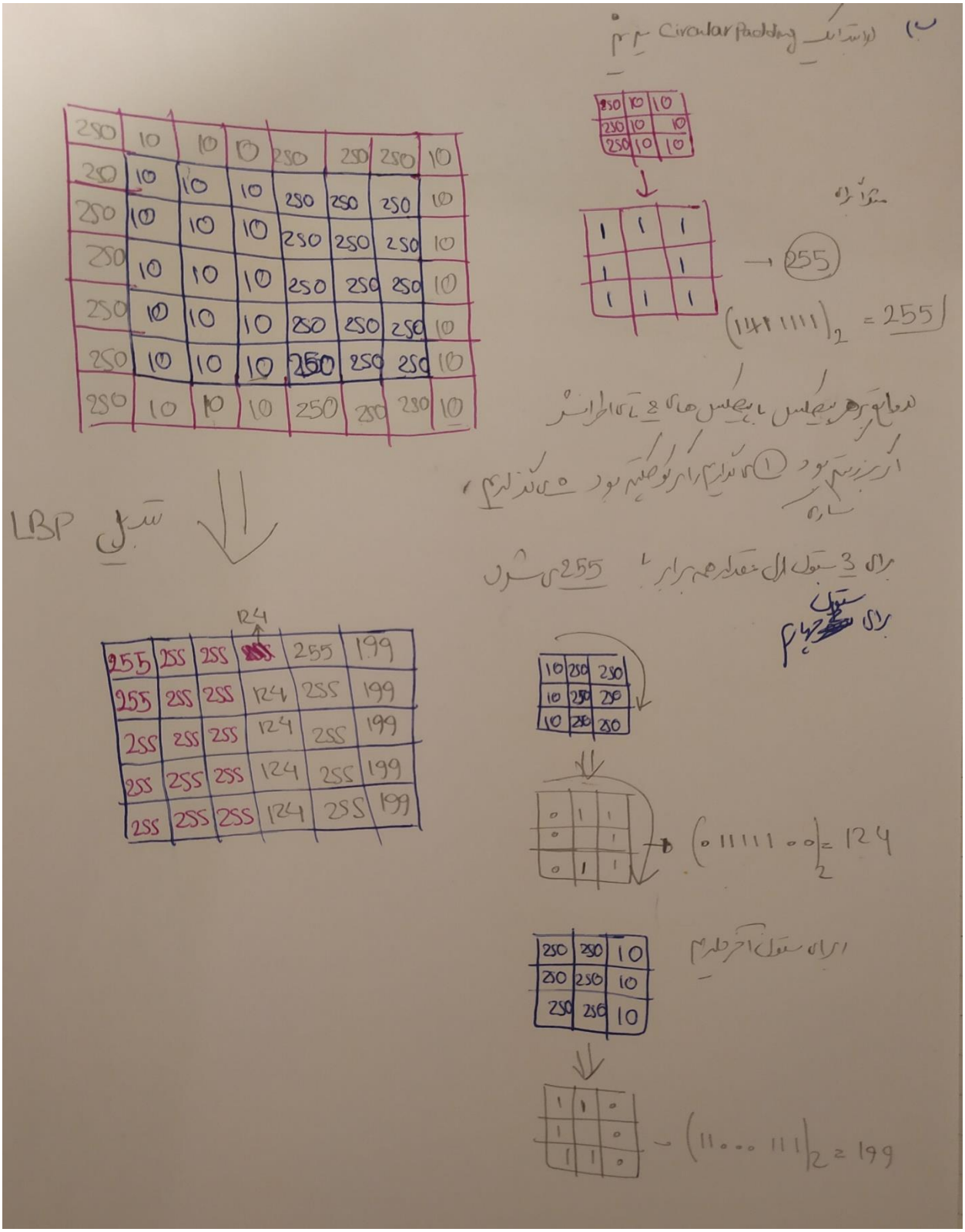
سوال یک

الف)بله نیاز به پدینگ داریم چون برای نقط گوشه و اطراف ماتریس نیاز به ۸ همسایه داریم و اگه پدینگ اضافه نکنیم ۸ همسایه نداریم من برای ماتریس از circular paddingاستفاده کردم.

ب)پاسخ به صورت عکس زیر است:

پاسخ در صفحه بعد است.



سوال دو

من سه توصیف گر compactness,solidity,eccentricity را در نظر گرفتم که فرمول این سه به صورت زیر است:

$$Compactness = \frac{4\pi\ Area}{Perimeter^2}$$

$$Solidity = \frac{Area}{ConvexArea}$$

$$Eccentricity = \sqrt{1 - \left(\frac{MinorAxisLength}{MajorAxisLength}\right)^2}$$

و هر تابع را با توجه به فرمول بالا پیاده سازی کردم.برای تابع distancecriteria نکات زیر را در نظر گرفتم:

دو شکل دایره داریم ،دایره compactness بالایی دارد که برای این دو شکل عددی بالای 0.8 است در واقع compactness دایره برابر با ۱ هست ولی این شکل کامل دایره نیستند.هم چنین solidity بالایی دارد که عددی نزدیک به یک هست ولی eccentricity ان برابر با ۰ است.به همین علت برای compactness بالای 0.8 یک شرط گذاشتم وفاصله دو ویژگی دیگر را حساب کردم که کد آن به صورت زیر است:

```
# check compatness
if x[0]>0.8 and y[0]>0.8:
    output=np.sqrt(abs(x[1]-y[1])**2+abs(x[2]-y[2])**2)
    return output
```

تمام اشکال در تصویر به جز شکل هایی که شبیه ستاره هستند دارای solidity بسیار بالا نزدیک به یک هستند به همین علت برای solidity زیر 0.7 یک شرط گذاشتم و تفاضل مقادیر compactness ان ها را حساب کردم به این علت eccentricity مقایسه نکردم چون مقدار eccentricity برای یک از این ستاره مقدار بالایی هست وبرای یکی مقدار کمتری است چون یکیش مقدار بیشتری کشیده هست. کد ان به صورت زیر است:

```
elif  x[1]<0.7 and y[1]<0.7:
    return  abs(x[0]-y[0])
```

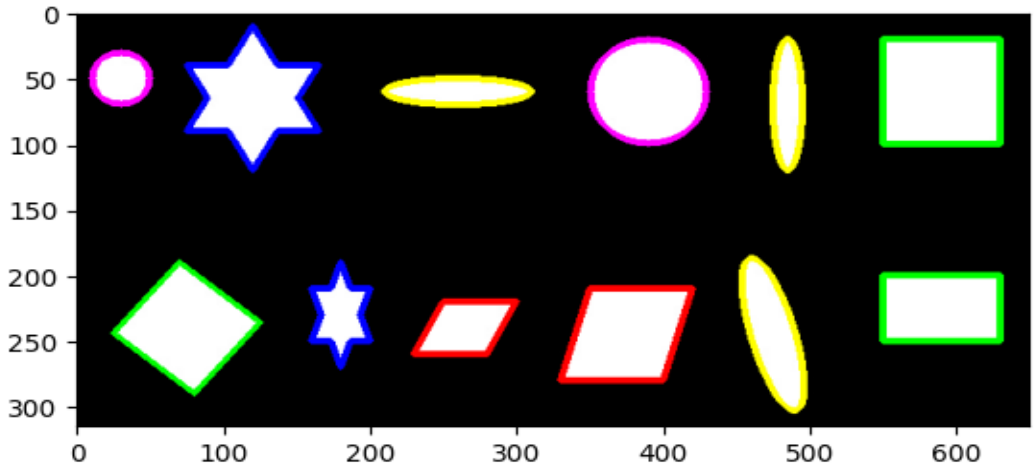
بیضی مقدار compactness نسبتا کمی دارد به همین علت برای آن نیز یک شرط گذاشتم و فاصله دو ویژگی دیگر را حساب کردم

```
# check compatness
if 0.35<x[0]<0.5 and 0.35<y[0]<0.5:
    output=np.sqrt(abs(x[1]-y[1])**2+abs(x[2]-y[2])**2)
    return output
```

مقدار eccentricity برای مربع و مستطیل برابر با ۰ هست ولی برای یکی از مربع ها در شکل یکمی مقدارش بیشتر بود به همین علت برای eccentricity کمتر از۰.۲۵ نیز یک شرط گذاشتم

```
#check eccentricity
elif  x[2]<0.25 and y[2]<0.25:
    output=np.sqrt(abs(x[0]-y[0])**2+abs(x[1]-y[1])**2)
    return output
```

و در نهایت اگر هیچ کدام از این شرط ها برقرار نبود فاصله سه ویژگی را حساب میکنم.شکل نهایی به صورت زیر است:



سوال سوم

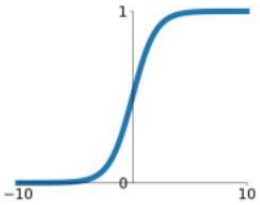
توابع فعال‌ساز (activation functions) در شبکه‌های عصبی به دلیل خطی بودن ضرب داخلی وجود توابع فعال سازی غیرخطی ضروری است. وظیفه اصلی توابع فعال‌ساز، تبدیل ورودی شبکه به خروجی مطابق با طبقه‌بندی مورد نظر است.

۱. sigmoid:

تابع sigmoid دامنه آن بین ۰ و ۱ است. تابع sigmoid به دلیل داشتن خروجی محدود بین ۰ و ۱، برای مسائلی که خروجی باید بین این دو مقدار باشد، مانند مسائل طبقه‌بندی دو دسته‌ای (binary classification)، مناسب است. همچنین، این تابع در مسائلی که مقادیر ورودی باید نرمال شوند، مانند مسائل با مقیاس‌های مختلف ورودی، مفید است. اما مشکل اصلی این تابع،این است که خروجی ان همیشه مثبت است و در بعضی مسائل ممکن است دچار مشکل شویم به همین علت تابع Tanh مطرح کردیم.فرمول تابع sigmoid به صورت زیر است:

Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

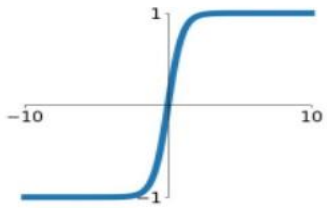


۲. Tanh:

تابع Tanh شبیه sigmoid است فقط دامنه آن بین -۱ و ۱ است. این تابع به دلیل اینکه مقادیر خروجی آن در بازه منفی و مثبت است، برای مسائلی که باید خروجی متعادلی داشته باشند، مانند مسائل طبقه‌بندی چند دسته‌ای (multi-class classification)، مناسب است. همچنین، در مقایسه با تابع sigmoid، این تابع کمتر از مشکل اشباع شدن در ورودی‌های بزرگ و کوچک رنج می‌کند.

tanh

$$\tanh(x)$$



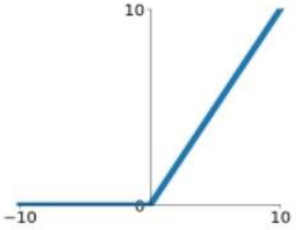
۳. ReLU:

تابع ReLU (Rectified Linear Unit) یک تابع خطی‌نمایی است که در صورتی که ورودی مثبت باشد، به صورت خطی و در صورتی که ورودی منفی باشد، صفر می‌دهد. به دلیل سادگی و سرعت پردازش بالا، تابع ReLU در شبکه‌های عصبی بسیار محبوب است. همچنین، این تابع به دلیل عدم خطی بودن خروجی، می‌تواند مسائلی را که تابع خطی نمی‌تواند حل کند، مانند مسائل غیرخطی و تشخیص تصاویر.در مقایسه با sigmoid,tanh چون برای مقادیر بالا مشتق به سمت ۰ میل میکند و بعد از یک مدتی اشباع میشوند ولی relu این مشکل را ندارد.

مشکل اصلی تابع ReLU این است که ورودی‌های منفی را صفر می‌کند و به این دلیل، اگر ورودی‌هایی با مقدار کوچکتر از صفر داشته باشیم، این تابع آن‌ها را به صفر می‌رساند و این مسئله به عنوان مشکل شروع به از دست دادن اطلاعات می‌کند. بنابراین، توابعی مانند PReLu به صورت تغییر داده شده ازتابع ReLU استفاده می‌شوند تا این مشکل را حل کنند.

ReLU

$$\max(0, x)$$

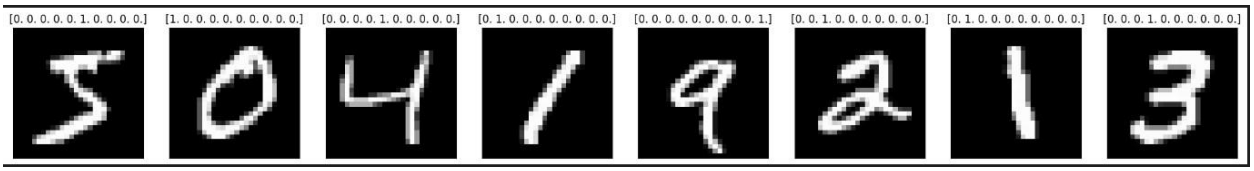


۴. PReLu:

تابع PReLu (Parametric Rectified Linear Unit) نیز مانند تابع ReLU عمل می‌کند، اما در صورتی که ورودی منفی باشد، به جای صفر، یک مقدار منفی از پارامتری که در فرآیند آموزش تعیین می‌شود، استفاده می‌شود. این تابع مشکل از دست دادن اطلاعات در ورودی‌های کوچک را حل کرده و همچنین مانند تابع ReLU، سرعت پردازش بالایی دارد. به دلیل اینکه پارامتر اضافی برای تعیین مقدار منفی در ورودی‌های منفی اضافه شده است، این تابع نسبت به تابع ReLU پیچ بار پیچیده‌تر است و نیاز به بیشترین تلاش در فرآیند آموزش دارد.

سوال چهارم

در ابتدا داده‌های آموزش و آزمون بارگیری میشوند. سپس برای تبدیل برچسب‌های کلاس به فرمت ماتریس دودویی، از تابع to_categorical کتابخانه Keras استفاده می‌شود. با استفاده از این تابع، برچسب‌های کلاس به فرمت ماتریس دودویی تبدیل می‌شوند که در آن هر سطر متناظر با یک نمونه است و در ستون‌های آن، برچسب کلاس مربوط به آن نمونه به صورت دودویی ذخیره می‌شود. به عنوان مثال، برای یک داده با برچسب ۳، مقدار متناظر با آن در ماتریس دودویی به صورت [۰, ۰, ۰, ۱, ۰, ۰, ۰, ۰, ۰, ۰] قرار می‌گیرد. در نهایت، تعداد کلاس‌ها به عنوان ۱۰ تعریف شده و در متغیر num_classes قرار داده می‌شود.



در کد، ابتدا یک شیء از کلاس Sequential تعریف شده و به متغیر model اختصاص داده می‌شود. سپس با استفاده از تابع add، لایه‌های شبکه به ترتیب اضافه می‌شوند. در اینجا ابتدا با استفاده از تابع Input یک لایه ورودی به شکل تصویر تعریف می‌شود. اندازه تصویر ورودی با استفاده از ویژگی shape از داده‌های آموزش به عنوان اندازه ورودی به شبکه تعیین می‌شود. سپس با استفاده از تابع Flatten، تصویر ورودی به یک بردار یک بعدی تبدل شده و به عنوان ورودی لایه بعدی استفاده می‌شود. در نهایت با استفاده از تابع Dense، یک لایه کاملاً متصل با تابع فعال‌سازی softmax تعریف می‌شود. این لایه شامل تمام نورون‌های لازم برای دسته‌بندی تصاویر به کمک یکی از ۱۰ دسته مختلف دیجیت‌های دستنویس MNIST است.

در نهایت، مقدار num_classes که در قسمت قبل تعریف شده است، به عنوان تعداد نورون‌های خروجی لایه کاملاً متصل تعیین می‌شود.

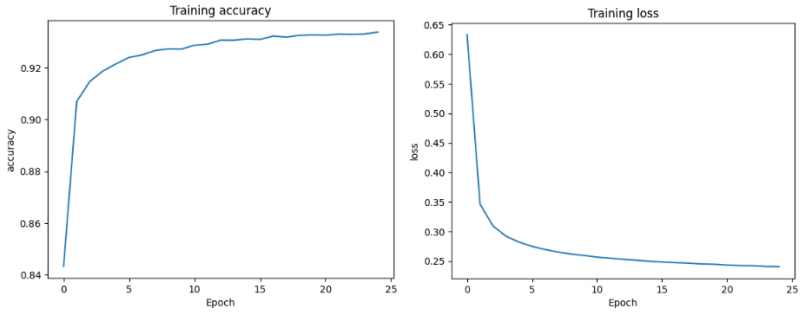
```
# define model
model = keras.Sequential()
model.add(keras.layers.Input(shape=x_train[0].shape))
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(units=num_classes, activation='softmax'))
```

در ادامه مدل شبکه عصبی که در قسمت قبل تعریف شده بود، با استفاده از تابع `compile` کامپایل می‌شود. کامپایل کردن شبکه به این معنی است که مدل با تعیین تابع هزینه، بهینه‌ساز و معیار دقت آموزش، آماده آموزش می‌شود.

در این کد، تابع هزینه `categorical_crossentropy`، بهینه‌ساز `adam` و معیار دقت `accuracy` برای مدل شبکه تعیین شده است. تابع هزینه `categorical_crossentropy` یک تابع هزینه مناسب برای مسائل دسته‌بندی چند کلاسه است و به صورت پیش‌فرض برای مسائل دسته‌بندی چند کلاسه در Keras استفاده می‌شود.

در ادامه کد، ابتدا داده‌های آموزش و برچسب‌های آن‌ها به تابع fitداده می‌شود. سپس با استفاده از پارامتر batch_size، تعداد نمونه‌هایی که در هر بار آموزش مورد استفاده قرار می‌گیرد، تعیین می‌شود. در اینجا، تعداد ۱۰۰ نمونه برای هر بار آموزش استفاده می‌شود.سپس با استفاده از پارامتر epochs، تعداد دوره‌های آموزش مدل تعیین می‌شود. در اینجا، تعداد ۲۵ دوره برای آموزش مدل استفاده شده است.پس از آموزش مدل، با استفاده از پارامتر validation_data، داده‌های ارزیابی به مدل داده می‌شود.در نهایت، تاریخچه آموزش مدل در متغیر historyذخیره می‌شود که شامل معیارهای مختلفی مانند دقت و هزینه برای هر دورهآموزش مدل است.

بعد از train دو نمودار زیر را داریم:




```
test_loss, test_acc = model.evaluate(x_test,  y_test, verbose=2)

print('\nTest accuracy:', test_acc)
```

Python

```
313/313 - 1s - loss: 0.2641 - accuracy: 0.9264 - 665ms/epoch - 2ms/step

Test accuracy: 0.9264000058174133
```

از خروجی بالا بر روی داده های تست نتیجه میگیریم که مدل ما overfit نشده است.

در ادامه برای تعریف مدل شبکه عصبی با استفاده از روش Functional API از کتابخانه Keras استفاده می‌شود. در این روش، به جای استفاده از کلاس `Sequential`، از توابع `Input` و `Model` برای تعریف لایه‌های شبکه استفاده می‌شود.

در این کد، ابتدا با استفاده از تابع Input، یک لایه ورودی به شکل تصویر ورودی به شبکه تعریف می‌شود. سپس با استفاده از تابع Flatten، تصویر ورودی به یک بردار یک بعدی تبدل شده و به عنوان ورودی لایه بعدی استفاده می‌شود. در نهایت با استفاده از تابع `Dense`، یک لایه کاملاً متصل با تابع فعال‌سازی softmax تعریف می‌شود. این لایه شامل تمام نورون‌های لازم برای دسته‌بندی تصاویر به کمک یکی از ۱۰ دسته مختلف دیجیت‌های دستنویس MNIST است.در نهایت، با استفاده از تابع Model، مدل شبکه با ورودی لایه ورودی و خروجی لایه‌های تعریف شده، تعریف می‌شود و به عنوان مدل نهایی در متغیر model1 ذخیره می‌شود.

تفاوت اصلی بین دو روش Sequential و Functional API، در تعریف شبکه است. در روش Sequential، لایه‌های شبکه به صورت ترتیبی به مدل اضافه می‌شوند. در حالی که در روش Functional API، لایه‌ها در قالب یک گراف برای تعریف شبکه استفاده می‌شوند و می‌توان به صورت پیچیده‌تر و چندشاخه‌ای شبکه را تعریف کرد. همچنین، در روش Functional API، می‌توان مدل‌هایی با چند ورودی و چند خروجی تعریف کرد که با روش Sequential امکان‌پذیر نیست. به طور کلی، روش Functional API برای شبکه‌های پیچیده و با پیوستگی بیشتر مناسب است، در حالی که روش Sequential برای شبکه‌های ساده و خطی‌تر مناسب است.

```
inputs = keras.layers.Input(shape=x_train[0].shape)
x = keras.layers.Flatten()(inputs)
x = keras.layers.Dense(units=num_classes, activation='softmax')(x)
model1 = keras.models.Model(inputs=inputs, outputs=x)
```

ادامه کد برای functional مانندsequential است.

سوال پنجم

خیر، هر شبکه‌ای که به صورت Functional پیاده‌سازی شود، نمی‌تواند به صورت Sequential نیز پیاده‌سازی شود. دلیل این امر این است که شبکه‌های Sequential و Functional دو نوع شبکه متفاوت هستند و دارای ساختارهای متفاوتی هستند.در شبکه‌های Sequential، لایه‌ها به ترتیب خاصی به یکدیگر متصل شده‌اند و خروجی یک لایه به عنوان ورودی لایه بعدی استفاده می‌شود. به عبارت دیگر، ورودی شبکه به یک لایه داده می‌شود و خروجی آن به عنوان ورودی لایه بعدی استفاده می‌شود و این فرآیند به ترتیب ادامه می‌یابد تا به خروجی نهایی برسیم.در شبکه‌های Functional، می‌توان لایه‌های مختلف را به صورت غیرخطی و با ساختارهای مختلف به هم وصل کرد و شبکه‌های با ساختار غیرخطی و غیرترتیبی را پیاده‌سازی کرد. بنابراین، شبکه‌های Functional به طور کلی برای پیاده‌سازی شبکه‌هایی که ساختار غیرخطی دارند و یا می‌خواهند از ارتباطات

غیرخطی میان لایه‌ها استفاده کنند، مناسب هستند.بنابراین، نمی‌توان به همیشه یک شبکه Functional را به شبکه Sequential تبدیل کرد.

سوال ششم

الف)اگر با یک کرنل 7*7 کانوالو کنیم خروجی ما ابعادش 1*1 هست و تعداد کانال ها نیز سه تا است.

ب)در مرحله اول خروجی ما ابعادش 5=1+3-7 یعنی 5*5 خواهد بود در مرحله دوم 3=1+3-5 یعنی 3*3 خواهد بود و در مرحله اخر 1=1+3-3 یعنی ابعاد 1*1 و تعداد کانال ها 3 تا است.

ج) در مورد استفاده از یک کرنل 7 در 7برای کانوالو کردن یک تصویر 7در 7، باید گفت که این روش در مقایسه با استفاده از چند کرنل با اندازه کوچکتر، مزایای کمتری دارد. به عنوان مثال، استفاده از کرنل‌های ۳*۳، امکان استخراج ویژگی‌های مختلفی را از تصویر ورودی فراهم می‌کند و باعث افزایش دقت در پردازش تصویر می‌شود. همچنین، در این روش تعداد پارامترها بسیار بیشتر است و این می‌تواند منجر به بالا بردن زمان آموزش و پیچیدگی مدل شودو هم چنین باعث استخراج ویژگی های عمیق تری از تصویر میشود.

استفاده از کرنل‌های کوچکتر ۳*۳ در سه مرحله کانوالو، به دلیل استفاده از تابع فعال‌سازی غیرخطی مانند ReLU، منجر به استخراج ویژگی‌های غیرخطی از تصویر ورودی می‌شود. در واقع، استفاده از تابع فعال‌سازی غیرخطی، مانع از خطا مدل می‌شود و می‌تواند به بهبود دقت و کیفیت ویژگی‌های استخراج شده در مدل کمک کند.در مقابل، استفاده از یک کرنل ۷*۷ برای کانوالو کردن یک تصویر ۷*۷ باعث استخراج ویژگی‌های خطی تر از تصویر ورودی می‌شود،زیرا این روش به صورت مستقیم از تصویر ورودی استفاده می‌کند و از تابع فعال‌سازی خطی استفاده می‌کند. بنابراین، از لحاظ خطی یا غیر خطی بودن، استفاده از سه کرنل ۳*۳در سه مرحله کانوالو مزیت بیشتری دارد، زیرا با استفاده از تابع فعال‌سازی غیرخطی، امکان استخراج ویژگی‌های پیچیده‌تر از تصویر ورودی و بهبود دقت و کیفیت ویژگی‌های استخراج شده وجود دارد.