سوال یک

الف) مسئله ناپدید شدن و انفجار گرادیان در شبکه‌های عصبی عمیق به دلیل وجود لایه‌های بسیار عمیق در شبکه‌ها و انجام عملیات ماتریسی بر روی وزن‌های بسیار کوچک یا بزرگ رخ می‌دهد. این مشکلات مانع از آموزش بهینه شبکه و کاهش دقت و همچنین افت کارایی شبکه می‌شوند.مسئله ناپدید شدن گرادیان به معنی کاهش گرادیان‌ها در لایه‌های عمیق شبکه است. به عبارت دیگر، گرادیان‌ها به مرور لایه‌های شبکه کوچکتر و کوچکتر می‌شوند تا در نهایت به صفر نزدیک شوند. این مشکل باعث می‌شود که وزن‌ها در لایه‌های عمیق به روزرسانی نشوند و شبکه به دقت کافی برای پیش‌بینی خروجی نرسد.مسئله انفجار گرادیان به معنی افزایش شدید گرادیان‌ها در لایه‌های عمیق شبکه است. به عبارت دیگر، گرادیان‌ها به مرور لایه‌های شبکه بزرگتر و بزرگتر می‌شوند تا در نهایت به مقادیر بسیار بزرگی برسند. این مشکل باعث می‌شود که الگوریتم بهینه‌سازی شبکه (مانند الگوریتم گرادیان کاهشی) نتواند به درستی به روزرسانی وزن‌ها را انجام دهد و شبکه به دقت کافی برای پیش‌بینی خروجی نرسد.

برای تشخیص مشکل ناپدید شدن و انفجار گرادیان در شبکه‌های عصبی عمیق، می‌توانید به روش‌های زیر رجوع کنیم:

برای مشکل انفجار گرادیان:

عدم یادگیری مدل
یکی از نشانه‌هایی که ممکن است مدل شما دچار مشکل انفجار گرادیان شده باشد، کمبود یادگیری مدل می‌باشد. با کمبود یادگیری، می‌توانید درک خوبی از داده‌های آموزشی نداشته باشید و همین موضوع می‌تواند نشانه‌ای برای این باشد که مدل شما دچار مشکل انفجار گرادیان شده است.

تغییرات بزرگ در خطای مدل
هنگامی که مدل شما دچار مشکل انفجار گرادیان شده است، ممکن است تغییرات بزرگی در خطای مدل در هر بار به‌روزرسانی پارامترها به وجود آید. این موضوع نشانه‌ای برای عدم پایداری مدل شما است.

خطاهای NaN
وقتی مدل شما دارای مشکل انفجار گرادیان است، ممکن است خطاهای NaN در طول آموزش نشان داده شوند. این مشکل به‌طور خاص در مدل‌هایی که از تابع فعال‌سازی ReLU استفاده می‌کنند، پیش می‌آید.

رشد نامنظم وزن‌ها:
وقتی مدل شما دارای مشکل انفجار گرادیان است، وزن‌های شبکه به‌طور نامنظم و با نرخ رشد بسیار زیادی رشد خواهند کرد. این موضوع نشانه‌ای برای مشکل انفجار گرادیان است.

برای مشکل ناپدید شدن گرادیان می توانیم از نشانه های زیر تشخیص دهیم:

عدم پیشرفت مدل
یکی از نشانه‌هایی که ممکن است مدل شما دچار مشکل ناپدید شدن گرادیان شده باشد، پیشرفت کند مدل در طول آموزش است. با پیشرفت کند، می‌توانید درک خوبی از داده‌های آموزشی نداشته باشید و همین موضوع می‌تواند نشانه‌ای برای این باشد که مدل شما دچار مشکل ناپدید شدن گرادیان شده است.

تغییرات کم در وزن‌های لایه‌های نزدیک به ورودی:
وقتی مدل شما دارای مشکل ناپدید شدن گرادیان است، تغییرات در وزن‌های لایه‌های نزدیک به ورودی به‌طور قابل توجهی کمتر خواهند بود و این موضوع به نشانه‌ای برای مشکل ناپدید شدن گرادیان است.

کاهش نامنظم وزن‌ها:
وقتی مدل شما دارای مشکل ناپدید شدن گرادیان است، وزن‌های شبکه به‌طور نامنظم و با نرخ رشد بسیار کمی رشد خواهند کرد. این موضوع نشانه‌ای برای مشکل ناپدید شدن گرادیان است.

وزن‌های صفر:
هنگامی که مدل شما دارای مشکل ناپدید شدن گرادیان است، ممکن است وزن‌های شبکه به صفر برسند. این مشکل به‌طور خاص در مدل‌هایی که از تابع فعال‌سازی sigmoid و tanh استفاده می‌کنند، پیش‌می‌آید.

چند راهکار برای مقابله با این مشکلات داریم:

کاهش تعداد لایه‌ها:
یکی از راه‌حل‌های ساده برای حل مشکل ناپدید شدن و انفجار گرادیان، کاهش تعداد لایه‌های شبکه عصبی است. با کاهش تعداد لایه‌ها، پیچیدگی مدل کمتر می‌شود و امکان ناپدید شدن گرادیان در لایه‌های عمیق کاهش می‌یابد. با این حال، با کاهش تعداد لایه‌ها، شبکه کمتر قابلیت نمایش مدل‌های پیچیده را دارد.

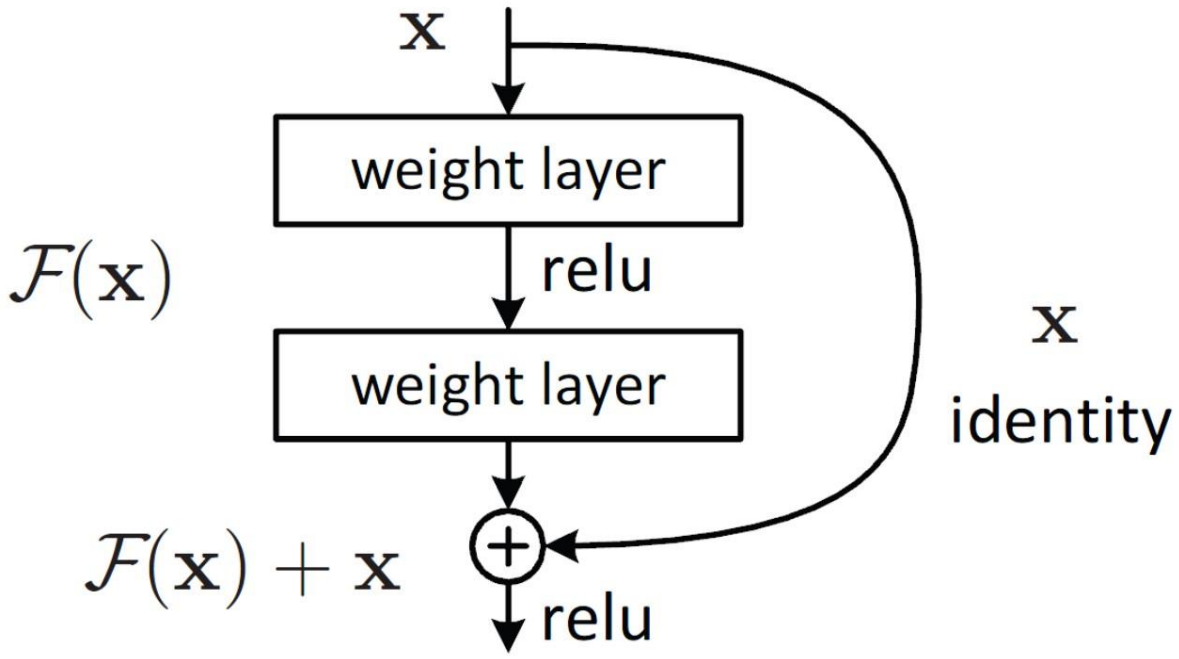
Gradient Clipping:
یک روش دیگر برای حل مشکل انفجار گرادیان، استفاده از Gradient Clipping است. در این روش، حداکثر

مقدار گرادیان مشخص می‌شود و هر گرادیانی که بیشتر از این حد باشد،همان مقدار حداکثر را میگذاریم. این روش از این جهت مفید است که می‌تواند به‌صورت مؤثری از انفجار گرادیان جلوگیری کند.

Weight Initialization:

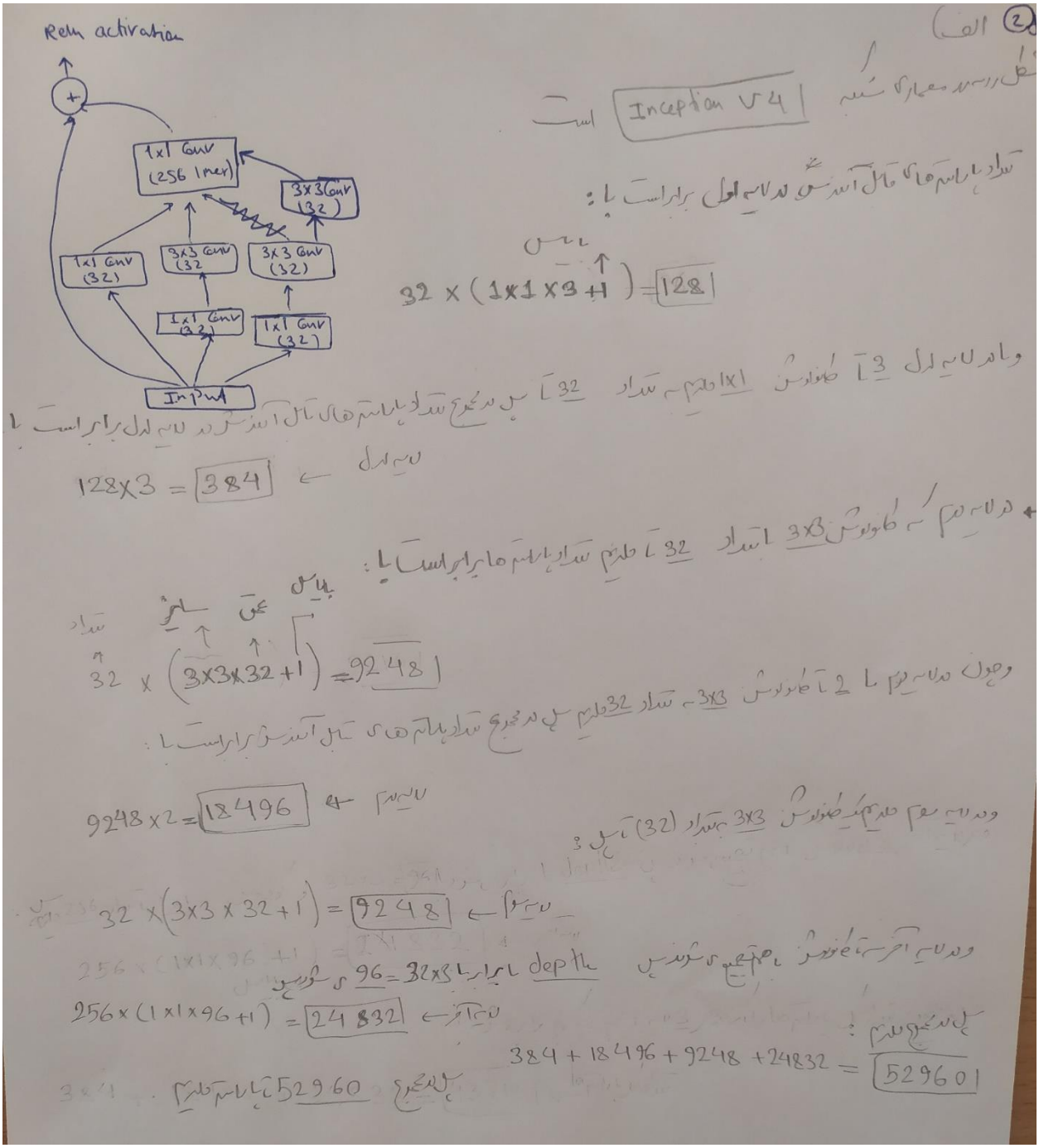
یک روش دیگر برای حل مشکل ناپدید شدن و انفجار گرادیان، انتخاب مرتب‌سازی دقیق برای مقداردهی اولیه وزن‌های شبکه است. مقداردهی اولیه وزن‌های شبکه به‌صورت تصادفی انجام می‌شود و انتخاب نادرست آن می‌تواند باعث مشکلاتی مانند ناپدید شدن و انفجار گرادیان شود. انتخاب یک مقداردهی اولیه مناسب، می‌تواند به‌صورت مؤثری از این مشکلات جلوگیری کند.

ب) شبکه‌هایی که قبل از ResNet معرفی شدند، برای عمیقتر شدن دچار مشکل ناپدید شدن گرادیان (Gradient Vanishing) بودند. برای حل این مشکل، شبکه ResNet ارائه شد. این شبکه با استفاده از یک ماژول خاص به نام Residual block، مشکل ناپدید شدن گرادیان را حل کرده است. در این ماژول، به جای اینکه ورودی به صورت مستقیم به خروجی لایه بعدی برسد، ابتدا این ورودی از مسیر موازی گذر کرده و سپس با خروجی لایه بعدی جمع می‌شود. این عمل باعث می‌شود که گرادیان‌ها به آسانی از لایه‌های دورتر به لایه‌های نزدیک به ورودی منتقل شوند و این مشکل ناپدید شدن گرادیان حل شود.در مقابل، مشکل انفجار گرادیان (Gradient Exploding) در شبکه‌های قبل از ResNet وجود داشت، اما با استفاده از روش Residual این مشکل حل نشده است.

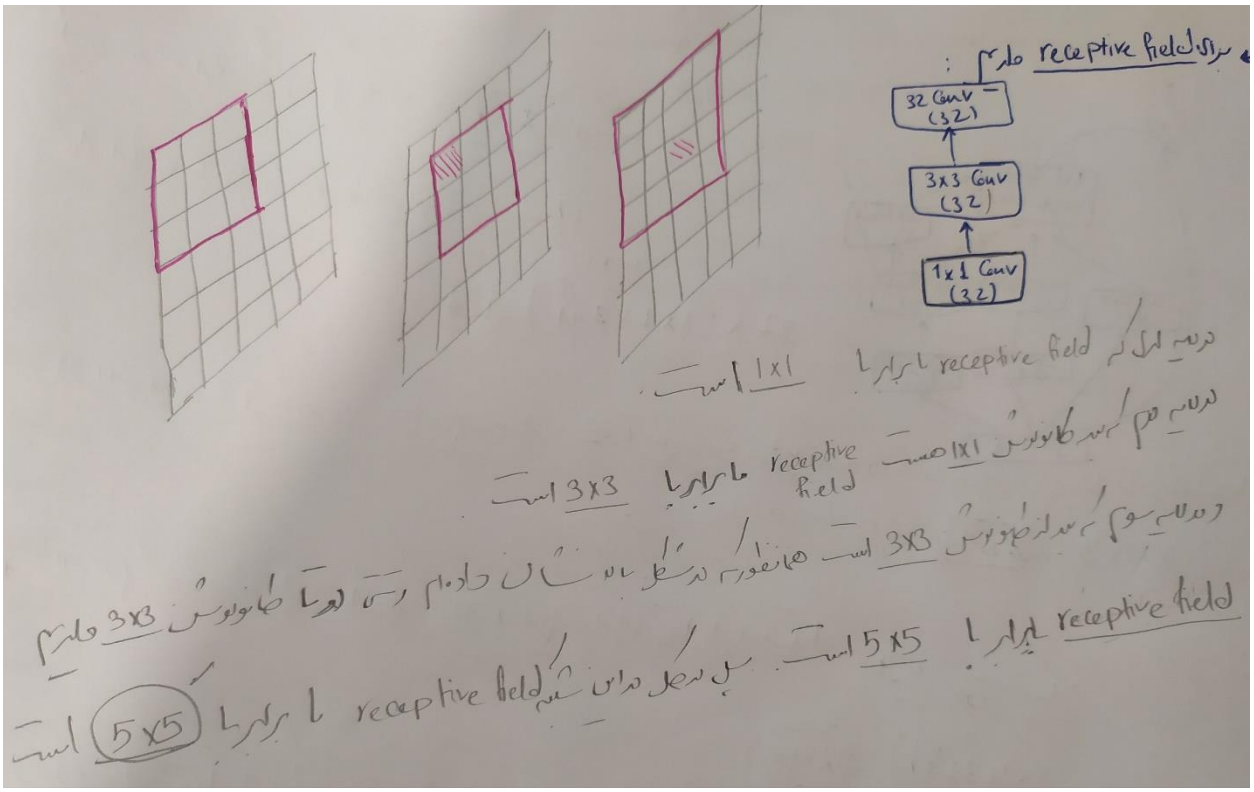


سوال دو

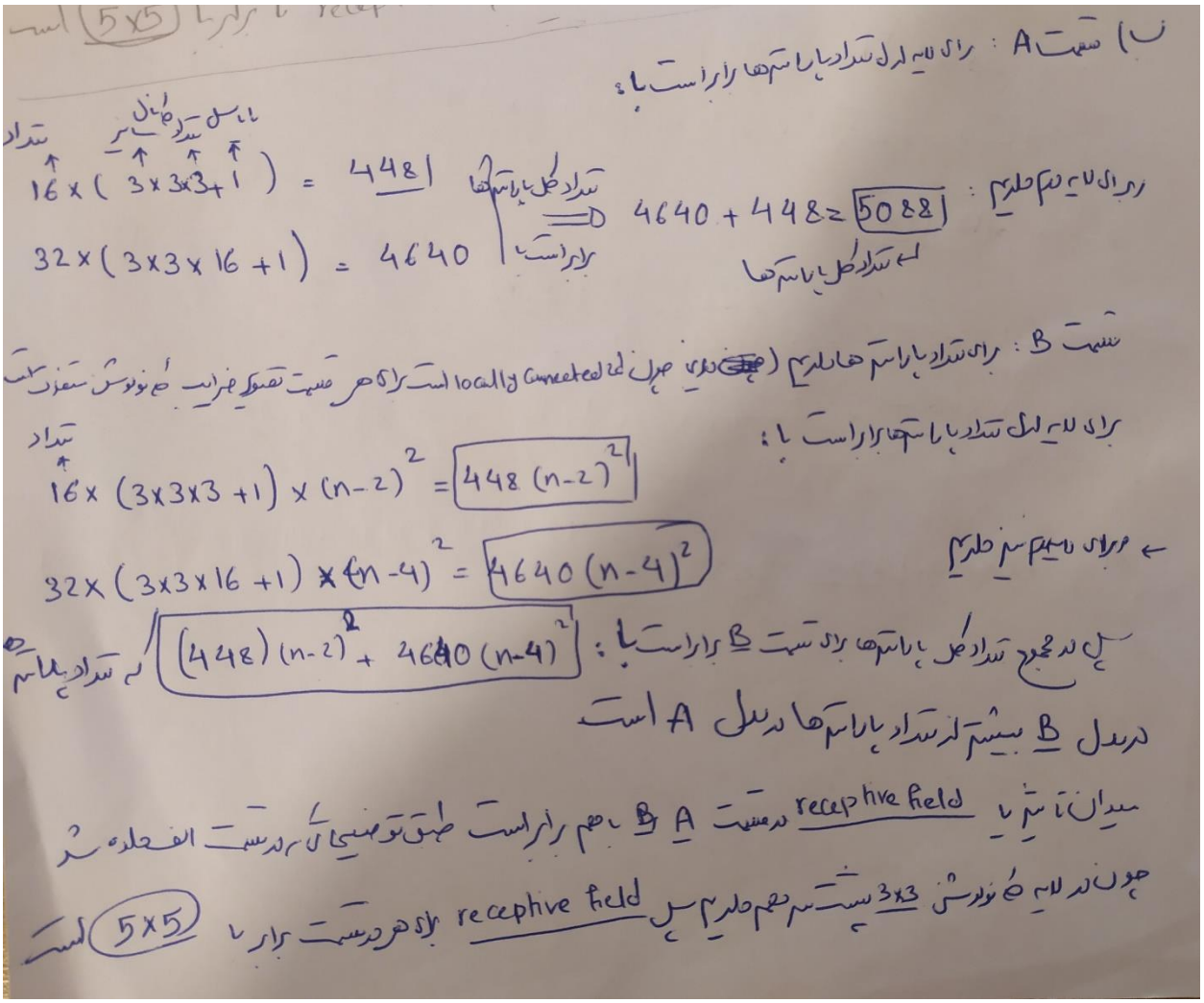
الف) تعداد کل پارامترها برابر با شکل زیر است:



و برای receptive field برابر است با:

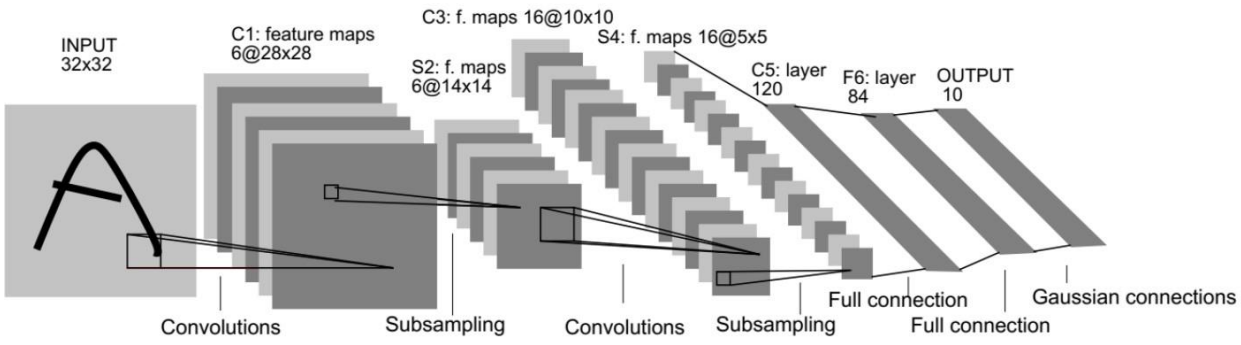


ب)جواب در صفحه بعد است.

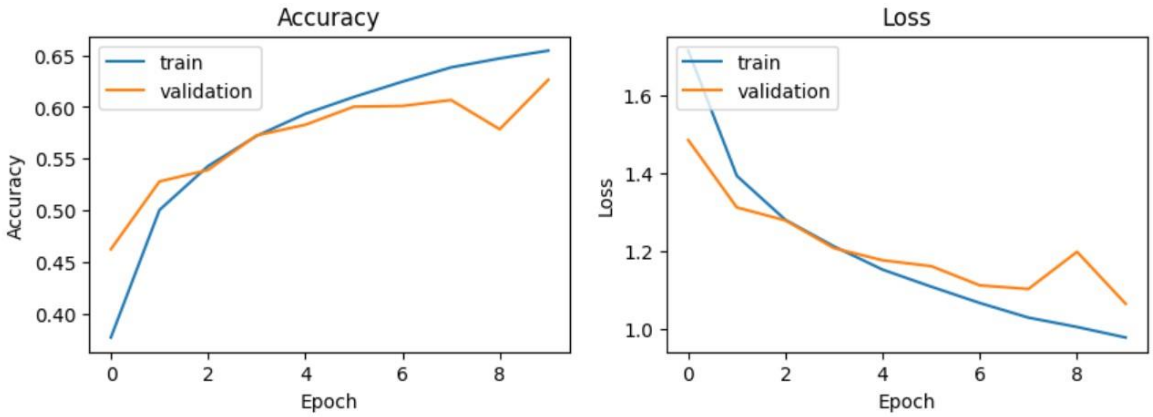


سوال سوم

الف)من شبکه LeNet-5 را پیاده سازی کردم که به صورت زیر است:



و نمودار دقت و ضرر هم به صورت زیر میشود:



ب) تبدیلات مختلف با استفاده از ImageDataGenerator در Keras تعریف شده‌اند.در ابتدا یک شی ImageDataGenerator با تعدادی تبدیلات مختلف ایجاد شده است. تعدادی از تبدیلاتی که اعمال شده‌اند عبارتند از:

rotation_range=30: این تبدیل از تصاویر با یک زاویه تصادفی تا حداکثر ۳۰ درجه دوران اعمال می‌کند.

width_shift_range=0.1 و height_shift_range=0.1: این تبدیلات تصاویر را به صورت تصادفی به سمت چپ، راست، بالا و پایین جابجا می‌کنند تا حداکثر ۱۰٪ از اندازه تصویر.

shear_range=0.1: این تبدیل از تصاویر با یک زاویه تصادفی تا حداکثر ۱۰ درصد خمیدگی اعمال می‌کند.

zoom_range=0.1: این تبدیل از تصاویر با یک مقیاس تصادفی تا حداکثر ۱۰ درصد بزرگتر یا کوچکتر می‌کند.

horizontal_flip=True: این تبدیل تصاویر را به صورت تصادفی در افقی برعکس می‌کند.

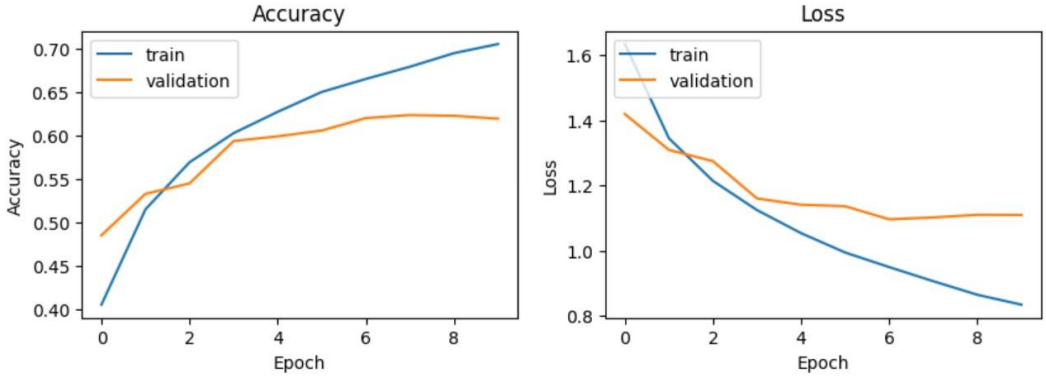
و بعد دیتاهای جدید را با دیتاهای قبلی concat می کنیم.

```
new_train_generator =datageneration.flow(x_train, y_train, batch_size=100)


# # combine two dataset
x_train_combined = np.concatenate([x_train, new_train_generator.x])
y_train_combined = np.concatenate([y_train,new_train_generator.y])


print(x_train_combined.shape)
print(y_train_combined.shape)
```

ج)بعد از اضافه کردن داده ها نمودار دقت و ضرر به صورت زیر شد:

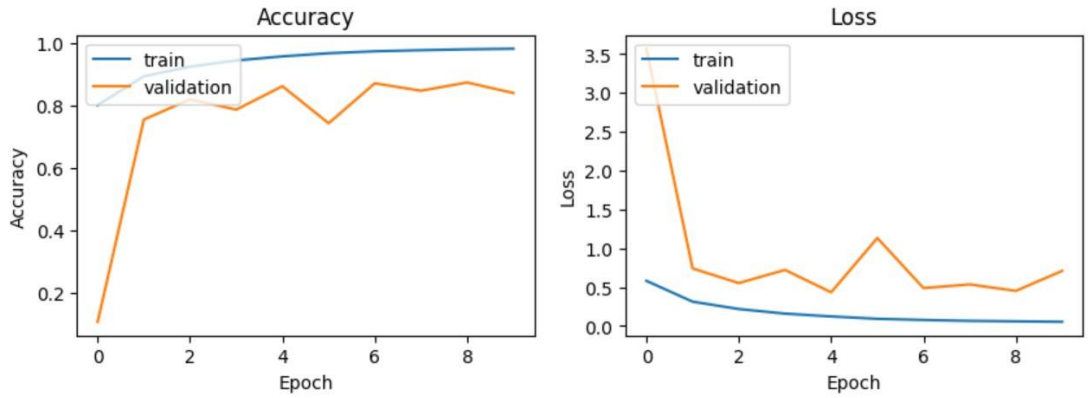


بعد از داده افزایی ضرر و دقت برای داد های تست تغییر چندانی نکرد ولی برای داده های train بهتر شد همان جورکه در نمودار مشاهده میکنید فاصله این دو تا داده یعنی تست و اموزش در هر دو نمودار نسبت به حالت قبل بیشتر است در حالت قبل بدون داده افزایی نمودار ها به هم نزدیک بود پس دچار overfitting نشده بودیم ولی اینجا فاصله زیاد است پس می توانیم بگوییم بعد از داده افزایی دچار overfitting شده ایم(نتیجه برعکس چیزی شد که انتظار داشتیم.).دقت و ضرر در داده های اموزش بعد از داده افزایی خوب است ولی در هر دو حالت چه در داده های تست و چه در داده های اموزش انقدر کم نیست که بگوییم دچار underfitting شده ایم.یک دلیل اینکه برعکس چیزی شد که انتظار داشتیم این است که شبکه ما ساده بود واین باعث شد بعد از داده افزایی داده های تست حفظ کنه و روی ان ها عملکرد خوبی داشته باشد ولی رو داده های تست تغییری ایجاد نشود.

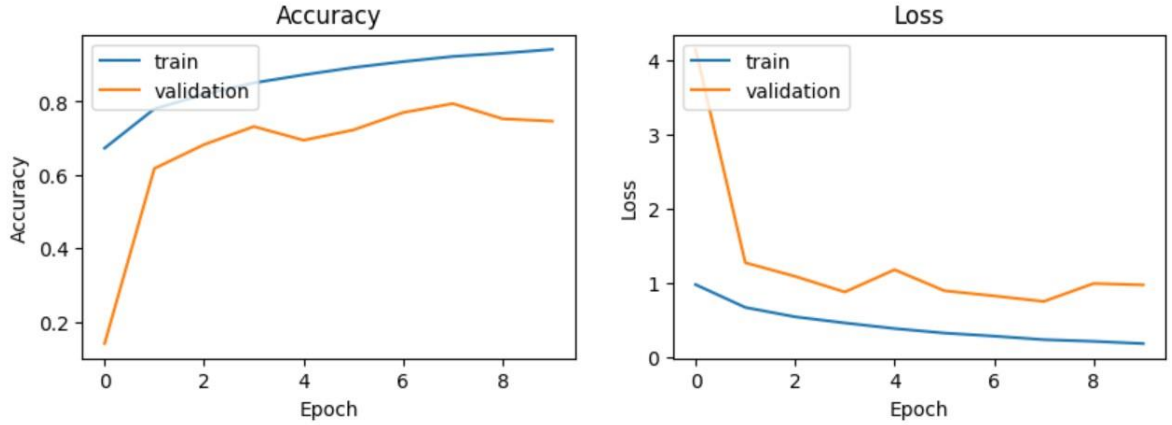
د)در مدل خود یک لایه برای ریسایز کردن اضافه میکنیم

```
modelres = keras.Sequential([
    tf.keras.layers.Resizing(224, 224),
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(num_classes, activation='softmax')
])
modelres.build(input_shape=(None, 224, 224, 3))
```

با اضافه کردن مدل reznet پیشرفت خیلی خوبی هم در دقت و ضرر در داده های تست داشتیم نمودار ان به صورت زیر شد:

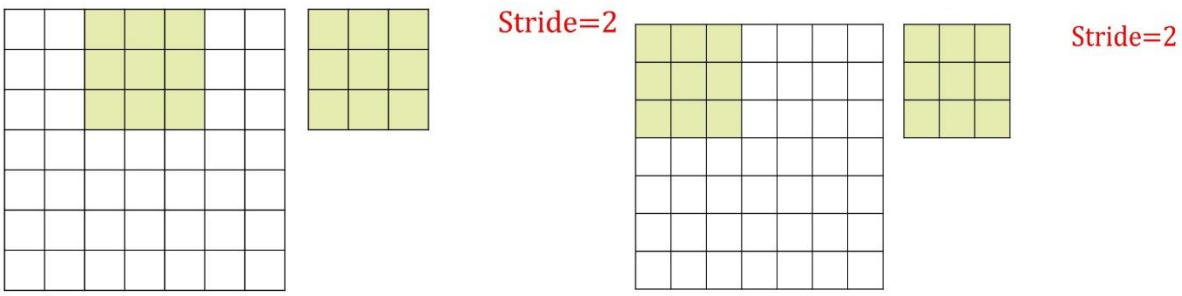


ه) در این کد، خروجی لایه به نام "conv3_block4_out" را از مدل ResNet50 استخراج می‌کند. این خروجی یک نقشه ویژگی است که ویژگی‌های یادگرفته شده تصویر ورودی را پس از گذر از لایه‌های قبلی مدل ResNet50 نشان می‌دهد.سپس چند لایه دیگر را به بالای خروجی استخراج شده اضافه می‌کند. ابتدا، یک لایه GlobalAveragePooling2D برای کاهش ابعاد اضافه می‌شود. سپس، یک لایه fully connected dense با ۲۵۶ نورون و فعال‌سازی ReLU برای یادگیری انتقالات غیرخطی بین ویژگی‌های ورودی و کلاس‌های خروجی اضافه می‌شود. یک لایه Dropout برای جلوگیری از بیش‌برازشی با حذف تصادفی برخی نورون‌ها در طول آموزش اضافه می‌شود. در انتها، یک لایه خروجی dense با فعال‌سازی softmax برای تولید احتمالات پیش‌بینی برای هر یک از کلاس‌های خروجی اضافه می‌شود.نتایج هم داده های تست و هم داده های trian در دقت وضرر نسبت به قسمت قبل بدتر هست ولی نسبت به قسمت الف و ب بهتر هست نمودار ان به صورت زیر است:

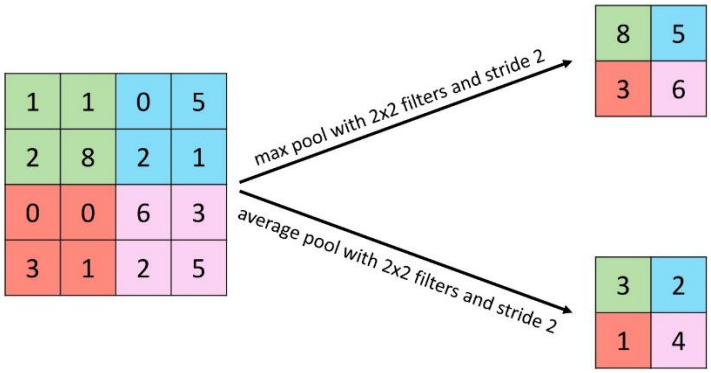


سوال چهارم

الف)Stride در لایه‌های کانولوشنی یک پارامتر است که نشان می‌دهد که در هر گام (step) از پاسخ لایه قبلی، فقط چه تعداد پیکسل را برای اعمال فیلتر جدید بررسی کنیم. به عنوان مثال، اگر stride برابر با ۱ باشد، هر بار فیلتر یک پیکسل به سمت راست و یک پیکسل به سمت پایین حرکت می‌کند تا به تمام نقاط تصویر برسد. اما اگر stride برابر با ۲ باشد، فیلتر هر بار دو پیکسل به سمت راست و دو پیکسل به سمت پایین حرکت می‌کند مانند دو شکل زیر:



ولیPooling نیز یک عملیات هست که با استفاده از آن، تصویر را به صورت مقیاس کوچکتری با حفظ ویژگی‌های مهم کاهش داده می‌شود. در لایه‌های Pooling، یک پنجره با اندازه مشخصی روی تصویر حرکت کرده و در هر گام، مقدار خاصی از تصویر را (مثلاً ماکسیمم یا میانگین) به عنوان خروجی این لایه ارائه می‌کند.



تفاوت اصلی بین Stride و Pooling در این است که Stride برای اعمال فیلتر در لایه کانولوشنی و استخراج ویژگی‌ها از تصویر استفاده می‌شود، در حالی که Pooling برای کاهش ابعاد تصویر و حفظ ویژگی‌های مهم استفاده می‌شود.

Stride و Pooling هر دو می‌توانند تاثیر خوبی در عملکرد شبکه‌های عصبی داشته باشند. با افزایش مقدار Stride، اندازه خروجی لایه کانولوشنی کاهش می‌یابد که ممکن است باعث از دست رفتن اطلاعات مهم و کاهش دقت دسته‌بندی تصاویر شود. همچنین با افزایش اندازه پنجره Pooling، ابعاد تصویر کاهش می‌یابد و اطلاعات مهم تصویر کمتر می‌شود که ممکن است باعث کاهش دقت دسته‌بندی شود. بنابراین، انتخاب مقادیر مناسب برای Stride و Pooling می‌تواند در بهبود دقت دسته‌بندی تصاویر و بهبود عملکرد شبکه‌های عصبی مفید باشد ولی در عین حال باعث کاهش پارامترها میشود.

ب ۱) برای لایه میانی شبکه عصبی کانولوشنی، توابع فعال‌سازی زیادی وجود دارند که می‌توان از آن‌ها استفاده کرد، اما تابع فعال‌سازی ReLU به عنوان یکی از پراستفاده‌ترین توابع فعال‌سازی برای لایه میانی استفاده میشه. دلیل استفاده از تابع ReLU در لایه میانی، این است که این تابع خطی در نواحی منفی عمل نمی‌کند و مقادیر منفی را به صفر تبدیل می‌کند. این خاصیت باعث می‌شود که این تابع برای شناسایی ویژگی‌های پراهمیت تصاویر مفید باشد و باعث افزایش سرعت و عملکرد شبکه در فرآیند آموزش شود.برای لایه آخر شبکه که به منظور طبقه‌بندی دسته‌های مختلف استفاده می‌شود، تابع فعال‌سازی Softmax به عنوان یکی از پراستفاده‌ترین توابع فعال‌سازی برای لایه آخراستفاده میشه. دلیل استفاده از تابع Softmax در لایه آخر، این است که این تابع به صورت خروجی احتمالات پیش‌بینی شده برای هر دسته را محاسبه می‌کند. با تقسیم خروجی شبکه به یک مجموعه تابع احتمالی که مجموع این مقادیر یک است، این تابع احتمالات پیش‌بینی شده برای هر دسته را محاسبه می‌کند. این خاصیت باعث می‌شود که تابع Softmax برای مسائل طبقه‌بندی به خوبی عمل کند و بتواند دقت مدل را بهبود بخشد.بنابراین، به طور خلاصه می‌توان گفت که برای لایه میانی، تابع فعال‌سازی ReLU و برای لایه آخر، تابع فعال‌سازی Softmax پیشنهاد می‌کنم.

ب ۲) در مسئله طبقه‌بندی تصاویر محصولات معیوب و سالم در خط تولید کارخانه، با توجه به اینکه دو دسته محصولات سالم و محصولات معیوب وجود دارند، تابع خطای binary cross-entropy می‌تواند یک گزینه مناسب برای این مسئله باشد.تابع خطای binary cross-entropy یک تابع خطای مناسب برای مسائل طبقه‌بندی دو دسته‌ای است. این تابع با محاسبه خطای بین دو دسته، می‌تواند دقت و عملکرد مدل را بررسی کند. همچنین، استفاده از تابع binary cross-entropy از پایداری آموزش برخوردار است و می‌تواند به سرعت به جواب مطلوب برسد.بنابراین، تابع خطای binary cross-entropy برای مسئله طبقه‌بندی تصاویر محصولات معیوب و سالم در خط تولید کارخانه، یک گزینه مناسب است.

ب ۳)

Precision به ما نشان می‌دهد که چه مقدار از محصولاتی که مدل به عنوان معیوب شناسایی کرده است، واقعا معیوب هستند. به عبارت دیگر، این معیار به ما نشان می‌دهد که تعداد محصولاتی که به دست مشتری معیوب ارسال می‌شود چقدر است. با افزایش دقت مدل در شناسایی محصولات معیوب، precision نیز افزایش می‌یابد و تعداد محصولات معیوبی که به دست مشتری می‌رسد، کاهش می‌یابد.

Recall به ما نشان می‌دهد که چه مقدار از محصولات معیوب، توسط مدل به درستی شناسایی شده‌اند. به عبارت دیگر، این معیار به ما نشان می‌دهد که چقدر از تعداد کل محصولات معیوب، توسط مدل شناسایی شده‌اند. با افزایش recall، تعداد محصولات معیوبی که به دست مشتری می‌رسد، کاهش می‌یابد.با توجه به اینکه در این مسئله، تعداد محصولات معیوبی که به دست مشتری می‌رسد، باید به حداقل رسانده شود، می‌توان بهترین مدل را با معیار Precision انتخاب کرد. به علت بالا بودن Precision، تعداد محصولات معیوبی که به دست مشتری می‌رسد، به حداقل رسیده و به همین دلیل، Recall ممکن است کمی کاهش یابد.

ج) ۱) شبکه‌های عصبی کانولوشنی می‌توانند برای طبقه‌بندی موضوع متن به کار گرفته شوند و به دلیل قابلیت استخراج ویژگی‌های مختلف از متن، در این زمینه موفق عمل می‌کنند. بنابراین، شبکه‌های عصبی کانولوشنی در این کاربرد موفق خواهند بود. با استفاده از لایه‌های کانولوشن، این شبکه‌ها قادر به شناسایی ویژگی‌های مختلف از جمله الگوهای مختلف کلمات، اندازه و شکل کلمات، ترتیب کلمات و ... هستند. بنابراین، شبکه‌های عصبی کانولوشنی برای درک کلیت متن و شناسایی مفاهیم و کلمات بسیار موثر هستند.

۲) شبکه‌های عصبی کانولوشنی می‌توانند برای تشخیص گوینده از روی صدا به کار گرفته شوند، اما به دلیل محدودیت‌هایی که در استخراج ویژگی‌های صوتی وجود دارد، ممکن است در این کاربرد با دقت پایینی مواجه شوند. به طور مثال، تفاوت بین صداهای متفاوت از یک گوینده، تفاوت در شرایط ضبط صدا و... می‌تواند باعث کاهش دقت شبکه شود.برای تشخیص صدا زیاد مناسب نیستند.

۳) شبکه‌های عصبی کانولوشنی معمولا برای تحلیل تصاویر و سیگنال‌های دو بعدی به کار گرفته می‌شوند و برای تحلیل داده‌های جدولی به صورت مستقیم مناسب نیستند. برای پیش‌بینی رفتار بعدی هر مشتری می‌توان از شبکه‌های عصبی ساده با لایه‌های مخفی چندتایی استفاده کرد که بتوانند از داده‌های جدولی ویژگی استخراج کنند و پیش‌بینی رفتار بعدی را از آنها بگیرند. بنابراین، شبکه‌های عصبی کانولوشنی در این کاربرد به صورت مستقیم موفق نخواهند بود.

د)

تعمیم‌پذیری: در صورتی که مدل بر روی داده‌های آموزشی عملکرد خوبی داشته باشد، اما بر روی داده‌های جدید و ناشناخته نتواند به خوبی عمل کند، مشکل تعمیم‌پذیری در مدل وجود دارد. زمانی که مدل با داده‌های آموزشی بیش از حد learn شود و تعداد داده‌های آموزشی کم باشد، این مشکل رخ می‌دهد.

انتقال‌پذیری: در صورتی که مدل برای یک مسئله خاص طراحی شده باشد و برای مسائل دیگر به کار گرفته شود، ممکن است به دلیل تفاوت‌هایی در ویژگی‌های داده‌ها، عملکرد خوبی نداشته باشد.

بیش‌برازش: اگر شبکه بیش از حد پیچیده باشد و تعداد پارامترهای آن بیشتر از حد نیاز باشد، ممکن است به مشکل بیش‌برازش (overfitting) برخورد کند. در این حالت، مدل بر روی داده‌های آموزشی به خوبی عمل می‌کند، اما بر روی داده‌های جدید نتواند به خوبی عمل کند.

پایین بودن دقت: در برخی موارد، ممکن است دقت شبکه‌های عصبی کانولوشنی پایین باشد و برای بهبود دقت، نیاز به تغییر پارامترهای شبکه و یا افزایش تعداد داده‌های آموزشی باشد.

عدم قابلیت تفسیر: یکی از مشکلات شبکه‌های عصبی کانولوشنی، عدم قابلیت تفسیر آن‌ها است. به عبارت دیگر، ممکن است برای تشخیص الگوهای خاص در داده‌ها، شبکه‌های عصبی کانولوشنی از ویژگی‌های پیچیده و غیرقابل فهم استفاده کنند که برای محقق شدن یک تصمیم منطقی، نیاز به فهم عمیق‌تری از روند تصمیم‌گیری شبکه داریم.

برخورداری از ویژگی‌های محلی: شبکه‌های عصبی کانولوشنی برای تشخیص الگوهای موجود در داده‌ها، از ویژگی‌های محلی (local features) استفاده می‌کنند. بنابراین، در صورتی که ویژگی‌های مورد نظر برای تشخیص الگو در داده‌ها، در سطح بالاتری از داده‌ها وجود داشته باشند، شبکه‌های عصبی کانولوشنی ممکن است دچار مشکل شوند.

حساسیت به تغییرات: شبکه‌های عصبی کانولوشنی به طور کلی حساس به تغییرات در داده‌ها هستند. به عبارت دیگر، حتی با تغییر کوچکی در داده‌ها، ممکن است خروجی شبکه تغییر کند.

تحلیل دشوار: شبکه‌های عصبی کانولوشنی به دلیل پیچیدگی و تعداد زیاد پارامترهایی که دارند، تحلیل و فهم عملکرد آن‌ها دشوار است.

انتخاب پارامترها: برای آموزش شبکه‌های عصبی کانولوشنی، نیاز به انتخاب و تنظیم پارامترهای مختلفی از جمله اندازه فیلترها، تعداد لایه‌ها، تعداد نورون‌ها و ... است. انتخاب نادرست پارامترها می‌تواند باعث کاهش عملکرد شبکه شود.

سوال پنجم

ب) دو تابع loss BCE و IoU معمولا برای آموزش شبکه‌های segmentation semantic استفاده می‌شوند تابع loss BCE (Binary Cross-Entropy) یک تابع loss ساده است که معمولا برای مسائل دودویی استفاده می‌شود. این تابع برای هر پیکسل برچسبی بین ۰ و ۱ (یعنی دودویی) تولید می‌کند و هدف آن بهینه سازی پیش‌بینی‌های شبکه با برچسب‌های واقعی است. تابع loss BCE باعث می‌شود شبکه به پیش‌بینی‌هایی مانند ماسک‌های صحیح با مرزهای دقیق تر تمایل پیدا کند، اما مسئله این است که ممکن است باعث شود که شبکه به یک ماسک دقیق اما با اندازه‌ای نامناسب (مثلاً بسیار کوچک یا بسیار بزرگ) برسد که عملکرد شبکه را بهبود ندهد.تابع loss IoU (Intersection over Union) به عنوان یک تابع loss برای آموزش شبکه‌های segmentation semantic مورد استفاده قرار می‌گیرد. این تابع از IoU برای محاسبه دقت شبکه استفاده می‌کند. هدف این تابع بهینه سازی IoU بین پیش‌بینی‌های شبکه و برچسب‌های واقعی است. تابع loss IoU باعث می‌شود شبکه به پیش‌بینی‌هایی با اندازه مناسب و همچنین با مرزهای دقیق‌تر تمایل پیدا کند. به عبارت دیگر، این تابع باعث می‌شود شبکه به یک ماسک با اندازه و شکل مناسب برسد که در نتیجه باعث بهبود دقت شبکه می‌شود.به طور خلاصه، تابع loss BCE معمولا برای مسائل دودویی و تابع loss IoU برای مسائل segmentation semantic استفاده می‌شوند.