Deep Learning

Deep learning | spring 1401 | Dr.Mohammadi

Teacher Assistant:

Amirreza Fateh

Student name : **Amin Fathi**

Student id : **400722102**

Problem1.a

جستجو در فضای باز به دلیل ماهیت ترکیبی مسئله به طور غیرقابل قبولی زمان بر است بنابراین، چگونگی هرس سریع ترکیب‌های غیر ضروری عملگرها و کاهش تعداد نامزدهای احتمالی برای کاوش ضروری است چالش دیگر استفاده از الگوریتم‌های مشابه RL برای جستجو است ، این الگوریتم ها از عملکرد کاوش ضعیفی رنج می‌برد زیرا به راحتی در sub-optimal گیر می‌کنند. در نتیجه، مدل جستجو شده از فضای به‌دست‌آمده ممکن است عملکرد بهتری نسبت به مدل‌های یک فضای جستجوی طراحی‌شده دستی نداشته باشد. در واقع دو چالش اصلی که با ان رو به رو هستیم یکی پیچیدگی بیش از حد فضای جست و جو است که این خود پیچیدگی زمانی و محاسباتی و دیگری پیچیدگی محاسباتیِ فرایند ارزیابی مدل های به دست آمده در روش ها مبتنی بر NAS در فضا های جست و جوی مختلف است.

Problem1.b

به طور کلی الگوریتم های NASدر سه مرحله خلاصه میشوند. :۱طراحی یک فضای جستجو با مشخص کردن عملگرهای ابتدایی آن. .۲استفاده از یک الگوریتم جستجو برای کاوش در سطح فضای جستجو و انتخاب عملگرها در آن برای ساخت یک مدل کاندید .۳پیاده سازی یک استراتژی ارزیابی برای تأیید عملکرد مدل انتخابی. البته الگوریتم هایی اخیراً معرفی شدند که در دو مرحله ی جستجوی مدل و ارزیابی مدل خلاصه میشوند

رویه رایج الگوریتم‌های NAS اخیر برای کاهش پیچیدگی فضای جست و جوی کاوش ، استفاده از دانش قبلی انسان برای طراحی فضاهای جستجوی کوچک‌تر است، نظیر inverted residual block ها در مقالات زیر :

CVPR 2019 Open Access Repository (thecvf.com)

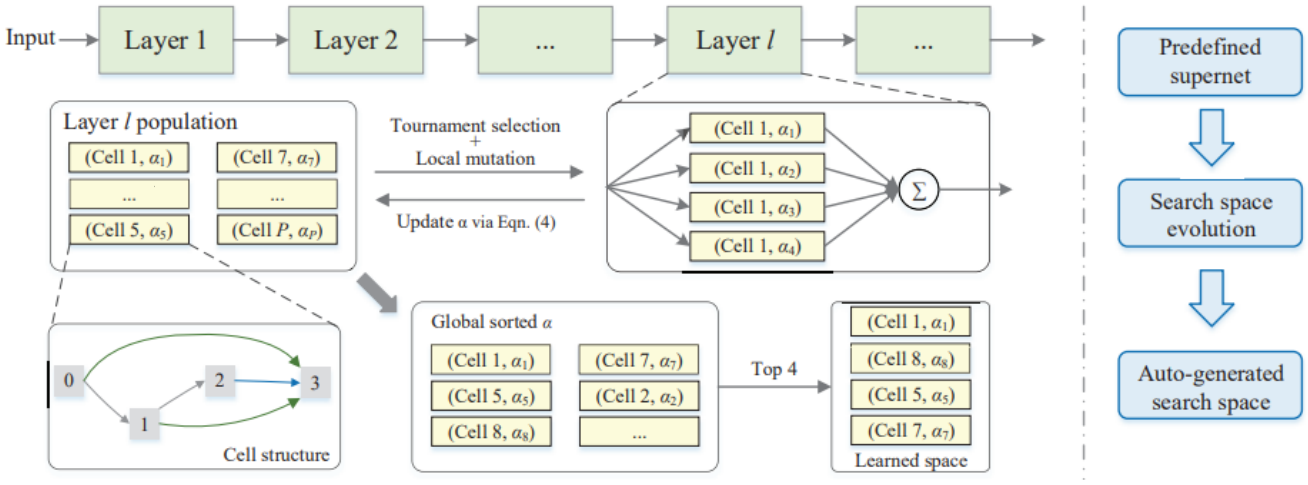
[PDF] MixConv: Mixed Depthwise Convolutional Kernels | Semantic Scholar

و یا channel shuffling block ها:

[1904.00420] Single Path One-Shot Neural Architecture Search with Uniform Sampling (arxiv.org)

همانطور که گفتیم از یک سو، استفاده از فضاهای جستجوی محدود در عمل، الگوریتم‌های NAS را قادر می‌سازد تا از کارایی بالاتری برخوردار شوند. اما از سوی دیگر، به دلیل دخالت های بسیار انسانی و محدودیت های حاصله ، امکان کشف معماری های بدیع و بهتر محدود شده است لازم به ذکر است چگونگی کاهش تلاش‌های انسان در طراحی فضای جستجو و خودکار کردن روش هنوز بسیار مورد توجه است، برای مطالعه بیشتر به مقالات زیر مراجعه شود :

[2006.02903] A Comprehensive Survey of Neural Architecture Search: Challenges and Solutions (arxiv.org)



روش پیشنهادی تولید خودکار فضای جستجو

یک الگوریتم تکاملی شامل ۳ مرحله است: ۱. تولید جمعیت ۲. ارزیابی هر فرد در جمعیت با استفاده از تابع امتیاز دهی.

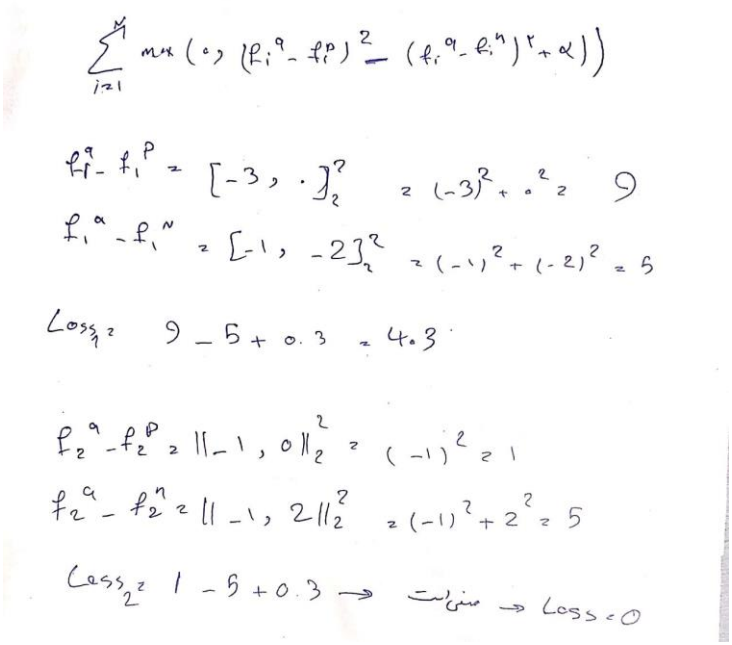
۳ .تولید جمعیت جدید از جمعیت قبلی و جهش در بعضی افراد جدید جامعه .

۲ مرحله آخر در الگوریتم های تکاملی به صورت تکرارشونده هستند تا در نهایت در جمعیت نهایی فرد یا افراد برنده به عنوان پاسخ مساله ارایه شود. این الگوریتم با چند چالش رو به رو است . چالش اول وجود افزونگی بسیاردر کل فضای جست و جو است که موجب سربار محاسباتی است . چالش دوم سربار زمانی است. چالش سوم در مورد این الگوریتم های تکاملی کند بودن همگرایی آن هاست . برای حل این سه چالش از یک DAGپیشنهادی استفاده میکنیم تا افزونگی را تا حد ممکن از بین ببرد سرعت همگرایی الگوریتم را افزایش دهد و میزان موازی سازی ارزیابی زیرفضا ها را تا حد ممکن افزایش دهد .

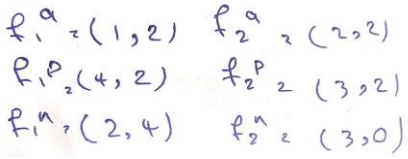
Problem2

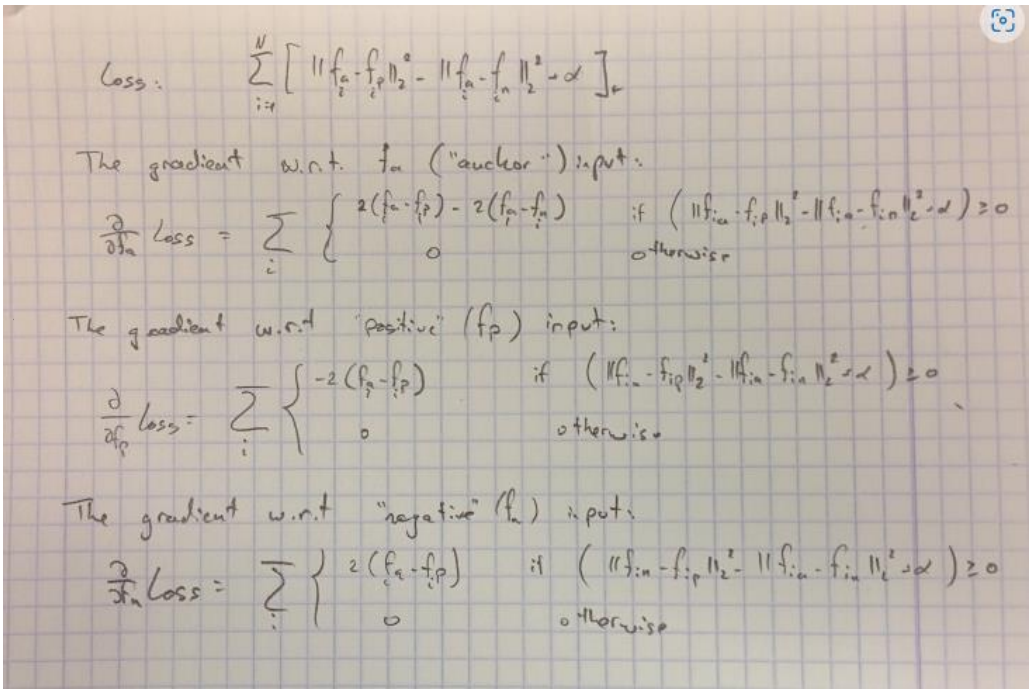
فرمول loss به شکل زیر است البته اگر مقدار منفی شود ۰ قرار می دهیم :

$$Loss = \sum_{i=1}^{N} \left[\|f_i^a - f_i^p\|_2^2 - \|f_i^a - f_i^n\|_2^2 + \alpha \right]$$

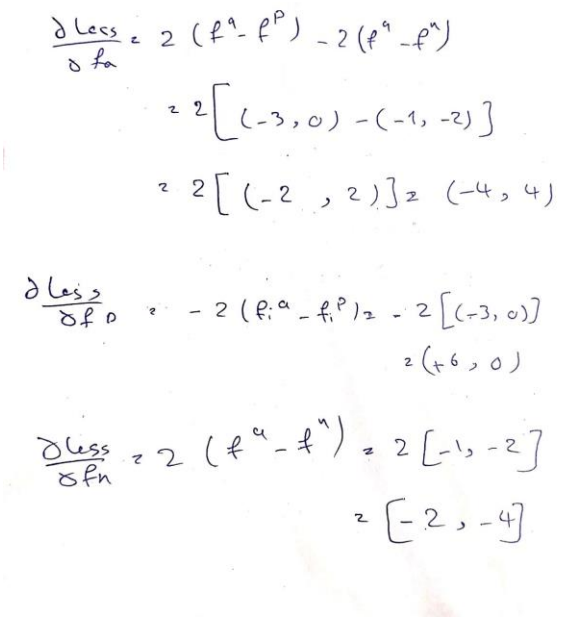


برای محاسبه loss کل فقط کافیست loss به دست آمده برای دو نمونه را با هم جمع کنیم ، تابع گردایانت قسمت ب هم به شکل زیر محاسبه می شود :





که در این مساله به شکل زیر به دست می آید .



منبع :

[computer vision - What's the triplet loss back propagation gradient formula? - Stack Overflow]

[multivariable calculus - Gradient of 2-norm squared - Mathematics Stack Exchange]

problem3

می توان فیچر ها و ویژگی هایی از source domain و target domain استخراج کرد و این دو را با هم ترکیب کرد و یک domain جدید ایجاد کرد ، در کل در این موارد بهتر است که یک domain ای که ترکیب دو domain متفاوت قبلی است را پیاده کرد و به دست آورد و از آن برای ادامه مسئله استفاده کرد حال یا بازسازی target domain با استفاده از ویژگی های استخراج شده از source domain ، یا تولید داده غیر واقعی با ترکیب دو domain . به این مشکل domain shift گویند.

Problem 4.a

یک فریم ورک بهینه سازی برای بهینه سازی ابرپارامتر هاست که می تواند بهترین هایپرپارامتر ها را برای مساله ما به دست آورد ، به صورت پیشفرض از الگوریتم بهینه سازی Bayesian استفاده می کند. به طور کلی با دوقضیه سر و کار داریم یکی sampling stategy است که در ان بهترین ترکیب پارامتر ها را با تمرکز بر مناقطی که ابرپارامتر ها نتایج بهتری تولید کرده اند ، انتخاب میکنند . مثال TPESamplet ابرپارامتر ها را در ابتدا تصادفا انتخاب میکند و آن ها را سپس مرتب می کند با توجه به score شان سپس هایپرپارامتر ها به دو گروه تقسیم شده و با استفاده از kernel density seimators بهترین پارامتر ها انتخاب میشوند .

در متد دیگر که pruning stategy نام دارد و بر پایه early stopping است، سعی می کند هنگام رسیدن به یک نقطه نامناسب فرایند جست و جوی بهترین ابرپارامتر را لغو کند و این ویژگی به طور خودکار آزمایش‌های بی‌نتیجه را در مراحل اولیه آموزش متوقف می‌کند .

Problem4.c

Purning را در بالا توضیح دادیم . ولی به نظر میرسد داده گان CIFAR100 بسیار دادگان نا منسابی برای این معماری حداکثری ذکر شده است و مدل در بهترین حالت هم نمیتواند عملکرد مناسبی داشته باشد ، لازم به ذکر است در هنگام اجرای مدل با توجه به طولانی بودن و محدودیت gpu دستی ادامه روند را قطع کرده ام.

Problem 4.d

افزایش epoch ها از ۱۰ به ۱۰۰ فقط سربار زمانی دارد و هیچ تفاوت محسوسی در عملکرد ندارد .

Problem4.e

استخوان بندی مدل ما در واقع تعداد لایه ها است و این ابرپارامتر اهیمت بالا تری نسبت به بقیه دارد ، بنابه تجربه نرخ یادگیری و نوع بهینه ساز هم از اهمیت دوم برخوردار هستند ضمن اینکه در تعامل با یکدیگر باید به دست بیایینند ، در نهایت می توان در مورد سایر ابر پارامتر ها نظیر اندازه کرنل و تعداد فیلتر ها و ... به مقدار بهینه دست یافت.

منبع :

https://towardsdatascience.com/state-of-the-art-machine-learning-hyperparameter-optimization-with-optuna-a315d8564de1

NPENAS: Neural Predictor Guided Evolution for Neural Architecture Search | IEEE Journals & Magazine | IEEE Xplore