با نام و یاد او

دانشکده مهندسی و علوم کامپیوتر
مبانی یادگیری ماشین –
دکتر ملک
نیم سال اول ۱۴۰۵-۱۴۰۴

گزارش پروژه شماره ۱
گردآورنده: آراس ولی‌زاده
موضوع: بازسازی تصاویر

مقدمه:

در این پروژه هدف ما بازسازی و بهبود کیفیت تصاویر چهره با استفاده از یک معماری مبتنی بر شبکه‌های عصبی کانولوشنی بوده است؛ به‌گونه‌ای که از یک تصویر تخریب‌شده، نسخه‌ای نزدیک به تصویر اصلی باکیفیت تولید شود. مسئله در چارچوب یک مسئله‌ی restoration تعریف شده است، با این تفاوت که علاوه بر کیفیت بصری، حفظ هویت چهره نیز اهمیت دارد. برای این منظور، کل پروژه به‌صورت ماژولار طراحی شد تا هر بخش از pipeline شامل داده، مدل، loss، آموزش و ارزیابی به‌صورت مستقل و قابل توسعه پیاده‌سازی شود.

ماژول face_dataset.py:

در بخش داده، ابتدا فرض بر این بوده که تصاویر مجموعه CelebA از نظر موقعیت چهره همتراز شده‌اند تا نوسانات مکانی چشم و دهان به حداقل برسد و مدل بتواند روی بازسازی جزئیات تمرکز کند. ماژول datasets/face_dataset.py مسئول بارگذاری تصاویر و آماده‌سازی آن‌ها برای آموزش است. در این ماژول، تصاویر ابتدا resize شده و به بازه‌ی عددی صفر تا یک نرمال می‌شوند. سپس افزایش داده روی تصویر تمیز اعمال می‌شود که شامل چرخش‌های جزئی، برگرداندن افقی و تغییرات محدود روشنایی و کنتراست است. هدف از این مرحله افزایش تنوع ظاهری بدون تخریب هویت چهره بوده است تا مدل نسبت به تغییرات طبیعی مقاوم‌تر شود. پس از آن، نسخه‌ی تخریب‌شده‌ی تصویر به‌صورت on-the-fly تولید می‌شود تا هر بار که تصویر خوانده می‌شود، نوع تخریب متفاوتی تجربه شود و مدل به یک الگوی ثابت عادت نکند.

ماژول utils/degradation.py:

منطق تولید تخریب‌ها در ماژول utils/degradation.py پیاده‌سازی شده است. در این بخش سه نوع تخریب در نظر گرفته شد. اولین نوع، افزودن نویز گاوسی با انحراف معیار تصادفی در بازه‌ی مشخص است که با استفاده از نویز تصادفی و clipping به بازه‌ی مجاز پیاده‌سازی شده است. دومین نوع، کاهش رزولوشن به‌صورت downsampling و سپس upsampling با ضرایب ۱.۲ و ۱.۴ است که منجر به از دست رفتن جزئیات فرکانس بالا می‌شود. سومین نوع، محوشدگی حرکتی یا motion blur است که با ساخت یک کرنل خطی در جهت‌های مختلف و اعمال convolution به‌صورت depthwise روی هر کانال رنگی انجام شده است. ترکیب این سه تخریب باعث شد مدل با شرایط واقعی‌تری مواجه شود و توان تعمیم‌پذیری آن افزایش یابد.

ماژول models/unet.py:

معماری بازسازی در ماژول models/unet.py پیاده‌سازی شده است. این معماری یک نسخه‌ی ساده‌شده از U-Net است که شامل یک مسیر encoder برای استخراج ویژگی‌ها و یک مسیر decoder برای بازسازی تصویر می‌باشد. در بخش encoder، با استفاده از کانولوشن‌های متوالی و pooling، نمایش‌های عمیق‌تری از تصویر استخراج می‌شود. در بخش decoder، از upsampling به‌همراه اتصال‌های میان‌بری (skip connections) استفاده شده است تا اطلاعات مکانی سطح پایین که در لایه‌های ابتدایی استخراج شده‌اند، مستقیماً به مراحل بازسازی منتقل شوند. این ساختار کمک می‌کند جزئیات لبه‌ها و ساختار چهره بهتر حفظ شود و مشکل از دست رفتن اطلاعات مکانی کاهش یابد. خروجی شبکه با تابع سیگموید محدود شده تا مقادیر در بازه‌ی صفر تا یک باقی بمانند.

ماژول losses/losses.py:

در بخش تابع هزینه، ماژول losses/losses.py مسئول تعریف restoration loss است که در این پروژه از L1 استفاده شده است، زیرا L1 نسبت به L2 تمایل کمتری به تولید تصاویر بیش‌ازحد صاف دارد و جزئیات را بهتر حفظ می‌کند. علاوه بر آن، برای دریافت امتیاز کامل، ماژول losses/identity_loss.py اضافه شده است که یک مدل تشخیص چهره از پیش‌آموزش‌دیده را بارگذاری می‌کند و embedding ویژگی چهره را استخراج می‌کند. در این بخش، فاصله‌ی کسینوسی بین embedding تصویر بازسازی‌شده و تصویر اصلی محاسبه می‌شود و به‌عنوان identity loss به تابع هزینه افزوده می‌گردد. ترکیب این دو loss باعث می‌شود شبکه نه‌تنها از نظر پیکسلی به تصویر اصلی نزدیک شود، بلکه از نظر هویت چهره نیز شباهت بیشتری حفظ کند. وزن این loss به‌صورت یک پارامتر در config.py قابل تنظیم است و پس از چند epoch ابتدایی فعال می‌شود تا ابتدا مدل پایه‌ی بازسازی را یاد بگیرد.

فایل train.ipynb:

فرآیند آموزش در فایل train.py پیاده‌سازی شده است. در این بخش، داده‌های train و validation از طریق DataLoader بارگذاری می‌شوند. در هر epoch، شبکه در حالت train قرار گرفته و گرادیان‌ها محاسبه می‌شوند. پس از پایان هر epoch، مدل در حالت evaluation روی مجموعه validation اجرا می‌شود و معیارهای PSNR و SSIM با استفاده از ماژول utils/evaluator.py محاسبه می‌شوند. این ماژول منطق ارزیابی را از train و evaluate جدا کرده و از تکرار کد جلوگیری می‌کند. در صورتی که مقدار PSNR نسبت به بهترین مقدار قبلی بهبود یابد، وزن‌های مدل ذخیره می‌شوند. همچنین مکانیزم early stopping در نظر گرفته شده است که اگر طی چند epoch متوالی بهبودی مشاهده نشود، فرآیند آموزش متوقف می‌شود تا از overfitting جلوگیری شود. نمودارهای تغییرات loss و PSNR نیز در پوشه‌ی نتایج ذخیره می‌شوند تا روند یادگیری قابل تحلیل باشد.

فایل evaluate.py:

فایل evaluate.py مسئول ارزیابی نهایی مدل روی مجموعه‌های validation، test و در صورت وجود، یک دیتاست چهره‌ی دیگر برای بررسی تعمیم‌پذیری است. این فایل مدل ذخیره‌شده را بارگذاری می‌کند و با استفاده از ماژول

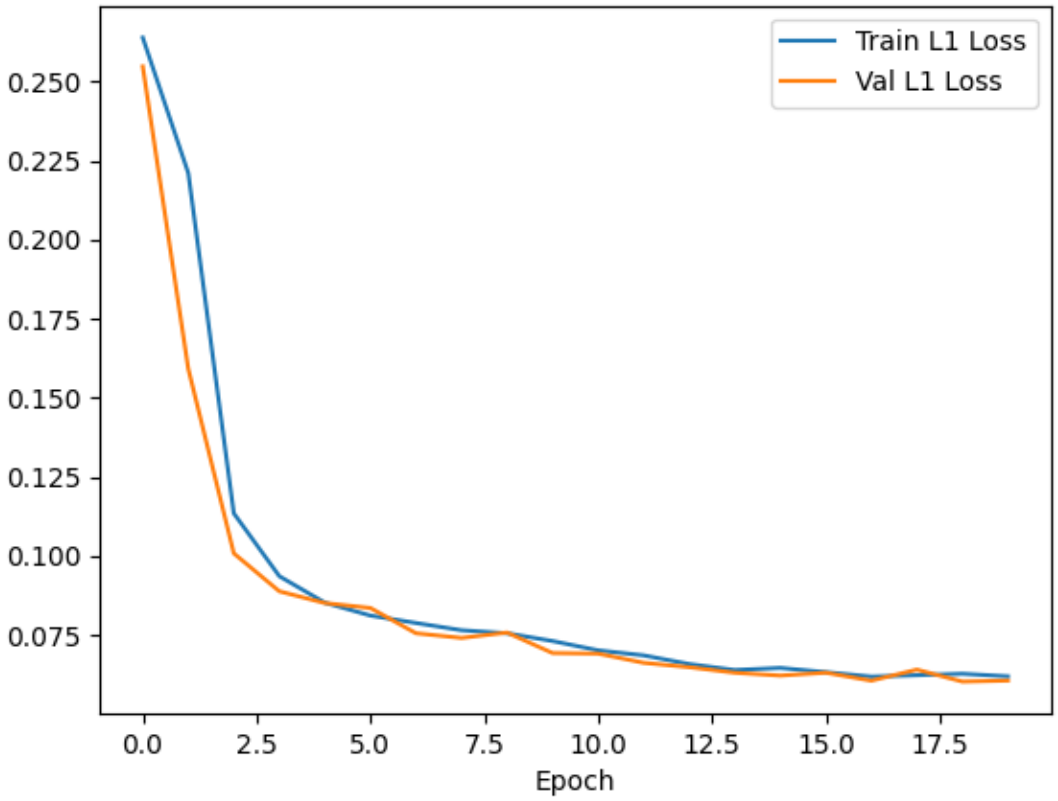
utils/evaluator.py میانگین PSNR و SSIM را گزارش می‌دهد. علاوه بر آن، با استفاده از ماژول utils/visualize.py نمونه‌هایی از تصاویر تخریب‌شده، بازسازی‌شده و تصویر اصلی در قالب یک grid ذخیره می‌شود تا مقایسه‌ی بصری نتایج امکان‌پذیر باشد. این تصاویر برای درج در گزارش نهایی پروژه استفاده می‌شوند.

فایل config.py:

در نهایت، فایل config.py نقش مدیریت تنظیمات را بر عهده دارد. در این فایل مسیر داده‌ها، اندازه‌ی batch، نرخ یادگیری، تعداد epoch، وزن identity loss و پارامترهای early stopping تعریف شده‌اند. این طراحی باعث شده تغییر تنظیمات بدون دستکاری منطق اصلی کد امکان‌پذیر باشد و ساختار پروژه تمیز و قابل توسعه باقی بماند.

به‌طور کلی، معماری پروژه به‌گونه‌ای طراحی شده که هر بخش از pipeline به‌صورت ماژول مستقل پیاده‌سازی شود. ماژول داده مسئول تولید جفت‌های degraded/clean است، ماژول مدل بازسازی را انجام می‌دهد، ماژول loss کیفیت پیکسلی و هویتی را کنترل می‌کند، ماژول آموزش روند یادگیری و انتخاب بهترین مدل را مدیریت می‌کند و ماژول ارزیابی عملکرد نهایی را گزارش می‌دهد. این ساختار علاوه بر برآورده کردن نیازهای پروژه، امکان توسعه‌های بعدی مانند افزودن perceptual loss، deep supervision یا معماری‌های پیشرفته‌تر را نیز فراهم می‌کند.

```
...   Epoch 1/20 | Train L1 Loss: 0.2640 | Val L1 Loss: 0.2549
      Best model saved.
      Epoch 2/20 | Train L1 Loss: 0.2211 | Val L1 Loss: 0.1592
      Best model saved.
      Epoch 3/20 | Train L1 Loss: 0.1135 | Val L1 Loss: 0.1009
      Best model saved.
      Epoch 4/20 | Train L1 Loss: 0.0936 | Val L1 Loss: 0.0888
      Best model saved.
      Epoch 5/20 | Train L1 Loss: 0.0852 | Val L1 Loss: 0.0851
      Best model saved.
      Epoch 6/20 | Train L1 Loss: 0.0812 | Val L1 Loss: 0.0836
      Best model saved.
      Epoch 7/20 | Train L1 Loss: 0.0788 | Val L1 Loss: 0.0756
      Best model saved.
      Epoch 8/20 | Train L1 Loss: 0.0765 | Val L1 Loss: 0.0741
      Best model saved.
      Epoch 9/20 | Train L1 Loss: 0.0756 | Val L1 Loss: 0.0758
      No improvement for 1/5 epochs.
      Epoch 10/20 | Train L1 Loss: 0.0731 | Val L1 Loss: 0.0693
      Best model saved.
      Epoch 11/20 | Train L1 Loss: 0.0702 | Val L1 Loss: 0.0691
      Best model saved.
      Epoch 12/20 | Train L1 Loss: 0.0686 | Val L1 Loss: 0.0662
```


Train L1 Loss
Val L1 Loss
0.250
0.225
0.200
0.175
0.150
0.125
0.100
0.075
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epoch