به نام خدا

قدم دوم)

```
feed_forward(W1, W2, W3, b1, b2, b3)
```

```
Accuracy: 0.14
```

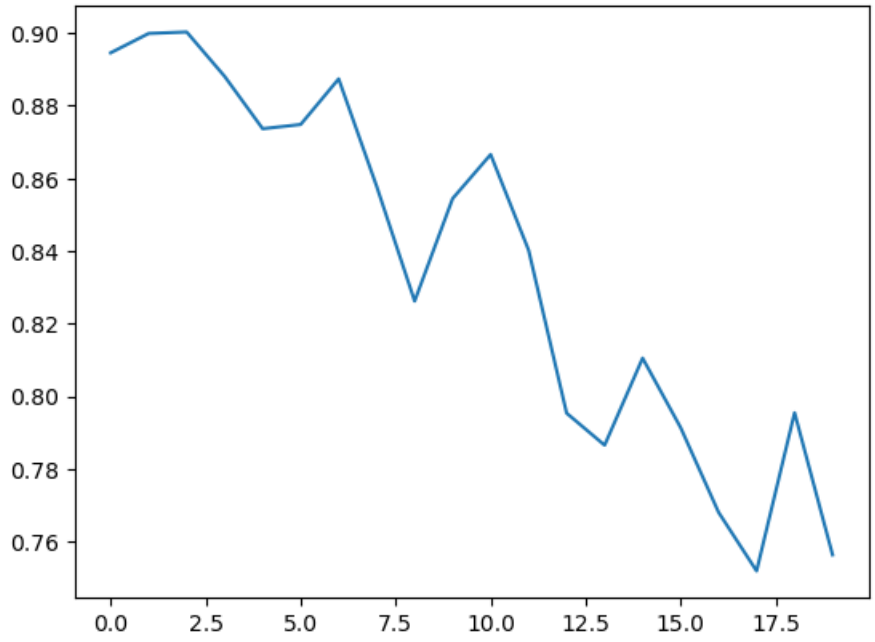
قدم سوم – backpropagation)

ایپاک = ۲۰

```
Accuracy without vectorization - samples = 100 | epoch = 20: 0.38
```

```
epochs = [i for i in range(epoch)]
plt.plot(epochs, costs)
```

```
[<matplotlib.lines.Line2D at 0x1c7a34b3490>]
```




```
print(f'Execution time: {stop-start}')
```

```
Execution time: 38.654752254486084
```

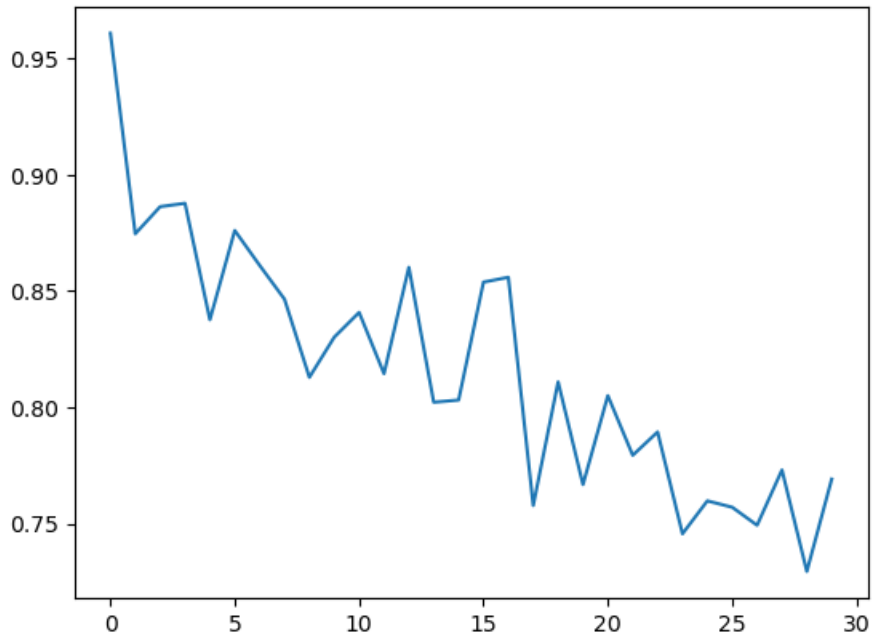
ایپاک = ۳۰

```
Accuracy without vectorization - samples = 100 | epoch = 30: 0.4
```

```
epochs = [i for i in range(epoch)]
plt.plot(epochs, costs)

print(f'Execution time: {stop-start}')
```

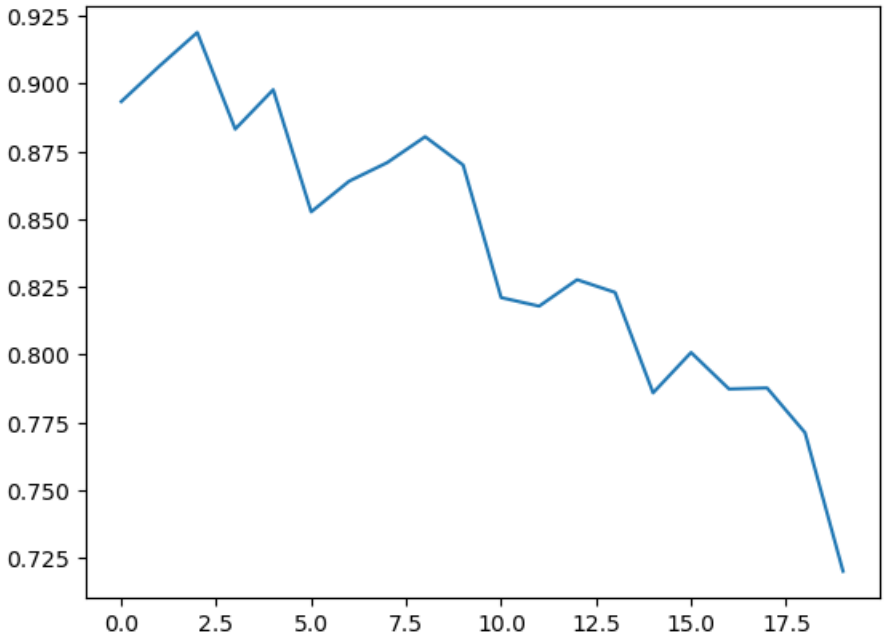
```
Execution time: 58.762372970581055
```



قدم چهارم – vectorization)

ایپاک = ۲۰

```
Accuracy with vectorization - samples = 100 | epoch = 20: 0.45
```

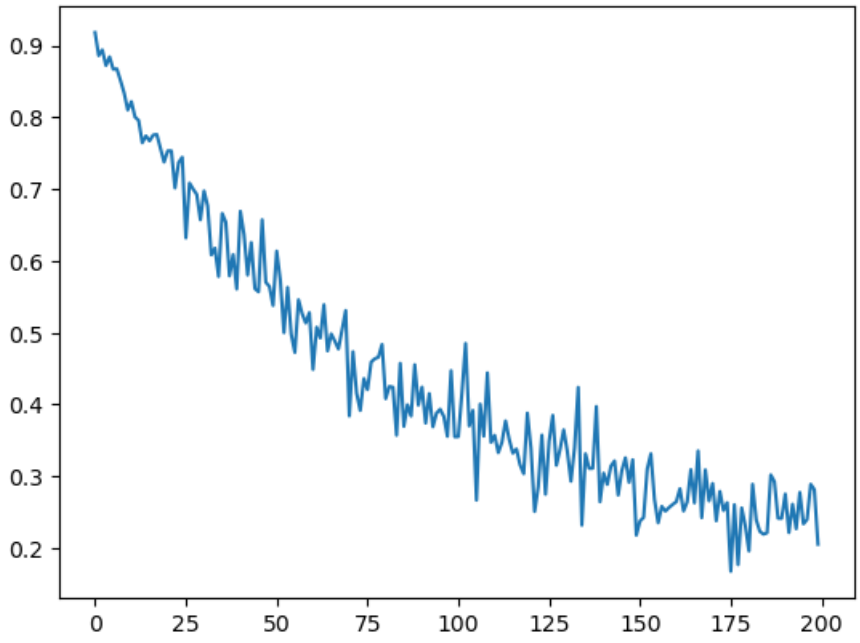



```
print(f'Execution time: {stop-start}')
```

```
Execution time: 0.32404446601867676
```

ایپاک = ۲۰۰

```
Accuracy without vectorization - samples = 100 | epoch = 200: 0.89
```



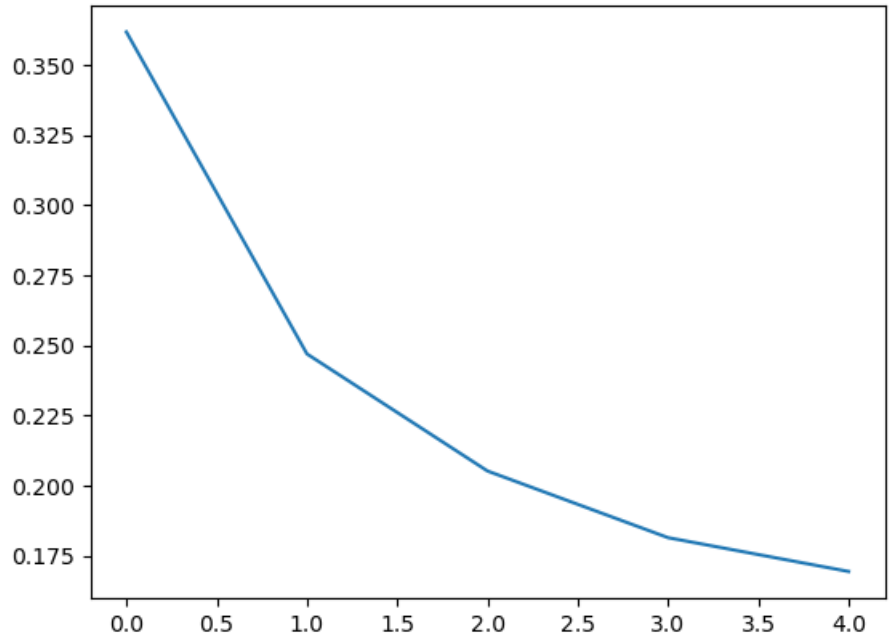
```
print(f'Execution time: {stop-start}')
```

```
Execution time: 3.231027603149414
```

قدم پنجم- تست

```
Train_set Accuracy with vectorization - samples = ALL | epoch = 5 | batchsize = 50: 0.89145
Test_set Accuracy with vectorization - samples = ALL | epoch = 5 | batchsize = 50: 0.8933
```

```
Execution time: 35.79459190368652
```



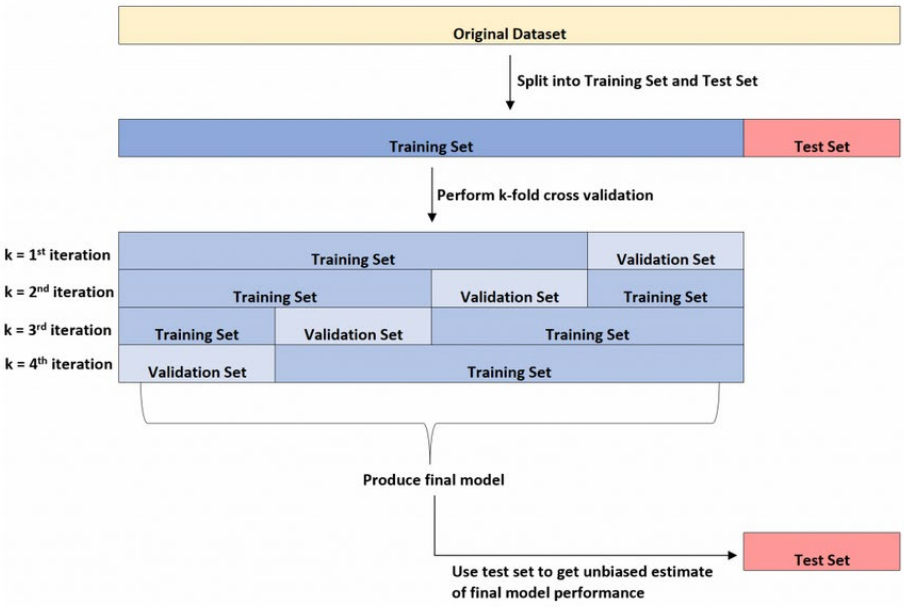
بخش تحقیقی امتیازی

سوال اول)

مجموعه داده به دو قسمت اصلی داده آموزشی و تست تقسیم می شود. داده آموزشی برای این است که شبکه مان را آموزش دهیم و داده تست تنها برای این است که نتیجه مدلمان را ببینیم؛ ولی ما نباید با نتایجی که از داده تست بدست می آوریم سعی کنیم پارامتر ها را تغییر دهیم و داده تست تنها برای دیدن نتیجه است. اما داده دیگری هم به نام داده اعتبار سنجی داریم که قسمتی از داده است که برای ارزیابی مدل هنگام تنظیم مدل استفاده می شود. این ارزیابی مقدار هایپرپارامتر های مدل را تعیین می کند. همچنین جدا کردن این داده باعث جلوگیری از overfitting می شود. KFold CV این گونه کار می کند که داده را به K قسمت تقسیم می کند. یکی از گروه ها را کنار

گذاشته آن را به عنوان تست در نظر می گیرد و باقی گروه ها را به عنوان مجموعه آموزشی و سعی میکند مدلی را روی داده آموزشی فیت کند و آن را روی تست امتحان کند و نتایج را بگیرد.

معمولا داده ها رو چند بار به خورد شبکه میدن تا از روی اون یاد بگیره و وزن ها اپدیت بشن اما از یه جایی به بعد مدل زیادی فیت میشه (overfit) اینجاست که از داده اعتبارسنجی استفاده می کنیم. در هر ایپاک، مدل را روی داده های اعتبارسنجی تست می کنیم و دقت مدل رو در میاریم.تا یه جایی با کم شدن خطای مدل روی داده های آموزشی، خطای داده های اعتبارسنجی هم پایین میاد اما از یه جا به بعد، بیش برازش اتفاق میوفته و تو این نقطه است که اموزش باید متوقف بشود



سوال دوم)

در گریان کاهشی دسته ای تمام داده های اموزشی در یک قدم در نظر گرفته می شوند. م متوسط گرادیان های تمام داده های آموزشی را گرفته و سپس، از آن برای آپدیت پارامتر ها استفاده می کنیم. پس در حقیقت یک قدم از گرادیان کاهشی است که در یک ایپاک است.

مزیت: نوسان های کمتر

مزیت استفاده از دستورات برداری -> افزایش سرعت

یک همگرایی گرادیان کاهشی پایدار تری را تولید میکند

از نظر محاسباتی به صرفه چون همه ی منابع کامپیوتر دنبال پردازش یک نمونه نیستند بلکه دنبال همه ی نمونه داده ها اند

معایب: گاهی گرادیان کاهشی پادیار ممکن است باعث مینیمم محلی شود و علیرغم الگوریتم SGD قدم های نویزی ای وجود ندارد که ما را از از مینمم محلی خلاص کند.

پردازش همه ی داده ممکن است خیلی زیاد باشد برای مموری و حافظه اضافی نیاز شود.

پردازش همه ی داده ها ممکن است خیلی طول بکشد

گرادیان کاهشی تصادفی: در گرادیان کاهشی دسته ای، ما تمام نمونه ها را برای هر قدم گرادیان کاهشی درنظر می گرفتیم ولی اگر مجموعه داده مان خیلی بزرگ باشد، این عمل زیاد کارآمد نیست. به جای آن در این الگوریتم، ما هر بار یک نمونه را در یک قدم در نظر می گیریم. ما کارهای زیر را در یک اپیاک انجام می دهیم.

یک نمونه میگیریم – آن را به شبکه می دهیم – گرادیان آن را حساب میکنیم – از این گرادیان استفاده و وزن ها را اپدیت می کنیم. – مراحل قبلی را برای تمام داده ها در مجموعه داده تکرار می کنیم. چون ما فقط یک نمونه را در هر بار در نظر می گیریم، هزینه در هنگام آموزش داده های آموزشی نوسان می کند و لزوما کاهش پیدا نمی کند اما در اجرای طولانی می بینیم که هزینه با نوسان کاهش می یابد. و چون هزینه خیلی نوسان می کند هیچوقت به مینیمم نمیرسد اما حدود آن نوسان می کند. این الگوریتم برای مجموعه داده های بزرگتر می تواند استفاده شود چون وقتی مجموعه داده بزرگ تر باشد باعث می شود اپدیت های بیشتری صورت گرفته و سریع تر همگرا شود.

مزایا: راحت تر به مموری فیت می شود چون یک نمونه را میاورد

از نظر محاسباتی چون در هر بار یک نمونه پردازش می شود سریع است

برای مجموعه داده های برگ سریع تر همگرا می شود

به خاطر اپدیت های مکرر، قدم ها به سمت مینیمم تابع هزینه می روندکه باعث خلاص شدن از مینیمم محلی می شود

معایب: اپدیت های مکرر -> قدم ها خیلی نویزی اند

همین ممکن است باعث دیرتر همگرا شدن بشود.

اپدیت های مکرر پر هزینه اند

مزیت عملگرهای برداری را از دست می دهد.

گرادیان کاهشی دسته ای کوچک:BGD برای منحنی های نرم تر استفاده می شود (نرم تر بودن به دلیل میانگین گیری روی تمام داده ها در یک قدم) و SGD نیز برای مجموعه داده های بزرگ استفاده می شود. BGD مستقیم به مینیمم همگرا می شود و SGD سریع تر همگرا می شود اما می تواند سرعت پایینی داشته باشد برای همین از ترکیب این دو استفاده می کنیم. یعنی یک دسته ای را در نظر می گیریم از تعداد ثابتی از نمونه های داده آموزشی که زیر مجموعه ای کوچکتر از مجموعه اصلی است. سپس بعد از درست کردن دسته های کوچک قدم های زیر را در یک ایپاک انجام می دهیم:

یک دسته کوچک انتخاب – به شبکه عصبی می دهیم – میانگین گرادیان دسته های کوچک محاسبه – از آن برای اپدیت وزن ها استفاده – قدم های فوق را برای دسته هایی که ساختیم تکرار می کنیم. مانند SGD هزینه متوسط در ایپاک های گرادیان دسته ای کوچک نوسان می کند چرا که ما روی تعداد کوچکی از نمونه ها در یکبار میانگین می گیریم.

مزایا: ترکیبی از موارد فوق: به مموری فیت ب راحتی فیت می شود؛ ا نظر محاسباتی کارآمد، استفاده از vectorization که باعث افزایش سرعت می شود؛ اگر در مینمیمم محلی گیر کنیم، بعضی قدم های نویزی می توانند از آن ها خارج شوند؛ متوسط دادهای آموزشی، ارور پایدار گرادیان و همگرایی را تولید می کند

بدی ها: loss برای هر دسته کوچک جدا استفاده می شود و در نتیجه هزینه نهایی انباشته شده همه دیسته های کوچک می شود.

سوال سوم)

نرمال سازی دسته ای متود الگوریتمی است که آموزش شبکه عمیق را سریع تر و پایدارتر می کند. و این کار را به کمک نرمال سازی ورودی های لایه ها با دوباره مرکز گیری و rescalingانجام میدهد.

برای افزایش پایداری یک شبکه یادگیری عمیق، نرمال سازی دسته ای با کم کردن میانگین دسته ای، و سپس تقسیم بر انحراف استاندارد دسته، بر خروجی لایه فعال سازی قبلی تأثیر می گذارد.

از آنجایی که این تغییر یا مقیاس بندی خروجی ها توسط یک پارامتر به طور تصادفی اولیه شده، دقت وزن ها را در لایه بعدی کاهش می دهد، یک گرادیان تصادفی کاهشی برای حذف این نرمال سازی در صورتی که تابع هزینه خیلی زیاد باشد، اعمال می شود.
نتیجه نهایی این است که نرمال سازی دسته ای دو پارامتر قابل آموزش اضافی را به یک لایه اضافه می کند: خروجی نرمال شده که در یک پارامتر گاما (انحراف استاندارد) ضرب می شود و پارامتر بتا (میانگین) اضافی. به همین دلیل است که نرمال سازی دسته ای همراه با گرادیان کاهشی کار می کند، به طوری که داده ها را می توان با تغییر فقط این دو وزن برای هر خروجی، «غیر عادی» کرد.

این منجر به از دست دادن داده های کمتر و افزایش پایداری در سراسر شبکه با تغییر تمام وزن های مرتبط دیگر می شود.
شامل بردارهای فعالیت نرمال سازی از لایه های پنهان است که از میانگین و واریانس دسته فعلی استفاده می کنند. این تکنیک نرمال سازی دقیقا قبل یا دقیقا بعد اجرای تابع غیرخطی استفاده می شود. نرمالسازی دسته ای به صورت متفاوتی در هنگام آموزش و تست استفاده می شود.

سوال چهارم)

بعد از هر بار عمل کانولوشن که در خروجی به ما یک نقشه‌ی ویژگی (Feature Map) می‌دهد یک بار فرایند ادغام (Pooling Layer) انجام می‌شود.

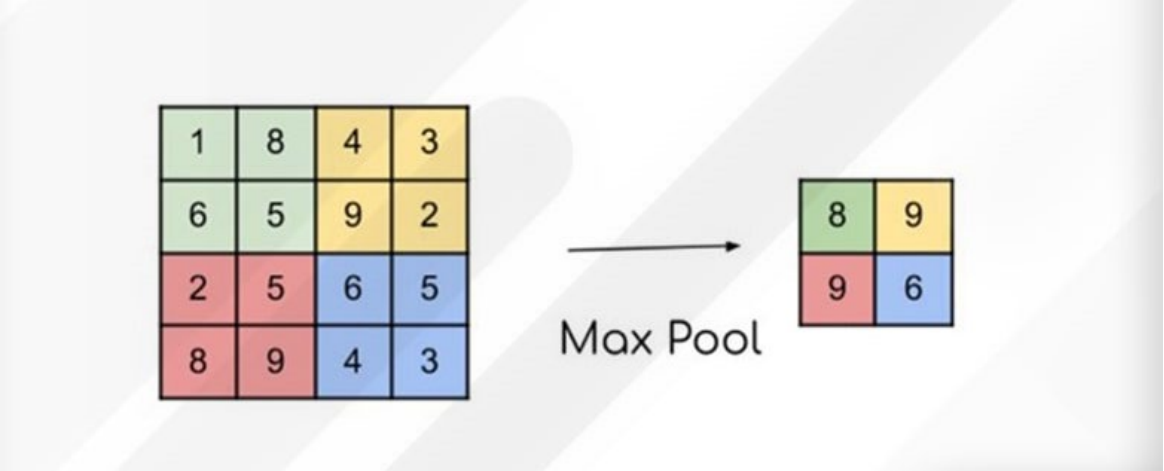
- لایه ادغام اندازه‌ی نقشه‌های ویژگی (Feature Maps) را کاهش می‌دهد که این خود تعداد پارامترهای یادگیری و هم‌چنین میزان محاسبات شبکه را کاهش می‌دهد.
- لایه ادغام ویژگی‌های موجود در هر ناحیه از نقشه‌ی ویژگی را که لایه‌ی کانولوشن تولید کرده است خلاصه می‌کند؛ درواقع مهم‌ترین آن‌ها را انتخاب می‌کند و به مرحله‌ی بعد منتقل می‌کند. این باعث می‌شود مدل در برابر تغییرات موقعیت ویژگی‌های موجود در تصویر ورودی مقاومت بیشتری داشته باشد.

چندین نوع ادغام مختلف وجود دارد: max polling, min pooling, sum pooling

در این نوع ادغام بزرگ‌ترین مقدار در ناحیه‌ای را که فیلتر پوشانده است انتخاب می‌شود؛ بنابراین در این حالت خروجی یک نقشه‌ی ویژگی (Feature Map) است که برجسته‌ترین ویژگی‌های نقشه ویژگی (Feature Map) قبلی را دارد. لایه‌ی ادغام با کوچک‌کردن اندازه‌ی عکس ورودی و خلاصه‌سازی ویژگی‌های اصلی و مهم موجود در عکس، به کاهش محاسبات شبکه و مشکل Overfittingکمک می‌کند؛ به‌همین دلیل، یکی از مهم‌ترین فرایندها در شبکه عصبی کانولوشنی (CNN / Convolutional Neural Network)محسوب می‌شود.

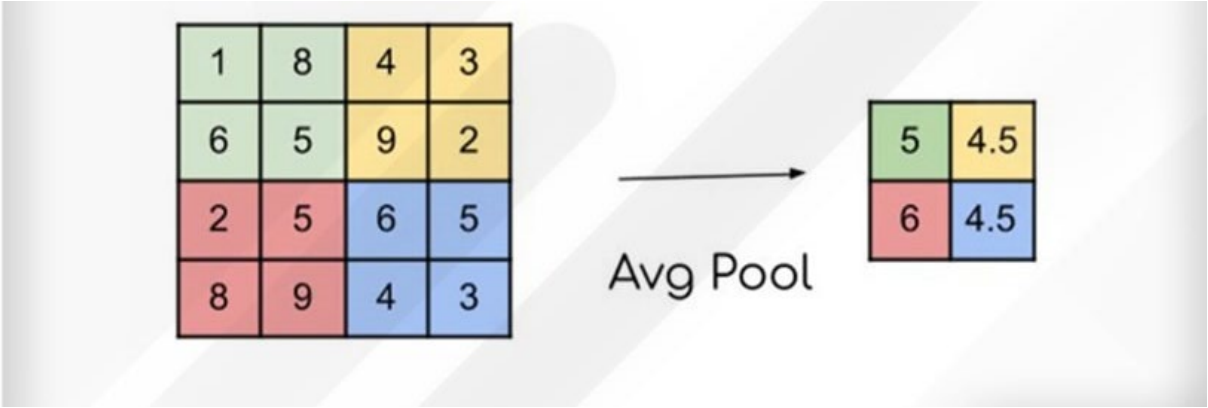
- Max Pooling

در این نوع ادغام بزرگ‌ترین مقدار در ناحیه‌ای را که فیلتر پوشانده است انتخاب می‌شود؛ بنابراین در این حالت خروجی یک نقشه‌ی ویژگی (Feature Map) است که برجسته‌ترین ویژگی‌های نقشه ویژگی (Feature Map) قبلی را دارد.



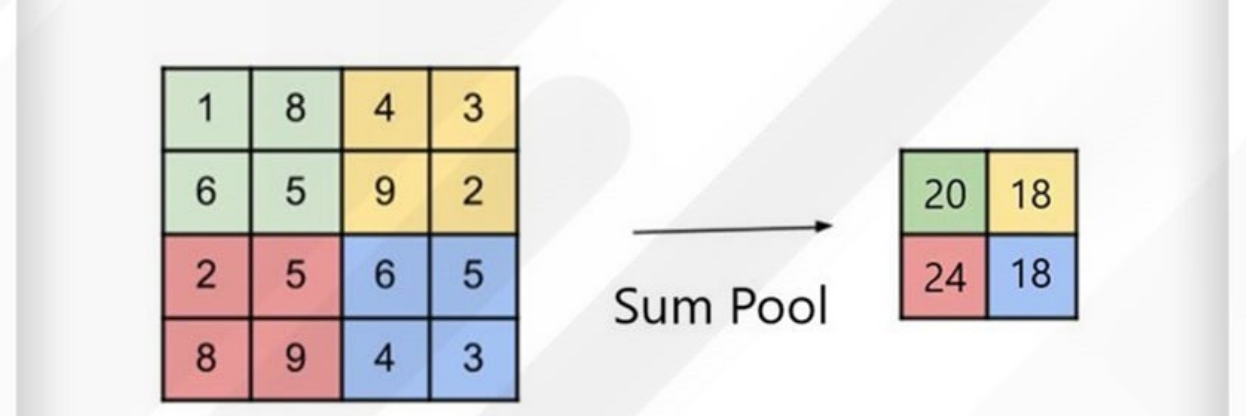
- Average Pooling

در این نوع ادغام مقدار میانگین ناحیه‌ای که فیلتر روی آن قرار می‌گیرد محاسبه می‌شود؛ بنابراین میانگین ویژگی‌های نقشه‌ی ویژگی قبلی را در خروجی ارائه می‌کند.



- Sum Pooling

در این نوع ادغام (Pooling) جمع کل ناحیه‌ای که فیلتر پوشانده است محاسبه می‌شود و یک نقشه‌ی ویژگی جدید را ایجاد می‌کند



شبکه های کانولوشنی از این جهت بهتر است که می تواند روابط بین پیکسل های کنار هم را بررسی کند و بهتر این کار را انجام دهد و در نتیجه قدرت تشخیص آن بالاتر می رود؛ همچنین حالت های مختلف یک عکس را می تواند با کارهایی مثل پولینگ در نظر بگیرد که در نتیجه قدرت تشخیص بالاتر می رود. همچنین می تواند چندین لایه از فیچر را با فیلتر کردن یاد بگیرد. همچنین تعداد پارامتر های شبکه برای یادگرفتن کمتر است و در نتیجه شانس بیش برازش کاهش می یابد.