گزارش پروژه سوم درس شبکه های عصبی مصنوعی

محمدرضا یادگاری

۱ مقدمه

در این پروژه سعی شده است تا با استفاده از شبکه های خودکدگذار [1] بر روی مجموعه داده ایی شامل تعدادی تصاویر غیر رنگی از چهره انسان همراه با تصاویر نقاشی شده آنها بتوانیم با توجه به تصویر ورودی، تصویر نقاشی شده چهره آن فرد را تولید کنیم. در این پروژه از شبکه Convolutional Autoencoder استفاده شده است که ساختار کلی آن را در شکل زیر مشاهده می‌کنید:

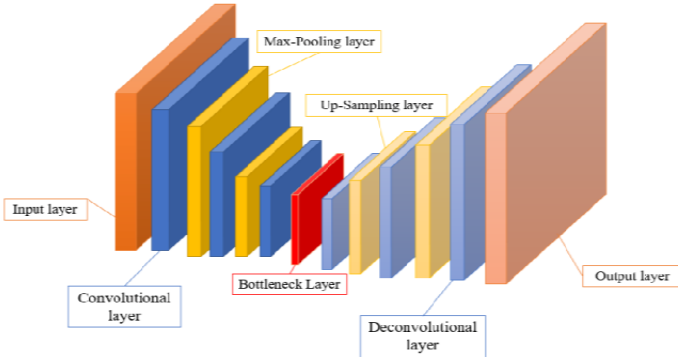

Fig. 1 Representation of Convolutional Autoencoder

[1] AutoEncoders

شبکه های Convolutional Autoencoder از دو بخش Encoder و Decoder تشکیل شده است که در هر دو بخش ذکر شده از ساختار کانولوشنی استفاده می‌کنیم. در مرحله Encoding ، داده به بعد پایین تر برده می‌شود.در واقع با این کار چکیده و عصاره ویژگی های داده را بدست می‌آوریم.در مرحله Decoding ، داده دوباره به بعد بالاتر برده می‌شود و با این کار می‌توانیم تصاویر مختلفی از روی تصویر ورودی بسازیم.مثلا می‌توانیم تصویر ورودی را دوباره بازنمایی کنیم.ما در این پروژه سعی داریم با تعریف تابع خطا و در طول فرآیند یادگیری به شبکه بفهمانیم که باید از روی تصویر غیر رنگی چهره یک انسان، تصویر نقاشی شده(سیاه و سفید) آن را بسازد.

۲ پیش پردازش داده ها(تصاویر)

مجموعه داده ما متشکل از ۱۱۹۴ تصویر غیر رنگی از چهره افراد مختلف به همراه تصاویر نقاشی شده آن ها است.سایز تصاویر غیر رنگی 384 x 256 است و سایز تصاویر نقاشی شده برابر 816 x 767 است. با استفاده از کتابخانه cv2 ابتدا سایز تصاویر را به 256 x 256 تبدیل می‌کنیم تا حجم محاسبات را کم کنیم. سپس هر پیکسل از یک تصویر را بر عدد ۲۵۵ تقسیم میکنیم. با این کار تمام پیکسل های یک تصویر به عددی بین ۰ و ۱ Scale میکنیم.این کار باعث می‌شود مدل سریع تر و دقیق تر آموزش ببیند و Generalization مدل افزایش یابد.
چند باری تلاش کردم که Data Augmentation را انجام بدهم و حداقل از هر عکس Flip آن را به دیتاست اضافه کنم اما متاسفانه در هر مرتبه، colab کرش می‌کرد و در آخر مجبور شدم بدون data augmentation مدل را اموزش بدهم.
با این حال یکبار موفق شدم که از هر تصویر flip اش را به دیتاست اضافه کنم و مدل را آموزش دهم اما نتایج خروجی با نتایج زمانی که دیتا آگمنتیشن انجام نداده بودم تقریبا یکسان بود و حتی کمی هم بدتر بود(نتایج بهترین مدل در بخش های بعدی گزارش شده اند).
در آخر داده های ترین و تست را به نسبت ۸۰ به ۲۰ جدا کردم و داده ها(تصاویر) را به tensor تبدیل کردم و با استفاده از دستورات کتابخانه پایتورچ، دیتاست ترین و دیتاست تست را آماده کردم.

۳ مدل به کار گرفته شده

همانطور که در بخش مقدمه ۱ گفته شد، مدل به کار گرفته شده یک Convolutional Autoencoder است.در زیر قطعه کد مربوط به مدلی که بهترین نتیجه را داشت را می‌بینیم(همانطور که گفتم، مدل های زیادی همراه با هایپرپارامتر های متنوعی تست شده است و در کد ارسال شده همراه با گزارش، یک مدل متفاوت از بهترین مدل را ارسال کرده‌ام و نتیجه آن مدل هم در خروجی کد ارسال شده قابل دیدن است.در این جا فقط نتایج بهترین مدل و قطعه کد بهترین مدل آورده شده است.گرچه هیچ کدام از مدل ها خوب نبودند و تفاوت زیادی با یکدیگر نداشتند.به همین دلیل تصمیم گرفتم فقط یک مدل را گزارش کنم.در کد ارسال شده همراه با گزارش، مدل دیگری استفاده شده است و نتایج خوبی هم ندارد.):

```
class AutoEncoder(nn.Module):
  def __init__(self):
    super().__init__()
    self.encoder = nn.Sequential(
          nn.Conv2d(1, 8, kernel_size = 3),
          nn.BatchNorm2d(8),
```

```
            nn.ReLU(),
            nn.Conv2d(8, 16, kernel_size = 3),
            nn.BatchNorm2d(16),
            nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(16, 32, kernel_size = 3),
            nn.BatchNorm2d(32),
            nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(32, 64, kernel_size = 3),
            nn.BatchNorm2d(64),
            nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(64, 128, kernel_size = 3),
            nn.BatchNorm2d(128),
            nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(128, 256, kernel_size = 3),
            nn.Dropout2d(),
        )
        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(256, 128, kernel_size = 2, stride= 2),
            nn.BatchNorm2d(128),
            nn.ReLU(),
            nn.Conv2d(128, 256, kernel_size = 5),
            nn.BatchNorm2d(256),
            nn.ReLU(),
            nn.Conv2d(256, 256, kernel_size = 5),
            nn.BatchNorm2d(256),
            nn.ReLU(),
            nn.ConvTranspose2d(256, 128,  kernel_size = 2, stride = 2),
            nn.BatchNorm2d(128),
            nn.ReLU(),
            nn.ConvTranspose2d(128, 64,  kernel_size = 2, stride = 2),
            nn.BatchNorm2d(64),
            nn.ReLU(),
            nn.ConvTranspose2d(64, 32,  kernel_size = 2, stride = 2),
            nn.BatchNorm2d(32),
            nn.ReLU(),
            nn.ConvTranspose2d(32, 1,  kernel_size = 2, stride = 2),
            nn.Sigmoid()
        )
    def forward(self, x):
        encoded = self.encoder(x)
        decoded = self.decoder(encoded)
        return decoded
```

در Encoder ، ابتدا با دولایه Convolution پشت سر هم با Kernel Size = 3 سعی شده است که اطلاعات اولیه بیشتری از تصویر بیرون کشیده شود. دو لایه Convolution با Kernel Size = 3 از لحاظ سایز خروجی همانند لایه Convolution با Kernel Size = 5 است اما با حجم محاسبات کمتر و اطلاعات کوچکتر بیرون کشیده شده از تصویر بیشتر.پس از هر لایه Convolution یک لایه Batch Normalization استفاده شده تا مانع از رشد بیش از حد اندازه وزن ها شود که این امر می‌تواند تا حدی مانع از اورفیتینگ مدل شود.بعد از Batch Normalization یک لایه تابع فعال ساز و سپس یک لایه Max Pooling گذاشته شده است . به همین ترتیب لایه ها پشت سر هم قرار می‌گیرند و در هر مرحله بعد تصویر کوچک تر می‌شود و در آخر به Latent Space می‌رسیم.در آخرین لایه Encoder از لایه Drop out استفاده شده است.این کار مانع از اورفیت شدن مدل می‌شود.
در Decoder با استفاده از لایه Transposed Convolution و لایه Batch Normalization و تابع فعال ساز به صورت پشت سرهم و در چند مرحله قصد داریم ابعاد Latent Space را به ابعاد تصویر ورودی برسانیم.در آخر هم از لایه Sigmoid استفاده شده است.

۴ نتایج

برای گرفتن نتایج مطلوب در این پروژه ، مدل ها و هایپرپارامتر [۲] های زیادی را تست کردم(در کدی که همراه با گزارش ارسال کرده‌ام مدلی را انتخاب کرده‌ام که به جای Transposed Convolution از Upsampling استفاده شده است). حداقل دو هفته درگیر تست کردن مدل ها و هایپرپارامتر های مختلف بودم اما متاسفانه هیچ کدام نتیجه مطلوبی نداشتند و جواب ها خیلی بد بود. با توجه به اینکه سایر دانشجویان هم(طبق گفته خودشان در گروه تلگرامی) به جواب های خوبی نرسیدند، من هم به جواب های بد قانع شدم و از بین این جواب های بد بهترینشان را اینجا گزارش می‌کنم.
مدل به کار گرفته شده در بهترین جواب ، همان مدلی است که در قسمت قبل کد آن را آورده بودم. (۳)
هایپرپارامتر های بهترین جواب در جدول زیر آمده است:

Learning Rate	Lr-scheduler	Epoch	Optimization	$\beta 1$	$\beta 2$	Weight-decay	Dropout	Batch Size
0.0002	StepLR	20	Adam	0.9	0.999	0.1	0.5	1

Table 1: Hyperparameters

[2] Hyperparameter

در قسمت آخر هم طبق خواسته سوال پروژه ، تصاویر ورودی و تصویر تولید شده توسط مدل در شکل زیر آمده است:

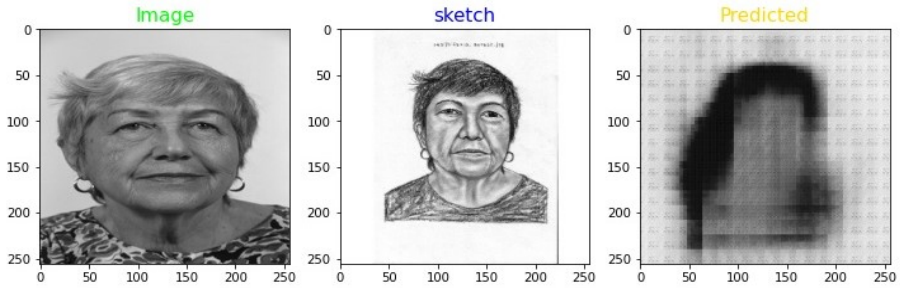

Figure 1: Predicted By Model

۵ جمع بندی

در این پروژه به خوبی درک کردم که تنظیم کردن درست هایپرپارامتر ها چقدر مهم است. Learning Rate اولیه من ۰٫۰۱ بودم و مدل فقط تصویر سیاه تولید می‌کرد.با تغییر مقدار Learning Rate به ۰٫۰۰۰۲ ، تصویر سیاه در خروجی تولید نمی‌شد و کمی امیدوار شدم که رفته رفته به جواب های خوب برسم.
انتظار داشتم که در تصویر نهایی تولید شده ، مقداری ”تاری” هم ببینیم چون از Transposed Convolution استفاده کردم اما کیفیت تصاویر تولید شده توسط مدل اینقدر بد بود که این مشکل زیاد به چشم نمی‌آمد.
دلیل استفاده از شبکه Convolution در Encoder این بود که با توجه به نکاتی که از درس یاد گرفتیم ، می‌دانستیم که شبکه های Convolution نسبت به تغییرات و جابجایی تصاویر ورودی Robust تر هستند و همچنین حجم محاسبات آنها نسبت به شبکه های Fully Connected کمتر است اما متاسفانه نتایج راضی کننده نبود.