مبانی هوش محاسباتی

دانشگاه فردوسی مشهد

پروژه چهارم

نیمسال اول تحصیلی ۱۴۰۳-۱۴۰۲

گروه مهندسی کامپیوتر — مهلت ارسال: ۱۵ بهمن ساعت ۲۳:۵۹

قصد داریم تصاویر دیتاست Oxford Flowers را با شبکه عمیق طبقه‌بندی کنیم. این دیتاست شامل ۱۰۰ [۱] کلاس از تصاویر انواع مختلف گل می‌باشد. در این پروژه از ۸۰ کلاس اول دیتاست، تمامی تصاویر آموزشی را در اختیار داریم و این مجموعه را دیتاست A می‌نامیم. از ۲۰ کلاس بعدی تنها ۱۰۰ داده در اختیار داریم، به طوری که از هر کلاس ۵ داده موجود است (این مجموعه را دیتاست B می‌نامیم). [۲]

- در فاز اول پروژه شبکه‌ای را بر دیتاست A آموزش می‌دهیم و عملکرد را بر داده‌های تست همان دیتاست بررسی می‌کنیم.
- در فاز دوم از این پروژه می‌خواهیم از وزن‌های شبکه آموزش دیده در فاز اول استفاده کنیم تا شبکه‌ای آموزش دهیم که علاوه بر ۸۰ کلاس دیتاست A ، ۲۰ کلاس جدید از دیتاست B را هم طبقه‌بندی کند. چالش فاز دوم تعداد کم داده‌های آموزشی دیتاست B است و **هدف نهایی** در این فاز آموزش شبکه جدیدی با دیتاست B و به کمک شبکه فاز قبل است که توانایی طبقه‌بندی تمام کلاس‌ها (۱۰۰ کلاس) را داشته باشد.

در ادامه ابتدا به معرفی دیتاست Oxford Flowers پرداخته شده، سپس فاز اول و دوم پروژه توضیح داده شده و در بخش چهارم معماری مدل آورده شده است.

۱) دیتاست Oxford Flowers

این دیتاست شامل ۱۰۰ کلاس است که هرکدام بین ۴۰ تا ۲۵۸ عدد داده دارند. برای توضیحات بیشتر و دیدن نمونه گل‌ها از این لینک استفاده کنید. همانطور که گفته شد برای انجام فازهای اول و دوم پروژه، این دیتاست به دو بخش A و B تقسیم شده.

دیتاست A فقط شامل ۸۰ کلاس اول از دیتاست اصلی می‌شود و هر داده از این دیتاست شامل تصویر رنگی با ابعاد متغیر و یک برچسب نشان دهنده نوع گل (عدد صحیحی بین ۰ تا ۸۰) است. دیتاست B فقط شامل ۲۰ کلاس بعدی است و هر داده از این دیتاست شامل تصویر رنگی با ابعاد متغیر و یک برچسب نشان دهنده نوع گل (عدد صحیحی بین ۸۰ تا ۱۰۰) است.

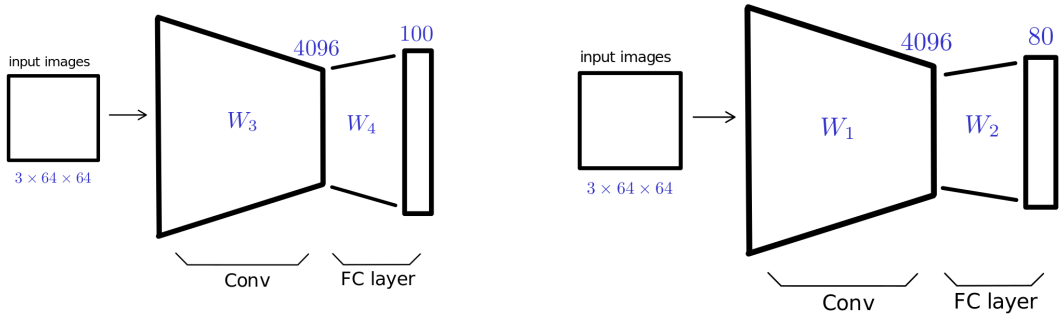
برای دسترسی به داده‌های آموزشی و تست دیتاست‌های A و B و همچنین داده‌های تست مربوط به همه ۱۰۰ کلاس از تابع `get_oxford_splits()` استفاده کنید این تابع پنج Dataloader خروجی می‌دهد که در بخش ضمیمه ۶.۱ توضیح داده شده است.

[۱] در اصل ۱۰۲ کلاس است اما در این پروژه تنها از ۱۰۰ کلاس استفاده می شود

[۲] دقت شود داده‌های آموزشی و تست دیتاست A فقط شامل ۸۰ کلاس اول می‌شود و داده‌های آموزشی و تست دیتاست B فقط شامل ۲۰ کلاس دیگر می‌شود.

۲) فاز اول

در فاز اول تنها دیتاست A را در اختیار داریم و می‌خواهیم شبکه عمیق پیچشی[۳] با معماری گفته شده در بخش ۴ را برای طبقه‌بندی این ۸۰ کلاس آموزش دهیم. برای این کار از تابع زیان Cross Entropy استفاده می کنیم.



(آ) نمایی کلی از شبکه فاز اول. W_1 را تمام پارامترهای قابل یادگیری لایه‌های پیچشی و W_2 را پارامترهای قابل یادگیری لایه یادگیری آخر شبکه (FC layer) می‌نامیم. **(ب)** نمایی کلی از شبکه فاز دوم. W_3 را تمام پارامترهای قابل یادگیری لایه‌های پیچشی و W_4 را پارامترهای قابل یادگیری لایه آخر شبکه (FC layer) می‌نامیم.

شکل ۱

برای این فاز منحنی یادگیری و دقت[۴] را برای داده‌های آموزشی و تست دیتاست A گزارش دهید.

۳) فاز دوم

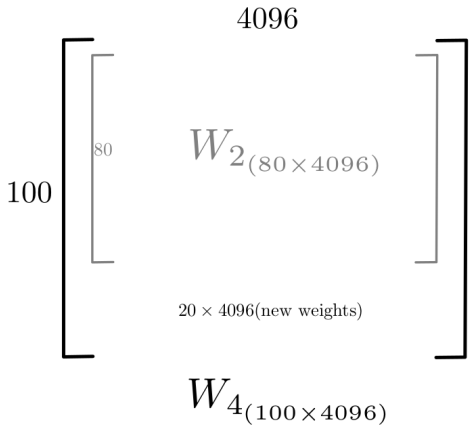
در این فاز در مرحله آموزش **تنها** دیتاست B را در اختیار داریم. آموزش شبکه از ابتدا با تعداد کم داده‌ای که در اختیار داریم نتیجه خوبی برای داده‌های تست ندارد، برای همین می‌خواهیم از شبکه‌ای که در بخش اول آموزش دادیم استفاده کنیم. به این صورت که وزن‌های شبکه آموزش دیده در بخش اول را در شبکه جدید کپی می‌کنیم. برای اینکار وزن‌های شبکه جدید را دو بخش در نظر می‌گیریم: W_3 که شامل تمام وزن‌ها و پارامترهای قابل یادگیری لایه‌های پیچشی می‌شود و W_4 که وزن‌ها و پارامترهای قابل یادگیری لایه آخر شبکه جدید است. (شکل ۱.ب)

از آنجایی که ابعاد لایه‌های پیچشی این شبکه جدید و شبکه فاز ۱ عینا همانند یک دیگر هستند، می‌توانیم W_1 را در W_3 کپی کنیم. اما ابعاد W_2 (80×4096) و W_4 (100×4096) متفاوت است. از آنجایی که نورون‌های ۰ تا ۸۰ لایه آخر شبکه فاز اول مربوط به ۸۰ کلاس اول و نورون‌های ۰ تا ۸۰ لایه آخر شبکه فاز دوم هم مربوط به ۸۰ کلاس اول هستند، می‌توانیم ۸۰ سطر اول از W_4 را با مقادیر W_2 پر کنیم. با اینکار بخشی از وزن‌های W_4 کاملا جدید هستند و در واقع 20×4096 پارامتر جدید داریم (شکل ۲).

---

[۳] Deep Convolutional Network

[۴] برای رسم این منحنی‌ها از تابع `custom_plot_training_stats()` در فایل utils.py که در اختیارتان قرار گرفته استفاده کنید



شکل ۲

برای آموزش این شبکه با دیتاست B سه روش مختلف را امتحان می‌کنیم که در ادامه توضیح داده شده است. دقت شود در این فاز تنها داده‌های ۲۰ کلاس بعدی را در اختیار داریم و مجاز به استفاده از داده‌های آموزشی دیتاست A نیستیم. برای هر کدام از سه روش موارد زیر را گزارش دهید:

- منحنی یادگیری و دقت داده‌های آموزشی (B_train_dl ، توضیح در ضمیمه)
- منحنی یادگیری و دقت داده‌های تست (B_test_dl)
- منحنی یادگیری و دقت داده‌های تست کل ۱۰۰ کلاس (train_all)
- ماتریس گمراهی[۵] برای داده‌های تست کل ۱۰۰ کلاس در هر مرحله از آموزش رسم کنید. (اگر شبکه را n مرحله (epoch) آموزش می‌دهید، n ماتریس گمراهی حاصل می‌شود)[۶]

۳).۱ روش اول

در روش اول هیچ محدودیتی بر وزن‌های شبکه نداریم و بعد از کپی کردن پارامترهای شبکه فاز اول در شبکه جدید، آن را به طور عادی با داده‌های آموزشی دیتاست B با تابع زیان Cross Entropy آموزشی می‌دهیم. دقت کنید که تمام پارامترهای شبکه جدید در آموزش آپدیت می‌شوند.

۳).۲ روش دوم

در روش دوم حین آموزش شبکه پارامترهای لایه پیچشی شبکه جدید (W_3) را آپدیت نمی‌کنیم. در واقع تنها پارامترهای لایه آخر (W_4) در آموزش آپدیت می‌شوند (شکل ۳.۴آ). با این روش ویژگی‌های شبکه پیچشی که در فاز اول برای تولید بردار ویژگی 4096 بعدی آموزش دیده، برای دیتاست B استفاده می‌شود اما این فضای ویژگی برای داده‌های کلاس‌های جدید آپدیت نمی‌شود.

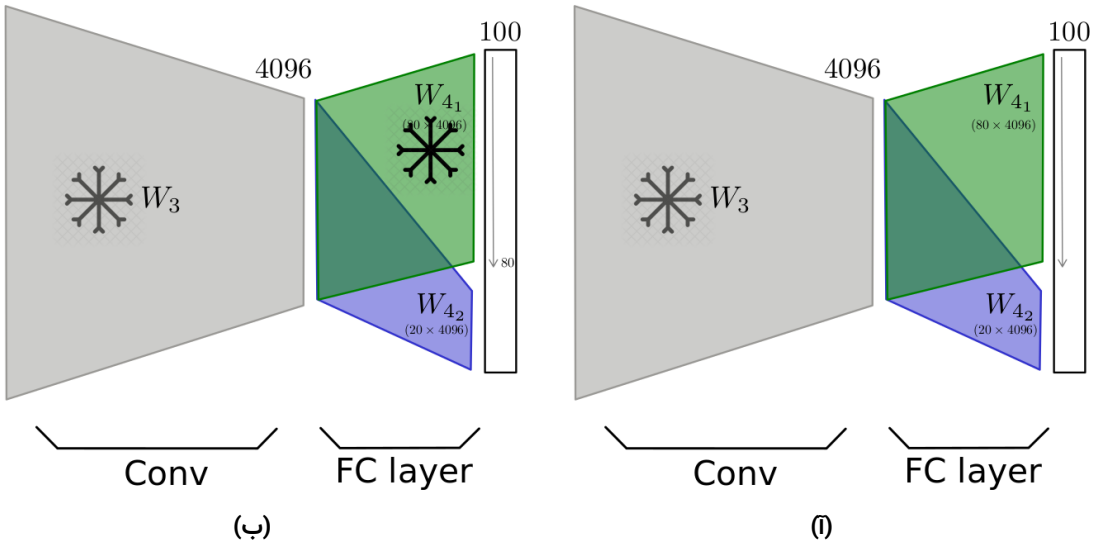
---

[۵] Confurion Matrix

[۶] برای رسم و ذخیره ماتریس گمراهی از تابع `plot_conf()` در فایل utils.py که در اختیارتان قرار گرفته استفاده کنید.

۳.۳) روش سوم

در روش آخر علاوه بر W_3 بخشی از وزن‌های لایه آخر که از شبکه فاز ۱ کپی کردیم را هم فریز می‌کنیم. در واقع اگر وزن‌های W_4 را به دو بخش W_{4_1} (آن وزن‌هایی که از شبکه فاز ۱ کپی کردیم) و W_{4_2} تقسیم کنیم، می‌خواهیم W_{4_1} را هم فریز کنیم. در این روش تنها W_{4_2} آپدیت می‌شود (شکل ۴.ب).



(ب) (ا)

شکل ۳

۴) معماری مدل

مدلی که در هر دو فاز استفاده می‌کنیم به غیر از لایه آخر معماری یکسانی دارند. این مدل تصاویری با ابعاد $3 \times 64 \times 64$ به عنوان ورودی می‌گیرد و یک بردار C بعدی برای پیشبینی کلاس تصاویر خروجی می‌دهد. که در اینجا C تعداد کلاس‌ها است (در فاز اول و دوم به ترتیب $C = 80$ و $C = 20$ است). معماری این مدل در جدول ۱ آمده است.

Layer Type	Input Size	Output Size	Layer Setup
Convolutional	$3 \times 64 \times 64$	$64 \times 64 \times 64$	Conv (3×3, 64 channels, stride 1, pad 1), Batch Norm, ReLU
Convolutional	$64 \times 64 \times 64$	$64 \times 64 \times 64$	$4\times$ [Conv (3×3, 64 channels, stride 1, pad 1), Batch Norm, ReLU]
Pooling	$64 \times 64 \times 64$	$64 \times 32 \times 32$	Max Pool (2×2)
Convolutional	$64 \times 32 \times 32$	$96 \times 32 \times 32$	$4\times$ [Conv (3×3, 96 channels, stride 1, pad 1), Batch Norm, ReLU]
Pooling	$96 \times 32 \times 32$	$96 \times 16 \times 16$	Max Pool (2×2)
Convolutional	$96 \times 16 \times 16$	$128 \times 16 \times 16$	$4\times$ [Conv (3×3, 128 channels, stride 1, pad 1), Batch Norm, ReLU]
Pooling	$128 \times 16 \times 16$	$128 \times 8 \times 8$	Max Pool (2×2)
Convolutional	$128 \times 8 \times 8$	$256 \times 8 \times 8$	$4\times$ [Conv (3×3, 256 channels, stride 1, pad 1), Batch Norm, ReLU]
Pooling	$256 \times 8 \times 8$	$256 \times 4 \times 4$	Max Pool (2×2)
Fully Connected	4096	C	-

جدول ۱: معماری شبکه عمیق فاز اول و دوم که مقدار C در آن‌ها به ترتیب ۸۰ و ۱۰۰ است. دقت کنید بعد از آخرین لایه Pooling باید بردارهای ویژگی را Flat کنید تا به عنوان ورودی به لایه Fully Connected بدهید.

۵) چه چیزی باید پیاده سازی و گزارش شود؟

موارد خواسته شده را با pytorch پیاده سازی کنید. (ویدیو آموزشی برای pytorch به همراه فایل `utils.py` در اینجا قرار گرفته) همچنین فایل گزارشی از موارد خواسته شده زیر و نتایج خود را ارسال کنید.

- ابتدا شبکه خواسته شده در قسمت ۴ را پیاده سازی کنید.
- شبکه مربوط به فاز اول را با استفاده از دیتاست A آموزش دهید و تست کنید:
  - منحنی یادگیری و دقت را برای داده‌های آموزشی و تست دیتاست A گزارش دهید.
- شبکه مربوط به فاز دوم را با استفاده از دیتاست B **با هر سه روش توضیح داده شده** آموزش دهید و تست کنید. در هر روش:
  - در انتخاب نرخ یادگیری[۷] و پارامترهای دیگر بهینه ساز[۸] دقت کنید.
  - منحنی یادگیری و دقت را برای داده‌های آموزشی، تست، و داده تست ۱۰۰ کلاس (B_train_dl, B_test_dl, test_all) گزارش دهید.
  - ماتریس گمراهی برای داده‌های تست کل ۱۰۰ کلاس در هر مرحله از آموزش رسم کنید.
- تغییرات ماتریس گمراهی در هر سه روش فاز دوم را بررسی و مقایسه کنید و نتایج خود را گزارش دهید. به طور دقیقتر، میزان نزول/صعود دقت کلاس‌هایی که از قبل آموزش دیده‌اند (۸۰ کلاس اول) و نزول و صعود دقت کلاس‌هایی که در حال آموزش هستند (۲۰ کلاس دیگر) را با کمک ماتریس گمراهی در هر روش مقایسه کنید.

[۷] Learning Rate
[۸] Optimizer

۶) ضمیمه

۱.۶) لود کردن داده‌ها

برای دسترسی به دیتاست A ، B و دیتاست تست همه ۱۰۰ کلاس از تابع `get_oxford_splits()` استفاده کنید:

```
def get_oxford_splits(
    batch_size: int,
    data_loader_seed: int = 111,
    pin_memory: bool = True,
    num_workers: int = 2,
    ):
    ...
    return A_train_dl, A_test_dl, B_train_dl, B_test_dl, test_all
```

از آنجایی که تصاویر گل دیتاست Oxford Flowers ابعاد متغیری دارد و شبکه باید تصاویر با ابعاد یکسان به عنوان ورودی بگیرد، این تابع تصاویر گل را به ابعاد 64×64 تغییر می‌دهد. همچنین این تابع مقدار سایز batch (`batch_size`) ، سید دیتالودر برای یکسان بودن اجراها (`data_loader_seed`) ، و دو متغیر دیگر را به عنوان ورودی می‌گیرد. برای اینکه در هر اجرا ترتیب لود شدن داده‌ها متفاوت باشد می‌توانید مقدار `data_loader_seed` را تغییر دهید. اگر کد را در سیستم خودتان اجرا می‌کنید، مقدار `pin_memory` را `False` و `num_workers` را `1` قرار دهید، در غیر اینصورت اگر از Google Colab استفاده می‌کنید مقادیر پیش فرض کفایت می‌کند.

این تابع پنج Dataloader خروجی می‌دهد:

- `A_train_dl` (شامل ۴۶۱۷ نمونه) و `A_test_dl` (شامل ۱۵۳۸ نمونه) به ترتیب حاوی حاوی داده‌های آموزشی و تست دیتاست A هستند.
- `B_train_dl` (شامل ۱۰۰ نمونه) و `B_test_dl` (شامل ۵۱۸ نمونه) به ترتیب حاوی داده‌های آموزشی و تست دیتاست B است
- `test_all` حاوی داده‌های تست که شامل هر ۱۰۰ کلاس می‌شود (۲۰۵۶ نمونه) ، دقت کنید این نمونه‌ها هیچ اشتراکی با داده‌های آموزشی قبلی ندارند.