل در حالت استاندارد دارای چند استیج و چند نوع لایه است؟ عملکرد هر کدام را به تفصیل شرح دهید.

مدل CNN در حالت استاندارد دارای دو استیج می‌باشد که به شرح زیر است. در هر استیج، لایه‌های آن نیز مورد بررسی قرار گرفته است.

۱. convolution: این استیج وظیفه‌ی نمایش ویژگی‌های تصاویر ورودی را بر عهده دارد که خود از دو لایه تشکیل شده است.

- کانولوشن:

می‌دانیم، تصاویر ورودی در ابتدا به صورت ماتریسی هستند و باید با استفاده از عمل flattened به بردار تبدیل شوند. در این لایه هر کدام از نورون‌ها روی محدوده‌ی کوچکی از ورودی‌ها عمل کانولو را انجام می‌دهند. این محدوده را حوزه‌ی دریافت نورون یا receptive field می‌نامند. در تمامی این محدوده، وزن‌ها برای نورون‌ها، ثابت و مشابه است. زیرا عمل کانولوشن نوعی فیلتر کردن است و مسلما این فیلتر باید برای تمام تصاویر ثابت باشد. همین محدوده باعث می‌شود نورون‌ها به تمامی تصاویر متصل نباشند و در نتیجه تعداد پارامترها، وزن‌ها و محاسبات را کم می‌کند و مشکلی که در MLPها برای ورودی‌های بزرگ داشتیم را دیگر به وجود نمی‌آورد. مجموعه وزن ها، کرنل فیلتر را تشکیل می‌دهند. پس از اینکه هر کرنل را بر روی هر تصویر پیاده کردیم، یک نقشه ویژگی یا feature map ایجاد می‌شود. اما توجه داشته باشید که گاها بر روی یک تصویر بیش‌تر از یک فیلتر اعمال می‌کنیم و در نتیجه خروجی مجموعه ای از نقشه های ویژگی می‌شود. در نتیجه در نهایت به اندازه‌ی تعداد فیلترها، مجموعه‌ی وزن خواهیم داشت.

- اعمال تابع غیرخطی (nonlinearity) بر روی نتیجهء کانولوشن:

زمانی که یک تابع غیرخطی را بر روی نتایج مرحله‌ی قبل پیاده می‌کنیم، یادگیری سریع‌تر رخ می‌دهد. از طرفی این عمل باعث می‌شود که زمان اجرای الگوریتم back-ptopagation کمتر گردد و انجام آن ساده‌تر شود. خروجی این لایه، خروجی لایهء اول یعنی کانولوشنال را می‌دهد.

۲. spatial pooling:

- زیر نمونه برداری (subsampling)

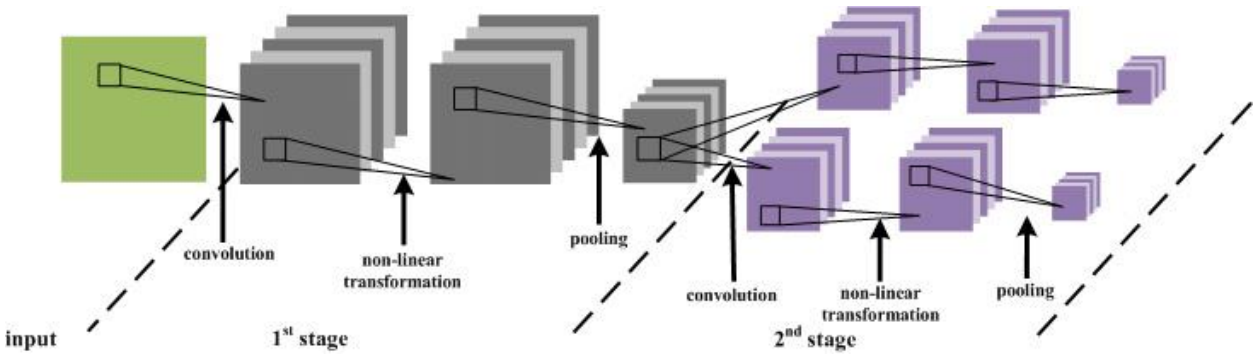
خروجی استیج قبل، ورودی این مرحله را تشکیل می‌دهد. این لایه نسبت به انتقالات جزئی، مقاومت ایجاد می‌کند. به دلیل بالا بودن حجم داده‌ها، یکی از اهداف مهم این بخش، کاهش آن حجم است. زیرا تعداد feature mapsها به صورت باورنکردنی‌ای بالاست و اندازه‌ی هر کدام، تقریبی از اندازه‌ی تصویر اصلی می‌باشد.. بنابراین شبکه به دنبال کاهش بعد و حذف اطلاعات اضافی است تا بتواند قوی‌ترین پاسخ را حفظ کند.

برای محقق ساختن اهداف بالا لازم است تعداد نقاط داخل receptive field را کم کنیم و برای این‌کار می‌توانیم از عملیاتی مانند گرفتن Max، Average یا Sum استفاده کنیم. تمامی فیلترهای ذکر شده، پاسخ مناسبی را ارائه می‌دهند ولی فیلتر Max، عموما جواب بهتری را به نمایش می‌گذارد. دلیل آن نیز واضح است، زیرا مشخص می‌کند که کدام پاسخ در محدوده، از همه قوی‌تر است.

باید توجه داشت، که ابعاد تصویر ورودی نمی‌توانند ابعادی دلخواه باشند زیرا برای مثال فرض کنید فیلتر 2×2 باشد و میزان جابه‌جایی نیز 2 در نظر گرفته شود، در این صورت اندازه‌ی هر بعد تصویر ورودی، نصف خواهد شد. البته جابه‌جایی فیلتر، می‌تواند به نحوی باشد که نواحی پذیرنده یا دارای همپوشانی یا فقد آن باشند. اما در مجموع، به نظر می‌آید که همپوشانی نواحی، هرچند که باعث بزرگ شدن فضای مسئله می‌شود، اما در بهبودبخشی نتیجه تاثیر به‌سزایی دارد.

- نرمال سازی (normilizing)

این مرحله می‌تواند قبل از مرحلهء subsampling رخ دهد. این مرحله می‌تواند بهبودی بر روی ویژگی‌های استخراج شده داشته باشد و جزئیات را بیشتر و بهتر نمایش دهد.



د. یکی از مهم ترین ویژگی های این مدل feature abstraction است. نحوهء عملکرد، کاربرد و عملکرد آن را شرح دهید.

در مدل‌های CNN به دنبال استخراج یک سلسله مراتب از ویژگی‌ها و چینش آن‌ها در لایه‌های مختلف هستیم. این امر باعث ایجاد عمق در شبکه می‌گردد.

در شبکه‌های عمیق، ۲ عمل اصلی انجام می‌گیرد: ۱. feature abstraction ۲. Classification

عمل اول، یعنی استخراج ویژگی، به دنبال این است که ویژگی‌های تصویر را به صورت لایه‌ای درآورد. این عمل در چند مرحله صورت می‌گیرد.

ابتدا مدل تصمیم می‌گیرد که چه ویژگی‌هایی در چه مراحلی استخراج شوند.

سپس عملیات تغییرات و مقاوم سازی را بر روی این ویژگی‌ها، داده‌های ورودی، وزن‌ها، نواحی پذیرنده، توابع و ... انجام می‌دهد.

استخراج ویژگی یا feature abstraction یا feature map دارای n تا stage می‌باشد که هر کدام متشکل از دو لایه‌ی مختلف هستند. ویژگی‌های استخراجی توسط feature abstraction، در ابتدا ساده هستند و به مرور پیچیده‌تر می‌شوند. در واقع مدل قادر به ارائه‌ی یک معماری ویژه است که در آن می‌توان یک چیدمان سلسله مراتبی از ویژگی‌ها را در لایه‌های عمقی مشاهده کرد.

روش کار در این بخش به این صورت است که تصاویر در ابتدا گرفته می‌شوند و به داخل شبکه‌ی کانولوشن فرستاده می‌شوند. در این شبکه فیلترهای مختلفی در لایه‌های متفات و به صورت موازی کار گذاشته شده است که باعث استخراج ویژگی می‌شوند. در مرحله‌ی بعد یعنی pooling، ویژگی‌های استخراج شده در مرحله‌ی قبل گرفته می‌شود و کاهش بعد صورت می‌گیرد.

به این فرایند که طی آن با عبور از لایه‌های مختلف به مرور از ویژگی‌های ساده‌تر به ویژگی‌های پیچیده‌تر می‌رسیم feature abstraction گویند.

ه. دلیل بکارگیری توابع غیرخطی در این مدل چیست؟ حداقل سه دلیل را نام برده و توضیح دهید.

زمانی که یک تابع غیرخطی را بر روی مدل پیاده می‌کنیم:

- یادگیری سریع‌تر رخ می‌دهد.
- از طرفی این عمل باعث می‌شود که زمان اجرای الگوریتم back-ptopagation کمتر گردد و انجام آن ساده‌تر شود.
- و همچنین از ورود نورون ها به نواحی اشباع جلوگیری می‌کند.

و. ورودی و خروجی لایه pooling چیست؟ این لایه چه تاثیری بر ورودی ها دارد و چگونه ورودی را به خروجی تبدیل میکند؟ توضیح دهید.

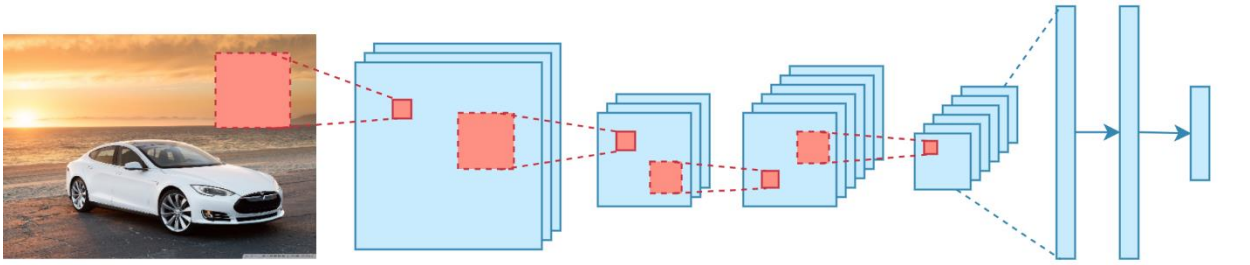
خروجی لایه‌ی کانولوشن، ورودی این مرحله را تشکیل می‌دهد. این لایه نسبت به انتقالات جزئی، مقاومت ایجاد می‌کند. به دلیل بالا بودن حجم داده‌ها، یکی از اهداف مهم این بخش، کاهش آن حجم است. زیرا تعداد feature mapsها به صورت باورنکردنی-ای بالاست و اندازه‌ی هر کدام، تقریبی از اندازه‌ی تصویر اصلی می‌باشد.. بنابراین شبکه به دنبال کاهش بعد و حذف اطلاعات اضافی است تا بتواند قوی‌ترین پاسخ را حفظ کند.

برای محقق ساختن اهداف بالا لازم است تعداد نقاط داخل receptive field را کم کنیم و برای این‌کار می‌توانیم از عملیاتی مانند گرفتن Max، Average یا Sum استفاده کنیم. تمامی فیلترهای ذکر شده، پاسخ مناسبی را ارائه می‌دهند ولی فیلتر Max، عموما جواب بهتری را به نمایش می‌گذارد. دلیل آن نیز واضح است، زیرا مشخص می‌کند که کدام پاسخ در محدوده، از همه قوی‌تر است.

باید توجه داشت، که ابعاد تصویر ورودی نمی‌توانند ابعادی دلخواه باشند زیرا برای مثال فرض کنید فیلتر 2×2 باشد و میزان جابه‌جایی نیز 2 در نظر گرفته شود، در این صورت اندازه‌ی هر بعد تصویر ورودی، نصف خواهد شد. البته جابه‌جایی فیلتر، می‌تواند به نحوی باشد که نواحی پذیرنده یا دارای همپوشانی یا فقد آن باشند. اما در مجموع، به نظر می‌آید که همپوشانی نواحی، هرچند که باعث بزرگ شدن فضای مسئله می‌شود، اما در بهبودبخشی نتیجه تاثیر بسزایی دارد.

۳) عملکرد کلی یک شبکه CNN را به طور کامل و دقیق شرح دهید. (حداکثر ده صفحه)

تمامی مدل‌های cnn از معماری‌ای که در تصویر زیر آمده است پیروی می‌کنند:



در واقع ورودی یک تصویر است. مدل نیز از لایه‌های مختلفی تشکیل شده است. این لایه مجموعه‌ای از لایه‌های convolution و pooling است که پس از آن تعدادی از لایه‌های fully connected قرار دارد.

Feature extraction:

۱) Convolution، بلاک اصلی cnn است. این لایه عملیات ریاضی را برای ادغام ۲ مجموعه از اطلاعات انجام می‌دهد. در این لایه یک فیلتر convolution روی داده‌ی ورودی اعمال می‌شود و یک feature map تولید می‌گردد.

در سمت چپ، تصویر ورودی قرار دارد و در سمت راست، فیلتر کانولوشن یا کرنل قرار دارد که دارای اندازه‌ی 3×3 است.

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

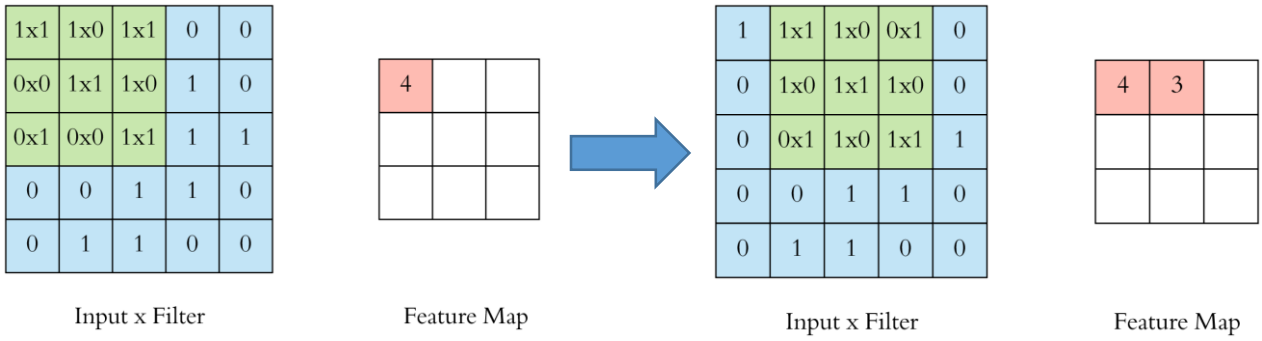
Input

1	0	1
0	1	0
1	0	1

Filter / Kernel

عملیات کانولو کردن با حرکت فیلتر بر روی ورودی اصلی صورت می‌گیرد. در هر مکان، هر خانه‌ی کرنل در هر خانه‌ی ورودی ضرب می‌شود و در انتها مجموع خروجی‌ها در feature map، نمایش داده می‌شود.

ناحیه‌ی سبز رنگ که محل قرارگیری کرنل را در این لحظه بر روی ورودی نمایش می‌دهد، receptive field نام دارد و اندازه‌ی آن برابر با اندازه‌ی کرنل است.



Input x Filter | Feature Map | Input x Filter | Feature Map

برای درک بهتر، می‌توانید عملیات کامل را در این گیف مشاهده کنید.

توضیحات بالا، یک کانولوشن دو بعدی را نمایش می‌داد. ولی در عمل و در واقعیت، تصاویر نمایش دهنده‌ی دنیای ۳ بعدی هستند و برای آن‌ها یک کانولوشن ۳ بعدی ایتفاده می‌شود که بعد سوم، رنگ (RGB) را نمایش می‌دهد.

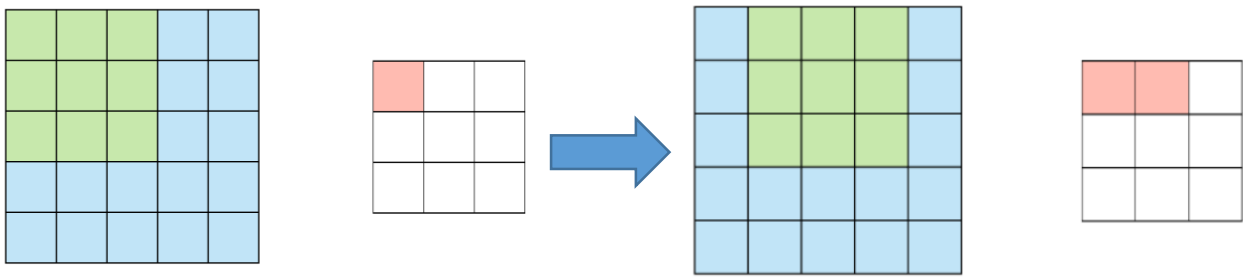
۲) تابع غیر خطی:

هر شبکه‌ی عصبی برای قدرتمند بودن نیازمند توابع غیرخطی است.

زمانی که یک تابع غیرخطی را بر روی نتایج مرحله‌ی قبل پیاده می‌کنیم، یادگیری سریع‌تر رخ می‌دهد. از طرفی این عمل باعث می‌شود که زمان اجرای الگوریتم back-ptopagation کمتر گردد و انجام آن ساده‌تر شود. خروجی این لایه، خروجی لایهء اول یعنی کانولوشنال را می‌دهد.

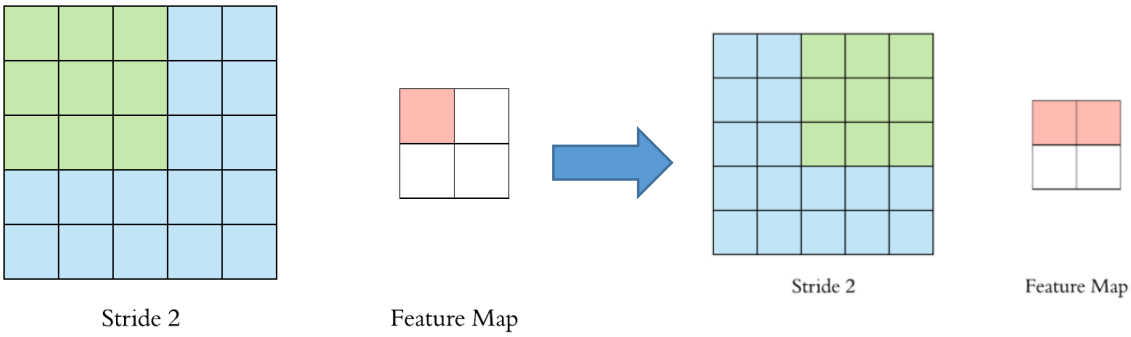
۳) stride and padding

stride: پنجره‌ی filter به فاصله‌های خاصی روی ورودی حرکت می‌کند. این فاصله را stride تعین می‌کند. این فاصله به صورت پیش‌فرض ۱ در نظر گرفته می‌شود.



Stride 1 | Feature Map | Stride 1 | Feature Map

برای همپوشانی کمتر بین receptive fieldها، نیازمند strideهای بزرگتری هستیم. اینکار، خروجی feature map را کوچکتر می‌کند. زیرا از بررسی بسیاری از مکان‌ها صرف نظر (RGB) می‌شود.



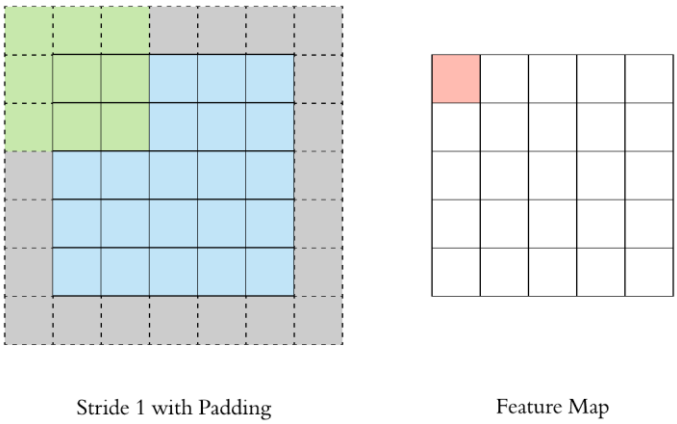
اندازه‌ی feature map، می‌تواند کوچکتر از ورودی باشد زیرا فیلتر کانولو باید بتواند داخل ورودی قرار گیرد و در آن موجود باشد.

حال اگر بخواهیم اندازه‌گیری را با بعد یکسان انجام دهیم، می‌توانیم از padding استفاده کنیم.

Padding دور تا دور ورودی را با صفر پر می‌کند.

ناحیه‌ی طوسی رنگ که دور ورودی را فراگرفته است، padding نام دارد. این ناحیه یا با صفر یا با مقادیر مرزی یال‌ها پر می‌شود.

حال بعد feature map با ورودی هماهنگ شد.



سایز feature map توسط ۳ پارامتر Depth، Stride، Padding مشخص می‌شود:

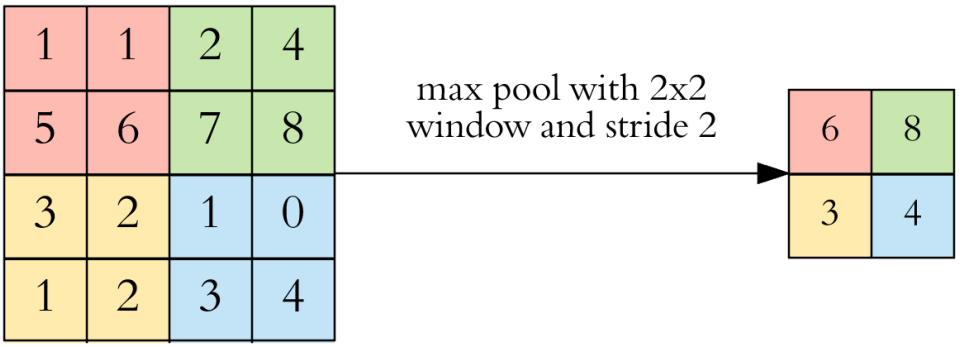
عمق مطابق تعداد فیلترهای استفاده شده برای کانولو کردن است.

۴) pooling:

پس از یک عمل کانولو ، معمولاً برای کاهش ابعاد pooling را انجام می شود. این کار به ما امکان می دهد تا تعداد پارامترها را کاهش دهیم، که هم باعث کاهش زمان آموزش می شود و هم با overfitting مقابله می‌کند. لایه های pooling به طور مستقل از هر نقشه ویژگی استفاده می کنند، ارتفاع و عرض را کاهش می دهند و عمق را دست نخورده نگه می دارند.

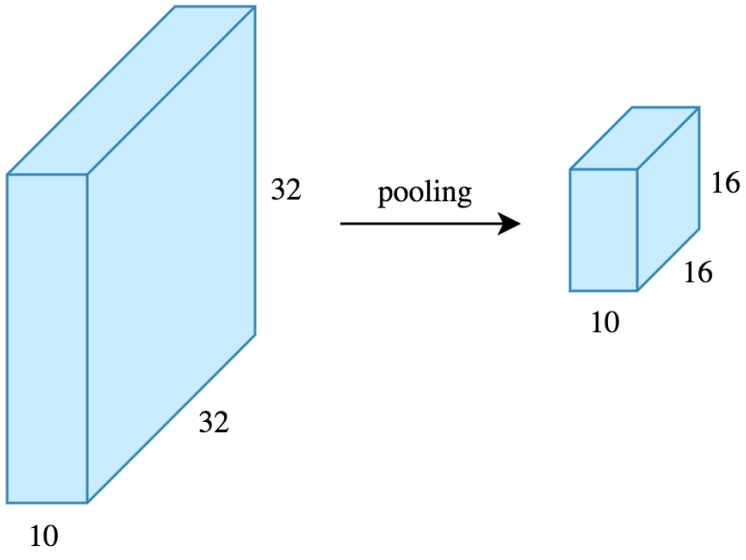
متداول ترین نوع pooling،max pooling است که فقط حداکثر مقدار را در پنجره pooling می گیرد. برخلاف عملیات کانولوشن ، جمع کردن هیچ پارامتری ندارد. این یک پنجره را روی ورودی خود قرار می دهد و به راحتی حداکثر مقدار را در پنجره می گیرد.

در اینجا نتیجه حداکثر تجمع با استفاده از یک پنجره 2×2 و stride ۲ است. هر رنگ نشان دهنده یک پنجره متفاوت است. از آنجا که هر دو اندازه و stride پنجره ۲ است ، پنجره ها با هم همپوشانی ندارند.



توجه داشته باشید که این پنجره و پیکربندی stride اندازه نقشه ویژگی را به نصف کاهش می دهد. این مورد اصلی استفاده از جمع کردن است ، در حالی که اطلاعات مهم را ذخیره می کنید ، از نقشه ویژگی نمونه برداری می کنید.

حال بیایید ابعاد نقشه ویژگی را قبل و بعد از pooling بررسی کنیم. اگر ورودی لایه pooling دارای ابعاد $32 \times 32 \times 10$ باشد ، با استفاده از همان پارامترهای pooling که در بالا توضیح داده شد ، نتیجه یک نقشه ویژگی $16 \times 16 \times 10$ خواهد بود. هر دو ارتفاع و عرض نقشه ویژگی به نصف کاهش می یابد ، اما عمق تغییر نمی کند زیرا pooling به طور مستقل بر روی هر عمق قطعه ورودی را کار می کند.



با نصف شدن ارتفاع و عرض ، تعداد وزن‌ها را به $\frac{1}{4}$ ورودی کاهش می دهیم. با توجه به اینکه ما در معماری CNN معمولاً با میلیون ها وزن سر و کار داریم ، این کاهش بسیار مهم است.

در معماری CNN ، pooling به طور معمول با پنجره های 2×2، stride ۲ و بدون padding انجام می شود. در حالی که کانولوشن با پنجره های 3×3، stride ۱ و padding انجام می‌شود.

۵) hyperparameter:

حال بیایید فقط یک لایه کانولوشن را نادیده بگیریم و از گزینه های hyperparameter که باید انتخاب کنیم ، استفاده کنیم.

ما ۴ ابر پارامتر مهم برای تصمیم گیری داریم:

اندازه فیلتر: ما معمولاً از فیلترهای 3×3استفاده می کنیم ، اما 5×5 و 7×7 نیز بسته به نوع کاربرد از آنها استفاده می شود. به یاد داشته باشید که این فیلترها سه بعدی هستند و دارای ابعاد عمق نیز هستند ، اما از آنجا که عمق یک فیلتر در یک لایه معادل با عمق ورودی آن برابر است ، ما آن را حذف می کنیم.

تعداد فیلتر: این متغیرترین پارامتر است ، در هر نقطه بین ۳۲ تا ۱۰۲۴ قدرت دارد. استفاده از فیلترهای بیشتر منجر به یک مدل قدرتمندتر می شود ، اما به دلیل افزایش تعداد پارامترها ، ما خطر overfiting را داریم. معمولاً در لایه های اولیه با تعداد کمی فیلتر شروع می کنیم و با ورود به عمق شبکه تعداد آنها را به تدریج افزایش می دهیم.

Stride: ما آن را در مقدار ۱ پیش فرض نگه می داریم.

Padding: ما معمولاً از padding استفاده می کنیم.

۶) fully connected:

بعد از لایه های کانولوشن + pooling ، دو لایه کاملاً متصل به هم اضافه می کنیم تا ساختار CNN را جمع بندی کنیم.

بخاطر داشته باشید که خروجی هر دو لایه کانولوشن و pooling ، حجم های سه بعدی هستند ، اما یک لایه کاملاً متصل انتظار یک بردار 1D از اعداد را دارد. بنابراین ما خروجی لایه نهایی جمع شدن را به یک بردار مسطح می کنیم و این ورودی لایه کاملاً متصل می شود. مسطح کردن به سادگی ترتیب دادن حجم سه بعدی اعداد به یک بردار ۱ بعدی است ، در اینجا هیچ اتفاقی رخ نمی دهد.

۷)training:

CNN همانند ANN آموزش داده می شود ، backpropagation with gradient descent.

گزارش پیاده‌سازی

آماده‌سازی دیتاست

فراخوانی و تغییر شکل دیتاست:

می‌دانیم دیتاست MNIST تصویر مجموعه‌ای از اعداد ۰ تا ۹ است که توسط افراد مختف نوشته شده است. هرکدام از این تصاویر دارای 28×28 پیکسل هستند به‌طوری که رنگ پیکسل‌ها در بازه‌ای از رنگ‌های خاکستری قرار دارند که میان رنگ سفید و مشکی تعریف می‌شوند. بنابراین برای راحتی کار و کار کردن با دیتاستی که تنها یک رنگ دارد، آن را reshape می‌کنیم. فرم دیتاست به صورت مقابل است. `(n_images, x_shape, y_shape, channels)`

کافی است میزان channels را ۱ قرار دهیم.(توجه داشته باشید که برای عکس‌هایی که دارای ۳ رنگ RGB هستند، می‌توان مقدار channels را ۳ قرار داد.)

پیش‌پردازش دیتاست:

– می‌دانیم بازه‌ی اعداد در تصاویر موجو در دیتاست از ۰ تا ۹ است. یعنی دیتاست دارای ۱۰ کلاس مختلف است. حال برای نمایش این کلاس‌ها می‌توان از on-hot-encoding استفاده کرد. این ویژگی تابعی به نام to_categorical دارد که برای هر کلاس، مقدار آن کلاس را ۱ و مقدار دیگر کلاس‌ها را صفر قرار می‌دهد.

– پیکسل‌ها در هر عکس، دارای مقداری بین ۰ تا ۲۵۵ هستند. این مقادیر درجه‌ی روشنایی آن‌ها از رنگ سفید تا مشکی را نمایش می‌دهد. برای نرمالایز کردن داده‌ها می‌توانیم آن‌ها را به بازه‌ی [0,1] ببریم. برای اینکار ابتدا باید اعداد صحیح را به اعشاری تبدیل کنیم و سپس، مقادیر هر پیکسل را بر بزرگترین مقدار یعنی ۲۵۵، تقسیم کنیم.

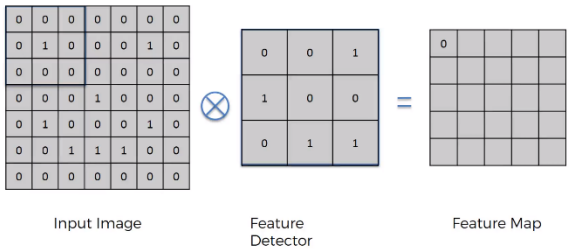
← می‌توان عملیات ذکر شده را به صورت روش اول که در پیاده‌سازی آمده است، پیاده کرد و یا به روش دوم. علت استفاده از روش دوم، این است که برنامه را می‌خواهیم به شکلی پیاده کنیم که مدل توسط واسط کاربری ارائه داده شود، بنابراین تعریف آن به صورت تابع، کار را راحت‌تر می‌کند و بعدا به راحتی می‌توان هرجایی از آن استفاده کرد.

طراحی مدل

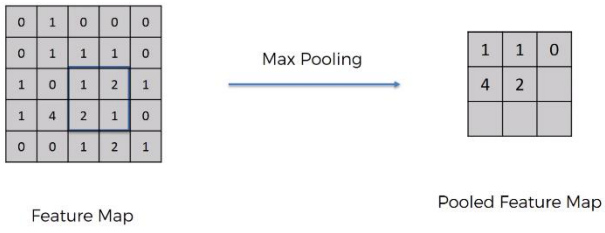
مدلی که طراحی می‌کنیم باید دارای دو جنبه‌ی اصلی باشد:

- feature extraction: که خود دارای دو لایه‌ی convolutional و pooling است.
- classifier: که عمل پیش‌بینی را انجام می‌دهد.

در لایه‌ی convolutional از یک feature map که دارای ۶۴ نورون است و یک feature detector با اندازه‌ی 3×3 برای ساخت feture map، استفاده می‌کنیم.



لایه‌ی pooling، pooled feture map را با حرکت دادن یک max pooling با اندازه‌ی 2×2 بر روی feture map، بدست می‌آورد.



تمامی لایه‌های از تابع RELU استفاده می‌کنند.

نرخ یادگیری را ۰,۰۱ و مقدار momentum را ۰,۹ قرار دادیم.

برای loss function از تابع categorical cross-entropy استفاده کردیم زیرا این تابع برای مسائل چند کلاسه خوب عمل می‌کند.

خروجی یک مساله‌ی طبقه‌بندی چند کلاسه، یک لایه با ۱۰ نود است. ولی دراینجا خروجی، احتمال توزیع یک عکس به هر یک از ۱۰ کلاس است. برای تغییر آن می‌توان از یک تابع softmax activation در لایه dense بین feature extraction و لایه-ی خروجی استفاده کرد. که در اینجا این لایه ۱۰۰ نود دارد.

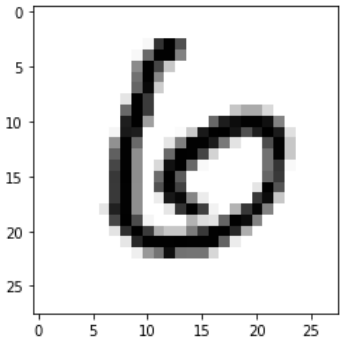
در نهایت پس از ۳ epoch، خروجی مدل fit شده بر روی داده‌ها به شرح زیر است. لازم به ذکر است، برای کم شدن زمان اجرای برنامه ۳ epoch در نظر گرفتیم. برا افزایش accuracy می‌توان تعداد epochهای بیشتری مانند ۱۰ یا ۲۰ در نظر گرفت. برای مثال پس از ۱۰ epoch، دقت الگوریتم به ۹۹٪ می‌رسد.

```
Epoch 1/3
1875/1875 [==============================] - 89s 47ms/step - loss: 0.3501 - accuracy: 0.8874 - val_loss: 0.0709 - val_accuracy:
0.9760
Epoch 2/3
1875/1875 [==============================] - 103s 55ms/step - loss: 0.0460 - accuracy: 0.9859 - val_loss: 0.0342 - val_accurac
y: 0.9882
Epoch 3/3
1875/1875 [==============================] - 107s 57ms/step - loss: 0.0290 - accuracy: 0.9911 - val_loss: 0.0332 - val_accurac
y: 0.9898
```

مقادیر test loss و test accuracy نیز به شرح زیر است:

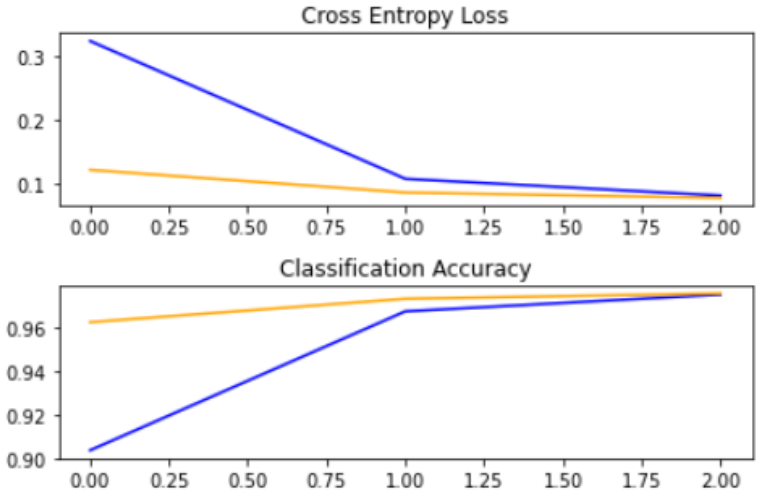
```
313/313 [==============================] - 5s 15ms/step - loss: 0.0332 - accuracy: 0.9898
Test loss 0.03324537351727486
Test accuracy 0.989799976348877
```

برای تشخیص کارکرد الگوریتم، داده‌ی ۱۱ام را از آن خروجی گرفتیم که عدد ۶ را به ما داد و تصویر ۱۱ ام نیز ۶ بود. یعنی الگوریتم به درستی عمل کرده است.



فیلترهای مربوط به لایه‌ی conolution به صورت زیر چاپ شده است:

در زیر نیز نمودارهایی را می‌بینید که یکی برای loss و دیگری برای accuracy است. خط آبی نشان‌دهنده‌ی عملکرد مدل بر روی داده‌های train است و خط نارنجی نمایش‌دهنده‌ی کارایی مدل بر روی داده‌های test است. این نمودار نشان می‌دهد که مدل به fit مناسبی بر روی داده‌ها دست پیدا کرده است و هیچ overfitting یا underfitting خاصی نداریم.



مقایسه مقادیر گفته شده در سوال:

اندازه‌ی کرنل

Kernel_size	1	3	5	7
Test loss	0.10707404464483261	0.03324537351727486	0.038108665496110916	0.037757255136966705
Test accuracy	0.968800008 2969666	0.98979997634 8877	0.98949998617 17224	0.98890000581 74133

نتیجه = با افزایش اندازه‌ی کرنل، دقت کاهش می‌یابد و مقدار loss افزایش می‌یابد.

تعداد کرنل

Kernel_number	32	64	128	256
Test loss	0.038096446 54393196	0.03324537351 727486	0.02637839317 3217773	0.01737735177 3217486
Test accuracy	0.988699972 6295471	0.98979997634 8877	0.99159997701 6449	0.99979997701 6547

نتیجه = با افزایش تعداد کرنل، دقت افزایش می‌یابد و مقدار loss کاهش می‌یابد.

نرخ یادگیری

Learning_rate	0.001	0.009	0.01	0.09
Test loss	0.033245373 51727486	0.03408449143 1713104	0.03761298209 428787	0.02924111858 010292
Test accuracy	0.989799976 348877	0.99059998989 10522	0.98930001258 8501	0.99029999971 38977

نتیجه = با افزایش نرخ یادگیری، دقت در بازه‌ی خود کاهش می‌یابد و مقدار loss افزایش می‌یابد.

توابع مختلف

Non-linear	RELU	SIGMOID	SELU	TANH
Test loss	0.033245373 51727486	0.04803386703 133583	0.05056547746 062279	0.03173898160 457611
Test accuracy	0.989799976 348877	0.98360002040 86304	0.98540002107 62024	0.98949998617 17224

نتیجه = تغییر فاحشی در دقت دیده نمی‌شود، برخی از آن‌ها دقت را کمی بالا برده و برخی دیگر کمی پایین

تعداد لایه‌ها

Layers_number	1	2	3
Test loss	0.046609025448560715	0.028474153950810432	0.03324537351727486
Test accuracy	0.9847000241279602	0.9901000261306763	0.9897999763488877

نتیجه = دقت بالا وپایین می‌رود و نمی‌توان تغییرات را بیان کرد.

منابع تشریحی

1	https://www.analyticsvidhya.com/blog/2018/07/top-10-pretrained-models-get-started-deep-learning-part-1-computer-vision/ https://www.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html https://github.com/onnx/models
2	https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148 https://medium.com/analytics-vidhya/cnn-convolutional-neural-network-8d0a292b4498 https://towardsdatascience.com/image-augmentation-for-deep-learning-histogram-equalization-a71387f609b2 منبع کمکی دیگر: توضیحات سرکار خانم شیخی در گروه تلگرامی درس در رابطه با cnn
3	https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/ https://www.superdatascience.com/blogs/the-ultimate-guide-to-convolutional-neural-networks-cnn https://deeplearning.ir/%D8%A2%D9%85%D9%88%D8%B2%D8%B4-%D8%B4%D8%A8%DA%A9%D9%87-%DA%A9%D8%A7%D9%86%D9%88%D9%84%D9%88%D8%B4%D9%86-%D8%A8%D8%AE%D8%B4-%D8%A7%D9%88%D9%84/ https://cs231n.github.io/convolutional-networks/ https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

منابع پیاده‌سازی

https://towardsdatascience.com/mnist-cnn-python-c61a5bce7a19

https://victorzhou.com/blog/intro-to-cnns-part-1/

https://victorzhou.com/blog/keras-cnn-tutorial/

https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-mnist-handwritten-digit-classification/