دانشکده مهندسی کامپیوتر

مباحث ویژه ۱ (یادگیری عمیق)

تمرین سری نهم

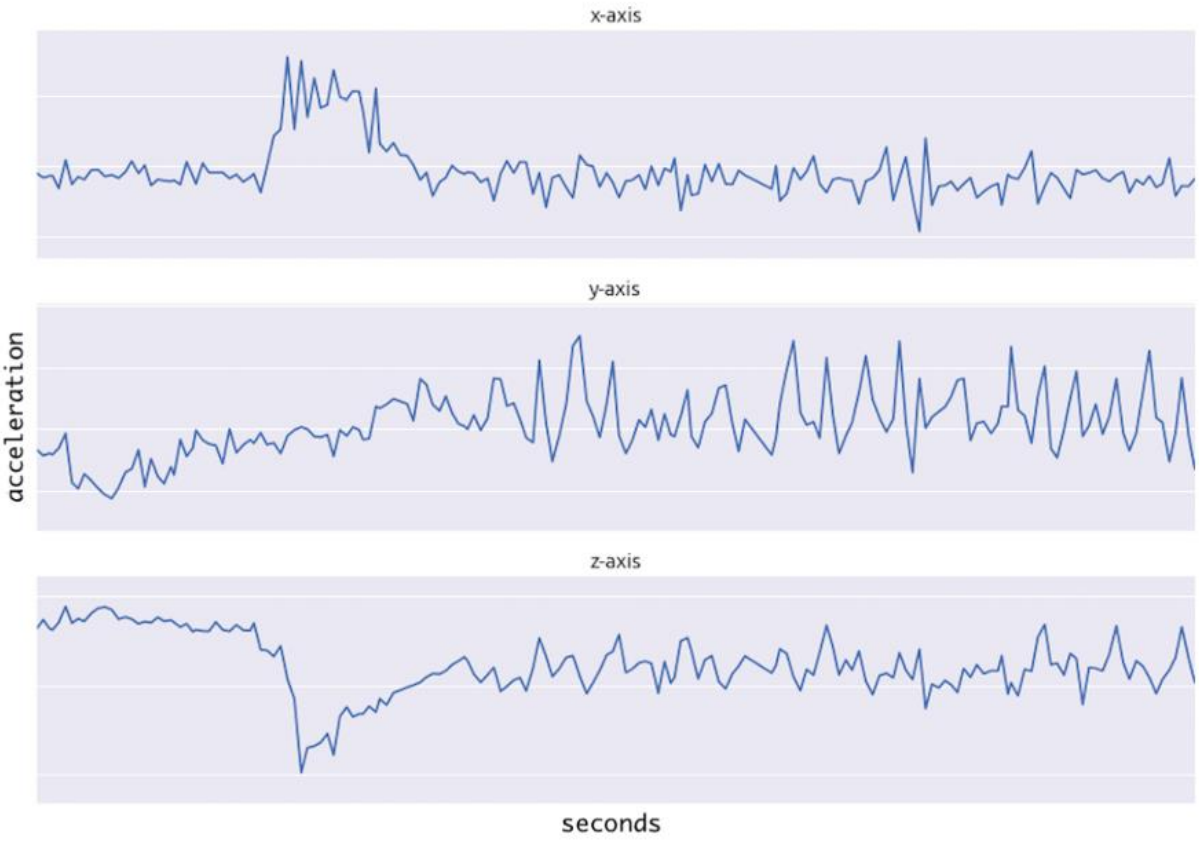
علی صداقی

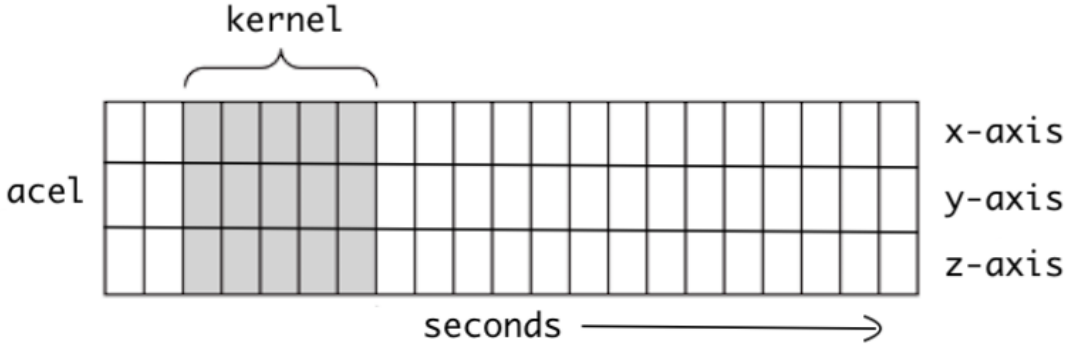
۹۷۵۲۱۳۷۸

۱ سوال اول

الف)

Conv1D: در سیگنال‌های ورودی همانند سیگنال صدا استفاده می‌شود. با استفاده از آنها می‌توان الگوهایی را در صدا بدست آورد. در واقع در آن کرنل فقط بر روی یک بعد حرکت می‌کند. یکی از کاربردهای آن پردازش داده‌های مربوط به یک شتاب‌سنج (Accelerometer) است. هنگامی که یک فرد یک شتاب‌سنج به دستش بسته و حرکت می‌کند حرکات او در ۳ محور x y z در طول زمان بدست می‌آید (Time series data). یک Conv1D می‌تواند فعالیت‌های فرد را در طول زمان زمان مانند پریدن، دویدن، ایستادن و راه رفتن را تشخیص دهد.





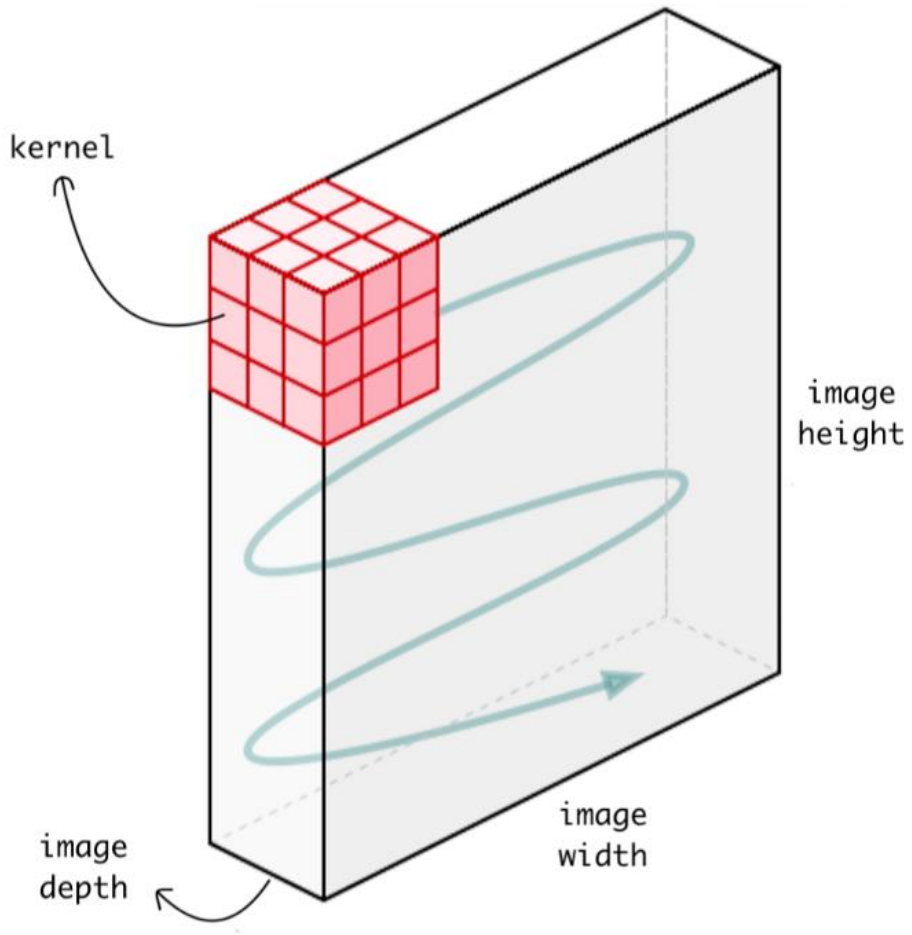
به صورت کلی می‌توان از آن در جاهایی که با یک Time Series Data سرکار داریم استفاده کنیم. مانند: صدا، متن و اطلاعات سنسوری مانند شتاب‌سنج و ...

تعداد پارامترها در آن کم می‌باشد. ورودی آنها یک تنسور ۲ بعدی است.

Conv2D: در تصاویر استفاده می‌شود و محبوب‌ترین نوع کانولوشن است. برای مثال اگر یک تصویر ۲ بعدی که چندین کانال دارد را داشته باشیم، هر فیلتر کانولوشن یک فیلتر سه بعدی خواهد بود که عملیات Cross correlation در ریاضیات را انجام می‌دهد. برای یافتن الگوهای موجود در تصویر مثل لبه‌ها، رنگ‌ها و ... مناسب است. عاملی که باعث محبوبیت آن شده این است که این کانولوشن می‌تواند ویژگی‌های مکانی (Spatial) را به خوبی استخراج کند. استفاده از آن شبکه را در کارهای مربوط به تصاویر Robust می‌کند.

کاربردها: Classification images، Segmentation، Object Detection و ... به صورت کلی هرجا که با image در ارتباط هستیم.

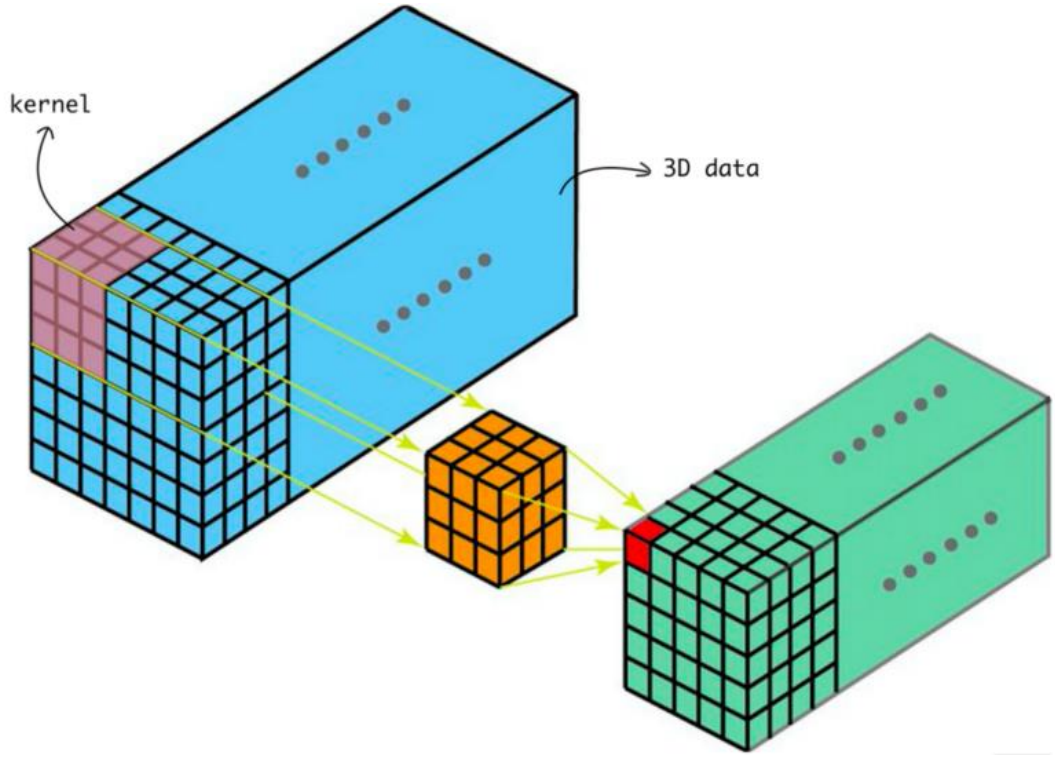
ورودی آنها یک تنسور ۳ بعدی است.



Conv3D: معمولا در ویدیو و فیلم که برای هر لحظه یک فریم داریم از آن استفاده می‌شود. نسبت به حالت 2D یک بعد بیشتر دارند که شامل مقادیر گسسته است و بیانگر مفهوم زمان است. کرنل در آن در ۳ بعد حرکت می‌کند.

کاربرد دیگر آن تصاویر ۳ بعدی مانند تصاویر MRI و CT است. در واقع این تصاویر از ترکیب مجموعه‌ای از تصاویر X-ray در زاویه‌های مختلف ایجاد می‌شوند (بعد سوم). شبکه‌های مبتنی بر Conv3D قادر هستند ویژگی‌های مناسب را از این تصاویر استخراج کنند و عملیات تشخیص را بهتر کنند.

تعداد پارامترها در آن زیاد می‌باشد. ورودی آنها یک تنسور ۴ بعدی است.



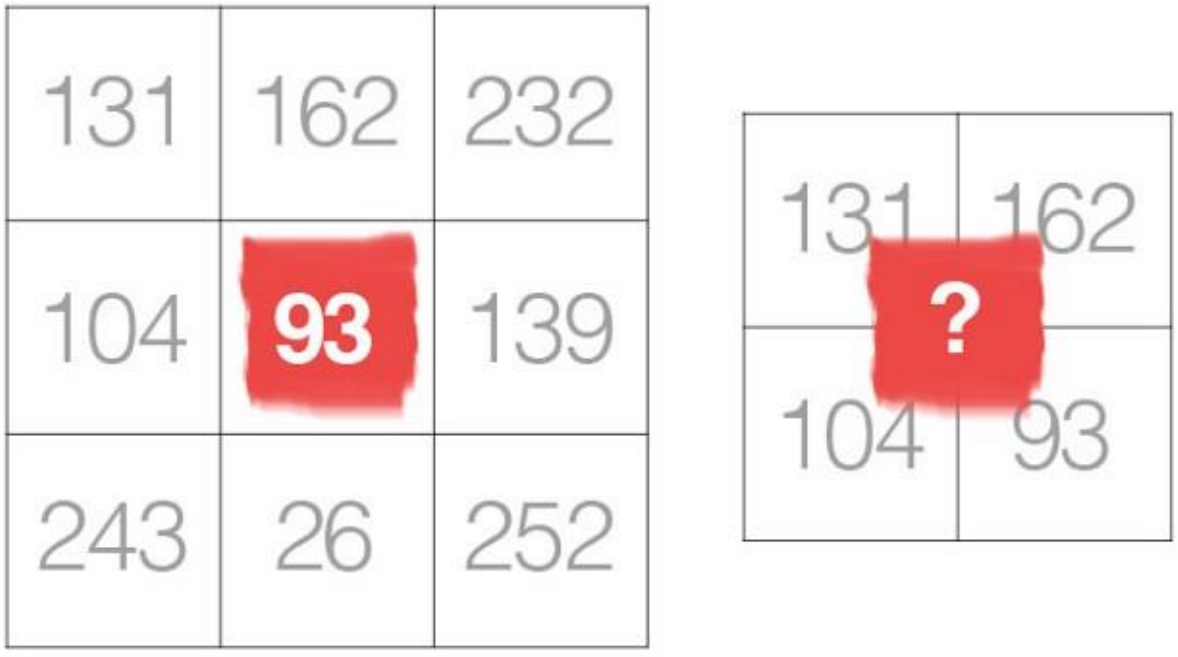
منابع:

https://datascience.stackexchange.com/questions/51470/what-are-the-differences-between-convolutional1d-convolutional2d-and-convoluti

https://xzz201920.medium.com/conv1d-conv2d-and-conv3d-8a59182c4d6

ب) فیلترها در Conv2D معمولا مربعی هستند یعنی ابعاد آن به صورت N*N است. دلیل آن این است که ما هیچ ترجیحی درباره اینکه یک الگو در چه جهتی یافت شود نداریم. برای مثال اگر فیلتر مستیطیلی باشد در یک سمت حرکت بیشتری خواهیم داشت و آن الگو(ویژگی) را در آن جهت بیشتر مورد توجه قرار می‌دهیم. دلیل دیگر آن حفظ تقارن (Symmetry) در طول شبکه است. به همین دلیل است که در شبکه‌های Inception اثر فیلترهای غیر مربعی را خنثی می‌کنیم. به طور خلاصه فیلتر مربعی به الگوهای افقی(Horizontal) و عمودی(Vertical) به یک اندازه توجه می‌کند. دلیل دیگر آن سادگی محاسبات در هنگام اضافه کردن Padding و Stride است. زیرا هرچه که درباره‌ی Height عکس محاسبه شده درباره Width نیز صادق است.

یکی از نکاتی که در انتخاب انداره کرنل باید به آن توجه کنیم فرد (Odd) بودن اندازه کرنل است. در این حالت کرنل دارای نقطه مرکزی است و می‌توان محل کرنل را با مختصات نقطه وسط آن آدرس دهی کرد. مزیت دیگر فرد بودن اندازه کرنل این است که تمامی پیکسل‌های لایه قبلی به صورت متقارن (Symmetric) حول پیکسل خروجی هستند. اما کرنل‌هایی که اندازه زوج دارند دارای اعواج (Distortions) و Aliasing بسیار هستند زیرا این تقارن را دیگر نداریم. همچنین در حالت فرد بودن اندازه می‌توان هر پیکسل خروجی را به یک پیکسل ورودی نگاشت کرد (نگاشت مرکز) و ویژگی‌های هر نقطه را درون‌یابی (Interpolate) کرد اما این کار را در حالت زوج نمی‌توان انجام داد.



نکته دیگری که باید به آن توجه کنیم اندازه کرنل است. بیشتر ویژگی‌های مهم تصویر به صورت محلی و در محل‌های کوچک قرار گرفته اند. پس باید اندازه فیلتر را کوچک انتخاب کنیم تا این ویژگی‌هایی که در محل‌های کوچک

نهفته شده اند را استخراج کنیم. همچنین تعداد پارامترهای یک کرنل با اندازه آن رابطه quadratically دارد و استفاده از کرنل‌های بزرگتر تعداد پارامترها را بسیار زیاد می‌کند. به همین دلیل ترجیح ما در استفاده از کرنل‌های 7*7 یا 5*5 یا 3*3 است. کرنل 3*3 محبوب‌ترین و پر استفاده‌ترین کرنل است. می‌توان کرنل‌های 7*7 و 5*5 را با چند کرنل 3*3 که بر روی هم قرار گرفته اند ایجاد کرد. اگر به تاریخچه شبکه‌های CNN نیز نگاه کنیم میبینم به مرور زمان از کرنل‌های 11*11 و 7*7 و به سمت کرنل‌های 3*3 تغییر رویه داده ایم.

اینکه در هر لایه Conv2D چه تعداد فیلتر بکار می‌بریم بستگی به تعداد ویژگی که می‌خواهیم استخراج کنیم دارد. اما به صورت کلی می‌توان گفت هر چه به لایه‌های عمیق‌تر می‌رویم Height و Width کاهش می‌یابد و تعداد فیلتر (Channel) بیشتر می‌شود.

منابع:

https://stackoverflow.com/questions/49003346/why-convolutional-nn-kernel-size-is-often-selected-as-a-square-matrix

https://towardsdatascience.com/deciding-optimal-filter-size-for-cnns-d6f7b56f9363

https://medium.com/analytics-vidhya/significance-of-kernel-size-200d769aecb1

ج) دلایل استفاده از Pooling:

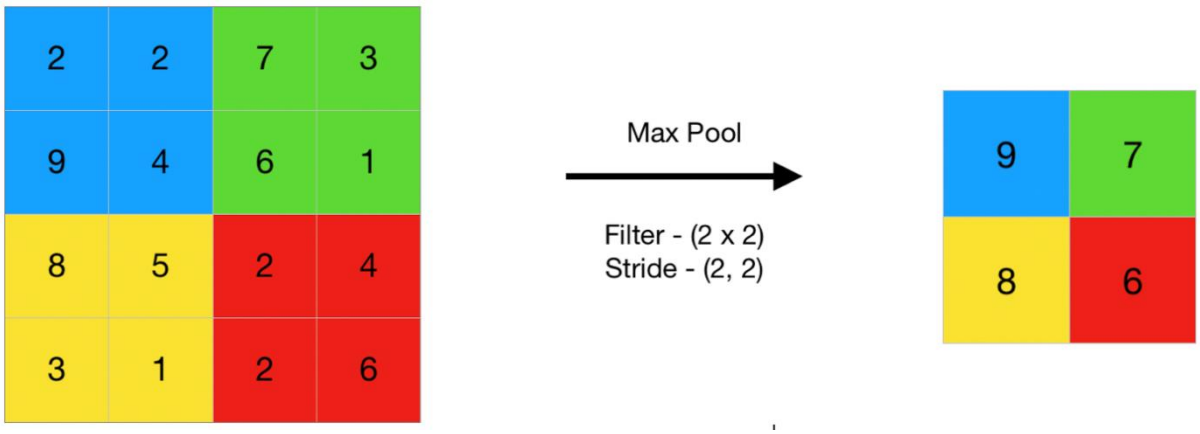
۱- کاهش ابعاد نقشه ویژگی ورودی در نتیجه کاهش پارامترها و افزایش سرعت شبکه

۲- خلاصه‌سازی ویژگی‌هایی که در یک ناحیه از Feature map قرار گرفته اند. در نتیجه آن محاسبات بعدی روی ویژگی‌های خلاصه شده انجام می‌شود و دیگر با محل دقیق ویژگی‌ها که توسط لایه Conv2D پیدا شده در طرف نیستیم. این کار باعث می‌شود مدل نسبت به گوناگونی‌های قرار گرفتن ویژگی‌ها Robust تر شود و تغییرات محلی کوچک در عکس مدل را به هم نزند.

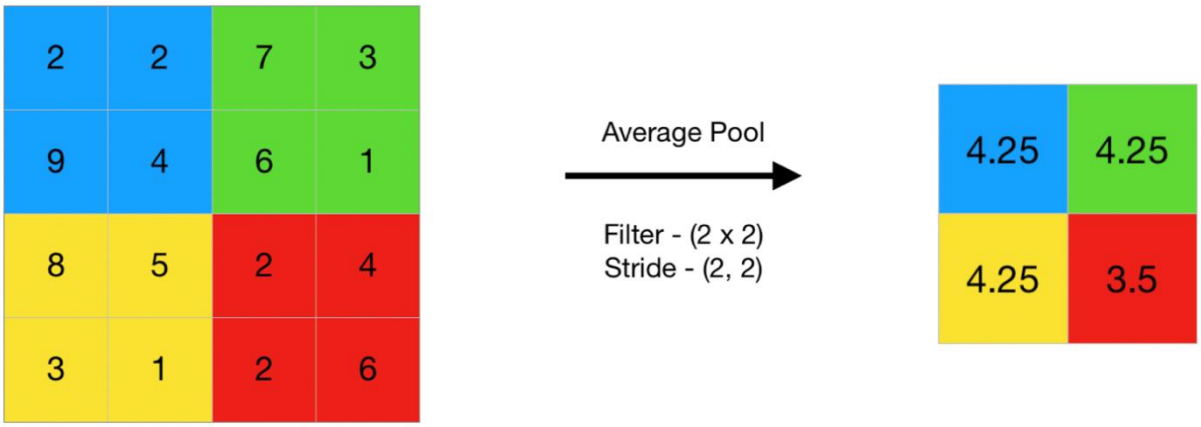
۳- این لایه‌ها پارامتر قابل آموزش ندارند و کمتر باعث پیچیده شدن شبکه می‌شوند.

انواع Pooling:

۱- Max Pooling: در آن ماکسیمم المان در هر ناحیه‌ای که کرنل روی آن قرار گرفته، خروجی داده می‌شود. خروجی آن شامل برجسته‌ترین ویژگی‌هایی ورودی است. معمولا پس در انتهای هر بلاک Conv از آن استفاده می‌شود تا هم سایز خروجی کم شود هم ویژگی‌های برجسته در هرد ناحیه خروجی داده شوند. این پولینگ کمک می‌کند تا شارپ‌ترین ویژگی‌های موجود در ورودی را استخراج کنیم.



۲- Average Pooling: در آن میانگین المان‌ها در هر ناحیه‌ای که کرنل روی آن قرار گرفته، خروجی داده می‌شود. بر خلاف Max Pooling که برجسته‌ترین ویژگی‌ها را بر می‌گرداند، Average Pooling میانگین ویژگی‌های یک ناحیه را در نظر می‌گیرد. استفاده از آن نسبت به MaxPool کمتر است زیرا هم تعداد عملیات بیشتری دارد، هم خروجی آن مشخص کننده وجود یک ویژگی در یک ناحیه نیست و ممکن است یک ویژگی در یک ناحیه وجود داشته باشد اما میانگین کم شود و ما وجود آن ویژگی در آن ناحیه را در نظر نگیریم. دلیل استفاده از این لایه در کاهش واریانس موجود در داده ورودی می‌باشد. در واقع ویژگی‌هایی که این لایه می‌دهد Smoothتر هستند.



۳- Global Pooling: با استفاده از آن می‌توان ماتریس موجود در هر کانال را به یک تک عدد تبدیل کرد. در واقع با ورودی گرفتن *H*W*C* خروجی *1*1*C* را تولید می‌کند. نحوه کاهش بعد آن به دو صورت می‌تواند باشد. ۱- *Global Average Pool(GAP)* ۲- *Global Max Pool(GMP)*.

معمولا قبل از بلاک Fully Connected استفاده می‌شود و به نوعی کار Flatten را انجام می‌دهد. معمولا هنگامی که از Flatten استفاده می‌کنیم احتمال Overfit وجود دارد اما درGlobal Pooling این احتمال کمتر است. در واقع GAP باعث ایجاد اثر Regularize می‌شود.

- ✓ Min Pooling هم داریم که دقیقا عکس کار ماکس پولینگ را انجام می‌دهد.

منابع:

https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/

https://analyticsindiamag.com/comprehensive-guide-to-different-pooling-layers-in-deep-learning/

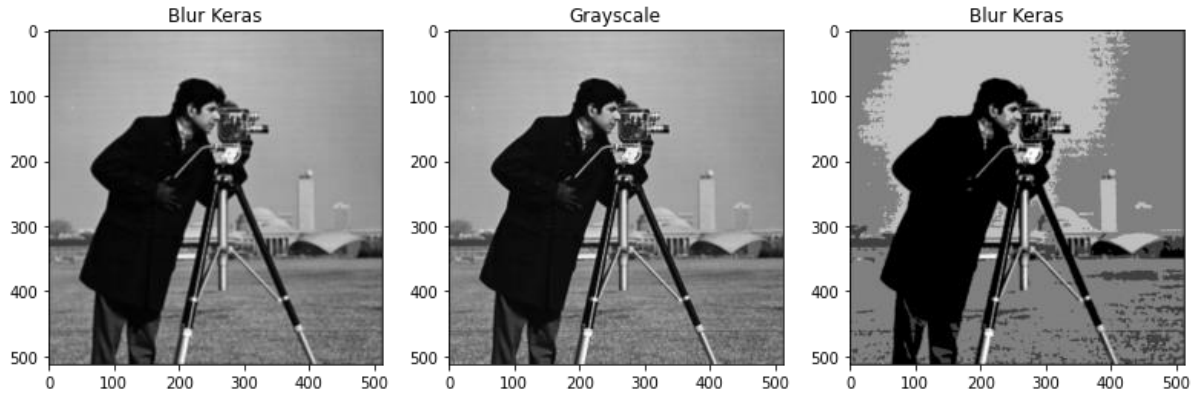
۲ سوال دوم

این سوال را به دو روش حل کردیم:

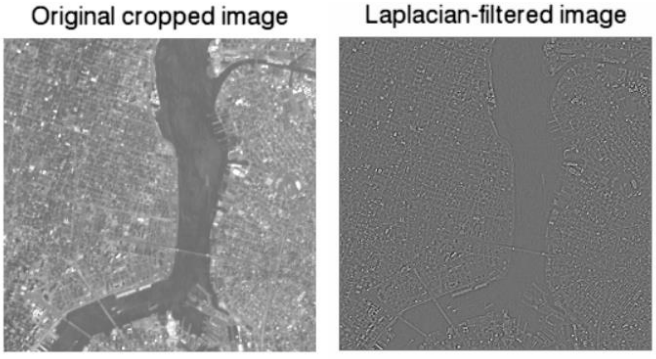
- روش اول: مدل کراس با یک لایه Conv2D که وزن‌های آن با تابع set weights دستی تنظیم شده. در نهایت با predict کردن مدل روی عکس ورودی
- روش دوم: تابع آماده cv2.filter2D

توضیح فیلترها:

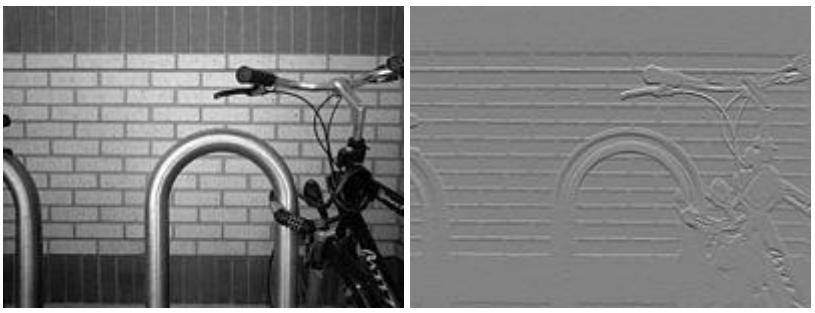
- فیلتر اول: یک فیلتر Blur کننده است. هر چه مخرج کسر بزرگ‌تر باشد با شدت بیشتری Blur می‌کنیم. مثلا: در تصاویر پایین تصویر سمت چپ بدون Blur (مخرج برابر 1)، تصویر وسط Blur با شدت 9 و تصویر سمت راست Blur با شدت 500 است. می‌توان این فیلتر را به عنوان یک Smooth کننده تصویر نیز استفاده کرد.



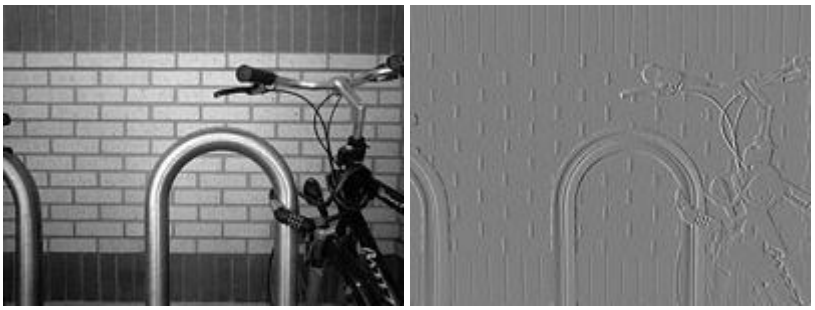
- فیلتر دوم: کرنل Laplacian می‌باشد که یک لبه یاب (Edge detector) است. این کرنل از ورودی مشتق دوم مکانی (Spatial) می‌گیرد و نرخ تغییرات مشتق اول عکس را نشان می‌دهد. در واقع اگر در مقدار پیکسل همسایه تغییری داشته باشیم یک Edge داریم. این کرنل نواحی ای که تغییرات سریع شدتی (rapid intensity change) دارند را به خوبی تشخیص می‌دهد. برای مثال:



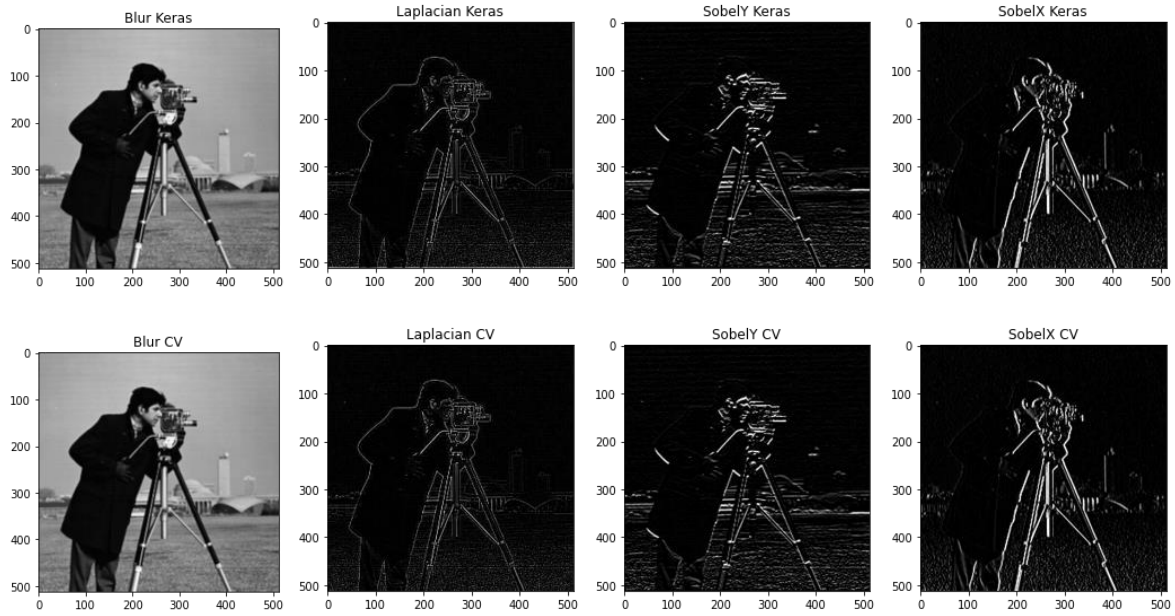
- فیلتر سوم: فیلتر Sobel در جهت محور y است. وظیفه آن تشخیص تغییرات افقی (Horizontal) در گرادیان عکس است. در واقع این فیلتر با ورودی گرفتن یک تصویر خروجی تولید می‌کند که مشخص کننده لبه‌های افقی (Horizontal edges) تصویر است. دلیل کارکرد آن هم بدلیل تفاوت مجموع سطر اول با سطر سوم است. یعنی مجموع مقادیر سطر اول برابر 4- و این مقدار در سطر سوم برابر 4+ است. برای مثال:



- فیلتر چهارم: فیلتر Sobel در جهت محور x است. وظیفه آن تشخیص تغییرات عمودی (Vertical) در گرادیان عکس است. در واقع این فیلتر با ورودی گرفتن یک تصویر خروجی تولید می‌کند که مشخص کننده لبه‌های عمودی (Vertical edges) تصویر است. دلیل کارکرد آن هم بدلیل تفاوت مجموع ستون اول با ستون سوم است. یعنی مجموع مقادیر ستون اول برابر ۴- و این مقدار در ستون سوم برابر ۴+ است. برای مثال:



نتایج حاصل شده به صورت زیر است:



منابع:

https://en.wikipedia.org/wiki/Sobel_operator

https://www.irya.unam.mx/computo/sites/manuales/IDL/Content/GuideMe/ImageProcessing/LaplacianFilters.html

https://www.l3harrisgeospatial.com/docs/laplacianfilters.html

۳ سوال سوم

الف) یک ابزار مقیاس‌پذیر برای بهینه‌سازی هایپرپارامترهای یک مدل است. این Framework مسئله پر دردسر و سخت پیدا کردن بهترین هایپرپارامتر را به شیوه‌ای آسان و نظام‌مند برای ما حل می‌کند. به آسانی می‌توانیم فضای جست‌وجو (Search Space) را تعریف کنیم و پیدا کردن بهترین حالت را به این ابزار بسپاریم. در این Framework الگوریتم‌های جست‌وجوی زیادی وجود دارد که هر کدام قادر هستند بهترین هایپرپارامترهای شبکه را بیابند. نحوه‌ی کار این ابزار به این صورت است که به جای هایپرپارامترها از توابع این Framework استفاده کنیم. توابعی که برای تعریف هایپرپارامترها استفاده می‌شوند عبارتند از:

- Boolean
- Choice
- Fixed
- Float
- Int

می‌توان برای هر یک از توابع بالا مقدار max، min، step و sampling را تعریف کرد. در هنگام Search به ازای توابع بالا مقادیر واقعی جایگزین می‌شوند. از نام هر یک نیز مشخص است چکاری می‌کند.

ب) در این ابزار Tunerهای مختلفی وجود دارد که هر یک با الگوریتم مخصوص خود فضای جست‌وجو را پیمایش می‌کنند و سعی در پیدا کردن بهترین هایپرپارامترها دارند.

- Base Tuner: این یک کلاس Parent برای سایر تیونرها است و باقی تیونرها از آن ارث می‌برند. وظیفه آن Build، Train، Evaluation و ذخیره سازی مدل است.
- Random Search: یک راه برای پیدا کردن بهترین هایپرپارامترها این است که تمامی ترکیب های ممکن را امتحان کنیم. به این کار Grid Search نیز می‌گویند. اما مشکل این روش این است که تعداد ترکیب‌ها با افزایش هایپرپارامترها به صورت نمایی افزایش می‌یابد. امتحان کردن هر یک از این ترکیب‌ها بسیار طول می‌کشد. جست‌وجوی تصادفی به ما کمک می‌کند تا زمانی کمتری نسبت به Grid Search صرف کنیم. مشکل جست‌وجوی تصادفی این است که تضمینی برای پیدا کردن بهترین هایپرپارامتر ندارد. اما نتیجه نزدیک به بهترین است.

- Hyperband: این روش سعی در رفع کردن یکی از مشکلات Random Search دارد. مشکل موجود در Random Search به این صورت است که ما هایپرپارامترهایی را امتحان می‌کنیم که واضح است نتیجه بدی خواهند داشت و ما زمان زیادی را برای آموزش ور ارزیابی این شبکه‌ی بد سپری می‌کنیم. روش Hyperband پس از انتخاب تصادفی یک مجموعه از هایپرپارامترها به جای اینکه شبکه را به صورت کامل آموزش دهد و در ادامه ارزیابی کند، مدل را تنها برای چند ایپاک آموزش می‌دهد و بهترین نتایج را در این چند ایپاک به مرحله‌بعدی آموزش می‌فرستد. این کار مکررا انجام می‌شود تا در نهایت آموزش کامل بر روی نمایندگان نهایی صورت گیرد.
- Bayesian Optimization: این روش سعی می‌کند مشکل مشترک در Random Search و Hyperband را حل کند. مشکل این است: تمامی هایپرپارامترها به صورت Random با هم ترکیب می‌شوند. این کار کمک می‌کند که فضای حالت را جست‌وجو کنیم اما تضمینی برای پیدا کردن بهترین و بهینه‌ترین هایپرپارامترها ندارد. در این روش به‌جای اینکه تمامی حالات به صورت رندوم باشند، چند حالت ابتدایی را رندوم در نظر می‌گیرد سپس یا توجه به نتایج این حالات ابتدایی، بهترین حالت ممکن بعدی را می‌سازد. در واقع در این روش از تاریخچه‌ی حالت‌های قبلی برای ساختن حالت جدید استفاده می‌شود. این کار تا زمانی ادامه می‌یابد که یا به بهترین جواب برسیم یا به ماکس تعداد آزمایش‌ها برسیم. در این روش با آرگومان beta مشخص می‌کنیم که باید به کاوش بپردازیم (Explore) یا از دانسته‌های قبلی استفاده کنیم (Exploit).
- Sklearn Tuner: این تیونر مخصوص مدل‌هایی است که با ابزار Sci-Kit Learn ایجاد شده اند. ما در این سوال مدل را با Keras ساخته ایم.

ما در این سوال از روش دوم یعنی Hyperband استفاده کردیم زیرا منابع سخت‌افزاری و زمانی محدودی داشتیم و نمی‌خواستیم زمان زیادی را صرف آموزش مدل‌های معیوب کنیم. اما اگر در منابع و زمان محدودیتی نداشتیم روش Grid Search گزینه‌ی بهتری بود زیرا تمامی حالات به صورت کامل بررسی می‌شد و مطمئن بودیم بهترین حالت انتخاب شده است. دلیل آنکه از روش بیزین استفاده نکردیم این است هایپرپارامترهای موجود Orthogonal نیستند و همچنین فضای حالت بزرگی نداریم.

ج) یک مدل طراحی کردیم که برخی از ویژگی‌های آن توسط Tuner تعیین می‌شود. فضای جست‌وجوی تیونر به صورت زیر است:

```
Search space summary
Default search space size: 6

conv_layers (Int)
{'default': None, 'conditions': [], 'min_value': 1, 'max_value': 3,
'step': 1, 'sampling': None}

filters (Int)
{'default': None, 'conditions': [], 'min_value': 8, 'max_value': 32,
'step': 8, 'sampling': None}
dense_layers (Int)
{'default': None, 'conditions': [], 'min_value': 1, 'max_value': 4,
'step': 1, 'sampling': None}

units (Int)
{'default': None, 'conditions': [], 'min_value': 128, 'max_value': 256,
'step': 64, 'sampling': None}

lr (Choice)
{'default': 0.0001, 'conditions': [], 'values': [0.0001, 0.0005,
0.001], 'ordered': True}

optimizer (Choice)
{'default': 'adam', 'conditions': [], 'values': ['adam', 'sgd'],
'ordered': False}
```

این ۶ هایپرپارامتر مطابق جدول درون سوال است. اما برای اینکه مدل حالت‌های واضح بد را امتحان نکند محدودیت‌هایی ایجاد کردیم:

۱- مدل از تعدادی بلاک Conv تشکیل شده و در هر بلاک ۲ لایه conv داریم که تعداد فیلتر یکسان دارند.

۲- بعد از هر بلاک Conv از یک لایه MaxPool با stride=2 و pool=2 استفاده کردیم.

۳- بعد از هر لایه MaxPool یک لایه Dropout استفاده کردیم که نرخ آن هرچه به لایه‌های عمیق‌تر می‌رویم افزایش می‌یابد.

۴- تعداد فیلترهای هر بلاک Conv دو برابر همین مقدار در بلاک قبلی است.

۵- از چندین لایه Dense با تعداد نورون یکسان استفاده کردیم. پس از هر لایه Dense یک لایه Dropout با نرخ 0.5 استفاده کردیم.

در هایپرپارامتر Hyperband مقدار Max Epochs را برابر 10 قرار می‌دهیم بنابر فرمول تعداد Trialها برابر 30 می‌شود. سپس با انتخاب بهترین مدل آن را در 50 ایپاک آموزش می‌دهیم و در نهایت روی داده تست ارزیابی می‌کنیم.

نتایج به صورت زیر است:

```
Trial 30 Complete [00h 03m 23s]
val_accuracy: 0.25049999356269836
Best val_accuracy So Far: 0.7250000238418579
Total elapsed time: 00h 34m 13s
INFO:tensorflow:Oracle triggered exit


Results summary
Results in ./Q3_Models/untitled_project
Showing 10 best trials
Objective(name='val_accuracy', direction='max')

Trial summary
Hyperparameters:
conv_layers: 2
filters: 16
dense_layers: 3
units: 192
lr: 0.001
optimizer: adam
tuner/epochs: 10
tuner/initial_epoch: 0
tuner/bracket: 0
tuner/round: 0
Score: 0.7250000238418579

Trial summary
Hyperparameters:
conv_layers: 2
filters: 16
dense_layers: 2
units: 192
lr: 0.001
optimizer: adam
tuner/epochs: 10
tuner/initial_epoch: 4
tuner/bracket: 1
tuner/round: 1
tuner/trial_id: 190e537282f0d80648d0ab4fe0e78879
Score: 0.7148000001907349

Trial summary
Hyperparameters:
conv_layers: 1
filters: 24
dense_layers: 1
units: 256
lr: 0.001
```

```
optimizer: adam
tuner/epochs: 10
tuner/initial_epoch: 4
tuner/bracket: 1
tuner/round: 1
tuner/trial_id: 2b437527cf02babb73f617ae69218328
Score: 0.6937000155448914
```

پس از انتخاب بهترین هایپرپارامترها مدل را دوباره می‌سازیم.

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 32, 32, 3)]       0

 conv2d_4 (Conv2D)           (None, 32, 32, 32)        896

 conv2d_5 (Conv2D)           (None, 32, 32, 32)        9248

 max_pooling2d_2 (MaxPooling  (None, 16, 16, 32)       0
 2D)

 dropout_4 (Dropout)         (None, 16, 16, 32)        0

 conv2d_6 (Conv2D)           (None, 16, 16, 64)        18496

 conv2d_7 (Conv2D)           (None, 16, 16, 64)        36928

 max_pooling2d_3 (MaxPooling  (None, 8, 8, 64)         0
 2D)

 dropout_5 (Dropout)         (None, 8, 8, 64)          0

 flatten_1 (Flatten)         (None, 4096)              0

 dense_3 (Dense)             (None, 192)               786624

 dropout_6 (Dropout)         (None, 192)               0

 dense_4 (Dense)             (None, 192)               37056

 dropout_7 (Dropout)         (None, 192)               0

 dense_5 (Dense)             (None, 192)               37056

 dropout_8 (Dropout)         (None, 192)               0

 dense_6 (Dense)             (None, 10)                1930

=================================================================
Total params: 928,234
Trainable params: 928,234
Non-trainable params: 0
```

آن را در 50 اپاک آموزش می‌دهیم و در نهایت ارزیابی می کنیم.

ارزیابی روی داده آموزشی:

```
782/782 [==============================] - 7s 8ms/step - loss: 0.3502 -
accuracy: 0.8951
```

ارزیابی روی داده تست:

```
157/157 [==============================] - 1s 8ms/step - loss: 0.6754 -
accuracy: 0.7775
```

دلیل بالا رفتن ناگهانی دقت روی داده آموزش این است که دیگر داده Validation نداریم و روی تمامی دیتاست ارزیابی را انجام دادیم.

- ✓ می‌توانستیم به نتیجه و دقت بهتری روی داده تست دست یابیم، اما محدودیت‌هایی نظیر مدت استفاده از GPU، زمان‌بر بودن فرایند آموزش، زمان‌بر بودن فرایند Search، تفاوت داده Validation در فازهای مختلف باعث شد عملکرد مطلوبی نداشته باشیم.
- ✓ در واقع هدف را در این سوال استفاده از Keras Tuner قرار دادیم نه بهترین دقت بر روی داده تست.

منابع:

https://medium.com/swlh/hyperparameter-tuning-in-keras-tensorflow-2-with-keras-tuner-randomsearch-hyperband-3e212647778f

۴ سوال چهارم

با استفاده از روابط زیر جدول را کامل می‌کنیم: (با فرض برابر بودن پارامترها در Height و Width)

$Input\ shape: (h, w, c)$

$Conv2D: f, (k, k), (s, s), (p, p)$

$Output\ shape: (\left\lfloor \frac{h - k + 2p}{s} + 1 \right\rfloor, \left\lfloor \frac{w - k + 2p}{s} + 1 \right\rfloor, f)$

$\#\ parameters: (k \times k \times c + 1) \times f$

$LocallyConnected2D: f, (k, k), (s, s), (p, p)$

$Output\ shape: (\left\lfloor \frac{h - k + 2p}{s} + 1 \right\rfloor, \left\lfloor \frac{w - k + 2p}{s} + 1 \right\rfloor, f)$

$\#\ parameters: (O_H \times O_W) \times (k \times k \times c \times f) + (O_H \times O_W) \times f$

$Input\ shape: (v)$

$Dense: u = number\ of\ units = Output\ shape$

$\#\ parameters: u \times v + u = u \times (v + 1)$

- ✓ لایه‌های Input، MaxPool2D و Flatten پارامتری برای یادگیری ندارند.
- ✓ تفاوت Locally Connected و Conv در اشتراک وزن‌هاست. در حالت Locally Connected باید کرنل را در جاهای مختلف و با وزن‌های مختلف قرار دهیم. پس در رابطه تعداد پارامترها، محل‌های قرار گیری کرنل را در نظر می‌گیریم.
- ✓ در واقع ترم $(O_H \times O_W)$ در لایه Locally Connected بیانگر جاگیری کرنل در جاهای مختلف با وزن‌های مختلف است.
- ✓ Output size از حاصل ضرب درایه‌های Output shape به دست می‌آید.
- ✓ به طور کلی هر چه در یک شبکه به لایه‌های عمیق‌تر می‌رویم اندازه خروجی (Activation size) کوچکتر می‌شود.
- ✓ مقادیر محاسبه شده به صورت زیر است.

Layer	Output shape	Out size	Parameters
Input	(28, 28, 1)	784	0
Conv2D filters = 20 kernel size = (7, 7)	(22, 22, 20)	9680	$(7 \times 7 \times 1 + 1) \times 20 = 1000$
MaxPool2D pool size = (2, 2) strides = (2, 2)	(11, 11, 20)	2420	0
Conv2D filters = 10 kernel size = (5, 5)	(7, 7, 10)	490	$(5 \times 5 \times 20 + 1) \times 10 = 5010$
LocallyConnected2D filters = 2 kernel size = (3, 3)	(5, 5, 2)	50	$(5 \times 5) \times (3 \times 3 \times 10 \times 2) + (5 \times 5) \times 2 = 4550$
Flatten	(50)	50	0
Dense units = 10	(10)	10	$10 \times 50 + 10 = 510$

$$Total\ trainable\ params = 1000 + 5010 + 4550 + 510 = 11070$$

از مقایسه نتایج خود با تابع summary درون کراس متوجه می‌شویم محاسبات درستی داشته ایم.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 22, 22, 20)        1000

 max_pooling2d (MaxPooling2D  (None, 11, 11, 20)       0
 )

 conv2d_1 (Conv2D)           (None, 7, 7, 10)          5010

 locally_connected2d (Locall  (None, 5, 5, 2)          4550
 yConnected2D)

 flatten (Flatten)           (None, 50)                0

 dense (Dense)               (None, 10)                510

=================================================================
Total params: 11,070
Trainable params: 11,070
Non-trainable params: 0
```