به نام خدا

درس یادگیری عمیق

تمرین سری هفتم

استاد درس : دکتر محمدرضا محمدی

دستیاران : فاطمه ستوده، رضا علیدوست،

علی سبحانی و مرتضی حاجی آبادی

دانشگاه علم و صنعت ایران، دانشکده مهندسی کامپیوتر

نیمسال دوم تحصیلی ۱۴۰۳ - ۱۴۰۴

مهلت تحویل : ۱۴۰۴/۰۵/۰۵

لطفا به نکات موجود در سند قوانین انجام و تحویل تمرین ها دقت فرمایید.

سوالات تئوری

۱.  روش‌های پیشرفته‌ی یادگیری ماشین خودکار مانند جستجوی معماری عصبی (NAS) و AutoAugment برای خودکارسازی جنبه‌های پیچیده طراحی مدل توسعه یافته‌اند. اصول اصلی این روش‌ها را توضیح دهید و موارد زیر را بررسی کنید. برای هریک از ۲ روش ذکر شده، موارد ذیل را به صورت جداگانه مورد بحث قرار دهید:(۲۰ نمره)

(آ) نحوه تعریف مسئله‌ی خودکارسازی توسط هر تکنیک

(ب) سازوکارها یا ساختارهای کنترلی مورد استفاده (مثل Controller-Trainer، sub-policies)

(ج) چالش‌های مرتبط با پیچیدگی فضای جستجو و هزینه محاسباتی

(د) نقش یادگیری تقویتی (RL)، بهینه‌سازی بیزی و تطابق چگالی در حل این چالش‌ها

(ه) ذکر نمونه‌ها یا نسخه‌های خاص هر روش (در صورت وجود)

جستجوی معماری عصبی (NAS)

- تعریف مسئله

NAS فرآیند طراحی خودکار معماری شبکه عصبی را به‌صورت مسئله‌ای بهینه‌سازی فرموله می‌کند که هدف آن یافتن معماری‌ای با بیشترین عملکرد (مثل دقت بالا) است.

- سازوکار کنترل

ساختار متداول در NAS، مدل کنترل‌گر-آموزش‌دهنده (Controller-Trainer) است:

- کنترل‌گر (معمولاً یک شبکه بازگشتی) معماری جدیدی را تولید می‌کند.
- این معماری توسط آموزش‌دهنده (Trainer) آموزش داده می‌شود.
- دقت مدل حاصل بر مجموعه اعتبارسنجی به‌عنوان پاداش در نظر گرفته می‌شود.
- پارامترهای کنترل‌گر بر اساس این پاداش با استفاده از RL به‌روزرسانی می‌شوند.

- چالش‌ها

فضای جستجوی معماری بسیار بزرگ و پیچیده است و آموزش هر معماری هزینه زیادی دارد.

- حل چالش‌ها با RL و بهینه‌سازی

از یادگیری تقویتی (RL) برای آموزش کنترل‌گر استفاده می‌شود تا معماری‌های بهتری تولید کند.

- نمونه‌ها و نسخه‌ها

- **جستجوی سلولی:** بسیاری از روش‌ها ابتدا یک ساختار پایه (Cell) را جستجو می‌کنند و سپس شبکه را از روی آن می‌سازند.
- **EfficientNet:** با الهام از MnasNet، بهینه‌سازی چندهدفه انجام می‌دهد (تعادل بین دقت و هزینه محاسباتی). معیار آن:

$$ACC \times \left(\frac{FLOPS}{T}\right)^{w}$$

- **DARTS:** با وزن‌دهی پیوسته به عملیات‌ها و حذف اتصالات ضعیف، فضای جستجو را به‌صورت قابل تفکیک و قابل مشتق‌گیری مدل می‌کند.

AutoAugment

- تعریف مسئله

هدف AutoAugment یافتن بهترین روش‌های افزایش داده (Data Augmentation) برای بهبود عملکرد مدل است.

- سازوکار کنترلی

- هر عملیات افزایش داده دارای سه پارامتر است: نوع عملیات (مثل چرخش، کشش)، احتمال اعمال، و شدت عملیات (intensity).

- یک sub-policy مجموعه‌ای از چند عملیات است که روی یک نمونه داده اجرا می‌شود.
- یک سیاست از چند زیرسیاست sub-policy تشکیل شده که یکی از آن‌ها در هر تکرار آموزش انتخاب می‌شود.

- چالش‌ها

فضای جستجو بسیار بزرگ است (برای مثال، 2.9×10^{32} حالت ممکن برای ۵ زیرسیاست با ۱۶ عملیات و ۱۰ شدت و احتمال).

- حل چالش‌ها با (RL، Bayesian Optimization، Density Matching)
  - AutoAugment اولیه از یادگیری تقویتی (RL) استفاده می‌کرد.
  - Fast AutoAugment برای کاهش هزینه محاسباتی، از بهینه‌سازی بیزی (BO) استفاده می‌کند.
  - همچنین، از تطبیق چگالی (Density Matching) استفاده می‌کند تا داده‌های افزایش‌یافته توزیعی مشابه داده اصلی داشته باشند.

۲.

(آ) ماتریس W زیر را در نظر بگیرید. ابتدا تعریف کنید که مرتبه (rank) یک ماتریس چیست و چه مفهومی دارد سپس مرتبه ماتریس W را به صورت دستی محاسبه کنید. در ادامه ماتریس W را به دو ماتریس تجزیه کنید به گونه ای که حاصل‌ضرب آن‌ها برابر با W باشد (از مرتبه ماتریس استفاده کنید). در نهایت محاسبه کنید که تعداد پارامترها در این تجزیه، نسبت به تعداد پارامترهای اولیه ماتریس W، چقدر کاهش پیدا کرده است. تاثیر مرتبه ماتریس را در تعداد پارامترها بررسی کنید. (۵ نمره)

$$W = \begin{bmatrix} 1 & 3 & 0 & 2 & 3 \\ 0 & 1 & -2 & 1 & 2 \\ 2 & 5 & 1 & 3 & 4 \\ 1 & 2 & 1 & 1 & 1 \\ 3 & 7 & 2 & 4 & 5 \end{bmatrix}$$

برای این سوال از پاسخ آقای ارشیا حسین زاده استفاده شده است.

مرتبه یک ماتریس، بزرگ‌ترین تعداد ستون‌های مستقل خطی (یا به‌طور معادل، ردیف‌های مستقل خطی) در آن ماتریس است. به عبارت دیگر، مرتبه ماتریس A که با rank(A) نشان داده می‌شود، برابر با بعد فضای ستونی (column space) یا فضای ردیفی (row space) ماتریس است. مرتبه نشان‌دهنده میزان اطلاعات غیرتکراری موجود در ماتریس است و در کاربردهایی مانند حل دستگاه‌های معادلات خطی، تجزیه ماتریس‌ها و فشرده‌سازی داده‌ها نقش مهمی دارد. به‌طور خلاصه، مرتبه معیاری از "بزرگی" ساختار خطی ماتریس است. اگر مرتبه یک ماتریس بسیار کمتر از تعداد سطرها و ستون‌های آن باشد، به این معناست که بسیاری از سطرها (یا ستون‌ها) ترکیبی خطی از سطرهای دیگر هستند و اطلاعات جدیدی به ماتریس اضافه نمی‌کنند. در واقع، اطلاعات زیادی در ماتریس تکراری و زائد است. اگر مرتبه یک ماتریس برابر با کمینه تعداد سطرها و ستون‌های آن باشد، به آن ماتریس مرتبه کامل می‌گویند. این یعنی تمام سطرها و ستون‌ها تا حد ممکن مستقل خطی هستند. برای محاسبه مرتبه ماتریس W ، از روش حذف گاوس (Gaussian elimination) استفاده می‌کنیم تا ماتریس را به شکل پله‌ای ردیفی (row echelon form) درآوریم. تعداد ردیف‌های غیرصفر در این شکل، مرتبه ماتریس را نشان می‌دهد.

صفر کردن درایه‌های ستون اول در زیر درایه محوری (W_{11}):

- $R_3 = R_3 - 2R_1$
- $R_4 = R_4 - R_1$
- $R_5 = R_5 - 3R_1$

صفر کردن درایه‌های ستون دوم در زیر درایه محوری (W_{22}):

- $R_3 = R_3 + R_2$
- $R_4 = R_4 + R_2$
- $R_5 = R_5 + 2R_2$

صفر کردن درایه‌های ستون سوم در زیر درایه محوری (W_{33}):

- $R_4 = R_4 - R_3$
- $R_5 = R_5 - 2R_3$

ماتریس حاصل دارای 3 سطر غیر صفر است. بنابراین، مرتبه ماتریس W برابر با 3 است.

ماتریس پله ای ردیفی
$$\begin{bmatrix} 1 & 3 & 0 & 2 & 3 \\ 0 & 1 & -2 & 1 & 2 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

با توجه به اینکه مرتبه ماتریس W برابر با 3 است، می‌توانیم W را به صورت حاصل‌ضرب دو ماتریس A و B تجزیه کنیم، به‌گونه‌ای که: $W = AB$ که در آن A یک ماتریس 5×3 و B یک ماتریس 3×5 است. این ابعاد از مرتبه ماتریس استفاده می‌کنند، زیرا مرتبه نشان‌دهنده حداقل تعداد ستون‌ها یا ردیف‌های مستقل مورد نیاز برای بازسازی ماتریس است. ماتریس A از ستون‌های محوری (Pivot Columns) ماتریس اصلی W تشکیل می‌شود. ستون‌های محوری ما ستون‌های 1، 2 و 3 بودند.

$$\begin{bmatrix} 1 & 3 & 0 \\ 0 & 1 & -2 \\ 2 & 5 & 1 \\ 1 & 2 & 1 \\ 3 & 7 & 2 \end{bmatrix} = A$$

اکنون باید ماتریس B را پیدا کنیم به‌گونه‌ای که $W = AB$ هر ستون از W را می‌توان به صورت ترکیب خطی ستون‌های A نوشت. برای هر ستون j از W، ضرایب ترکیب خطی را محاسبه می‌کنیم:

- ستون ۱ از W: $[3, 1, 2, 0, 1]^T$ برابر با ستون ۱ از A، پس ضرایب: $[0, 0, 1]^T$.
- ستون ۲ از W: $[7, 2, 5, 1, 3]^T$ برابر با ستون ۲ از A، پس ضرایب: $[0, 1, 0]^T$.
- ستون ۳ از W: $[2, 1, 1, -2, 0]^T$ برابر با ستون ۳ از A، پس ضرایب: $[1, 0, 0]^T$.
- ستون ۴ از W: $[4, 1, 3, 1, 2]^T$ معادله:

$$[4, 1, 3, 1, 2]^T = [2, 1, 1, -2, 0]^T \cdot c + [7, 2, 5, 1, 3]^T \cdot b + [3, 1, 2, 0, 1]^T \cdot a$$

$$B = \begin{bmatrix} 1 & 0 & 0 & -1 & -3 \\ 0 & 1 & 0 & 1 & 2 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

ماتریس W یک ماتریس 5×5 است، بنابراین تعداد پارامترهای اولیه آن ۲۵ تا می‌باشد اما پس از تجزیه دو ماتریس با ۱۵ پارامتر خواهیم داشت. در این مورد خاص، تعداد پارامترها نه‌تنها کاهش نیافته، بلکه ۵ واحد افزایش یافته است. این نتیجه به دلیل ابعاد نسبتاً کوچک ماتریس W و مرتبه آن (3) است که به‌طور نسبی به ابعاد ماتریس نزدیک است. مرتبه ماتریس نقش کلیدی در تعیین تعداد پارامترهای مورد نیاز در تجزیه دارد. تعداد پارامترها پس از تجزیه مرتبه از فرمول $m \times r + r \times n$ به دست می‌آید، در حالی که تعداد پارامترهای اولیه $m \times n$ است. کاهش پارامترها زمانی اتفاق می‌افتد که:

$$r < \frac{mn}{m+n}$$

برقرار باشد. یعنی وقتی مرتبه r به‌طور قابل‌توجهی کمتر از m و n باشد.

(ب)) با توجه به مقاله LoRA به سوالات زیر پاسخ دهید. مدل پیش‌آموزش دیده BERT با مشخصات زیر داده شده است:

Model dimension: 1024, number of blocks: 24, number of attention heads: 16, vocabulary size: 30000, Rank r of LoRA: 16

هدف ما آموزش دقیق این مدل پیش‌آموخته بر روی دو تسک Question Answering و Sentiment Analysis می‌باشد. تعداد پارامترهای قابل آموزش و تعداد پارامترهای ذخیره‌سازی برای inference را در دو حالت زیر بدست آورید: (۵ نمره)

- آموزش با LoRA
- تنظیم دقیق معمولی (بدون LoRA)

توجه: فقط پارامترهای بخش attention را در نظر بگیرید.

محاسبه پارامترهای بخش Attention در هر بلوک: در هر بلوک attention، داریم:

- ماتریس کوئری 1024×1024 پارامتر
- ماتریس کلید 1024×1024 پارامتر

- ماتریس Value با 1024×1024 پارامتر
- و ماتریس output projection با 1024×1024 پارامتر

کل پارامترهای attention در هر بلوک: $4,194,304$ پارامتر. کل پارامترهای attention در کل مدل: $24 \times 4,194,304 = 100,663,296$ پارامتر.

در حالت آموزش با LORA: در این روش، وزن‌های اصلی مدل BERT فریز شده و فقط ماتریس‌های کوچک LoRA (A, B) برای هر یک از چهار ماتریس attention آموزش داده می‌شوند. تعداد پارامترهای قابل آموزش برای یک ماتریس با LoRA از فرمول زیر به دست می‌آید:

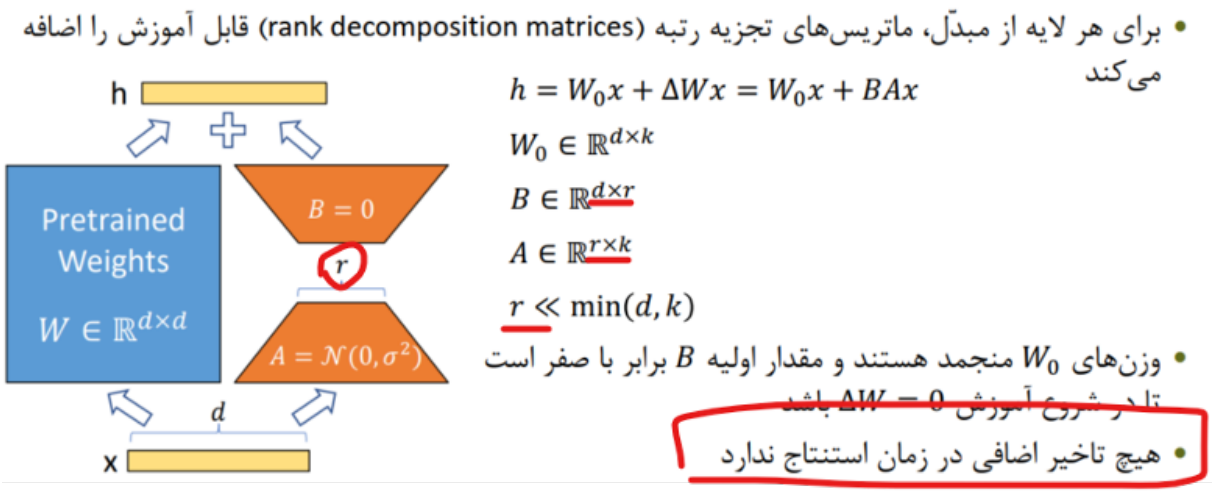
$$\text{Parameters} = 2 \times d_{\text{model}} \times r$$

بنابراین برای هر یک از ۴ ماتریس تعداد پارامترهای قابل آموزش برابر با $32,768$ می‌باشد.
برای هر کدام از تسک‌های sentiment و question answering:
پارامترهای قابل آموزش: $24 \times 4 \times 32,768 = 3,145,728$ پارامتر.
بنابراین، برای هر تسک (Question Answering یا Sentiment Analysis)، تنها حدود 3.15 میلیون پارامتر آموزش داده می‌شود. برای استنتاج، مدل اصلی و بزرگ BERT تنها یک بار ذخیره می‌شود. سپس برای هر تسک، فقط ماتریس‌های کوچک LoRA که آموزش داده شده‌اند، ذخیره می‌شوند.
پارامترهای ذخیره‌سازی برای inference: مدل اصلی (freeze شده): $100,663,296$ پارامتر + پارامترهای LoRA: $3,145,728$ پارامتر که در کل برابر با $103,809,024$ پارامتر می‌شود.

تنظیم دقیق معمولی (Full Fine-Tuning): در این روش، تمام پارامترهای اصلی ماتریس‌های attention در مدل BERT به‌طور کامل آپدیت و آموزش داده می‌شوند. همانطور که بالاتر اشاره شد تعداد کل پارامترها در 24 بلوک برابر با $100,663,296$ می‌باشد. بنابراین، برای هر تسک، حدود 100.7 میلیون پارامتر باید آموزش داده شود که بیش از 32 برابر حالت LoRA است.

(ج) با توجه به مقاله LoRA، توضیح دهید که چه میزان تأخیر (latency) در مرحله inference به مدل اضافه می‌شود. درستی جواب خود را اثبات کنید. همچنین بررسی کنید که مرتبه (rank) در کدام بخش از شبکه تاثیرگذار است. (۵ نمره)

بر اساس مقاله LoRA، این تکنیک هیچ تأخیر (latency) اضافه‌ای در مرحله استنتاج (inference) به مدل تحمیل نمی‌کند. عملکرد LoRA بر اساس افزودن یک مسیر موازی به ماتریس وزن‌های از پیش‌آموخته (W_0) است. خروجی یک لایه با LoRA به صورت شکل ۱ محاسبه می‌شود:
در مرحله استنتاج، تمام وزن‌ها ثابت هستند. از آنجایی که هم W_0 و هم ماتریس‌های



شکل ۱: محاسبه خروجی یک لایه LoRA

آموزش‌دیده B و A ثابت هستند، می‌توان حاصل‌ضرب BA را یک بار محاسبه کرد و با ماتریس اصلی ادغام نمود تا یک ماتریس وزن جدید (W') به دست آید. این محاسبه فقط یک بار قبل از شروع به کار مدل (deployment) انجام می‌شود. پس از آن، در زمان استنتاج، محاسبات به همان شکل مدل اصلی و بدون LoRA انجام می‌شود. بنابراین، از آنجایی که محاسبات زمان اجرا دقیقاً معادل یک ضرب ماتریسی (مانند مدل اصلی) است، هیچ تأخیر محاسباتی اضافی وجود ندارد.

عدم وجود تأخیر اضافه در استنتاج. زمانی که مدل در محیط عملیاتی به کار گرفته می‌شود، می‌توانیم به صورت صریح ماتریس $W = W_0 + BA$ را محاسبه و ذخیره کرده و استنتاج را مانند همیشه انجام دهیم. توجه داشته باشید که هر دو ماتریس W_0 و BA در فضای $\mathbb{R}^{d\times k}$ قرار دارند. هنگامی که نیاز به تغییر به یک تسک پایین‌دستی دیگر داریم، می‌توانیم با کم کردن BA از W_0، آن را بازیابی کرده و سپس یک ماتریس متفاوت $B'A'$ را به آن اضافه کنیم؛ این یک عملیات سریع با سربار حافظه بسیار ناچیز است.

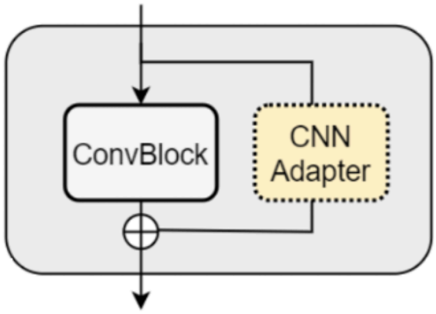
مرتبه یا رنک (r) در LoRA، مستقیماً بر تعداد پارامترهای قابل آموزش و در نتیجه بر اندازه آداپتور LoRA تأثیر می‌گذارد. رنک (r) بُعد میانی ماتریس‌های تجزیه‌شده A و B را تعیین می‌کند. این بدان معناست که:

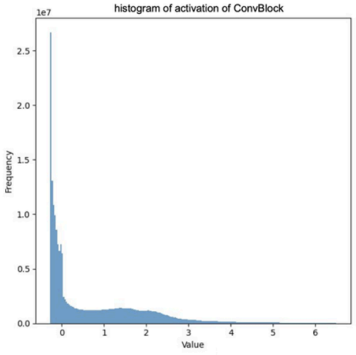
- **رنک بالاتر (Higher Rank):** تعداد پارامترهای قابل آموزش را افزایش می‌دهد. ظرفیت مدل برای یادگیری اطلاعات جدید و ویژه تسک را افزایش می‌دهد.
- **رنک پایین‌تر (Lower Rank):** تعداد پارامترهای قابل آموزش را کاهش می‌دهد و مدل را بسیار بهینه‌تر می‌کند. ظرفیت مدل برای یادگیری را محدودتر می‌کند، که ممکن است

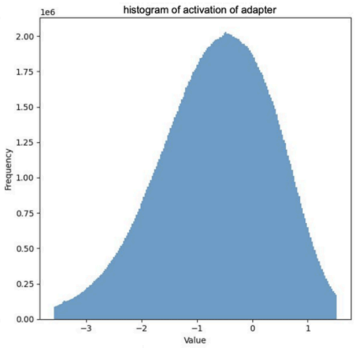
برای تسک‌های پیچیده کافی نباشد.

بنابراین، رنک به عنوان یک ابرپارامتر (hyperparameter) کلیدی عمل می‌کند که یک **trade-off** بین کارایی پارامتری (parameter efficiency) و قدرت مدل در حین فرآیند تنظیم دقیق (fine-tuning) ایجاد می‌کند.

(د) در تصویر زیر، یک معماری آداپتور برای شبکه‌های کانولوشنی نمایش داده شده است که ماژول "آداپتور" به طور موازی با یک بلوک کانولوشنی (ConvBlock) عمل می‌کند و خروجی آن به خروجی بلوک کانولوشنی اضافه می‌شود. در تصویر بعدی، هیستوگرام‌های feature map های خروجی هر دو ماژول نشان داده شده‌اند. این هیستوگرام‌ها چه تفاوت‌هایی دارند و دلیل این تفاوت‌ها چیست؟ این تفاوت‌ها چه مشکلاتی می‌توانند ایجاد کنند و برای رفع این مشکلات چه راه‌حل‌هایی پیشنهاد می‌دهید؟ (۵ نمره)







هیستوگرام خروجی convblock کاملاً نامتقارن و دارای چولگی به راست است. یک قله بسیار بزرگ و تیز در مقدار صفر وجود دارد و بقیه مقادیر در بازه مثبت پراکنده شده‌اند. درحالی که هیستوگرام ماژول آداپتور کاملاً متقارن است و شباهت زیادی به توزیع نرمال (گوسی) دارد. یکی از دلایل شکل هیستوگرام convblock این است که این بلوک بخشی از یک شبکه از پیش‌آموخته است. وجود توابع فعال‌سازی مانند ReLU (Rectified Linear Unit) پس از لایه‌های کانولوشنی، تمام مقادیر منفی را صفر می‌کند. این کار باعث ایجاد قله بزرگ در نقطه صفر و حذف مقادیر منفی می‌شود. وزن‌های این بلوک طی فرآیند آموزش روی داده‌های زیاد، به گونه‌ای تنظیم شده‌اند که ویژگی‌های معناداری استخراج کنند که منجر به چنین توزیع فعال‌سازی sparse ای می‌شود. اما ماژول اداپتور به تازگی به شبکه اضافه شده و هنوز به طور کامل آموزش ندیده است. همچنین وزن‌های آداپتور معمولاً با مقادیر تصادفی کوچک و برگرفته از یک توزیع نرمال (با میانگین صفر) مقداردهی اولیه می‌شوند. در نتیجه، خروجی اولیه آن

نیز شبیه به یک توزیع نرمال با میانگین صفر و واریانس پایین خواهد بود. این کار به این دلیل انجام می‌شود که در ابتدای آموزش، آداپتور نباید رفتار مدل از پیش‌آموخته را مختل کند. از سوی دیگر، ماژول آداپتور ممکن است از تابع فعال‌سازی متفاوتی مانند Tanh استفاده کند یا حتی فاقد تابع فعال‌سازی محدودکننده باشد. این طراحی اجازه می‌دهد که مقادیر منفی در فعال‌سازی‌ها حفظ شوند و محدوده وسیع‌تری از مقادیر تولید شود. علاوه بر این، نقش ماژول آداپتور به عنوان یک مسیر مکمل در معماری شبکه به گونه‌ای است که ویژگی‌های متفاوتی نسبت به بلوک کانولوشنی استخراج می‌کند، که می‌تواند به پراکندگی بیشتر و توزیع نرمال‌مانند منجر شود.

این تفاوت‌ها در توزیع فعال‌سازی‌ها می‌توانند هنگام جمع شدن خروجی‌های دو ماژول مشکلاتی ایجاد کنند. نخست، عدم تطابق در مقیاس و توزیع فعال‌سازی‌ها ممکن است باعث شود که یکی از ماژول‌ها بر دیگری غالب شود. برای مثال، اگر مقادیر ماژول آداپتور به دلیل پراکندگی بیشتر، دامنه وسیع‌تری داشته باشند، ممکن است تأثیر خروجی بلوک کانولوشنی را تحت‌الشعاع قرار دهند یا نویز ناخواسته‌ای به خروجی نهایی اضافه کنند. همچنین از آنجا که تأثیر اولیه آداپتور بر خروجی نهایی بسیار کم است، گرادیانی که برای به‌روزرسانی وزن‌های آن محاسبه می‌شود نیز بسیار کوچک خواهد بود (مشابه مشکل محو شدگی گرادیان یا Vanishing Gradient). در نتیجه، فرآیند یادگیری آداپتور بسیار کند شده یا به درستی انجام نمی‌شود. هدف آداپتور، ایجاد تغییرات ظریف در رفتار مدل است. اگر خروجی آن در مقایسه با بلوک اصلی ناچیز باشد، مدل نمی‌تواند به طور مؤثری خود را با داده‌های جدید تطبیق دهد.

برای حل مشکل عدم تطابق مقیاس و اطمینان از آموزش مؤثر آداپتور، راه‌حل‌های زیر پیشنهاد می‌شود:

- **استفاده از ضریب مقیاس‌پذیر:** این رایج‌ترین و مؤثرترین راه‌حل است. خروجی آداپتور قبل از جمع شدن با خروجی بلوک اصلی، در یک ضریب اسکالر s ضرب می‌شود. این ضریب s می‌تواند یک ابرپارامتر ثابت (مثلاً $s = 2$) یا یک پارامتر قابل یادگیری باشد که به شبکه اجازه می‌دهد به طور خودکار بهترین مقیاس را برای خروجی آداپتور پیدا کند. این کار باعث تقویت سیگنال آداپتور شده و به آن اجازه می‌دهد تأثیر معناداری بر خروجی و فرآیند یادگیری بگذارد.
- **نرمال‌سازی لایه (Layer Normalization):** می‌توان خروجی ConvBlock را قبل از عملیات جمع، با استفاده از یک لایه نرمال‌سازی (مانند LayerNorm) پردازش کرد. این کار باعث می‌شود خروجی بلوک اصلی دارای میانگین صفر و واریانس واحد شود و از نظر آماری به خروجی اولیه آداپتور نزدیک‌تر گردد.

۳.  فرض کنید در حال کار روی یک پروژه طبقه‌بندی تصاویر پزشکی برای تشخیص یک بیماری نادر هستید. شما با دو چالش اصلی روبرو هستید:

- تعداد تصاویر برچسب‌خورده بسیار محدود است.
- مشخص نیست که چه نوع معماری شبکه‌ای برای این داده‌های خاص بهترین عملکرد را خواهد داشت.

با الهام از روش‌های معرفی شده در درس یک راهبرد[1] جامع برای ساخت یک مدل طبقه‌بندی با کارایی بالا پیشنهاد دهید. در پاسخ خود به موارد زیر بپردازید(۱۵ نمره):

- برای انتخاب معماری، کدام رویکرد را انتخاب می‌کنید؟دلیل انتخاب خود را توضیح دهید.
- مزایا و معایب انتخاب شما در این شرایط خاص چیست؟
- چالش‌های اصلی در راهبرد پیشنهادی شما کدامند؟

مرحله اول: انتخاب معماری شبکه

رویکرد پیشنهادی: استفاده از روش جستجوی معماری مانند DARTS [2].

- **عدم قطعیت در معماری:** در حوزه‌های تخصصی مانند تصاویر پزشکی، معماری‌های استاندارد (که برای تصاویر طبیعی طراحی شده‌اند) لزوماً بهترین عملکرد را ندارند. ویژگی‌های بصری در تصاویر پزشکی (مانند بافت‌ها، مرزهای ظریف و کنتراست‌های خاص) ممکن است به ساختارهای متفاوتی در شبکه عصبی نیاز داشته باشند.
- **کارایی محاسباتی:** چالش اصلی در پروژه‌های پزشکی، محدودیت منابع محاسباتی است. روش‌های جستجوی معماری مبتنی بر یادگیری تقویتی (RL) یا الگوریتم‌های تکاملی، هزینه‌هایی معادل هزاران ساعت-GPU دارند که آن‌ها را غیرعملی می‌سازد. در مقابل، DARTS با هزینه جستجوی تنها ۵.۱ تا ۴ ساعت-GPU، یک جایگزین بسیار کارآمد است که می‌تواند یک معماری بهینه و متناسب با داده‌های پزشکی موجود را کشف کند. این روش می‌تواند بلوک‌های سازنده (operations) خاصی را کشف کند که برای استخراج ویژگی از این نوع تصاویر مناسب‌تر هستند.
- **جایگزین:** اگر منابع محاسباتی برای اجرای DARTS نیز فراهم نباشد، یک گزینه جایگزین خوب، استفاده از معماری DenseNet است. این معماری به دلیل پارامترهای کمتر و استفاده

[1] Strategy

[2] Differentiable Architecture Search

مجدد از ویژگی‌ها (feature reuse) شناخته شده و نشان داده است که در شرایط کمبود داده عملکرد بسیار خوبی دارد.

مرحله دوم: استراتژی داده‌افزایی

رویکرد پیشنهادی: استفاده از Fast AutoAugment برای کشف خودکار سیاست‌های داده‌افزایی.

- **مقابله با Overfitting:** با توجه به محدودیت شدید داده‌ها، داده‌افزایی برای جلوگیری از بیش‌برازش (Overfitting) و بهبود تعمیم‌پذیری مدل حیاتی است.
- **نیاز به داده‌افزایی تخصصی:** داده‌افزایی در تصاویر پزشکی بسیار حساس است. یک چرخش یا تغییر رنگ شدید که برای تصاویر طبیعی بی‌خطر است، می‌تواند معنای تشخیصی یک تصویر پزشکی را کاملاً تغییر دهد. برای مثال، برعکس کردن افقی یک تصویر رادیوگرافی قفسه سینه می‌تواند موقعیت قلب را جابجا کرده و منجر به تشخیص اشتباه شود. بنابراین، تعیین دستی پارامترهای مناسب (مانند حداکثر زاویه چرخش یا میزان تغییر کنتراست) دشوار و نیازمند تخصص پزشکی است.
- **راه‌حل خودکار و کارآمد:** Fast AutoAugment این مشکل را با جستجوی خودکار در فضای پارامترهای داده‌افزایی حل می‌کند. با تعریف یک فضای جستجوی اولیه شامل عملیات‌های "امن" برای تصاویر پزشکی (مانند چرخش‌های جزئی، تغییرات کنتراست و روشنایی محدود، و تغییر شکل‌های الاستیک)، این الگوریتم می‌تواند بهترین ترکیب، احتمال و شدت این عملیات‌ها را به صورت خودکار و متناسب با داده‌های موجود پیدا کند. هزینه محاسباتی پایین آن نیز این روش را برای این سناریو کاملاً عملی می‌سازد.

چالش‌ها

- **هزینه در مقابل دقت:** با وجود کارایی بالا، ترکیب DARTS و Fast AutoAugment همچنان از آموزش یک مدل استاندارد روی داده‌های موجود پرهزینه‌تر است. این یک بده‌بستان بین سرمایه‌گذاری محاسباتی اولیه برای جستجو و دقت بالقوه بالاتر در مدل نهایی است.
- **نمایندگی داده‌ها:** موفقیت این استراتژی به شدت به این وابسته است که مجموعه داده محدود اولیه، نماینده خوبی از توزیع واقعی داده‌های آن بیماری باشد. اگر داده‌های اولیه سوگیرانه (biased) باشند، معماری و سیاست‌های کشف‌شده نیز بهینه نخواهند بود.
- **نیاز به تعریف فضای جستجوی اولیه:** اگرچه فرآیند جستجو خودکار است، اما تعریف فضای عملیات اولیه برای داده‌افزایی نیازمند مقداری دانش دامنه است تا از اعمال تبدیلات نامعتبر از نظر پزشکی جلوگیری شود.

۴. Fast AutoAugment به عنوان یک راهکار برای رفع مشکل اصلی AutoAugment، یعنی هزینه محاسباتی بالا، معرفی شده است. با توجه به الگوریتم و توضیحات ارائه شده در درس، به سوالات زیر پاسخ دهید(۱۵ نمره):

(آ) ایده کلیدی پشت Fast AutoAugment که آن را سریع‌تر می‌کند، چیست؟ مفهوم تطابق چگالی[۳] را توضیح دهید.

(ب) استراتژی تقسیم داده‌ها به K-fold و استفاده از مجموعه‌های $D_{\mathcal{A}}^{(k)}, D_{\mathcal{M}}^{(k)}$ چگونه به کاهش هزینه محاسباتی کمک می‌کند؟

(ج) با استفاده از جدول مقایسه هزینه محاسباتی ، تفاوت سرعت این روش با AutoAugment را برای مجموعه داده ImageNet به صورت کمی بیان کرده و اهمیت این بهبود را تحلیل کنید.

روش Fast AutoAugment برای حل بزرگ‌ترین نقطه ضعف AutoAugment، یعنی هزینه محاسباتی بسیار بالا، طراحی شد. این بهبود از طریق یک نوآوری کلیدی در استراتژی جستجو و یک رویکرد هوشمندانه برای ارزیابی سیاست‌های داده‌افزایی به دست آمده است.

- **الف) ایده کلیدی: تطابق چگالی**

ایده اصلی و نوآورانه در Fast AutoAugment، جایگزین کردن هدف جستجو است. در حالی که AutoAugment اصلی سعی می‌کرد سیاستی را پیدا کند که دقت مدل را در مجموعه اعتبارسنجی بیشینه کند (فرآیندی که نیازمند آموزش کامل یک مدل برای هر سیاست بود)، Fast AutoAugment هدف متفاوتی را دنبال می‌کند: پیدا کردن سیاست‌های داده‌افزایی که توزیع داده‌های افزایش‌یافته را به توزیع داده‌های اصلی نزدیک‌تر کنند. این ایده با عنوان "تطابق چگالی" یا "همخوانی توزیع" شناخته می‌شود. منطق پشت این ایده این است که یک سیاست داده‌افزایی "خوب"، داده‌های جدیدی تولید می‌کند که از نظر آماری شبیه به داده‌های واقعی مجموعه آموزشی هستند و مدل را سردرگم نمی‌کنند. این رویکرد به الگوریتم اجازه می‌دهد تا کیفیت یک سیاست داده‌افزایی را بدون نیاز به اجرای یک فرآیند آموزش کامل و پرهزینه، سریع و کارآمد ارزیابی کند. به جای یادگیری تقویتی، این روش از Bayesian Optimization برای جستجوی کارآمد در فضای پیوسته پارامترهای داده‌افزایی استفاده می‌کند.

- **ب) نقش استراتژی K-fold در کاهش هزینه محاسباتی**

[3] Density Matching

- **تقسیم K-fold:** ابتدا، مجموعه داده آموزشی D_{train} به K بخش تقسیم می‌شود. این کار به الگوریتم اجازه می‌دهد تا جستجو را به صورت موازی روی بخش‌های مختلف داده انجام دهد.
- **تقسیم داخلی:** هر بخش خود به دو زیرمجموعه تقسیم می‌شود:
  - $D_M^{(k)}$: بخشی از داده‌ها که برای آموزش یک مدل پایه $M(\theta)$ استفاده می‌شود.
  - $D_A^{(k)}$: بخشی دیگر که برای ارزیابی سیاست‌های داده‌افزایی به کار می‌رود.
- **یک بار آموزش، ارزیابی چندباره:** مدل پایه $M(\theta)$ برای هر fold فقط یک بار روی $D_M^{(k)}$ آموزش می‌بینند. سپس، تعداد زیادی (B) سیاست داده‌افزایی روی $D_A^{(k)}$ با استفاده از همین مدلِ از قبل آموزش‌دیده ارزیابی می‌شوند.

این استراتژی، گلوگاه محاسباتی AutoAugment اصلی را از بین می‌برد، زیرا دیگر نیازی نیست که به ازای هر سیاست کاندید، یک مدل از ابتدا تا انتها آموزش داده شود.

- **ج) مقایسه کمی هزینه محاسباتی**

تفاوت در کارایی این دو روش در مقایسه مستقیم هزینه‌های محاسباتی آن‌ها مشهود است.

- AutoAugment: ۱۵۰۰۰ ساعت-GPU
- Fast AutoAugment: ۴۵۰ ساعت-GPU

این اعداد نشان‌دهنده یک کاهش هزینه بیش از ۳۳ برابر است. این بهبود چشمگیر، تکنیک جستجوی خودکار داده‌افزایی را از یک پروژه تحقیقاتی بسیار گران‌قیمت که تنها در توان شرکت‌های بزرگ بود، به ابزاری عملی و قابل دسترس برای جامعه گسترده‌تری از محققان و توسعه‌دهندگان تبدیل کرده است.

سوالات عملی

۵.  در این تمرین، قصد داریم فرآیند بهینه‌سازی یک شبکه عصبی را به صورت عملی تجربه کنیم. این فرآیند با ساخت یک مدل پایه آغاز شده و با پیاده‌سازی و مقایسه طیف وسیعی از تکنیک‌های بهینه‌سازی هایپرپارامتر، از روش‌های کلاسیک مانند جستجوی شبکه‌ای تا الگوریتم‌های پیشرفته مانند Optuna با قابلیت هرس کردن و الگوریتم ژنتیک، ادامه می‌یابد. وظیفه اصلی شما، تکمیل بخش‌های مشخص‌شده در نوت‌بوک HyperparameterOptimization.ipynb است. در نهایت، از

شما خواسته می‌شود تا نتایج روش‌های مختلف را تحلیل کرده و کارایی و عملکرد آن‌ها را با یکدیگر مقایسه نمایید(۳۰ نمره).

لطفا به فایل hyperparameteroptimization-answer.ipynb مراجعه نمایید.