به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین دوم

نام و نام خانوادگی	آرمان عطاردی – پوریا شیخ الاسلامی
شماره دانشجویی	۸۱۰۱۰۱۲۲۸ - ۸۱۰۱۰۰۳۹۴
تاریخ ارسال گزارش	۱۴۰۲.۰۱.۱۸

پاسخ۱. شبکه عصبی پیچشی کم عمق برای دسته بندی تصاویر

۱-۱. آماده سازی و پیش پردازش داده ها

۱-۲. توضیح لایه های های مختلف معماری شبکه

ابتدا یک لایه کانولوشن ۳ *۳ برای استخراج ویژگی های اولیه تصویر، سپس یک لایه نرمال کننده برای نرمال کردن میانگین و واریانس feature map های استخراج شده و سپس یک لای max pooling ۲*۲ برای کاهش پیچیدگی محاسبات و کاهش ابعاد.

پس از آن یک لایه کانولوشن ۳*۳ با ۶۴ عدد فیلتر برای استخراج ویژگی های عمیق تر و مانند قبل یک لایه نرمال کننده و max pooling .

سپس یک لایه fully connected با ۱۲۸۰ نورون برای دسته بندی ویژگی ها و استفاده از آنها برای طبقه بندی و پس از آن لایه drop out برای جلو گیری از over-fitting و افزایش robustness .

در نهایت یک لایه fully connected دیگر به تعداد کلاس های داده برای اعلام دسته هر عکس.

طبق مقاله مهم ترین لایه، لاهای های کانولوشن برای استخراج ویژگی ها است.

۱-۳. توضیح لایه های های مختلف معماری شبکه

با استفاده از کتاب خانه keras شبکه گفته شده را بدین شکل پیاده سازی می کنیم:

```
model = tf.keras.Sequential([
    tf.keras.layers.Resizing(28,28,interpolation="bilinear"),
    tf.keras.Input(shape = (28,28,1)),

       tf.keras.layers.Conv2D(filters = 32 , kernel_size = (3,3) , strides
 = (1,1) , padding = "same" ,
                           activation = "relu"),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling2D(pool_size = (2,2) , padding = "valid" ) ,

    tf.keras.layers.Conv2D(filters = 256 , kernel_size = (3,3) , strides =
 (1,1) , padding = "same" ,
                           activation = "relu"),
    tf.keras.layers.BatchNormalization(),
```

```
    tf.keras.layers.MaxPooling2D(pool_size = (2,2) , padding = "valid" ) ,

    tf.keras.layers.Flatten(),

    tf.keras.layers.Dense(180, activation='relu'),
    tf.keras.layers.Dropout(0.5),

    tf.keras.layers.Dense(10, activation='softmax')

])
model.compile(loss="categorical_crossentropy" ,
                    optimizer = tf.optimizers.SGD(learning_rate=0.002 ,
momentum=0.9),
                    metrics = ['accuracy'])
my_callbacl = [
            tf.keras.callbacks.EarlyStopping(patience=5)
]
```

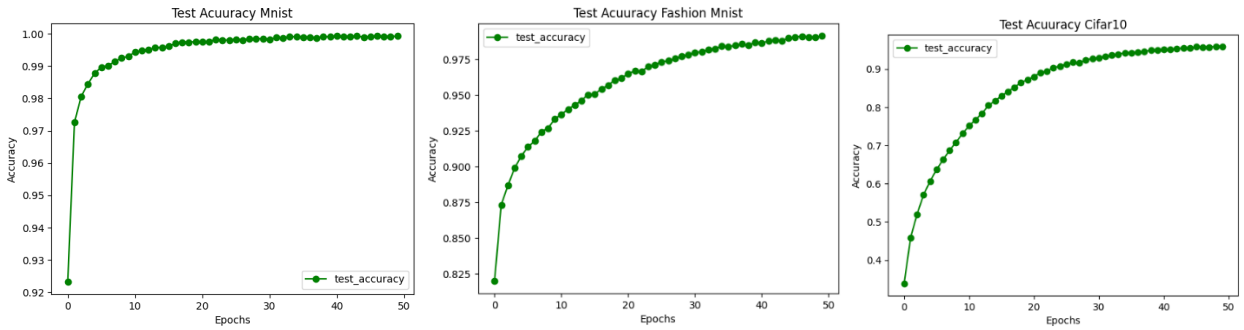
در پایان به انکود کردن لیبل ها برای خروجی one-hot شبکه را آموزش می دهیم.

برای ۲ دیتاست دیگر نیز به همین شکل عمل می کنیم.

به دلیل آنکه داده های cifar10 تصاویری با ابعاد ۳۲*۳۲ بودند و در شبکه پیشنهاد شده داده های ورودی ۲۸*۲۸ ویژگی داشتند در ابتدا از لایه resize استفاده کردیم همچنین یک لایه flatten بین لایه کانولوشن و fully connected استفاده شده که در مقاله به آن اشاره نشده است.

۱-۴. نتایج پیاده سازی

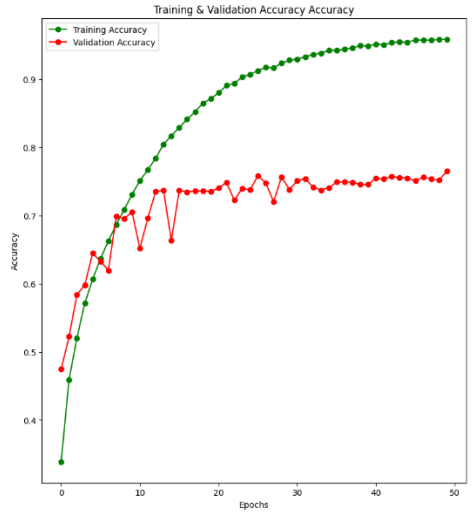
نتایج accuracy برای داده های test به شکل زیر است:

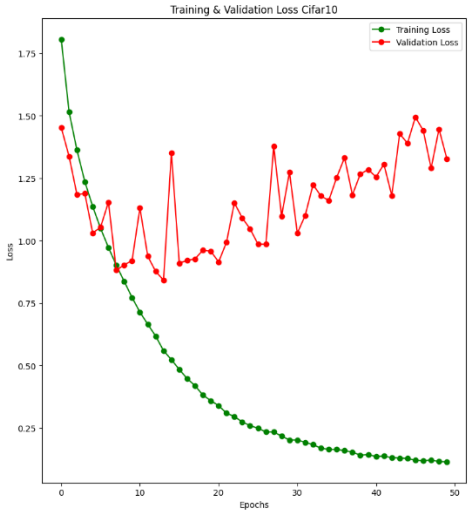


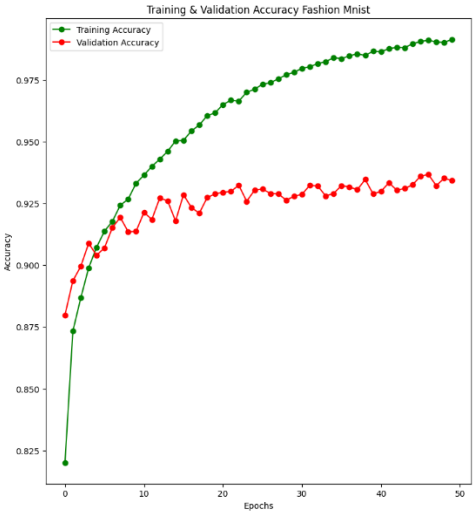
دقت برای هر شبکه نیز به صورت زیر است:

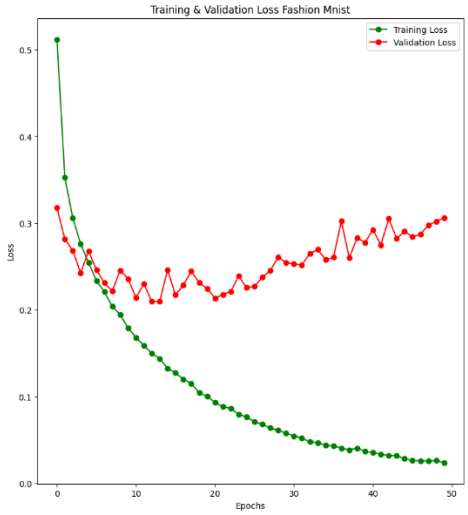
	دقت
شبکه‌ی cifar10	۷۴.۵۷
شبکه‌ی Mnist	۹۹.۳۰
شبکه Fashion Mnist	۹۲.۴۳

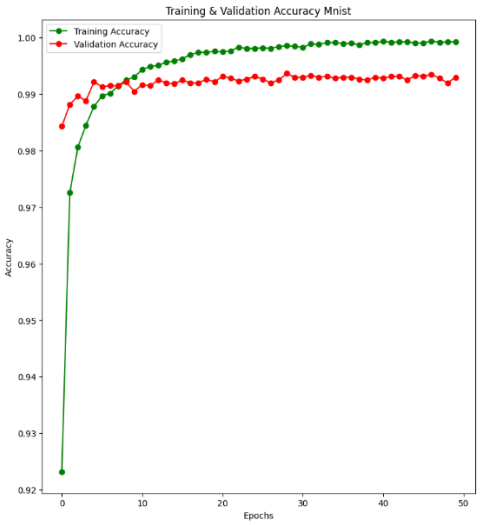
نمودار loss و accuracy برای داده های train به شکل زیر است:

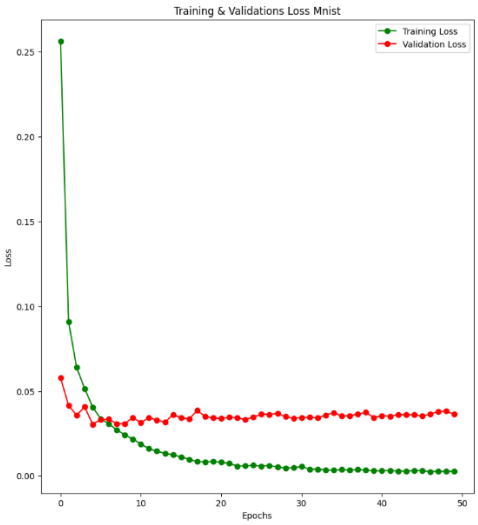












با مقایسه نتایج بدست آمده میبینیم که شبکه برای داده های cifar10 اورفیت شده و دقت آن برای داه های آموزش بالاتر از داده های تست است و قابلیت تعمیم آن کمتر است درحالی که برای داده های mnist و fashion-mnist دقت برای هر دو دسته نزدیک به هم است. همچنین افزایش دقت برای داده mnist به سرعت و در ایپاک های اولیه اتفاق افتاده اما برای داده های cifar10 و fashion-mnist به مرور دقت بیشتر شده.

دقت برای هر دو دسته آموزش و تست برای همه داده ها در هر ایپاک اختلاف داشته و این اختلاف در هر ایپاک به مرور کاهش پیدا کرده یا ثابت مانده است.

پاسخ۲. طبقه بندی تصاویر اشعه ایکس قفسه سینه

۲-۱. آماده سازی و پیش پردازش داده ها

الف- روشهایی که مقاله برای پیش پردازش و افزایش داده استفاده کرده است به شرح زیر است:

1-Re-Scale:

در این روش مقادیر پیکسل ها که به طور پیش فرض در بازه ۰تا ۲۵۵ قرار دارند به بازه ۰ تا ۱ تغییر پیدا می کنند. (همه مقاریر بر ۲۵۵ تقسیم می شوند)

2- Shear Range:

در این روش عکس ها در جهت x یا y به یک سمت متمایل می شوند و به شکل متوازی الاضلاع در می آیند.

3- width shift:

در این روش همه پیکسل ها به سمت راست یا چپ انتقال پیدا می کنند و پیکسل ها جدید با سیاست های مختلفی تولید می شوند.

4- Hight Shift

در این روش همه پیکسل ها به سمت بالا یا پایین انتقال پیدا می کنند و پیکسل ها جدید با سیاست های مختلفی تولید می شوند.

5- Rotation:

در این روش عکس ها مقداری چرخش می کنند.

6- Horizontal Flip:

برعکس کردن عکس به صورت افقی.

7- Zoom:

بزرگنمایی عکس.

۲-۲. توضیح لایه های مختلف معماری شبکه

پس از انجام پیش پردازش ها از شبکه Efficient Net برای پردازش عکس ها و پیدا کردن فیچر های مورد نظر استفاده می کنیم. دلیل استفاده از شبکه Efficient Net فشرده سازی و برآشفته سازی فیچر ها در چندین مرحله است که پیدا کردن فیچرهای مختلف کمک می کند. .همچنین گنجایش بالای این شبکه و درعین حال جمع و جور بودن آن دلیل دیگری برای انتخاب این شبکه است.

پس از آن یک لایه average pooling برای کوچک کردن ابعاد فیچر مپ ها. سپس یک لایه Fully connected برای طبقه بندی کردن فیچر ها، یک لایه Drop out برای افزایش robustness دوباره یک لایه Fully connected و باز هم Drop out و در نهایت یک نورون Fully connected برای دیدن خروجی به شکل ۰ یا ۱.

۲-۳. پیاده سازی شبکه

شبکه گفته شده را به شکل زیر پیاده سازی می کنیم:

ابتدا مدل پایه خود را لود کرده

```
base_model = tf.keras.applications.EfficientNetB2(input_shape=(128,128,3),
                                                   include_top=False,
                                                   weights='imagenet')
base_model.trainable = True
```

سپس لایه های اضافی را روی آن سوار می کنیم:

```
model = tf.keras.models.Sequential()
model.add(base_model)
model.add(tf.keras.layers.GlobalAveragePooling2D())
model.add(tf.keras.layers.Dense(128 , activation='relu'))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.Dense(64 , activation='relu'))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Dense(1 , activation='sigmoid'))
```

در نهایت خلاصه مدل به شکل زیر خواهد بود:

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 efficientnetb2 (Functional)  (None, 4, 4, 1408)       7768569

 global_average_pooling2d_2   (None, 1408)             0
 (GlobalAveragePooling2D)

 dense_6 (Dense)             (None, 128)               180352

 dropout_4 (Dropout)         (None, 128)               0

 dense_7 (Dense)             (None, 64)                8256

 dropout_5 (Dropout)         (None, 64)                0

 dense_8 (Dense)             (None, 1)                 65

=================================================================
Total params: 7,957,242
Trainable params: 7,889,667
                                    Non-trainable params: 67,575
```

همچنین برای augmentation از کد زیر استفاده می کنیم:

```
train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.2,
        width_shift_range=0.2,
        height_shift_range=0.2,
        rotation_range = 30,
        zoom_range=0.2,
        horizontal_flip=True)
val_Datagen = ImageDataGenerator(
        rescale = 1./255
)
train_generator = train_datagen.flow_from_directory(
        '/content/chest_xray/chest_xray/train',
        target_size=(128, 128),
        batch_size=128,
        class_mode='binary')
validation_generator = val_Datagen.flow_from_directory(
        '/content/chest_xray/chest_xray/val',
        target_size=(128, 128),
        batch_size=16,
        class_mode='binary')
```

```
test_generator = val_Datagen.flow_from_directory(
      '/content/chest_xray/chest_xray/test',
       target_size=(128, 128),
        batch_size=16,
        class_mode='binary')
```

۲-۴. نتایج پیاده سازی

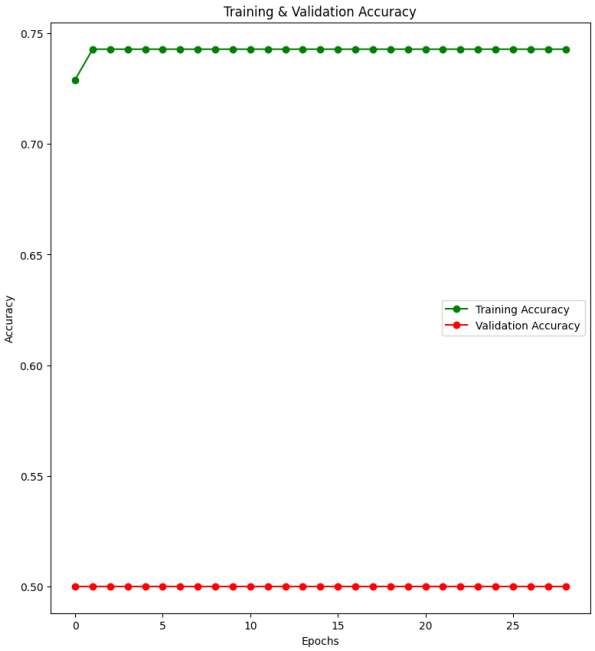
الف) مقادیر خواسته شده به شکل زیر است:

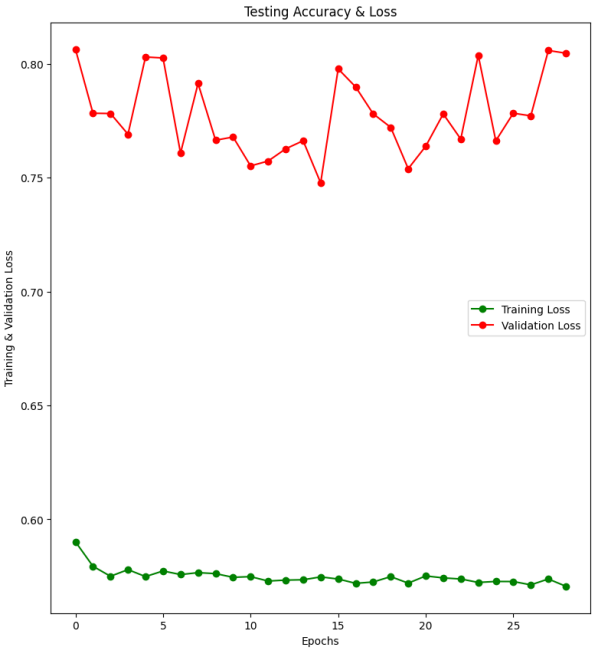
```
accuracy =  0.375
recall =  1.0
F1 =  0.5454545454545454
```

ب) ماتریس درهم ریختگی به شکل زیر است:

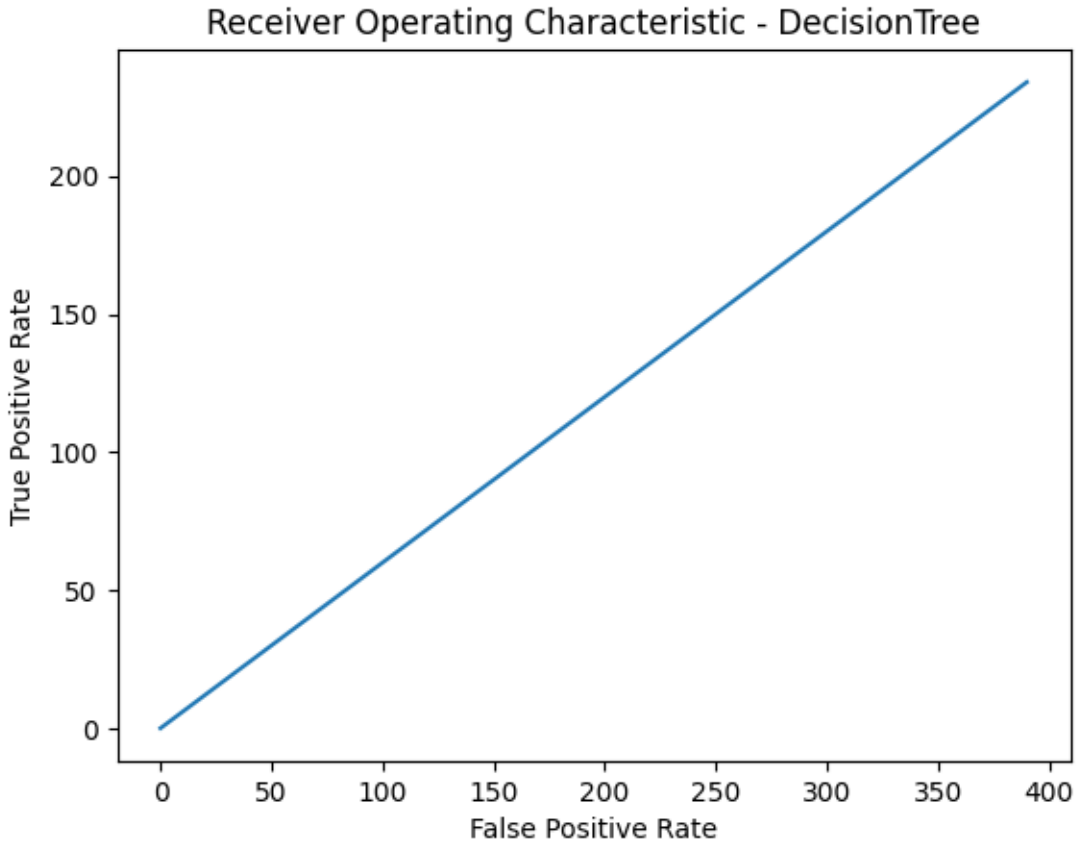
۰	۲۳۴
۰	۳۹۰

نمودار های accuracy و loss برای داده های train و validation به شکل زیر است:





همچنین نمودار ROC نیز به شکل زیر است:



ج) نتایج نشان دهنده overfit شدن داده ها به دیتای train هستند و بنده هر کاری کردم نتونستم این مشکل رو رفع کنم. شبکه همه داده ها را در کلاس ۱ قرار داده . من به ta هم گفتم اما ایشون هم جوابی ندادن به من.