برای مشکل imbalance بودن دیتاست من از روش داده افزایی استفاده کردم و مراحل زیر را طی کردم:

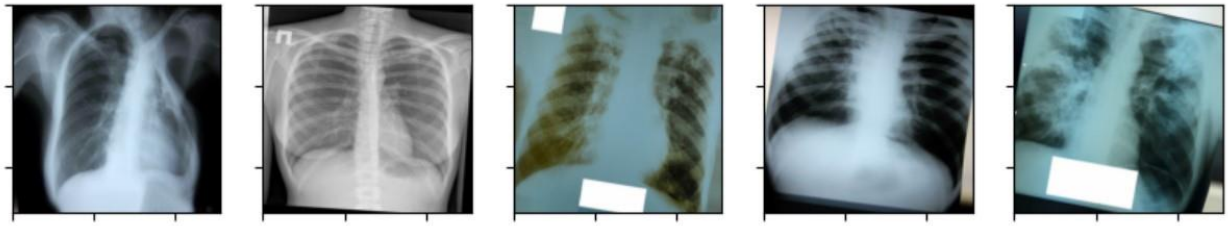
در ابتدا عکس های که مربوط به سل بود را از -۱۵ تا ۱۵ درجه به صورت رندوم با یک زاویه چرخانده میشوند از ImageDataGenerator استفاده کرد و هم چنین عکس چرخید شده را به ۰-۲۵۵ اسکیل کردم.

کد ان به صورت زیر است که برای بقیه داده افزایی هایی که در ادمه ذکر میکنم به صورت مشابه است.

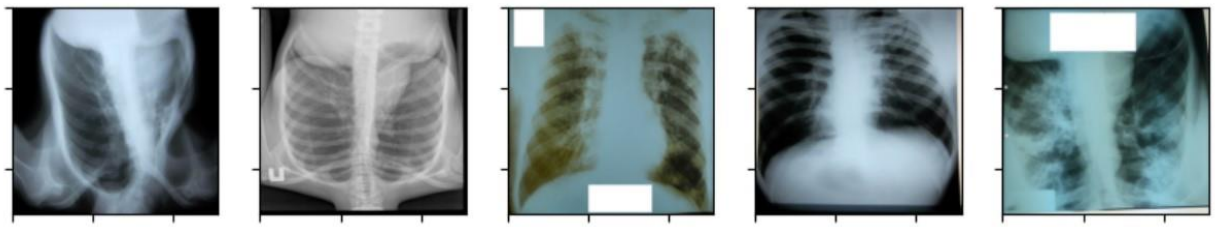
```
[ ]  # Define the ImageDataGenerator with random rotation augmentation
     datagen = ImageDataGenerator(rotation_range=15)
     # Generate randomly rotated images and store them in a new list
     imgstube_rotate = []
     for img in imgstube:
         # Reshape the image to add a channel dimension
         img = img.reshape((1,) + img.shape)
         # Generate a batch of randomly rotated images
         batch = datagen.flow(img, batch_size=1, shuffle=False)
         # Extract the first image from the batch and add a channel dimension
         rotated_img = batch.next()[0]
         rotated_img = minmax_scale(rotated_img.reshape(-1,1), feature_range=(0,255)).reshape(rotated_img.shape)
         rotated_img = rotated_img.astype('uint8')
         # Store the rotated image in the list
         imgstube_rotate.append(rotated_img)

     # Convert the list of rotated images to a numpy array
     imgstube_rotate = np.array(imgstube_rotate)
     print(imgstube_rotate.shape)
     # Plot the first 10 randomly rotated images
     plt.figure(figsize=(10,10))
     for i in range(10):
         ax=plt.subplot(2,5,i+1)
         plt.imshow(imgstube_rotate[i][:,:,::-1])
         ax.set_yticklabels([])
```

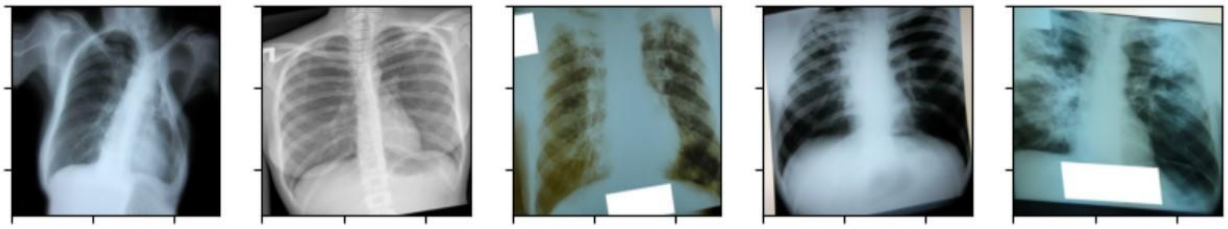
و برای نمونه عکس های داده افزایی چرخش هم به صورت زیر است:



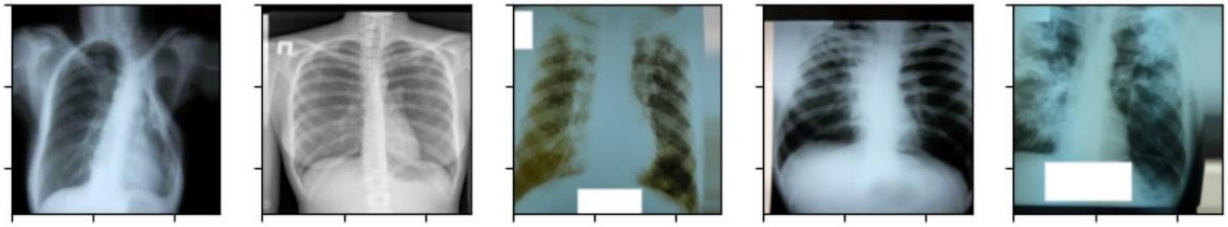
در ادمه از داده افزایی های horizontal_flip,vertical_flip استفاده کردم که تصاویر نمونه ان به صورت زیر است:



در ادامه از داده افزایی zoom_range, rotation_range, shear_range استفاده کردم در واقع از ترکیب این سه تا :



و برای داده افزایی اخر از تابع گوسین در opencv و هم چنین از شیفت height,width استفاده کردم که تصاویر نمونه به صورت زیر است:



و بعد از این چهارتا داده افزایی این چهارتا نامپای array را با داده اصلی عکس های سلcontact کردم و تعداد تصاویر سل نیز برابر با ۳۵۰۰ شد و در درایو ذخیره کردم.برای عکس های نورمال نیز من ۳۵۰۰ عکس ان را به ۵ قسمت مساوی ۷۰۰ تایی تقسیم کردم و روی چهارتا از دسته های ۷۰۰ تایی ان نیز این چهارتا عملی که بر روی عکس های سل انجام دادم را نیز انجام دادم.نکته:تعداد عکس های نورمال بعد از عملیات تعدادش برابر با ۳۵۰۰ میماند.

مدل:

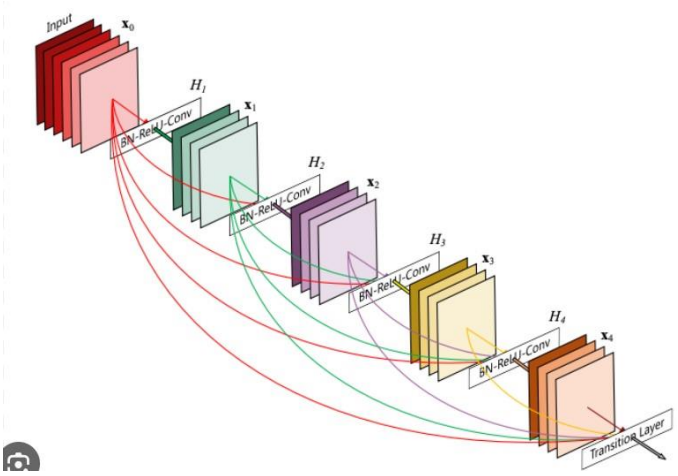
من بعد از داده افزایی عکس ها را به ۲۵۶ در ۲۵۶ ریسایز کردم و بعد روی دیتاست یک shuffle زدم و ۰.۲ داده ها را به عنوان داده تست جدا کردم.

من سه مدل را امتحان کردم که دوتای انها از transfer learning استفاده کردم و یک مدل densenet را نیز پیاده سازی کردم در ابتدا از densenet را توضیح میدم

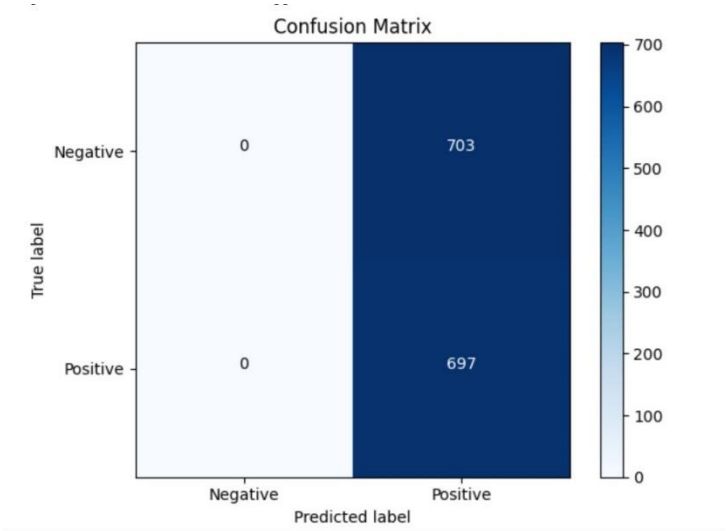
مدل densenet

یکی از مشکلات شبکه‌های عصبی عمیق سنتی این است که با افزایش عمق مدل، نیاز به تعدادی پارامتر بسیار بزرگ و همچنین بررسی همبستگی‌های غیرمستقیم بین لایه‌ها افزایش می‌یابد. این موضوع می‌تواند باعث کاهش سرعت آموزش و افزایش پیچیدگی مدل شود.

در شبکهDenseNet ، هر لایه با همه لایه‌های قبلی ارتباط دارد و ورودی‌ها در هر لایه با یکدیگر ترکیب می‌شوند. این روش باعث کاهش تعداد پارامترها و نیاز به منابع محاسباتی بیشتر می‌شود و در عین حال بهبود عملکرد شبکه و جلوگیری از مشکل کاهش گرادیان (gradient vanishing) کمک می‌کند.



این مدل روی دیتاست بیماری سل اصلا خوب کار نکرد و confusion matrix ان برای داده های تست به صورت زیز شد:



البته یکی از دلایل این بود که تعدادepoch را برابر با ۱۰ گذاشتم که مقدار کمی هست ولی با استفاده از transfer learning با همین تعداد epoch به دقت خیلی خوبی رسیدم که درادامه توضیح میدم:

من از یک مدل پیش‌آموزش دیده ResNet-50 روی دیتاست imagenet استفاده میکنم:

base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(256, 256, 3))در این بخش، از مدل ResNet-50 پیش‌آموزش دیده استفاده می‌کنم که وزن‌های آن از دیتاست ImageNet بارگذاری شده‌اند include_top=False .به معنی این است که لایه‌های Fully Connected آخر را حذف می‌شود و فقط بخش اولیه شبکه را استفاده می‌کنم.

x = base_model(inputs, training=False)شما تصاویر ورودی را از طریق مدل پایه ResNet-50 بگذرانید. با تنظیم training=False، این لایه ها در مرحله اموزش قرار نمی گیرند.

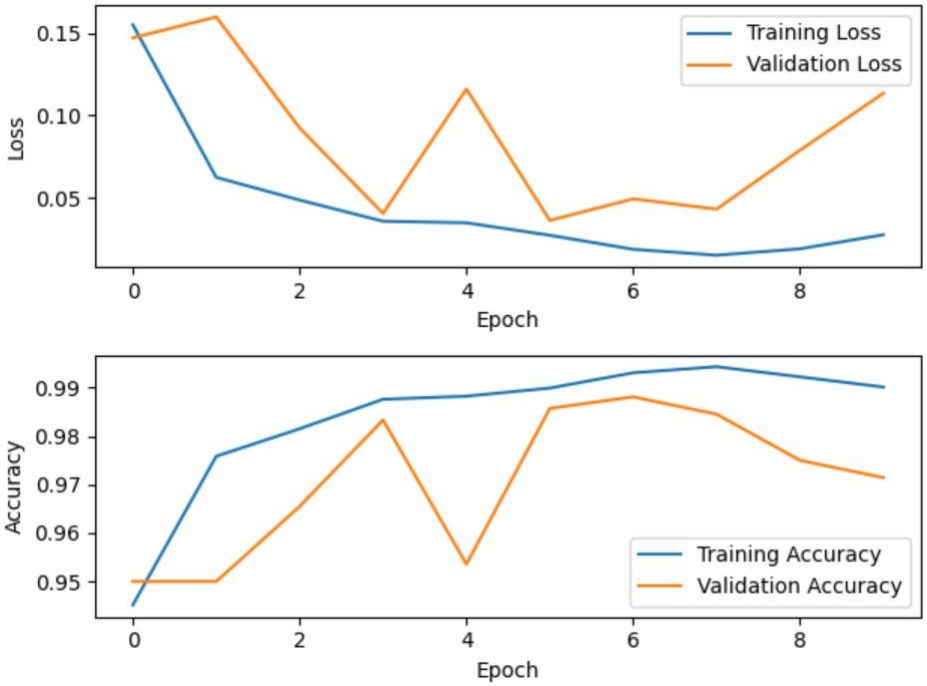
x = keras.layers.GlobalAveragePooling2D()(x)شما از لایه Global Average Pooling استفاده می‌کنید تا ویژگی‌های استخراج شده توسط ResNet-50 را به شکل یک بردار یک بعدی با ابعاد ثابت تبدیل کند

x = keras.layers.Dropout(0.2)(x)شما از Dropout با نرخ ۰.۲ استفاده می‌کنید تا باعث کاهش overfitشدن مدل شود.

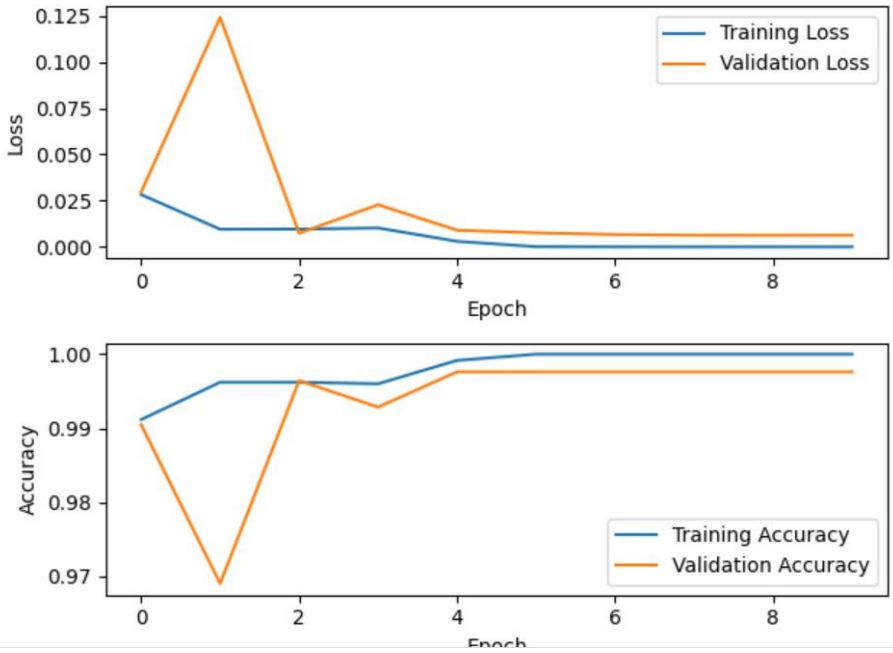
x = keras.layers.Dense(1024, activation='relu')(x)یک لایه Dense با ۱۰۲۴ نرون و تابع فعال‌سازی ReLU اضافه می‌کنم

outputs = keras.layers.Dense(2,activation='softmax')(x)در لایه آخر، من یک لایه Dense با ۲ نرون (برای دسته‌بندی دو کلاس) و تابع فعال‌سازی softmax اضافه می‌کنم تا احتمالات برای هر کلاس را تولید کند.

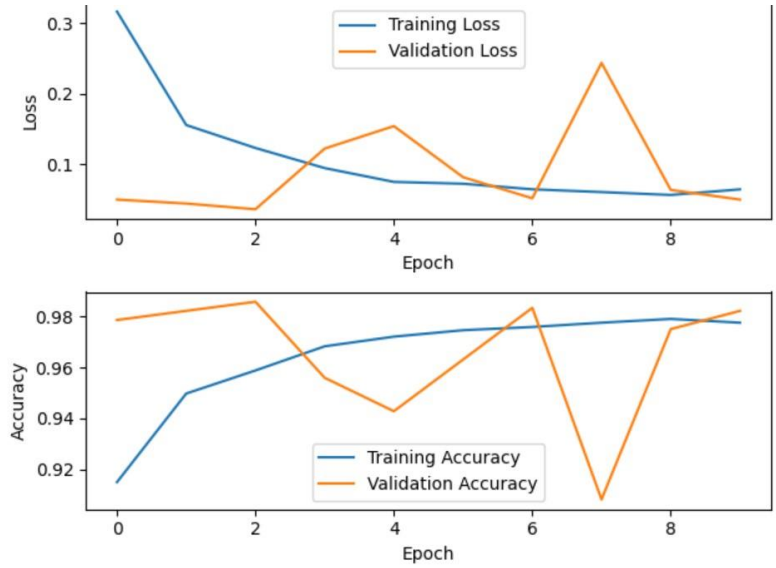
نمودار های مربوط به accuracy,loss به صورت زیر است:



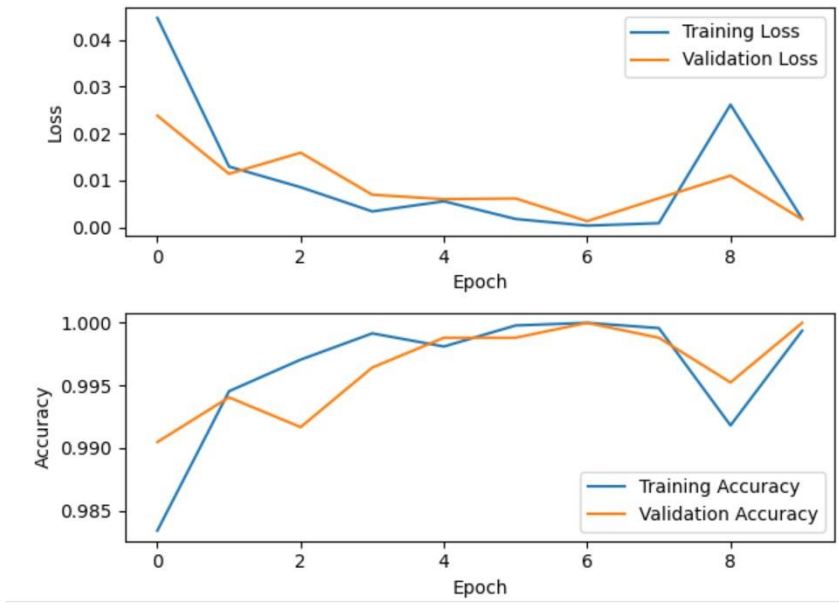
سپس مدل را finetune کردم که نمودارها در این مرحله به صورت زیر شد:



و دقت را بر روی داده های تست نیز ارزیابی کردم وبرابر با 0.9964 شد که بسیار دقت خوبی هست.در ادامه از مدل پیش اموزش دیده VGG16 نیز استفاده کردم که نمودار های ان نیز به صورت زیر شد:



و بعد از finetune کردن نیز نمودارها به صورت زیر شد:



و دقت را بر روی داده های تست نیز ارزیابی کردم که برابر با 0.9993 در این مدل بود.