Q1

الف

استفاده از نرخ یادگیری بالا در آموزش مدل هوش مصنوعی می‌تواند مشکلات زیر را ایجاد کند:

بیش‌برازش :(Overfitting) استفاده از نرخ یادگیری بالا می‌تواند منجر به بیش‌برازش شود. در این حالت، مدل به طور خاص به داده‌های آموزشی عادت می‌کند و قدرت تعمیم‌پذیری آن در مورد داده‌های جدید کاهش می‌یابد. به عبارت دیگر، مدل به طور دقیق برازش می‌شود و نمی‌تواند الگوهای کلی را در داده‌های جدید تشخیص دهد.

سرعت آموزش بالا: استفاده از نرخ یادگیری بالا می‌تواند منجر به سرعت بالای آموزش مدل شود. این ممکن است باعث شود که مدل به سرعت به نقاط بهینه محلی برسد و به نقاط بهینه سراسری نرسد. این به معنای این است که مدل ممکن است در یک نقطه محلی مطلوب قرار گیرد، اما دقت و عملکرد آن در داده‌های جدید کاهش می‌یابد.

عدم پایداری: استفاده از نرخ یادگیری بالا ممکن است منجر به عدم پایداری در آموزش مدل شود. این به این معنی است که مدل ممکن است به طور پیوسته در حال تغییر و بهبود باشد و نتایج آموزش آن قابل اعتماد نباشد. این مشکل می‌تواند باعث شود که مدل در طول زمان نتایج نامناسبی را تولید کند و نتواند به طور پایدار به یک حالت مطلوب برسد.

حساسیت به داده‌های نویزی: استفاده از نرخ یادگیری بالا ممکن است باعث شود که مدل بیش‌برازش به داده‌های نویزی شود. این به این معنی است که حتی داده‌های نویزی کوچک هم می‌توانند تأثیر زیادی بر روی آموزش مدل داشته باشند و به نتایج نامناسبی منجر شوند.

بهترین راه برای تشخیص مشکلات مربوط به لرنینگ ریت بالا، چک کردن نمودارهای ارزیابی حین train شدن مدل است. در صورتی که loss به طور ناگهانی افزایش پیدا کند احتمالا learning rate بالا بوده و مدل دچار ناپایداری شده.

مشکلات دیگر را میتوان از روی مشاهده متریک های مدل برای داده های تست و validation پیدا کرد مثلا:

مشاهده نمودار عملکرد: بررسی نمودار عملکرد مدل در طول زمان می‌تواند مفید باشد. اگر عملکرد مدل در داده‌های آموزش بهبود پیدا کرده ولی در داده‌های ارزیابی (validation) بهبودی نداشته یا حتی بهبودی نسبت به مرحله‌ی قبل نداشته باشد، این ممکن است نشان دهنده بیش‌برازش باشد.

بررسی خطاها و دقت: بررسی خطاها و دقت مدل در داده‌های آموزش و ارزیابی می‌تواند نشان دهنده وجود بیش‌برازش باشد. اگر مدل در داده‌های آموزش خطاها را به طور قابل توجهی کاهش می‌دهد، اما در داده‌های ارزیابی این کاهش خطا کمتر یا حتی عدم بهبودی را نشان می‌دهد، این ممکن است نشان دهنده بیش‌برازش باشد.

ارزیابی عملکرد در داده‌های جدید: بررسی عملکرد مدل در داده‌های جدید و مستقل از داده‌های آموزش و ارزیابی می‌تواند نشان دهنده قدرت تعمیم‌پذیری مدل باشد. اگر مدل در داده‌های جدید نتایج نامطلوبی را تولید می‌کند یا دقت آن کاهش می‌یابد، این ممکن است نشان دهنده بیش‌برازش باشد.

استفاده از معیارهای ارزیابی مشتقه: استفاده از معیارهای ارزیابی مشتقه مانند ماتریس درهم‌ریختگی(Confusion Matrix) ، دقت (Precision)، بازخوانی (Recall) و اندازه‌گیری‌های دیگر می‌تواند در تشخیص بیش‌برازش و مشکلات مرتبط با استفاده از نرخ یادگیری بالا مفید باشد. این معیارها می‌توانند نشان دهنده عملکرد مدل در طول زمان و در داده‌های مختلف باشند.

استفاده از روش‌های ارزیابی خارجی: استفاده از روش‌های ارزیابی خارجی مانند اعتبارسنجی متقابل (Cross-validation) و آزمون در داده‌های نهان (Holdout Testing) می‌تواند در تشخیص بیش‌برازش و مشکلات مرتبطبقه بندی‌های ثابت می‌کند. برای این منظور، می‌توانید داده‌ها را به دو بخش تقسیم کنید: مجموعه آموزش و مجموعه ارزیابی. سپس مدل را با استفاده از مجموعه آموزش آموزش داده و عملکرد آن را روی مجموعه ارزیابی ارزیابی کنید. در صورتی که عملکرد مدل در مجموعه آموزش بسیار خوب بوده ولی در مجموعه ارزیابی نتایج ضعیفی ارائه می‌دهد، ممکن است مدل شما دچار مشکل بیش‌برازش شده باشد.

ب

استفاده از نرخ یادگیری پایین در آموزش مدل هوش مصنوعی نیز ممکن است باعث بروز مشکلات زیر شود:

عدم همگرایی: نرخ یادگیری پایین می‌تواند باعث کند شدن فرایند آموزش شود و مدل به صورت کامل همگرا نشود. در نتیجه، ممکن است مدل به دقت و عملکرد مطلوب نرسد.

عدم تطبیق مناسب: با استفاده از نرخ یادگیری پایین، ممکن است مدل به طور کامل به داده‌های آموزش تطبیق پیدا نکند و توانایی تعمیم‌پذیری آن کاهش یابد. به عبارتی، مدل ممکن است به داده‌های جدید یا ناشناخته نتواند به خوبی پاسخ دهد.

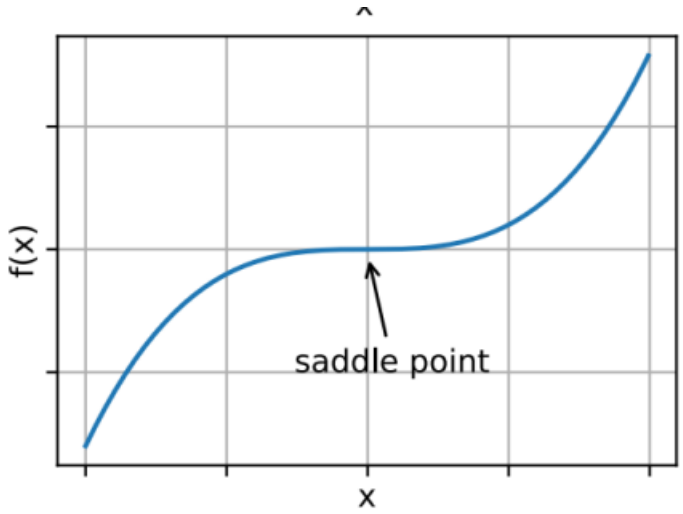
برای تشخیص مشکلات نرخ یادگیری پایین، می‌توان از روش‌های زیر استفاده کرد:

رصد تغییرات عملکرد: بررسی تغییرات عملکرد مدل در طول زمان می‌تواند نشان دهنده مشکلات ناشی از نرخ یادگیری پایین باشد. اگر دقت مدل به طور مداوم بهبود یا تغییر نکند، احتمالاً نرخ یادگیری پایین می‌باشد.

تحلیل نمودارها: بررسی نمودارهایی مانند نمودار تابع هزینه (loss function) در طول زمان می‌تواند روند آموزش مدل را نشان دهد. اگر تابع هزینه به طور کامل همگرا نشود یا به سرعت کاهش یابد، مشکلات نرخ یادگیری پایین ممکن است وجود داشته باشد.

پ

نقطه زینی (saddle point) یک نقطه در فضای تابع هزینه است که در آن گرادیان تابع در جهت‌های مختلف صفر می‌شود، اما نقطه نه به عنوان یک مینیمم محلی و نه به عنوان یک ماکزیمم محلی شناخته می‌شود. در این نقطه، سطوح منحنی تابع هزینه شبیه به یک سهمی‌گشتاپ است.



آدام: الگورتیم آدام به علت وجود momentum در آن، از نقاط زینی عبور میکند و متوقف نمیشود. همچنین به خاطر وجود RMSprop و momentum، خیلی سریع به نقطه مینیمم همگرا میشود و به دقت بسیار خوبی میرسد.

مزایای این الگوریتم، دقت بالا، همگرایی سریع، عدم گیر افتادن در نقاط مینیمم محلی و زینی است.

معایب این الگوریتم اما، این است که نیاز به تنظیم دو پارامتر B1 و B2 دارد. در صورتی که این پارامترها اشتباه تنظیم شوند نتایج اصلا مطلوب نخواند بود.

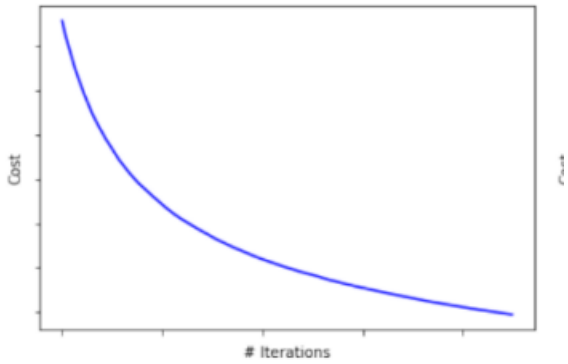
SGD: این الگوریتم هم به علت انتخاب نمونه‌های مختلف، به نسبت GD کم تر در نقاط زینی گیر میافتد. در GD تمام نمونه‌ها با هم میانگین گرفته میشوند و در صورتی که مدل در یک نقطه زینی گیر بیافتد، هیچ راه فراری ندارد. اما در SGD، احتمال این که در یک نقطه، مشتق تابع هزینه برای هر کدام از نمونه ها به صورت جدا جدا صفر باشد خیلی کم تر است. در واقع SGD کمی تصادفی تر عمل میکند که در نهایت آن را از نقطه زینی خارج میسازد.

مزایای این الگوریتم، عدم گیر افتادن در نقاط مینیمم محلی و نقاط زینی است. همچنین پارامتری ندارد که نیاز به تنظیم کردن داشته باشد.

معایب این الگوریتم، سرعت همگرای پایین تر نسبت به Adam است و همچنین به علت تصادفی بودن عملکرد SGD، مدل نمیتواند خیلی به نقطه مینیمم نزدیک شود و دور نقطه مینیمم نوسان میکند. یعنی دقت نهایی آن کمتر از Adam است.

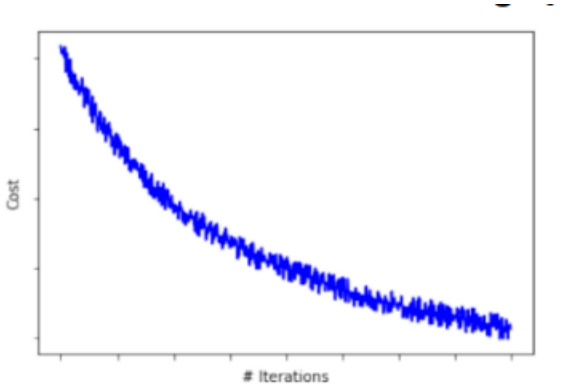
ت

نمودار سمت چپ نشان دهنده Batch Gradient Decent است.



در روش Batch Gradient Decent، تمام نمونه‌ها به مدل داده شده و از loss نهایی همگی میانگین گرفته میشود و روی میانگین مشتق گرفته میشود. در قدم بعد هم دقیقا همین نمونه دوباره به مدل داده میشوند و این روند تکرار میشود. به همین علت در BGD، همواره مقدار cost کاهش میابد. چون داده ها هیچگاه عوض نشده اند و اطمینان داریم که عملیات back propagation، باعث کم تر شدن cost داده های میشوند. یعنی نمودار cost همواره نزولی است.

اما نمودار سمت راست نشان دهنده mini Batch Gradient Decent است.



در روش mini BGD، هر بار به جای ورودی دادن تمام داده ها و میانگین گرفتن از loss آنها، تنها دسته ای کوچک از داده ها انتخاب میشوند و به مدل داده شده و از لاس همان دسته کوچک میانگین و مشتق گرفته میشوند. در این روش چون داده ها مرتبا تغییر میکنند، در هر قدم مدل سعی میکند که خود را با دسته ای جدید از داده ها مطابقت دهد که لزوما برابر با دسته قبلی نیستند. به همین علت مدل کمی حالت تصادفی به خود میگیرد و نمودار cost، کاملا نزولی نیست. اما در نهایت مدل همگرا میشود. این به خاطر ویژگی های مشترکی است که تمام دسته ها دارند و مدل هم باید همین ویژگی های مشترک را یاد بگیرد نه تفاوت های بین دسته ها.

Q2

پاسخ سوال در فایل زیپ موجود است.

Q3

الف

پاسخ در فایل زیپ موجود است.

ب

از نظر محاسباتی، در لایه های کانولوشنی دو بعدی، فیلتر دو بعدی است. فیلتر بر روی یک صفحه مسطح مانند عکس قرار میگیرد و محاسبات انجام میشوند. اما لایه کانولوشنی سه بعدی، فیلتری 3 بعدی دارد. مانند یک حجم روی فضا قرار میگیرد و محاسبات انجام میشوند. فضا میتواند یک جسم سه بعدی باید یا یک ویدیو با زمان باشد. اما از نظر کاربرد:

لایه کانولوشنی دو بعدی

لایه کانولوشنی دو بعدی در پردازش تصاویر و ویدئوها بسیار مؤثر است. این لایه با استفاده از عملیات کانولوشن، الگوها و ویژگی‌های مختلف را در تصاویر استخراج می‌کند. این لایه برای استخراج ویژگی‌های محلی از تصاویر استفاده می‌شود و می‌تواند مثلاً لبه‌ها، نقاط زینی، خطوط و الگوهای ساده را تشخیص دهد. لایه کانولوشنی دو بعدی معمولاً در شبکه‌های عصبی پیچشی (Convolutional Neural Networks) که برای بینایی ماشین، تشخیص الگو و دسته‌بندی تصاویر استفاده می‌شوند.

لایه کانولوشنی سه بعد:

لایه کانولوشنی سه بعدی در علاوه بر عملیات کانولوشن در دو بعد، زمان را نیز در نظر می‌گیرد. این لایه در پردازش ویدئوها و داده‌های سه بعدی مانند داده‌های حجمی (volumetric data) مورد استفاده قرار می‌گیرد. با در نظر گرفتن زمان، لایه کانولوشنی سه بعدی می‌تواند ویژگی‌های مکانی و زمانی را از داده‌ها استخراج کرده و درک بهتری از رفتار داده‌ها در طول زمان داشته باشد. این لایه برای وظایفی مانند تشخیص حرکت، شناسایی رفتار ویدئویی، پردازش سیگنال صوتی و تحلیل داده‌های سه بعدی در حوزه‌های پزشکی و شبکه‌های عصبی بازیابی اطلاعات (Neural Information Retrieval) استفاده می‌شود.

Q4

نمودار ها و نتایج در فایل نوتبوک موجود است. شرح کدها را در ادامه نوشته‌ام:

در ابتدا که import ها را انجام میدهیم:

```
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, Conv2D, MaxPool2D, Activation, Input, Flatten, Rescaling
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
```

برای دانلود دیتاست، از کتابخانه gdown استفاده کردم:

```
import gdown
url = 'https://drive.google.com/uc?id=1SCpVEdJ6_YOAcy2iW05ENlMh-OCcFz3P'
output = '/content/dataset.zip'

gdown.download(url, output, quiet=False)
```

سپس دیتاست را unzip کردن و فایل های اضافی را پاک کردم تا در پوشه اصلی فقط دو پوشه yes و no باقی بمانند:

```
! rm dataset.zip
! rm sample_data -r -f
```

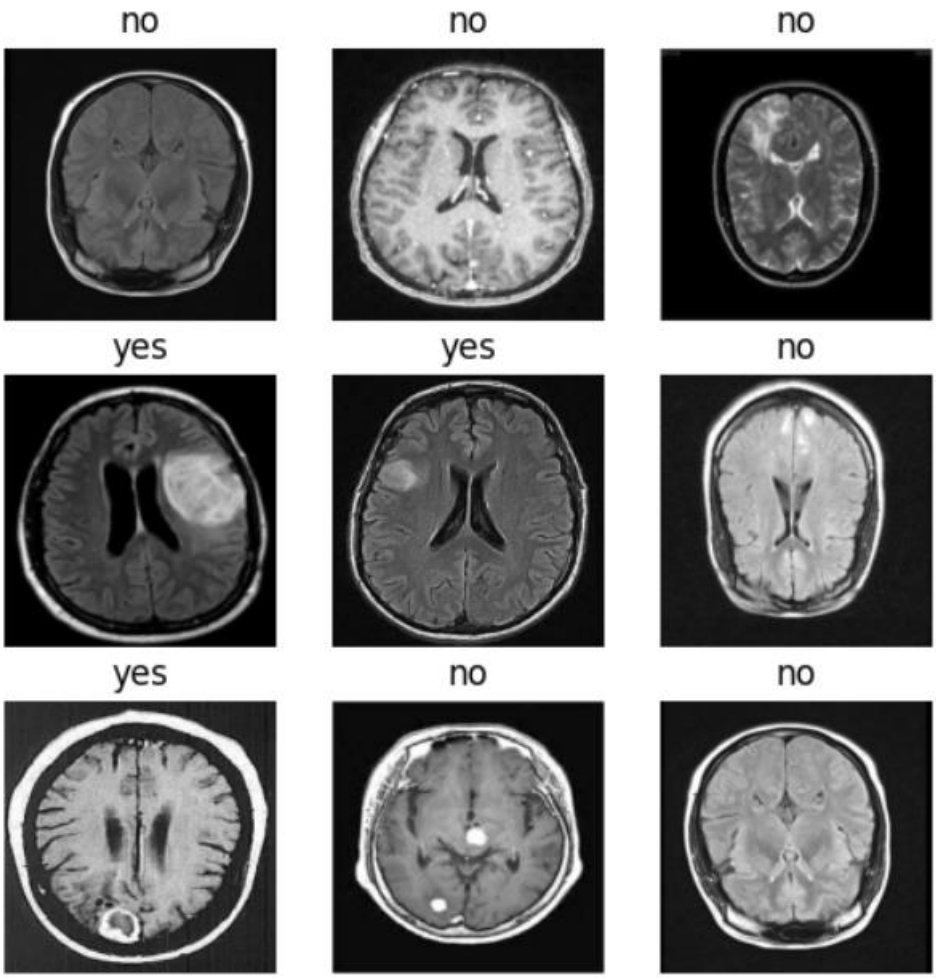
برای خواندن تصاویر از تابع موجود در لینک نوشته شده کمک گرفتم:

```
ds_train, ds_test = keras.utils.image_dataset_from_directory(
    '/content/',
    labels="inferred",
    label_mode="binary",
    class_names=['yes', 'no'],
    color_mode="grayscale",
    batch_size=32,
    image_size=(256, 256),
    shuffle=True,
    seed=27,
    validation_split=0.2,
    subset='both',
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False,
)
```

پارامتر labels مشخص میکند که لیبل ها به صورت خودکار انتخاب شوند و پارامتر label_mode مشخص میکند که لیبل ها به صورت 0 و 1 باشند. همچنین نام کلاس ها را هم بر اساس نام پوشه ها انتخاب کردم. در color_mode تصاویر را به صورت سیاه و سفید خواندم و همه تصاویر را به 256*256 ریسایز کردم. همچنین نسبت ولیدیشن و ترین را هم همانطور که خواسته شده بود مشخص کردم.

به کمک matplotlib، برخی از سمپل ها را چاپ کردم:

```python
classes = ['yes', 'no']

some_samples = ds_train.take(1)
iterator = some_samples.as_numpy_iterator()
images, labels = next(iterator)
fig = plt.figure(figsize=(6, 6))
for i in range(9):
  image, label = images[i], labels[i]
  fig.add_subplot(3, 3, i+1)
  plt.imshow(image, cmap='gray')
  plt.axis('off')
  plt.title(classes[int(label[0])])
```



سپس به سراغ ساخت مدل ها رفتم. برای ساختار مدل از ChatGPT کمک گرفتم.

مدل را با این ساختار به صورت Sequential تعریف کردم:

```
model = tf.keras.Sequential([
    Input(shape=(256, 256, 1)),
    Conv2D(16, (3, 3), activation='relu'),
    MaxPool2D((2, 2)),
    Conv2D(32, (3, 3), activation='relu'),
    MaxPool2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPool2D((2, 2)),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid')
])
```

خلاصه مدل:

```
model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_17 (Conv2D)          (None, 254, 254, 16)      160

 max_pooling2d_15 (MaxPooli  (None, 127, 127, 16)      0
 ng2D)

 conv2d_18 (Conv2D)          (None, 125, 125, 32)      4640

 max_pooling2d_16 (MaxPooli  (None, 62, 62, 32)        0
 ng2D)

 conv2d_19 (Conv2D)          (None, 60, 60, 64)        18496

 max_pooling2d_17 (MaxPooli  (None, 30, 30, 64)        0
 ng2D)

 flatten_5 (Flatten)         (None, 57600)             0

 dense_8 (Dense)             (None, 64)                3686464

 dense_9 (Dense)             (None, 1)                 65

=================================================================
Total params: 3709825 (14.15 MB)
Trainable params: 3709825 (14.15 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

از آدام برای بهینه‌سازی استفاده کردم و به خاطر دسته بندی دو کلاسه بودن مسئله، از loss باینری استفاده کردم:

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

سپس مدل را به اندازه 20 ایپاک آموزش دادم. مدل به خوبی نتیجه داد و دقت آن روی داده‌های تست به 95 درصد رسید:

```
history = model.fit(
    ds_train,
    epochs=20,
    validation_data=ds_test
)
```

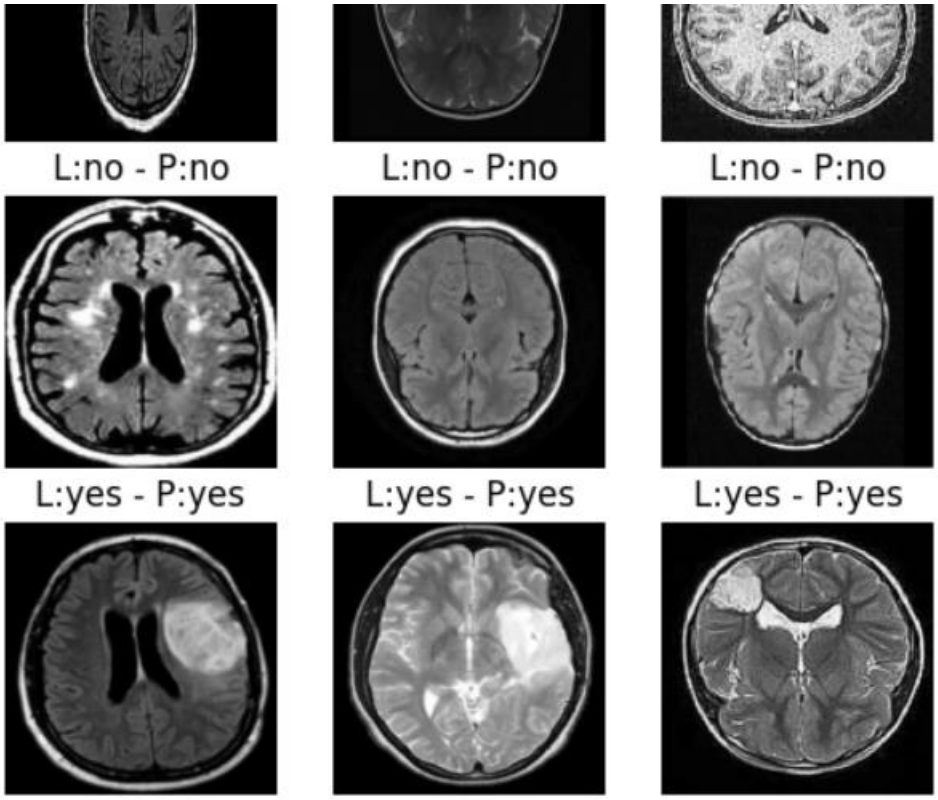
```
Epoch 18/20
75/75 [==============================] - 5s 66ms/step - loss: 0.2013 - accuracy: 0.9629 - val_loss: 0.2110 - val_accuracy: 0.9550
Epoch 19/20
75/75 [==============================] - 4s 45ms/step - loss: 0.1956 - accuracy: 0.9629 - val_loss: 0.1979 - val_accuracy: 0.9583
Epoch 20/20
75/75 [==============================] - 4s 45ms/step - loss: 0.1927 - accuracy: 0.9617 - val_loss: 0.2072 - val_accuracy: 0.9583
```

برای تست مدل هم ابتدا متریک ها را پرینت کردم:

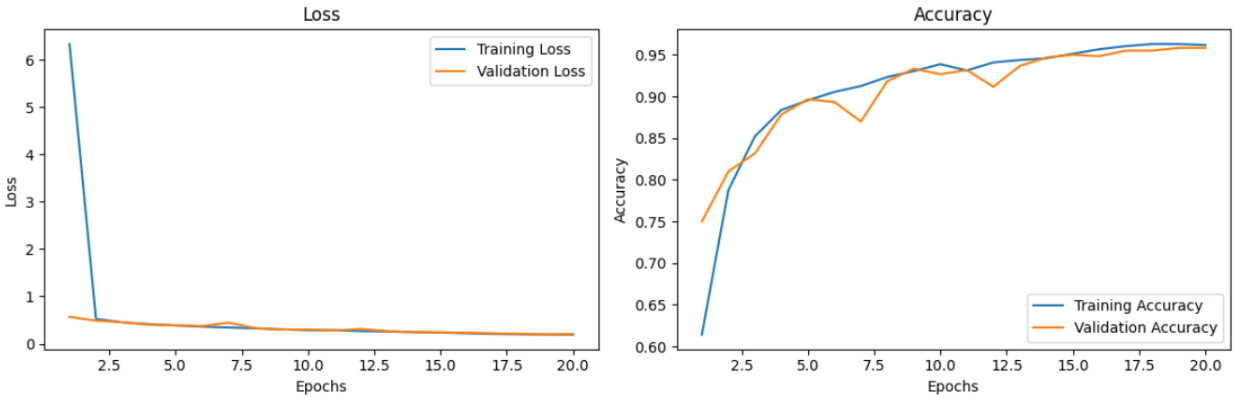
```
score = model.evaluate(ds_test)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
19/19 [==============================] - 1s 31ms/step - loss: 0.2072 - accuracy: 0.9583
Test loss: 0.20715492963790894
Test accuracy: 0.9583333134651184
```

سپس چند prediction هم انجام دادم و با لیبل اصلی مقایسه کردم:



در نهایت به کمک کدی که از چت GPT برای رسم نمودار ها گرفتم، نمودارها را چاپ کردم:



برای ساخت مدل به صورت فانکشنال، بار دیگه همون مدل رو به صورت فانکشنال تعریف کردم:

```
inputs = tf.keras.Input(shape=(256, 256, 1))
x = Conv2D(16, (3, 3), activation='relu')(inputs)
x = MaxPool2D((2, 2))(x)
x = Conv2D(32, (3, 3), activation='relu')(x)
x = MaxPool2D((2, 2))(x)
x = Conv2D(64, (3, 3), activation='relu')(x)
x = MaxPool2D((2, 2))(x)
x = Flatten()(x)
x = Dense(64, activation='relu')(x)
outputs = Dense(1, activation='sigmoid')(x)

model = tf.keras.Model(inputs=inputs, outputs=outputs)
```

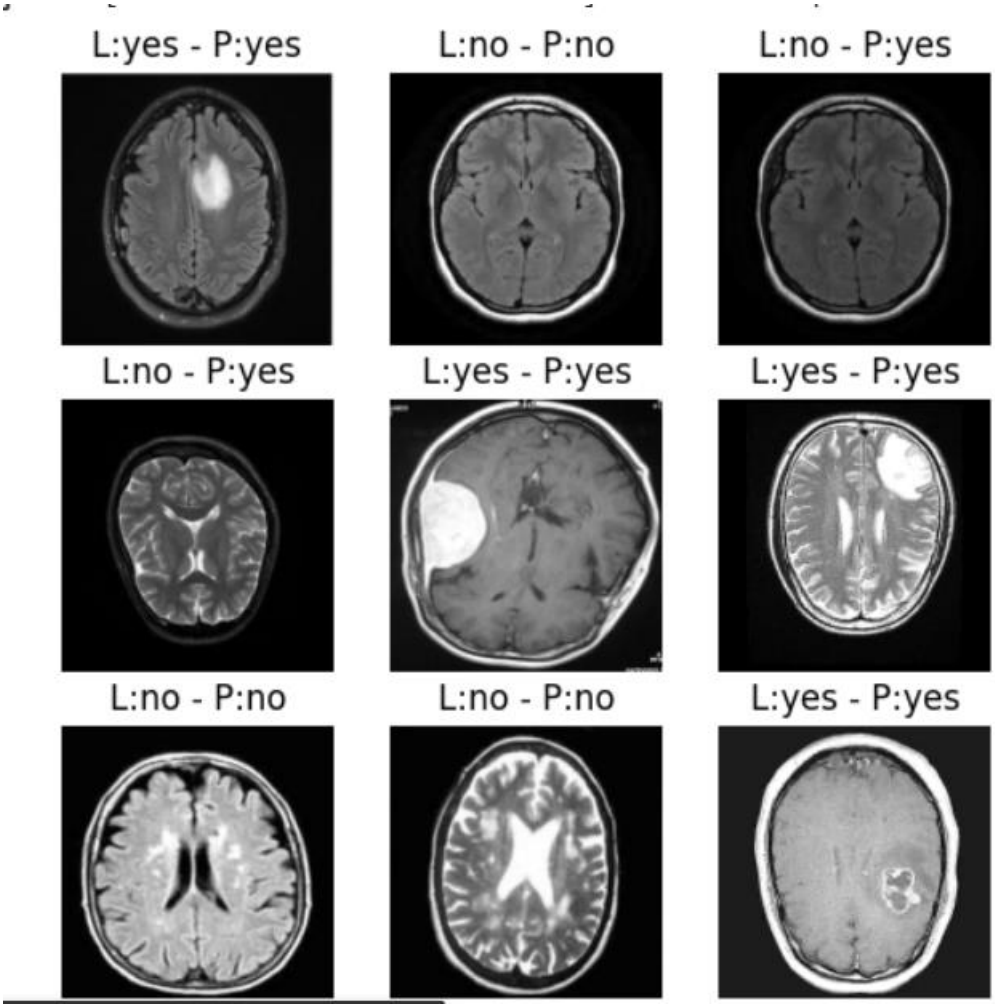
باقی کدها مشابه مدل قبلی بودند. تنها نتایج را برای مقایسه نشان میدهم.

نتیجه مدل فانکشنال روی داده‌های تست:

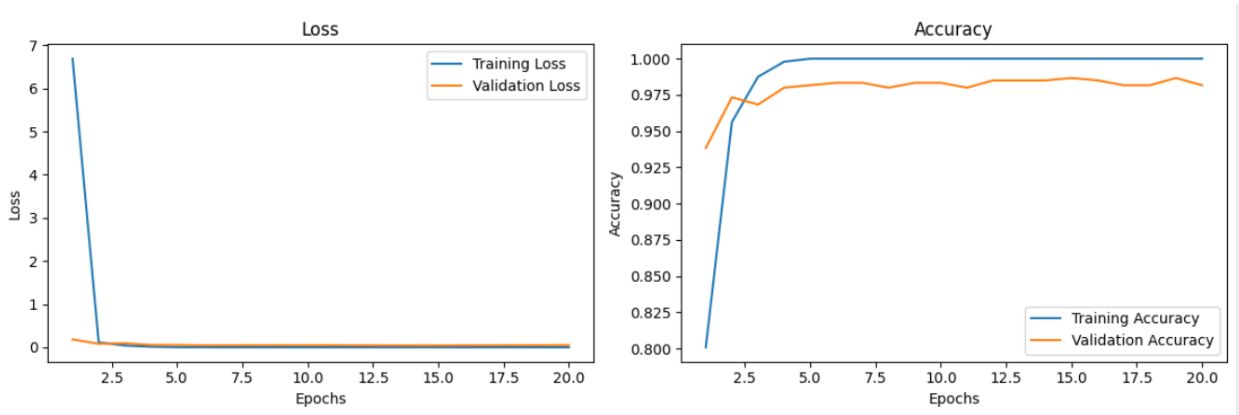
```
score = model.evaluate(ds_test)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
19/19 [==============================] - 1s 44ms/step - loss: 0.0503 - accuracy: 0.9817
Test loss: 0.05027418211102486
Test accuracy: 0.9816666841506958
```

پیشبینی مدل:



نمودارهای لاس و دقت:



Q5

لایه‌های کانولوشنی یکی از اجزای اصلی شبکه‌های عصبی کانولوشنی هستند و در پردازش تصویر و تشخیص الگو بسیار کارآمد هستند. در ادامه، مزایا و معایب این لایه‌ها را توضیح خواهم داد:

مزایا:

استخراج ویژگی محلی: لایه‌های کانولوشنی قادر به استخراج ویژگی‌های محلی از تصاویر هستند. با استفاده از عملیات کانولوشن، قادر به تشخیص الگوها و ویژگی‌های مختلف در تصویر می‌شوند. این ویژگی‌ها می‌توانند شامل لبه‌ها، خطوط، نقاط نورانی، و سایر الگوهای محلی باشند.

قابلیت انتقال ویژگی: لایه‌های کانولوشنی قابلیت انتقال ویژگی را دارند، به این معنی که می‌توانند اطلاعات مربوط به الگوها و ویژگی‌های مشابه را از تصویرهای آموزشی به تصاویر جدید منتقل کنند. این ویژگی باعث می‌شود که شبکه‌های کانولوشنی قابلیت عمومی بیشتری در تشخیص الگوها و اشیاء در تصاویر نمایش دهند.

کاهش تعداد پارامترها: لایه‌های کانولوشنی با استفاده از اشتراک پارامترها و استفاده از فیلترهای کانولوشنی، تعداد پارامترهای مورد نیاز برای شبکه را کاهش می‌دهند. این کاهش تعداد پارامترها باعث می‌شود که شبکه‌های کانولوشنی به صورت کلی سریعتر آموزش داده شوند و از تمرکز محاسباتی بیشتری برخوردار باشند.

معایب:

احساس محدودیت مکانی: لایه‌های کانولوشنی در تشخیص الگوها و ویژگی‌ها در تصاویر محدودیت مکانی دارند. به این معنی که برخی روابط بین ویژگی‌ها را نادیده می‌گیرند و تنها به ویژگی‌های محلی توجه می‌کنند. در برخی موارد، این محدودیت می‌تواند باعث از دست رفتن اطلاعات مهم در تصویر شود.

نیاز به حجم بالای داده آموزشی: لایه‌های کانولوشنی برای آموزش کافی نیاز به حجم بالایی از داده‌های آموزشی دارند. برعلاوه بر این، این لایه‌ها نیاز به تنظیمات پیشرفته‌تری نسبت به لایه‌های مستقیم دارند و ممکن است نیاز به تنظیم پارامترهای مختلفی مانند اندازه فیلترها، تعداد فیلترها، و مقدار پارامترهای رگولاریزه داشته باشند.

در نهایت، لایه‌های کانولوشنی به دلیل محدودیت مکانی خود و ممکن بودن از دست رفتن اطلاعات مهم در تصویر، ممکن است در مواردی که نیاز به ارتباطات دسته‌بندی شده بین ویژگی‌ها وجود دارد، کارایی کمتری نسبت به روش‌های دیگر داشته باشند.

به طور خلاصه، لایه‌های کانولوشنی به دلیل قابلیت استخراج ویژگی‌های محلی، قابلیت انتقال ویژگی، و کاهش تعداد پارامترها، در بسیاری از مسائل پردازش تصویر و تشخیص الگو موفق عمل می‌کنند. اما باید توجه داشت که در برخی موارد خاص، ممکن است محدودیت‌های مکانی و نیاز به حجم بالای داده آموزشی دقت و کارایی آن‌ها را کاهش دهد.

Q6

الف

هدف استفاده از فیتلرهای 1*1 ، کاهش تعداد تعداد فیچرمپ‌ها(لایه های به دست آمده از فیلترهای مختلف) و در عین حال حفظ ویژگی‌های مهم است.

این فیلتر عملیات یکپارچه (pointwise operation) را بر روی نقشه‌های ویژگی انجام می‌دهند، به این معنی که برای هر پیکسل در نقشه‌های ورودی، یک ترکیب خطی از ویژگی‌های آن پیکسل در لایه‌های مختلف را محاسبه می‌کنند. در نتیجه، تعداد کانال‌های خروجی کاهش می‌یابد و از این طریق می‌توان از بارزسازی ویژگی‌ها استفاده کرده و حجم محاسباتی را کاهش داد.

با تنظیم وزن‌های فیلترهای ۱×۱، می‌توان تاثیر و اهمیت ویژگی‌ها را تغییر داد. به عبارت دیگر، فیلترهای ۱×۱ قادرند ویژگی‌های مهم را برجسته کنند و ویژگی‌های کم اهمیت را کاهش دهند. این کار باعث بهبود کارایی شبکه و افزایش دقت در تشخیص ویژگی‌ها می‌شود.

ب

به طور کلی، نقشه ویژگی حاصل از اعمال فیلتر ۱×۱ نماینده‌ای از اطلاعات ترکیبی و تعامل کانال ها در نقشه ورودی است و می‌تواند اطلاعات جدیدی را ارائه کند که می‌تواند در وظایف مختلف شبکه‌های عصبی مانند تشخیص الگو، دسته‌بندی، یا استخراج ویژگی مفید باشد. به طور جزعی تر:

نقشه ویژگی حاصل از فیلتر ۱×۱ می‌تواند اطلاعات مختلفی را ارائه کند، از جمله:

اطلاعات کانال: نقشه ویژگی خروجی حاوی اطلاعاتی درباره ترکیب و تعامل بین کانال‌های ورودی است. با تنظیم وزن‌های فیلتر ۱×۱، می‌توان تاثیر و اهمیت کانال‌های ورودی را تغییر داد و اطلاعات مهم را برجسته کرد.

اطلاعات مکانی: نقشه ویژگی خروجی می‌تواند اطلاعات مکانی را نیز حاوی باشد. با توجه به محاسبات فیلتر ۱×۱، اطلاعات مکانی پیکسل‌ها نیز در نقشه ویژگی خروجی محفوظ می‌شود.

ترکیب ویژگی‌ها: فیلتر ۱×۱ قادر است ویژگی‌های مختلف را ترکیب کند و ویژگی‌های جدیدی را تولید کند. این ترکیبات خطی از ویژگی‌های ورودی می‌توانند نقشه ویژگی خروجی را بهبود ببخشند و ویژگی‌های مهم را برجسته کنند.

پ

تعداد کانال‌ها:

نقشه ویژگی حاصل از فیلتر ۱×۱ تعداد کانال‌ها را کاهش می‌دهد. در صورتی که تصویر اصلی یا فیلترهای دیگر دارای بیش از یک کانال باشند، فیلتر ۱×۱ باعث کاهش تعداد کانال‌ها می‌شود و نقشه ویژگی خروجی فقط یک کانال خواهد داشت.

اطلاعات مکانی:

نقشه ویژگی حاصل از فیلتر ۱×۱ اطلاعات مکانی را حفظ می‌کند. این فیلتر برای هر پیکسل در نقشه ورودی، یک ترکیب خطی از ویژگی‌های آن پیکسل را محاسبه می‌کند و اطلاعات مکانی پیکسل‌ها در نقشه ویژگی خروجی حفظ می‌شود.

ترکیب و تعامل ویژگی‌ها:

فیلتر ۱×۱ قادر است ویژگی‌های مختلف را ترکیب کند و ویژگی‌های جدیدی را تولید کند. با تنظیم وزن‌های فیلتر ۱×۱، می‌توان تاثیر و اهمیت کانال‌ها و ویژگی‌ها را تغییر داد و نقشه ویژگی خروجی را بهبود بخشید.

ت

فیلتر ۱×۱ در مدل‌های عمیق شبکه‌های عصبی به عنوان یک عنصر اصلی استفاده می‌شود و در بسیاری از معماری‌ها و مدل‌ها استفاده شده است. در زیر، مدل‌های معروفی را ذکر می‌کنم که از فیلتر ۱×۱ در ساختار خود استفاده کرده‌اند:

InceptionNet

مدل InceptionNet که توسط Google توسعه داده شده است، از فیلتر ۱×۱ در معماری خود استفاده می‌کند. این فیلتر در این مدل برای کاهش تعداد کانال‌ها و افزایش ابعاد فضایی استفاده می‌شود.

ResNet

مدل ResNet نیز از فیلتر ۱×۱ در ساختار خود استفاده می‌کند. این فیلتر در این مدل برای کاهش تعداد کانال‌ها و کنترل ابعاد فضایی استفاده می‌شود.

MobileNet

مدل MobileNet یک مدل سبک و قابل استفاده در دستگاه‌های محدود منابع است. این مدل از فیلتر ۱×۱ برای کاهش تعداد کانال‌ها و افزایش کارایی در محیط‌های محدود استفاده می‌کند.

DenseNet

مدل DenseNet از فیلتر ۱×۱ برای افزایش تعامل بین لایه‌ها و کاهش تعداد پارامترها استفاده می‌کند.

SqueezeNet

مدل SqueezeNet نیز یک مدل سبک است که برای استفاده در دستگاه‌های با منابع محدود طراحی شده است. این مدل از فیلتر ۱×۱ برای کاهش تعداد کانال‌ها و افزایش کارایی استفاده می‌کند.

این فقط چند نمونه از مدل‌هایی است که از فیلتر ۱×۱ استفاده کرده‌اند. در واقع، این فیلتر در بسیاری از مدل‌ها به عنوان یک ابزار مهم برای کاهش پیچیدگی و حجم مدل، افزایش کارایی و کنترل ابعاد فضایی استفاده می‌شود.

ث

بله، در برخی حالات استفاده از فیلترهای ۱×۱ ممکن است مفید نباشد. دلایل زیر می‌توانند منجر به عدم مفید بودن استفاده از فیلترهای ۱×۱ در برخی مدل‌ها باشند:

۱ .عدم نیاز به کاهش تعداد کانال‌ها: فیلتر ۱×۱ اصطلاحاً برای کاهش تعداد کانال‌ها استفاده می‌شود. اگر در مدلی که در حال طراحی استفاده می‌شود، نیازی به کاهش تعداد کانال‌ها وجود نداشته باشد، استفاده از فیلتر ۱×۱ ممکن است مفید نباشد.

۲ .افزایش پیچیدگی محاسباتی: استفاده از فیلتر ۱×۱ ممکن است باعث افزایش پیچیدگی محاسباتی شود، زیرا هر کانال از ورودی با همه کانال‌های خروجی تعامل دارد و در نتیجه تعداد عملیات مورد نیاز برای هر نقطه افزایش می‌یابد. در برخی موارد، این افزایش پیچیدگی محاسباتی می‌تواند منجر به کاهش کارایی و سرعت مدل شود.

۳ .اثرات غیرمطلوب بر روی اطلاعات مکانی: استفاده از فیلتر ۱×۱ می‌تواند منجر به از دست رفتن اطلاعات مکانی در ورودی شود، زیرا فیلتر ۱×۱ تنها یک نقطه را در نظر می‌گیرد و ارتباطات مکانی بین نقاط را نادیده می‌گیرد. در برخی مسائل، نقش اطلاعات مکانی بسیار مهم است و استفاده از فیلتر ۱×۱ ممکن است این اطلاعات را از دست بدهد.

بنابراین، استفاده از فیلترهای ۱×۱ باید با توجه به نیازها و ویژگی‌های مسئله و مدل مورد بررسی قرار گیرد. در برخی موارد، استفاده از فیلتر ۱×۱ می‌تواند مفید باشد، اما در برخی حالات دیگر ممکن است بهینه نباشد.

ج)

کدها در نوتبوک q6_lastpart.ipynb موجود است. شرح کد ها:

دیتاست را به صورت تصادفی به این شکل ساختم:

```
x = np.random.random((100, 30, 30, 3))
y = np.array([np.mean(x[i]) > 0.5 for i in range(x.shape[0])])
```

یعنی صد داده که مقدار x آنها، یک آرایه 3*30*30 تصادفی است(نمایانگر یک تصویر 30*30 rgb) و y آنها بزرگتر یا کوچکتر بودن میانگین اعداد از 0.5 است.

```
keras.utils.set_random_seed(27)
model = keras.Sequential()
model.add(Input(shape=(30, 30, 3)))
model.add(Conv2D(filters=16, kernel_size=5, activation='relu'))
model.add(Conv2D(filters=1, kernel_size=1, activation='relu'))
model.add(Flatten())
model.add(Dense(units=1, activation='sigmoid'))
```

مدل را به این صورت ساختم. میبینیم که بعد یک یک لایه کانولوشنی عادی با 16 فیلتر، یک لایه کانولوشنی 1*1 قرار داده ام و باقی مدل را تکمیل کرده ام.

```
model.summary()
```

```
Model: "sequential_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_24 (Conv2D)          (None, 26, 26, 16)        1216

 conv2d_25 (Conv2D)          (None, 26, 26, 1)         17

 flatten_9 (Flatten)         (None, 676)               0

 dense_11 (Dense)            (None, 1)                 677

=================================================================
Total params: 1910 (7.46 KB)
Trainable params: 1910 (7.46 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

در خلاصه مدل به خوبی میتوانیم ببینیم که فیتلر 1*1 چگونه عمل میکند. ورودی این لایه، یک تصویر 26*26 با 16 کانال مختلف است. اما در این لایه، تمام کانال ها با یک ترکیب خطی به هم وصل و تبدیل به یک کانال میشوند. به همین علت ابعاد width و height ورودی تغییر نکرده و در خروجی هم همان 26*26 است اما تعداد کانال ها از 16 به 1 رسیده است.

نمیدانم آموزش مدل هم جزو اهداف مسئله بوده یا نه اما مدل پس از 30 ایپاک به این دقت رسید:

```
4/4 [==============================] - 0s 21ms/step - loss: 0.3937 - accuracy: 0.9500
Epoch 29/30
4/4 [==============================] - 0s 20ms/step - loss: 0.3738 - accuracy: 0.9600
Epoch 30/30
4/4 [==============================] - 0s 22ms/step - loss: 0.3548 - accuracy: 0.9600
<keras.src.callbacks.History at 0x7c9fe8953610>
```

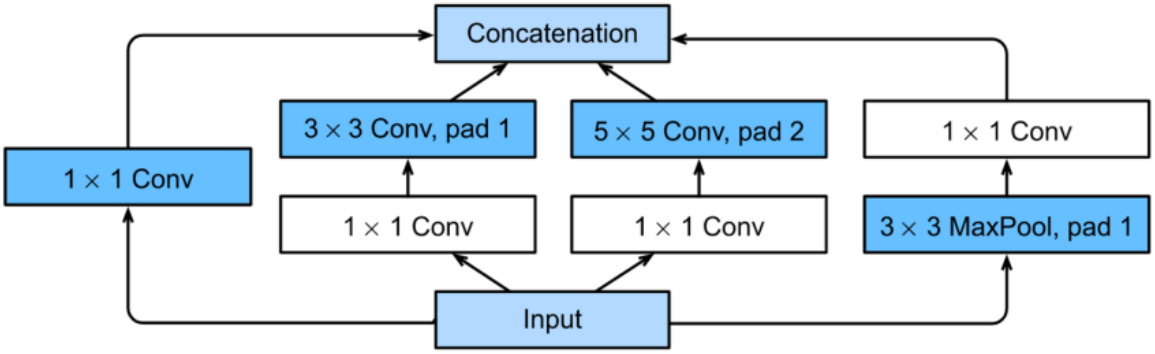
Q7

کد های این سوال در نوتبوک HW_Q7_Inception در فایل زیپ موجود میباشد. ابتدا به سوالات تشریحی پاسخ میدهم و بعد کدها را شرح میدهم. همچنین ساختار مدلی که ایجاد کردم هم در تصویر Q7_model_summary.png قابل مشاهده است.

الف

ساختار ماژول Inception که توسط گوگل ارائه شد، به این صورت است که به جای انجام عملیات های کانولوشن به صورت لایه به لایه، چند کانولوشن را روی یک ورودی میزنیم و نتایج را به هم concatenate میکنیم. البته فقط کانولوشن نیست ممکن است MaxPool هم استفاده کنیم اما هدف اصلی همین Concat کردن نتایج است:

۷



با استفاده از ماژولInception ، شبکه قادر است تا اندازه‌ها و ابعاد مختلفی از ویژگی‌ها را در نظر بگیرد. این ماژول با ترکیبی از لایه‌های پیچشی با اندازه‌های مختلف، امکان استخراج ویژگی‌های مختلف را فراهم می‌کند. به عنوان مثال، با استفاده از فیلترهای ۱×۱، ماژول Inceptionمی‌تواند تعداد کانال‌ها را کاهش دهد و با استفاده از فیلترهای ۳×۳ و ۵×۵، ویژگی‌های مکانی و سطح بالا را استخراج کند.

ترکیب خروجی‌های مختلف این ماژول، به شبکه امکان می‌دهد تا ویژگی‌های متنوع را با هم ترکیب کند و اطلاعات پیچیده‌تری را از تصاویر استخراج کند.

ب

با بزرگ تر کردن گام، میدان تاثیر لایه ها بیشتر میشود. بنابراین در هر پیکسل از لایه های آخر، اطلاعاتی از محدوده بزرگی از تصویر ورودی وجود دارد. مثلا یک پیکسل از مپ خروجی مشخص میکند که در یک محدوده بزرگ از تصویر اصلی، یک تصویر ماشین وجود دارد یا نه.

همچینین بزرگ تر کردن گام باعث کاهش اندازه فیچر مپ میشود. این کار به کاهش هزینه محاسبات کمک زیادی میکند.

البته گاهی بزرگ تر کردن گام باعث از دست رفتن برخی اطلاعات تصویر میشود که بسته به شرایط میتوان در مورد استفاده یا عدم استفاده از گام بزرگ، تصمیم گرفت.

پ

لایه های کانولوشنی در مدل پیاده سازی شده ام، نقش اصلی را دارند. عملیات هایی که این لایه ها انجام میدهند عبارتند از:

تشخیص الگوها: با استفاده از فیلترها، لایه کانولوشنی قادر است الگوها و قابلیت‌های مختلف را در تصویر تشخیص دهد. هر فیلتر در لایه کانولوشنی به دنبال یک الگو یا ویژگی خاص در تصویر است، مانند لبه‌ها، خطوط، نقاط روشن و تاریک و ... . با استفاده از فیلترهای مختلف و متنوع، لایه کانولوشنی می‌تواند ویژگی‌های پیچیده‌تر و سطوح بالاتر را شناسایی کند.

تجمیع و تلخیص اطلاعات: با استفاده از عملیات پولینگ (Pooling)، لایه کانولوشنی می‌تواند اطلاعات مهم را تلخیص کرده و ابعاد خروجی را کاهش دهد. عملیات پولینگ معمولاً با استفاده از ماکسیمم (MaxPooling) یا میانگین (AveragePooling)، بخشی از اطلاعات را انتخاب کرده و به عنوان ویژگی‌های تجمیع شده ارائه می‌دهد. این کاهش ابعاد و تلخیص اطلاعات موجب کاهش تعداد پارامترها و محاسبات مورد نیاز و همچنین افزایش مقاومت شبکه در برابر تغییرات مکانی می‌شود.

استخراج ویژگی سلسله مراتبی: لایه‌های کانولوشنی در یک شبکه عصبی کانولوشنی به صورت سلسله مراتبی قرار می‌گیرند. در لایه‌های ابتدایی، ویژگی‌های ساده‌تر مانند لبه‌ها و خطوط استخراج می‌شوند و در لایه‌های بعدی، ویژگی‌های پیچیده‌تر و سطوح بالاتر مانند الگوها و شکل‌های معینت‌دار تشخیص داده می‌شوند. این استخراج ویژگی سلسله مراتبی امکان توصیف و شناسایی اجزای مختلف تصویر را بهبود می‌بخشد و به شبکه کمک می‌کند تا اطلاعات مهم را به صورت سلسله مراتبی و ساختارمند استخراج کند.

حالا کدهای نوتبوک را توضیح میدهم:

ابتدا کتابخانه های لازم را ایمپورت کردم:

```
import keras
from keras.layers import *
import matplotlib.pyplot as plt
```

سپس دیتاست را لود کردم و ابعاد آن را مشاهده کردم:

```
[50] (x_train, y_train), (x_test, y_test) = keras.datasets.cifar10.load_data()
```

```
[51] print(x_train.shape)

     (50000, 32, 32, 3)
```

و برخی نمونه ها را برای درک بهتر دیتاست چاپ کردم:

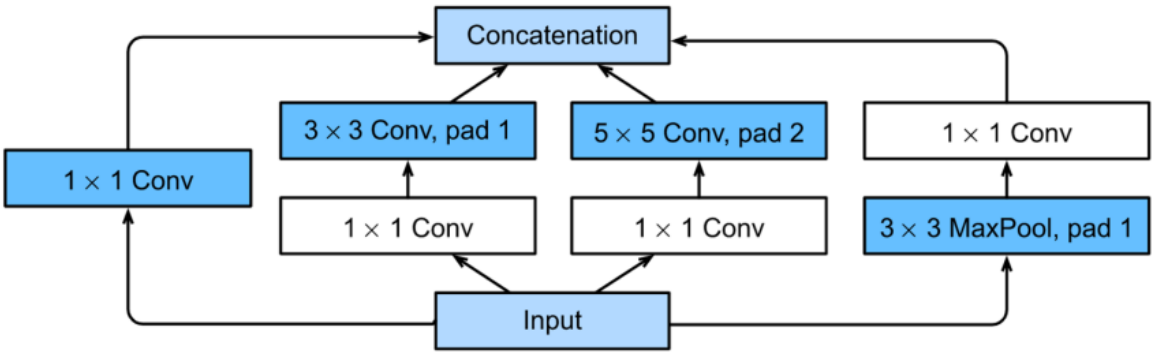
```
classes = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']

fig = plt.figure(figsize=(6, 6))
for i in range(9):
  image, label = x_train[i], y_train[i]
  fig.add_subplot(3, 3, i+1)
  plt.imshow(image)
  plt.title(classes[label[0]])
  plt.axis('off')
```

البته مشکلی که دیتاست داشت این بود که لیبل ها به صورت integer از 0 تا 9 بودند. با استفاده از تابع to_categorical در keras، لیبل ها را به صورت one_hot در اوردم:

```
] y_train = keras.utils.to_categorical(y_train)
  y_test = keras.utils.to_categorical(y_test)
```

ابتدا بلوک Inception را مطابق اسلایدها و به کمک ChatGPT ایجاد کردم:

```
def Inception(x, filters):
  x11 = Conv2D(filters=filters[0], kernel_size=1, padding='same', activation='relu')(x)

  x33 = Conv2D(filters=filters[1], kernel_size=1, padding='same', activation='relu')(x)
  x33 = Conv2D(filters=filters[2], kernel_size=3, padding='same', activation='relu')(x33)

  x55 = Conv2D(filters=filters[3], kernel_size=1, padding='same', activation='relu')(x)
  x55 = Conv2D(filters=filters[4], kernel_size=5, padding='same', activation='relu')(x55)

  xpool =  MaxPool2D(pool_size=(3, 3), strides=(1, 1), padding='same')(x)
  xpool = Conv2D(filters=filters[5], kernel_size=1, padding='same', activation='relu')(xpool)

  output = concatenate([x11, x33, x55, xpool], axis=3)
  return output
```



و بعد مدلم را ایجاد کردم که از دو بلوک Inception و در انتها هم دولایه Dense تشکیل شده. برای انتخاب ساختار مدل از ChatGPT کمک گرفتم:

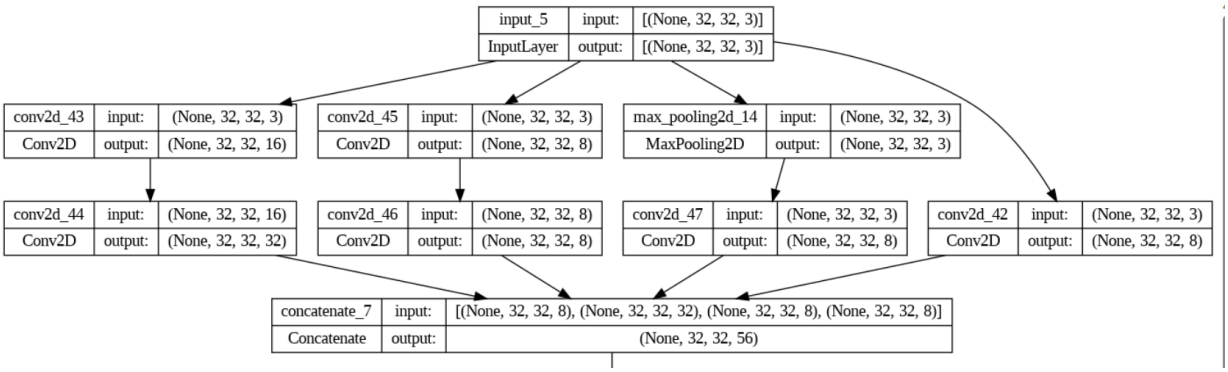
```
inputs = Input(shape=(32, 32, 3))

x = Inception(inputs, [8, 16, 32, 8, 8, 8])
x = MaxPool2D()(x)
# x = Dropout(0.25)(x)

x = Inception(x, [16, 32, 32, 16, 16, 16])
x = MaxPool2D(pool_size=(2, 2))(x)
# x = Dropout(0.25)(x)

x = Flatten()(x)
x = Dense(100, activation='relu')(x)
# x = Dropout(0.5)(x)
outputs = Dense(10, activation='softmax')(x)
```

البته در ابتدا ChatGPT لایه Dropout هم پیشنهاد داده بود و تعداد فیلترها هم خیلی بزرگ تر در نظر گرفته بود. اما با اجرای train مدل، متوجه شدم که به علت زیاد بودن پارامترها(حدود 16 میلیون) مدل نمیتواند آموزش ببیند. برای همین لایه های Dropout را حذف کردم و همچنین تعداد فیلترها هم خیلی کمتر کردم.

خلاصه مدل:

```
=================================
Total params: 539742 (2.06 MB)
Trainable params: 539742 (2.06 MB)
Non-trainable params: 0 (0.00 Byte)
```

ساختار مدل در تصویر Q7_model_summary در فایل زیپ موجود است. برای این که نشان بدهم از ماژول Inception استفاده کرده ام یک قسمت از آن را نشان میدهم:



همانطور که میبینید این قسمت در واقع یک ماژول اینسپشن است.

در نهایت مدل را Train کردم و در ایپاک 13 ام دقت آموزش به 80 درصد رسید و پس از 20 ایپاک دقت 86 درصد بود:

```
1563/1563 [==============================] - 13s 8ms/step - loss: 0.5958 - accuracy: 0.7907
Epoch 13/20
1563/1563 [==============================] - 13s 8ms/step - loss: 0.5653 - accuracy: 0.8024
Epoch 14/20
1563/1563 [==============================] - 13s 8ms/step - loss: 0.5259 - accuracy: 0.8149
Epoch 15/20
1563/1563 [==============================] - 12s 8ms/step - loss: 0.4938 - accuracy: 0.8278
Epoch 16/20
1563/1563 [==============================] - 12s 8ms/step - loss: 0.4626 - accuracy: 0.8370
Epoch 17/20
1563/1563 [==============================] - 12s 8ms/step - loss: 0.4361 - accuracy: 0.8444
Epoch 18/20
1563/1563 [==============================] - 13s 8ms/step - loss: 0.4086 - accuracy: 0.8563
Epoch 19/20
1563/1563 [==============================] - 13s 8ms/step - loss: 0.3915 - accuracy: 0.8619
Epoch 20/20
1563/1563 [==============================] - 13s 8ms/step - loss: 0.3682 - accuracy: 0.8681
```

در نهایت، پیشبینی مدل برای برخی تصاویر را میبینیم:

```
from numpy import argmax

classes = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
predictions = model.predict(x_train[:10])
fig = plt.figure(figsize=(10, 6))
for i in range(9):
  image, label, predict= x_train[i], y_train[i], predictions[i]
  fig.add_subplot(3, 3, i+1)
  plt.imshow(image)
  plt.title(f'L:{classes[argmax(label)]} - P:{classes[argmax(predict)]}')
  plt.axis('off')
```

قبل از رسیدن به این مدل، مدل های دیگری هم تست شد اما دقتشان کم تر بود. در نهایت این مدل بهترین دقت را داشت.