دانشکده مهندسی کامپیوتر

عنوان کارآموزی

محل کارآموزی:

شرکت روشن

نام دانشجو:

فاطمه عسکری

نام استاد کارآموزی:

دکتر محمدی

تابستان ۱۴۰۲


بسم الله الرحمن الرحیم

تأییدیه‌ی صحت و اصالت نتایج

بسمه تعالی

اینجانب فاطمه عسکری به شماره دانشجویی ۹۸۴۷۱۴۱۴دانشجوی رشته مهندسی کامپیوتر مقطع تحصیلی کارشناسی تأیید می‌نمایم که کلیه‌ی مطالب مندرج در این گزارش حاصل حضور و کار اینجانب در شرکت روشن بدون هرگونه دخل و تصرف است و موارد نسخه‌برداری‌شده از آثار دیگران را با ذکر کامل مشخصات منبع ذکر کرده‌ام. درصورت اثبات خلاف مندرجات فوق، به تشخیص دانشگاه مطابق با ضوابط و مقررات حاکم آموزشی، پژوهشی و انضباطی با اینجانب رفتار خواهد شد و حق هرگونه اعتراض درخصوص احقاق حقوق مکتسب و تشخیص و تعیین تخلف و مجازات را از خویش سلب می‌نمایم.

فهرست مطالب

فصل ۱:

معرفی حوزه کارآموزی

۱-۱-مقدمه

در این قسمت به معرفی حوزه کارآموزی پرداخته شده است.در ابتدا یک وضعیت کلی از شرکت مورد نظر ذکر شده وسپس محصولات،مشتری ها،نوع فروش، به صورت خلاصه آورده شده است.

شرکت دانش‌بنیان «راهکار پردازش ژرف» در سال ۱۳۹۵ با تمرکز بر موضوع هوش مصنوعی و با همت گروهی از پژوهشگران و متخصصان جوان، فعالیت خود را آغاز کرد. این شرکت در سال‌های بعد با نام تجاری «روشن» دایره فعالیت‌های خود را گسترش داد و توانست اعتماد بسیاری از کسب‌وکارها و سازمان‌ها را به دست آورد. باعث افتخار است که مجموعه‌هایی نام‌آور و خوش‌نام، محصولات این شرکت را انتخاب کرده‌اند.

۱-۲-تکنولوژی های استفاده شده

برخی از تکنولوژی های مورد استفاده در این شرکت به شرح زیر است:

۱. Python
۲. Tensorflow
۳. Pytorch
۴. Docker
۵. Gitlab
۶. React
۷. Django
۸. Flutter

۱-۳-محصولات و فروش

شرکت روشن در حوزه های زیر فعالیت میکند:

۱. پردازش تصویر و کشف محتوا
۲. تبدیل تصویر به نوشته(OCR)
۳. تبدیل گفتار به متن:با استفاده از هزاران ساعت گفتار با صدای افراد مختلف،زبان فارسی را یاد گرفته و می تواند متن صحبت ها را بنویسد.
۴. تحلیل متن های فارسی:تمیز و مرتب کردن متن،تقطیع جمله هاو واژه ها،ریشه یابی واژه ها،ریشه یابی واژه ها،تحلیل صرفی جمله،تجزیه نحوی جمله

فصل ۲:

مشروح کارهای انجام شده در محل استقرار

۲-۱-مقدمه

در این بخش مجموعه فعالیت هایی که دربازه کارآموزی انجام شده ذکر شده است.همچنین چالش و راهکار هایی که در پروژه انجام داشتم نیز آورده شده است.

۲-۲-واحد کارآموزی

شرکت در دو حوزه نرم افزار و هوش مصنوعی کارآموز می پذیرد من در حوزه هوش مصنوعی و در دو پروژه پردازش تصویر فعالیت داشتم.

۲-۳-پروژه ها

در طول زمان کارآموزی روی دو پروژه سرچ تصویر و تشخیص بیماری سل از روی تصاویر سی تی اسکن فعالیت داشتم.

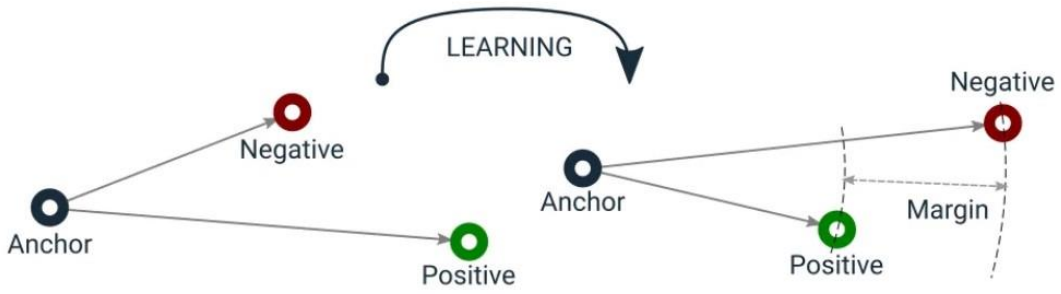
۲-۳-۱-پروژه شباهت سنجی تصویر

در این تسک سرچ تصویر، هدف ما آموزش مدلی است که بتواند برای هر تصویر، یک بردار ویژگی تولید کند که این بردار ویژگی برای تصاویر مشابه بسیار شبیه به هم و برای تصاویر غیر مشابه، دور از هم باشد. سپس با کمک این مدل، می‌توان برای هر تصویر، نزدیک ترین داده ها به بردار ویژگی تصویر ورودی را به عنوان تصاویر مشابه خروجی داد.برای آموزش چنین مدلی، روش‌های زیادی وجود دارد که دو مورد از معروف ترین آنها، آموزش به کمک triplet loss و centered loss است. از آنجایی که آموزش به کمک triplet loss به حجم بالای داده و توان پردازشی بسیار بالا نیاز دارد، در این تسک می‌خواهیم مدلی را به کمک centered loss آموزش دهیم و دقت آن را در یافتن تصاویر مشابه بسنجیم.برای آموزش مدل، از دیتاستی شامل تصویر ۱۰۲ گل مختلف استفاده می‌کنیم.در ابتدا مقدمه ای از روش های موجود برای شباهت سنجی تصاویر می گویم و بعد توضیحی درباره روشی که پیاده سازی کردم میدهم.

Triple loss

هدف Triple loss، ایجاد فضایی است که نماینده‌های هر دسته به طور جداگانه و دسته‌ها از یکدیگر به طور کامل تمایز پیدا کنند. برای این منظور، سه مرجع استفاده می‌شود: مرجع مثبت(positive) ، مرجع منفی (negative)و مرجع مقصد (anchor) مرجع مثبت با تصویر مقصد در یک دسته قرار دارد. مرجع منفی نمونه‌هایی از دسته‌های دیگر است که با تصویر مقصد در یک دسته قرار ندارد.

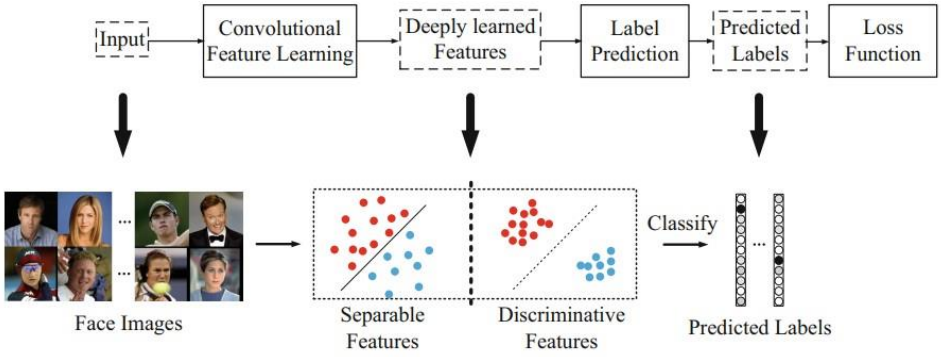
در واقع در روش Triple loss هدف این است که فاصله مقصد تا مرجع مثبت کم بشود و فاصله آن با مرجع منفی زیاد بشود مطابق شکل زیر:



با توجه به اینکه برای triple loss نیاز بود که batch size را عدد نسبتا بالایی بگذاریم و منابع سخت افزاری محدود بود به سراغ پیاده سازی این روش نرفتم.

Center loss

برای آشنایی با این تابع ضرر ابتدا مقاله " A Discriminative Feature Learning Approach for Deep Face Recognition" را مطالعه کردم و متوجه شدم علاوه براینکه باید classify کنیم باید سعی کنیم تا فیچرهای هر کلاس را به هم نزدیک کنیم و برای اینکار تابع ضرر cross entropy به تنهایی کافی نیست و باید از یک تابع ضرر دیگر نیز استفاده کنیم شکل زیر نشان میدهد که ما دنبال چه چیزی هستیم:



در مقاله نیز اشاره میشود استفاده از تابع ضرر های contrastive loss و triple loss باید تعداد جفت هایی که برای train میگیریم زیاد باشد و به کندی مدل همگرا میشود.

هدف اصلی روشCenter Loss ، ایجاد فضایی است که نماینده‌های هر دسته به طور جداگانه و دسته‌ها از یکدیگر به طور کامل تمایز پیدا کنند. برای این منظور، از مفهوم مرکز (center) برای هر دسته استفاده می‌شود.

مرکز یک دسته برابر با میانگین نمونه‌های آن دسته در فضای ویژگی‌ها است. به عبارت دیگر، برای هر دسته یک مرکزی تعیین می‌شود که نماینده آن دسته است.در واقع center ها نیز در فرایند آموزش آپدیت میشوند.

روش Center Loss معمولاً با تابع ضرر Cross-Entropy Lossکه در آموزش شبکه عصبی عمومی استفاده می‌شود، ترکیب می‌شود. تابع ضرر Center Lossقصد دارد فاصله بین ویژگی‌های استخراج شده از تصاویر و مراکز متناظر دسته‌ها را کاهش دهد. با کاهش این فاصله، نماینده‌های هر دسته به طور جداگانه و دسته‌ها از یکدیگر به طور کامل تمایز پیدا می‌کنند.

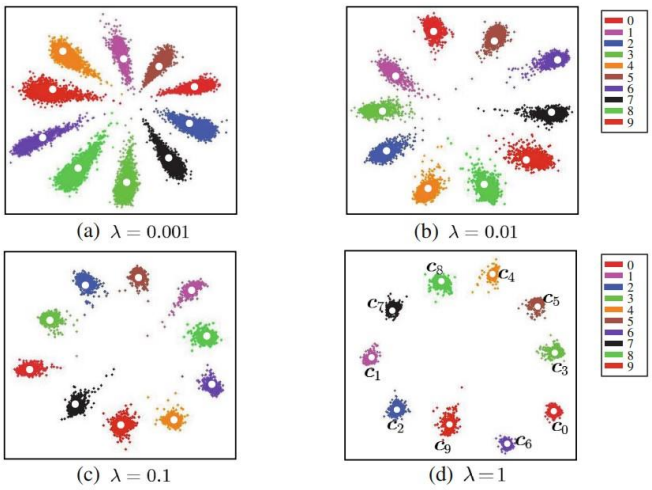
فرمول center loss به صورت زیر است:

$$\mathcal{L}_C = \frac{1}{2}\sum_{i=1}^{m} \|\boldsymbol{x}_i - \boldsymbol{c}_{y_i}\|_2^2$$

منتها ما center loss را با cross entropy ترکیب می کنیم به این علت که اگر فقط از فرمول بالا استفاده کنیم احتمالا همه center ها رو هم میفتند و classify انجام نمیشه و اگر فقط از cross entropy استفاده کنیم discrimitive انجام نمیشه فرمول ترکیب این دو تابع ضرر به صورت زیر است:

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_S + \lambda \mathcal{L}_C \\ &= -\sum_{i=1}^{m} \log \frac{e^{W_{y_i}^T \boldsymbol{x}_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T \boldsymbol{x}_i + b_j}} + \frac{\lambda}{2}\sum_{i=1}^{m} \|\boldsymbol{x}_i - \boldsymbol{c}_{y_i}\|_2^2\end{aligned}$$

پارامتر لاندا اهمیت بسیار زیادی در discrimitive کردن دارد در تصویر زیر اهمیت آن مشخص است:



پیاده سازی

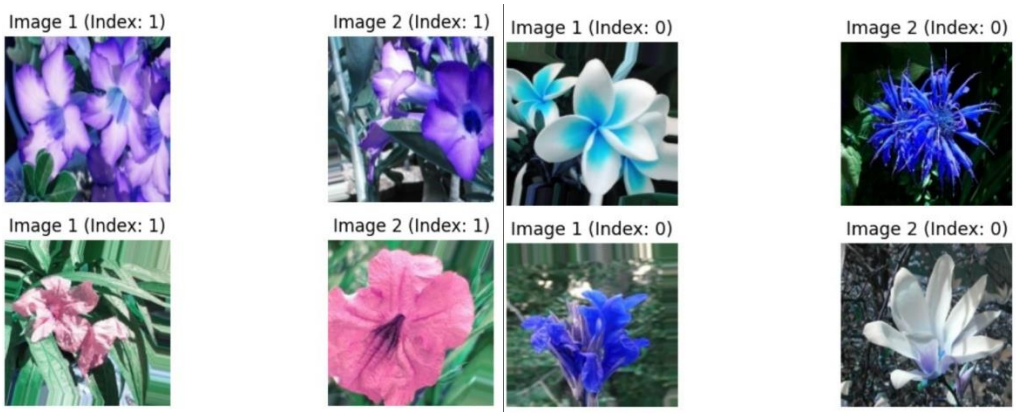
نمونه ای از تصاویر دیتاست ما به صورت زیر است:

برای ارزیابی اینکه دیتاست ما چگونه عمل میکند تصمیم گرفتیم از دیتاست تصاویر گل خود یک دیتاست مانند دیتاست LFW بسازیم به این صورت که یکسری pair داشته باشیم نصف انها جفت هایی باشد که در یک کلاس هستند و نصف انها تصاویری باشد که در یک دسته نیستند و بعد با معیار ROC-AUC که در ادامه توضیح خواهم داد بتوانیم تفاوت discrimitive کردن قبل و بعد از استفاده از Center loss بسنجیم.

درست کردن دیتاست LFW

در دیتاست ما ۱۰۲ دسته گل داریم من ۲۰ کلاس به داده های تست اختصاص دادم و ۸۲ کلاس به داده های trainو validation اختصاص دادم و با آن ۲۰ کلاس دیتاست LFW را ساختم به این صورت که ۵۰۰۰ جفت ان لیبل ان برابر با ۱ هست و ۵۰۰۰ تای ان لیبل ان برابر با ۰ است لیبل ۱ یعنی دو تصویر در یک دسته قرار دارند و لییبل ۰ یعنی دو تصویر در یک دسته نیستند در ابتدا با استفاده از augmentation تعداد گل های هر دسته را به ۲۵۰ رساندم و بعد برای تشکیل ۵۰۰۰ جفت با لیبل ۱ امدم و در هر دسته ایندکس اول با ایندکس دوم،ایندکس دوم را با اینکس سوم و....ایندکس اخر را با ایندکس اول جفت کردم و چون ۲۰ دسته گل داریم و در هر دسته ۲۵۰ تا جفت داریم جمعا ۵۰۰۰ تا جفت با لیبل ۱ تشکیل میشود.برای تشکیل ۵۰۰۰ تا جفت با لیبل ۰ در ابتدا از هر دسته ۲۵ تصویر را به صورت رندوم انتخاب کردم و در ادامه ایندکس n ام هر دسته را با

ایندکس n ام دسته های دیگر جفت کردم.به عنوان نمونه در سمت چپ دو جفت زیر دو تصویر از یک دسته هستند و در سمت راست از یک دسته نیستند.



تغیر سایز تصاویر

یکی از مشکلاتی که داشتم این بود که سرعت لود تصاویر به علت تعداد زیاد و سایز تصاویر بسیار کند بود به همین علت تصمیم گرفتم تا تصاویر را به ۷۲ در ۷۲ ریسایز کنم و به صورت فایل pickle ذخیره کنم.

استخراج بردارهای ویژگی

در ابتدا بدون هیچ train بردارهای ویژگی از مدل پیش آموزش دیده Resnet50 استخراج می کنیم(طول بردار ویژگی ۲۰۴۸ است.) و بعد با استفاده از cosin_similarity میزان شباهت را حساب می کنیم مانند کد زیر:

```
def loadmodel(imgs):
  model = tf.keras.applications.ResNet50(weights='imagenet', include_top=False, pooling='avg'

  def extract_features(img):
      x = tf.keras.preprocessing.image.img_to_array(img)
      x = tf.expand_dims(x, axis=0)
      x = tf.keras.applications.resnet50.preprocess_input(x)
      features = model.predict(x)
      print(features.shape)
      return features


  similarities = np.zeros(len(imgs))
  for i in range(len(imgs)):
      features1 = extract_features(imgs[i][0])
      features2 = extract_features(imgs[i][1])
      similarity = cosine_similarity(features1, features2)
      similarities[i] = similarity[0, 0]
      print(i)
```

و در ادامه دقت ROC-AUC را حساب می کنیم یک توضیحی در رابطه با ROC-AUC در ادامه می دهم

معیار ROC-AUC

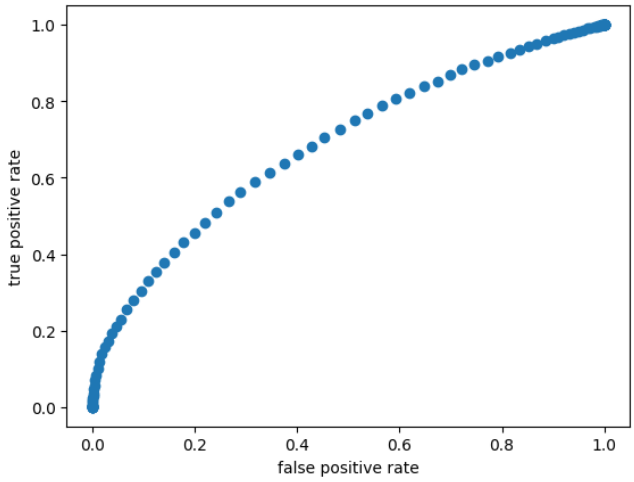
معیار ROC-AUC (Receiver Operating Characteristic - Area Under the Curve) یک معیار ارزیابی عملکرد برای مدل‌های دسته‌بندی باینری است. این معیار بر اساس منحنی ROC ساخته شده توسط مدل بر روی مجموعه داده آزمون (test set) محاسبه می‌شود.منحنی ROC یک نمودار دوبعدی است که نشان می‌دهد که مدل در مختلف آستانه‌های (thresholds) تصمیم‌گیری، چه تعادلی بین نرخ تشخیص درست True) Positive Rate)و نرخ تشخیص نادرست (False Positive Rate) دارد. محور افقی منحنی ROC نمایش‌دهنده False Positive Rate (FPR) است که نسبت تعداد نمونه‌هایی که اشتباهاً به عنوان مثبت تشخیص داده شده‌اند به تعداد کل نمونه‌های منفی است. محور عمودی نمایش‌دهنده True Positive Rate (TPR)یا همان نرخ تشخیص درست است که نسبت تعداد نمونه‌هایی که به درستی به عنوان مثبت تشخیص داده شده‌اند به تعداد کل نمونه‌های مثبت است.

معیار AUC نیز مساحت زیر منحنی ROC را نشان می‌دهد و ارزیابی می‌کند که مدل چقدر توانایی تفکیک بین نمونه‌های مثبت و منفی را دارد. مقدار AUC بین ۰ تا ۱ قرار می‌گیرد، که یک مدل با AUC برابر با ۱ به معنای دقت بسیار بالا در تشخیص بین دو کلاس است، در حالی که AUC برابر با ۰.۵ نشان‌دهنده یک مدل تصادفی است و AUC کمتر از ۰.۵ به معنای یک مدلی است که عملکرد نسبت به تصادف بدتر است.

بنابراین، ROC-AUCمعیاری است که ارزیابی می‌کند که مدل چقدر توانایی تمایز بین کلاس‌های مثبت و منفی را دارد، و با محاسبه مساحت زیر منحنیROC ، این توانایی را به صورت عددی بیان می‌کند.

محاسبه ROC-AUC بر روی دیتاست test

بازه بین ۰ تا ۱ به ۱۰۰ بازه مساوی تقسیم کردم و با این۱۰۰ مقدار threshold مقادیر tp,fp,tn,fn را محاسبه کردم بعد با محاسبه tpr,fpr نمودار ROC را کشیدم ومساحت زیر نمودار را محاسبه کردم.



مساحت زیر نمودار در شکل بالا برابر با 0.6863 است ما می خواهیم تاثیر استفاده از center loss را بر روی ROC پس از استفاده مشاهده کنیم.

پیاده سازی center loss

برای پیاده سازی center loss از دو روش استفاده کردم ولی در نهایت به خاطر اینکه میخواستم چندتا custom metric بنویسم روش دوم را انتخاب کردم.

روش اول پیاده سازی center loss

در روش اول با استفاده از لایه embedding بردارهای مربوط به هر center را استخراج میکرد که کد آن به صورت زیر است:

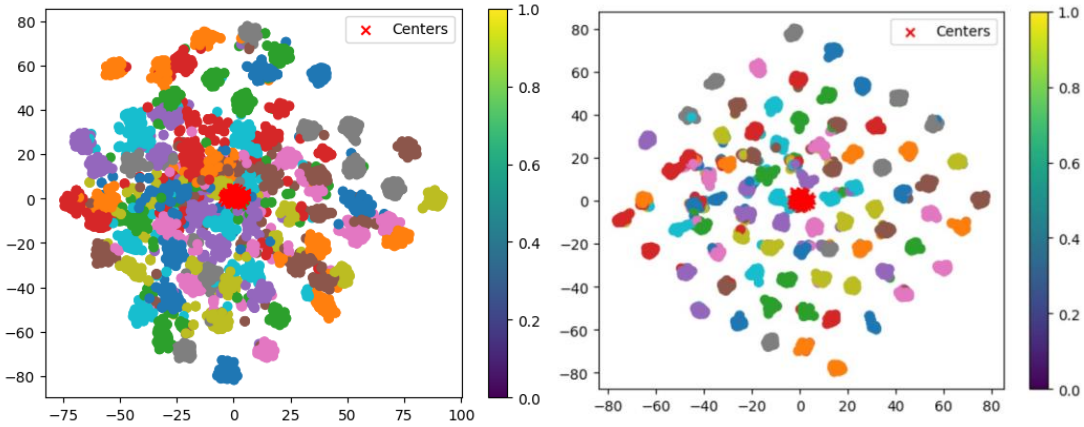
```
num_classes = 80
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(72, 72, 3))

base_model.trainable = False
x = base_model.output
out1 = GlobalAveragePooling2D()(x)
print(out1 .shape[1])
out2 = Dense(num_classes, activation='softmax')(out1)
model = Model(inputs=base_model.input, outputs=out2)


lambda_c=1
input_=Input(shape=(1,))
centers=Embedding(num_classes,out1 .shape[1])(input_)
intra_loss=Lambda(lambda x:K.mean(K.square(x[0]-x[1][:,0]),1,keepdims=True))([out1,centers])
model_center_loss=Model([base_model.input,input_],[out2,intra_loss])
model_center_loss.compile(optimizer="adam",
                          loss=["categorical_crossentropy",lambda y_true,y_pred:y_pred],
                          loss_weights=[1,lambda_c/2.],
                          metrics=["acc"])
model_center_loss.summary()
```

که مدل دو ورودی و دو خروجی دارد و یک بار نیز با استفاده از مدل Resnet50 و تابع ضرر بالا train کردم و بعد از آن fine-tune کردم که نتایج را گزارش میدهم.

قبل از train کردن تصمیم گرفتم بردار فیچرهایی که از مدل پیش اموزش دیده Resnet50 استخراج کردیم یک نمایشی در دو بعد داشته باشم.با استفاده از با استفاده از TSNE در کتابخانه sklearn اینکار انجام دادم نمودار ها را قبل و بعد از train به صورت زیر است نودار سمت چپ برای قبل از train است و نمودار سمت راست برای بعداز train است.:



مشاهده میکنید که کلاس ها در شکل سمت راست discrimitive شده اند.

مساخت زیر نمودار ROC برابر با 0.72163 شد که نزدیک به ۶ درصد نسبت به حالت قبل از استفاده از center loss بیشتر شده است.

روش دوم پیاده سازی center loss

تصمیم گرفتم loss را به صورت یک کلاس تعریف کنم که کد آن به صورت زیر است:

```python
class CenterLoss(losses.Loss):
    def __init__(self, num_classes, feature_dims, alpha=0.5, reduction=losses.Reduction.AUTO
        super(CenterLoss, self).__init__(reduction=reduction, name=name)
        self.num_classes = num_classes
        self.feature_dims = feature_dims
        # hyper parameter for updating the center point
        self.alpha = alpha
        self.centers = tf.Variable(tf.zeros(shape = (self.num_classes, self.feature_dims)))

    def call(self, y_true, y_pred):
        """
            y_ture : same teacher signal as for classification (1-hot vector)
                     shape = (batch_size, num_classes)
            y_pred : output of features in the middle layer of the model
                     shape = (batch_size, feature_dims)
        """
        labels = tf.argmax(y_true, axis=-1)
        centers_batch = tf.gather(self.centers, labels)
        diff = centers_batch - y_pred
        loss = tf.reduce_mean(tf.square(diff))
        unique_label, unique_idx, unique_count = tf.unique_with_counts(labels)
        appear_times = tf.gather(unique_count, unique_idx)
        appear_times = tf.reshape(appear_times, [-1, 1])
```

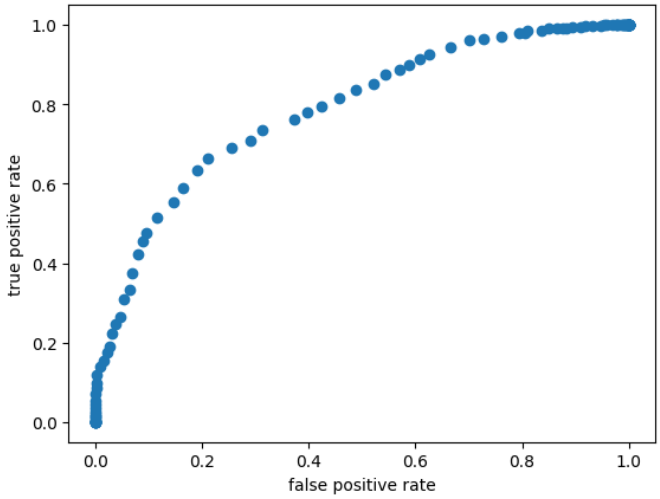
در ابتدا طول هر بردار center خود را برابر با feature_dims و همگی مقادیر را برابر با صفر گذاشتیم در ادامه در تابع call با استفاده از gather سنترها را بر اساس لیبل ها مرتب کردیم و فاصله center هر کلاس را تا بردار های مربوط به ان کلاس حساب کردیم و بعد برای اینکه بخواهیم میانگین بگیریم حساب م کنیم که هر از هر کلاس چند instance داریم و در انتها برای اپدیت کردن center ها از تابع compat.v1.scatter_sub استفاده می کنیم.

پیاده سازی دو custom metric

فاصله اقلیدسی بردار فیچر هر instance تا center را حساب کنیم و میانگین ان را به عنوان یک متریک در نظر بگیریم که در تابع each_img_distance پیاده سازی کردم.متریک دوم میانگین فاصله center ها از هم هست.هدف از تعریف این دو متریک این هست که مشاهده کنیم در فرایند آموزش فاصله هر instance تا center اش نسبت به فاصله center ها از هم باید کاهش یابد.

نتایج

برای مدل پیش آموزش دیده Resnet50نمودار ROC-AUC به صورت زیر شد:

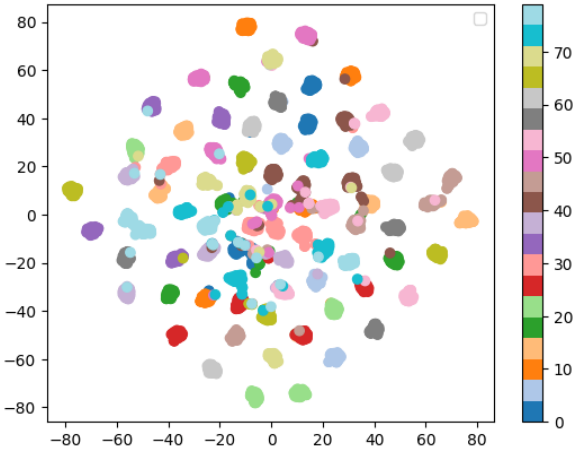


و مساحت زیر نودار ان برابر با 0.786918 شد و هم چنین نتایج متریک ما نیز به صورت زیر شد:

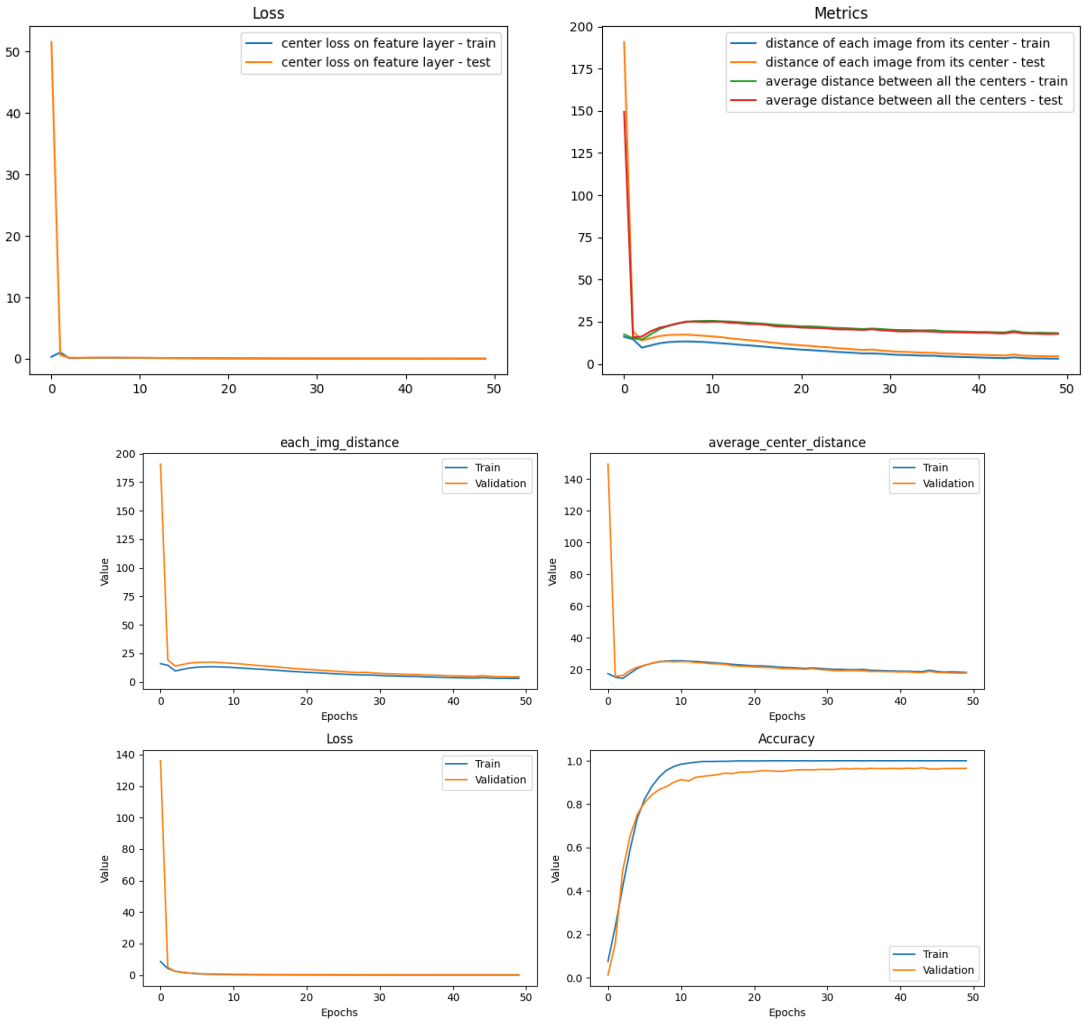
```
features_each_img_distance: 10.5428 - features_average_center_distance: 24.0265
features_each_img_distance: 10.1515 - features_average_center_distance: 23.6949
features_each_img_distance: 9.6418 - features_average_center_distance: 23.2116
features_each_img_distance: 9.2044 - features_average_center_distance: 22.8939
features_each_img_distance: 8.8625 - features_average_center_distance: 22.5542
features_each_img_distance: 8.4368 - features_average_center_distance: 22.2137
features_each_img_distance: 8.1786 - features_average_center_distance: 22.2213
features_each_img_distance: 7.8014 - features_average_center_distance: 22.0082
features_each_img_distance: 7.5057 - features_average_center_distance: 21.6062
features_each_img_distance: 7.0865 - features_average_center_distance: 21.2946
features_each_img_distance: 6.8212 - features_average_center_distance: 21.1552
features_each_img_distance: 6.5414 - features_average_center_distance: 20.8829
features_each_img_distance: 6.1576 - features_average_center_distance: 20.6334
```

مشاهده میکنید که میزان کاهش فاصله کلاس ها از center خودشون خیلی بیشتر از کاهش میانگین center ها از هم هست.

و هم چنین بردار فیچرها را برای داده های validation در فضا دو بعد رسم کردم به صورت زیر شد:



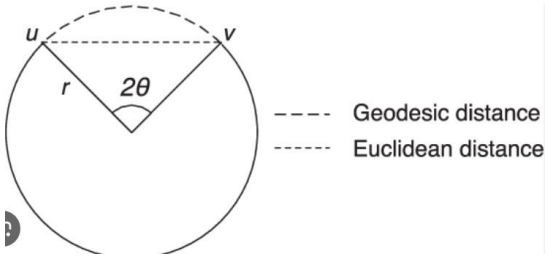
نمودار metric ها نیز به صورت زیر شد:



استفاده از ArcFace loss

استفاده از تابع ضرر center loss یه بدی هایی هم داره از جمله اینکه حساب کردن فاصله اقلیدسی بردار فیچر تمامی instance ها تا center هاشون زمان بر هست و هم چنین ما در طول train هم بایدcenter ها رو اپدیت کنیم و از قبل نمی تونیم مقدار دقیق center ها رو محاسبه کنیم که این نیز زمان بر هست.

یکی دیگه از تابع ضررها که استفاده میکنند arcface هست که به جای فاصله اقلیدسی از فاصله geodesic استفاد میکنه شکل زیر فرق این دو فاصله را به خوبی نشان میدهد:



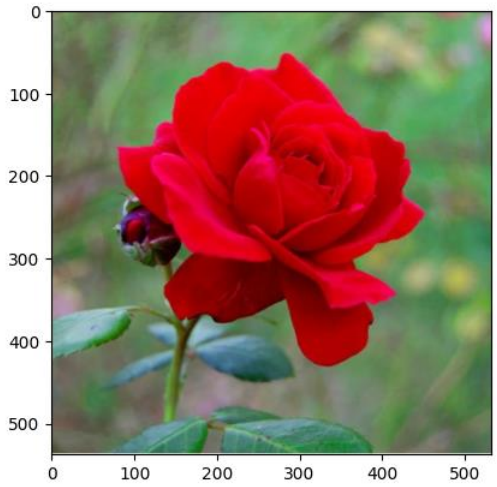
و فرمول آن به صورت زیر است:

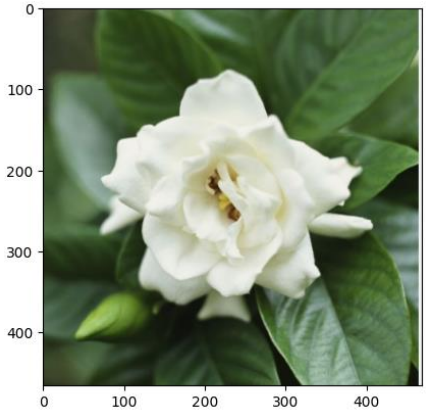
$$-\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{s*(\cos(\theta_{y_i}+m))}}{e^{s*(\cos(\theta_{y_i}+m))}+\sum_{j=1,j\neq y_i}^{n} e^{s*cos\theta_j}}$$

where θ_j is the angle between the weight W_j and the feature x_i
s - feature scale, the hypersphere radius
m - angular margin penalty

مشاهده چندین تصویر query

در انتها چندین تصویر در اینترنت به عنوان تصویر کوئری دادم تا ۱۰ تا از تصویر های که شبیه ترین به تصویر هستند در دیتاست نشان دهد نتایج به صورت زیر شد:


0
100
200
300
400
500
0
100
200
300
400
500


0
100
200
300
400
0
100
200
300
400


0
100
200
300
400
0
100
200
300
400
500
600

۲-۳-۲-پروژه تشخیص بیماری سل

در این پروژه هدف این هست از روی عکس های سی تی اسکن ریه تشخیص دهیم که فرد مبتلا به بیماری سل هست یا نه چالشی که در این پروژه داریم این هست که دیتاست ما بالانس نیست یعنی ۳۵۰۰ عکس برای عکس مربوط به کسانی است که به بیماری سل مبتلا نیستند و ۷۰۰ عکس مربوط به کسانی هست که به بیماری سل مبتلا هستند.برای مقابله با imbalance بودن دیتاست میتوانیم راه های زیر را برویم:

۱. Weighted Class Approach

در این روش، وزن‌های مختلفی به نمونه‌های داده اختصاص داده می‌شود تا تأثیر کلاس‌های کمتر موجود در دیتاست را افزایش دهد.

۲. Under-sampling approach

در این روش، تعداد نمونه‌های کلاس اکثریت کاهش می‌یابد تا به تعداد نمونه‌های کلاس کمتریت نزدیک شود. این روش منجر به کاهش داده‌های بیشماری می‌شود و ممکن است اطلاعات مهمی را از بین ببرد.

۳. Data Augmentation for Minority Class

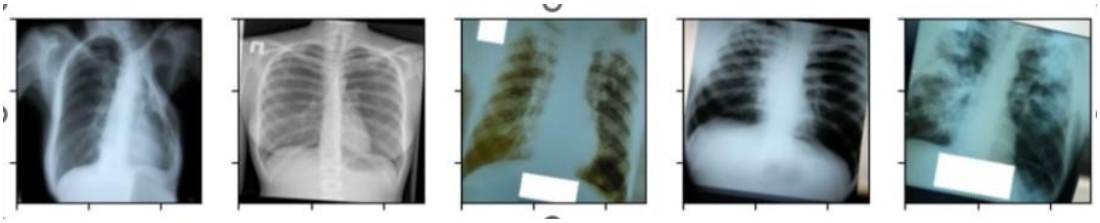
در این روش، نمونه‌های کلاس اقلیت با استفاده از تکنیک‌های تکثیر داده (مانند تغییرات کوچک در تصاویر، چرخش، بزرگنمایی، کاهش مقیاس و ...) تکثیر می‌شوند.

۴. SMOTE (Synthetic Minority Over-sampling Technique)

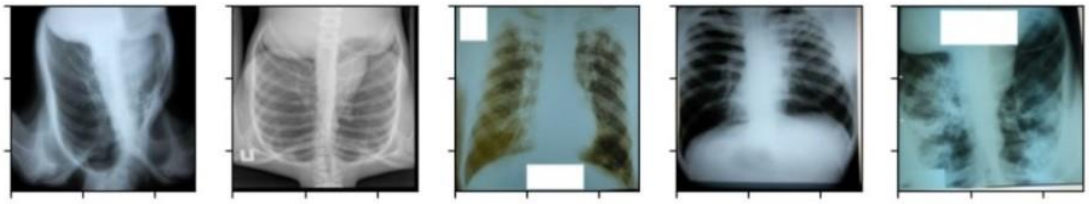
SMOTEیک روش پرکاربرد برای تولید نمونه‌های مصنوعی کلاس اقلیت است. در این روش، نمونه‌های جدید بر اساس ترکیب وزن‌دار نمونه‌های موجود از کلاس اقلیت تولید می‌شوند. این نمونه‌های جدید به عنوان نمونه‌های جدید در دیتاست اضافه می‌شوند و تعداد نمونه‌های کلاس اقلیت را افزایش می‌دهند.

من با توجه مقالات که درباره عکس های سی تی اسکن ریه بود تصمیم گرفتم برای مقابله با نامتعادل بودن دیتاست از روش داده افزایی استفاده کنم.

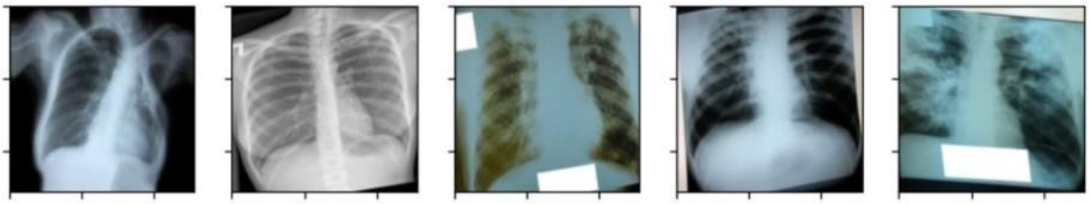
در ابتدا عکس های که مربوط به سل بود را از -۱۵ تا ۱۵ درجه به صورت رندوم با یک زاویه چرخانده میشوند از ImageDataGenerator استفاده کرد و هم چنین عکس چرخید شده را به ۰-۲۵۵ اسکیل کردم.



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:



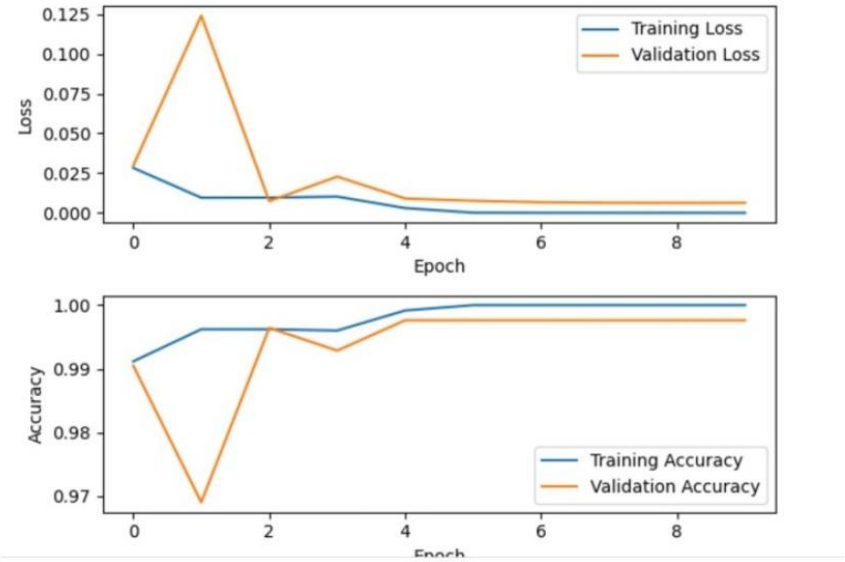
و بعد از این چهارتا داده افزایی این چهارتا نامپای array را با داده اصلی عکس های سل contact کردم و تعداد تصاویر سل نیز برابر با ۳۵۰۰ شد و در درایو ذخیره کردم.برای عکس های نورمال نیز من ۳۵۰۰ عکس ان را به ۵ قسمت مساوی ۷۰۰ تایی تقسیم کردم و روی چهارتا از دسته های ۷۰۰ تایی ان نیز این چهارتا عملی که بر روی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

انتخاب مدل

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.

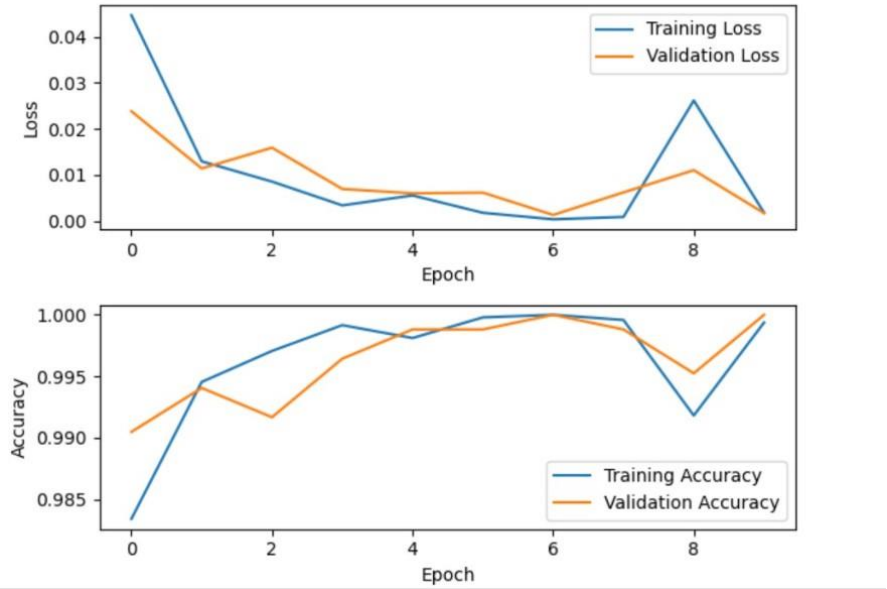
من در ابتدا مدل Densenet را پیاده سازی کردم ولی وقتی رو دیتاست train کردم به نتایج خوبی نرسیدم.به همین علت به سمت transfer learning رفتم.روی دو مدل VGG16 و Resnet50 تست کردم وfine tune نیز کردم و نتایج به صورت زیر شد:

نمودار برای مدل Resnet50 :



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:



در این مدل بر روی داده های تست به دقت 0.9993 رسیدم.

در این مساله نتایج استفاده از شبکه پیش آموزش دیده VGG16 بهتر از Resnet50 بود.

فصل ۳:

نتیجه گیری و پیشنهادها

۳-۱-مقدمه

فعالیت من در این شرکت باعث آشنایی بیشتر با keras و یادگیری pytorch شد و افزایش مهارت مقاله خواندندم شد هم چنین باعث آشنایی با معماری های مختلف مهارت پیاده سازی شد.

۳-۲-محتوا

در این بخش به معرفی فعالیت های انجام شده و پیشنهاداتی برای رفع مشکلات داده شده است.

۳-۲-۱-خلاصه فعالیت های انجام شده

به صورت خلاصه در مدت روی دو پروژه در حوزه بینایی کار شد در پروژه شباهت سنجی توصیر هدف این بود که تاثیر center loss را در discrimitive کردن کلاس ها مشاهده کنیم که از چند طریق بهبود را نشان دادم و در پروژه دوم هدف کار کردن با دیتاست imbalance بود و راهکارهای مقابله با آن بود.

۳-۳-اعلام پیشنهادهایی برای رفع چالش های حوزه کارآموزی

در دوره کارآموزی، یکی از مسائلی که با آن مواجه بودم مشکلات سخت‌افزاری بود که تأثیر زیادی بر دقت پروژه شباهت سنجی تصویر داشت. مشکل اصلی کمبود منابع GPU بود که باعث شد تا عملیات پردازش تصاویر با دقت کافی انجام نشود. به دلیل این محدودیت، مجبور شدم تصاویر را ریسایز کنم تا بتوانم آنها را پردازش کنم. این کاهش اندازه تصاویر به دلیل از دست رفتن جزئیات و اطلاعات مهم، باعث کاهش دقت در شباهت سنجی تصاویر شد. به طور کلی، مشکلات مربوط به منابع سخت‌افزاری، به ویژه کمبودGPU ، می‌توانند تأثیر قابل توجهی بر روی پروژه های مربوط به هوش می گذارد.