بینایی ماشین با استفاده از شبکه‌های عصبی پیچشی[1]

نگاه کلی:

این فصل در مورد بینایی ماشین و اجرای آن با استفاده از شبکه‌های عصبی است. شما پردازش تصویر و ساختن مدل‌های دسته‌بندی را با استفاده از شبکه‌های عصبی پیچشی یاد خواهید گرفت. ما همچنین در مورد معماری شبکه‌های عصبی پیچشی و تکنیک‌های چگونگی پیاده‌سازی آن‌ها، از جمله تجمیع حداکثری[2] و مسطح‌سازی[3]، نگاشت ویژگی‌ها[4] و تشخیص ویژگی‌ها[5]، بحث خواهیم کرد. در انتهای این فصل شما قادر خواهید بود که نه‌تنها دسته‌بند[6] خود را بسازید، بلکه آن را به‌صورت کارایی ارزیابی کنید.

مقدمه

در فصل گذشته نحوه‌ی ارزیابی مدل با جزئیات پوشش داده شد. ما در مورد دقت[7] و دلایلی که ممکن است باعث اشتباه شود صحبت کردیم. مخصوصاً این اشتباهات ممکن است در مجموعه داده‌هایی[8] که در آن، دسته‌ها به‌شدت نامتعادل هستند اتفاق بیوفتد. در این مجموعه‌های داده موارد مثبت نسبت به موارد منفی بسیار نادر هستند مانند مجموعه داده مربوط به پیش‌بینی تندبادها در اقیانوس آرام یا پیش‌بینی این که آیا فردی وام خود را پرداخت می‌کند یا خیر.

---

[1] Convolutional Neural Networks (CNNs)
[2] Max pooling
[3] flattening
[4] Feature mapping
[5] feature detection
[6] classifier
[7] accuracy
[8] datasets

برای مقابله با این عدم تعادل، تکنیک‌هایی آموختیم مبنی بر اینکه چگونه مدل را بهتر ارزیابی کنیم. از جمله این تکنیک‌ها محاسبه‌ی سنجه‌های ارزیابی مدل مانند حساسیت[9]، صحت تشخیص[10]، نرخ پاسخ‌های مثبت غلط[11]، مساحت زیر منحنی، و کشیدن نمودار ROC است. در این فصل چگونگی دسته‌بندی نوع دیگری از مجموعه داده (مجموعه داده عکس) را یاد خواهیم گرفت. همان‌طور که خواهید دید دسته‌بندی عکس بسیار کاربردی است و کاربردهای زیادی در دنیای واقعی دارد.

بینایی ماشین یکی از مفاهیم بسیار مهم در یادگیری ماشین و هوش مصنوعی است. با وجود تلفن‌های همراه که روزانه عکس می‌گیرند، داده‌های تولید شده‌ای که به‌صورت عکس هستند، به‌صورت نمایی در حال افزایش هستند. بنابراین نیاز به متخصصانی که در حوزه بینایی ماشین تخصص دارند بیشتر از هر زمانی است. با استفاده از فرآیندهای این حوزه، صنعت بهداشت در آستانه‌ی یک انقلاب قرار دارد.

این فصل شما را با بینایی ماشین و صنایعی که بینایی ماشین در آن‌ها به‌کار می‌رود، آشنا می‌کند. ما همچنین در مورد شبکه‌های عصبی پیچشی (CNN)، که استفاده‌شده‌ترین نوع شبکه عصبی برای پردازش تصویر است، آشنا می‌شویم. مانند همه شبکه‌های عصبی، CNN نیز از نورون‌هایی تشکیل شده‌است که ورودی را دریافت می‌کنند. این ورودی‌ها با استفاده از جمع وزنی و توابع فعال‌سازی[12] پردازش می‌شوند. برخلاف شبکه‌های عصبی مصنوعی (ANN)[13] که یک بردار را به‌عنوان ورودی دریافت می‌کنند، شبکه عصبی پیچشی تصویر را به‌عنوان ورودی دریافت می‌کند. در این فصل شبکه‌های عصبی پیچشی با جزئیات بیشتری مورد مطالعه قرار می‌گیرد. به‌علاوه مفاهیمی مانند تجمیع حداکثری، مسطح‌سازی، نگاشت ویژگی‌ها و انتخاب ویژگی‌ها[14]، پوشش داده می‌شوند. ما از Keras به‌عنوان ابزار پردازش تصویر روی تصاویر واقعی استفاده می‌کنیم.

[9] sensitivity
[10] specificity
[11] false positive rate
[12] activation functions
[13] artificial neural network (ANN)
[14] feature selection

بینایی ماشین

برای شناخت بینایی ماشین، بهتر است از بینایی انسان صحبت کنیم. بینایی انسان، توانایی چشم و مغز انسان برای دیدن و شناختن اشیاء است. بینایی ماشین فرآیندی است که به ماشین درک مشابهی از دیدن و شناسایی اشیاء در جهان واقعی می‌دهد.

برای چشم انسان بسیار ساده است که با دقت بالایی تشخیص دهد که یک حیوان ببر است یا شیر، اما یک سیستم کامپیوتری باید آموزش‌های زیادی ببیند تا بتواند چنین اشیائی را کاملاً تمیز دهد. همچنین بینایی ماشین می‌تواند به این صورت تعریف شود: ساختن مدل‌هایی ریاضی که می‌توانند عملکرد چشم و مغز انسان را تقلید کنند. اساساً این موضوع در مورد آموزش کامپیوتر برای درک و پردازش تصویر و فیلم است.

بینایی ماشین جزوی از حوزه‌های پیشرفته رباتیک، بهداشت و سلامت (X-ray, MRI CT scan و غیره)، پهبادها، خودروهای خود ران، ورزش و غیره است. تقریباً تمام کسب و کارها به بینایی ماشین نیاز دارند تا موفق شوند.

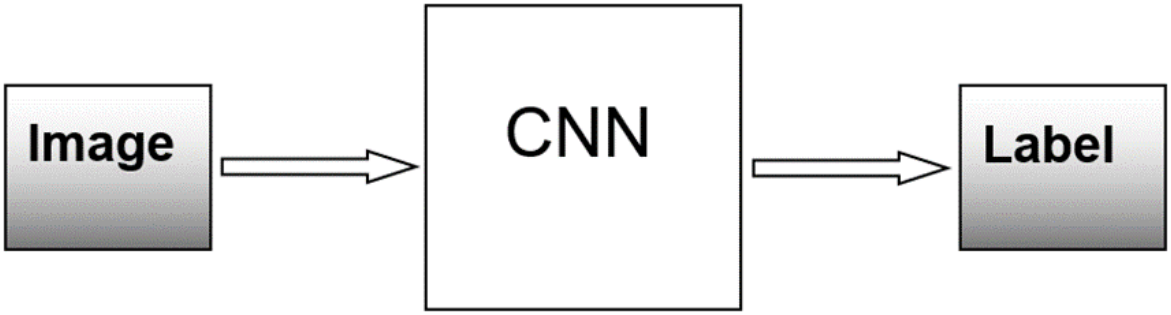
تعداد زیاد داده‌هایی که هر روز تولید می‌شوند را تصور کنید. این داده‌ها به واسطه دوربین‌های مدار بسته، گوشی‌های همراه، فیلم‌های به‌اشتراک گذاشته‌شده در سایت‌هایی مانند یوتیوب و عکس‌های به‌اشتراک گذاشته‌شده در شبکه‌های اجتماعی مانند فیسبوک وجود دارند. تمامی این‌ها حجم زیادی از داده‌های تصویری را تولید می‌کنند. برای پردازش و تحلیل این داده‌ها و هوشمندسازی کامپیوترها از منظر فرآیندی، نیاز به متخصصان سطح بالایی است که در بینایی ماشین تخصص دارند. بینایی ماشین یک حوزه سود آور در یادگیری ماشین است. بخش بعدی به چگونگی رسیدن به بینایی ماشین توسط شبکه‌های عصبی (و مخصوصاً شبکه‌های عصبی پیچشی که برای بینایی ماشین عالی عمل می‌کنند) پرداخته شده است.

شبکه‌های عصبی پیچشی

صحبت از بینایی ماشین درواقع صحبت از شبکه‌های عصبی پیچشی نیز است. این شبکه‌های عصبی دسته‌ای از شبکه‌های عصبی عمیق[15] هستند که اکثراً در حوزه بینایی ماشین و تصویر استفاده می‌شوند. شبکه‌های عصبی پیچشی برای شناسایی تصاویر، خوشه‌بندی آن‌ها از روی شباهت‌هایشان و شناسایی اشیاء در صحنه استفاده

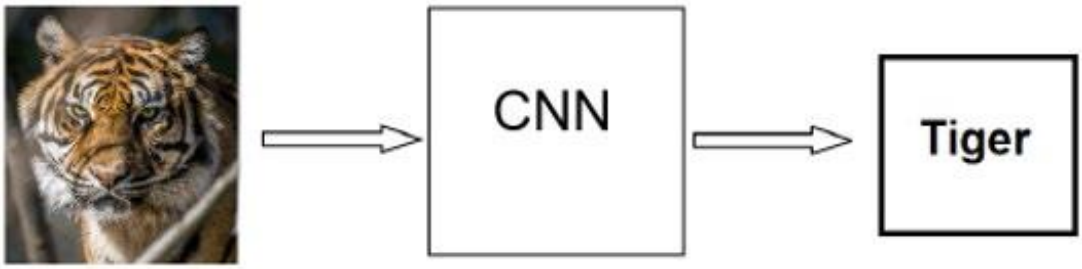
[15] Deep neural network

می‌شوند. شبکه‌های عصبی پیچشی دارای لایه‌های مختلفی هستند که شامل لایه ورودی، لایه خروجی و چند لایه مخفی می‌باشد. این لایه‌های مخفی شبکه عصبی پیچشی شامل لایه‌های کاملاً متصل[16]، لایه‌های پیچشی، یک لایه ReLU[17] به‌عنوان تابع فعال‌سازی، لایه نرمال‌سازی و لایه‌های تجمیع[18] می‌باشد. در یک حالت ساده، شبکه‌های عصبی پیچشی به ما کمک می‌کنند که تصاویر را شناسایی کنیم و به آن‌ها برچسب[19] مناسب بزنیم. به‌عنوان مثال تصویر یک ببر به‌عنوان یک ببر شناسایی شود.



شکل ۷-۱: حالت کلی شبکه عصبی پیچشی

شکل زیر یک مثال از دسته‌بندی ببر توسط شبکه عصبی پیچشی است.



شکل ۷-۲: یک شبکه عصبی که تصویر یک ببر را به دسته "ببر" اختصاص داده‌است

معماری یک شبکه عصبی پیچشی

اجزای اصلی معماری یک شبکه عصبی پیچشی به‌شرح ذیل است:

- تصویر ورودی

[16] Fully connected

[17] توضیح مترجم: این تابع rectified linear unit یا واحد یکسو ساز خطی می‌باشد که به ازای xهای مثبت f(x) = x و در غیر اینصورت f(x) =0 است. در این شبکه عصبی این تابع به‌عنوان تابع فعال‌سازی استفاده می‌شود.

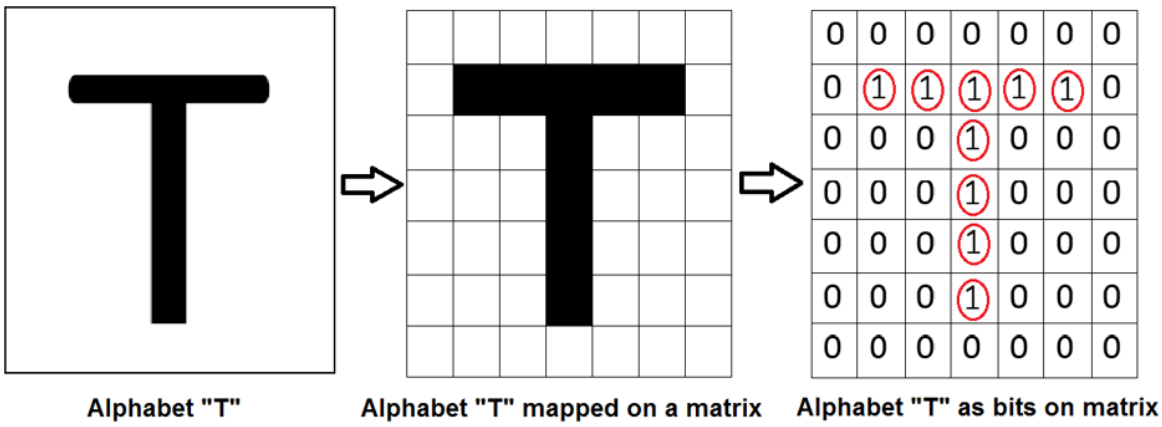
[18] pooling

[19] label

- لایه‌ی پیچشی
- لایه تجمیع
- لایه مسطح‌سازی

تصویر ورودی

یک تصویر ورودی اولین جزء معماری یک شبکه عصبی پیچشی را تشکیل می‌دهد. یک تصویر می‌تواند از هر نوعی باشد، مثلاً تصویری از یک انسان، یک حیوان، منظره، یک تصویر X-ray و غیره. هر تصویر به یک ماتریس ریاضی از صفرها و یک‌ها تبدیل می‌شود. تصویر زیر نشان می‌دهد که چگونه یک کامپیوتر تصویر حرف T را می‌بیند. تمامی بلوک‌هایی که مقدار یک دارند نشان‌دهنده وجود داده هستند و مقدار صفر نشان‌دهنده فضای خالی است.



شکل۷–۳: ماتریس برای حرف "T"

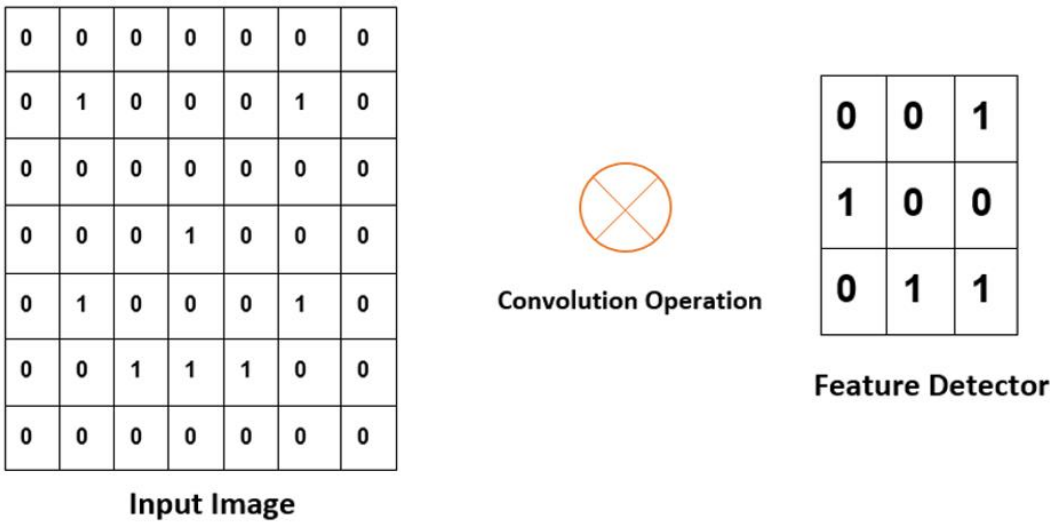
لایه پیچشی

لایه پیچشی جایی است که پردازش تصویر آغاز می‌شود. یک لایه پیچشی شامل دو قسمت می‌شود:

- تشخیص‌دهنده ویژگی یا فیلتر[۲۰]
- نگاشت ویژگی (نقشه ویژگی)[۲۱]

[20] Feature detector or filter
[21] Feature Map

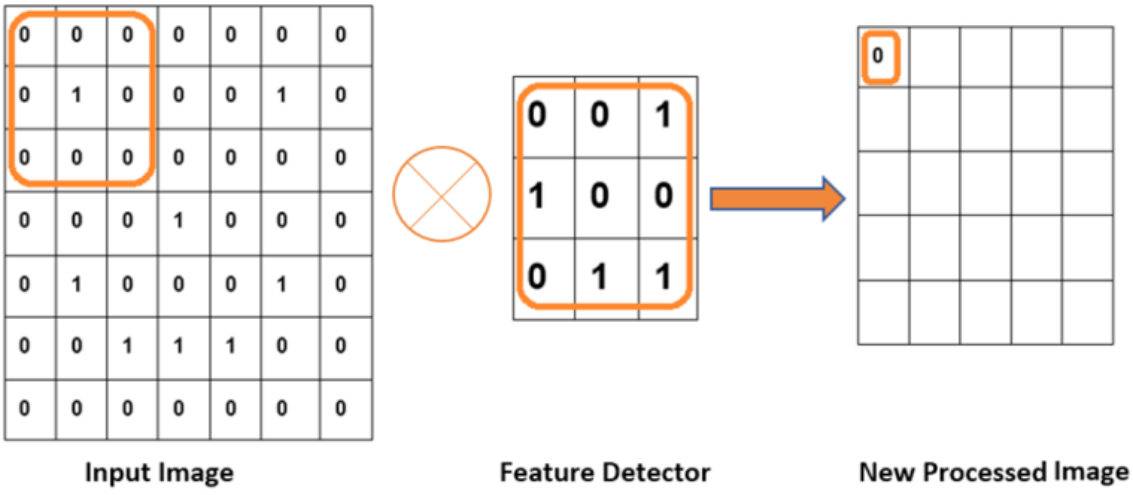
تشخیص‌دهنده ویژگی یا فیلتر: یک ماتریس یا الگو است که روی یک تصویر قرار داده می‌شود تا آن تصویر را به یک نگاشت از آن ویژگی تبدیل کند.



شکل ۷-٤: تشخیص‌دهنده ویژگی

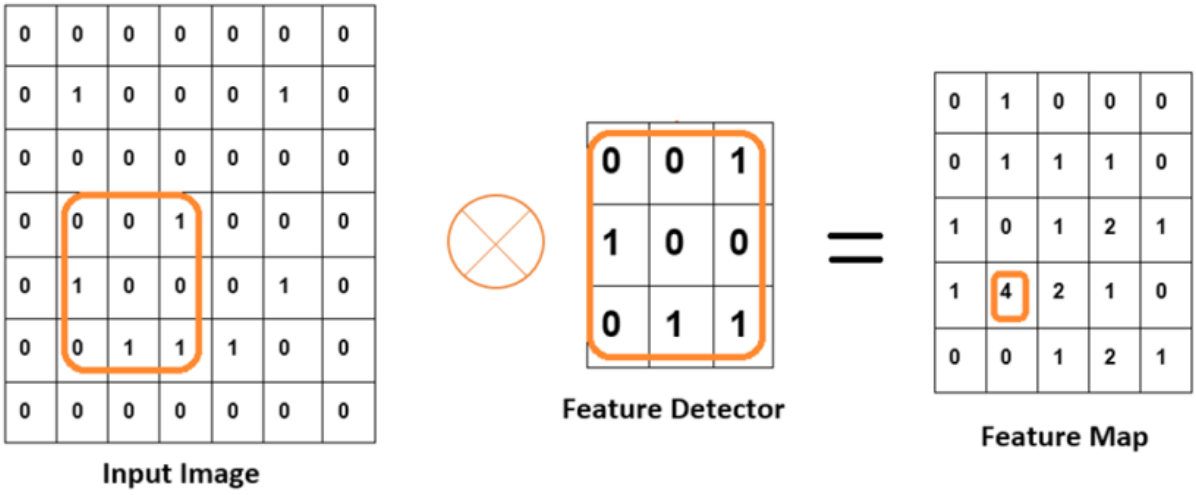
همان‌طور که مشاهده می‌شود این تشخیص‌دهنده ویژگی روی تصویر اصلی قرار داده شده (سوار شده) و محاسبات روی بلوک‌های متناظر انجام شده است. محاسبات به‌صورت مجموع حاصل ضرب اعداد بلوک‌های متناظر انجام می‌شود. این فرآیند برای تمامی سلول‌ها تکرار می‌شود و منجر به یک تصویر پردازش شده می‌شود. به‌عنوان مثال محاسبه برای سلول بالا سمت چپ به این صورت است:

$$(0 \times 0 + 0 \times 0 + 0 \times 1) + (0 \times 1 + 1 \times 0 + 0 \times 0) + (0 \times 0 + 0 \times 1 + 0 \times 1) = 0$$



شکل ۷-۵: تشخیص‌دهنده ویژگی که روی یک تصویر پیاده‌سازی شده

نگاشت ویژگی: نگاشت ویژگی یک تصویر کاهش‌یافته[۲۲] است که با پیچش یک تصویر و تشخیص‌دهنده ویژگی تولید شده است. باید تشخیص‌دهنده ویژگی را روی تمامی محل‌های ممکن از تصویر اصلی قرار دهیم و یک تصویر کوچکتر را از آن بدست بیاوریم. این تصویر بدست آمده همان نگاشت ویژگی از تصویر ورودی است.

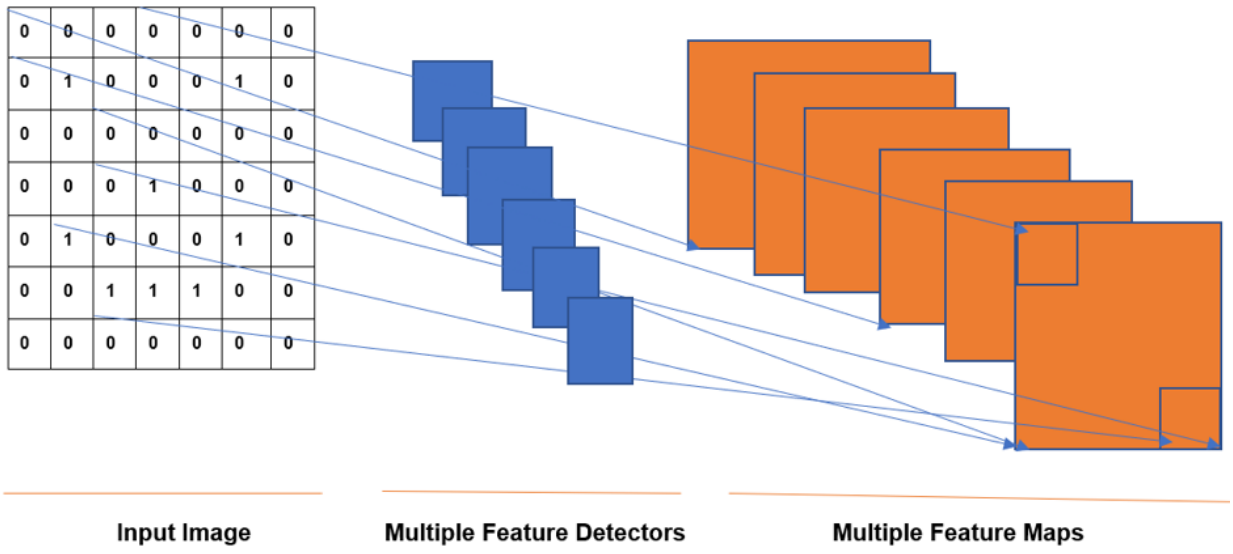


شکل ۷-٦: نگاشت ویژگی

نکته: در اینجا تشخیص‌دهنده ویژگی، فیلتر است و نگاشت ویژگی، تصویر کاهش‌یافته است. در هنگام کاهش تصویر (تولید نگاشت ویژگی) مقداری از اطلاعات از دست می‌رود.

در یک شبکه عصبی واقعی، تعدادی تشخیص‌دهنده ویژگی استفاده می‌شوند تا تعدادی نگاشت ویژگی تولید کنند که در تصویر زیر نشان داده شده است.

[22] Reduced image



شکل۷–۷: چند تشخیص‌دهنده ویژگی و نگاشت ویژگی

لایه تجمیع

لایه تجمیع به ما کمک می‌کند که اطلاعاتی از تصویر که اهمیت کمتری دارند را درنظر نگیریم و درعین حال با نگهداشتن ویژگی‌های مهم، تصویر را بیشتر کاهش بدهیم. به سه تصویر زیر که در مجموع شامل چهار گربه می‌شوند توجه کنید[۲۳].

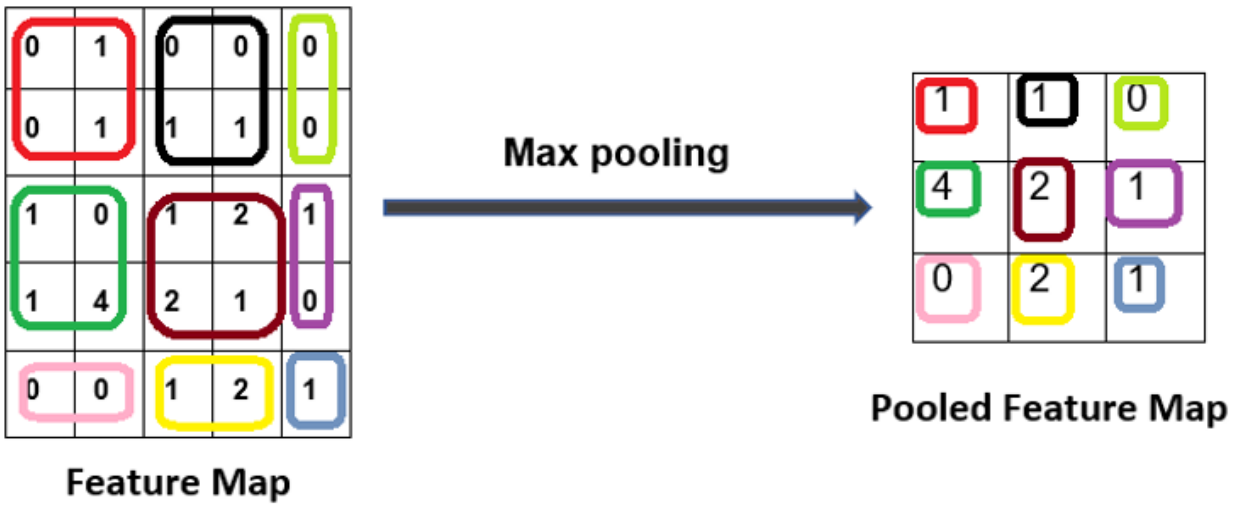
شکل ۷–۸: مثالی از تصاویر گربه

برای این که مشخص شود در یک تصویر گربه وجود دارد یا خیر، شبکه عصبی، تصویر را تحلیل می‌کند. شبکه عصبی ممکن است به شکل گوش، چشم و غیره توجه کند. درعین حال تصویر شامل ویژگی‌های بسیار زیاد دیگری است که ربطی به گربه ندارند. درخت و برگ‌ها در دو تصویر اول، برای شناسایی گربه بلا استفاده هستند.

[23] مترجم: متن کتاب گفته در این سه تصویر چهار گربه وجود داره. من هر چی گشتم گربه ی چهارم رو پیدا نکردم! اما خدا رو چه دیدی، شاید ☺یکیشون حاملس

مکانیزم تجمیع به الگوریتم کمک می‌کند که متوجه شود که کدام قسمت‌های تصویر مرتبط و کدام قسمت‌ها نامرتبط هستند.

نگاشت ویژگی که از لایه پیچشی بدست آمده بود به لایه تجمیع وارد می‌شود تا درحین نگهداشتن قسمت‌های مرتبط از تصویر، آن را بیشتر کاهش دهد. لایه تجمیع شامل توابعی مانند تجمیع حداکثری، تجمیع حداقلی[۲۴] و تجمیع متوسط[۲۵] است. کاری که در این قسمت انجام می‌شود این است که یک اندازه برای ماتریس مشخص می‌شود، مثلاً 2 × 2 و تمام نگاشت ویژگی اسکن می‌شود و بزرگترین عددی که در آن ماتریس 2 × 2 قرار گرفته بود انتخاب می‌شود. تصویر زیر ایده‌ی بهتری در مورد نحوه‌ی کار تجمیع حداکثری می‌دهد. بر اساس رنگ‌ها، بزرگترین عددی که در هرکدام از جعبه‌های رنگی در نگاشت ویژگی قرار دارد انتخاب می‌شود و در نگاشت ویژگی تجمیع شده قرار می‌گیرد.



شکل ۷-۹: تجمیع

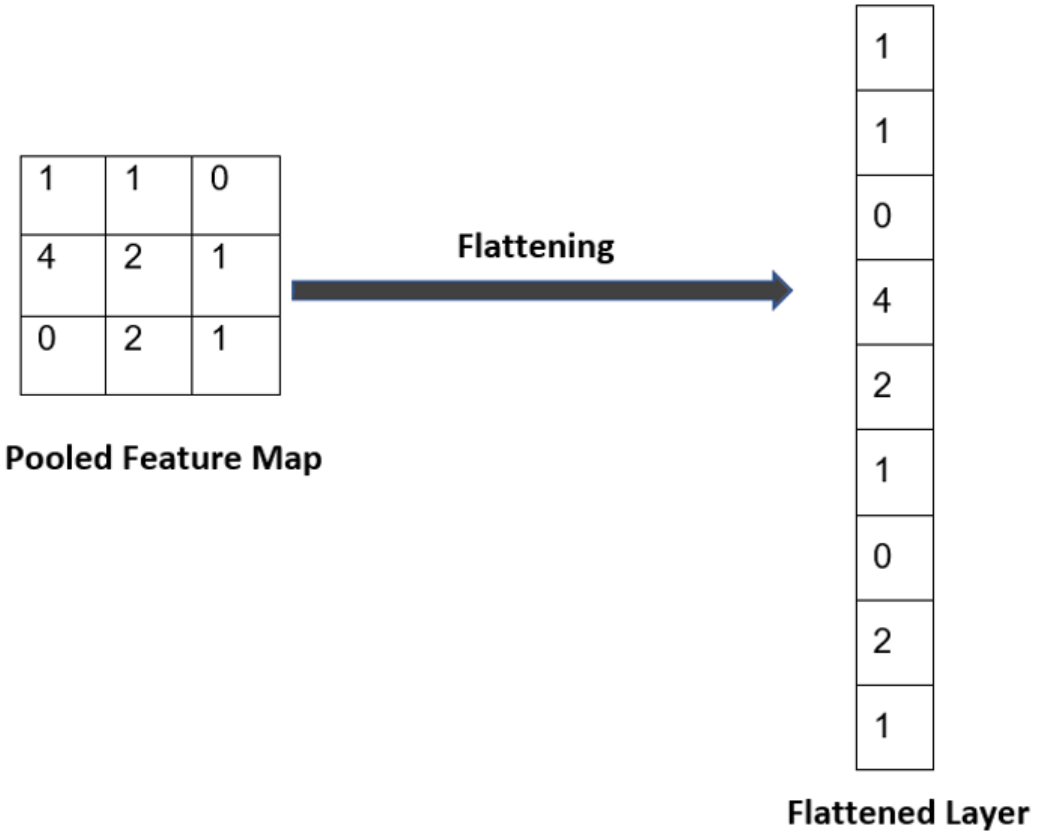
به‌عنوان مثال جعبه‌ای که عدد ٤ را دارد درنظر بگیرید. فرض کنید عدد ٤ نشان‌دهنده گوش گربه باشد و فضای خالی اطراف گوش اعداد ۰ و ۱ باشند. پس ما اعداد ۰ و ۱ از آن جعبه را درنظر نمی‌گیریم و فقط عدد ٤ را انتخاب می‌کنیم. یک نمونه از کدی که برای اضافه کردن لایه تجمیع نیاز داریم در زیر آورده شده‌است. در این‌جا Maxpool12D استفاده شده‌است تا کمک کند که مهم‌ترین ویژگی‌ها شناسایی شوند.

```
classifier.add(MaxPool2D(2,2))
```

[24] Min pooling
[25] Average pooling

مسطح‌سازی

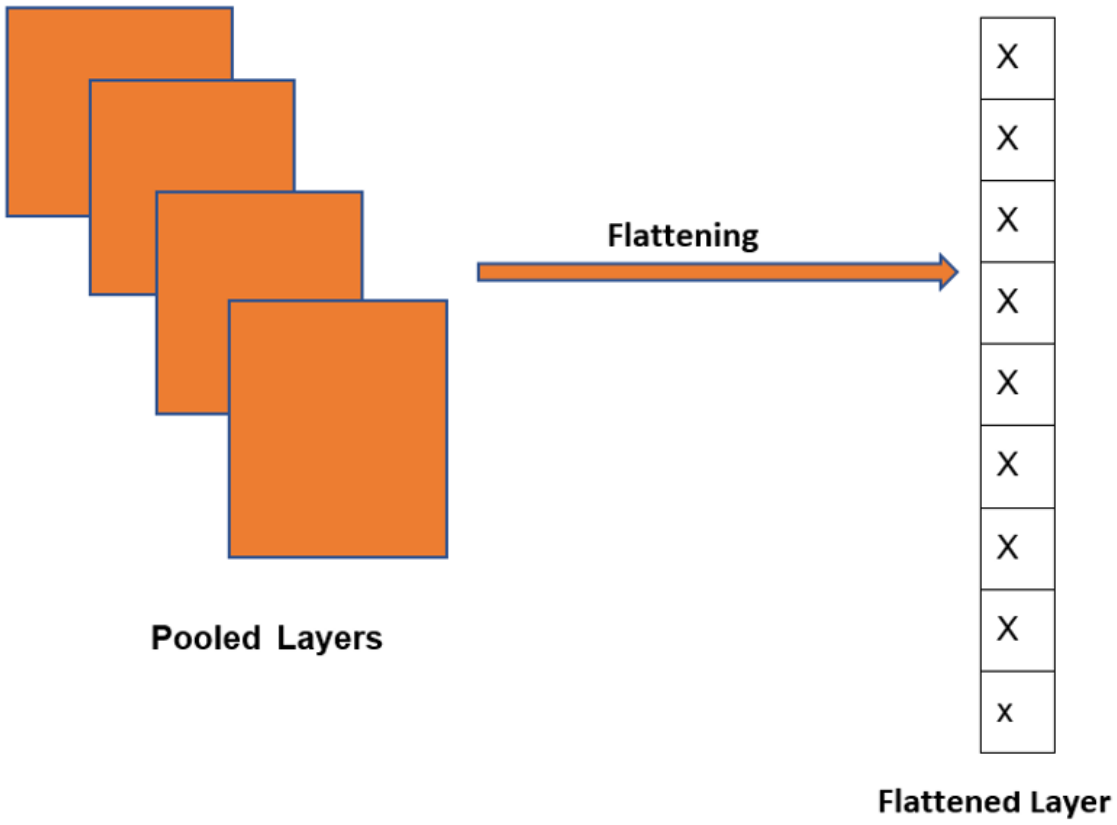
مسطح‌سازی در قسمتی از شبکه عصبی پیچشی استفاده می‌شود که تصویر، برای استفاده به‌عنوان ورودی یک شبکه عصبی مصنوعی، آماده می‌شود. همان‌طور که از اسم مشخص است، در این قسمت تصویر تجمیع شده، مسطح می‌شود و به یک ستون تبدیل می‌گردد. هر سطر به یک ستون تبدیل می‌شود و روی هم سوار می‌شوند. در اینجا یک ماتریس 3×3 به یک ماتریس $n \times 1$ [۲۶] تبدیل شده که در این مورد n برابر ۹ است.



شکل ۷-۱۰: مسطح‌سازی

ما به‌طور هم‌زمان تعدادی نگاشت ویژگی تجمیع شده داریم که همه‌ی آن‌ها را به‌صورت تک ستون مسطح کرده‌ایم. این تک ستون به‌عنوان ورودی یک شبکه عصبی مصنوعی استفاده می‌شود. شکل زیر تعدادی لایه تجمیع شده که به‌صورت تک ستون در آمده‌اند را نشان می‌دهد.

[۲۶] نظر مترجم: نمیدونم اشتباه تایپی بوده توی متن کتاب یا چه موردی بوده اما ماتریس ۳*۳ به یک ماتریس n*1 تبدیل شده اما در متن نوشته 1*n

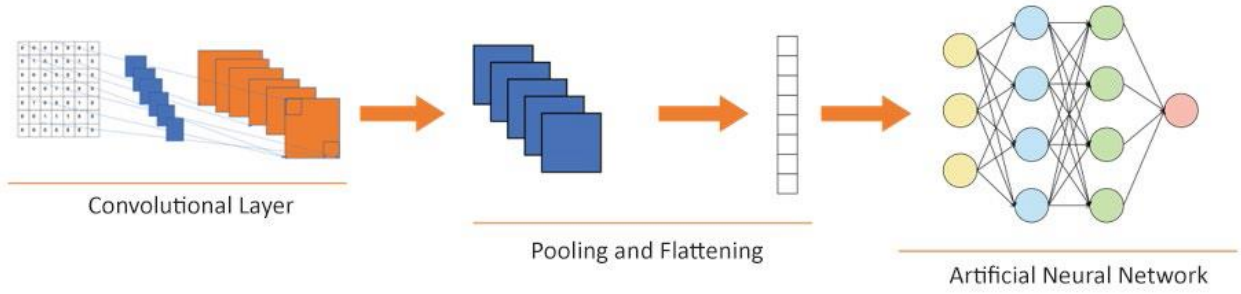


شکل ۷-۱۱ تجمیع و مسطح‌سازی

نمونه کدی که باید برای اضافه کردن لایه مسطح‌سازی استفاده شود در زیر آمده. در اینجا برای مسطح‌سازی شبکه عصبی پیچشی از Flatten استفاده شده است.

```
classifier.add(Flatten())
```

حال نگاهی به ساختار کلی شبکه عصبی پیچشی می‌اندازیم.



شکل ۷-۱۲: ساختار شبکه عصبی پیچشی

نمونه کدی که باید برای اضافه کردن لایه اول به شبکه عصبی پیچشی استفاده کنیم در زیر آورده شده‌است.

```
classifier.add(Conv2D(32,3,3,input_shape=(64,64,3),activation='relu'))
```

32,3,3 نشان‌دهنده این است که ۳۲ عدد تشخیص‌دهنده ویژگی وجود دارد که اندازه هرکدام 3×3 است. بهتر است همیشه با ۳۲ آغاز شود، می‌توان ٦٤ یا ۱۲۸ را بعداً امتحان کرد.

Input_shape: از آن‌جایی که تصاویر دارای اندازه‌های متفاوتی هستند، input_image تمامی تصاویر را به یک شکل و اندازه تبدیل می‌کند. (64 , 64) ابعاد و اندازه‌ی تصویر تغییر شکل یافته است. می‌توان این اعداد را ۱۲۸ یا ۲٥٦ درنظر گرفت، اما اگر با CUP یک لپتاپ کار می‌کنید توصیه می‌شود که همان ٦٤ استفاده شود. آخرین آرگومان (۳) استفاده شده است زیرا تصویر، رنگی (RGB) است. اگر تصویر سیاه و سفید باشد این عدد می‌تواند ۱ درنظر گرفته شود. در نهایت تابع فعال‌سازی استفاده شده ReLU است.

نکته: ما از Keras با TensorFlow به‌عنوان عقبه[۲۷] استفاده می‌کنیم. اگر عقبه Theano باشد، آنگاه input_Image به‌صورت (3,64,64) وارد می‌شد.

قدم نهایی، برازش داده‌هایی است که ایجاد شده‌اند. این کار در کد زیر نشان داده شده است.

```
classifier.fit_generator(training_set,
steps_per_epoch = 5000,
epochs = 25,
validation_data = test_set,
validation_steps = 1000)
```

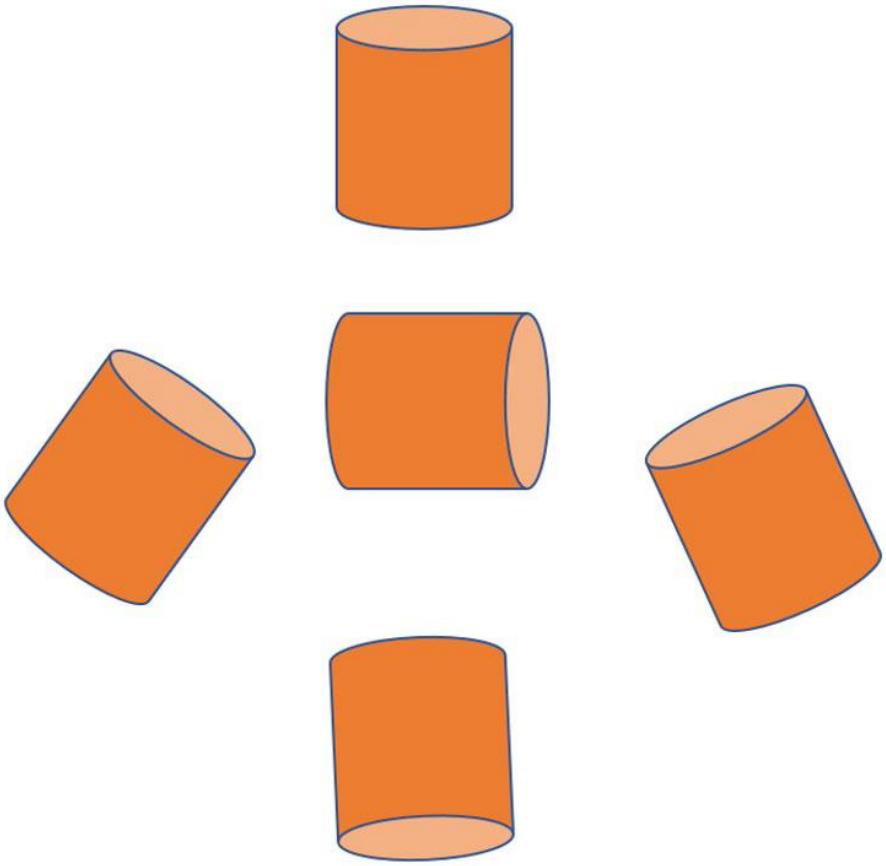
نکته: steps_per_epoch تعداد تصاویر آموزش می‌باشد. validation_steps تعداد تصاویر تست است.

افزایش تصاویر

افزایش تصاویر یا داده‌ها، دسته‌های[۲۸] زیادی از تصاویر تولید می‌کند. سپس تغییرات تصادفی روی تصاویر تصادفی موجود در هر دسته اعمال می‌کند. این تغییرات می‌تواند چرخش، جابجایی، پشت و رو کردن تصاویر و غیره باشد. با انجام این تبدیلات تصاویر متنوع‌تر و بیشتری درون هر دسته خواهیم داشت.

[27] Backend(مترجم: توصیه می شود از کلمه بک‌اند استفاده شود)
[28] batch

یک استوانه می‌تواند در زوایای مختلفی چرخانده و دیده شود. در تصویر زیر، یک استوانه در ۵ زاویه مختلف نشان داده شده‌است. به این صورت به‌طور کارایی ۵ تصویر متفاوت از ۱ تصویر ساخته‌ایم.



شکل ۷-۱۳ افزایش تصاویر یک استوانه

نمونه کدی که برای افزایش تصاویر باید استفاده شود در زیر آمده است. در اینجا کلاس ImageDataGenerator برای پردازش استفاده شده است. shear_range، zoom_range و horizontal_flip برای تغییرات روی تصاویر استفاده می‌شوند.

```
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255.0,
                                   shear_range = 0.3,
                                   zoom_range = 0.3,
                                   horizontal_flip = False)
test_datagen = ImageDataGenerator(rescale = 1./255.0)
```

مزایای افزایش تصاویر

افزایش تصاویر یک بخش مهم در پردازش تصاویر است:

- **کاهش بیش‌برازش**[۲۹]: داشتن نسخه‌های مختلف از یک تصویر در زوایای مختلف، به کاهش بیش‌برازش کمک می‌کند.
- **افزایش تعداد تصاویر**: یک تصویر، مانند چند تصویر عمل می‌کند. بنابراین مجموعه داده، دارای تعداد کمتری عکس می‌باشد و با افزایش تصاویر، هر عکس می‌تواند به چند عکس تبدیل شود. افزایش تصاویر باعث افزایش تعداد عکس‌ها می‌شود و الگوریتم هر عکس را عکس متفاوتی می‌بیند.
- **سادگی پیش‌بینی تصاویر جدید**: تصور کنید که هر عکس از توپ فوتبال از زوایای مختلف دیده شود و هر زاویه یک عکس مجزا درنظر گرفته شود. این بدین معناست که الگوریتم در پیش‌بینی عکس‌های جدید بسیار دقیق‌تر خواهد بود:

شکل۷-۱٤ افزایش تصویر برای یک عکس از توپ فوتبال

حال که مفاهیم و تئوری مربوط به شبکه‌های عصبی پیچشی را یاد گرفتیم روی تعدادی مثال عملی تمرکز می‌کنیم.

[29] overfitting

با یک مثال شروع می‌کنیم که در آن یک شبکه عصبی پیچشی ساده می‌سازیم. در تمرینات بعدی شبکه عصبی پیچشی خود را بهبود خواهیم داد که با ترکیبی از موارد زیر امکان پذیر است:

- اضافه کردن لایه‌های شبکه عصبی پیچشی
- اضافه کردن لایه‌های شبکه عصبی مصنوعی
- تغییر تابع بهینه‌ساز[30]
- تغییر تابع فعال‌سازی

حال وارد ساختن اولین شبکه عصبی پیچشی می‌شویم که قادر است تصاویر خودروها و گل‌ها را در کلاس مربوط به هرکدام دسته‌بندی کند.

تمرین ۷-۰۱: ساختن یک شبکه عصبی پیچشی و شناسایی تصاویر خودروها و گل‌ها

برای این تمرین تصاویری از خودروها و گل‌ها در اختیار داریم. این مجموعه داده به دو دسته آموزش و تست تقسیم‌بندی شده‌است و باید یک شبکه عصبی مصنوعی به‌گونه‌ای بسازیم که تشخیص دهد آیا در یک تصویر خودرو وجود دارد یا گل.

نکته: تمامی تمرین‌ها و فعالیت‌ها در این فصل در jupyter notebook انجام می‌شوند. لطفاً مخزن GitHub کتاب، به همراه همه‌ی تمپلیت‌های آماده‌شده را از <https://packt.live/39tID2C> دانلود کنید.

قبل از شروع مطمئن شوید که مجموعه داده‌ی تصویری این کتاب را از GitHub دانلود و در سیستم خود ذخیره کرده باشید. شما به یک پوشه training_set برای آموزش مدل و یک پوشه test_set برای تست مدل نیاز دارید. هرکدام از این پوشه‌ها شامل یک پوشه cars برای تصاویر خودروها و یک پوشه flowers برای تصاویر گل‌ها می‌شوند.

مراحل برای تکمیل این تمرین به‌شرح ذیل است:

[30] Optimizer function

۱. کتابخانه numpy و کتابخانه‌ها و کلاس‌های ضروری برای Keras را فراخوانی[۳۱] کنید:

```
# Import the Libraries
from keras.models import Sequential from keras.layers import Conv2D,
MaxPool2D, Flatten, Dense
import numpy as np
from tensorflow import random
```

۲. یک هسته[۳۲] مشخص کنید و مدل را با استفاده از کلاس‌های ترتیبی[۳۳] شروع کنید:

```
# Initiate the classifier
seed = 1
np.random.seed(seed)
random.set_seed(seed)
classifier = Sequential()
```

۳. اولین لایه شبکه عصبی پیچشی را اضافه کنید. ابعاد ورودی را (64,64,3) و تابع فعال‌سازی را ReLU درنظر بگیرید:

```
classifier.add(Conv2D(32,3,3,input_shape=(64,64,3),
activation='relu'))
```

32,3,3 نشان می‌دهد که ۳۲ عدد تشخیص‌دهنده ویژگی با اندازه 3×3 وجود دارد.

٤. حال لایه تجمیع را با اندازه تصویر 2×2 اضافه کنید:

```
classifier.add(MaxPool2D(2,2))
```

۵. خروجی لایه تجمیع را با اضافه کردن یک لایه مسطح‌سازی به شبکه عصبی پیچشی، مسطح کنید:

```
classifier.add(Flatten())
```

٦. اولین لایه متراکم[۳٤] شبکه عصبی مصنوعی را اضافه کنید. در اینجا ۱۲۸ تعداد نورون‌های خروجی این لایه است. این عدد برای شروع خوب است. تابع فعال‌سازی نیز relu است:

```
classifier.add(Dense(128, activation='relu'))
```

---

[31] import
[32] seed
[33] Sequential class
[34] Dence <مترجم: این مفهوم در فصل‌های قبل توضیح داده شده است>

۷. لایه خروجی شبکه عصبی مصنوعی را اضافه کنید. چون این یک مسئله دسته‌بندی است، اندازه لایه خروجی ۱ است. همچنین تابع فعال‌سازی sigmoid می‌باشد:

```
classifier.add(Dense(1, activation='sigmoid'))
```

۸. شبکه را با یک بهینه‌ساز adam کامپایل کنید و دقت مدل را در حین فرآیند آموزش محاسبه نمایید:

```
#Compile the network
classifier.compile(optimizer='adam',loss='binary_crossentropy',
metrics=['accuracy'])
```

۹. یک تولیدکننده داده[۳۵] برای آموزش و تست ایجاد کنید. اندازه داده‌های آموزش و تست را به 1/255 کاهش دهید تا همه‌ی مقادیر بین ۰ و ۱ شوند. این پارامترها را تنها برای تولیدکننده داده‌های آموزش (مترجم: و نه داده‌های تست)، به‌صورت روبرو تعیین کنید: shear_range=0.2, zoom_range=0.2, and horizontal_flip=True

```
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)
test_datagen = ImageDataGenerator(rescale = 1./255)
```

۱۰. یک مجموعه آموزش از پوشه آموزش ایجاد کنید. "…/dataset/training_set" جایی است که مجموعه آموزش قرار داده شده‌است. در شبکه عصبی پیچشی ما اندازه تصویر 64×64 است. بنابراین باید تصاویری با همین اندازه به شبکه داده شود. batch_size تعداد تصاویر موجود در یک دسته[۳۶] را مشخص می‌کند که در اینجا ۳۲ است. Class_mode به‌صورت binary تعیین شده است زیرا ما با یک دسته‌بند دوتایی کار می‌کنیم:

```
training_set = train_datagen.flow_from_directory('../dataset/training_
set',
target_size = (64, 64),
batch_size = 32,
class_mode = 'binary')
```

[۳۵] منظور ImageDataGenarator است که در بخش افزایش تصاویر گفته شد.

[36] batch

۱۱.مرحله ۱۰ را برای مجموعه داده تست تکرار کنید. دقت کنید که باید پوشه محل قرارگیری داده‌های تست مشخص شود که به‌صورت '../dataset/test_set' است:

```
test_set = test_datagen.flow_from_directory('../dataset/test_set',
target_size = (64, 64),
batch_size = 32,
class_mode = 'binary')
```

۱۲.درنهایت با این داده‌ها مدل را برازش کنید. مقدار steps_per_epoch را ۱۰۰۰۰ و مقدار validation_steps را ۲۵۰۰ قرار دهید. اجرای این مرحله ممکن است مقداری زمان‌گیر باشد:

```
classifier.fit_generator(training_set,
steps_per_epoch = 10000,
epochs = 2,
validation_data = test_set,
validation_steps = 2500,
shuffle=False)
```

کد معرفی شده این خروجی را خواهد داد:

```
Epoch 1/2
10000/10000 [==============================] - 1994s 199ms/step -
loss: 0.2474 - accuracy: 0.8957 - val_loss: 1.1562 - val_accuracy:
0.8400
Epoch 2/2
10000/10000 [==============================] - 1695s 169ms/step -
loss: 0.0867 - accuracy: 0.9689 - val_loss: 1.4379 - val_accuracy:
0.8422
```

دقت در مجموعه ارزیابی 84.22% است.

نکته: برای گرفتن نتایج دقیق‌تر، سعی کنید که تعداد اپوک‌ها[۳۷] را تا حدود ۲۵ افزایش دهید. این باعث می‌شود که زمان لازم برای پردازش داده‌ها افزایش یابد. البته این زمان بستگی به پیکربندی سیستم شما هم دارد.

در اینجا این تمرین روی پردازش تصویر و شناسایی محتوای تصویر به اتمام رسید. نکته مهم این است که این کد، یک کد مقاوم[۳۸] برای هر دسته‌بندی دوتایی در بینایی ماشین است. این بدین معناست که کد، یکسان باقی می‌ماند، حتی اگر داده‌های تصاویر تغییر کنند. در فعالیت بعدی تعدادی از پارامترهای مدل را تغییر می‌دهیم و کارایی مدل را مورد ارزیابی قرار می‌دهیم.

[37] epochs
[38] robust code

فعالیت ۷-۰۱: اصلاح مدل با چندین لایه و استفاده از softmax

از آن‌جایی که یک مدل شبکه عصبی پیچشی را با موفقیت اجرا کردیم، مرحله بعدی این است که سعی کنیم کارایی الگوریتم را بهبود ببخشیم. راه‌های زیادی برای بهبود کارایی وجود دارد، اما یکی از سرراست‌ترین آن‌ها افزودن چندین لایه شبکه عصبی مصنوعی به مدل است که در این فعالیت خواهیم آموخت. همچنین تابع فعال‌سازی را از sigmoid به softmax تغییر خواهیم داد. با انجام این کارها می‌توان نتیجه را با مدل قبلی مقایسه کرد. برای انجام این کارها این مراحل را طی کنید:

۱. برای ساختن یک شبکه عصبی پیچشی کتابخانه را فرا بخوانید، یک هسته مشخص کنید و یک کلاس ترتیبی ایجاد نمایید. Conv2D، MaxPool2D، Flatten و Dense را فرا بخوانید. Conve2D برای ساختن لایه پیچشی استفاده می‌شود. از آن‌جایی که تصاویر ما دو بعدی هستند اینجا هم از دو بعدی استفاده کرده‌ایم. به‌طور مشابه MaxPool2D برای تجمیع حداکثری، Flatten برای مسطح‌سازی شبکه عصبی پیچشی و Dense برای اضافه کردن یک شبکه عصبی پیچشی کاملاً متصل به یک شبکه عصبی مصنوعی استفاده شده‌اند.
۲. با استفاده از کتابخانه‌های معرفی شده، معماری شبکه عصبی پیچشی را آغاز کنید. پس از اضافه کردن اولین لایه، دو لایه مازاد نیز به شبکه عصبی پیچشی اضافه کنید.
۳. یک لایه تجمیع و مسطح‌سازی اضافه کنید که به‌عنوان ورودی شبکه عصبی مصنوعی عمل می‌کند.
٤. یک شبکه عصبی مصنوعی کاملاً متصل بسازید که ورودی آن، خروجی شبکه عصبی پیچشی باشد. بعد از اضافه کردن اولین لایه شبکه عصبی مصنوعی، سه لایه مازاد نیز اضافه کنید. برای لایه خروجی شبکه عصبی مصنوعی از تابع فعال‌سازی softmax استفاده کنید و مدل را اجرا کنید.
۵. برای پردازش و تغییر داده‌ها از افزایش تصویر استفاده کنید. کلاس ImageDataGeneration برای پردازش استفاده می‌شود. shear_range، zoom_range و horizontal_flip برای تغییر تصاویر استفاده می‌شوند.
٦. مجموعه داده آموزش و تست را ایجاد کنید.
۷. در انتها مدل را برازش کنید.

پس از اجرای این مراحل باید این خروجی را دریافت کنید:

```
Epoch 1/2
10000/10000 [==============================] - 2452s 245ms/step -
loss: 8.1783 - accuracy: 0.4667 - val_loss: 11.4999 - val_accuracy:
0.4695
Epoch 2/2
10000/10000 [==============================] - 2496s 250ms/step -
loss: 8.1726 - accuracy: 0.4671 - val_loss: 10.5416 - val_accuracy:
0.4691
```

نکته: پاسخ این فعالیت در صفحه ۳۹۳ یافت می‌شود.

در این فعالیت، ما شبکه عصبی پیچشی خود را اصلاح کردیم و سعی کردیم دقت دسته‌بند را افزایش دهیم. ما لایه‌های پیچشی مازاد و لایه‌های مازد کاملاً متصل شبکه عصبی مصنوعی را اضافه کردیم و تابع فعال‌سازی در لایه خروجی را تغییر دادیم. با انجام این کارها دقت ما کاهش یافت. در تمرین بعد تابع فعال‌سازی را دوباره به sigmoid تغییر می‌دهیم و سپس کارایی را بر اساس دقت بررسی می‌کنیم.

تمرین ۷-۰۲: بهبود مدل با بازگشت به تابع فعال‌سازی sigmoid

در این تمرین از تابع فعال‌سازی softmax به sigmoid باز می‌گردیم. با این کار می‌توانیم دقت این مدل را با دقت مدل قبلی مقایسه کنیم. برای انجام این تمرین مراحل زیر را دنبال کنید:

۱. کتابخانه numpy و کتابخانه‌ها و کلاس‌های ضروری برای Keras را فرا بخوانید:

```
# Import the Libraries
from keras.models import Sequential
from keras.layers import Conv2D, MaxPool2D, Flatten, Dense
import numpy as np
from tensorflow import random
```

۲. یک هسته مشخص کنید و مدل را با استفاده از کلاس‌های ترتیبی شروع کنید:

```
# Initiate the classifier
seed = 43
np.random.seed(seed)
random.set_seed(seed)
classifier = Sequential()
```

۳. اولین لایه شبکه عصبی پیچشی را اضافه کنید. ابعاد ورودی را (64,64,3) و تابع فعال‌سازی را ReLU درنظر بگیرید. سپس ۳۲ عدد تشخیص‌دهنده ویژگی با اندازه 3×3 اضافه کنید. دو لایه پیچشی مازاد با ۳۲ عدد تشخیص‌دهنده ویژگی با اندازه 3×3 و تابع فعال‌سازی ReLU اضافه کنید:

```
classifier.add(Conv2D(32,3,3,input_shape=(64,64,3),activation='relu'))
classifier.add(Conv2D(32, (3, 3), activation = 'relu'))
classifier.add(Conv2D(32, (3, 3), activation = 'relu'))
```

٤. حال لایه تجمیع را با اندازه تصویر 2×2 اضافه کنید:

```
classifier.add(MaxPool2D(2,2))
```

۵. یک Conv2D دیگر با پارامترهایی مشابه مرحله ۳، و برای تکمیل آن، یک لایه تجمیع دیگر با پارامترهای مشابه مرحله ٤ اضافه کنید:

```
classifier.add(Conv2D(32, (3, 3), activation = 'relu'))
classifier.add(MaxPool2D(pool_size = (2, 2)))
```

٦. خروجی لایه تجمیع را با اضافه کردن یک لایه مسطح‌سازی به شبکه عصبی پیچشی، مسطح کنید:

```
classifier.add(Flatten())
```

۷. اولین لایه متراکم شبکه عصبی مصنوعی را اضافه کنید. در اینجا ۱۲۸ تعداد نورون‌های خروجی این لایه است. این عدد برای شروع خوب است. تابع فعال‌سازی نیز relu است. سه لایه مازاد دیگر با پارامترهای مشابه اضافه کنید:

```
classifier.add(Dense(128,activation='relu'))
classifier.add(Dense(128,activation='relu'))
classifier.add(Dense(128,activation='relu'))
classifier.add(Dense(128,activation='relu'))
```

۸. لایه خروجی شبکه عصبی مصنوعی را اضافه کنید. چون این یک مسئله دسته‌بندی است، اندازه لایه خروجی ۱ است. همچنین تابع فعال‌سازی sigmoid می‌باشد:

```
classifier.add(Dense(1,activation='sigmoid'))
```

۹. شبکه را با یک بهینه‌ساز adam کامپایل کنید و دقت مدل را در حین فرآیند آموزش محاسبه نمایید:

```
classifier.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
```

۱۰. یک تولیدکننده داده برای آموزش و تست ایجاد کنید. اندازه داده‌های آموزش و تست را به 1/255 کاهش دهید تا همه‌ی مقادیر بین ۰ و ۱ شوند. این پارامترها را تنها برای تولیدکننده داده‌های آموزش، به‌صورت روبرو تعیین کنید: shear_range=0.2, zoom_range=0.2, and horizontal_flip=True

```
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)
 test_datagen = ImageDataGenerator(rescale = 1./255)
```

۱۱. یک مجموعه آموزش از پوشه آموزش ایجاد کنید. "dataset/training_set/..." جایی است که مجموعه آموزش قرار داده شده‌است. در شبکه عصبی پیچشی ما، اندازه تصویر 64×64 است. بنابراین باید تصاویری با همین اندازه به شبکه داده شود. batch_size تعداد تصاویر موجود در یک دسته را مشخص می‌کند که در اینجا ۳۲ است. Class_mode به‌صورت binary تعیین شده‌است زیرا ما با یک دسته‌بند دوتایی کار می‌کنیم:

```
training_set = train_datagen.flow_from_directory('../dataset/training_
set',
target_size = (64, 64),
batch_size = 32,
class_mode = 'binary')
```

۱۲. مرحله ۱۱ را برای مجموعه داده تست تکرار کنید. دقت کنید که باید پوشه محل قرارگیری داده‌های تست مشخص شود که به‌صورت '../dataset/test_set' است:

```
test_set = test_datagen.flow_from_directory('../dataset/test_set',
target_size = (64, 64),
batch_size = 32,
class_mode = 'binary')
```

۱۳. درنهایت با این داده‌ها مدل را برازش کنید. مقدار steps_per_epoch را ۱۰۰۰۰ و مقدار validation_steps را ۲۵۰۰ قرار دهید. اجرای این مرحله ممکن است مقداری زمان‌گیر باشد:

```
classifier.fit_generator(training_set,
steps_per_epoch = 10000,
epochs = 2,
validation_data = test_set,
validation_steps = 2500,
shuffle=False)
```

کد معرفی شده، این خروجی را خواهد داد:

```
Epoch 1/2
10000/10000 [==============================] - 2241s 224ms/step -
loss: 0.2339 - accuracy: 0.9005 - val_loss: 0.8059 - val_accuracy:
0.8737
Epoch 2/2
10000/10000 [==============================] - 2394s 239ms/step -
loss: 0.0810 - accuracy: 0.9699 - val_loss: 0.6783 - val_accuracy:
0.8675
```

دقت در مجموعه ارزیابی 86.75% است که به‌طور مشخص از دقت مدلی که در تمرین قبل ساخته بودیم بهتر است. این موضوع، اهمیت تابع فعال‌سازی را نشان می‌دهد. تنها با تغییر تابع فعال‌سازی از SoftMax به sigmoid دقت از 46.91% به 86.75% افزایش یافت. دلیل این افزایش قابل توجه در دقت، این است که تابع فعال‌سازی SoftMax توزیع احتمال را به‌عنوان خروجی می‌دهد و بنابراین باید جمع تمامی نورون‌های خروجی برابر ۱ شود. از آن‌جایی که در اینجا تنها یک نورون خروجی داریم (مترجم: و این نورون همواره مقدار ۱ را می‌دهد)، درواقع دقت مدل با تابع SoftMax برابر درصد تصاویری از مجموعه داده تست است که برچسب ۱ دارند[۳۹]. در تمرین بعدی بهینه‌سازهای مختلف را امتحان می‌کنیم و مشاهده می‌کنیم که چگونه بر کارایی مدل تأثیر می‌گذارند.

نکته: برای مسائل دسته‌بندی (مانند مسئله ما که دسته‌بندی خودرو و گل است)، همواره بهتر است که از تابع فعال‌سازی sigmoid برای خروجی استفاده شود.

تمرین ۷-۰۳: تغییر بهینه‌ساز از Adam به SGD

در این تمرین دوباره مدل را با تغییر بهینه‌ساز به SGD بهبود می‌دهیم. با این کار می‌توانیم دقت را با مدل‌های قبلی مقایسه کنیم. برای انجام این تمرین مراحل زیر را دنبال کنید:

۱. کتابخانه numpy و کتابخانه‌ها و کلاس‌های ضروری برای Keras را فرا بخوانید:

```
# Import the Libraries
from keras.models import Sequential
from keras.layers import Conv2D, MaxPool2D, Flatten, Dense
import numpy as np
from tensorflow import random
```

[39] مترجم: واقعا زیبا بود!

۲. مدل را با استفاده از کلاس‌های ترتیبی شروع کنید:

```
# Initiate the classifier
seed = 42
np.random.seed(seed)
random.set_seed(seed)
classifier = Sequential()
```

۳. اولین لایه شبکه عصبی پیچشی را اضافه کنید. ابعاد ورودی را (64,64,3) و تابع فعال‌سازی را ReLU درنظر بگیرید. سپس ۳۲ عدد تشخیص‌دهنده ویژگی با اندازه 3×3 اضافه کنید. دو لایه پیچشی مازاد با همان تعداد تشخیص‌دهنده ویژگی و همان اندازه اضافه کنید:

```
classifier.add(Conv2D(32,3,3,input_shape=(64,64,3),activation='relu'))
classifier.add(Conv2D(32,3,3,activation='relu'))
classifier.add(Conv2D(32,3,3,activation='relu'))
```

٤. حال لایه تجمیع را با اندازه تصویر 2×2 اضافه کنید:

```
classifier.add(MaxPool2D(2,2))
```

۵. یک Conv2D دیگر با پارامترهایی مشابه مرحله ۳، و برای تکمیل آن، یک لایه تجمیع دیگر با پارامترهای مشابه مرحله ٤ اضافه کنید:

```
classifier.add(Conv2D(32, (3, 3), input_shape = (64, 64, 3),
activation = 'relu'))
classifier.add(MaxPool2D(2,2))
```

٦. یک لایه مسطح‌سازی برای تکمیل معماری شبکه عصبی پیچشی اضافه کنید:

```
classifier.add(Flatten())
```

۷. اولین لایه متراکم شبکه عصبی مصنوعی را با اندازه ۱۲۸ اضافه کنید. سه لایه متراکم مازاد دیگر با پارامترهای مشابه اضافه کنید:

```
classifier.add(Dense(128,activation='relu'))
classifier.add(Dense(128,activation='relu'))
classifier.add(Dense(128,activation='relu'))
classifier.add(Dense(128,activation='relu'))
```

۸. لایه خروجی شبکه عصبی مصنوعی را اضافه کنید. چون این یک مسئله دسته‌بندی است، اندازه لایه خروجی ۱ است. همچنین تابع فعال‌سازی sigmoid می‌باشد:

```
classifier.add(Dense(1,activation='sigmoid'))
```

۹. شبکه را با یک بهینه‌ساز SGD کامپایل کنید و دقت مدل را در حین فرآیند آموزش محاسبه نمایید:

```
classifier.compile(optimizer='SGD',loss='binary_crossentropy',
metrics=['accuracy'])
```

۱۰. یک تولیدکننده داده برای آموزش و تست ایجاد کنید. اندازه داده‌های آموزش و تست را به 1/255 کاهش دهید تا همه‌ی مقادیر بین ۰ و ۱ شوند. این پارامترها را تنها برای تولیدکننده داده‌های آموزش، به‌صورت روبرو تعیین کنید: shear_range=0.2, zoom_range=0.2, and horizontal_flip=True

```
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)
test_datagen = ImageDataGenerator(rescale = 1./255)
```

۱۱. یک مجموعه آموزش از پوشه آموزش ایجاد کنید. "…/dataset/training_set" جایی است که مجموعه آموزش قرار داده شده است. در شبکه عصبی پیچشی ما، اندازه تصویر 64×64 است. بنابراین باید تصاویری با همین اندازه به شبکه داده شود. batch_size تعداد تصاویر موجود در یک دسته را مشخص می‌کند که در اینجا ۳۲ است. Class_mode به‌صورت binary تعیین شده است زیرا ما با یک دسته‌بند دوتایی کار می‌کنیم:

```
training_set = train_datagen.flow_from_directory('../dataset/training_
set',
target_size = (64, 64),
batch_size = 32,
class_mode = 'binary')
```

۱۲. مرحله ۱۱ را برای مجموعه داده تست تکرار کنید. دقت کنید که باید پوشه محل قرارگیری داده‌های تست مشخص شود که به‌صورت '../dataset/test_set' است:

```
test_set = test_datagen.flow_from_directory('../dataset/test_set',
target_size = (64, 64),
batch_size = 32,
class_mode = 'binary')
```

۱۳.درنهایت با این داده‌ها مدل را برازش کنید. مقدار steps_per_epoch را ۱۰۰۰۰ و مقدار validation_steps را ۲۵۰۰ قرار دهید. اجرای این مرحله ممکن است مقداری زمان‌گیر باشد:

```
classifier.fit_generator(training_set,
steps_per_epoch = 10000,
epochs = 2,
validation_data = test_set,
validation_steps = 2500,
shuffle=False)
```

کد معرفی شده این خروجی را خواهد داد:

```
Epoch 1/2
10000/10000 [==============================] - 4376s 438ms/step -
loss: 0.3920 - accuracy: 0.8201 - val_loss: 0.3937 - val_accuracy:
0.8531
Epoch 2/2
10000/10000 [==============================] - 5146s 515ms/step -
loss: 0.2395 - accuracy: 0.8995 - val_loss: 0.4694 - val_accuracy:
0.8454
```

دقت در مجموعه ارزیابی 84.54% است. در اینجا از چند لایه شبکه عصبی مصنوعی و بهینه‌ساز SGD استفاده کردیم.

تاکنون در چند حالت مختلف و با ترکیبات متفاوتی مدل را ساختیم. به‌نظر می‌رسد بهترین دقت در مدل با انجام دادن موارد زیر بدست می‌آید:

- اضافه کردن چند لایه شبکه عصبی پیچشی
- اضافه کردن چند لایه شبکه عصبی مصنوعی
- استفاده از sigmoid به‌عنوان تابع فعال‌سازی
- استفاده از بهینه‌ساز adam
- افزایش اندازه epoch به حدود ۲۵ (این کار به زمان محاسباتی بالایی نیاز دارد- مطمئن شوید که GPU برای انجام این کار داشته باشید). این کار باعث افزایش دقت پیش‌بینی می‌شود.

در نهایت، یک تصویر ناشناخته را پیش‌بینی می‌کنیم. این تصویر را به الگوریتم می‌دهیم و بررسی می‌کنیم که آیا تصویر به‌درستی دسته‌بندی شده‌است یا خیر. در تمرین بعدی نشان می‌دهیم که برای دسته‌بندی عکس‌های جدید چگونه از مدل استفاده کنیم.

تمرین ۷-۰٤: دسته‌بندی یک تصویر جدید

در این تمرین سعی می‌کنیم که یک تصویر جدید را دسته‌بندی کنیم. تصویر هنوز توسط الگوریتم دیده نشده است، بنابراین از این تمرین استفاده می‌کنیم تا الگوریتم را تست کنیم. شما می‌توانید هرکدام از الگوریتم‌های این فصل را اجرا کنید (اگرچه الگوریتمی که بیش‌ترین دقت را داشت ترجیح داده می‌شود) و از مدل برای دسته‌بندی تصویر استفاده کنید.

نکته: عکسی که برای تمرین بعدی استفاده می‌شود را می‌توان در مخزن GitHub کتاب در آدرس < https://packt.live/39tID2C > پیدا کرد.

قبل از انجام تمرین مطمئن شوید که test_image_1 را از مخزن GitHub کتاب در سیستم خود دانلود کرده باشید. این تمرین به‌دنبال تمرین‌های گذشته آمده‌است پس باید حتماً یکی از الگوریتم‌های ایجاد شده در این فصل را آماده برای اجرا داشته باشید.

برای انجام این تمرین مراحل زیر را دنبال کنید:

۱. تصویر را بارگذاری کنید. "test_Image_1.jpg" مسیر تصویر تست است. لطفاً مسیر را به مسیری که تصویر در سیستم شما ذخیره شده‌است تغییر دهید. تصویر را نگاه کنید و آن را شناسایی کنید:

```
from keras.preprocessing import image
new_image = image.load_img('../test_image_1.jpg', target_size = (64,
64))
new_image
```

۲. برچسب‌های کلاس را که در ویژگی class_indices مربوط به داده‌های آموزش قرار دارد، پرینت بگیرید:

```
training_set.class_indices
```

۳. تصویر را پردازش کنید:

```
new_image = image.img_to_array(new_image)
new_image = np.expand_dims(new_image, axis = 0)
```

٤. تصویر جدید را پیش‌بینی کنید:

```
result = classifier.predict(new_image)
```

٥. تابع پیش‌بینی خروجی را به‌صورت ۱ یا صفر می‌دهد. برای این که این‌ها را به "گل" و "خودرو" تبدیل کنید از if...else به‌صورت زیر استفاده کنید:

```
if result[0][0] == 1:
     prediction = 'It is a flower'
else:
     prediction = 'It is a car'
print(prediction)
```

٦. کد نوشته‌شده این خروجی را می‌دهد:

```
It is a car
```

Test_image_1 تصویری از یک خودرو است (می‌توانید خودتان تصویر را مشاهده کنید) و به‌درستی توسط مدل پیش‌بینی شده بود.

در این تمرین مدل را آموزش دادیم و سپس به آن یک تصویر از خودرو دادیم. با این کار متوجه شدیم که الگوریتم به‌درستی تصویر را دسته‌بندی کرده‌است. شما می‌توانید مدل را روی هر نوع عکسی با همین فرآیند آموزش دهید. به‌عنوان مثال اگر مدل را روی تصاویر ریه‌های عفونت کرده و ریه‌های سالم آموزش دهید، مدل قادر خواهد بود که تشخیص دهد یک عکس جدید مربوط به ریه عفونت کرده است یا ریه سالم.

در فعالیت بعدی تمرینی داریم روی مدلی که در "تمرین ۷-٠٤: دسته‌بندی یک تصویر جدید"، بدست آمد.

فعالیت ۷-۰۲: دسته‌بندی یک تصویر جدید

در این فعالیت شما سعی خواهید کرد که یک تصویر جدید را دسته‌بندی کنید، درست مانند کاری که در تمرین قبلی کردیم. تصویر هنوز توسط الگوریتم دیده نشده‌است، بنابراین از این فعالیت استفاده می‌کنیم تا الگوریتم را تست کنیم. شما می‌توانید هرکدام از الگوریتم‌های این فصل را اجرا کنید (اگرچه الگوریتمی که بیش‌ترین دقت را داشت ترجیح داده می‌شود) و از مدل برای دسته‌بندی تصویر استفاده کنید. مراحل انجام این فعالیت به‌شرح ذیل است:

١. یکی از الگوریتم‌های این فصل را اجرا کنید.

۲. تصویر (test_image_2) را از سیستم خود بارگذاری کنید.
۳. تصویر را با استفاده از الگوریتم پردازش کنید.
٤. موضوع تصویر را پیش‌بینی کنید. می‌توانید تصویر را خودتان ببینید و بررسی کنید که آیا پیش‌بینی درست بوده است یا خیر.

نکته: عکسی که در این فعالیت استفاده شد را می‌توان در مخزن GitHub کتاب در آدرس < https://packt.live/39tID2C > پیدا کرد.

قبل از انجام فعالیت مطمئن شوید که test_image_2 را از مخزن GitHub کتاب در سیستم خود دانلود کرده باشید. این فعالیت به‌دنبال تمرین‌های گذشته آمده‌است پس باید حتماً یکی از الگوریتم‌های ایجاد شده در این فصل را آماده برای اجرا داشته باشید.

بعد از اجرای این مراحل می‌توانید انتظار داشته باشید که این خروجی را بگیرید:

```
It is a flower
```

نکته: پاسخ این فعالیت در صفحه ۳۹٦ آمده است.

در این فعالیت ما کاراترین مدل این فصل را با تغییر پارامترها آموزش دادیم. از جمله این پارامترها بهینه‌ساز و تابع فعال‌سازی در لایه خروجی هستند. ما دسته‌بند را بر اساس یک تصویر جدید تست کردیم و مشخص شد که تصویر به‌درستی دسته‌بندی شده‌است.

خلاصه

ما در این فصل در مورد چرایی بینایی ماشین و نحوه کار آن مطالعه کردیم. آموختیم که چرا بینایی ماشین یکی از جذاب‌ترین حوزه‌های یادگیری ماشین است. سپس با شبکه‌های عصبی پیچشی کار کردیم، در مورد معماری آن‌ها آموختیم و نحوه‌ی ساختن آن‌ها را برای کاربردهای واقعی یادگرفتیم. همچنین سعی کردیم که الگوریتم‌های خود را با اضافه کردن لایه‌های بیش‌تری به شبکه عصبی مصنوعی و شبکه عصبی پیچشی و تغییر تابع فعال‌سازی و بهینه‌ساز، بهبود بدهیم. ما توابع فعال‌سازی مختلفی را امتحان کردیم.

در انتها قادر بودیم که به‌درستی تصاویر جدید خودرو و گل را توسط الگوریتم، دسته‌بندی کنیم. به‌یاد داشته باشید که تصاویر خودرو و گل می‌تواند توسط هر نوع دیگری از تصویر جایگزین شود؛ مثلاً تصویر ببر و آهو،

یا تصاویر MRI مغز با تومور یا بدون تومور. هر مسئله دسته‌بندی تصویر دوتایی کامپیوتری با همین روش قابل حل است.

در فصل بعدی روشی کاراتر برای کار کردن روی بینایی ماشین را مطالعه خواهیم کرد. این روش کمتر وقت می‌گیرد و پیاده‌سازی ساده‌تری دارد. این فصل به ما یاد خواهد داد که چگونه مدل‌های از پیش آموزش دیده شده را برای کاربردهای خودمان تنظیم کنیم که باعث می‌شود مدل‌های دقیق‌تری با سرعت آموزش بیش‌تری ایجاد شوند. مدل‌هایی که استفاده می‌شوند VGG-16 و ResNet50 نام دارند که مدل‌های از پیش آموزش دیده‌ی معروفی برای دسته‌بندی عکس‌ها هستند.

منابع فارسی جهت تسلط بیشتر

https://fa.wikipedia.org/wiki/%D8%AD%D8%B3%D8%A7%D8%B3%DB%8C%D8%AA_%D9%88_%D9%88%DB%8C%DA%98%DA%AF%DB%8C

https://planet.sito.ir/%D8%A7%D8%B3%D8%AA%D9%81%D8%A7%D8%AF%D9%87-%D8%A7%D8%B2-image-data-augmentation-%D8%AF%D8%B1-%DA%A9%D8%AA%D8%A7%D8%A8%D8%AE%D8%A7%D9%86%D9%87-keras/

https://blog.faradars.org/introduction-deep-learning-keras/

http://blog.class.vision/1397/03/train-convolutional-neural-network-in-keras/

https://blog.faradars.org/deep-learning-with-python/

https://blog.faradars.org/convolutional-neural-networks/

https://blog.faradars.org/%D8%A2%D8%B4%D9%86%D8%A7%DB%8C%DB%8C-%D8%A8%D8%A7-%D8%B4%D8%A8%DA%A9%D9%87%E2%80%8C%D9%87%D8%A7%DB%8C-%D8%B9%D8%B5%D8%A8%DB%8C-%D9%BE%DB%8C%DA%86%D8%B4%DB%8C-cnn/