به نام خدا

تمرین سری نهم درس یادگیری عمیق

دکتر محمدی

محمد یارمقدم

۹۶۴۶۲۱۰۴

سوال اول)

الف)

تفاوت این دو لایه در حرکت در ابعاد مختلف است. کانولوشن دو بعدی(Conv 2D) بر روی دو بعد x و y حرکت می کند در حالیکه کانولوشن سه بعدی بر روی سه بعد x و y و z حرکت می کند.

ورودی این دو لایه نیز متفاوت است. در هر دوی این لایه ها در ورودی batch_size و channels وجود دارد و ست میشود. تفاوت در آرگومان دوم است که در لایه دوبعدی (H, W) است و در لایه سه بعدی (H, w, D) است.

Conv 2D: input → (batch_size, (H, W), channels)

Conv 3D: input → (batch_size, (H, w, D), chnnels)

موارد کاربرد این دو لایه نیز متفاوت است. لایه دو بعدی برای یافتن ویژگی ها استفاده می شود و لایه سه بعدی برای یافتن ویژگی های ویدیو استفاده می شود. چون ویدیو در هر زمان فریم جداگانه دارد و بنابراین به جای دو بعد که در تصاویر موجود است سه بعد خواهد داشت.

ب)

علت انتخاب فیلتر مربعی این است که شانس یافتن یک الگو در داده های ورودی بیشتر نشود. به طور مثال ممکن است یک الگو می تواند فقط خطوط افقی یا عمودی باشد یا فقط تشخیص دایره یا مربع در عکس باشد اما هر دو ویژگی مهم هستند و نیاز است که شبکه آن ها را یادگیری و ثبت کند.

به علاوه با فیلتر های که به شکل مربع هستند، می توان الگو های متفاوتی را ایجاد کرد و ویژگی ها متفاوتی را از شبکه دریافت کرد که این موضوع در فیلتر ها با اشکال دیگر موجود نیست.

سایز و تعداد و اندازه فیلتر ها به تصویر و مشخصات و جزییات آن بستگی دارد. در تصاویر بزرگ با جزییات بالا و پیچیده معمولا استفاده از فیلتر های بزرگ تر مانند 7*7 یا 9*9 مرسوم است. در ای حالت الگو های بزرگ تر و پیچیده تر تولید شده و از آن در تصاویر استفاده می شود. در عکس های کوچک تر که جزییات کمتر و پیچیدگی کمتری دارند استفاده از فیلتر های 3*3 یا 5*5 مرسوم است. در برخی الگو ها تعداد پیکسل های مورد نیاز برای بدست ویژگی زیاد است پس اندازه فیلتر نیز طبیعتا باید بزرگ باشد. مرسوم ترین اندازه فیلتر ها از 3*3 آغاز و تا 11*11 ادامه دارد. اما سایر ابعاد نیز با توجه به ابعاد و ویژگی های عکس ها انتخاب می شوند.

ج)

در شبکه های عمیق از لایه pooling برای کاهش پیچیدگی های محاسباتی و واریانس استفاده می شود. چهار گونه برای این لایه وجود دارد:

- Average pooling
- Adaptive pooling
- Maximum pooling
- Minimum pooling

اعمال کاهش واریانس در این لایه با کم کردن پارامتر های قابل یادگیری انجام می شود و به علاوه سرعت شبکه نیز افزایش می یابد.

حال به مقایسه تفاوت این چهار گونه می پردازیم:

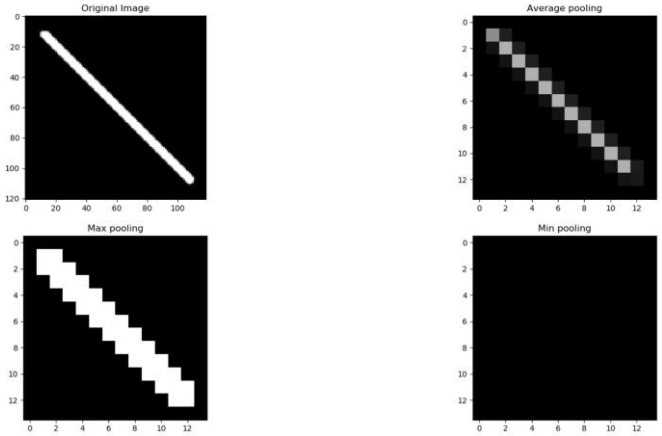
در نوع اول average pooling است. از آنجا که تصاویر در نهایت مجموعه ای از داده های مرتب شده هستند، پس مقدار متوسط مقادیری که در فیلتر ظاهر می شوند را نگه می دارد.

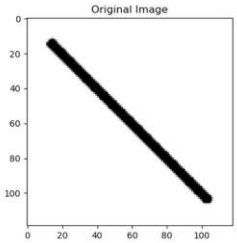
در نوع دوم adaptive pooling قرار دارد. در این نوع به جای ست کردن هایپر پارامتر های شبکه به صورت دستی، شبکه را طوری تنظیم می کنیم که در روند یادگیری خود شبکه بیاموزد که چه مقادیری برای این مقادیر مناسب است. هایپر های مختلفی آموزش دیده می شود مانند: stride و padding و ... . در این حالت تنها چیزی که باید تنظیم شود سایز خروجی در لایه pooling است. مقادیر که برای هر فیلتر قرار می گیرد یکی از سه حالت max، min و یا average از مقادیر موجود در فیلتر است.

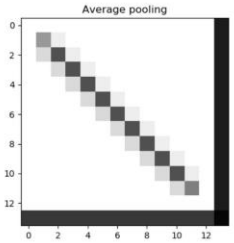
در نوع سوم max pooling قرار دارد. در هر قسمت از فیلتر بزرگ ترین مقدار موجود در خروجی فیلتر قرار می دهد.

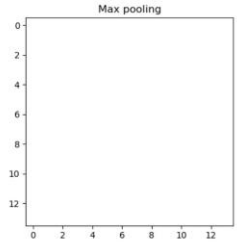
در نوع آخر نیز min pooling قرار دارد که برعکس حالت بالا بوده و کمترین مقدار را در هر قسمت از فیلتر انتخاب کرده و در خروجی قرار می دهد.

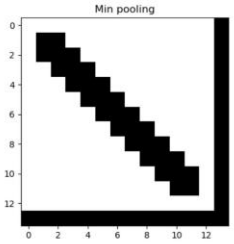
کاربرد هر کدام از ای لایه ها متفاوت است. Max pooling نقاط روشن عکس ها را انتخاب می کند و در مواردی که پس زمینه روشن است ضعیف عمل کند. در حالت مقابل min pooling قرار دارد که نقاط تاریک را انتخاب می کند و در حالتی که پس زمینه تاریک باشد نمی تواند خروجی مطلوبی داشته باشد. حالت average pooling نیز ویژگی هایی مانند لبه ها و خطوط را در عکس ها یافته و انتخاب می کند.











لینک های کمکی:

https://stackoverflow.com/questions/49003346/why-convolutional-nn-kernel-size-is-often-selected-as-a-square-matrix

https://medium.com/@bdhuma/which-pooling-method-is-better-maxpooling-vs-minpooling-vs-average-pooling-95fb03f45a9

سوال دوم)

در ابتدا عکس ورودی که دارای ابعاد 512*512 است را در فولدر content در درایو گوگل ذخیره کرده و سپس با استفاده از تابع imread در پایتون عکس را از درایو خوانده و در متغیر input_image ذخیره می کنیم. سپس مراحل مشخص شده در کد را از روی لینک پیش می رویم. در این قسمت تابع convolve داده شده را نوشته و توسط آن خروجی فیلتر های مختلف روی عکس داده شده را بدست می آوریم.

در قسمت زیر خروجی فیلتر های مختلف برای عکس را به ترتیب از سمت راست به چپ نمایش داده شده است:

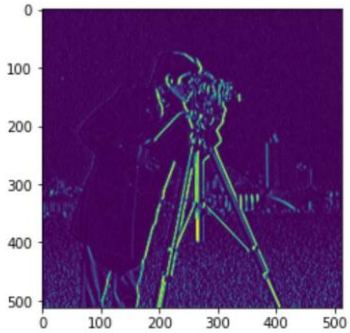
$$\frac{1}{9}\times\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$$\begin{bmatrix}-1&-1&-1\\-1&8&-1\\-1&-1&-1\end{bmatrix}$$

$$\begin{bmatrix}-1&-2&-1\\0&0&0\\1&2&1\end{bmatrix}$$

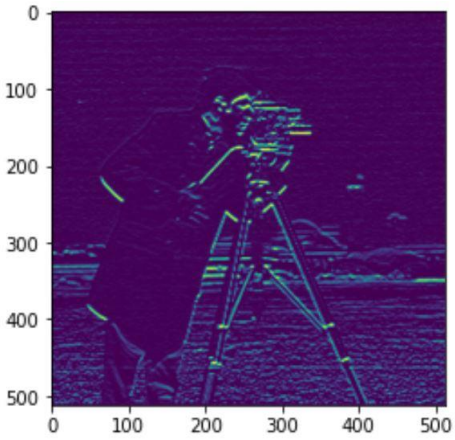
$$\begin{bmatrix}-1&0&1\\-2&0&2\\-1&0&1\end{bmatrix}$$

خروجی فیلتر سمت راست:



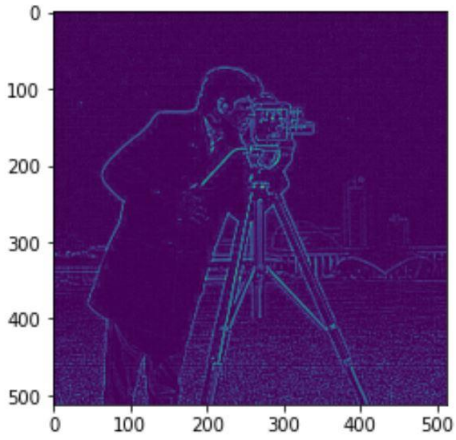
این فیلتر خطوط عمودی تصویر را بدست می آورد و انتخاب می کند. زیرا در مواردی خروجی عددی نزدیک به صفر نیست که پیکسل های ستون اول مقدار متفاوتی از مقدار پیکسل های ستون سوم داشته باشد.

خروجی فیلتر وسط سمت راست:



این فیلتر خطوط افقی تصویر را انتخاب می کند. زیرا در مواردی خروجی خواهد داشت که پیکسل های سطر اول مقدار متفاوتی از پیکسل های سطر سوم داشته باشد.

خروجی فیلتر وسط سمت چپ:

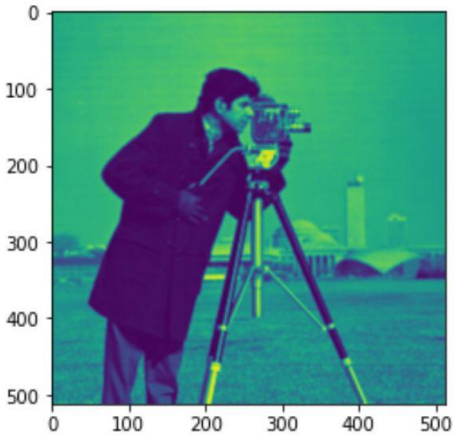


این فیلتر نقاطی را انتخاب و مشخص می کند که تغییر رنگ اتفاق افتاده است. چون اگر در نواحی باشیم که مقادیر پیکسل ها نزدیک باشد مقدار خروجی صفر می شود ولی اگر در نقاط لبه باشد که رنگ عوض شده باشد و مقادیر اختلاف زیادی دارد خروجی غیر صفر و آن نقطه مشخص می شود.

خروجی فیلتر سمت چپ:

این فیلتر فقط برای smoothing کاربرد دارد و لبه های تصویر را در اصطلاح soft می کند. ولی تصویر تغییر محسوسی نخواهد داشت.

چون در هر ناحیه فقط مقادیر هر ناحیه را در یک ضریب ضرب و جمع می کند. بنابراین در کلیت عکس تفاوتی ایجاد نمی کند.



سوال سوم)

الف)

در این قسمت هدف معرفی keras tuner است. این ابزار در واقع یک بهینه ساز خودکار است. به این صورت که هایپرپارامتر های شبکه را در در تکرار ها مداوم مقداردهی می کند تا مقادیر بهترین نتیجه را بتواند بدست بیاورد. فقط می بایست بازه موردنظر برای هر کدام از پارامتر ها را مشخص کرد.

توابع این ابزار شامل randomsearch و search_space_summary است. توسط این توابع می توان تعیین کرد که به ترتیب کار tuning پارامتر های شبکه چند بار و روی چه مدلی شورت گیرد و خلاصه هر مدل برای پارامتر های شبکه چیست.

ب)

چهار نوع tuner در این ابزار موجود است:

- Sklearn
- Hyperband
  بررسی تمامی ترکیب های رندوم و آموزش مدل در تعداد epoch پایین و انتخاب بهترین مدل
- RandomSearch
  انتخاب رندوم ترکیب هایپرپارامتر ها
- BayesianOptimizer
  مقداردهی اولیه به پارامتر ها و تعیین مقادیر بعدی توسط آنان(بدون استفاده از تصمیم گیری رندوم)

از آنجا جدیدترین نسخه Hyperband و مشکلات نسخه های قبلی در آن حل شده است و ورژن به روز شده RandomSearch است، انتخاب در این تمرین این نسخه است.

ج)

در ابتدا کتابخانه CIFAR را لود کرده و داده های آنرا برای ست شدن در لایه های تعیین شده نرمالایز و reshape می کنیم.

ابعاد تصاویر در این داده ها 32*32 است و ۵۰ هزار داده آموزش و ۱۰ هزار داده تست در آن موجود است. ۱۰ کلاس نیز در خروجی آن وجود دارد.

در مرحله بعد مدل را با ویژگی های خواسته شده تشکیل می دهیم. از آنجا که داده ورودی را reshape کردیم بنابراین در اولین لایه از conv با فیلتر های 3*3 و تعداد unit در بازه (32, 256) استفاده کردم.

سپس طبق نمونه های موجود در لینک لایه های بعدی را به صورت زیر تعیین کردم. در بین لایه های از تابع choice برای انتخاب بهترین بهینه ساز از بین SGG و RMSProp و Adam انتخاب کردم.

لایه های کاملا متصل را با تعداد نورون در بازه (32, 256) استفاده کردم. و تابع ضرر در بهینه ساز ها از categorical استفاده کردم و learning_rate آن را در بازه (0.001, 0.0001) قرار دادم.

در نهایت از hyperband به عنوان tuner استفاده و تابع search را روی مدل ساخته شده با max_epoch=7 و executable_per_trial=4 استفاده کردم. در نهایت پس از ران شدن شبکه در بیست epoch و روی 30% داده اعتبارسنجی نتایج زیر بدست آمد:

{'input_units': 128, 'num_of_conv_layers': 2, 'conv_0_units': 128, 'num_of_dense_layers': 2, 'num_of_nodes': 256, 'optimizer': 'RMSprop', 'lr': 0.00101, 'tuner/epochs': 7, 'tuner/initial_epoch': 3, 'tuner/bracket': 1, 'tuner/round': 1, 'conv_1_units': 32, 'tuner/trial_id': '78c7f18adf3a0eececbd80ede0b3a7a82'}
Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 128)	3584
activation (Activation)	(None, 30, 30, 128)	0
max_pooling2d (MaxPooling2D)	(None, 15, 15, 128)	0
conv2d_1 (Conv2D)	(None, 13, 13, 128)	147584
activation_1 (Activation)	(None, 13, 13, 128)	0
conv2d_2 (Conv2D)	(None, 11, 11, 32)	36896
activation_2 (Activation)	(None, 11, 11, 32)	0
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 32)	0
flatten (Flatten)	(None, 800)	0
dense (Dense)	(None, 256)	205056
activation_3 (Activation)	(None, 256)	0
dense_1 (Dense)	(None, 256)	65792
activation_4 (Activation)	(None, 256)	0
dense_2 (Dense)	(None, 10)	2570
activation_5 (Activation)	(None, 10)	0

```
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 128)       3584

 activation (Activation)     (None, 30, 30, 128)       0

 max_pooling2d (MaxPooling2D  (None, 15, 15, 128)      0
 )

 conv2d_1 (Conv2D)           (None, 13, 13, 128)       147584

 activation_1 (Activation)   (None, 13, 13, 128)       0

 conv2d_2 (Conv2D)           (None, 11, 11, 32)        36896

 activation_2 (Activation)   (None, 11, 11, 32)        0

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 32)         0
 2D)

 flatten (Flatten)           (None, 800)               0

 dense (Dense)               (None, 256)               205056

 activation_3 (Activation)   (None, 256)               0

 dense_1 (Dense)             (None, 256)               65792

 activation_4 (Activation)   (None, 256)               0

 dense_2 (Dense)             (None, 10)                2570

 activation_5 (Activation)   (None, 10)                0

=================================================================
```

```
313/313 [==============================] - 3s 7ms/step - loss: 1.0422 - accuracy: 0.6523
loss: 1.0422279834747314
accuracy: 0.6523000001907349
```

در این مورد به دقت 65% در داده اعتبار دست یافتیم که در تعداد بیست epoch نتیجه قابل قبولی است.

لینک های کمکی:

https://medium.com/swlh/hyperparameter-tuning-in-keras-tensorflow-2-with-keras-tuner-randomsearch-hyperband-3e212647778f

https://medium.com/analytics-vidhya/convolutional-neuronal-network-with-keras-tuner-on-cifar-10-b4271ca4643d

سوال چهارم)

تعداد پارامترهای شبکه را در هر لایه محاسبه و سپس جمع می‌کنیم.

first layer:

۲۰ فیلتر با سایز ۷×۷ دارد که هر کدام یک بایاس نیز دارد.

Parameter = ۷×۷ + ۱ = ۵۰ ⇒ total-parameter = ۵۰×۲۰ = ۱۰۰۰

خروجی کانوالو شده (۲۲,۲۲,۱) است. پس:

output: (28,28) - ((7,7) - (1,1)) = (22,22)

از آنجا که تعداد فیلتر ها ۲۰ است پس خروجی در نهایت (22,22,20) می‌شود.

Second layer:

لایه pooling پارامتر قابل یادگیری ندارد. از آنجا که stride = 2 است پس سایز خروجی نصف می‌شود پس سایز (11,11,20) می‌شود.

third layer:

این لایه کانولوشن ۱۰ فیلتر با سایز ۵×۵ دارد و چون هر فیلتر یک بایاس دارد داریم:

total-parameters = (۵×۵×۲۰+۱) × ۱۰ = ۵۰۱۰

این لایه در خروجی دارای فرمت زیر است:

output = (11,11) - (۵,۵) = (۷,۷)

از آنجا که تعداد فیلتر ۱۰ است پس در نهایت سایز (۷,۷,۱۰) است.

fourth layer:

ضربت لایه محلی آن است که پارامترهای هر فیلتر مختص خودش است.

پس مقداری که در مراحل قبل محاسبه کردیم در تعداد usecase فیلتر ضرب می‌شود.

۲ فیلتر با سایز ۳×۳ داریم. عمق نیز ۱۰ است پس داریم:

$(۳\times۳\times۱۰)+۱\times۲ = ۱۸۲$ ⇒ پارامترهای همه فیلترها

$total\_parameter = ۱۸۲ \times ۵\times۵ = ۴۵۵۰$

چون هر مجموعه ۵۵ بار روی ورودی قرار می‌گیرد و پارامتر یکسان نیست

$output = (۷,۷)-(۳,۳)$

$= (۵,۵)$

Fifth layer:

این لایه پارامتر ندارد.

reshape $(۵,۵,۲)$ to (۵۰)

Sixth layer:

هر لایه ۱۰ نورون دارد که ۵۰ ورودی و ۱ بایاس دارد:

$total\_parameter: (۵۰+۱)\times۱۰ = ۵۱۰$

$\Rightarrow final = \underline{۱۱۰۷۰}$