نوید ابراهیمی ۹۹۵۲۱۰۰۱

سوال ۱)

الف)

استفاده از نرخ یادگیری بسیار بالا در مدل های یادگیری ماشینی می تواند چندین مشکل ایجاد کند:

۱. Overshooting the Minima: نرخ یادگیری اندازه گام را در طول نزول گرادیان تعیین می کند. اگر نرخ یادگیری خیلی بالا باشد، مدل ممکن است از حداقل ها عبور کند و در سمت دیگر قرار گیرد. این می تواند باعث شود که مدل در حول مینیمم نوسان کند بدون اینکه واقعاً به آن برسد.

۲. واگرایی: در موارد شدید، نرخ یادگیری بسیار بالا می تواند باعث شود که پارامترهای مدل به گونه ای به روز شوند که مدل کاملاً واگرا شود. این بدان معنی است که به جای به حداقل رساندن تابع ضرر، ضرر با هر تکرار افزایش می یابد.

۳. همگرایی ضعیف: حتی اگر مدل واگرا نباشد، نرخ یادگیری بالا همچنان ممکن است مانع از همگرایی موثر آن به حداقل شود. مدل ممکن است در اطراف حداقل ها به چرخش خود ادامه دهد و منجر به ضرر نهایی بالاتر از حد لازم شود.

۴. پرش به یک بهینه محلی بدتر: این مدل ممکن است از یک حداقل محلی خوب به یک بدتر به دلیل گام های بزرگ ناشی از نرخ یادگیری بالا بپرد. هنگامی که مدل نزدیک به این بهینه محلی بدتر است، گرادیان ممکن است نزدیک به صفر باشد، که مانع از توانایی مدل برای حرکت به بیرون می شود.

۵. وزن‌های نهایی کمتر از حد مطلوب: نرخ‌های یادگیری بزرگ‌تر اغلب منجر به مجموعه‌ای از وزن‌های نهایی نابهینه می‌شود. این به این دلیل است که مدل ممکن است به دلیل گام‌های بزرگی که در فضای پارامتر ۱ برمی‌دارد، زمان کافی برای تنظیم دقیق وزن‌ها نداشته باشد.

برای تشخیص این مشکلات، می توانید Loss را در طول تمرین کنترل کنید. اگر ضرر به شدت در نوسان باشد یا پس از کاهش اولیه افزایش یابد، ممکن است به دلیل نرخ بالای یادگیری باشد. به‌علاوه، اگر عملکرد مدل در مجموعه اعتبارسنجی علی‌رغم از دست دادن آموزش کم، ضعیف باشد، می‌تواند نشانه‌ای از بالا بودن نرخ یادگیری باشد که باعث می‌شود مدل بیش از حد برازش کند.

ب)

۱. همگرایی آهسته: این مدل ممکن است زمان زیادی طول بکشد تا همگرا شود یا اصلاً همگرا نشود. این به این دلیل است که مدل در هر تکرار به‌روزرسانی‌های بسیار کوچکی را برای وزن‌ها انجام می‌دهد.

۲. گیر کردن در حداقل های محلی: مدل ممکن است در حداقل های محلی ضعیف یا مناطق مسطح در چشم انداز تلفات گیر کند. این به این دلیل است که مراحل آنقدر کوچک هستند که به مدل اجازه نمی دهد از این مناطق فرار کند.

۳. ناپدید شدن گرادیان ها: نرخ یادگیری بسیار کم می تواند منجر به ناپدید شدن گرادیان ها، به خصوص در شبکه های عمیق شود. این بدان معناست که گرادیان‌های از دست دادن با توجه به پارامترهای مدل بسیار کوچک و تقریباً صفر می‌شوند و باعث می‌شود مدل قادر به یادگیری نباشد.

برای تشخیص این مشکلات، می توانید از دست دادن را در طول تمرین کنترل کنید. اگر از دست دادن بسیار آهسته کاهش یابد یا اصلاً کاهش یابد، ممکن است به دلیل نرخ بسیار پایین یادگیری باشد. بعلاوه، اگر عملکرد مدل در مجموعه اعتبارسنجی علیرغم زمان طولانی آموزش ضعیف باشد، می تواند نشانه ای از پایین بودن نرخ یادگیری باشد.

پ)

نقطه زینی در ریاضیات نقطه‌ای روی نمودار یک تابع است که در آن شیب‌ها (مشتق‌ها) در جهت‌های متعامد همگی صفر هستند (نقطه بحرانی)، اما منتهی موضعی تابع نیست. این نقطه ای است که تابع نه به حداکثر مقدار محلی و نه به یک مقدار حداقل محلی می رسد.

مزایا و معایب:

ADAM:

- مزایا: ADAM تمایل دارد سریعتر همگرا شود. این نقص واریانس بالا SGD را اصلاح می کند. مزایای دو افزونه دیگر SGD، یعنی RMSProp و Momentum را ترکیب می کند.

- معایب: ADAM به مقادیر خاصی از نرخ یادگیری بسیار حساس است و اگر نرخ یادگیری بیش از حد بالا باشد ممکن است نتواند همگرا شود. اغلب به یک "حداقل تیز" همگرا می شود که ممکن است در داده های تست عملکرد خوبی نداشته باشد.

SGD:

- مزایا: SGD اغلب به راه حل های بهینه تر همگرا می شود. این دارای پایه های نظری قوی است و هنوز هم در آموزش شبکه های عصبی لبه استفاده می شود. به یک "حداقل مسطح" همگرا می شود و در داده های تست عملکرد خوبی دارد.

- معایب: SGD از نظر محاسباتی سنگین است. به سرعت ADAM همگرا نمی شود.

از نظر نقاط زین، هر دو Adam و SGD راه های مخصوص به خود را برای ناوبری دارند. با این حال، انتخاب بین Adam و SGD به نیازهای خاص مدل و داده ها بستگی دارد.

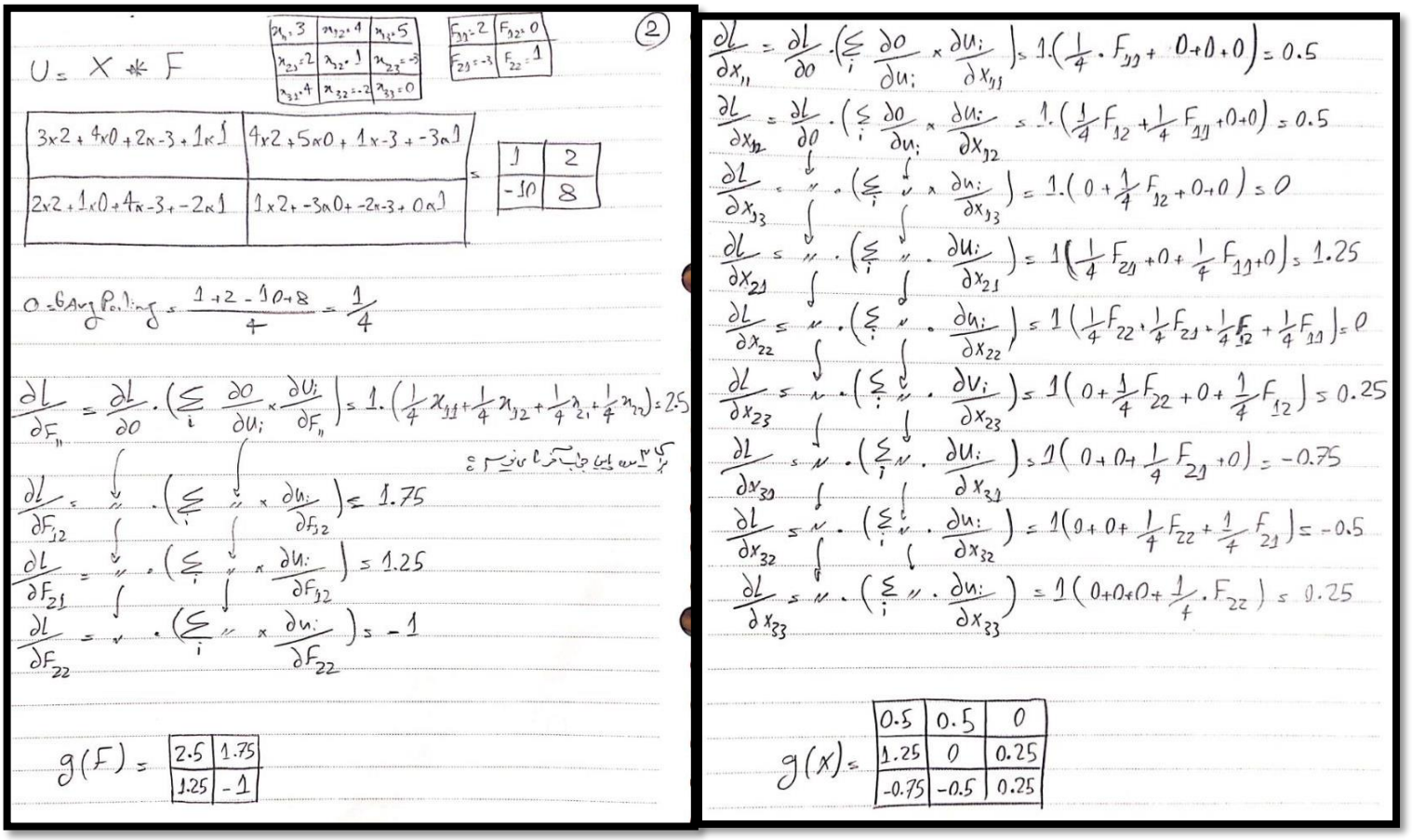
ت)

عکس سمت راست بیانگر mini-batch و عکس سمت چپ بیانگر batch است.

در نمودار Batch Gradient Descent، معمولاً می بینید که تابع هزینه به طور پیوسته در طول دوره ها کاهش می یابد. این به این دلیل است که کل مجموعه داده برای محاسبه گرادیان و به روز رسانی پارامترهای مدل در هر تکرار استفاده می شود.

از سوی دیگر، در نمودار نزولی گرادیان دسته ای کوچک، تابع هزینه به همین راحتی کاهش نمی یابد. ممکن است به دلیل واریانسی که با استفاده از زیرمجموعه‌ها (مجموعه‌های کوچک) مجموعه داده برای هر تکرار ایجاد می‌شود، کمی بالا و پایین نوسان داشته باشد. با این حال، به طور کلی از روند کاهشی تابع هزینه در طول دوره ها پیروی می کند.

سوال ۲)



سوال ۳)

۱. لایه ورودی: این لایه هیچ پارامتری ندارد، فقط شکل ورودی را مشخص می کند.

۲. Conv1D Layer 1: تعداد پارامترهای یک لایه Convolutional توسط (kernel_size * filters * input_channels) + filters داده می شود. در اینجا filters=16، kernel_size=3 و input_channels=7 (از شکل ورودی). بنابراین، کل پارامترها (۱۶*۳*۷) + ۱۶ = ۳۵۲ هستند.

۳. MaxPool1D Layer 1: لایه های Max Pooling هیچ پارامتری ندارند زیرا فقط یک عملیات downsampling را انجام می دهند و چیزی یاد نمی گیرند.

۴. Conv1D Layer 2: در اینجا filters=32، kernel_size=5 و input_channels=16 (از خروجی لایه Conv1D قبلی). بنابراین، کل پارامترها (۳۲*۵*۱۶) + ۳۲ = ۲۵۹۲ هستند.

۵. MaxPool1D Layer 2: باز هم هیچ پارامتری در این لایه وجود ندارد.

۶. Conv1D Layer 3: در اینجا filters=64، kernel_size=5 و input_channels=32 (از خروجی لایه قبلی Conv1D) .بنابراین، کل پارامترها (۶۴*۵*۳۲) + ۶۴ = ۱۰۳۰۴ هستند.

۷. MaxPool1D Layer 3: باز هم هیچ پارامتری در این لایه وجود ندارد.

۸. Flatten Layer: این لایه نیز هیچ پارامتری ندارد. فقط ورودی را صاف می کند.

۹. Dense Layer 1: تعداد پارامترهای یک لایه متراکم با (input_size * units) + units داده می شود. در اینجا، units=128 و input_size به شکل خروجی لایه Flatten بستگی دارد. فرض کنید شکل خروجی بعد از Flatten N باشد. بنابراین، کل پارامترها 128 + (128*N) هستند.

۱۰. Dense Layer 2 (لایه خروجی): در اینجا واحدها=۵ و input_size=128 (از خروجی لایه متراکم قبلی). بنابراین، کل پارامترها (۵*۱۲۸) + ۵ = ۶۴۵ هستند.

ب)

- Conv2D (پیچیدگی دو بعدی): این برای پیچیدگی فضایی روی تصاویر استفاده می شود. هسته در Conv2D در دو جهت (ارتفاع و عرض) حرکت می کند. بنابراین، به طور گسترده ای برای کارهای مربوط به تصویر استفاده می شود که در آن ورودی یک تصویر دو بعدی است. با این حال، Conv2D با هر فریم به طور مستقل برخورد می‌کند و ممکن است زمینه زمانی را در وظایف مبتنی بر ویدیو رعایت نکند.

- Conv3D (Three-Dimensional Convolution): این برای پیچش حجمی استفاده می شود. هسته در Conv3D در سه جهت (ارتفاع، عرض و عمق) حرکت می کند. زمانی استفاده می شود که داده های ورودی سه بعدی باشد. به عنوان مثال، در پردازش ویدئو یا تصویربرداری پزشکی، که در آن ورودی دنباله ای از تصاویر دو بعدی است که حجم سه بعدی را تشکیل می دهند. Conv3D می‌تواند ویژگی‌های هر فریم را در تعامل قرار دهد و ممکن است زمینه‌های زمانی را بیاموزد، بنابراین نشان داده می‌شود که در وظایف مبتنی بر ویدیو عملکرد بهتری دارد.

کاربردهای Conv3D:

- پردازش ویدیو: Conv3D در کارهایی مانند تشخیص عملکرد در فیلم ها استفاده می شود، جایی که رابطه زمانی بین فریم ها مهم است.

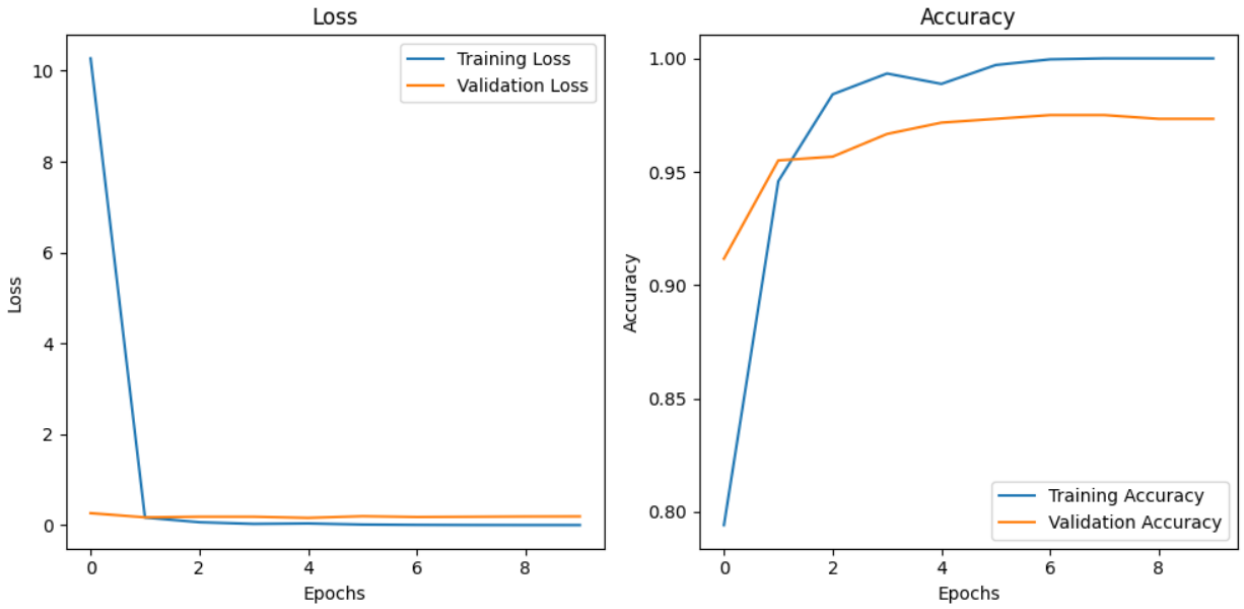
- تصویربرداری پزشکی: در تصویربرداری پزشکی، تصاویر سه بعدی توسط برش ها گرفته شده و سپس بازسازی می شوند. همه برش‌هایی که با هم اضافه می‌شوند باید به‌طور کلی تجزیه و تحلیل شوند، بنابراین استفاده از کانولوشن سه بعدی منطقی است زیرا اگر تصاویر تکی بگیریم و کانولوشن دو بعدی را اعمال کنیم، ممکن است روابط از بین بروند.

- تقسیم بندی حجم: Conv3D را می توان برای بهبود عملکرد چندین معماری سه بعدی CNN در معیارهای مربوط به تقسیم بندی ویدئویی چندگانه استفاده کرد.

سوال ۴)

توضیحات مربوط به توابع را داخل کد قرار داده‌ام.

```
Epoch 1/10
75/75 [==============================] - 49s 66ms/step - loss: 10.2729 - accuracy: 0.7942 - val_loss: 0.2634 - val_accuracy: 0.9117
Epoch 2/10
75/75 [==============================] - 6s 73ms/step - loss: 0.1671 - accuracy: 0.9458 - val_loss: 0.1710 - val_accuracy: 0.9550
Epoch 3/10
75/75 [==============================] - 5s 59ms/step - loss: 0.0635 - accuracy: 0.9842 - val_loss: 0.1853 - val_accuracy: 0.9567
Epoch 4/10
75/75 [==============================] - 7s 94ms/step - loss: 0.0273 - accuracy: 0.9933 - val_loss: 0.1847 - val_accuracy: 0.9667
Epoch 5/10
75/75 [==============================] - 5s 59ms/step - loss: 0.0370 - accuracy: 0.9887 - val_loss: 0.1597 - val_accuracy: 0.9717
Epoch 6/10
75/75 [==============================] - 5s 63ms/step - loss: 0.0122 - accuracy: 0.9971 - val_loss: 0.1962 - val_accuracy: 0.9733
Epoch 7/10
75/75 [==============================] - 6s 74ms/step - loss: 0.0047 - accuracy: 0.9996 - val_loss: 0.1788 - val_accuracy: 0.9750
Epoch 8/10
75/75 [==============================] - 6s 77ms/step - loss: 0.0017 - accuracy: 1.0000 - val_loss: 0.1839 - val_accuracy: 0.9750
Epoch 9/10
75/75 [==============================] - 5s 59ms/step - loss: 6.1118e-04 - accuracy: 1.0000 - val_loss: 0.1898 - val_accuracy: 0.9733
Epoch 10/10
75/75 [==============================] - 6s 83ms/step - loss: 3.3332e-04 - accuracy: 1.0000 - val_loss: 0.1917 - val_accuracy: 0.9733
```



سوال ۵)

بیایید یک مثال عملی از طبقه بندی متن با استفاده از شبکه های عصبی کانولوشنال (CNN) را در نظر بگیریم. طبقه بندی متن یک کار رایج در پردازش زبان طبیعی (NLP) است که در آن متن را به کلاس های از پیش تعریف شده دسته بندی می کنیم. یک مثال می تواند تجزیه و تحلیل احساسات باشد، که در آن هدف طبقه بندی متن به عنوان مثبت، منفی یا خنثی است.

در اینجا نحوه استفاده از CNN برای این کار آمده است:

- جمع آوری داده ها: ما با جمع آوری مجموعه داده بزرگی از متن و برچسب های مربوط به آنها شروع می کنیم. این می تواند نقدهای فیلم با برچسب مثبت یا منفی باشد.

- پیش پردازش: داده های متنی از پیش پردازش می شوند که شامل رمزگذاری (تجزیه متن به کلمات جداگانه) و رمزگذاری نشانه ها به عنوان بردارهای عددی با استفاده از تکنیک هایی مانند رمزگذاری تک داغ یا جاسازی کلمه است.

- آموزش مدل: سپس یک CNN را بر روی این مجموعه داده آموزش می دهیم. لایه‌های کانولوشن در شبکه می‌توانند وابستگی‌های محلی را در متن ثبت کنند (مانند n-gram) و لایه‌های ادغام می‌توانند ابعاد را کاهش دهند و به قوی‌تر کردن نمایش کمک کنند.

- ارزیابی و استقرار: مشابه مثال طبقه‌بندی تصویر، ما مدل را در یک مجموعه آزمایشی جداگانه ارزیابی می‌کنیم و زمانی که از عملکرد آن راضی بودیم، آن را برای استفاده در دنیای واقعی مستقر می‌کنیم.

مزایا:

اتصال محلی: در داده های متنی، کلماتی که به هم نزدیک هستند اغلب همبستگی بالاتری دارند. لایه های کانولوشنال با اتصال هر نورون تنها به ناحیه کوچکی از حجم ورودی (میدان گیرنده) از این مزیت استفاده می کنند. این به شبکه اجازه می‌دهد تا روی ویژگی‌های محلی در داده‌های متنی، مانند n-gram (توالی‌های پیوسته از n مورد از یک نمونه معین از متن یا گفتار) تمرکز کند.

وزن های مشترک: همه نورون ها در یک لایه کانولوشن دارای وزن های یکسان هستند. این بدان معنی است که همه آنها فقط در مکان های مختلف ورودی به دنبال یک ویژگی هستند. این منجر به تغییر ناپذیری ترجمه می شود، به عنوان مثال، توانایی تشخیص یک ویژگی بدون توجه به موقعیت آن در ورودی. در زمینه طبقه بندی متن، این بدان معنی است که شبکه می تواند ویژگی های مهم (مانند n-gram) را بدون توجه به موقعیت آنها در متن تشخیص دهد.

نقشه های چندگانه: یک لایه کانولوشن از چندین نقشه ویژگی تشکیل شده است که هر کدام مجموعه ای از وزن های خاص خود را دارند. این اجازه می دهد تا لایه یاد بگیرد چندین ویژگی مختلف را در داده های ورودی تشخیص دهد. در زمینه طبقه بندی متن، این بدان معنی است که شبکه می تواند یاد بگیرد که انواع ویژگی های مهم را در متن تشخیص دهد، که می تواند برای طبقه بندی دقیق متن بسیار مهم باشد.

ادغام/نمونه‌گیری فرعی: لایه‌های کانولوشن اغلب با لایه‌های ادغام دنبال می‌شوند که ابعاد فضایی داده‌ها را کاهش می‌دهد. این باعث می شود شبکه در برابر ترجمه های کوچک و اعوجاج در داده های ورودی قوی تر شود و

پیچیدگی محاسباتی شبکه کاهش یابد. در زمینه طبقه بندی متن، این می تواند به تقویت شبکه در برابر تغییرات متن، مانند تغییر در ترتیب کلمات یا وجود مترادف ها کمک کند.

معایب:

عدم وجود اطلاعات موقعیتی: لایه های کانولوشن به دلیل خاصیت تغییر ناپذیری ترجمه، می توانند اطلاعات مربوط به موقعیت و ترتیب ویژگی ها را در ورودی از دست بدهند. این می تواند در کارهایی که موقعیت ویژگی ها مهم است یک نقطه ضعف باشد. به عنوان مثال، در وظایف پردازش زبان طبیعی، ترتیب کلمات حاوی اطلاعات قابل توجهی است که ممکن است در لایه های کانولوشنی از بین برود.

فرض سلسله مراتب فضایی: لایه های کانولوشنال فرض می کنند که داده ها دارای سلسله مراتب فضایی هستند، به عنوان مثال، الگوهای محلی کوچک در الگوهای بزرگتر و پیچیده تر ساخته می شوند. این برای تصاویر یا داده‌های سری زمانی به خوبی کار می‌کند، اما برای انواع دیگر داده‌ها (مانند داده‌های جدولی)، این فرض ممکن است برقرار نباشد و لایه‌های کانولوشن بهترین انتخاب نباشند.

پیچیدگی محاسباتی: در حالی که لایه های کانولوشن از طریق اشتراک وزن تعداد پارامترها را کاهش می دهند، محاسبات همچنان می تواند فشرده باشد، به خصوص برای داده های ورودی بزرگ یا مدل های پیچیده با لایه های زیاد. این می تواند آموزش و استنتاج را به ویژه در دستگاه هایی با منابع محاسباتی محدود کند کند.

نیاز به مجموعه داده‌های بزرگ: شبکه‌های کانولوشن معمولاً برای آموزش مؤثر به مقادیر زیادی داده برچسب‌گذاری شده نیاز دارند. جمع آوری و برچسب گذاری چنین مجموعه داده های بزرگی می تواند چالش برانگیز و وقت گیر باشد. علاوه بر این، اگر داده‌های موجود به اندازه کافی متنوع یا نماینده فضای مسئله نباشد، مدل ممکن است به خوبی به داده‌های دیده نشده تعمیم نکند.

دشواری در تفسیرپذیری: لایه‌های کانولوشن، مانند بسیاری از روش‌های یادگیری عمیق، از فقدان تفسیرپذیری رنج می‌برند. درک اینکه چرا یک شبکه کانولوشنیک پیش بینی خاصی انجام داده است، می تواند مشکل باشد، که می تواند در برنامه هایی که قابلیت تفسیر مهم است، مانند مراقبت های بهداشتی یا مالی، مشکل ساز باشد.

سوال ۶)

الف)

پیچیدگی 1x1 که به عنوان پیچیدگی نقطه‌ای نیز شناخته می‌شود، تکنیکی است که در شبکه‌های عصبی کانولوشن (CNN) برای اهداف مختلف استفاده می‌شود:

۱. کاهش ابعاد: از فیلترهای کانولوشنال 1x1 می توان برای کاهش ابعاد نقشه های ویژگی ورودی و در عین حال حفظ ویژگی های مهم استفاده کرد. این با اعمال یک فیلتر 1x1 که کانال های خروجی کمتری نسبت به کانال های ورودی دارد به دست می آید. این عملیات عمق نقشه های ویژگی را کاهش می دهد و در نتیجه پیچیدگی محاسباتی شبکه را کاهش می دهد.

۲. افزایش غیرخطی بودن: لایه کانولوشن 1x1، زمانی که با یک تابع فعال سازی غیرخطی مانند ReLU (واحد خطی اصلاح شده) استفاده می شود، می تواند غیرخطی بودن بیشتری را به مدل معرفی کند. این می تواند به شبکه کمک کند تا الگوهای پیچیده تری را در داده ها بیاموزد.

۳. تجمیع ویژگی از نظر کانال: در یک عملیات کانولوشن معمولی، فیلترها در تمام کانال های ورودی اعمال می شوند. با این حال، با کانولوشن های 1x1، فیلترها را می توان برای هر کانال ورودی به طور جداگانه اعمال کرد و به شبکه اجازه می دهد تا ویژگی های کانال را یاد بگیرد.

به طور خلاصه، پیچیدگی‌های 1x1 با کاهش ابعاد، افزایش غیرخطی بودن، و امکان یادگیری ویژگی از نظر کانال، نقش مهمی در افزایش کارایی و عملکرد CNN ایفا می‌کنند. آنها به ویژه در شبکه های عمیق که در آن منابع محاسباتی یک نگرانی هستند مفید هستند.

هنگامی که یک پیچیدگی 1x1 به ورودی با چندین کانال (یا نقشه های ویژگی) اعمال می شود، یک عملیات dot product بین وزن فیلتر و مقادیر موجود در هر کانال در مکان مکانی مربوطه انجام می دهد. این عملیات به طور موثر اطلاعات همه کانال های ورودی را در یک مقدار خروجی واحد ترکیب می کند.

اگر لایه کانولوشنال 1x1 فیلترهای کمتری نسبت به تعداد کانال های ورودی داشته باشد، نقشه های ویژگی خروجی کمتری تولید می کند و در نتیجه ابعاد را کاهش می دهد. با وجود این کاهش، ویژگی‌های مهم همچنان قابل حفظ هستند، زیرا هر فیلتر 1x1 یاد می‌گیرد که مفیدترین اطلاعات را از کانال‌های ورودی در طول فرآیند آموزش حفظ کند.

به عنوان مثال، اگر یک ورودی با ۲۵۶ کانال داشته باشید و یک کانولوشن 1x1 با ۶۴ فیلتر اعمال کنید، خروجی دارای ۶۴ کانال خواهد بود. هر یک از این ۶۴ کانال خروجی ترکیبی از ویژگی های ۲۵۶ کانال ورودی است و شبکه در طول آموزش یاد می گیرد که کدام ویژگی ها را ترکیب و حفظ کند.

به این ترتیب، کانولوشن های 1x1 می توانند به طور موثری تعداد نقشه های ویژگی را کاهش دهند و در عین حال مهم ترین ویژگی ها را حفظ کنند. این به ویژه در شبکه های عمیق که بازده محاسباتی یک نگرانی است مفید است.

ب)

پس از اعمال یک فیلتر 1x1، نقشه ویژگی به دست آمده یک نمایش فشرده از نقشه های ویژگی ورودی را ارائه می دهد. هر مقدار در نقشه ویژگی خروجی، مجموع وزنی مقادیر در مکان مکانی متناظر در تمام نقشه های ویژگی ورودی است. وزنه ها در طول فرآیند تمرین یاد می گیرند.

نکته کلیدی این است که هر فیلتر 1x1 یاد می گیرد که نوع خاصی از ویژگی را از ورودی استخراج کند. به عنوان مثال، یک فیلتر ممکن است یاد بگیرد لبه ها را تشخیص دهد، دیگری ممکن است بافت ها را بیاموزد و غیره. نقشه ویژگی خروجی هر فیلتر 1x1 نشان دهنده حضور ویژگی آموخته شده در هر مکان فضایی در ورودی است.

بنابراین، در اصل، نقشه ویژگی پس از اعمال فیلتر 1x1 مجموعه‌ای از ویژگی‌های سطح بالا را ارائه می‌کند که از نقشه‌های ویژگی ورودی آموخته شده‌اند. این ویژگی های سطح بالا می توانند توسط لایه های بعدی در شبکه برای پیش بینی یا استخراج ویژگی های پیچیده تر استفاده شوند. اینگونه است که پیچش های 1x1 به حفظ ویژگی های مهم و در عین حال کاهش ابعاد کمک می کنند.

سایر اطلاعات:

۱. اطلاعات کانال: نقشه ویژگی خروجی بسته به تعداد فیلترهای 1x1 استفاده شده می تواند تعداد کانال های متفاوتی نسبت به ورودی داشته باشد. این اغلب برای تغییر عمق نقشه ویژگی برای کنترل پیچیدگی مدل استفاده می شود.

۲. تبدیل های غیر خطی: نقشه ویژگی خروجی، تبدیل های غیرخطی نقشه ویژگی ورودی را نشان می دهد. این به این دلیل است که هر پیچیدگی 1x1 معمولاً توسط یک تابع فعال سازی غیر خطی مانند ReLU دنبال می شود.

۳. اطلاعات مکانی: نقشه ویژگی خروجی، اطلاعات مکانی نقشه ویژگی ورودی را حفظ می کند. این به این دلیل است که یک پیچیدگی 1x1 روی هر پیکسل به صورت جداگانه عمل می کند، بدون اینکه اطلاعات پیکسل های همسایه را جمع آوری کند.

بنابراین، نقشه ویژگی خروجی یک کانولوشن 1x1 یک نسخه کاهش‌یافته و غیرخطی از نقشه ویژگی ورودی در حالی که اطلاعات مکانی آن را حفظ می‌کند، با ابعاد کاهش‌یافته و غیرخطی ارائه می‌کند. این به مدل اجازه می دهد تا ویژگی های پیچیده تری را بدون افزایش قابل توجه هزینه محاسباتی بیاموزد.

پ)

نقشه ویژگی به‌دست‌آمده از یک فیلتر کانولوشنال 1x1 با تصویر اصلی و نقشه‌های ویژگی به‌دست‌آمده از اندازه‌های فیلتر دیگر از چند طریق متفاوت است:

۱. ابعاد: تصویر اصلی دارای سه کانال رنگی (RGB) است، در حالی که عمق نقشه ویژگی خروجی (تعداد کانال) به تعداد فیلترهای 1x1 استفاده شده بستگی دارد. این می تواند بیشتر یا کمتر از سه باشد و به مدل اجازه می دهد پیچیدگی و هزینه محاسباتی را کنترل کند.

۲. غیر خطی بودن: برخلاف تصویر اصلی، نقشه ویژگی خروجی، تبدیلات غیرخطی ورودی را نشان می دهد. این به این دلیل است که هر پیچیدگی 1x1 معمولاً توسط یک تابع فعال سازی غیر خطی مانند ReLU دنبال می شود.

۳. تجمیع فضایی: فیلترهای بزرگتر از 1x1 (مانند 3x3 یا 5x5) اطلاعات را از همسایگی پیکسل‌ها جمع‌آوری می‌کنند که می‌توانند الگوهای محلی مانند لبه‌ها یا بافت‌ها را ثبت کنند. در مقابل، یک پیچیدگی 1x1 روی هر

پیکسل به صورت جداگانه عمل می کند و اطلاعات مکانی نقشه ویژگی ورودی را حفظ می کند اما این الگوهای محلی را نمی گیرد.

۴. یادگیری ویژگی: در حالی که تصویر اصلی فقط حاوی داده های پیکسل خام است، نقشه ویژگی خروجی نشان دهنده ویژگی های آموخته شده است. این ویژگی ها در طول فرآیند آموزش برای کمک به مدل در پیش بینی دقیق یاد می شوند.

بنابراین، نقشه ویژگی خروجی یک کانولوشن 1x1 یک نسخه کاهش‌یافته و غیرخطی از نقشه ویژگی ورودی در حالی که اطلاعات مکانی آن را حفظ می‌کند، با ابعاد کاهش‌یافته و غیرخطی ارائه می‌کند. این به مدل اجازه می دهد تا ویژگی های پیچیده تری را بدون افزایش قابل توجه هزینه محاسباتی بیاموزد.

ت)

فیلتر کانولوشنال 1x1 در چندین مدل موفق شبکه عصبی کانولوشن (CNN) استفاده شده است. در اینجا چند نمونه هستند:

GoogleNet: فیلتر 1x1 به صراحت برای کاهش ابعاد و برای افزایش ابعاد نقشه های ویژگی پس از ادغام در طراحی ماژول اولیه استفاده شد.

ResNet: ResNet همچنین از پیچش 1x1، عمدتاً برای کاهش ابعاد و بازسازی استفاده می کند.

SqueezeNet: SqueezeNet از یک بلوک "squeeze-and-excitation" استفاده می کند که از پیچیدگی های 1x1 برای فشرده کردن نقشه ویژگی ورودی (کاهش ابعاد آن) قبل از اعمال 3x3 convolutions استفاده می کند.

این مدل‌ها از پیچیدگی‌های 1x1 برای کنترل پیچیدگی مدل، تنظیم تعداد کانال‌ها و گرفتن ویژگی‌ها در مقیاس‌های مختلف استفاده می‌کنند.

ث)

بله، شرایطی وجود دارد که استفاده از فیلترهای 1x1 ممکن است سودمند نباشد. در اینجا چند نمونه هستند:

۱. گرفتن اطلاعات فضایی: پیچیدگی های 1x1 بر روی پیکسل های منفرد عمل می کنند و اطلاعات محله خود را جمع نمی کنند. بنابراین، آنها برای ثبت الگوهای فضایی محلی در نقشه ویژگی ورودی مناسب نیستند. فیلترهای بزرگتر (مانند 3x3 یا 5x5) یا لایه های جمع کننده معمولاً برای این منظور استفاده می شوند.

۲. استخراج ویژگی های سطح پایین: در لایه های اولیه CNN، مدل اغلب نیاز به یادگیری ویژگی های سطح پایین مانند لبه ها یا بافت ها دارد. این ویژگی‌ها معمولاً شامل روابط فضایی بین پیکسل‌ها می‌شوند، بنابراین فیلترهای بزرگتر معمولاً مؤثرتر هستند.

۳. Overfitting: اگر تعداد فیلترهای 1x1 بیش از حد زیاد باشد، می تواند منجر به مدلی با پارامترهای بیش از حد شود که می تواند باعث overfitting شود. این امر به ویژه اگر مقدار داده های آموزشی محدود باشد، نگران کننده است.

۴. عدم تناسب: برعکس، اگر تعداد فیلترهای 1x1 خیلی کم باشد، مدل ممکن است ظرفیت کافی برای یادگیری ویژگی‌های پیچیده را نداشته باشد و منجر به عدم تناسب شود.

بنابراین، در حالی که کانولوشن های 1x1 ابزار قدرتمندی در CNN هستند، همیشه بهترین انتخاب نیستند. معماری بهینه اغلب شامل ترکیبی از انواع مختلف لایه ها و اندازه فیلترها، بسته به وظیفه و داده خاص است.

ج)

افزایش از ۳ به ۱۶ نتیجه تعداد فیلترهای استفاده شده در لایه کانولوشن است. در یک شبکه عصبی کانولوشنال (CNN)، هر فیلتر یک کانال خروجی تولید می کند. بنابراین، اگر از ۱۶ فیلتر در یک لایه استفاده کنید، خروجی با ۱۶ کانال دریافت خواهید کرد.

در مورد کانولوشن 1x1، هر فیلتر ۳ کانال ورودی را می گیرد و یک تبدیل (که در طول آموزش آموخته می شود) را برای تولید یک کانال خروجی اعمال می کند. وقتی ۱۶ فیلتر از این قبیل اعمال می کنید، ۱۶ کانال خروجی دریافت می کنید.

این افزایش در کانال ها به مدل اجازه می دهد تا نمایش های پیچیده تری از داده های ورودی را بیاموزد. هر کانال در خروجی می تواند یاد بگیرد که ویژگی های متفاوتی را از ورودی بگیرد و در نتیجه قدرت بیان مدل را افزایش دهد. این به ویژه در مدل های یادگیری عمیق که در آن ثبت الگوهای پیچیده و سلسله مراتبی در داده ها برای کارهایی مانند تشخیص تصویر، پردازش زبان طبیعی و غیره مهم است مفید است.

```
seq_model = Sequential([
    Conv2D(8, kernel_size=(1, 1), padding='same', activation='relu', input_shape=(256, 256, 3)),
    Conv2D(16, kernel_size=(1, 1), activation='relu'),
    Flatten(),
    Dense(2, activation='sigmoid')
])
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 256, 256, 8)       32

 conv2d_3 (Conv2D)           (None, 256, 256, 16)      144

 flatten_1 (Flatten)         (None, 1048576)           0

 dense_1 (Dense)             (None, 2)                 2097154

=================================================================
Total params: 2097330 (8.00 MB)
Trainable params: 2097330 (8.00 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

سوال ۷)

الف)

ساختار:

۱. ورودی: تابع دارای هفت پارامتر است:

- model: این tensor ورودی ماژول Inception است.
- filters_1x1, filters_3x3_reduce, filters_3x3, filters_5x5_reduce, filters_5x5, filters_pool_proj: اینها تعداد فیلترها در لایه های کانولوشن مربوطه در ماژول Inception هستند.

۲. tower ها: ماژول Inception از چهار برج موازی تشکیل شده است:

- tower_1: این یک لایه کانولوشنال 1x1 است.

- tower_2: این tower دارای یک لایه کانولوشن 1x1 و به دنبال آن یک لایه کانولوشن 3x3 است.
- tower_3: این tower دارای یک لایه کانولوشن 1x1 و به دنبال آن یک لایه کانولوشن 5x5 است.
- tower_4: این tower دارای یک لایه 3x3 max pooling و به دنبال آن یک لایه کانولوشن 1x1 است.

۳. Concatenation: خروجی های هر چهار tower در امتداد محور کانال (axis=3) به هم متصل می شوند تا خروجی ماژول Inception را تشکیل دهند.

۴. خروجی: تابع tensor خروجی را پس از Concatenation برمی گرداند.

این ساختار به شبکه اجازه می‌دهد تا هم ویژگی‌های محلی را از طریق کانولوشن‌های کوچک‌تر و هم ویژگی‌های سطح بالاتر انتزاعی را از طریق کانولوشن‌های بزرگ‌تر به صورت موازی یاد بگیرد و کارایی و عملکرد شبکه را افزایش دهد. پیچیدگی های 1x1 (همچنین به عنوان پیچش های نقطه ای شناخته می شوند) برای کاهش ابعاد ورودی استفاده می شوند که به کنترل پیچیدگی محاسباتی مدل کمک می کند. حداکثر لایه ادغام برای ارائه عدم تغییر فضایی استفاده می شود.

۱. محاسبات کارآمد: ماژول Inception با انجام کانولوشن با اندازه فیلترهای متعدد به صورت موازی، امکان محاسبات کارآمدتر را فراهم می کند و نیاز به شبکه عمیق تر یا گسترده تر را کاهش می دهد.

۲. استخراج ویژگی چند سطحی: با اعمال چندین فیلتر کانولوشن با اندازه های مختلف (1x1، 3x3، 5x5) به طور همزمان به ورودی، شبکه می تواند اطلاعات را در مقیاس ها و پیچیدگی های مختلف دریافت کند. این به مدل اجازه می‌دهد تا الگوها را در ابعاد و پیچیدگی‌های فضایی مختلف تشخیص دهد.

۳. کاهش ابعاد: استفاده از پیچش 1x1 به عنوان روشی برای کاهش ابعاد عمل می کند. این امر پیچیدگی محاسباتی و تعداد پارامترها را بدون از دست دادن عمق در شبکه کاهش می دهد و باعث می شود مدل کارآمدتر و کمتر مستعد بیش از حد برازش شود.

۴. ادغام: گنجاندن یک شاخه ادغام موازی (معمولاً حداکثر ادغام) شکل دیگری از تجمع فضایی را فراهم می کند که به تغییر دادن ویژگی ها برای ترجمه های کوچک کمک می کند.

۵. بهبود عملکرد: شبکه‌های دارای ماژول‌های اولیه عملکرد بهتری را در مجموعه داده‌های معیار مختلف برای شناسایی و طبقه‌بندی تصویر نشان داده‌اند.

ب)

پارامتر گام در لایه های پیچشی یک شبکه عصبی کانولوشن (CNN) به طور قابل توجهی بر ابعاد فضایی نقشه های ویژگی تأثیر می گذارد. در اینجا به این صورت است:

۱. اندازه گام و ابعاد فضایی: اندازه گام تعیین می کند که فیلتر یا هسته چقدر در حجم ورودی حرکت می کند. گام ۱ فیلتر را هر بار یک پیکسل حرکت می دهد، در حالی که گام ۲ آن را هر بار دو پیکسل حرکت می دهد و غیره. یک گام بزرگتر منجر به ابعاد فضایی کوچک‌تر در نقشه ویژگی خروجی می‌شود، زیرا فیلتر در گام‌های بزرگ‌تر در سراسر ورودی حرکت می‌کند و حجم ورودی را سریع‌تر پوشش می‌دهد و پیکسل‌های خروجی کمتری تولید می‌کند.

۲. Downsampling: Stride اغلب به عنوان یک روش ساده برای نمونه برداری از نقشه های ویژگی استفاده می شود. با استفاده از یک گام بزرگتر از ۱، شبکه می تواند ابعاد فضایی نقشه های ویژگی را کاهش دهد و به طور موثر نمایش انتزاعی تری از ورودی را یاد بگیرد و پیچیدگی محاسباتی را کاهش دهد.

۳. میدان دید: یک گام بزرگتر میدان دریافتی نورون ها را در لایه خروجی افزایش می دهد، به این معنی که آنها بخش بزرگتری از ورودی را می بینند. این می تواند برای گرفتن ویژگی های در مقیاس بزرگتر مفید باشد، اما ممکن است باعث شود که شبکه ویژگی های محلی و ظریف تر را از دست بدهد.

۴. از دست دادن اطلاعات: یک گام بزرگتر ممکن است منجر به از دست دادن اطلاعات شود، زیرا موقعیت های کمتری توسط عملیات کانولوشن ارزیابی می شود. این امر به ویژه در صورتی صادق است که گام بزرگتر از اندازه فیلتر باشد، زیرا باعث می شود فیلتر به طور کامل از برخی از قسمت های ورودی عبور کند.

به طور خلاصه، در حالی که یک گام بزرگتر می تواند به کاهش پیچیدگی محاسباتی و افزایش میدان دریافت کمک کند، همچنین ممکن است منجر به نمونه برداری پایین، از دست دادن اطلاعات بالقوه، و کاهش توانایی گرفتن ویژگی های دقیق تر و محلی شود. بنابراین، انتخاب اندازه گام در CNN ها یک مبادله است که باید بر اساس وظیفه و مجموعه داده خاص به دقت در نظر گرفته شود.

پ)

۱. لایه های Conv2D: این لایه ها کانولوشن های دو بعدی را روی داده های ورودی انجام می دهند. آنها بلوک های ساختمان اولیه CNN ها هستند و مسئول استخراج ویژگی ها هستند. هر لایه Conv2D از مجموعه ای از

فیلترهای قابل یادگیری استفاده می کند. هر فیلتر در عرض و ارتفاع حجم ورودی پیچیده می‌شود، محصول نقطه‌ای را بین ورودی‌های فیلتر و ورودی محاسبه می‌کند و یک نقشه فعال‌سازی دوبعدی تولید می‌کند. این فرآیند برای هر فیلتر در لایه تکرار می شود، و در نتیجه یک پشته از نقشه های فعال سازی، یک برای هر فیلتر ایجاد می شود. پارامترهای کلیدی برای این لایه ها عبارتند از:

- فیلترها: این تعداد فیلترها در پیچیدگی است. هر فیلتر یک ویژگی خاص را در ورودی تشخیص می دهد.

- kernel_size: این اندازه فیلتری است که استفاده می شود (ارتفاع و عرض). ابعاد متداول عبارتند از 1x1، 3x3، و 5x5.

- padding: این می تواند "معتبر" یا "همان" باشد. اگر «یکسان» باشد، از padding روی ورودی لایه استفاده می‌شود تا اطمینان حاصل شود که خروجی دارای عرض و ارتفاع یکسان است. این اغلب برای حفظ ابعاد فضایی هنگام گام برداشتن روی تصاویر استفاده می شود.

- فعال سازی: این تابع فعال سازی است که باید از آن استفاده کنید. در این مورد، relu (واحد خطی اصلاح شده) استفاده می شود، که غیرخطی بودن را به مدل معرفی می کند و به کاهش مشکل گرادیان ناپدید شدن کمک می کند.

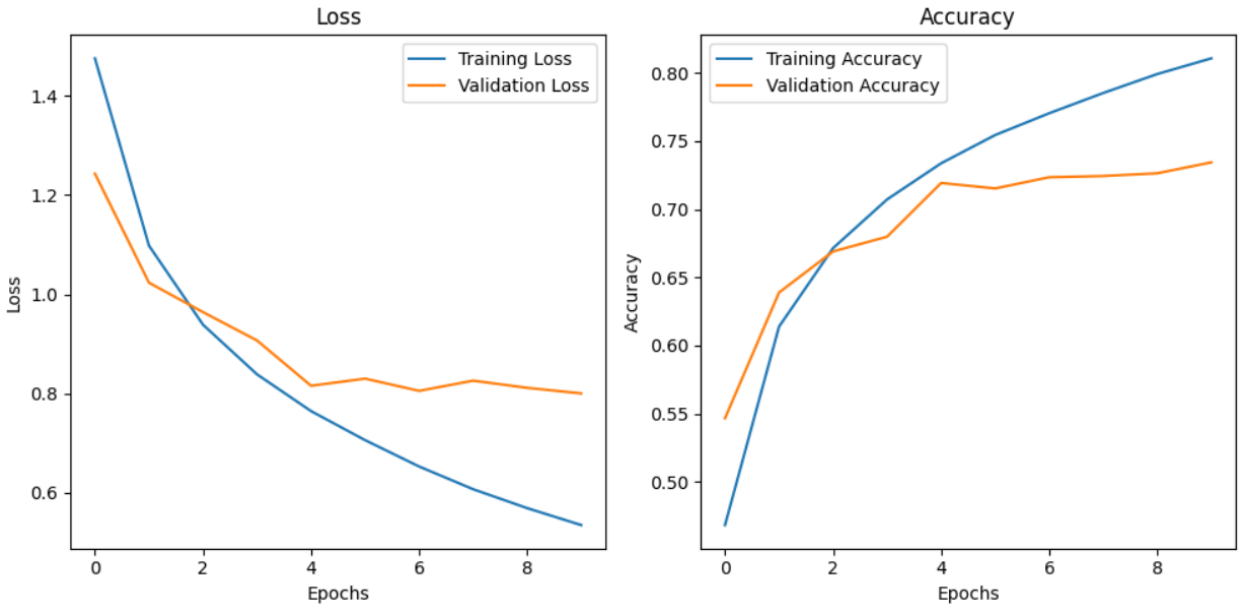
۲. لایه‌های MaxPooling2D: این لایه‌ها با گرفتن حداکثر مقدار روی پنجره تعریف‌شده با pool_size برای هر بعد در امتداد محور ویژگی‌ها، نمونه‌برداری فضایی را روی ورودی انجام می‌دهند. پنجره با «گام‌هایی» در امتداد هر بعد جابه‌جا می‌شود. استفاده از لایه‌های ادغام، رویکردی را برای نمونه‌برداری از نقشه‌های ویژگی با خلاصه کردن حضور ویژگی‌ها در تکه‌های نقشه ویژگی فراهم می‌کند. این برای استخراج ویژگی مهم است زیرا سطحی از تغییر ناپذیری ترجمه را به نمایش داخلی ورودی معرفی می کند.

۳. Inception Module: این ساختار پیچیده ای از لایه های کانولوشن و لایه های ادغام است. این به شبکه اجازه می‌دهد تا هم ویژگی‌های محلی را از طریق کانولوشن‌های کوچک‌تر و هم ویژگی‌های انتزاعی سطح بالاتر را از طریق کانولوشن‌های بزرگ‌تر به صورت موازی یاد بگیرد و کارایی و عملکرد شبکه را افزایش دهد. پیچیدگی های 1x1 برای کاهش ابعاد ورودی استفاده می شود که به کنترل پیچیدگی محاسباتی مدل کمک می کند.

به طور خلاصه، لایه های کانولوشن مسئول استخراج ویژگی در داده های ورودی هستند. آنها یاد می گیرند که ویژگی های مختلف را در داده های ورودی شناسایی کنند (مانند لبه ها، گوشه ها و غیره در داده های تصویر)، و پیچیدگی ویژگی هایی که یاد می گیرند با عمق لایه در شبکه افزایش می یابد. لایه های ادغام برای کاهش ابعاد فضایی داده ها، کنترل بیش از حد برازش و معرفی بی تغییری ترجمه استفاده می شود. ماژول Inception به شبکه اجازه می دهد تا ویژگی ها را در مقیاس های مختلف به صورت موازی یاد بگیرد و سپس خروجی ها را به یکدیگر متصل کند. این کار شبکه را کارآمدتر می کند و عملکرد را بهبود می بخشد. تابع فعال‌سازی غیرخطی بودن را به مدل معرفی می‌کند که به مدل اجازه می‌دهد الگوهای پیچیده در داده‌ها را بیاموزد. padding تضمین می کند که ابعاد فضایی داده ها پس از پیچش ها حفظ می شود. فیلترها و اندازه آنها پیچیدگی و مقیاس ویژگی هایی را که مدل می تواند یاد بگیرد را تعیین می کند. گام‌ها تعیین می‌کنند که فیلتر هنگام انجام پیچش چقدر جابجا شود. همه این عوامل با هم عملکرد و کارایی مدل را تعیین می کنند.

نتایج:

```
Epoch 1/10
1563/1563 [==============================] - 15s 8ms/step - loss: 1.4758 - accuracy: 0.4684 - val_loss: 1.2433 - val_accuracy: 0.5467
Epoch 2/10
1563/1563 [==============================] - 11s 7ms/step - loss: 1.0982 - accuracy: 0.6139 - val_loss: 1.0235 - val_accuracy: 0.6389
Epoch 3/10
1563/1563 [==============================] - 11s 7ms/step - loss: 0.9388 - accuracy: 0.6714 - val_loss: 0.9642 - val_accuracy: 0.6690
Epoch 4/10
1563/1563 [==============================] - 11s 7ms/step - loss: 0.8388 - accuracy: 0.7071 - val_loss: 0.9071 - val_accuracy: 0.6798
Epoch 5/10
1563/1563 [==============================] - 12s 8ms/step - loss: 0.7647 - accuracy: 0.7336 - val_loss: 0.8158 - val_accuracy: 0.7192
Epoch 6/10
1563/1563 [==============================] - 12s 8ms/step - loss: 0.7061 - accuracy: 0.7543 - val_loss: 0.8302 - val_accuracy: 0.7152
Epoch 7/10
1563/1563 [==============================] - 12s 8ms/step - loss: 0.6530 - accuracy: 0.7703 - val_loss: 0.8055 - val_accuracy: 0.7234
Epoch 8/10
1563/1563 [==============================] - 12s 8ms/step - loss: 0.6072 - accuracy: 0.7852 - val_loss: 0.8261 - val_accuracy: 0.7243
Epoch 9/10
1563/1563 [==============================] - 12s 8ms/step - loss: 0.5691 - accuracy: 0.7991 - val_loss: 0.8116 - val_accuracy: 0.7263
Epoch 10/10
1563/1563 [==============================] - 11s 7ms/step - loss: 0.5350 - accuracy: 0.8106 - val_loss: 0.8003 - val_accuracy: 0.7343
```


Loss
Training Loss
Validation Loss
Loss
Epochs
Accuracy
Training Accuracy
Validation Accuracy
Accuracy
Epochs