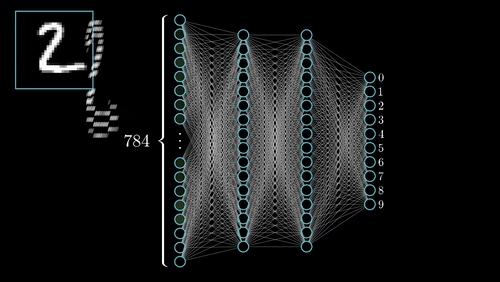
**Neural Network**



شبکه های عصبی یک شبکه ای از نورون ها هستند که تشکیل شده اند از یک لایه ورودی، یک

لایه خروجی و تعدادی لایه مخفی(Hidden layer) دارند. مثلا توی این شکل که دارید مشاهده

می کنید این یک شبکه ی عصبی هست که کارش تشخیص ارقام هست. که لایه خروجی 10 تا

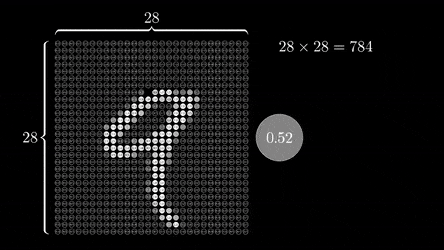
نورون داره از 0 تا 9 ، لایه ورودی 784 نورون دارد لایه ورودی عکس هست و این عکس 28 \* 28

هست و 28 \* 28 را ضرب کنیم می شود که می شود 784. و 784 تعداد پیکسل های عکس

هست و یک بردار به طول 784 داریم. و هر عکس می شود یک بردار و از لایه ها رد می شود و

خروجی می دهد که این خروجی پیش بینی شبکه های عصبی هست.

**Input Features**



این یک شبکه های عصبی معمول هست و ورودی ما همیشه مانند یک بردار هست و حتی

اگر ورودی ما یک عکس هم باشد تمام سطر و ستون را به هم می چسباند و به صورت یک

بردار می شود و ان را به عنوان یک ورودی به شبکه های عصبی می دهیم . از شبکه های

عصبی هم میتوان برای عکس و هم متن استفاده کرد.

**ConvNets = Convolutional Neural Networks**



زیر مجموعه ای از شبکه های عصبی معمول هستند . این مدل هم لایه ورودی، خروجی، لایه

های مخفی و وزن و بایاس دارند. اما یک سری تغییرات با شبکه های معمول دارند. ورودی

کانولوشن ها همیشه عکس یا فیلم و چیزهای مرتبط با کامپیوتر ویژن هست. پس ورودی

همیشه عکس هست هر چند برای پردازش متن هم استفاده می شود. همانطور که در شکل

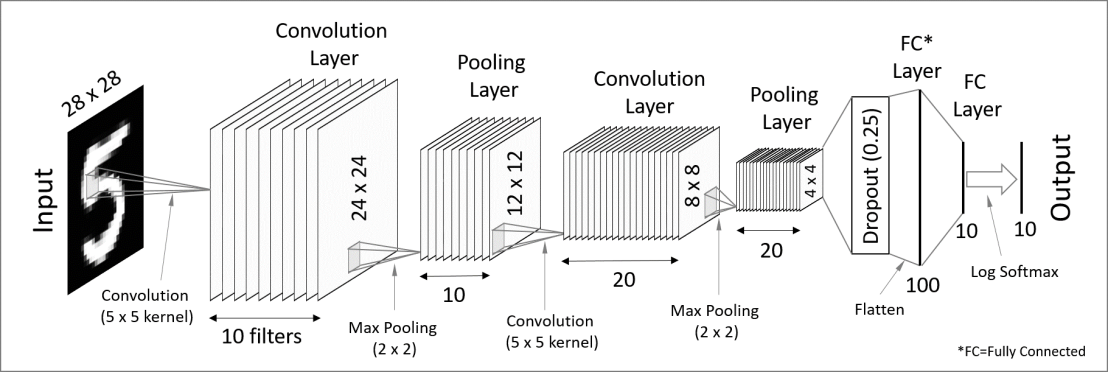
مشاهده می کنید چیزی که ما انسان ها می بینیم تعدادی سگ در یک هست ولی چیزی که

که کامپیوتر تشخیص می دهد مک ماتریس هست. این عکس ها می توانند سیاه و سفید یا

Gray، 2 بعدی (طول و عرض) باشند و یا رنگی که 3 بعدی(طول و عرض و عمق) هست داشته

باشند و عکس رنگی 3 تا کانال دارد که R به معنای قرمز و G سبز و B ابی هست.

**Input is image**



اگر توی این شکل نگاه کنید یک مدل کانولوشن Neural Network هست و ورودی ما به حالت

یک بردار نیست به حالت یک ماتریس 2 بعدی هست که سیاه و سفید هست به همین علت

عمق عکس 1 هست (1\*28\*28) که می شود ورودی ورودی مدل. توی CNN ها 4 نوع لایه

داریم. در بعدی از لایه ورودی، لایه Convolutional Layer هست و بعد از ان Pooling Layer

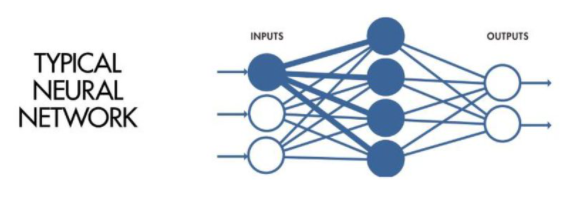
هست و ما میتوانیم به هر تعداد که لازم باشد این لایه را روی شبکه ی خودمون اعمال کنیم.

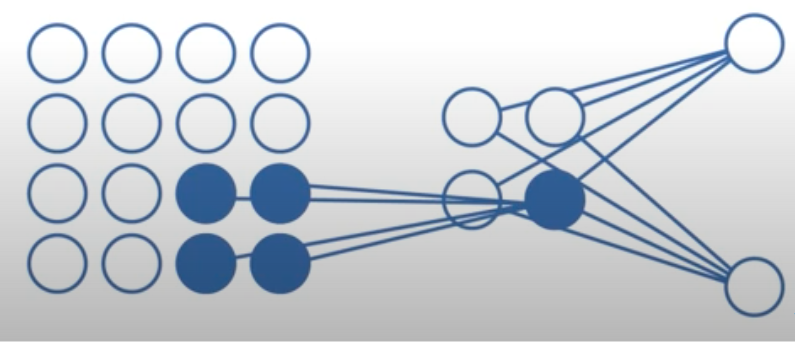
و پس از این لایه ها 1 تا 2 لایه Fully connected layer (FC) داریم و پس از این لایه ، لایه

خروجی هست.

**ConvNets vs Regular-NN**

Local Receptive Field





توی CNN ها یک پیکسل با پیکسل های اطراف خودش ارتباط بیشتری دارند و هر چقدر پیکسل

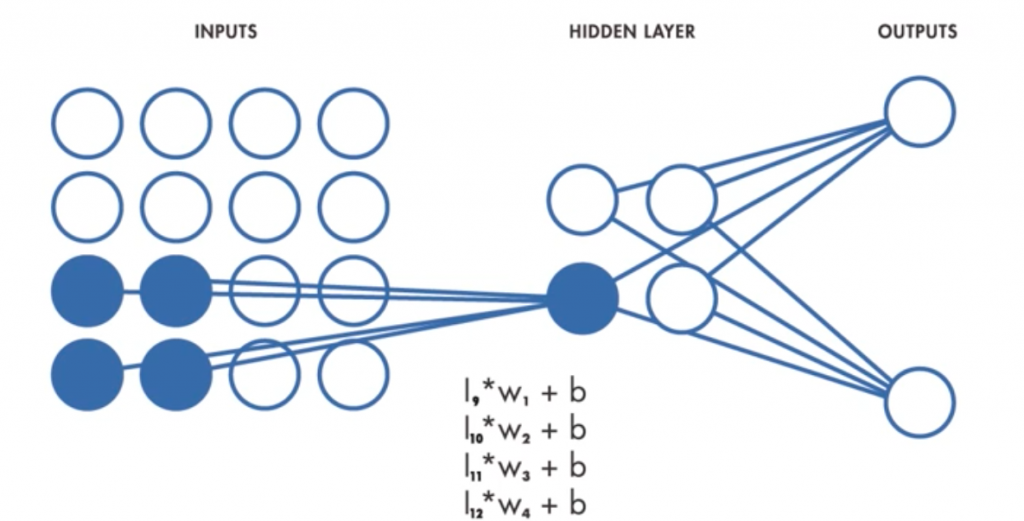
ها از هم دورتر باشند ارتباط ضعیف تر می شود. در NN هر نورون ورودی به نورون های لایه

مفخی بعدی وصل هست. ولی توی CNN ها یک قسمت کوچکی از ورودی به لایه مفخی بعدی

وصل هست. همانطور که توی عکس می بینید به این ناحیه Local Receptive Field و کاری که

می کند روی ورودی Slide می کند تا همه ورودی را پوشش دهد و به لایه مفخی وصل کند.

Shared Weights and Biases

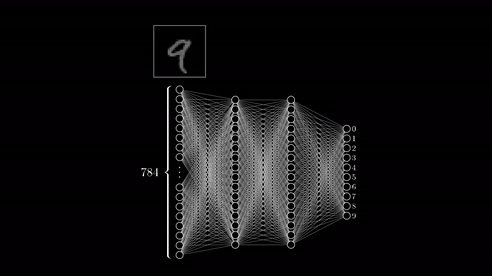


تفاوت بعدی CNN ها با NN دارند بر خلاف NN معمول توی هر نورون توی هر لایه به تمام نورون

های لایه بعدی وصل هست و برای خودش یک سری وزن و بایاس متفاوت دارد. اما توی CNN ها

همانطور که در شکل می بینید تمام وزن ها و بایاس ها ثابت هستند.(اگر به اعمال ریاضی توجه کنید).

**ConvNets-Feature-Learning**



Hidden Layer توی شبکه های عصبی یک سری ویژگی یا Feature از ورودی(عکس) برای ما

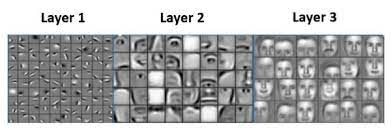
پیدا می کند. اگر بخواهیم روی این عکس Classification انجام بدهیم لایه مفخی اول که بعد از

لایه ورودی هست میاد یه تیکه هایی یا خط هایی از این عدد را پیدا می کند و توی لایه مفخی

بعدی این این ویژگی کامل تر می شود مثلا میاد شکل دایره عدد 9 را پیدا می کند و یا قسمت

پائین عدد را پیدا می کند. مثلا اگر لایه های مفخی بیشتری داشته باشیم ویژگی های بیشتری

از عکس را برای ما پیدا می کند و کامل تر می شود.



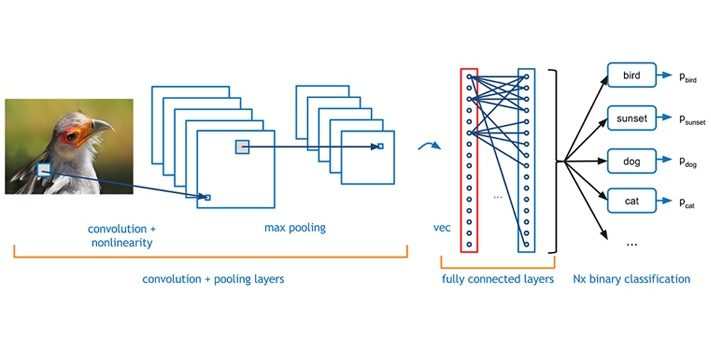
مثلا ما توی عکس بالا از 3 لایه Hidden Layer استفاده کردیم و همانطور که می بینید در لایه 1

یک قسمت هایی از تصویر را نمایش می دهد و در لایه 2 ویژگی های بیشتری از عکس رانمایش

می دهد و در لایه 3 فیچر ها یا ویژگی های بیشتری را نمایش می دهد و پیشرفت بهتری را

دارد و عکس را با تمام ویژگی ها و بصورت کامل نمایش می دهد.

**Convolutional Neural Network Architecture**



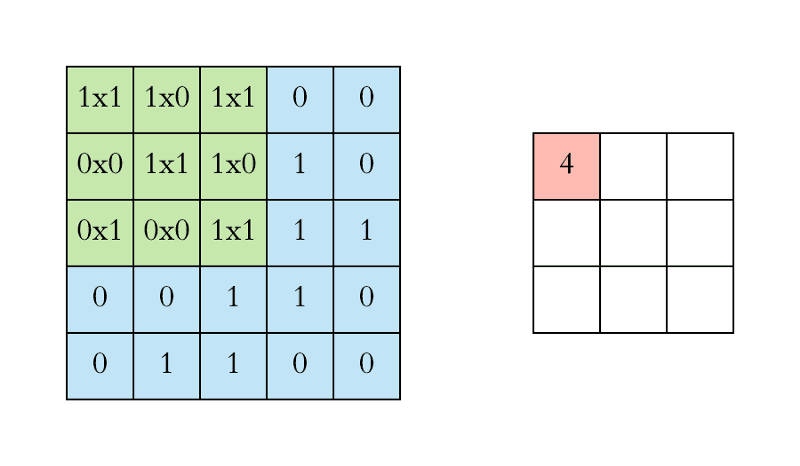
اگر به مدل نگاه کنید ورودی ما یک عکس رنگی هست بعدش یک تعداد لایه کانولوشن داریم که

یک سری عملیات Pooling را انجام می دهد. که بستگی به نیاز این لایه را میتوانیم بیشتر کنیم

و بعد از ان هم 2 تا لایه Fully Connected داریم که نورون های هر لایه به نورون های لایه بعدی

وصل هست .

**Convolutional Layer**



توی Convolutional Layer یک ورودی 5 \* 5 داریم و با استفاده از Kernel ، 3 \* 3 کار فیلترینگ را

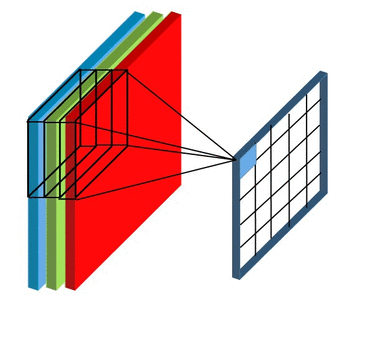
انجام می دهیم که این کار فیلترینگ به صورت چرخیدن یا لغزیدن هست که اصطلاحا به ان Slide

می گویند و عملیات ضرب را انجام می دهد و نهایتا یه خروجی 3 \* 3 به ما می دهد و این خروج

ی اصطلاحا Feature Map نام دارد. و اینکه چند فیلتر باید انجام دهیم خودمان تعیین می کنیم و

می توانیم هر چقدر فیلتر که بخواهیم انجام دهیم. مثلا اگر 100 تا فیلتر انجام دهیم یک ماتریس

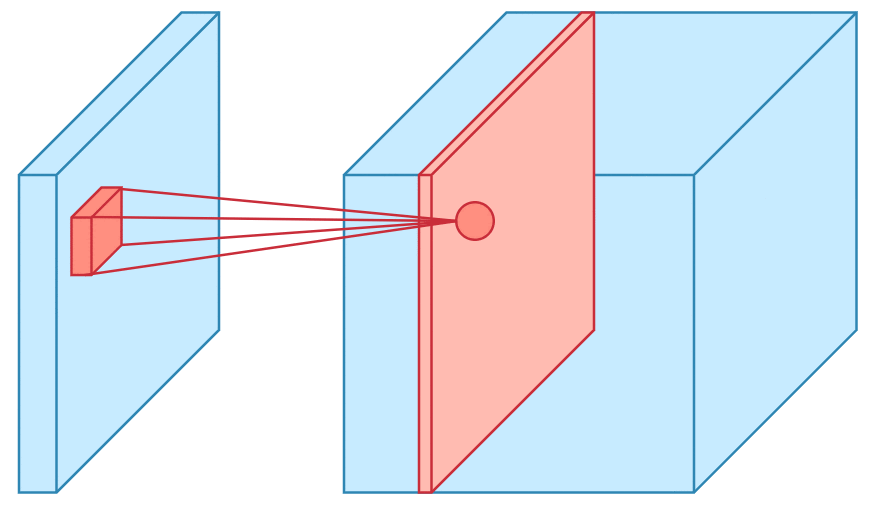
100\*3\*3 داریم که 100 عمق ماتریس هست.



اگر ما 3 تا کانال 5 \* 5 داشته باشیم عملیات فیلترینگ روی هر 3 انجام می شود و جواب هر 3 با

هم جمع می شود و یک ماتریس خروجی به ما می دهد.

**Convolutional Layer**

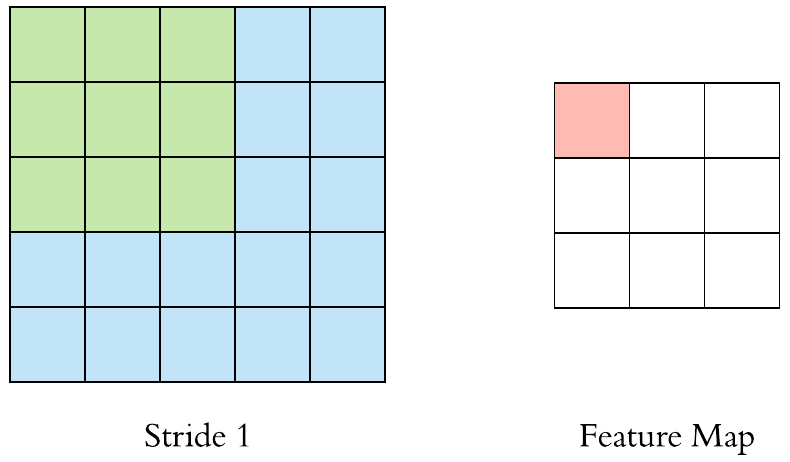


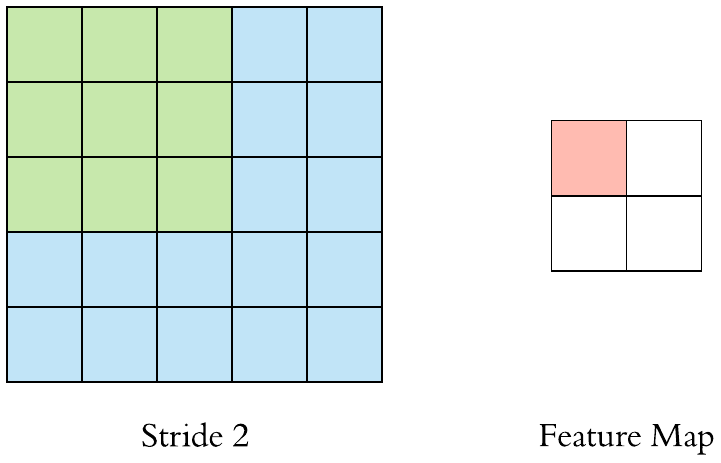
زمانی که ماتریس خروجی را بعد از فیلترینگ گرفتیم کاری را که باید انجام بدهیم باید عملیات

تابع Activation را روی این ماتریس انجام بدهیم که معمولا از تابع Relu استفاده می کنیم و این

می شود خروجی .

**Stride**





زمانی که عملیات MaxPooling را انجام می دهیم ماتریس کرنل را یک دانه یک دانه می لغزیدیم

و این کار را هم می شود 2 تا 2 تا انجام داد و این کار را هر چند تعداد میتوانیم انجام دهیم به این

کار اصطلاحا Stride می گویند. مثلا اگر 2 تا 2 تا حرکت بدیم Stride ما سایزش 2 هست اگر

یکی یکی انجام بدهیم Stride ما سایزش 1 هست. همیشه ماتریس Feature Map بعدش از

ورودی اولیه کمتر هست اگر توجه کنید ورودی من بعدش 5 \* 5 هست و وقتی که کار فیلتر را

انجام می دهیم خروجی ما یک ماتریس 3 \* 3 هست حالا اگر ما Stride را برابر 2 قرار دهیم

سایز Feature Map ما 2 \* 2 می شود . معمولا سایز Stride را 1 می گذارند.

**Padding**



موضوع بعدی که ما توی لایه Convolution Layer انجام می دهیم Padding هست. که دور

ماتریس ورودی یک سری سطر و ستون از 0 اضافه می کنیم که علتش این هست که وقتی ما

عملیات فیلترینگ را انجام می دهیم و سایز خروجی Feature Map نسبت به ورودی کوچکتر

هست و بعضی مواقع ما میخواهیم که سایز ماتریس خروجی برابر با ماتریس ورودی باشد به

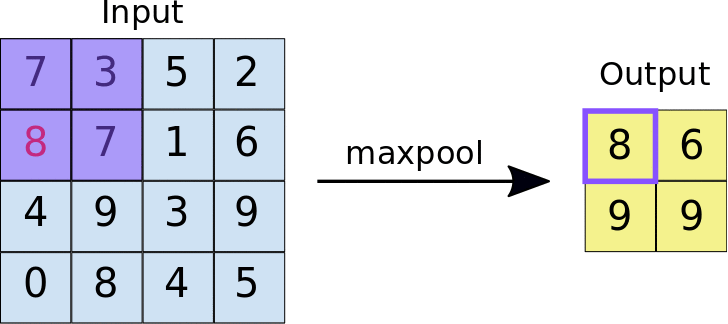
همین دلیل این عملیات را انجام می دهیم. مثلا اگر قسمت های ابی رنگ ورودی ما 5 \* 5

هست. وقتی 0 را اضافه می کنیم به جای اینکه Sliding را از عدد 60 شروع کند Sliding را

از عدد 0 سطر اول شروع می کند. وقتی که Feature Map ما بزرگتر باشد یک سری اطلاعات

بیشتری از ورودی درسافت می کنیم.

**Max Pooling Layer**



زمانی که Convolution Layer ,Filtering ,Stride ,Padding یک Feature Map خروجی داریم

مثلا ما اگر 100 Feature Map را پشت سر هم Stack می کنیم که این می شود ورودی ما برای

لایه بعد و در ادامه ما می توانیم یک لایه ای از Pooling را روی ورودی اعمال کنیم.

Pooling هیچ تاثیری از لحاظ Learning و Training روی مدل ما ندارد. تنها کارایی ان ورودی را

دریافت می کند و ابعادش را کوچک می کند. به صورت یک ماتریس 2 \* 2 روی ورودی عملیات

MaxPooling را انجام می دهد. و در هر Step بزرگترین عدد انرا برای ماتریس خروجی انتخاب

می کند و در نهایت یک ماتریس 2 \* 2 را به عنوان خروجی می دهد.

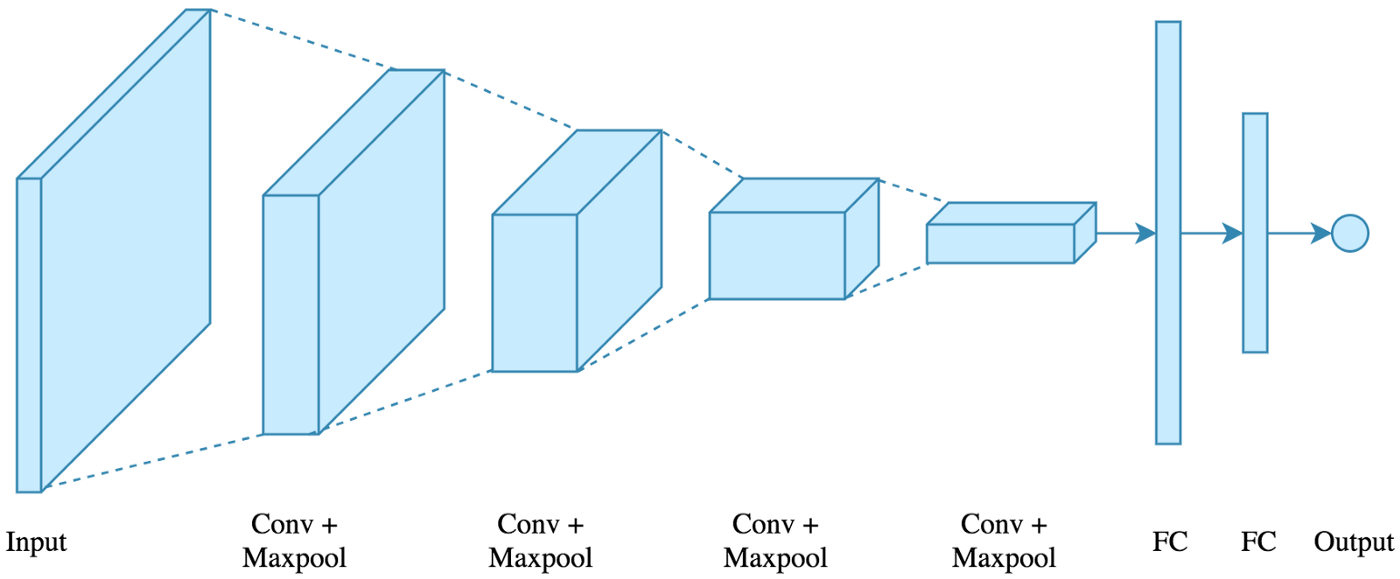
علت اینکه ما عملیات MaxPooling را انجام می دهیم و ورودی را کوچک می کنیم. این هست

که ما پارامتر ها را کمتر می کنیم و وقتی تعداد پارامترها کمتر باشد مدت زمان Training کمتر

می شود و از Overfitting جلوگیری می کند و ما میتوانیم این لایه ها را چند بار تکرار کنیم و هر

دفعه که تکرار می کنیم خروجی ما کوچکتر می شود.

**Fully connected Layer**



بعد از چند بار تکرار Convolutional Layer و Pooling Layer در ادامه ما 1 الی 2 تا FC Layer

اضافه می کنیم و بعد از FC هم لایه خروجی داریم. چیزی را که باید توجه کنیم FC ها بصورت

یک بردار هستند. یعنی ورودی باید تا قبل از FC باید بصورت یک بردار باشد.لایه هایConvolution

و MaxPooling به صورت مکعب هستند که شامل طول و عرض و عمق هستند. قبل از اینکه لایه

های قبلی را به FC وصل کنیم این لایه ها را Flatten می کنیم و تبدیل می کنیم به یک بردار و

بردار را لایه FC وصل می کنیم. همانطور که می بینید لایه Convolutional و MaxPooling به

صورت مکعب هستند زیرا 3 بعدی هستند و دارای 3 کانال RGB هستند. به همین خاطر که این

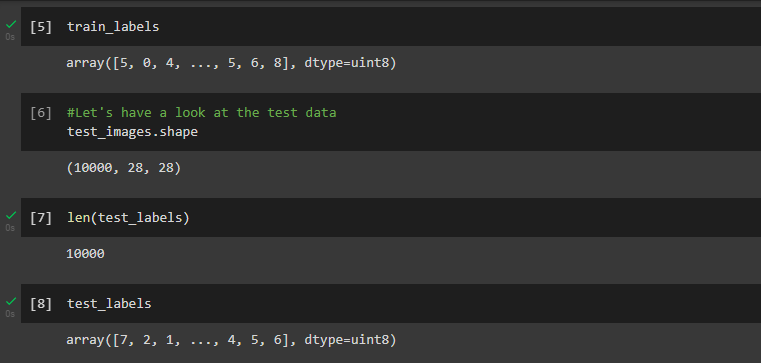
لایه ها دارای ابعاد مکعبی شکل هستند و ما اونقدر عملیات MaxPooling و Convolutional روی

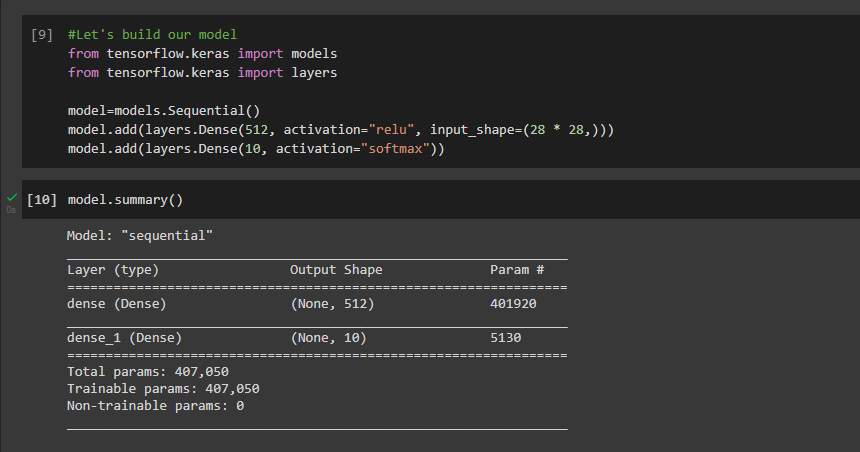
این لایه ها انجام می دهیم که از لحاظ طول و عرض کمتر می شود. اما قطر لایه ما ضخیم تر

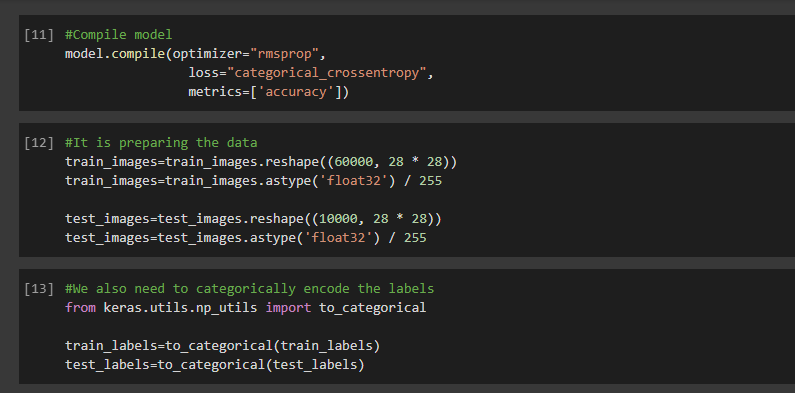
می شود . و در پایان هم لایه خروجی هست که تصویر را به ما نمایش می دهد.

**A first look at a neural network**

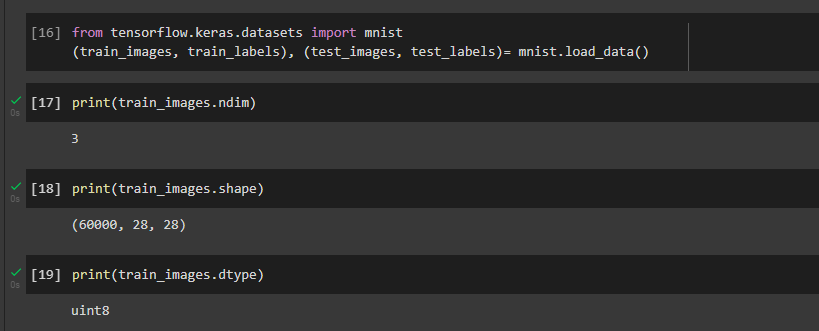


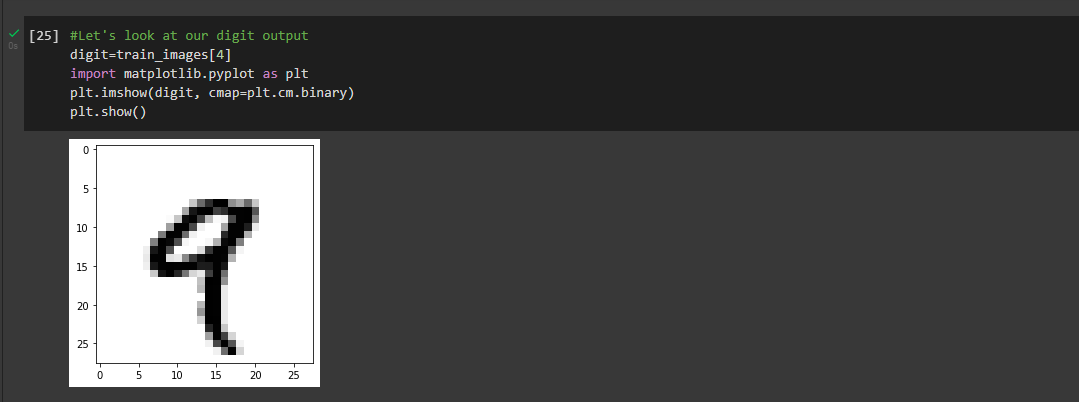












References

* <https://ieeexplore.ieee.org/document/8308186>
* <https://arxiv.org/pdf/1512.07108.pdf%C3%A3%E2%82%AC%E2%80%9A>
* <https://lamiae-hana.medium.com/an-intro-to-convolutional-neural-networks-cnn-9f1c2d888fa1>
* <https://www.geeksforgeeks.org/applying-convolutional-neural-network-on-mnist-dataset/>
* <https://en.wikipedia.org/wiki/Convolutional_neural_network>