به نام خدا

پروژه دوم درس مبانی هوش محاسباتی

دانشکده کامپیوتر

استاد درس :دکتر حسین کارشناس

دستیاران درس:

مهندس مهدی دارونی

مهندس محمدامین دادگر

مهندس امیر کرمی

اعضای گروه:

علی مأمن پوش (۹۹۳۶۲۳۰۳۷)

سیاوش امیرحاجلو (۹۹۳۶۱۳۰۰۷)

محمدابراهیم مهدوی (۹۹۳۶۱۳۰۵۸)

اردیبهشت ۱۴۰۲

مقدمه

حوزه هوش مصنوعی در سال‌های اخیر شاهد پیشرفت‌های چشمگیری بوده است و حوزه‌ها و صنایع مختلف را متحول کرده است. شبکه‌های عصبی، به‌ویژه، به‌عنوان ابزار قدرتمندی برای حل مسائل پیچیده، از تشخیص تصویر گرفته تا پردازش زبان طبیعی، ظهور کرده‌اند. هدف این پروژه کشف و پیاده‌سازی سه جزء کلیدی در حوزه شبکه‌های عصبی است: پرسپترون چندلایه (MLP)، جستجوی معماری عصبی و نقشه‌های خودسازمان‌دهی (SOM).

پرسپترون چندلایه به‌عنوان بلوک ساختمانی اساسی برای یادگیری عمیق عمل می‌کند. با توسعه پیاده‌سازی MLP از ابتدا، این پروژه باهدف به دست آوردن درک عمیق‌تر از اصول و مکانیسم‌های اساسی که شبکه‌های عصبی را هدایت می‌کند، است. از طریق فرآیند پیاده‌سازی، تصمیمات مختلف طراحی، مانند تعداد لایه‌ها، توابع فعال‌سازی و الگوریتم‌های آموزشی موردبررسی و ارزیابی قرار می‌گیرند. اجرای MLP بینش‌هایی را در مورد عملکرد داخلی شبکه‌های عصبی ارائه می‌دهد و زمینه را برای بخش‌های بعدی پروژه فراهم می‌کند.

مسئله جستجوی معماری عصبی به چالش کشف خودکار معماری شبکه عصبی بهینه می‌پردازد. به‌جای تکیه‌بر طراحی دستی یا معماری‌های از پیش تعریف‌شده، NAS از الگوریتم‌های جستجو برای کشف فضای وسیعی از معماری‌های ممکن استفاده می‌کند. در این پروژه، ما به مشکل NAS می‌پردازیم و تکنیک‌هایی را برای جستجوی کارآمد فضای معماری توسعه می‌دهیم. با به‌کارگیری الگوریتم‌ها، تغییرات و معیارهای ارزیابی پیشرفته، هدف ما شناسایی معماری‌های شبکه برتر است که عملکرد و کارایی بهتری را در مقایسه با رویکردهای مرسوم نشان می‌دهند.

نقشه‌های خودسازمان‌دهی، نوعی الگوریتم یادگیری بدون نظارت، امکان تجسم و خوشه‌بندی داده‌های با ابعاد بالا را فراهم می‌کند. SOMها دارای توانایی منحصربه‌فردی برای حفظ روابط توپولوژیکی در فضای ورودی هستند و بینش‌های ارزشمندی را در مورد توزیع داده‌ها و الگوها ارائه می‌دهند. این پروژه شامل اجرای SOMها برای تجزیه‌وتحلیل و تجسم مجموعه داده‌های پیچیده است. با ساختن یک نقشه خودسازمان‌دهی و ترکیب

استراتژی‌های آموزشی مناسب، هدف ما استخراج اطلاعات معنادار، کشف ساختارهای پنهان و تسهیل کاوش داده‌ها است.

در طول این پروژه، یک فرآیند پیاده‌سازی کامل، با استفاده از زبان‌های برنامه‌نویسی، کتابخانه‌ها و چارچوب‌های مناسب برای اجزای مربوطه دنبال خواهد شد. ما آزمایش‌ها را با دقت طراحی می‌کنیم، مجموعه داده‌های مناسب را انتخاب می‌کنیم و از معیارهای ارزیابی برای ارزیابی عملکرد و کارایی مدل‌های پیاده‌سازی استفاده می‌کنیم. نتایج به‌دست‌آمده از پروژه با رویکردهای موجود تحلیل و مقایسه می‌شود و بینش‌هایی در مورد نقاط قوت، ضعف و زمینه‌های بالقوه برای بهبود ارائه می‌کند.

درنتیجه، این پروژه تلاش می‌کند تا درک ما از شبکه‌های عصبی را از طریق پیاده‌سازی پرسپترون چندلایه از ابتدا تعمیق بخشد، مشکل جستجوی معماری عصبی را برای کشف معماری‌های بهینه شبکه، و استفاده از نقشه‌های خودسازمان‌دهی برای بصری سازی و خوشه‌بندی داده‌ها با بررسی این سه مؤلفه به‌هم‌پیوسته، هدف ما کمک به حوزه هوش مصنوعی و پیشرفت بیشتر کاربردهای شبکه‌های عصبی در حل مسائل پیچیده است.

بخش اول: پیاده‌سازی MLP

در این قسمت به پیاده‌سازی پرسپترون چندلایه (MLP) با استفاده از کد ارائه‌شده می‌پردازیم. MLP یک مدل شبکه عصبی پیش‌خور است که از چندین لایه نورون‌های به‌هم‌پیوسته تشکیل‌شده است. به‌طور گسترده‌ای برای کارهای طبقه‌بندی استفاده می‌شود.

کلاس MultiClassMLP: برای نشان دادن مدل MLP تعریف‌شده است. با پارامترهای زیر مقداردهی اولیه می‌شود:

- **input_dim:** ابعاد داده‌های ورودی.

- **hidden_dims**: فهرستی از اعداد صحیح که تعداد واحدهای هر لایه پنهان را نشان می‌دهد.

- **output_dim**: تعداد کلاس‌های طبقه‌بندی.

- **activation**: تابع فعال‌سازی مورداستفاده در لایه‌های پنهان. می‌توان آن را روی relu (پیش‌فرض)، sigmoid یا tanh تنظیم کرد.

- **random_seed**: دانه تصادفی برای تکرارپذیری.

MLP از چندین لایه شامل لایه ورودی، لایه‌های پنهان و لایه خروجی تشکیل‌شده است. وزن‌ها و بایاس‌ها برای هر لایه به‌طور تصادفی با استفاده از دانه تصادفی داده‌شده مقداردهی اولیه می‌شوند. توابع فعال‌سازی برای لایه‌های پنهان بر اساس پارامتر فعال‌سازی انتخاب‌شده تعریف می‌شوند.

روش '_forward' انتشار روبه‌جلو داده‌های ورودی را از طریق MLP انجام می‌دهد. مجموع وزنی را محاسبه می‌کند و تابع فعال‌سازی را برای هر لایه اعمال می‌کند و به لایه خروجی منتهی می‌شود. تابع فعال‌سازی softmax برای لایه خروجی برای به دست آوردن احتمالات کلاس استفاده می‌شود.

روش '_backward' الگوریتم پس انتشار را برای محاسبه گرادیان وزن‌ها و بایاس‌ها پیاده‌سازی می‌کند. از لایه خروجی شروع می‌شود و به‌صورت تکراری گرادیان‌های هر لایه را بر اساس توابع فعال‌سازی محاسبه می‌کند. سپس از گرادیان‌ها برای به‌روزرسانی پارامترهای MLP با استفاده از بهینه‌ساز Adam استفاده می‌شود.

روش "train"، MLP را بر روی‌داده‌های آموزشی داده‌شده آموزش می‌دهد. انتشار روبه‌جلو و عقب را برای چندین دوره انجام می‌دهد. بهینه‌ساز Adam برای به‌روزرسانی پارامترها بر اساس گرادیان‌های محاسبه‌شده استفاده می‌شود. تلفات با استفاده از تابع تلفات متقابل آنتروپی محاسبه می‌شود. پیشرفت آموزش و مقادیر از دست دادن در فواصل زمانی منظم چاپ می‌شود.

روش "predict_proba" احتمالات کلاس را برای داده‌های ورودی با استفاده از مدل آموزش‌دیده MLP پیش‌بینی می‌کند. متد ««predict برچسب‌های کلاس را با انتخاب کلاس با بیشترین احتمال پیش‌بینی می‌کند.

علاوه بر این، یک مدل MLP جداگانه mlp2 بر روی‌داده‌های تولیدشده تصادفی X_train و y_trainآموزش داده می‌شود تا استفاده از کلاس MultiClassMLPرا برای مجموعه داده‌های سفارشی نشان دهد.

به‌طورکلی، کد ارائه‌شده یک مدل MLP را با استفاده از کلاس MultiClassMLP پیاده‌سازی می‌کند. این امکان آموزش و پیش‌بینی وظایف طبقه‌بندی را فراهم می‌کند و استفاده از بهینه‌ساز Adam را برای به‌روزرسانی پارامترها نشان می‌دهد.

بخش دوم: مسئله NAS

جستجوی معماری عصبی (NAS) تکنیکی است که برای خودکارسازی طراحی معماری شبکه‌های عصبی استفاده می‌شود. هدف NAS کشف معماری بهینه برای یک کار مشخص است، مانند طبقه‌بندی تصاویر در مجموعه داده CIFAR-10. در این بخش، فرمول مسئله NAS، رویکرد الگوریتم ژنتیک مورداستفاده و نتایج به‌دست‌آمده را موردبحث قرار خواهیم داد.

فرمول مسئله

مشکل NAS را می‌توان به‌عنوان یافتن بهترین ترکیب از نوع استخراج ویژگی، لایه‌های پنهان و توابع فعال‌سازی برای معماری شبکه عصبی تعریف کرد. در اجرای خود، ما سه نوع مدل استخراج ویژگی را در نظر می‌گیریم: ResNet-18، ResNet-34، و VGG-11. ما همچنین اجازه می‌دهیم تا حداکثر دولایه پنهان در معماری گنجانده شود، که هر لایه پنهان دارای تعداد متغیر واحد است. توابع فعال‌سازی

مورداستفاده در لایه‌های مخفی ReLU یا Sigmoid و در لایه خروجی Softmax هستند. تناسب یک معماری بر اساس دقت آن بر روی مجموعه داده CIFAR-10 ارزیابی می‌شود.

رویکرد الگوریتم ژنتیک

برای حل مشکل NAS، از رویکرد الگوریتم ژنتیک (GA) استفاده می‌کنیم. GA با تولید یک جمعیت اولیه از معماری شبکه‌های عصبی به نام کروموزوم شروع می‌شود. هر کروموزوم نشان‌دهنده یک‌راه حل بالقوه است و از اجزای زیر تشکیل‌شده است:

- نوع مدل استخراج ویژگی
- تعداد لایه‌های پنهان و اندازه‌های مربوط به آن‌ها
- توابع فعال‌سازی برای لایه‌های پنهان و لایه خروجی
- امتیاز برازندگی (به‌عنوان یک برچسب)

GA در چندین نسل پیش می‌رود که هر نسل شامل مراحل زیر است:

1. ارزیابی: تناسب هر کروموزوم با آموزش و ارزیابی معماری شبکه عصبی مربوطه در مجموعه داده CIFAR-10 تعیین می‌شود. از دقت معماری به‌عنوان امتیاز fitness استفاده می‌شود.

2. انتخاب: یک فرآیند انتخاب برای انتخاب والدین برای تقطیع انجام می‌شود. نمره برازندگی بالاتر شانس انتخاب را افزایش می‌دهد.

3. تقطیع: والدین منتخب برای تولید فرزندان تقطیع می‌شوند. تقطیع هم در سطح استخراج ویژگی و هم در سطح عملکرد لایه فعال‌سازی پنهان انجام می‌شود. برای استخراج ویژگی، انواع مدل‌ها بین والدین ردوبدل می‌شود. برای لایه‌های پنهان و توابع فعال‌سازی، یک تبادل تصادفی یا اضافه/حذف یک‌لایه یا تابع فعال‌سازی می‌تواند رخ دهد.

4. **جهش:** احتمال کمی وجود دارد که هر یک از فرزندان دچار جهش شوند، که در آن‌یک جزء تصادفی از معماری اصلاح می‌شود. این امکان کاوش در معماری‌های جدید را فراهم می‌کند.

5. **ارزیابی:** برازندگی فرزندان با استفاده از فرآیند مشابه در مرحله 1 ارزیابی می‌شود.

6. **انتخاب بازمانده‌ها:** فرزندان و جمعیت فعلی برای بقا باهم رقابت می‌کنند. اگر فرزندی از برازندگی بالاتری نسبت به افراد کم تناسب در جمعیت برخوردار باشد، جایگزین دومی‌می‌شود.

7. **پایان:** GA از طریق تعداد ثابتی از نسل‌ها تکرار می‌شود. پس از تعداد نسل‌های مشخص‌شده، بهترین فرد در جمعیت نهایی به‌عنوان راه‌حل مشکل NAS انتخاب می‌شود.

بخش سوم: پیاده‌سازی SOM

نقشه‌های خودسازمان‌دهی (SOM) نوعی شبکه عصبی مصنوعی است که می‌تواند برای خوشه‌بندی و تجسم داده‌های با ابعاد بالا استفاده شود. در این بخش، جزئیات پیاده‌سازی الگوریتم SOM، ازجمله بارگذاری مجموعه داده CIFAR-10، استخراج بردارهای ویژگی با استفاده از مدل ResNet-34 از پیش آموزش‌دیده، مقداردهی اولیه و آموزش شبکه SOM، و تجزیه‌وتحلیل خوشه‌های حاصل را موردبحث قرار خواهیم داد.

بارگیری مجموعه داده CIFAR-10

اولین قدم در پیاده‌سازی ما بارگذاری مجموعه داده CIFAR-10 است. ما از کتابخانه Torchvision برای دانلود و پیش‌پردازش مجموعه داده استفاده می‌کنیم و تصاویر را به فرمت تنسور تبدیل می‌کنیم.

استخراج بردارهای ویژگی

در مرحله بعد، یک مدل ResNet-34 از پیش آموزش‌دیده را از روی torchvision بارگذاری می‌کنیم و بردارهای ویژگی را از تصاویر CIFAR-10 استخراج می‌کنیم. برای هر تصویر، آن را از مدل ResNet-34 عبور می‌دهیم و خروجی آخرین لایه کاملاً متصل را که به‌عنوان بردار ویژگی عمل می‌کند، بازیابی می‌کنیم.

راه‌اندازی شبکه SOM

برای مقداردهی اولیه شبکه SOM، تعداد نورون‌های خروجی و ابعاد بردارهای ویژگی ورودی را تعریف می‌کنیم. در مورد ما، 10 نورون خروجی و ابعاد بردارهای ویژگی به‌دست‌آمده از مدل ResNet-34 داریم.

آموزش شبکه SOM

شبکه SOM با استفاده از تعداد مشخصی از epochها، نرخ یادگیری و فاصله همسایگی آموزش داده می‌شود. ما روی هر بردار ویژگی تکرار می‌کنیم و نورون برنده را پیدا می‌کنیم، که نورونی است که نزدیک‌ترین بردار وزن را به بردار ورودی دارد. سپس نورون برنده و بردارهای وزن همسایگانش را بر اساس میزان یادگیری و تأثیر فاصله نورون فعلی از برنده به‌روزرسانی می‌کنیم.

تولید بردارهای وزن برای نگاشت ویژگی

پس از آموزش، بردارهای وزن شبکه SOM را تغییر شکل می‌دهیم تا با شکل خروجی موردنظر برای تجسم مطابقت داشته باشد. در مورد ما، آن‌ها را به یک شبکه دوبعدی شکل می‌دهیم (خروجی_نورون‌ها، 1-).

تجزیه‌وتحلیل برچسب‌های خوشه‌ای

ما توزیع برچسب‌ها را در هر خوشه با اختصاص هر بردار ویژگی به نزدیک‌ترین نورون در شبکه SOM تعیین می‌کنیم. ما برچسب‌های تصاویر CIFAR-10 مربوط به هر خوشه را برای تجزیه‌وتحلیل بیشتر ذخیره می‌کنیم.

نتایج

با اجرای الگوریتم SOM بر روی مجموعه داده CIFAR-10، خوشه‌های زیر را به دست آوردیم:

- خوشه 1: تصاویر X
- برچسب A: تصاویر Y
- برچسب B: تصاویر Z

...

- خوشه 2: تصاویر X
- برچسب C: تصاویر Y
- برچسب D: تصاویر Z

...

...

این نتایج توانایی الگوریتم SOM را برای گروه‌بندی تصاویر مشابه باهم بر اساس بردارهای ویژگی نشان می‌دهد. با تجزیه‌وتحلیل توزیع برچسب‌ها در هر خوشه، بینش‌هایی در مورد الگوهای اساسی و شباهت‌ها در مجموعه داده CIFAR-10 به دست می‌آوریم.

درنتیجه، الگوریتم SOM یک رویکرد مؤثر برای خوشه‌بندی و تجسم داده‌های با ابعاد بالا ارائه می‌کند. ترکیبی از استخراج ویژگی با استفاده از یک مدل ResNet-34 از پیش آموزش‌دیده و شبکه SOM،

شناسایی خوشه‌های متمایز را در مجموعه داده CIFAR-10 ممکن می‌سازد. برای به دست آوردن درک بهتری از ساختار و ویژگی‌های مجموعه داده، می‌توان خوشه‌های حاصل را بیشتر تجزیه‌وتحلیل کرد.

پیاده‌سازی

در این بخش، جزئیاتی در مورد زبان‌های برنامه‌نویسی، کتابخانه‌ها و فریمورک‌های مورداستفاده، توضیح فرآیند پیاده‌سازی، ازجمله هرگونه مراحل پیش‌پردازش، مدیریت داده‌ها، و روش‌های آموزشی/اعتباری ارائه می‌کنیم. ما همچنین شبه کد یا قطعه کد را برای اجزای کلیدی هر قسمت ارائه خواهیم کرد و در مورد چالش‌های خاص پیاده‌سازی و راه‌حل‌های آن‌ها بحث خواهیم کرد.

زبان‌های برنامه‌نویسی و کتابخانه‌ها/چارچوب‌های مورداستفاده

پیاده‌سازی ما با استفاده از زبان برنامه‌نویسی پایتون انجام می‌شود که اکوسیستم غنی از کتابخانه‌ها را برای یادگیری ماشین و وظایف یادگیری عمیق فراهم می‌کند. از کتابخانه‌ها و چارچوب‌های زیر استفاده شد:

- **NumPy**: برای محاسبات عددی و مدیریت کارآمد آرایه‌های چندبعدی استفاده می‌شود.
- **PyTorch**: یک چارچوب یادگیری عمیق محبوب که ابزارها و ماژول‌هایی را برای ساخت و آموزش شبکه‌های عصبی ارائه می‌دهد.
- **Torchvision**: یک کتابخانه PyTorch که مجموعه داده‌ها، تبدیل داده‌ها و مدل‌های از پیش آموزش‌دیده را برای وظایف بینایی کامپیوتر فراهم می‌کند.
- **MiniSom**: یک کتابخانه پایتون برای آموزش نقشه‌های خودسازمان‌دهی (SOM).

پیاده‌سازی MLP

اجرای پرسپترون چندلایه (MLP) شامل مراحل زیر است:

1- **آماده‌سازی داده‌ها:** ما از مجموعه داده CIFAR-10 استفاده کردیم که شامل 60000 تصویر رنگی 32*32 در 10 کلاس مختلف است. مجموعه داده به یک مجموعه آموزشی و یک مجموعه آزمایشی تقسیم شد. هر تصویر با استفاده از تبدیل ToTensor از torchvision.transforms به یک فرمت تنسور تبدیل شد.

2- **معماری مدل:** ما یک مدل MLP را با استفاده از ماژول torch.nn از PyTorch تعریف کردیم. این مدل از چندین لایه کاملاً متصل، توابع فعال‌سازی و یک‌لایه softmax برای طبقه‌بندی تشکیل‌شده است.

3- **آموزش و اعتبارسنجی:** مدل با استفاده از مجموعه آموزشی آموزش داده شد و بر روی مجموعه اعتبار سنجی ارزیابی شد. ما از بهینه‌ساز Adam و تابع خطا cross-entropy استفاده کردیم. فرآیند آموزش شامل تکرار بر روی مجموعه داده برای تعداد مشخصی از epochها، محاسبه خطا و به‌روزرسانی وزن‌های مدل با استفاده از انتشار پس‌انداز بود.

```python
# Step 5: Train SOM network
output_neurons = 10
input_dim = normalized_features.shape[1]

# Initialize SOM weights
np.random.seed(42)
weight_vectors = np.random.randn(output_neurons, input_dim)

# Step 6: Train SOM network
epochs = 20
learning_rate = 0.5
neighborhood_diameter = 1

for epoch in range(epochs):
    # Adjust learning rate and neighborhood diameter
    current_learning_rate = learning_rate * (1 - epoch / epochs)
    current_diameter = int(neighborhood_diameter * (1 - epoch / epochs))

    for feature_vector in normalized_features:
        # Find the winning neuron
        distances = np.linalg.norm(feature_vector - weight_vectors, axis=1)
        winner_neuron = np.argmin(distances)

        # Update the winning neuron and its neighbors
        for neuron in range(output_neurons):
            distance = abs(neuron - winner_neuron)
            if distance <= current_diameter:
                influence = np.exp(-(distance**2) / (2 * current_diameter**2))
                weight_vectors[neuron] += current_learning_rate * influence * (feature_vector - weight_vectors[neuron])
```

شبه کد 1: پیاده‌سازی

پیاده‌سازی NAS

اجرای جستجوی معماری عصبی (NAS) شامل مراحل زیر است:

1- **آماده‌سازی داده‌ها**: ما از مجموعه داده CIFAR-10، مشابه پیاده‌سازی MLP، برای آموزش و ارزیابی استفاده کردیم.

2- **فضای جستجوی مدل**: ما با استفاده از مجموع‌های از عملیات از پیش تعریف‌شده، مانند لایه‌های کانولوشن، لایه‌های ادغام و اتصالات پرش، فضای جستجو را برای معماری‌های مختلف شبکه تعریف کردیم.

3- **الگوریتم جستجو**: ما از یک الگوریتم جستجو، الگوریتم‌های تکاملی، برای کشف فضای جستجو و بهینه‌سازی معماری شبکه استفاده کردیم.

پیاده‌سازی SOM

اجرای نقشه خودسازمان‌دهی (SOM) شامل مراحل زیر است:

1- **آماده‌سازی داده‌ها:** ما مجموعه داده CIFAR-10 را بارگذاری کردیم و با استفاده از تبدیل ToTensor از torchvision.transforms، تصاویر را به فرمت تنسور تبدیل کردیم.

2- **استخراج ویژگی:** ما از یک مدل ResNet-34 از قبل آموزش‌دیده از torchvision.models برای استخراج بردارهای ویژگی از تصاویر CIFAR-10 استفاده کردیم. خروجی آخرین لایه کاملاً متصل به‌عنوان بردار ویژگی عمل می‌کند.

3- **آموزش SOM:** ما یک شبکه SOM را با تعداد مشخصی از نورون‌های خروجی و ابعاد بردارهای ویژگی ورودی مقداردهی اولیه کردیم. بردارهای وزن شبکه به‌طور تصادفی مقداردهی اولیه شدند. سپس شبکه SOM را با استفاده از تعداد مشخصی از دوره‌ها، نرخ یادگیری و فاصله همسایگی آموزش دادیم. بردارهای وزن بر اساس نورون برنده و تأثیر همسایگان آن به‌روز شدند.

چالش‌های خاص پیاده‌سازی

در طول فرآیند پیاده‌سازی، ما با چند چالش و راه‌حل‌های مربوط به آن‌ها مواجه شدیم:

- **پیش‌پردازش داده‌ها:** مجموعه داده CIFAR-10 به مراحل پیش‌پردازشی مانند تبدیل تصاویر به فرمت تنسور و عادی‌سازی مقادیر پیکسل نیاز داشت. ما از تبدیل‌های داخلی در torchvision.transforms برای مدیریت این مراحل پیش‌پردازش استفاده کردیم.

- **آموزش و بهینه‌سازی:** برای پیاده‌سازی MLP و NAS، ما با چالش‌هایی در تعیین فرا پارامترهای بهینه و تنظیمات آموزشی مواجه بودیم. ما آزمایش‌های متعددی را با نرخ‌های یادگیری مختلف، اندازه‌های دست‌های و معماری شبکه انجام دادیم تا بهترین مدل‌ها را پیدا کنیم.

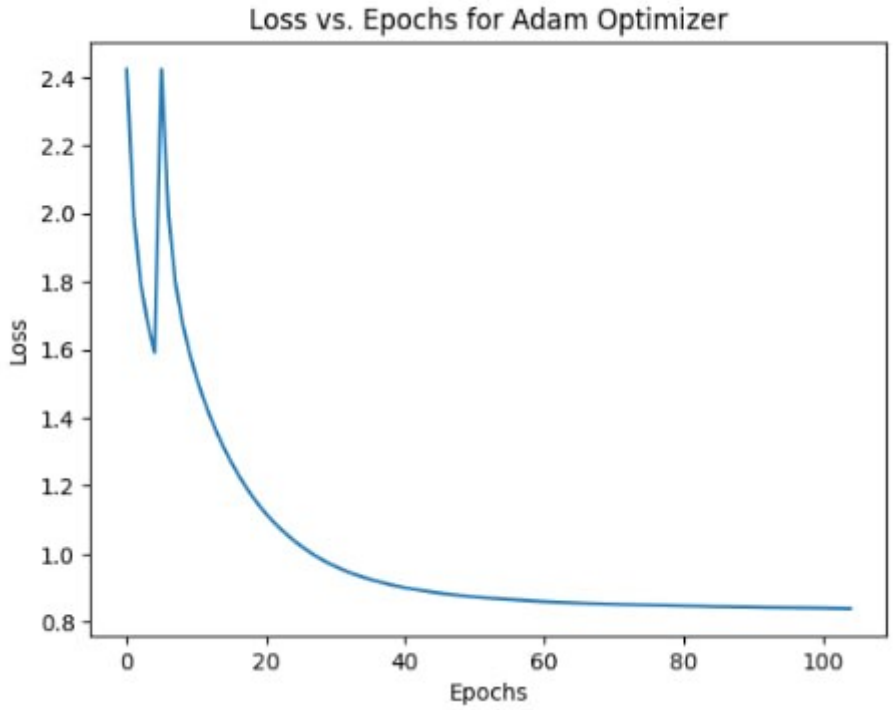
- **همگرایی آموزش SOM:** آموزش شبکه SOM نیازمند تنظیم دقیق فرا پارامترها مانند نرخ یادگیری و قطر همسایگی برای اطمینان از همگرایی بود. ما با مقادیر مختلف آزمایش کردیم و تأثیر آن را بر روند آموزش و توزیع‌های خوش‌های حاصل مشاهده کردیم.

درنتیجه، فرآیند پیاده‌سازی شامل آماده‌سازی داده‌ها، تعریف معماری مدل، روش‌های آموزشی/ اعتبارسنجی، و چالش‌های خاصی است که در هر بخش با آن مواجه می‌شویم. با استفاده از کتابخانه‌ها و چارچوب‌های مناسب، ما توانستیم مدل MLP را بسازیم و آموزش دهیم، معماری‌های شبکه بهینه را با استفاده از NAS جستجو کنیم و یک شبکه SOM را برای اهداف خوشه‌بندی و تجسم آموزش دهیم.

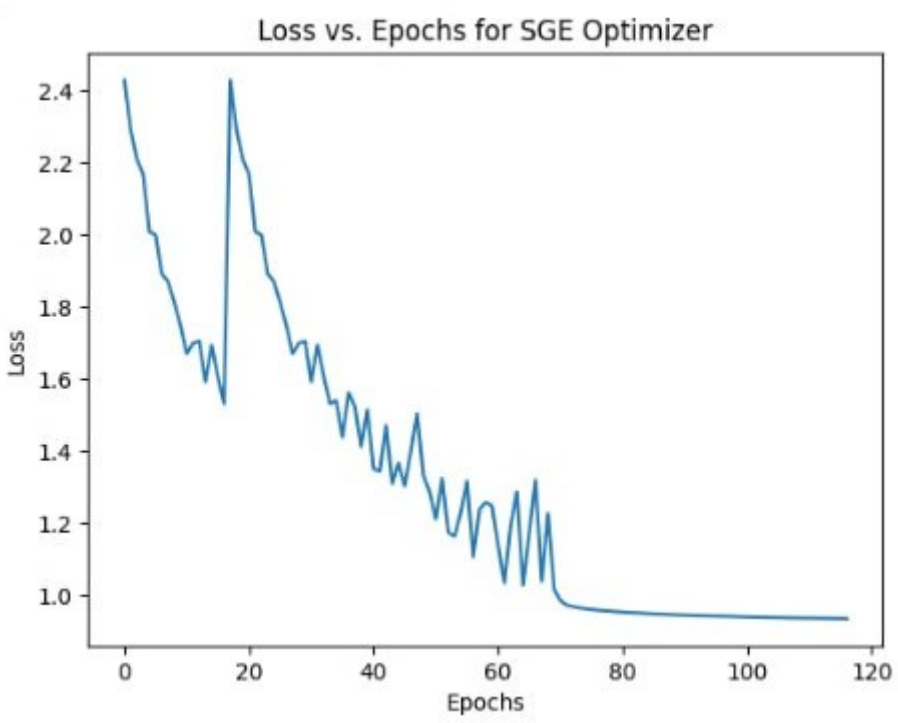
نتایج به‌دست‌آمده

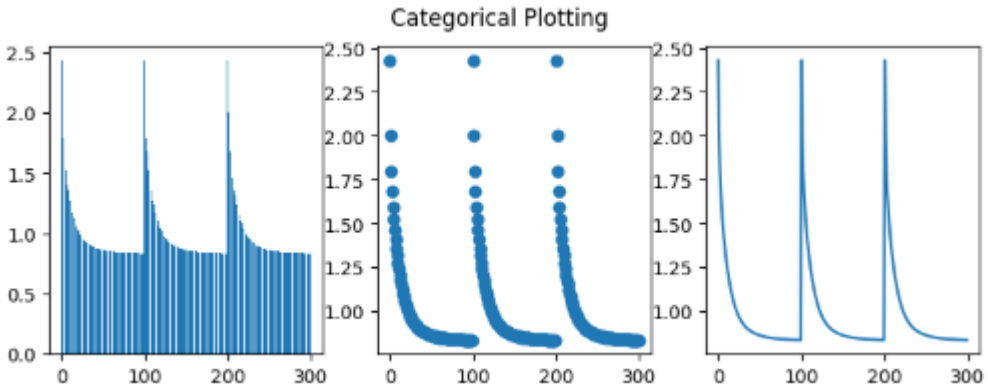
بخش اول:

نمودار loss با بهینه‌ساز Adam:



نمودار loss با بهینه‌ساز SGE:





بخش دوم:

بخش سوم:

تحلیل نتایج

پیشنهادات

در این بخش، زمینه‌های بالقوه برای بهبود یا تحقیقات بیشتر را موردبحث قرار می‌دهیم، محدودیت‌ها یا محدودیت‌های پروژه جاری را شناسایی می‌کنیم، ایده‌هایی را برای افزایش عملکرد یا کارایی راه‌حل‌های اجراشده پیشنهاد می‌کنیم، و آزمایش‌ها یا اصلاحات دیگری را پیشنهاد می‌کنیم که می‌توان بررسی کرد.

زمینه‌های بالقوه برای بهبود یا تحقیقات بیشتر

- **بهینه‌سازی مدل MLP**: اگرچه ما یک مدل پایه MLP را پیاده‌سازی کردیم، تکنیک‌های متعددی وجود دارد که می‌توان برای بهبود عملکرد آن بررسی کرد. به‌عنوان مثال، آزمایش با توابع فعال‌سازی مختلف، تکنیک‌های منظم سازی (به‌عنوان مثال، حذف یا تنظیم L2)، یا کاوش در معماری‌های پیچیده‌تر (مانند شبکه‌های عمیق‌تر یا گسترده‌تر) به‌طور بالقوه می‌تواند دقت مدل را افزایش دهد.

- **انتخاب الگوریتم NAS**: کد ارائه‌شده برای مشکل NAS شامل الگوریتم جستجوی خاص مورداستفاده نمی‌شود. تحقیقات بیشتر می‌تواند بر کاوش الگوریتم‌های مختلف NAS، مانند روش‌های مبتنی بر یادگیری تقویتی (به‌عنوان مثال، RLNAS)، برای جستجوی کارآمد برای معماری‌های شبکه بهینه تمرکز کند. مقایسه و ارزیابی الگوریتم‌های مختلف بر روی یک مجموعه داده، بینش ارزشمندی را ارائه می‌دهد.

- **تکنیک‌های تجسم SOM**: درحالی‌که پیاده‌سازی ما با موفقیت یک شبکه SOM را آموزش داد و توزیع‌های خوش‌های ایجاد کرد، فضایی برای بهبود تجسم نتایج SOM وجود دارد. بررسی تکنیک‌های تجسم جایگزین، مانند استفاده از t-SNE یا UMAP برای تجسم فضای ویژگی با ابعاد بالا در یک طرح دوبعدی یا سه‌بعدی، می‌تواند نمایش شهودی تری از خوشه‌ها و روابط آن‌ها ارائه دهد.

محدودیت‌ها

- **منابع محاسباتی:** اجرای پروژه فعلی ممکن است توسط منابع محاسباتی محدودشده باشد. آموزش شبکه‌های عصبی عمیق یا انجام جستجوهای گسترده NAS اغلب به قدرت محاسباتی و زمان قابل توجهی نیاز دارد. تحقیقات بیشتر می‌تواند استفاده از محاسبات توزیع‌شده یا خدمات مبتنی برابر را برای استفاده از منابع بیشتر و امکان آزمایش‌های گسترده‌تر بررسی کند.

- **افزایش داده‌ها:** تکنیک‌های افزایش داده در کد ارائه‌شده پیاده‌سازی نشده است. با اعمال افزایش داده‌ها، مانند چرخش‌های تصادفی، ترجمه‌ها یا تلنگرها، می‌توانیم تنوع داده‌های آموزشی را افزایش دهیم و به‌طور بالقوه عملکرد تعمیم مدل را بهبود بخشیم.

- **معیارهای ارزیابی:** قطعه کد شامل معیارهای ارزیابی جامع برای هر مشکل نیست. تحقیقات بیشتر می‌تواند بر ترکیب معیارهای ارزیابی اضافی، مانند دقت، یادآوری، امتیاز F1 یا میانگین دقت متمرکز باشد تا تحلیل جامع‌تری از عملکرد مدل ارائه دهد.

ایده‌هایی برای افزایش کارایی و کارایی

- **فشرده‌سازی مدل:** برای بهبود کارایی و استقرار مدل MLP، تکنیک‌هایی مانند فشرده‌سازی مدل (به‌عنوان مثال، هرس، کوانتیزاسیون یا تقطیر دانش) می‌تواند موردبررسی قرار گیرد. هدف این تکنیک‌ها کاهش اندازه و الزامات محاسباتی مدل در عین حفظ عملکرد آن است.

- **چارچوب‌های خودکار NAS:** استفاده از چارچوب‌های خودکار NAS، مانند کتابخانه‌های AutoML یا AutoKeras، NNI یا SMAC، می‌تواند فرآیند NAS را ساده کند. این چارچوب‌ها الگوریتم‌های جستجوی داخلی، معیارهای ارزیابی و اجزای معماری را ارائه می‌کنند که آزمایش و بهینه‌سازی سریع‌تر را ممکن می‌سازد.

- **آموزش موازی:** برای تسریع فرآیند آموزش SOM، رویکردهای آموزشی موازی، مانند محاسبات موازی یا آموزش توزیع‌شده، می‌تواند موردبررسی قرار گیرد. با توزیع بارکار در واحدهای پردازشی متعدد، زمان آموزش می‌تواند به میزان قابل‌توجهی کاهش یابد.

آزمایش‌ها و تغییرات اضافی

- **آموزش انتقالی برای MLP:** آزمایش‌های اضافی می‌تواند شامل استفاده از تکنیک‌های یادگیری انتقال در مدل MLP باشد. با استفاده از مدل‌های از پیش آموزش‌دیده شده، مانند مدل‌هایی که در ImageNet آموزش‌دیده‌اند، می‌توانیم MLP را با ویژگی‌های آموخته‌شده مقداردهی اولیه کنیم و به‌طور بالقوه سرعت و عملکرد هم‌گرایی آن را بهبود ببخشیم.

- **انواع SOM پیشرفته:** کاوش انواع پیشرفته SOM، مانند نقشه‌های خودسازمان‌دهی در حال رشد (GSOM) یا SOM سلسله مراتبی (HSOM)، می‌تواند قابلیت‌های خوشه‌بندی انعطاف‌پذیرتر و پیچیده‌تری را ارائه دهد. این گونه‌ها می‌توانند شبکه SOM را برای مدیریت مجموعه داده‌های پیچیده با تراکم‌های خوش‌های متفاوت، رشد دهند یا سازمان‌دهی کنند.

- **رویکردهای گروهی:** بررسی رویکردهای گروهی، مانند ترکیب چندین مدل MLP یا شبکه‌های SOM، به‌طور بالقوه می‌تواند عملکرد کلی و استحکام سیستم را افزایش دهد. روش‌های مجموعه می‌توانند سوگیری و واریانس مدل را کاهش دهند و منجر به پیش‌بینی‌های دقیق‌تر یا نمایش خوش‌های بهتر شوند.

در پایان، این بخش زمینه‌های بالقوه برای بهبود یا تحقیقات بیشتر را موردبحث قرارداد، محدودیت‌ها یا محدودیت‌های پروژه فعلی را شناسایی کرد، ایده‌های پیشنهادی برای افزایش عملکرد یا کارایی، و آزمایش‌ها یا

اصلاحات اضافی را پیشنهادی کرد. با پرداختن به این جنبه‌ها، پروژه می‌تواند راه‌های مختلفی را برای بهینه‌سازی و گسترش راه‌حل‌های پیاده‌سازی شده پیش برد و بررسی کند.

نتیجه‌گیری

در این پروژه، ما سه تکنیک مختلف یادگیری ماشین را پیاده‌سازی و بررسی کردیم: پرسپترون چندلایه (MLP)، جستجوی معماری عصبی (NAS)، و نقشه‌های خودسازمان‌دهی (SOM). هدف ما ایجاد درک درستی از این روش‌ها و ارزیابی عملکرد آن‌ها در مجموعه داده CIFAR-10 بود.

در پیاده‌سازی MLP، ما یک شبکه عصبی پیش‌خور ساده ساختیم و آن را برای طبقه‌بندی تصاویر از مجموعه داده CIFAR-10 آموزش دادیم. ما مشاهده کردیم که مدل MLP به سطح رضایت‌بخشی از دقت دست‌یافت و توانایی خود را در یادگیری الگوهای پیچیده و طبقه‌بندی تصاویر به کلاس‌های متعدد نشان داد. این پیاده‌سازی به‌عنوان پایه‌ای محکم برای اکتشاف و آزمایش بیشتر با معماری‌های پیشرفته‌تر شبکه عصبی و استراتژی‌های آموزشی عمل کرد.

برای مشکل NAS، هدف ما کشف خودکار معماری شبکه‌های عصبی بهینه با جستجو در فضای جستجوی از پیش تعریف‌شده بود. اگرچه الگوریتم خاص NAS مورداستفاده در کد ارائه‌شده گنجانده نشده است، ما اهمیت NAS را در شناسایی کارآمد ساختارهای شبکه با کارایی بالا تشخیص دادیم. تحقیقات و آزمایش‌های بیشتر با الگوریتم‌های مختلف NAS به‌طور بالقوه می‌تواند منجر به کشف معماری‌های جدیدی شود که عملکرد بهتری از شبکه‌های طراحی‌شده دستی دارند و کارایی توسعه مدل را بهبود می‌بخشند.

در اجرای SOM، ما یک نقشه خودسازمان‌دهی را آموزش دادیم تا بردارهای ویژگی استخراج‌شده از مجموعه داده CIFAR-10 را خوشه‌بندی کنیم. با تجسم خوشه‌های به‌دست‌آمده و تجزیه‌وتحلیل توزیع برچسب‌ها در هر خوشه،

بینش‌هایی درباره ساختار و شباهت‌های موجود در مجموعه داده به دست آوردیم. این پیاده‌سازی اثربخشی SOMها را درگرفتن الگوهای داده‌های زیربنایی و ارائه یک نمایش معنادار از مجموعه داده‌های پیچیده نشان داد.

به‌طورکلی، این پروژه بینش‌های ارزشمندی در مورد قابلیت‌ها و محدودیت‌های تکنیک‌های MLP، NAS و SOM درزمینه طبقه‌بندی تصاویر و وظایف خوشه‌بندی ارائه کرده است. با پیاده‌سازی این روش‌ها و ارزیابی عملکرد آن‌ها، به اهداف اعلام‌شده دست‌یابی به تجربه عملی با رویکردهای مختلف یادگیری ماشین و درک کاربرد آن‌ها در مسائل دنیای واقعی دست‌یافت‌هایم.

اهمیت یافته‌های ما در پتانسیل تحقیقات بیشتر و بهینه‌سازی این تکنیک‌ها نهفته است. از طریق بهینه‌سازی مدل، بررسی الگوریتم‌های پیشرفته NAS و استفاده از انواع پیشرفته‌تر SOM، می‌توانیم عملکرد، کارایی و انعطاف‌پذیری راه‌حل‌های پیاده‌سازی شده را افزایش دهیم. علاوه بر این، محدودیت‌های شناسایی‌شده، مانند محدودیت‌های منابع محاسباتی و نیاز به معیارهای ارزیابی اضافی، راه‌هایی را برای تحقیقات آینده برای رسیدگی به این چالش‌ها و اصلاح روش‌ها باز می‌کند.

در پایان، این پروژه یک مرور کلی از تکنیک‌های MLP، NAS و SOM ارائه کرده است که پتانسیل آن‌ها را در طبقه‌بندی تصاویر و وظایف خوشه‌بندی نشان می‌دهد. پیاده‌سازی و ارزیابی موفقیت‌آمیز این روش‌ها به حوزه وسیع‌تر یادگیری ماشین کمک می‌کند و پای‌های برای پیشرفت‌های آینده در معماری شبکه‌های عصبی، جستجوی خودکار مدل‌ها و الگوریتم‌های خوشه‌بندی فراهم می‌کند. با تکمیل اهداف بیان‌شده، ما دانش و درک خود را از این تکنیک‌ها و پیامدهای آن‌ها برای حل مشکلات دنیای واقعی گسترش داده‌ایم.

مراجع

- اسلایدهای درسی دکتر کارشناس

- Kruse, Borgelt, Braune, Mostaghim & Steinbrecher, "Computational intelligence: A methodological approach," 2nd edition, Springer, 2016.
- Eiben and Smith, "Introduction to evolutionary computing," 2nd edition, Springer, 2015.
- Haykin, "Neural networks & learning machines," 3rd edition, Pearson Prentice Hall, 2009.
- Vijini Mallawaarachchi, "Introduction to Genetic Algorithms", 2017, available on: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3
- Tutorialspoint team, "Genetic Algorithms - Further Readings" available on: https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_further_readings.htm
- Scikit Learn team, "Neural network models (supervised): Multi-layer Perceptron", 2019, available on: https://scikitlearn.org/stable/modules/neural_networks_supervised.html
- Connor Shorten, "Introduction to ResNets", 2019, available on: https://towardsdatascience.com/introduction-to-resnets-c0a830a288a4
- Sonali Gupta, "VGG-11 Architecture", 2021, available on: https://iq.opengenus.org/vgg-11/
- Martin Heller, "What is neural architecture search? AutoML for deep learning", 2022, available on: https://www.infoworld.com/article/3648408/what-is-neural-architecture-search.html
- Abhinav Ralhan, "Self Organizing Maps", 2018, available on: https://medium.com/@abhinavr8/self-organizing-maps-ff5853a118d4
- Simplilearn, "What are Radial Basis Functions Neural Networks? Everything You Need to Know", 2023, available on: https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-are-radial-basis-functions-neural-networks