بسمه تعالی

دانشکده مهندسی کامپیوتر

بینایی کامپیوتر

نام استاد: دکتر محمدی

تمرین یازدهم

نام دانشجو: آرمان حیدری

شماره دانشجویی: ۹۷۵۲۱۲۵۲

دی ۱۴۰۱

فهرست

پاسخ سوال اول ۳

الف) ۳

ب) ۳

پ) ۳

ت) ۴

ث) ۴

منابع) ۴

پاسخ سوال دوم ۵

الف) ۵

ب) ۵

پ) ۶

منابع) ۶

پاسخ سوال سوم ۷

الف) ۷

ب) ۷

پ) ۷

ت) ۷

پاسخ سوال اول

الف)

زیرا این لایه ها برای استخراج ویژگی های محلی مناسب هستند و لایه های fully connected به تمام تصویر متصل هستند که ممکن است در خیلی از مسائل مورد نظر ما نباشد. همچنین تعداد پارامترهای قابل یادگیری با استفاده از کانولوشنی بسیار کمتر است و پردازش موازی انجام می شود و می توانیم شبکه های عمیق تری داشته باشیم.

ب)

طبق فرمول زیر:

- $W_2 = (W_1 - F + 2P)/S + 1$
- $H_2 = (H_1 - F + 2P)/S + 1$
- $D_2 = K$

و میدانیم که w2 = w1 پس 16 = 16 – 5 + 2p + 1 و p=2. همچنین برای H هم تفاوتی ندارد چون عکس مربع بوده است.

تعداد پارامترهای این لایه طبق فرمول k*(m*n*d+1) برابر است با:

Number of parameters = (5*5*5 + 1) * 16 = 126 * 16 = 2016

پ)

طول و عرض هردو با رابطه بخش قبل برابر 28 می شوند. 28 = 1 + 0*2 + 5 – 32

و چون ۳ تا بود، خروجی پس از این لایه 3*28*28خواهد بود.

حالا همان عکس را به دو لایه گفته شده می دهیم. که مشابه قبل، پس از عبور از اولی به ابعاد 9*30*30 میرسیم. و بعد به لایه مشابهی دوباره میدهیم که 9*28*28 خواهد شد.

ت)

روش average pooling تصویر را smooth می کند و از این رو ممکن است هنگام استفاده از این روش pooling، ویژگی های واضح شناسایی نشود. Max Pooling پیکسل های روشن تر را از تصویر انتخاب می کند. زمانی مفید است که پس‌زمینه تصویر تاریک است و ما فقط به پیکسل‌های روشن‌تر تصویر علاقه‌مندیم. ولی اگر تصویر نویزی باشد استفاده از average توصیه می شود.

اما global pooling به طور کلی در لایه های میانی استفاده نمی شود و پس از آخرین کانولوشنی و قبل از fully connected استفاده می شود که جایگزین Flatten شود و تعداد پارامترها را کاهش دهد.

ث)

ایده اصلی resnet این است که چیزی مانند shortcut بین لایه ها ایجاد میکند که باعث جلوگیری از gradient vanishing میشود و میتوان تعداد لایه ها را بسیار بالا برد.

ایده اصلی vgg هم استفاده از کانولوشن های ۳*۳ بود و مثل شبکه های قبل از خودش از کانولوشن های بزرگ استفاده نمیکرد. زیرا بیان میکرد که دو تا 3*3 پیاپی میتوانند همان کارکرد یک کانولوشنی بزرگ را داشته باشند. به این شکل پیاده سازی هم به صورت بلکو های یکسان می شد و ساده تر بود.

دلیل سریعتر بودن resnet استفاده از bottleneck های ۱ در ۱ است که که برای ساختن بلاک استفاده می شوند. که تعداد ضرب های ماتریسی را به طرز معناداری کاهش می دهد. ولی در vgg چنین مواردی نداریم.

منابع)

- https://towardsdatascience.com/understanding-and-calculating-the-number-of-parameters-in-convolution-neural-networks-cnns-fc88790d530d#:~:text=Number%20of%20parameters%20in%20a%20CONV%20layer%20would%20be%20%3A%20((,1)*number%20of%20filters).
- https://datagen.tech/guides/computer-vision/resnet-50/

پاسخ سوال دوم

همانطور که در نوتبوک مشخص است، شبکه های دلخواهی تعریف کرده ام و تعداد پارامترها به ترتیب حدود ۳ میلیون و ۱.۵ میلیون هستند و شرط نصف بودن رعایت شده است. از تابع فعالسازی softmax در لایه آخر استفاده کرده ام که با تابع ضرر categorical cross_entropy به خوبی سازگار است.

الف)

```
Fully Connected Model
Epoch 1/5
1563/1563 [==============================] - 6s 4ms/step - loss: 1.9495 - accuracy: 0.3213
Epoch 2/5
1563/1563 [==============================] - 6s 4ms/step - loss: 1.7356 - accuracy: 0.3795
Epoch 3/5
1563/1563 [==============================] - 6s 4ms/step - loss: 1.6874 - accuracy: 0.3947
Epoch 4/5
1563/1563 [==============================] - 7s 4ms/step - loss: 1.6440 - accuracy: 0.4108
Epoch 5/5
1563/1563 [==============================] - 7s 4ms/step - loss: 1.6214 - accuracy: 0.4211


Loss and Accuracy on Test set :
313/313 [==============================] - 1s 3ms/step - loss: 1.6336 - accuracy: 0.4189


Convolutional Model
Epoch 1/5
1563/1563 [==============================] - 9s 5ms/step - loss: 1.4228 - accuracy: 0.4831
Epoch 2/5
1563/1563 [==============================] - 8s 5ms/step - loss: 1.0200 - accuracy: 0.6399
Epoch 3/5
1563/1563 [==============================] - 8s 5ms/step - loss: 0.8139 - accuracy: 0.7140
Epoch 4/5
1563/1563 [==============================] - 8s 5ms/step - loss: 0.6513 - accuracy: 0.7704
Epoch 5/5
1563/1563 [==============================] - 8s 5ms/step - loss: 0.4984 - accuracy: 0.8232


Loss and Accuracy on Test set :
313/313 [==============================] - 1s 4ms/step - loss: 1.0817 - accuracy: 0.6869
```

میبینیم که در شبکه کانولوشنی با وجود پارامترهای کمتر، خطای کمتر و دقت بیشتری روی داده های train و test داریم. بله به طور معمول خطای کمتر نمایانگر دقت بیشتر است. چون برای محاسبه ضرر، بردار های one hot با خروجی لایه آخر softmax مقایسه می شود (فرمول لگاریتمی)، و هر چقدر کلاس پیشبینی شده درست تر و به ۱ نزدیک باشد، حاصل خطا کمتر می شود.

ب)

در مدل کاملا متصل حدود ۶ ثانیه و در مدل کانولوشنی حدود ۸ ثانیه است.

پ)

خیر میبینیم که تعداد پارامتر کمتر، منجر به زمان بیشتری شده است پس ارتباطی که شاید تصور کنیم مستقیم است وجود ندارد. در واقع چون فاکتورهای مختلفی در زمانی که طول می کشد تاثیر دارد. مثلا تعداد لایه ها عامل بسیار مهمی است چون با این که پردازش ها در GPU به صورت موازی انجام می شود، ولی هر لایه باید تمام شود و بعد لایه بعدی شروع شود و این قسمت پردازش sequential است. در شبکه دوم پارامترهای کمتر ممکن است منجر به زمان کمتر شود اما از طرفی عمق بیشتر تاثیر عکس دارد. همچنین عواملی مانند توابع فعالسازی و ضرر و optimizer و پیش پردازش داده و ... هم موثرند که بین این دو شبکه یکسان بوده است.

منابع)

- https://birupakshyamahapatra.com/effect-of-model-characteristics-on-training-time-of-deep-learning-problems/

پاسخ سوال سوم

الف)

این کار را در دو مرحله یعنی ابتدا اضافه کردن حاشیه سیاه رنگ به مقدار مورد نیاز به صورت دستی، و بعد تبدیل عکس مربعی به ابعاد خواسته شده با تابع resize در opencv انجام داده ام.

ب)

پس از لود کردن ResNet50 با وزن های رندوم (قسمت های fully connected را لود نمیکنیم) یک global average pooling و دو لایه کاملا متصل به بلاک های کانولوشنی آن اضافه میکنیم و شبکه را از پایه آموزش میدهیم.

پ)

دقیقا مشابه قسمت قبل عمل میکنیم فقط با دو تفاوت، یکی این که وزن های آموزش داده شده روی دیتاست ImageNet را برای بلوک های کانولوشنی در نظر میگیریم، دوم این که این بلوک ها را از حالت قابل آموزش خارج میکنیم تا فقط لایه های انتهاییی که خودمان افزودیم آموزش ببینند و اصطلاحا fine tune میکنیم.

ت)

- مدل resnet با وزن های رندوم:

```
Epoch 1/3
65/65 [==============================] - 75s 964ms/step - loss: 1.8292 - acc: 0.5262
Epoch 2/3
65/65 [==============================] - 63s 958ms/step - loss: 0.3188 - acc: 0.8922
Epoch 3/3
65/65 [==============================] - 63s 965ms/step - loss: 0.2051 - acc: 0.9299
<keras.callbacks.History at 0x7f7fb819fb80>

33/33 [==============================] - 19s 534ms/step - loss: 9.0497 - acc: 0.0974
[9.049718856811523, 0.09742765128612518]
```

- مدل resnet با استفاده از image net:

```
Epoch 1/3
65/65 [==============================] - 37s 528ms/step - loss: 2.7627 - acc: 0.2215
Epoch 2/3
65/65 [==============================] - 34s 524ms/step - loss: 1.8914 - acc: 0.5062
Epoch 3/3
65/65 [==============================] - 34s 524ms/step - loss: 1.3411 - acc: 0.6505


33/33 [==============================] - 18s 498ms/step - loss: 1.1146 - acc: 0.7502
```

با همین ۳ آموزش مدل اول به شدت دچار overfit شده است. چون تعداد خیلی زیادی پارامتر رندوم دارد که فقط داده ها را حفظ میکنند. در حالی که با fine tuning تقریبا تعداد پارامتر ها را یک دهم کرده ایم و فقط دو لایه اخر را آموزش داده ایم و به استخراج ویژگی های imagenet اکتفا کرده ایم که فرض بدی هم نیست. دقت تست این حالت ۷۵ و حالت رندوم فقط ۹ درصد است.