یادگیری عمیق

تمرین هفتم

استاد درس: دکتر محمدی

مهسا موفق بهروزی

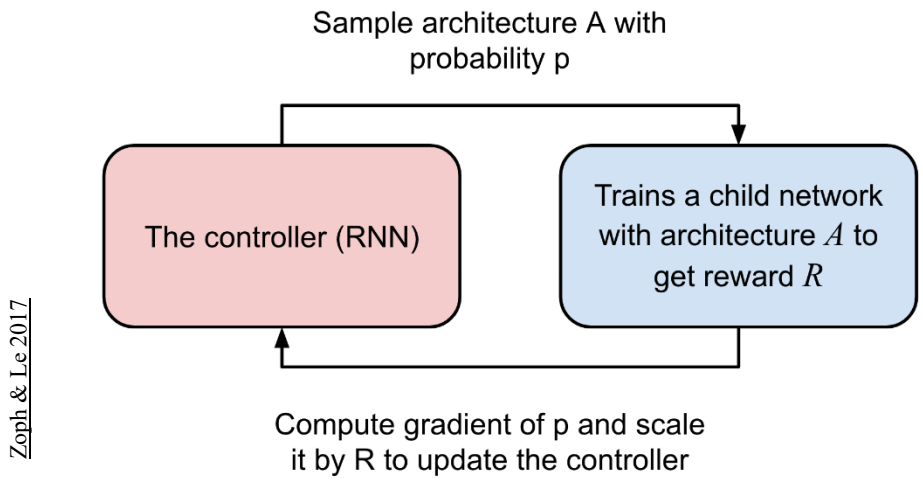
تابستان ۱۴۰۲

سوال یک)

الف)

یادگیری تقویتی

طراحی اولیه‌ی NAS شـــامل یک کنترل‌کننده مبتنی بر RL برای پیشـــنهاد معماری می‌باشـــد. کنترل‌کننده به‌صورت یک RNN پیاده‌سازی می‌شود و به عنوان خروجی یک دنباله با طول متغیر از توکن‌های مورد استفاده برای پیکربندی یک معماری شبکه را برمی‌گرداند.



نمای کلی سطح بالایی از NAS، شامل یک کنترل کننده RNN و یک خط لوله برای ارزیابی

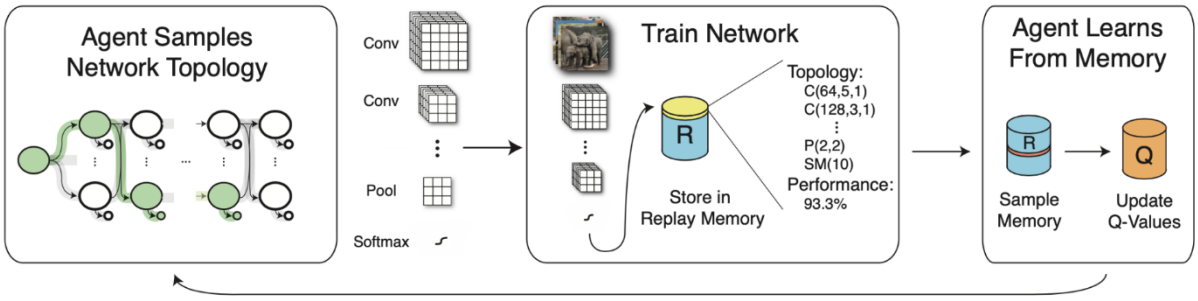
Action space: فهرســـتی از tokenها برای تعریف شـــبکه فرزندِ پیش‌بینی‌شـــده توســـط کنترل‌کننده اســـت.
Controller، عمل $a_{1:T}$ را که در آن T تعداد کل توکن‌هاست، تولید می‌کند.
Reward: دقت شبکه فرزند به عنوان پاداش کنترل‌کننده R در نظر گرفته شود.
Loss: NAS پارامترهای کنترلر (θ) را با REINFORCE loss بهینه می‌کند. به دنبال به حداکثر رساندن پاداش مورد انتظار (دقت) با اسـتفاده از گرادیان هسـتیم. و نکته مثبت policy gradient این اسـت که حتی اگر پاداش مشتق‌پذیر نباشد نیز کار می‌کند.

$$\nabla_\theta J(\theta) = \sum_{t=1}^{T} \mathbb{E}\left[\nabla_\theta \log P(a_t | a_{1:(t-1)}; \theta) R\right]$$

MetaQNN یک عامل را آموزش می‌دهد تا به طور متوالی لایه‌های CNN را با استفاده از یادگیری Q با یک استراتژی اکتشاف epsilon-greedy و experience replay انتخاب کند. پاداش نیز دقت validation است.

$$Q^{(t+1)}(s_t, a_t) = (1-\alpha)Q^{(t)}(s_t, a_t) + \alpha(R_t + \gamma \max_{a \in \mathcal{A}} Q^{(t)}(s_{t+1}, a'))$$

حالت s_t یک چندتایی از عملیات لایه‌ای (عملیات لایه‌ای به نوع یا پیکربندی خاصی از یک لایه در یک شبکه عصبی گفته می‌شود) و پارامترهای مرتبط است؛ و عمل a اتصال بین عملیات را تعیین می‌کند. *Q-value* متناسب با میزان اطمینان ما در دو عملیات متصل است که منجر به دقت بالا می‌شود.



Baker et al. 2017

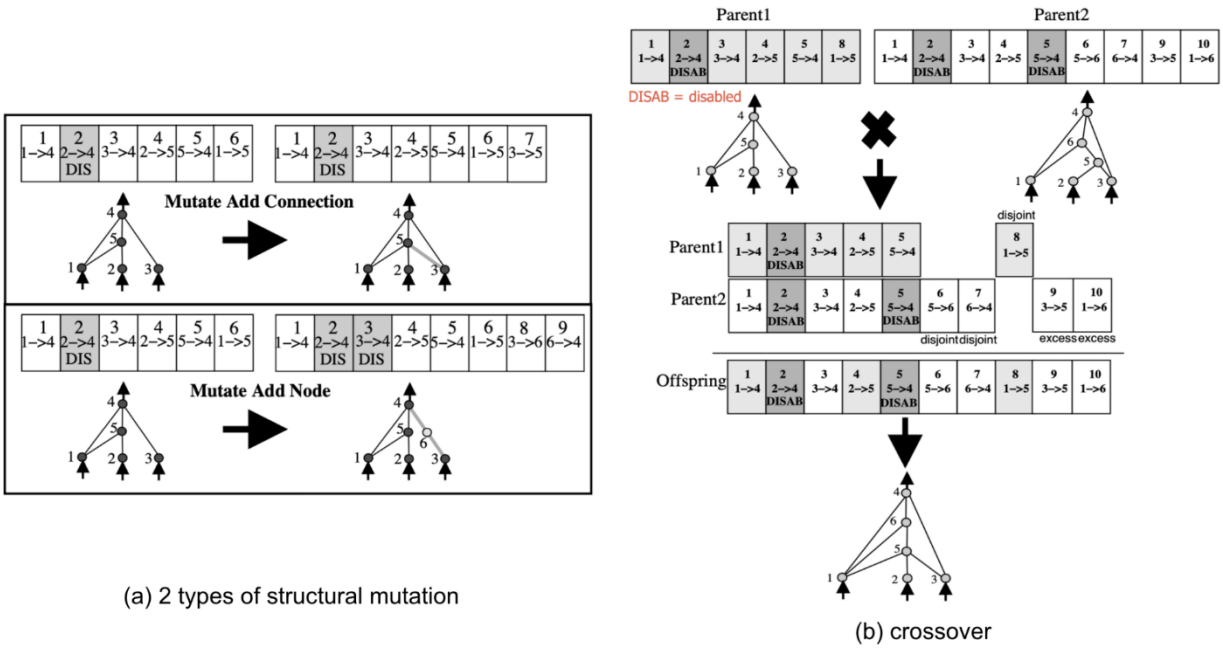
مروری بر MetaQNN – طراحی مدل‌های CNN با Q-Learning

الگوریتم‌های تکاملی

NEAT (مخفف NeuroEvolution of Augmenting Topologies) رویکردی برای تکامل توپولوژی شبکه‌های عصبی با الگوریتم ژنتیک (GA) است که توسط Miikkulainen & Stanley در سال ۲۰۰۲ پیشنهاد شد. NEAT وزن‌های اتصال و توپولوژی شبکه را با هم تکامل می‌دهد. هر ژن اطلاعات کامل پیکربندی یک شبکه را رمزگذاری می‌کند. جمعیت با اعمال جهش بر روی وزن‌ها و اتصالات و همچنین crossover بین دو ژن والد رشد می‌کند.

Real و همکاران الگوریتم‌های تکاملی (EA) را به عنوان راهی برای جستجوی معماری‌های شبکه با کارایی بالا، به نام AmoebaNet، اتخاذ کردند و از روش tournament selection برای این کار استفاده می‌کنند که در هر تکرار، بهترین کاندیدا را از میان مجموعه تصادفی نمونه‌ها انتخاب می‌کند و فرزندان جهش یافته آن را دوباره در جمعیت قرار می‌دهد.

Stanley & Miikkulainen, 2002



جهش در الگوریتم NEAT

AmoebaNet، tournament selection را تغییر داد تا به نفع ژنوتیپ‌های جوان‌تر باشد و همیشه قدیمی‌ترین مدل‌ها را در هر چرخه کنار بگذارد. چنین رویکردی که aging evolution نام دارد، به AmoebaNet اجازه می‌دهد تا فضای جستجوی بیشتری را پوشش دهد و کاوش کند، نه اینکه مدل‌هایی با عملکرد خوب را خیلی زود محدود کند.

در هر چرخه از tournament selection با منظم‌سازی aging:

۱) S مدل از جمعیت انتخاب شده و مدلی با بالاترین دقت به عنوان والد (parent) انتخاب می‌شود.

۲) یک مدل فرزند (child model) با اعمال جهش بر روی والد تولید می‌شود.

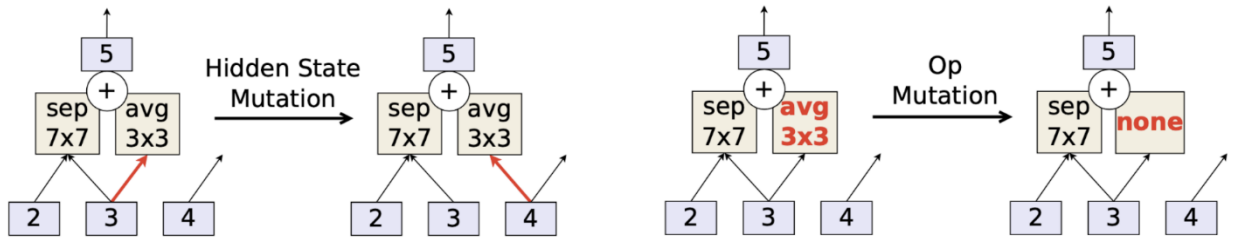
۳) مدل فرزند، train و ارزیابی شده و به جمعیت اضافه می‌گردد.

۴) قدیمی‌تریم مدل از جمعیت حذف می‌شود.

دو نوع جهش اعمال می‌شود:

Hidden state mutation و Operation mutation

Real et al. 2018



دو نوع جهش در AmoebaNet

در آزمایشات Real و همکاران، EA و RL از نظر دقت validation نهایی، هر دو به یک اندازه خوب کار می‌کنند اما EA عملکرد بهتری در هر زمان دارد و می‌تولند مدل‌های کوچک‌تری را پیدا کند. البته استفاده از EA در NAS از نظر محاسباتی گران است و هر آزمایش ۷ روز با ۴۵۰ GPU طول کشیده است.

HNAS (Liu و همکاران ۲۰۱۷) نیز از الگوریتم‌های تکاملی (original tournament selection) به عنوان استراتژی جست‌وجوی خود استفاده می‌کنند. در فضای جستجوی ساختار سلسله مراتبی، هر یال یک عملیات است. بنابراین جهش ژنوتیپ در آزمایشات آن‌ها با جایگزینی یک لبه تصادفی با یک عملیات متفاوت اعمال می‌شود. مجموعه جایگزین شامل یک عملیات none است، بنابراین می‌تواند یک لبه را تغییر دهد، حذف و اضافه کند. مجموعه اولیه ژنوتیپ‌ها با اعمال تعداد زیادی جهش تصادفی روی "الگوهای ساده" (تمامی تطابق‌های هویتی) ایجاد می‌شود.

گرادیان کاهشی

رویکردهای مبتنی بر گرادیان کاهشی در جستجوی معماری شبکه عصبی شامل به‌روزرسانی مکرر پارامترهای معماری شبکه عصبی در جهتی است که یک تابع خطا تعریف شده را به حداقل می‌رساند. مراحل کلی استفاده از آن برای جستجوی معماری به شرح زیر است:

۱. تعریف فضای جست‌وجو: ابتدا باید یک فضای جست‌وجو تعریف شود که مجموعه‌ای از معماری‌های ممکن را برای کاوش نشان می‌دهد. فضای جست‌وجو می‌تواند شامل اجزای مختلف معماری مانند تعداد لایه‌ها، انواع لایه‌ها (به عنوان مثال، کانولوشن، بازگشتی و ...)، اتصالات بین لایه‌ها، ابرپارامترها و .... باشد.
۲. مقداردهی اولیه پارامترهای معماری: پارامترهای معماری که پیکربندی خاصی از معماری شبکه عصبی را در فضای جستجو تعیین می‌کند، مقداردهی اولیه می‌کنیم. این پارامترها را می توان به صورت تصادفی مقداردهی اولیه کرد یا روی برخی مقادیر اولیه تنظیم کرد.

۳. ارزیابی عملکرد معماری: آموزش و ارزیابی عملکرد معماری شبکه عصبی با مجموعه فعلی پارامترهای معماری در یک task معین. عملکرد را می توان با استفاده از یک متریک از پیش تعریف شده، مانند دقت یا ضرر اندازه‌گیری کرد.
۴. محاسبه گرادیان: گرادیان پارامترهای معماری را با توجه به معیار عملکرد انتخابی محاسبه می‌کنیم.
۵. به روز رسانی پارامترهای معماری: با برداشتن یک گام در جهت مخالف گرادیان‌های محاسبه شده، پارامترهای معماری را به‌روز می‌کنیم. این مرحله برای به حداقل رساندن عملکرد خطا انجام می‌شود. به‌روزرسانی را می‌توان با استفاده از انواع مختلف گرادیان کاهشی، مانند SGD، آدام، یا RMSprop انجام داد.
۶. مراحل ۳ تا ۵ را تا زمانی که شرط خاتمه برآورده شود تکرار می‌کنیم. این شرط می‌تواند تعداد تکرار، همگرایی پارامترهای معماری یا رسیدن به آستانه عملکرد رضایت بخش باشد.

۷. انتخاب بهترین معماری: پس از تکمیل فرآیند جست‌وجو، معماری با بهترین عملکرد را بر اساس معیار ارزیابی انتخاب می‌کنیم. این معماری نتیجه نهایی جستجوی معماری شبکه عصبی را نشان می‌دهد.

ب)

یادگیری تقویتی: می‌تواند یک رویکرد مناسب برای بهینه‌سازی پارامترهایی مانند اندازه ورودی و تعداد لایه‌های کانولوشن در مدل‌های تشخیص اشیا باشد. عامل از طریق آزمون و خطا در محیط این موارد را می‌آموزد. عامل می‌تواند ترکیب‌های مختلفی از اندازه‌های ورودی و پیکربندی‌های لایه کانولوشنی را بررسی کند، در حالی که تأثیر آن بر معیارهایی مانند دقت تشخیص شی و کارایی محاسباتی را در نظر می‌گیرد. با آموزش عامل با یک سیگنال پاداش که اهداف مورد نظر را منعکس می‌کند (به عنوان مثال، دقت بالا، هزینه محاسباتی کم)، RL می‌تواند فرآیند جست‌وجو را به سمت یافتن مقادیر پارامتر موثر هدایت کند.

یادگیری تکاملی: با ایجاد جمعیتی (population) از راه‌حل‌های کاندید با اندازه‌های ورودی مختلف و پیکربندی‌های لایه کانولوشن، EA می‌تواند فضای پارامتر را بررسی کند و افراد امیدوارکننده را بر اساس عملکرد آن‌ها انتخاب کند. از طریق فرآیندهایی مانند جهش و crossover، جمعیت در طول نسل‌ها تکامل می‌یابد و به تدریج تنظیمات پارامتر را بهبود می‌بخشد. EA می‌تواند پارامترهای گسسته‌ای مانند تعداد لایه‌های کانولوشن و پارامترهای پیوسته مانند اندازه ورودی را کنترل کند و برای یافتن ترکیب‌های بهینه که دقت، سرعت و سایر معیارها را متعادل می‌کند، مناسب است.

یادگیری مبتنی بر گرادیان: به‌ویژه تکنیک‌هایی مانند گرادیان کاهشی، معمولاً برای بهینه‌سازی پارامترهای شبکه عصبی استفاده می‌شود، اما ممکن است مستقیماً برای پارامترهای گسسته مانند تعداد لایه‌های کانولوشن قابل استفاده نباشد. با این حال، هنوز هم می‌توان از آن برای بهینه سازی پارامترهای پیوسته مانند اندازه ورودی استفاده کرد. به عنوان مثال، اندازه ورودی را می‌توان به عنوان یک متغیر پیوسته در نظر گرفت، و بهینه سازی مبتنی بر گرادیان را می‌توان برای یافتن یک مقدار بهینه در یک محدوده از پیش تعریف شده استفاده کرد. با تعریف یک تابع هدف مناسب (به عنوان مثال، به حداکثر رساندن عملکرد تشخیص شی) و به کارگیری روش‌های بهینه‌سازی مبتنی بر گرادیان، می‌توان اندازه ورودی را برای دستیابی به تعادل مطلوب بین دقت و کارایی محاسباتی تنظیم کرد.

سوال دوم)

۱. توقف زودهنگام (Early Stopping): به جای آموزش مدل‌های کاندید تا زمان همگرایی، می‌توان از تکنیک‌های توقف زودهنگام استفاده کرد که شامل نظارت بر عملکرد مدل در حین آموزش و توقف فرآیند در زمانی که عملکرد به طور قابل توجهی بهبود نمی‌یابد، است. با زودتر پایان دادن به آموزش، می‌توان زمان ارزیابی را کاهش داد و در عین حال تخمین معقولی از پتانسیل مدل به‌دست آورد.
۲. به اشتراک‌گذاری وزن (Weight Sharing): تکنیک‌های اشتراک وزن را می‌توان برای کاهش زمان آموزش برای مدل های کاندید مورد استفاده قرار داد. به جای آموزش هر مدل از ابتدا، می‌توان وزن‌ها را بین چندین مدل به اشتراک گذاشت. این رویکرد امکان محاسبات مشترک و به‌روزرسانی پارامترها را فراهم می‌کند که در نتیجه زمان‌های آموزش و ارزیابی سریع‌تر می‌شود.
۳. هرس شبکه (Network Pruning): تکنیک‌های هرس شبکه شامل حذف اتصالات یا کانال‌های کم اهمیت از شبکه است. با کاهش پیچیدگی مدل، زمان آموزش و ارزیابی را می‌توان به طور قابل توجهی کاهش داد و در عین حال سطح معقولی از عملکرد را حفظ کرد.
۴. آموزش افزایشی (Incremental training): به جای آموزش کل معماری مدل از ابتدا، می‌توان از آموزش افزایشی استفاده کرد. با شروع از معماری پایه و افزودن اجزا یا لایه‌های جدید به تدریج، صرفا با آموزش بخش‌های جدید اضافه شده، می‌توان زمان ارزیابی را کاهش داد.

سوال سوم)

الف) در روش‌های مقابله با داده‌های نامتوازن، به طور کلی دو رویکرد اتخاذ می‌شود:

- استفاده از رویکردهای داده محور مانند oversampling، undersampling و ...
- استفاده از رویکردهای مدل محور مانند تعریف کردن یک تابع ضرر کمکی، یا استفاده از class weights برای افزایش ضرر ناشی از قضاوت اشتباه هر نمونه در کلاس‌های کم‌جمعیت‌تر.

در این چالش اما، با توجه به multi-task بودن مدل، نمی‌توان از رویکردهای داده محور استفاده کرد. چرا که ممکن است تغییر در توزیع داده‌ها در جهت توازن بخشیدن داده‌ها برای یک تسک موجب بر هم خوردن توازن در توزیع داده‌ها در تسک دیگر بشود. حتی اگر بتوانیم بخشی از داده‌ها را پیدا کنیم که توزیع یکنواختی از تسک topic classification داشته باشند اما با oversampling از این زیرمجموعه از داده باعث توازن در تسک sentiment analysis شویم، متاسفانه باید گفت که مدل بر روی این بخش از داده احتمال بایاس زیادی خواهد داشت و روند آموزش مدل از حالت طبیعی (در جهت generalization) دچار آسیب خواهد شد. بنابراین در این چالش از روش‌های مبتنی بر مدل استفاده می‌کنیم. می‌توانیم با استفاده از روش class-weight برای مسئله sentiment analysis و یا استفاده از توابع ضرری مانند Focal loss برای تاکید بیشتر بر نمونه‌های درست طبقه بندی نشده استفاده کرد.

ب) یکی از چالش‌های شبکه‌های عمیق، یادگیری correlationهای سطحی برای تولید پاسخ است. این correlationها، مبتنی بر علّیت معنادار یا روابط اساسی بین featureها و target نمی‌باشند. در یک یادگیری multi task، DNN از لایه‌های مشترک برای یادگیری چندین task استفاده می‌کند. این لایه‌های مشترک، بازنمایی‌هایی را می‌آموزند که به همه وظایف مرتبط هستند. با یادگیری مشترک این بازنمایی‌ها، شبکه می‌تواند الگوها و روابط اساسی‌ای که بین taskها ثابت هستند را به جای تکیه بر correlationهای سطحی خاص یک task، بیاموزد. یادگیری بازنمایی‌های کلی و معنادارتری که به چندین task مرتبط هستند، به شبکه کمک می‌کند تا بهتر تعمیم یابد (generalization) و خطر overfitting را کاهش می‌دهد. پس از بخش مشترک، برای هر Task یک classifier مجزا در نظر گرفته می‌شود.
(اشتراک وزن‌های غیرصفر در لایه‌ی اول classifierها، نشان‌دهنده‌ی generalization مدل و اجتماع وزن‌های صفر classifierها، یعنی جایی که وزن‌های یک کلاسیفایر صفر و همان وزن‌ها در یک کلاسیفایر دیگر غیرصفراند، نشان دهنده رپرزنتیشن‌های task specific می‌باشد.)

ج) یکی از اهداف ما در استفاده از مدل‌های pretrained، انتخاب نقطه شروع مناسب (جلوگیری از شروع تصادفی) و در نهایت رسیدن به یک مدل با قابلیت generalization مناسب می‌باشد؛ زیرا که مدل‌های زبانی‌ای مانند BERT یا GPT، بر روی مجموعه‌ی بزرگی از داده‌های متنی آموزش داده شده‌اند که باعث می‌شود مدل، الگوهای زبانی عمومی را یاد بگیرد. این مدل‌های زبانی از ترنسفورمر استفاده می‌کنند، که دارای دو بخش encoder و decoder می‌باشد؛ باید توجه کنیم که ورودی این دو یکسان نبوده و ورودی decoder در هر گام، خروجی تولید شده تا به این لحظه است. با توجه به این نکته، می‌توان گفت:
مرحله اول، استفاده از وزن‌های encoder برای بخش shared می‌باشد.

در مرحله دوم، برخی از وزن‌ها را فریز می‌کنیم. این کار را برای این انجام می‌دهیم که از overfit شـــدن مدل، زمانی‌که دیتای کمی داریم، جلوگیری کنیم. می‌توان با توجه به حجم دیتاســـت، تمام وزن‌ها یا برخی از آن‌ها را فریز کرد. به طور مثال در صورتی که دیتای کافی داشته باشیم، می‌توان وزن پارامترهای key, query و value را کاملا آزاد قرار داد، در غیر این‌صـــورت value را فریز کرده و باقی را آزاد قرار دهیم. و یا حتی فقط پارامترهای بخش FFN را آزاد قرار داده و مابقی را فریز کنیم.

د) با توجه به imbalance بودن دیتا، نمی‌توان از معیار دقت استفاده کرد. به جای آن می‌توان از دقت f1 استفاده کرد که از ترکیب precision و recall استفاده می‌کند.

سوال چهارم)

الف) ماژول mapping network، از ۸ لایه‌ی کاملا متصـــل (FC) تشـــکیل شـــده اســـت و ابعاد تمام ورودی و خروجی‌های آن، از جمله z و W، ۵۱۲ اسـت. هدف mapping network، نگاشـت latent code z به یک latent space میانی‌ســـت (W) که generator را از طریق adaptive instance normalization (AdaIN) در هر لایه کانولوشن، کنترل می‌کند.
هدف Mapping network معرفی یک فضـای latent میانی‌ست که نیازی به پیروی از توزیع داده‌های آموزشـی ندارد و نســـبت به z، disentanglement بالاتری دارد. زمانی که میگویم disentanglement یک رپرزنتیشـــن بالاســـت، یا یک رپرزنتیشـــن disentanglement اســـت، یعنی مولفه های آن رپرزنتیشـــن (در اینجا W) از هم مستقل و معنادارند. این به این معنا نیست که همه ۵۱۲ مولفه آن هم، از هم مستقلند و هم، از هم معنادارند. به این معنی ست که نسبت به z مولفه‌های بیشتری مستقل و معنادارند.

ب) تفاوت اصلی style GAN با GANهای سـنتی، در اسـتفاده از latent space اسـت. در GANهای سـنتی، generator، latent code را به عنوان ورودی دریافت کرده و تصـــاویر را مســـتقیما از آن تولید می‌کند؛ که این رویکرد کنترل صـــریحی بر جنبه‌های مختلف ترکیب تصـــویر، مانند ســـبک و محتوا، ندارد. اما style GAN یک فضـــای latent میانی (W) را معرفی می‌کند که اســـتفاده از آن، منجر به جداســـازی ویژگی‌های ســـطح بالا و در نهایت تولید تصاویر با کیفیت بالاتری می‌گردد.

ج) این ماژول برای کنترل style تصـاویر تولید شـده با اعمال نرمال‌سـازی تطبیقی بر روی فیچرمپ‌های هر لایه کانولوشـــن، اســـتفاده می‌شـــود. AdaIN دو ورودی دریافت می‌کند: فیچرمپ‌ها از لایه‌ی قبل و اطلاعات style از فضـــای latent میانی w. ابتدا فیچرمپ‌ها را با تفریق میانگین و تقســـیم بر انحراف معیار، نرمال می‌کند و ســـپس تبدیل‌های affine آموخته شـــده را بر فیچرمپ‌های نرمال‌شـــده بر اســـاس اطلاعات style اعمال می‌کند. با اعمال

AdaIN در هر لایه کانولوشـن، اطلاعات style به generator تزریق شـده و امکان کنترل دقیق بر ظاهر تصـاویر تولید شده را فراهم می‌کند.

د) این ماژول برای کنترل عملیات AdaIN است و مسئول اعمال تبدیل‌های Affine آموخته شده در فضای نهان میانی است. AdaIN هر کانال از فیچرمپ ورودی را به‌گونه‌ای نرمال می‌کند که میانگین آن صفر و واریانس آن یک باشد و سپس مقیاس‌ها و بایاس‌ها را بر اساس style اعمال می‌کند. ماژول affine transform پارامترهای این مقیاس‌ها و بایاس‌ها را یاد می‌گیرد و به مولد اجازه می‌دهد تا آمار هر کانال فیچرمپ را بر اساس style خاص تغییر دهد.

ه) style mixing یک تکنیک منظم‌سازی است که شامل تولید تصاویر با ترکیب دو latent code تصادفی و جابجایی بین آنها در یک نقطه ،که به صورت رندوم انتخاب می‌شود، در شبکه سنتز است. هدف style mixing این است که از این فرض که styleهای مجاور، corelated هستند در شبکه جلوگیری کند. با فعال کردن style mixing، شبکه یاد می‌گیرد که ویژگی‌های مختلف تصویر تولید شده را به شیوه‌ای محلی‌تر و مستقل‌تر کنترل و دستکاری کند. این تکنیک به جدا کردن ویژگی‌های سطح بالا کمک می‌کند و منجر به بهبود کیفیت تصویر و تفکیک‌پذیری در فضای نهان میانی می‌شود.

سوال پنجم)

```
# Adversarial Loss
fakeB = genAB(realA)  # Generate fake images in domain B using genAB
pred_fake = discB(fakeB)  # Predict the authenticity of generated fake images using discB

loss_adv_AB = -log(1 - pred_fake)  # Compute adversarial loss for genAB using binary cross-entropy
# The adversarial loss encourages the generated images to be realistic and fool the discriminator



# Consistency Cycle Loss
reconstructedA = genBA(fakeB)  # Reconstruct original domain A from generated domain B using genBA
loss_cycle_A = abs(realA - reconstructedA)  # Compute cycle consistency loss for domain A using L1 loss
# The cycle consistency loss ensures that the generated images can be reversed back to the original domain

# Total Generator Loss
loss_genAB = loss_adv_AB + loss_cycle_A  # Combine adversarial and cycle consistency losses
# The generator tries to minimize this combined loss to improve the quality and consistency of the generated images
```

لاس Adversarial برای مولد برای به حداقل رساندن اختلاف بین پیش‌بینی‌های دیسکریمینیتور (که ورودی آن تصاویر جنریت شده است) با برچسب‌های ground truth می‌باشد.
و لاس cycle مربوط به این است که درصورت تبدیل تصویری از دامنه A به دامنه B، بازگشت آن به A باید مشابه تصویر اصلی باشد.

مشابه فرمول‌های اسلایدهای کلاس، شبه کد بالا نوشته شده است.

- گرادیان کاهشی برای شبکه مولد

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

$$\begin{aligned}\mathcal{L}_{\text{cyc}}(G, F) = &\mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] \\ &+ \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1]\end{aligned}$$