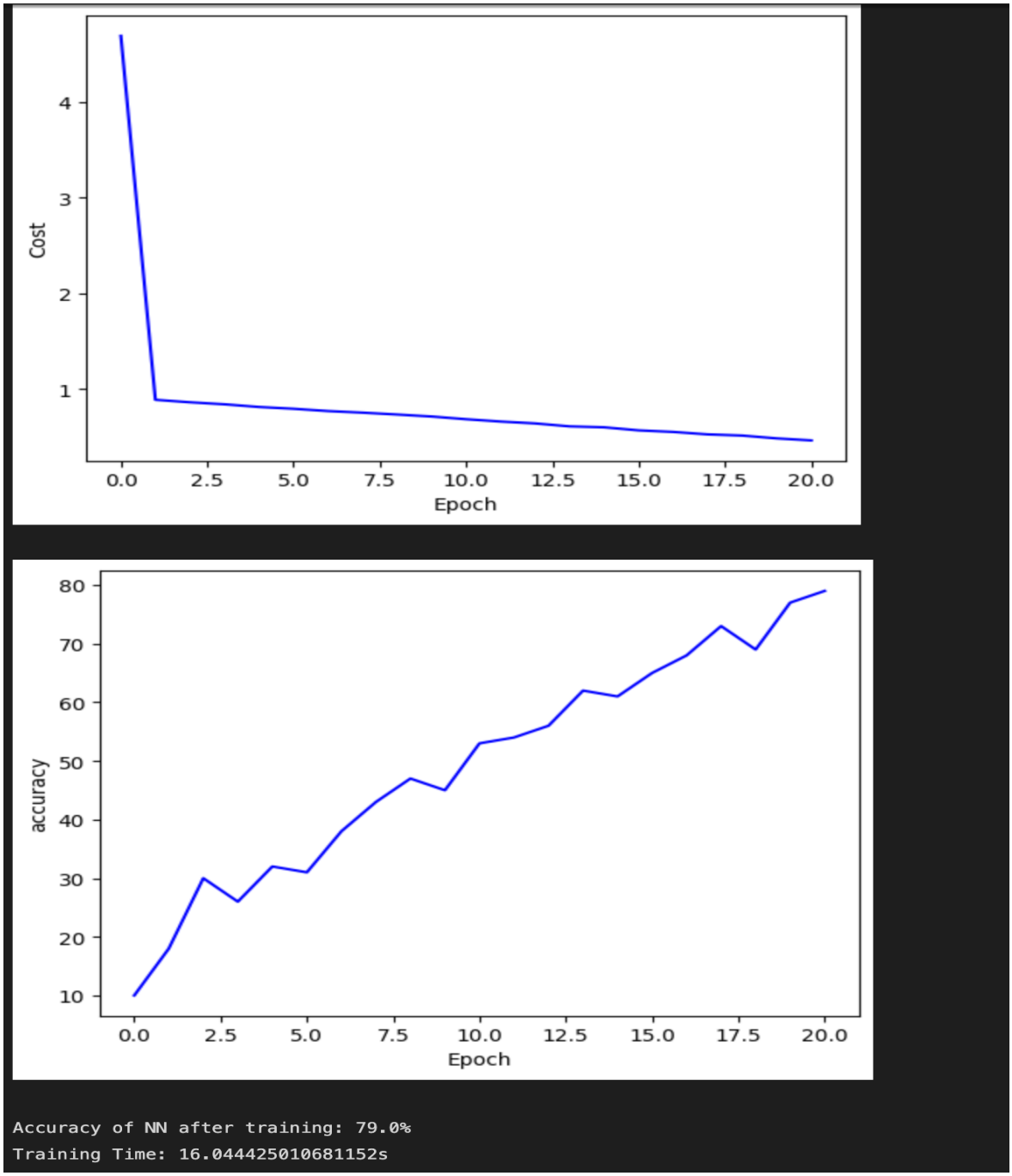
گزارش کار پروژه اول هوش محاسباتی

امیررضا طربخواه ۹۸۳۱۰۴۱

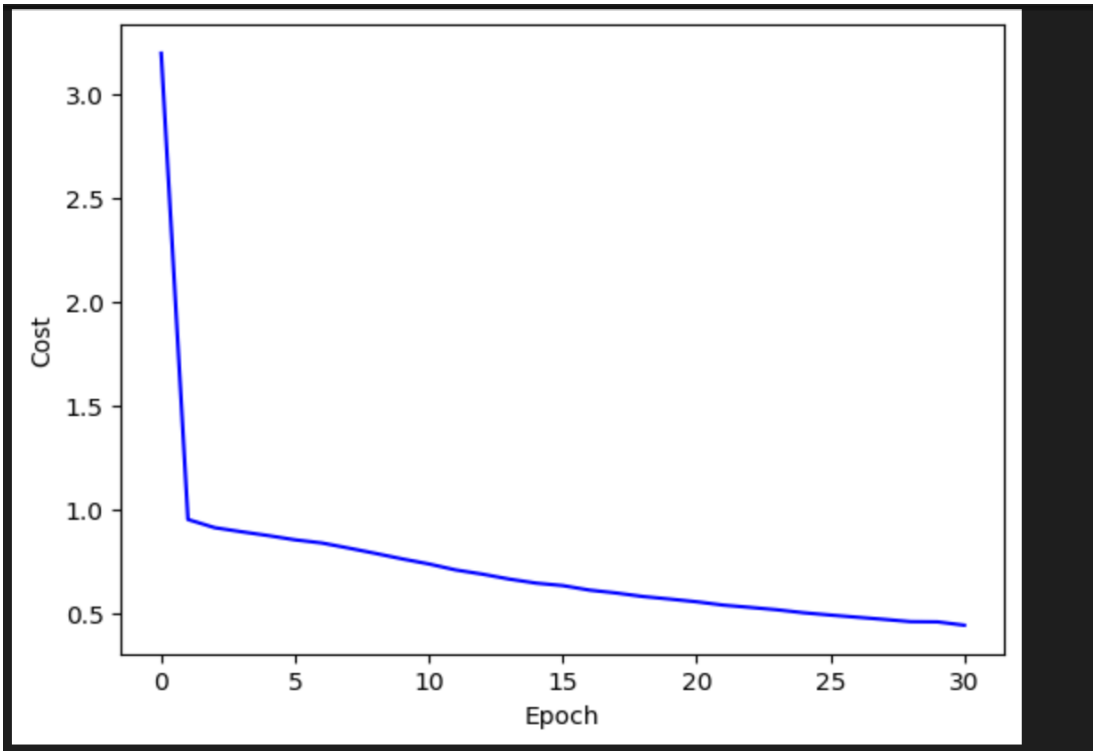
برای دقت feed forward داریم:

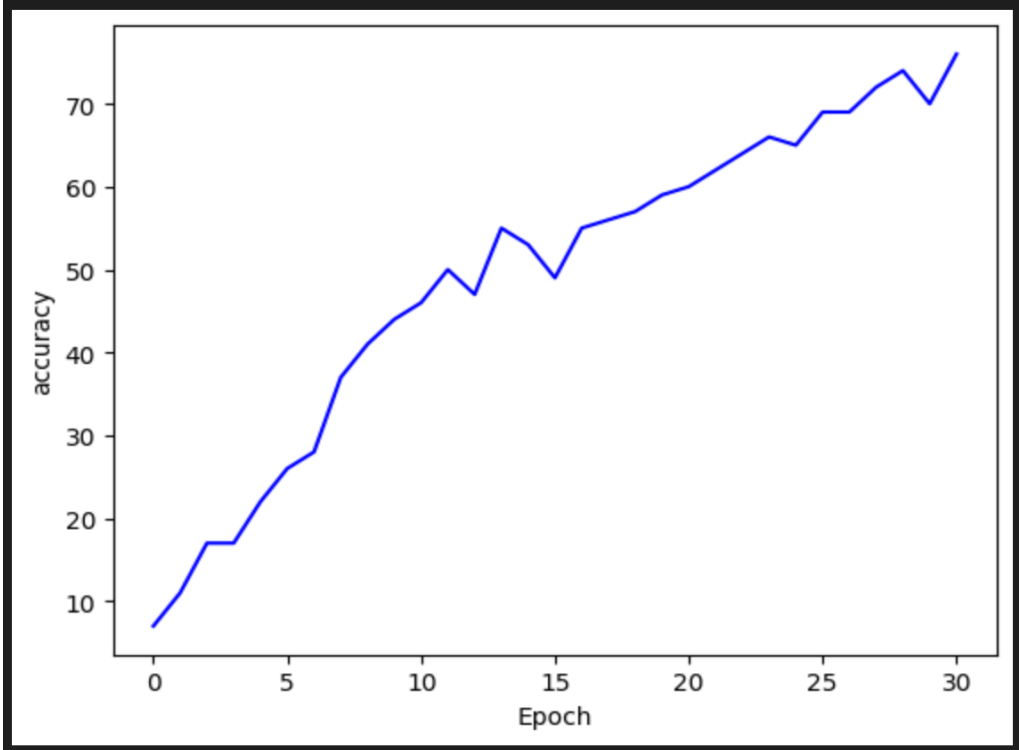
```
Accuracy of NN without training: 10.0%
```

برای costها و دقت و زمان مدل داریم:



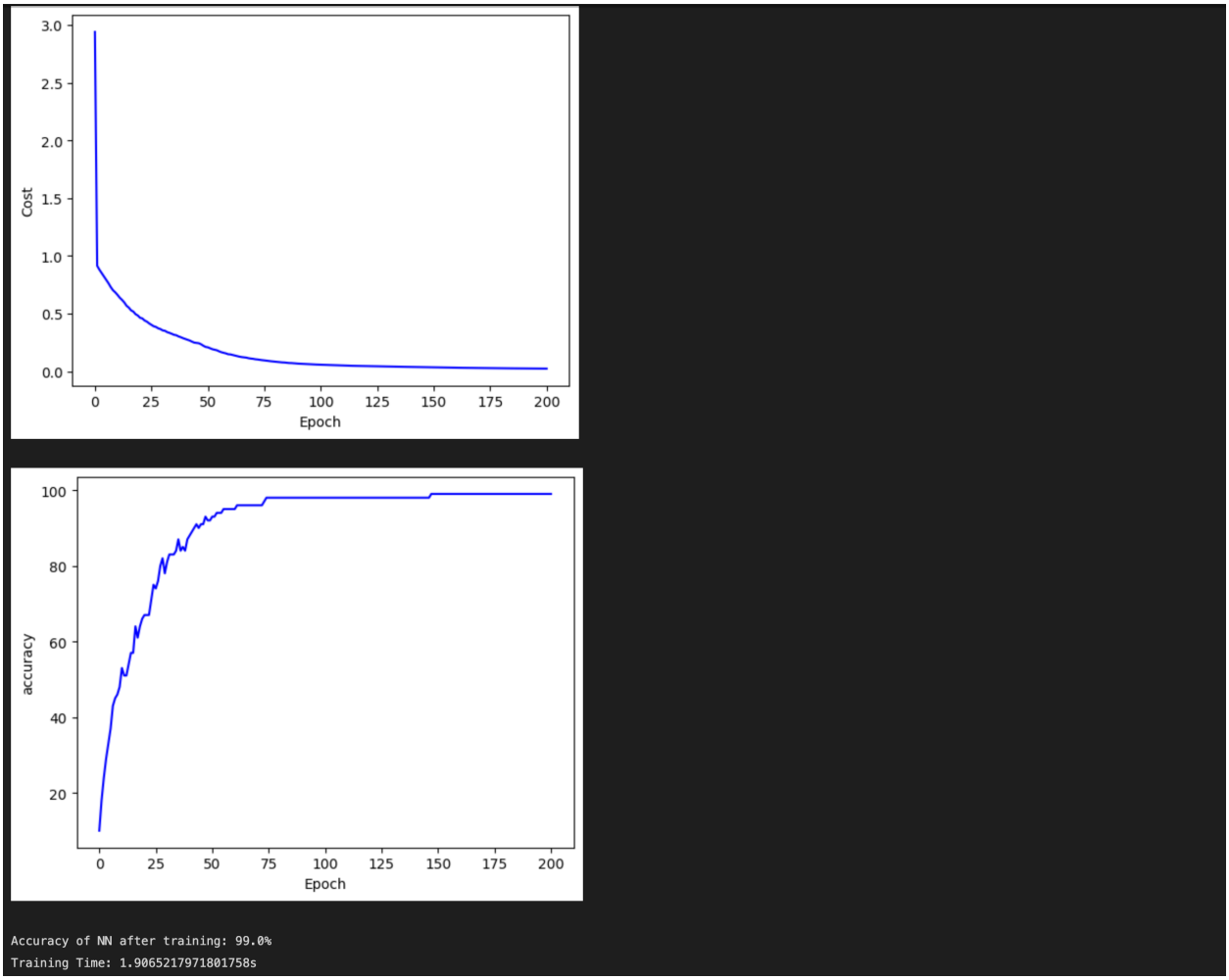
برای حالت ۳۰ تا epoch داریم:



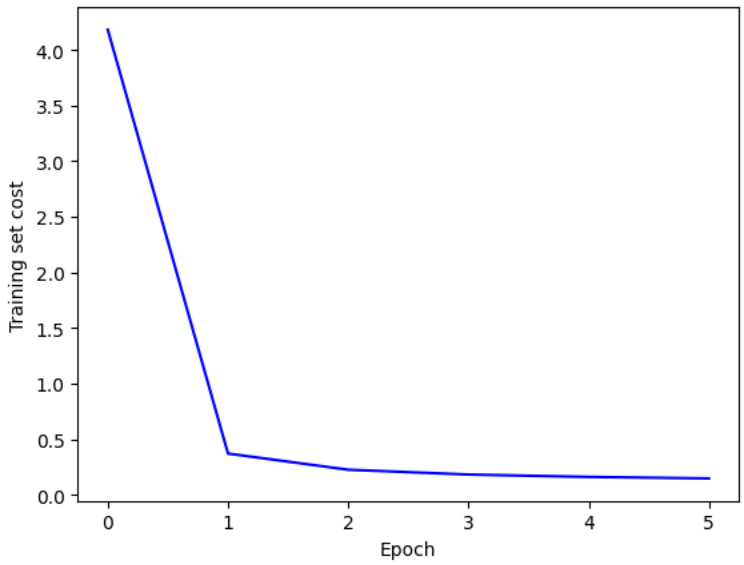


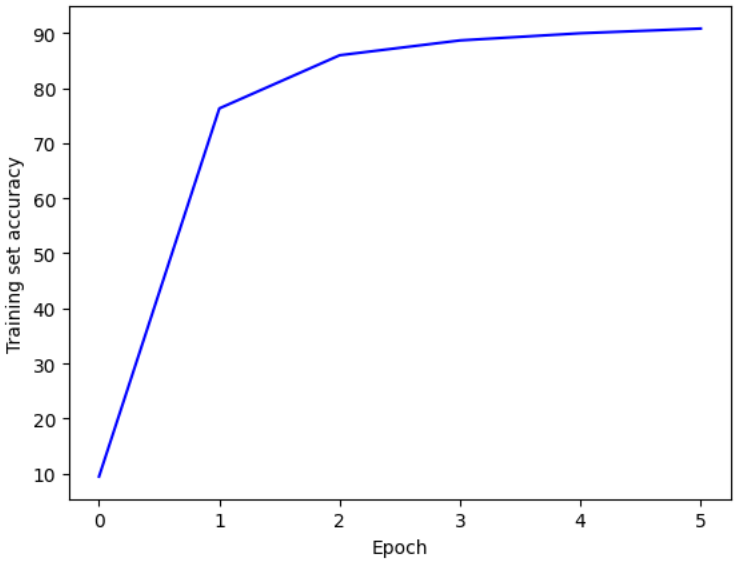

```
Accuracy of NN after training: 76.0%
Training Time: 23.916679859161377s
```

برای حالت برداری شده با epoch ۲۰۰ داریم:



برای داده‌های تست داریم:






```
Accuracy of NN after training (Training set): 90.79833333333333%
Training Time: 15.679389238357544s

Accuracy of NN after training (Test set): 90.36999999999999%


Now we can repeat this process to get the average accuracy of network.
```

سوالات تحقیقی:

۱)Cross validation تکنیکی برای تخمین مهارت مدل‌های یادگیری ماشین استفاده می‌شود. از cross validation برای تشخیص overfitting استفاده میشود. مجموعه اعتبار سنجی: مجموعه ای از مثال هایی که برای تنظیم پارامترها استفاده میشود. مثلا انتخاب تعداد hidden layer. – مجموعه تست: مجموعه ای از مثال ها که فقط برای ارزیابی عملکرد یک طبقه بندی کننده کاملاً مشخص استفاده می شود.

۲) در گرادیان کاهشی، تمام داده‌های آموزشی برای برداشتن یک مرحله در نظر گرفته می شود. میانگین گرادیان تمام مثال‌های آموزشی را می‌گیریم و سپس از آن گرادیان میانگین برای به‌روزرسانی پارامترهایمان استفاده می‌کنیم. بنابراین این فقط یک مرحله از gradient descent در یک دوره است.

گرادیان کاهشی برای منیفولدهای خطای محدب یا نسبتا صاف عالی است. در این مورد، ما تا حدودی مستقیماً به سمت یک راه حل بهینه حرکت می کنیم. نمودار هزینه در مقابل دوره ها نیز کاملاً صاف است زیرا ما در حال میانگین گیری از تمام گرادیان های داده های آموزشی برای یک مرحله واحد هستیم. هزینه در طول دوره ها کاهش می یابد.

در گرادیان کاهشی ، تمام مثال‌ها را برای هر مرحله از Gradient Descent در نظر می‌گرفتیم. اما چه می شود اگر مجموعه داده ما بسیار بزرگ باشد. مدل های یادگیری عمیق مشتاق داده هستند. هر چه داده ها بیشتر باشد، شانس یک مدل برای خوب بودن بیشتر است. فرض کنید مجموعه داده ما دارای ۵ میلیون نمونه باشد، سپس فقط برای برداشتن یک مرحله، مدل باید گرادیان تمام ۵ میلیون نمونه را محاسبه کند. این روش کارآمدی به نظر نمی رسد. برای مقابله با این مشکل ما Stochastic Gradient Descent داریم. در نزول گرادیان تصادفی (SGD)، ما در هر زمان فقط یک مثال را برای برداشتن یک گام در نظر می گیریم. از آنجایی که ما در هر زمان فقط یک مثال را در نظر می گیریم، هزینه بر روی نمونه های آموزشی همچنین از آنجایی که هزینه آن بسیار متغیر است، هرگز به حداقل نمی رسد، اما به نوسان در اطراف آن ادامه می دهد.

SGD را می توان برای مجموعه داده های بزرگتر استفاده کرد. وقتی مجموعه داده بزرگ باشد سریعتر همگرا می شود زیرا باعث به روز رسانی بیشتر پارامترها می شود. ر نوسان است و لزوما کاهش نمی یابد. اما در دراز مدت با نوسانات شاهد کاهش هزینه خواهید بود. ما Batch Gradient Descent را دیده ایم. ما همچنین شاهد نزول گرادیان تصادفی هستیم Batch Gradient Descent . را می توان برای منحنی های صاف تر استفاده کرد. زمانی که مجموعه داده بزرگ باشد می توان از SGD استفاده کرد Batch . Gradient Descentمستقیماً به حداقل همگرا می شود SGD .برای مجموعه داده های بزرگتر سریعتر همگرا می شود. اما، از آنجایی که در SGD ما در هر زمان فقط از یک مثال استفاده می‌کنیم، نمی‌توانیم پیاده‌سازی برداری شده را روی آن پیاده‌سازی کنیم. این می تواند محاسبات را کند کند. برای مقابله با این مشکل، ترکیبی از Batch Gradient Descent و SGD استفاده می شود.
نه ما از همه مجموعه داده به طور همزمان استفاده می کنیم و نه از مثال واحد در یک زمان استفاده می کنیم. ما از مجموعه ای از تعداد ثابت نمونه های آموزشی استفاده می کنیم که کمتر از مجموعه داده واقعی است و آن را یک دسته کوچک می نامیم. انجام این کار به ما کمک می کند تا به مزایای هر دو نوع قبلی که دیدیم دست پیدا کنیم. بنابراین، پس از ایجاد دسته های کوچک با اندازه ثابت، مراحل زیر را در یک دوره انجام می دهیم:

یک مینی دسته انتخاب کنید
آن را به شبکه عصبی تغذیه کنید
میانگین گرادیان دسته کوچک را محاسبه کنید
از گرادیان میانگینی که در مرحله ۳ محاسبه کردیم برای به روز رسانی وزن ها استفاده کنید
مراحل ۱-۴ را برای مینی بچ هایی که ایجاد کردیم تکرار کنید

دقیقاً مانند SGD، میانگین هزینه در طول دوره‌ها در نزول گرادیان دسته‌ای کوچک نوسان دارد، زیرا ما تعداد کمی از نمونه‌ها را در یک زمان میانگین می‌گیریم.
منبع: https://towardsdatascience.com/batch-mini-batch-stochastic-gradient-descent-7a62ecba642a
۳) نرمال سازی دسته ای (همچنین به عنوان هنجار دسته ای شناخته می شود) روشی است که برای آموزش شبکه های عصبی مصنوعی سریعتر و پایدارتر از طریق عادی سازی ورودی لایه ها با مرکزیت مجدد و مقیاس گذاری مجدد استفاده می شود. این توسط سرگئی آیوف و کریستین سگدی در سال ۲۰۱۵ پیشنهاد شد.

در حالی که اثر عادی سازی دسته ای مشهود است، دلایل اثربخشی آن همچنان مورد بحث است. اعتقاد بر این بود که می‌تواند مشکل تغییر متغیر داخلی را کاهش دهد، که در آن مقدار اولیه پارامتر و تغییرات در توزیع ورودی‌های هر لایه بر نرخ یادگیری شبکه تأثیر می‌گذارد. اخیراً، برخی از محققان استدلال کرده اند که نرمال سازی دسته ای تغییر متغیر کمکی داخلی را کاهش نمی دهد، بلکه تابع هدف را هموار می کند، که به نوبه خود عملکرد را بهبود می بخشد. با این حال، در زمان اولیه، نرمال سازی دسته ای در واقع باعث انفجار شدید گرادیان در شبکه های عمیق می شود، که تنها با اتصالات پرش در شبکه های باقیمانده کاهش می یابد. برخی دیگر معتقدند که نرمال سازی دسته ای به جداسازی جهت طول می رسد و در نتیجه شبکه های عصبی را تسریع می بخشد. اخیراً یک تکنیک برش گرادیان عادی و تنظیم فراپارامتر هوشمند در شبکه‌های بدون نرمالیزور معرفی شده است که به نام «NF-Nets»نامیده می‌شود که نیاز به نرمال‌سازی دسته‌ای را کاهش می‌دهد.

منبع: https://en.wikipedia.org/wiki/Batch_normalization

۴) وظیفه pooling در شبکه عصبی پیچشی این است که فیچر مپ‌ها را کوچک‌تر میکند. یک حالتی داریم که در آن بیشترین مقدار موجود برای ما بازگردانده میشود که به max pooling معروف است به زبان دیگر Max Pooling یک عملیات ادغام است که حداکثر عنصر را از ناحیه نقشه ویژگی تحت پوشش فیلتر انتخاب می کند و در حالت دیگر مقدار میانگین آن بازگردانده میشود که به average pooling معروف است. به زبان دیگر میانگین ادغام میانگین عناصر موجود در منطقه نقشه ویژگی تحت پوشش فیلتر را محاسبه می کند. حالت سوم حالت global pooling است. ادغام جهانی هر کانال در نقشه ویژگی را به یک مقدار کاهش می دهد. بنابراین، یک نقشه ویژگی $n_h*n_w*n_c$ به نقشه ویژگی $1*1*n_c$ کاهش می یابد. این معادل استفاده از فیلتری با ابعاد n_h*n_w است، یعنی ابعاد نقشه ویژگی.

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