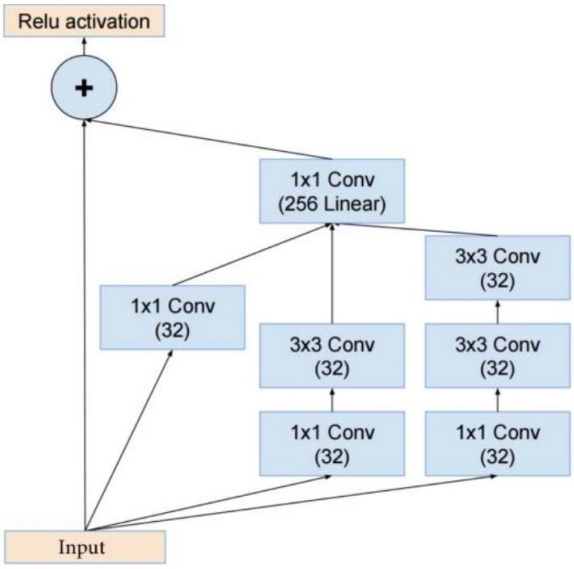
نیکی مجیدی فر 98522382

تمرین سری ششم

سوال 1)پیوست در فایل تمرین

سوال 2)



Parameters : (((filtersize * filtersize * channels) +1) * filternumber

Layer1 :Conv1A: 1*1 Conv(32) : ((1*1*3)+1)*32 = 128

Layer1:Conv1B : 1*1 Conv(32) : ((1*1*3)+1)*32 = 128

Layer1:Conv1C: 1*1 Conv(32) : ((1*1*3)+1)*32 = 128

Layer1 parameters = 384

Layer2:Conv2A(input is Conv1A) : ((3*3*32)+1) * 32 = 9248

Layer2:Conv2B(input is Conv1B) : ((3*3*32)+1) * 32 = 9248

Layer2params = 18496

Layer3:Conv3A(input is Conv2A) : ((3*3*32)+1) *32 = 9248

Layer3params : 9248

Layer4 = depth = 32*3 (there are 32 channels from each of inputs) :((1*1 * 96)+1) *256 = 24832

All parameters : 384+18496+9248 + 24832 = 52960

Receptive field :

برای لایه اول، تنها یک کانولوشن 1*1 انجام شده است در نتیجه برای این لایه 1*1 است. در لایه دوم بعد از یک کانولوشن 1*1، کانولوشن 3*3 انجام میشود در نتیجه برای این لایه 3*3 است. در لایه سوم، بعد از یه کانولوشن 3*3 ، یک کانولوشن دیگر انجام شده که معادل کانولوشن 5*5 است در نتیجه receptive fieldاین مدل 5*5 است.

ب)

A)
Conv2D(filters=16, kernel_size=(3, 3), padding='valid')
Conv2D(filters=32, kernel_size=(3, 3) , padding='valid')

B)
LocallyConnected2D(filters=16, kernel_size=(3, 3) , padding='valid')
LocallyConnected2D (filters=32, kernel_size=(3, 3) , padding='valid')

A)Parameters:

Conv2d1 : ((3*3 *3) +1) * 16 = 448

Conv2d2:((3*3*16) +1)*32 = 4640 ➔ all : 4640 + 448 : 5088

Receptive field : 2 convolutions : 3*3 ➔ 3*3 : as behave as 5*5 so : 5*5

B)locallyconnected2d:

لایه LocallyConnected2D مشابه لایه Conv2D کار می کند، با این تفاوت که وزن ها به اشتراک گذاشته نمی شوند، یعنی مجموعه فیلترهای متفاوتی در هر پچ مختلف ورودی اعمال می شود.

Parameters:

LC2d1 :((3*3*3) +1)*16*(n-2)*(n-2) = 488(n-2)(n-2)

LC2d2 : ((3*3*16) +1)*32 *(n-4)*(n-4) : 4640(n-4)(n-4)

Sum : 488(n-2)(n-2) + 4640(n-4)(n-4)

به طور واضح تعداد پارامتر های این بخش بیشتر است.

Receptive field : 2 convolutions : 3*3 ➔ 3*3 : as behave as 5*5 so : 5*5

سوال 3)

الف)

ابتدا داده هارا از دیتاست cifar دریافت کرده و آن ها را بین 0-1 در می آوریم و از روش one-hot encoding استفاده می کنیم.

```
def preproccess(data ,label):
  data = data.astype("float32") / 255
  data = np.expand_dims(data, -1)
  label = keras.utils.to_categorical(label, 10)
  return data , label
x_train , y_train = preproccess(x_train , y_train)
x_val , y_val = preproccess(x_val ,y_val)
```

سپس مدل خود را تعریف می کنیم.

مدل تعریف شده یک مدل sequential است که از دو لایه کانولوشنی تشکیل شده است.

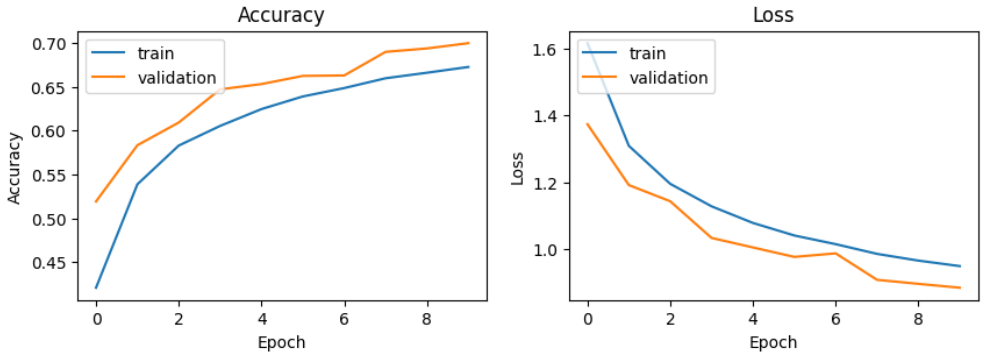
```
shape = (32,32,3)
model = keras.Sequential(
    [
        keras.Input(shape=shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(10, activation="softmax"),
    ]
)
```

مدل را با optimizer adam و loss function cross-entropy کامپایل می کنیم.

```
model.compile(loss="categorical_crossentropy", optimizer="adam",
metrics=["accuracy"])
```

مدل را فیت کرده و داده هار را آموزش می دهیم.

```
history = model.fit(x_train, y_train, epochs= 10,
validation_data=(x_val, y_val), batch_size= 64)
```



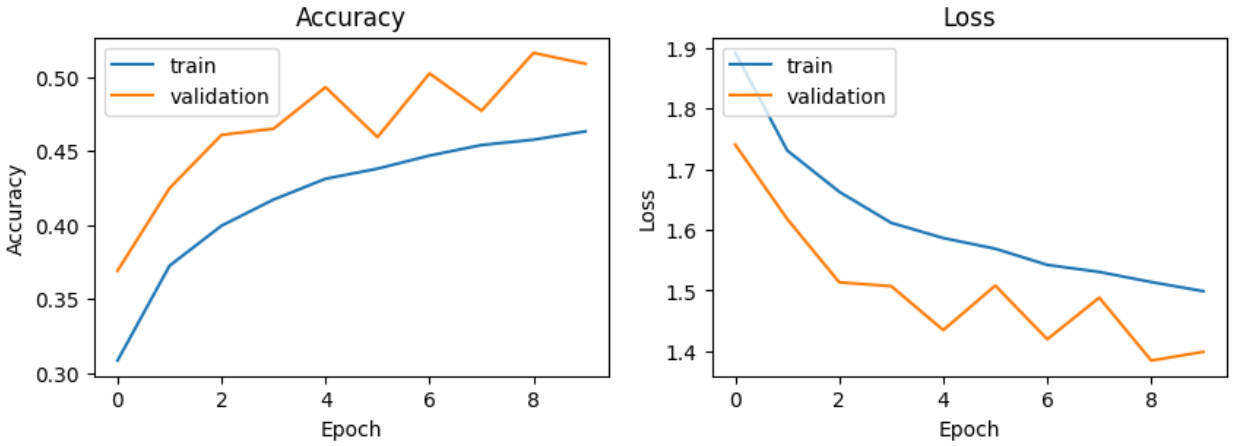
به ظور کلی، روند داده های train و validation شبیه به یک دیگر بوده و در نمودار accuracy داده های validationبا سرعت خوبی رشد کرده است و نمودار ضرر هم هر دو داده ی train loss شبیه به یک دیگر عمل کرده اند.

ب)

برای بخش داده افزایی ، همان طور که در لینک کمکی نیز توضیح داده شده بود، میتوان این تکنیک ها را به مدل آموزشی اضافه کرد. برای مثال در این سوال از دو تکنیک randomflipو randomrotationاستفاده میکنیم.

```
model = keras.Sequential(
    [
        keras.Input(shape=(32,32,3)),
        layers.RandomFlip("horizontal_and_vertical"),
        layers.RandomRotation(0.2),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(10, activation="softmax"),
    ]
)
```

در نهایت این مدل را کامپایل می کنیم و آن را آموزش میدهیم(مانند قسمت بالا)



در این نمودار همان طور که مشاهده می کنید. رفتار داده های validation در اواخر اموزش به صورت سینوسی عمل می کند اما به طور کلی فاصله آن از داده های train بیشتر است. یعنی متوجه میشویم که با داده افزایی داده های validation نتیجه مطلوب تری دارد.

ج) با مقایسه دو نمودار ها، به این موضوع در میابیم که به ظوز کلی میزلن یادگیری در حالت اول درصد بالا تری دارد(سرعت accuracy در نمودار اول بیشتر است و در انتها میزان خطای آن کم تر است. اما متوجه می شویم که در داده افزایی نتیجه ما بر روی داده های validationبهتر است و همجنین فاصله خط نموداار های آن از داده های train بهتر است در نتیجه می توان فهمید که با کمک داده افزایی اگر چه شیب سرعت یادگیزی و تابع خطا کمی کم تر است اما در نهایت داده های validation نسبت به داده های trainنتیجه ی بهتری دارند و در هیچ کدام از نمودار ها overfit یا underfitting نداریم. همچنین data augmentation یکی از روش های مرسوم است که می توان از overfiting در مدل سازی جلوگیری نماییم

د)

انتقال داده به کمک شبکه resnet و با داده های .imagenet

ابتدا نیاز داریم که داده هایمان مطابق با شبکه ی resnet پردازش کنیم در نتیجه خواهیم داشت:

```
(x_train, y_train), (x_val, y_val) = cifar10.load_data()
x_train = keras.applications.resnet50.preprocess_input(x_train)
x_val = keras.applications.resnet50.preprocess_input(x_val)
y_train = keras.utils.to_categorical(y_train,  10)
y_val = keras.utils.to_categorical(y_val,  10)
```

سپس نیاز داریم که مدل از پیش پردازش شده مان را تعریف کنیم.

ورودی داده را 3*224 *224می دهیم و مقدار imclude_topرا فالس می دهیم که مقدار ورودی دلبخواه خودمان را بدهیم.

```
#define model
input = keras.Input(shape = (224,224,3))
resnet = keras.applications.ResNet50(weights = "imagenet" ,
include_top = False ,input_tensor = input)
#freezing
resnet.trainable = False
```

حال یک شبکه از پیش پردازش شده با تعدادی پارامتر از داده های imagenet داریم. برای این که این داده هارا داشته باشیم و شبکه را تا جایی که می توانیم از تغییر این وزن ها و پارامتر ها جلو گیری کنیم و همچنین در محاسبات و هزینه ها صرفه جویی شود. کل شبکه حالت trainableبودن آن False می شود یا به اصطلاح freezeمی شود.

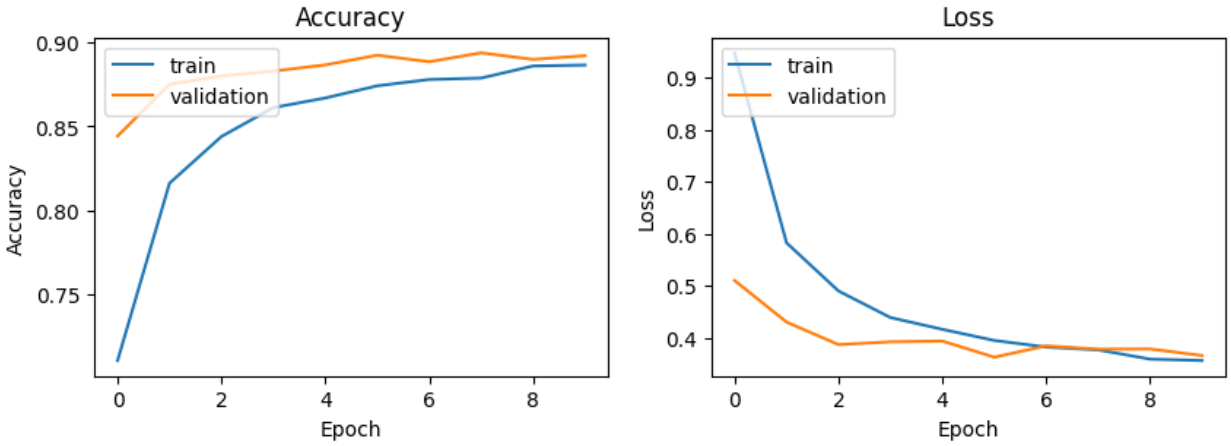
حال مدلمان را تعریف می کنیم. در سوال گفته شده است که اندازه تصویر را تغییر دهیم اما می توانیمدر ابتدای شبکه لایه ای تعیین کنیم که تمام ورودی خود را طبق ورودی خواسته شده reshape کند و در نهایت مدل تعریف شده به صورت زیر است.

```
model = keras.Sequential(
    [layers.Lambda(lambda image: tensorflow.image.resize(image,
(224,224))),
        resnet,
        layers.RandomFlip("horizontal_and_vertical"),
        layers.RandomRotation(0.2),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(10, activation="softmax"),])
```

در نهایت مانند قسمت های قبل آن را , compile و train می کنیم.

نتیجه نمودار :

با مقایسه با قسمت های قبلی، متوجه می شویم که شیب loss accuracy , در هر دو نمودار بیشتر است نسبت به حالت های قبل و مشخصا یادگیری در این حالت بهتر و بیشتر است.عملکرد روی داهد های ولیدیشن دارای یک شیب ثابت است و در نمودار Accuracy با یک شیب نسبتا ثابتی است و مقدار خطای آن برای validation از ابتدا پایین است و مقدار کمی overfiting رخ داده است چرا در نمودار خطا در اخر نمودار مقدار خطای داده های validation از train بیشتر است.



حالت چهارم)

میخواهیم بدون عوض کردن اندازه داده ها و استفاده از تنها سه لایه ی اول شبکه RESNETمدل را پیاده سازی کنیم.

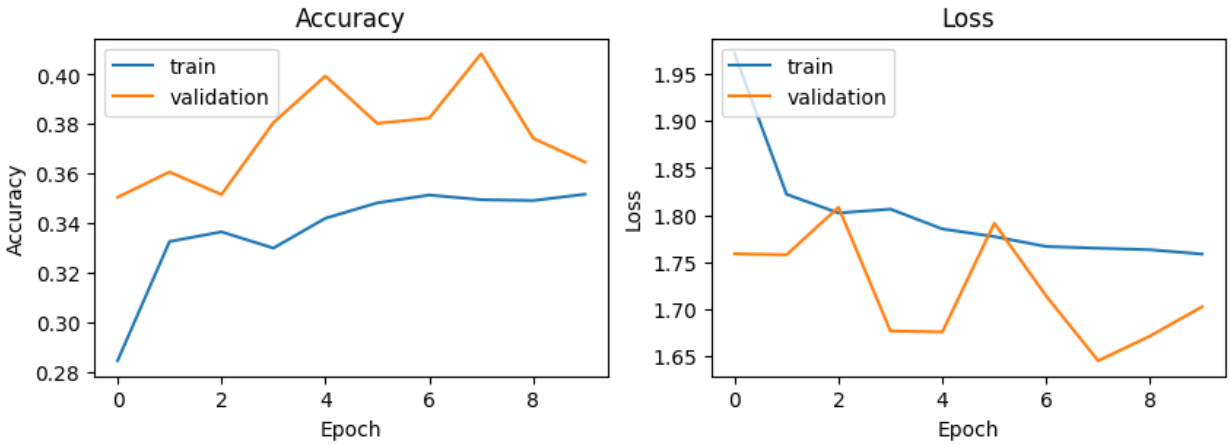
برای اینکار همان طور که گفتیم از همان ابتدای کار ورودی INCLUDE_TOP FALSE است در نتیجه ورودی ما می تواند ورودی دلخواه باشد.

در مدلمان یک اینپوت به اندازه ورودی داده های CIFAR10می دهیم. یعنی 32*32*3 سپس سه لایه اول میانی را به مدلمان اضافه می کنیم.

```
odel = keras.Sequential(
    [
         keras.Input(shape = (32,32,3)),
        resnet.layers[1] ,
        resnet.layers[2],
        resnet.layers[3] ,
        layers.RandomFlip("horizontal_and_vertical"),
        layers.RandomRotation(0.2),
        layers.Conv2D(128, kernel_size=(3, 3),
activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(10, activation="softmax"),
    ])
```

بعد از آن همانند قسمت های قبل آن را TRAINمی کنیم.

نتیجه



همان طور که مشاهده می شود ، در اینجا بدون نیاز به تغییر سایز از سه لایه اول استفاده کردیم. اما نتیجه مطلوب نیست چرا که داده ها دارای نواسانات زیادی است و در خیلی از نقاط مقدار خطای داده های VALIDATION بیشتر است در نتیجه نتیجه مطلوبی نیست و مقدار خطا و ACCURICITY خوب نیست و پایین تر از حد مظلوب است.

سوال 4)

الف)

گام یا STRIDE تعداد گام هایی است که فیلتر با آن بر روی ورودی حرکت می‌کند. با استفاده از stride، ما می‌توانیم قدم هایمان را برای حرکت فیلترمشخص کنیم. برای مثال اگر STRIDE دو یاشد.به جای این که از مرکز هز پیکسل یک دانه یک دانه به جلو برویم دو تا دو تا به جلو می رویم.

تفاوت بین stride و عملکرد pooling در لایه‌های کانولوشنی در استفاده از اطلاعات است. وقتی از stride استفاده می‌کنیم، هرچقدر stride بزرگتر باشد، ورودی به نسبت کوچکتر و نمونه‌برداری بیشتر خواهد شد. این به معنای کاهش اندازه خروجی و اطلاعاتی است که در شبکه عصبی حفظ می‌شوند. از سوی دیگر، pooling به منظور کاهش ابعاد فضا و تقلیل از تعداد پارامترها استفاده می‌شود، اما بدون از دست دادن کامل اطلاعات. در عمل، pooling برخلاف stride عملکردی غیرخطی دارد و بیشتر برای استخراج ویژگی‌های مهم از تصویر استفاده می‌شود.

تأثیر stride بر روی شبکه‌های عصبی در تعداد پارامترها، اندازه خروجی و اطلاعاتی است که در فرآیند کانولوشن حفظ می‌شوند، با افزایش stride، تعداد پارامترها کاهش می‌یابد و اندازه خروجی نیز کوچکتر می‌شود. این می‌تواند به کاهش پیچیدگی محاسباتی و تعداد پارامترها کمک کند، اما ممکن است که اطلاعات مهم از بین بروند.

ب)

1) لایه های میانی: بهترین توابع برای این مسئله ACTIVATIONهای reluو یا PARAMETRIC RELU می باشند.

این یک فعال ساز غیر خطی است و همچنین جلوی VANISHING GTADIANT یا EXPLODE GRADIANT را تا حدی می گیرد.

این فعال ساز بین 0 X ماکس را انتخاب می کندو انتخای کم هزینه سریغ و به صزفه ای است.

لایه ی اخر :

به طور کلی برای مسایل چند کلاسه ، توابع ACTIVATION SOFTMAX مناسب هستند.از SIGMOIDنیز در صورتی که مسئله دو کلاسه (معیوب ، سالم) باشد می توان استفاده کرد.

ب)

برای تایع ضرر می توانیم از تایع CROSS ENTROPYاستفاده کنیم چرا که برای مسئله های CLASSIFICATIONمناسب ترین است.

تایع خطای آنتروپی باتوجه به تفاوت بین توزیع پیش بینی شده و LABEL های واقعی به شبکه کمک می کند تا در طبقه بندی درستمحصولات معیوب و یا سالم بهبود یابد و باعث می شود که شبکه بیشترین احتمال را به تشخیص صحیح داده ها بدهد.

ب3)

برای ارزیابی عملکرد مدل در تشخیص محصولات معیوب از سالم و کاهش تعداد محصولات معیوب که به دست مشتری می‌رسد، معیار دقت (precision) می‌تواند معیار مناسبی باشد.

دقت (precision) تعداد مثبت‌های درست (True Positives) را بر تعداد مثبت‌های پیش‌بینی شده (True Positives + False Positives) محاسبه می‌کند. در این حالت، مثبت‌ها نمایانگر تشخیص محصولات معیوب است. با استفاده از دقت، ما می‌توانیم میزان دقت در تشخیص محصولات معیوب رابررسی کنیم و بفهمیم که چه درصد از محصولاتی که توسط مدل به عنوان معیوب شناسایی شده‌اند، واقعاً معیوب هستند.با این رویکرد، مدلی را پیشترین می شود د که دقت بالا و تعداد معیوباتی که به دست مشتری می‌رسد را به حداقل ممکن برساند.

به طور خلاصه، با استفاده از معیار دقت (precision) می‌توانید عملکرد مدل در تشخیص محصولات معیوب را ارزیابی کنید و بررسی کنید که چه درصد از محصولات شناسایی شده به عنوان معیوب، واقعاً معیوب هستند. با تلاش برای افزایش دقت، می‌توانید تعداد محصولات معیوبی که به دست مشتری می‌رسد را به حداقل ممکن کاهش دهد.

پ)

الف) تشخیص متن در تصویر:

در این حالت شبکه های کانولوشنی نمی توانند به خوبی عمل کنند. بیشتر شبکه های کانولوشنی وابستگی مکانی دارند و احتملا ارتباط بین کلمه ها و بار معنایی آن هارا به خوبی متوجه نشوند. برای این حالت از شبکه های RNNاستفاده می شود.

ب) تشخیص گوینده از روی صوت:

شبکه های کانولوشنی برای این حالت می تواند مناسب باشد. صوت نیز دارای فرکانس های خاص ،راز و نشیب و تغییراتی است و شبکه کانولوشنی به خوبی می تواند آن را تشخیص بدهد.(. تشخیص الگوهای صوتیو مکانیک ویژگی‌های صوتی و داده های زمانی)

پ) تحلیل جدول مربوط به مشتریان یک فروشگاه برای پیش بینی رفتار بعدی هر مشتری:

شبکه ی کانولوشنی نمی تواند مورد مناسبی برای این موضوع باشد. چرا که باید علت رفتار انسانی را بشناسد و تحلیل کند .

د)

1. نیاز به حجم بالای داده: شبکه‌های عصبی کانولوشنی برای آموزش و عملکرد بهتر نیاز به حجم بالای داده دارند. اگر که داده محدود یا ناکافی باشد، شبکه ممکن است به درستی عمل نکند و دقت کمتری داشته باشد.

2. پارامترهای قابل تنظیم زیاد: شبکه‌های CNN دارای تعداد زیادی پارامتر قابل تنظیم هستند که برای آموزش بهینه و جلوگیری از برازش بیش‌ازحد می‌بایست مورد تنظیم قرار گیرند.

3. حساسیت به ورودی‌های ناهنجار: شبکه‌های CNN ممکن است در مواجهه با ورودی‌های ناهنجار، مانند داده‌های تغییر شده یا نویزی، به خوبی عمل نکنند و دقت کاهش یابد. برای مقابله با این مشکل، روش‌هایی مانند تکنیک‌های نرمال‌سازی و افزودن لایه‌های Dropout به شبکه می‌تواند مورد استفاده قرار گیرد.

5. انتقال‌پذیری محدود: شبکه‌های CNN برای وظایف خاصی طراحی می‌شوند و در صورتی که بخواهیم آن‌ها را به وظایف دیگر منتقل کنیم، نیاز به تغییرات و تنظیمات گسترده‌ای خواهیم داشت. این ممکن است محدودیت‌هایی برای انتقال‌پذیری شبکه‌های CNN به وظایف جدید ایجاد کند.

سوال 5)

الف)

درنوت بوک SS_NOTE BOOK مراحل خواسته شده مرحله به مرحله پیاده سازی شده است.

در ابتدا، داده هایمان را مرتب می کنسم. یعنی آن ها را متناسب با برچسب هایشان جدا کرده و در مسیر های جدا با فرمت PNG ذخیره می کنیم.

سپس هر کدام از مسیر عکس ها را در لیست های جدا ذخیره کرده و از آن یک دیتا فریم مشابه با پایگاه داده میسازیم.

همان طور که در سوال گفته شده است برای هر غکس مسیر خود عکس و ماسک آن را در دریک ردیف قرار می دهیم و سه ستون برای id، مسیر عکس و مسیر ماسک وجود دارد. در نهایت این دیتا فریم را با نسبت 0.7و 0.3 به عنوان داده های train validation قرار می دهیم.

بعد از این مرحله، آن ها را داخل تابع preproccess – augmentation – decode تبدیل به datasetهای tensorflowکرده و به قسمت تعریف مدل می رویم

```
train_dataset =
train.cache().shuffle(BUFFER_SIZE).batch(BATCH_SIZE).repeat()
train_dataset = train_dataset.prefetch(tf.data.AUTOTUNE)
valid_dataset =
train.cache().shuffle(BUFFER_SIZE).batch(BATCH_SIZE).repeat().prefetch
(tf.data.AUTOTUNE)
```

مدل) .

مدل ما مدل unetاست که مبتنی بر لایه های شبکه از پیش پردازش شده mobilenet2 است. همچنین از تکنیک انتقال داده استفاده می کنیم.

در این قسمت یک مدل فانکشنال تعریف کرده و ورودی را ورودی mobilenet و خروجی را لیستی از لایه های خواسته شده قرار میدهیم.

در مرحله ی بعذی یعنی decoder upsampling را برای مدل unet تعریف می کنیم به به صورتی که یک مدل seq که از یک لایه ی convolotionaltranspose وbatchnormalization را قرار می دهیم و خود سوال لیستی را از upstackهای مختلف برای دیکود کردن تعریف کرده است.

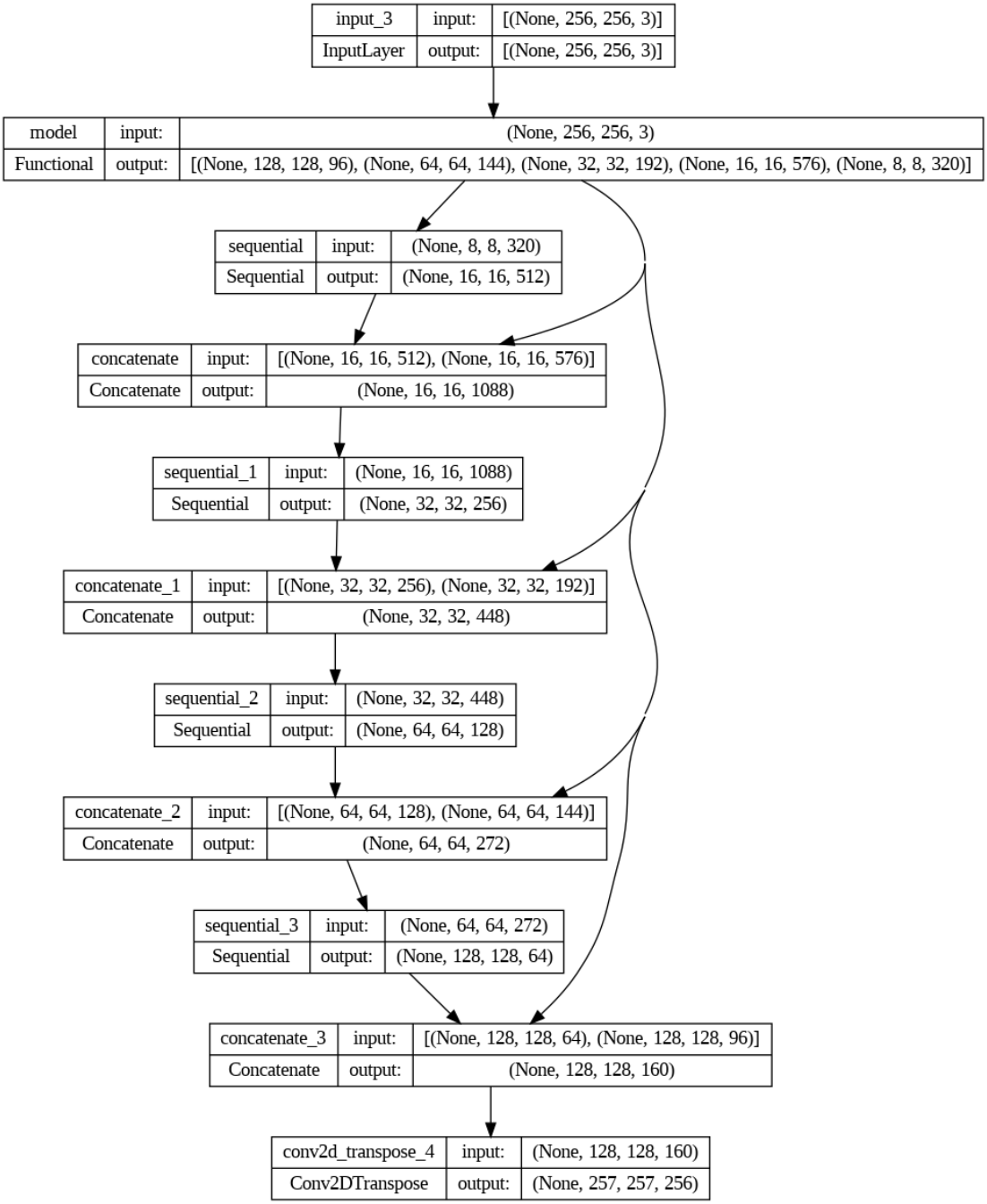
در مرحله آخر ، نوبت وصل کردن این لایه ی backbone و استفاده از دیکودر است. به این صورت که با دادن اینپوت به backbone لیستی از لایه های خروجی آن را میگیریم و روی آن به صورت برعکس یک فور زده و هر لایه را به اپ استک می دهیم و در نهایت با لایه های backbone ترکیب می کنیم تا در نهایت مدل زیر تشکیل شود.

```
 input = tf.keras.layers.Input(shape = (256,256,3))
    out_list = backbone(input)
    output = out_list[-1]

    for i,stack in enumerate(up_stack):
      result = stack(output)
      output = tf.keras.layers.Concatenate()([result, out_list[3-i]])

    out =tf.keras.layers.Conv2DTranspose(output_channels, (3,3),
strides=(2, 2) , activation='sigmoid' )(output)
    model = tf.keras.Model(input , out)
```

در نهایت از این خروجی و ورودی اولیه آن یک مدل فانکشنال ساخته و آن را با loss function داده شده کامپایل می کنیم.



ب امتیازی)

. هرکدام از این توابع خطای جهت بهبود عملکرد مدل در تشخیص و تفکیک شیءها در تصویر استفاده می‌شوند،

BCE Loss یک تابع خطای استاندارد است که در مسائل دسته‌بندی دو کلاسه استفاده می‌شود. در مسئله تشخیص شیء و نشانه‌گذاری، هر پیکسل در تصویر به یکی از دو دسته "شیء" یا "پس زمینه" تعلق می‌گیرد. BCE Loss از تابع sigmoid و

تابع cross-entropy برای محاسبه خطا استفاده می‌کند. این تابع خطا به صورت مستقل بر روی هر پیکسل محاسبه می‌شود و به مدل کمک می‌کند .

IoU Loss یا Dice Loss معیاری است که بر اساس اشتراک بخشی که مساحت مشترک دسته تشخیص داده شده با مساحت کل دسته است، تعریف می‌شود. IoU معمولاً برای ارزیابی عملکرد در مسائل نشانه‌گذاری استفاده می‌شود، اما به عنوان یک تابع خطای مستقیم نیز قابل استفاده است. IoU Loss به مدل کمک می‌کند تا دقت تفکیک و اندازه‌بندی شیء‌ها را بهبود ببخشد و به شدت به اندازه و شکل شیء توجه می‌کند.

در کل، هر دو تابع خطا BCE Loss و IoU Loss در مسئله تشخیص و نشانه‌گذاری معیارهایی را ارائه می‌دهند که می‌تواند بهبودی در دقت و کیفیت تفکیک شیء‌ها را به همراه داشته باشد، اما رویکردها و روش‌های محاسبه آنها متفاوت است. انتخاب مناسب بین این دو تابع خطا وابسته به مسئله خاص و اهداف مورد نظر می‌باشد.