بهینه سازی شبکه عصبی عمیق

دانشکده علوم کامپیوتر

چکیده

شبکه‌های عصبی عمیق قدرت زیادی در پیدا کردن و یادگیری از داده با استفاده از انواع توابع را نشان داده‌اند. معماری‌های عمیق پیشرفته مانند ،RNN CNN و.. عمدتاً توسط الگوریتم بهینه سازی گرادیان کاهشی بهینه می‌شوند که با در نظر گرفتن گرادیان‌های گذشته و حال حاضر، وزن‌ها را به‌روز می‌کند. با این وجود، الگوریتم گرادیان کاهشی از مشکل فراجهش رنج می‌برد که باعث کاهش همگرایی آموزش شبکه می‌شود. با الهام از کنترل‌کننده پی‌آی‌دی (PID) با کنترل خودکار، رویکرد PID را برای تسریع بهینه‌سازی شبکه‌های عمیق پیشنهاد می‌کنیم. ابتدا درمورد کنترل‌کننده پی‌آی‌دی توضیح میدهیم، سپس الگوریتم بهینه‌سازی را با استفاده از تشبیه کردن پارامتر های شبکه عصبی به حرکت جریان الکتریکی در یک مدار را ارائه می‌دهیم. روش PID پیشنهادی پدیده فراجهش در الگوریتم گرادیان کاهشی را به میزان قابل توجهی کاهش می‌دهد. ین الگوریتم به‌طور پویا نرخ یادگیری را تنظیم می‌کند و پارامترهای مدل را در طول فرآیند آموزش تغییر می‌دهد، به گونه‌ای که جریان الکتریکی را برای دستیابی به همگرایی بهینه تقلید می‌کند. ارزیابی‌های تجربی نشان می‌دهد که این روش می‌تواند سرعت همگرایی و عملکرد کلی شبکه‌های عصبی عمیق را بهبود بخشد. این امر پتانسیل ادغام بینش‌های مهندسی برق را در حوزه یادگیری ماشین نشان می‌دهد.

۱) مقدمه

با دسترسی به مجموعه‌های داده‌های تصویری بزرگ مانند ImageNet [۱]، شبکه‌های عصبی عمیق ،(DNN) به ویژه شبکه‌های عصبی پیچشی عمیق ،(CNN) دقت سیستم را در بسیاری از مسائل بینایی کامپیوتر، مانند طبقه‌بندی تصویر [۲]، تشخیص اشیاء [۳] و تشخیص چهره [۴] به طور قابل توجهی بهبود بخشیده‌اند. با وجود موفقیت‌های بزرگ یادگیری عمیق، آموزش شبکه‌های عمیق روی مجموعه‌های داده‌های بزرگ معمولاً از نظر محاسباتی پرهزینه است و با استفاده از رایانه‌های شخصی رده بالا مجهز به GPU چندین روز یا حتی هفته طول می‌کشد. بررسی چگونگی افزایش سرعت آموزش مدل‌های عمیق بدون کاهش دقت، امری بسیار مهم است و می‌تواند باعث صرفه‌جویی در زمان و هزینه حافظه، به ویژه برای کاربردهای محدود از نظر منابع شود.

مؤلفه کلیدی در آموزش شبکه های عصبی عمیق نحوه بهینه سازی است که نحوه به‌روزرسانی میلیون‌ها یا حتی میلیاردها پارامتر یک مدل عمیق را تعریف می‌کند. با توجه به نحوه تنظیم نرخ یادگیری، بهینه‌سازهای یادگیری عمیق را می‌توان به دو گروه تقسیم کرد: بهینه‌سازهای نرخ یادگیری تنظیم‌شده دستی مانند کاهش گرادیان تصادفی (SGD) [۶]، Momentum SGD [۷] و Momentum Nesterov's [۷] و بهینه‌سازهای نرخ یادگیری خودکار مانند AdaGrad [۸]، RMSProp [۹] و Adam [۱۰] و غیره. بهینه‌سازهای نرخ یادگیری خودکار به شکل پویایی نرخ یادگیری را برای هر پارامتر تنظیم می‌کنند. چنین هدفی از تغییر پارامتر ها حرکت جذابی است که با استفاده از آنها نتایج یادگیری مدل عمیق شرایت بهتر (بهینه) ای را به همراه داشته باشد.

استراتژی SGD–Momentum به این شکل است که برای به‌روزرسانی پارامترهای شبکه، هم گرادیان‌های گذشته و هم گرادیان‌های حال را در نظر میگیرد. با این حال، SGD–Momentum از مشکل فراهش رنج می‌برد که به پدیده‌هایی اشاره می‌کند که مقدار وزنه از مقدار هدف خود بسیار فراتر رفته و جهت به‌روزرسانی خود را تغییر نمی‌دهد. چنین مشکلی همگرایی SGD–Momentum را مختل می‌کند و زمان و منابع آموزشی بیشتری را هدر می‌دهد. بررسی اینکه آیا می‌توانیم یک بهینه‌ساز DNN جدید طراحی کنیم که بدون مشکل overshoot باشد و در عین حال سرعت همگرایی سریع‌تری داشته باشد و دقت خوبی را حفظ کند، از اهمیت زیادی برخوردار است.

جریان الکتریکی یکی از مفاهیم بنیادی در الکتریسیته می باشد که با استفاده از روش های کنترل و فیدبک تغیرات در پارامتر های بسیاری از مدار ها را نظارت و تنظیم میکند. به عنوان مثال، می‌توان از کنترل جریان الکتریکی برای کنترل دمای یک قطعه الکتریکی، تنظیم سرعت یک موتور الکتریکی یا حفظ ولتاژ یک منبع تغذیه استفاده کرد. روش های کنترل و فیدبک با استفاده از اطلاعات مربوط به خروجی سیستم، مقدار خروجی مطلوب را محاسبه و سپس اقدام لازم را برای حفظ خروجی سیستم در مقدار مطلوب انجام می دهند. در کنترل جریان الکتریکی، اطلاعات مربوط به خروجی سیستم معمولاً شامل جریان فعلی مدار است. با استفاده از این اطلاعات، کنترل‌کننده جریان الکتریکی می‌تواند مقدار جریان مطلوب را محاسبه و سپس اقدام لازم را برای حفظ جریان مدار در مقدار مطلوب انجام دهد. ایده اصلی کنترل PID این است که عمل کنترل باید متناسب با خطای جاری (اختلاف بین خروجی سیستم و خروجی مطلوب)، انتگرال خطای گذشته در طول زمان و مشتق خطا، که نشان‌دهنده روند آینده است، باشد. برای کنترل جریان الکتریکی، می‌توانیم از کنترل‌کننده‌های PID استفاده کنیم. در این حالت، خطای جاری به عنوان ورودی برای کنترل‌کننده PID استفاده می‌شود، سپس کنترل‌کننده PID یک سیگنال کنترل تولید می‌کند که این سیگنال کنترل به مدار اعمال می‌شود و باعث تغییر جریان مدار می‌شود.

شبکه‌های عصبی الگوریتم‌های یادگیری ماشینی هستند که از ساختار شبکه عصبی انسان الهام گرفته شده‌اند. این شبکه ها از مجموعه‌ای از گره‌ها یا نورون‌ها تشکیل شده‌اند که به یکدیگر متصل هستند. هر نورون دارای یک وزن است که میزان تأثیر آن بر نورون‌های دیگر را تعیین می‌کند. در هنگام آموزش شبکه عصبی، وزن‌های نورون‌ها به گونه‌ای تنظیم می‌شوند که شبکه بتواند عملکرد مورد نظر را انجام دهد. این کار معمولاً با استفاده از روش‌های یادگیری ماشینی مانند یادگیری نظارت‌شده انجام می‌شود. زن‌های شبکه عصبی نقش مهمی در عملکرد شبکه ایفا می‌کنند. وزن‌های مناسب می‌توانند باعث شوند که شبکه عملکرد مورد نظر را با دقت بالایی انجام دهد. وزن‌های نامناسب می‌توانند باعث شوند که شبکه عملکرد نامناسبی داشته باشد یا حتی اصلاً کار نکند. پتانسیومتر یک قطعه الکترونیکی است که می‌تواند برای تنظیم مقاومت استفاده شود. پتانسیومتر از یک سیم مقاومتی تشکیل شده است که به دو سر آن یک اتصال و به یک سر دیگر آن یک اتصال متحرک متصل شده است. با حرکت اتصال متحرک، مقدار مقاومت تغییر می‌کند. می‌توان از پتانسیومتر برای تبدیل وزن‌های شبکه عصبی به مقاومت استفاده کرد. برای این کار، وزن‌های شبکه را

به مقادیر مقاومتی تبدیل می‌کنیم. سپس، پتانسیومتر را به گونه‌ای تنظیم می‌کنیم که مقاومت آن برابر با وزن‌های تبدیل‌شده باشد. در سیستم کنترل مبتنی بر مقاومت پتانسیومتر، جریان مدار ثابت فرض می‌شود که برای کنترل جریان مدار، میتوان میزان مقاومت اجزای مدار را تغییر دهیم استفاده میکنیم.

۲) کارهای مرتبط

نرخ یادگیری مهمترین ابرپارامتر برای آموزش شبکه‌های عصبی عمیق است [۹]. با توجه به نحوه تنظیم نرخ یادگیری، دو دسته از روش‌های بهینه‌سازی یادگیری عمیق قابل طبقه‌بندی هستند. دسته اول روش‌های نرخ یادگیری ثابت مانند SGD [۶]، Momentum SGD [۷] و Momentum Nesterov's [۷] و غیره را نشان می‌دهد و دسته دوم شامل روش‌های نرخ یادگیری خودکار، مانند AdaGrad [۸]، RMSProp [۹] و Adam [۱۰] و غیره است. کار ما بر اساس روش‌های نرخ یادگیری ثابت است، با توجه به اینکه نتایج برتر فعلی روی مجموعه‌های داده CIFAR۱۰، CIFAR۱۰۰، ImageNet، VOC PASCAL و COCO MS عمدتاً توسط شبکه‌های عصبی باقی مانده که با استفاده از Momentum SGD آموزش دیده شده‌اند، به دست آمده است.

کاهش گرادیان تصادفی (SGD) یک الگوریتم بهینه‌سازی برای یادگیری ماشین و در حالت خصوصا برای یادگیری عمیق استفاده می‌شود. SGD معمولا از یک نرخ یادگیری ثابت استفاده می‌کند. این به این دلیل است که تخمین‌گر گرادیان SGD یک منبع نویز (نمونه‌گیری تصادفی m داده یادگیری) را معرفی می‌کند و این نویز حتی زمانی که ضرر به حداقل می‌رسد، از بین نمی‌رود.

Momentum SGD برای تسریع یادگیری، به ویژه در مورد گرادیان‌های کوچک و ثابت طراحی شده است. الگوریتم تندی یک میانگین از کاهش گرادیان‌های گذشته را جمع می‌کند و به حرکت در جهت ثابت ادامه می‌دهد. نام تندی اقتباس شده از پدیده فیزیکی آن گرفته شده است که در شبکه عصبی گرادیان منفی نیرویی می باشد که یک ذره در فضای پارامتر های ما حرکت می‌دهد. یک ابرپارامتر $\alpha \in (0, 1)$تعیین می‌کند که چه مقدار از گرادیان‌های گذشته به تغییر وزن های فعلی کمک کنند.

Momentum Nesterov's یک نوع از الگوریتم تندی است که توسط روش گرادیان شتاب‌دهنده نستروف ایجاد شده است. تفاوت بین تندی نستروو و تندی معمولی در جایی است که گرادیان ارزیابی می‌شود. با تندی نستروف، گرادیان پس از اعمال سرعت فعلی تخمین زده می‌شود. بنابراین می‌توان تندی نستروف را به عنوان تلاش برای اضافه کردن یک عامل اصلاح به روش استاندارد تندی تفسیر کرد. اخیراً، روش تندی نستروف به عنوان یک معادله دیفرانسیل معمولی مرتبه دوم در گام های کوچک توصیف شده است .

۳) رابطه عمومی با GDS

کنترل‌کننده PID از اطلاعات حال، گذشته و آینده برای پیشبینی خطا در کنترل یک سیستم فیدبک استفاده می‌کند . کنترل‌کننده مبتنی بر PID از قرن نوزدهم برای کنترل سرعت استفاده شده است. پایه نظری برای عملکرد PID برای اولین بار توسط ماکسول در سال ۱۸۶۸ در مقاله اصلی خود با عنوان "On Governors" ارائه شد. سپس Minorsky به آن فرمول‌بندی ریاضی داد. در طول سال‌ها، الگوریتم‌های کنترل زیادی پیشنهاد شده است. با این حال، اکثر کنترلرهای صنعتی با این الگوریتم اجرا می‌شوند، زیرا این الگوریتم به نسبت بقیه الگوریتم ها ساده، قوی و راحت برای استفاده می باشد . یک کنترل‌کننده PID به طور پیوسته خطا $e(t)$ را محاسبه می‌کند که این مقدار اختلاف بین خروجی مطلوب مورد نظر و خروجی اندازه‌گیری شده سیستم است (که تشبیه آن در شبکه های عصبی انجام خطایابی برای پیدا کردن پارامتر های شبکه می باشد). با استفاده از محاسبه خطا، یک اصلاح u(t) را به سیستم بر اساس تناسب ،(P) انتگرال (I) و مشتق

(D) بر حسب e(t) را اعمال می‌کند. این معادله را میتوان به شکل زیر نوشت:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{d}{dt} e(t),$$

که در اینجا k_i، k_p و k_d به ترتیب ضرایب تقویت کننده i، p و d می باشند.

می‌توان دید که خطای $e(t)$، که به عنوان اختلاف بین مقدار مطلوب و خروجی واقعی تعریف می‌شود، همان نقش گرادیان را در بهینه‌سازی یادگیری عمیق ایفا می‌کند. سه ضرایب دیگر سهم خطاهای حال، گذشته و آینده را تعیین می‌کنند. چنین تحلیل‌هایی ما را به استفاده از تکنیک‌های کنترل PID در زمینه بهینه‌سازی شبکه‌های عمیق ترغیب می‌کند. همانطور که در ادامه این مقاله خواهیم دید، بهینه‌ساز پیشنهادی، مزایای خارق‌العاده کنترل‌کننده PID را به ارث می‌برد و در عین حال ساده و کارآمد باقی می‌ماند.

در این بخش، ارتباط بین کنترل PID و بهینه‌سازی عمیق مبتنی بر SGD را آشکار می‌کنیم. این ارتباطها ما را به ارائه یک روش بهینه‌سازی جدید برای تسریع آموزش هاDNN سوق می‌دهد. همینطور درمورد شیوه الهام جریان الکتریکی برای به ەروزرسانی وزن‌ها در یک شبکه عمیق با استفاده از کنترل PID توضیح میدهیم که با این شیوه تعادل یک مدار/شبکه عصبی را کنترل میکنیم. در سیستم‌های الکتریکی، کنترل‌کننده‌های PID برای حفظ جریان و ولتاژ در یک مقدار مطلوب استفاده می‌شوند. به‌عنوان مثال، در یک مدار الکترونیکی، کنترل‌کننده PID می‌تواند جریان را به‌گونه‌ای کنترل کند که ولتاژ در یک مقدار مطلوب حفظ شود. کنترل‌کننده PID یک متغیر کنترلی $u(t)$ را بر اساس مقادیر کنونی، گذشته و آینده (یعنی مشتق) خطای $e(t)$ محاسبه می‌کند(همانطور که در معادله (۱) نشان داده شده است) .

هدف یادگیری عمیق پیدا کردن یک تابع تقریب یا تابع نگاشت f با پارامترهای θ برای نگاشت ورودی x به خروجی مطلوب y، یعنی $y = f(x, \theta)$، با فرض اینکه روابط پیچیده یا علیتی بین x و y وجود دارد می باشد که موجب "یادگیری" شبکه میشود.

با داده‌های آموزشی کافی، یادگیری عمیق می‌تواند شبکه‌ای با میلیون‌ها پارامتر (وزن (w را برای تطبیق با روابط پیچیده که نمی‌توان با استفاده از توابع تحلیلی فرمول‌بندی کرد، آموزش دهد. معمولاً یک تابع هزینه L بر اساس خروجی مطلوب y و خروجی پیش‌بینی شده $f(x, \theta)$ تعریف می‌شود تا اندازه‌گیری شود که آیا به هدف رسیده است یا خیر. هزینه از طریق انجام "backpropagation" بر وزن‌ها تأثیر می‌گذارد. به عبارت دیگر، خطا را با محاسبه گرادیان‌های وزن‌ها به هر گره توزیع می‌کند. اگر هزینه L به اندازه کافی کوچک نباشد، شبکه وزن‌های خود θ را بر اساس گرادیان‌های $\frac{\delta L}{\delta \theta}$ به‌روز می‌کند. بنابراین، مرتبط کردن "خطا" در کنترل PID با "گرادیان" در یادگیری عمیق منطقی است. این روش تا زمانی که L همگرا شود یا به اندازه کافی کوچک شود، تکرار می‌شود. بهینه‌سازهای زیادی برای به حداقل رساندن هزینه L با به‌روزرسانی θ با استفاده از گرادیان‌های $\frac{\delta L}{\delta \theta}$ پیشنهاد شده‌اند، از جمله SGD، SGD–Momentum، Adam و غیره.

۴) رابطه با جریان مدار

از مفهوم جریان الکتریکی می توان برای گرفتن تغییرات تابع عصبی و بهینه سازی آن استفاده نمود. در یک مدار الکتریکی، جریان الکتریکی توسط یک منبع ولتاژ و مقاومت کنترل می شود. منبع (مثلا باتری) انرژی لازم را برای انتقال الکترون ها از طریق مدار تأمین می کند که با استفاده از تغییر وزن (مقاومت) در رزیستور ها این کار امکان پذیر میشود. در شبکه عصبی وزن ها تعیین می کنند که قدرت ارتباطات بین نورون های شبکه به چه میزان باشد. هدف آموزش یک شبکه عصبی عمیق یافتن وزن های بهینه ای است که به شبکه اجازه می دهد تا ورودی ها را به خروجی های دقیق نگاشت کند.

بهینه ساز PID که میتوان آن را به شکل الگوریتم دیجیتال و یا آنالوگ ساخت، با استفاده از قانون اهم به ما درمورد تغییرات بعدی در ضرایب شبکه عصبی اطلاع دهد که باعث میشود تابع خطای شبکه کمینه شود. همینطور خطا می تواند به عنوان اختلاف ولتاژ بین خروجی دلخواه و خروجی واقعی شبکه در نظر گرفته شود.

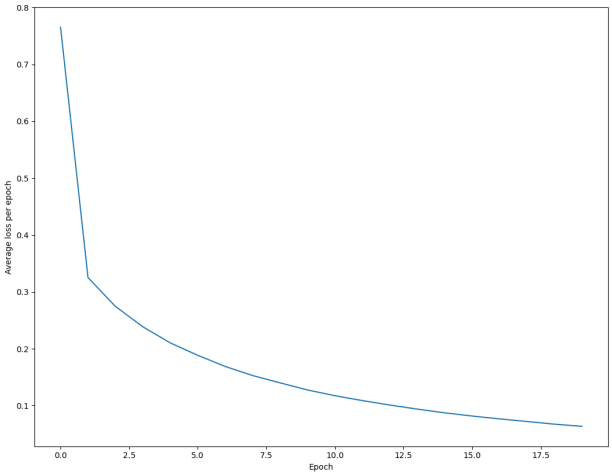
همچنین میتوان این الگوریتم را به شکل آنالوگ پیاده سازی کرد تا با متصل کردن این مدار آنالوگ به یک شبکه عصبی، شبکه را بهینه کرد. این کار را به این دلیل انجام میدهیم چون پیاده سازی این الگوریتم در سیستم های آنالوگ بسیار ساده و بهینه می باشد.

۵) PID

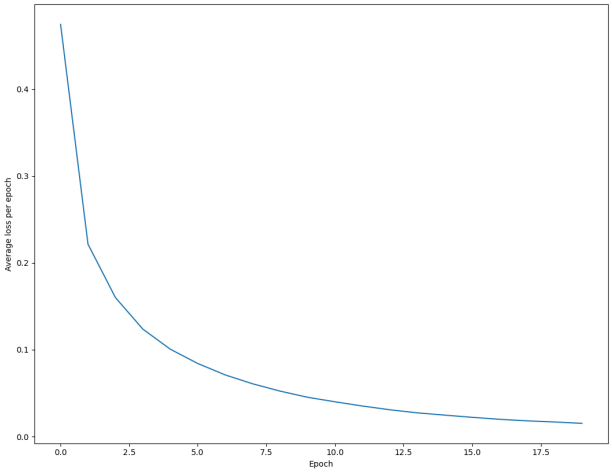
الگوریتم بهینه ساز، با گرفتن میزان خطا با استفاده از PID بعد از تکرار دفعات، ضرایب شبکه را پیدا میکند. در این الگوریتم، P نسبت خطا است که با استفاده از کم کردن این مقدار میتوان خطا را کم کرد. البته استفاده نسبت به خودی خود به اندازه کافی بهینه نیست و مدل ما دچار فراجهش (مقایسه (۳)) میشود که نیاز به مقادیر دیگر پیدا میکنیم. با استفاده از مشتق ضریب خطا میتوان مقدار تناوب و فراجهش را کم نمود. باید توجه کرد که تاثیر مشتق خطا به نسبت فاصله خطا بسیار کمتر است بنابراین ضریب K_d معمولا بزرگ تر از ضریب K_p می باشد. با این حال همچنان به نقطه بهینه مطلوب دست پیدا نکرده ایم و به سراغ انتگرال خطا میرویم، اوقاتی پیش می آید که دیگر فراجهش نداریم ولی نویز داده باعث میشود تا تغییراتی که مشتق به ضرایب وارد میکند با نویز همدیگر را خنثی کنند که باعث میشود تا به نقطه نیمه_بهینه (sub-optimal) برسیم و یادگیری مدل در مراتب بعد ایست کند. با استفاده از انتگرال خطا میتوان میزان حجم خطا را به دست آورد که جلوی این پدیده گرفته میشود.

۶) نتایج

در این بخش، ما الگوریتمی بهینه سازی برای شبکه عصبی با الهام از سیستم کنترل PID نوشتیم و آن را بر روی دیتاست معروف MNIST که اعداد دست نویس هستند آموزش دادیم تا بهینگی و تعمیم این الگوریتم بر روی این دیتاست را اندازه گیری کنیم. لازم به ذکر است که به جز پارامتر اضافی k_d که توسط معادله (۱۴) تنظیم می‌شود، سایر فراپارامتر ها در بهینه‌ساز PID ما همانند SGD-Momentum تنظیم شده‌اند، همینطور نرخ یادگیری از ۰۱.۰ شروع می‌شود. شبکه عصبی با تابع فعال کننده ReLU و ۲۵۶ گره پنهان در لایه پنهان، به دنبال آن یک لایه خروجی softmax توسط الگوریتم خطای آنتروپی متقاطع است. آموزش روی مینی‌بچ‌های با ۱۰۰ تصویر در هر بچ و برای ۲۰ تکرار روی مجموعه آموزش انجام شد. از این آزمایش می‌توان مشاهده کرد که بهینه‌ساز PID نه تنها سریع‌تر از SGD-Momentum با تلفات کمتر و دقت بالاتر همگرا می‌شود، بلکه توانایی تعمیم بالاتری نیز روی مجموعه داده اعتبارسنجی دارد. در مجموعه داده تست، بهینه‌ساز PID به دقت ٪۹۸ و SGD-Momentum به دقت ٪۹۶ دست می‌یابد. استفاده از ایده های مختلف در مهندسی برق برای توسعه الگوریتم های بهینه سازی جدید برای شبکه های عصبی عمیق یک زمینه تحقیقاتی فعال است. با ادامه بررسی این ایده ها، می توانیم روش های جدید و کارآمدتری برای آموزش شبکه های عصبی عمیق ایجاد کنیم.



شکل ۱: میزان خطا در هر مرحله (SGD)



شکل ۲: میزان خطا در هر مرحله (PID)

۷) تحلیل

با الهام از جریان الکتریکی که با استفاده از کنترل‌کننده PID در سیستم‌های الکتریکی برای حفظ ولتاژ و جریان در یک مقدار مطلوب استفاده می‌شود، ما یک رویکرد جدید کنترل‌کننده PID برای بهینه‌سازی شبکه‌های عمیق ارائه دادیم. بهینه‌ساز PID پیشنهادی از اطلاعات لحظه، گذشته و تغییر گرادیان برای به‌روزرسانی پارامترهای شبکه استفاده می‌کند و فرآیند یادگیری هاDNN را تسریع می‌کند. در سیستم‌های الکتریکی، جریان الکتریکی می‌تواند تحت تأثیر عوامل مختلفی مانند مقاومت، ولتاژ و نویز قرار گیرد. کنترل‌کننده PID می‌تواند برای جبران خطاهای ناشی از این عوامل در جریان الکتریکی استفاده شود. آزمایش ما نشان داد که بهینه‌ساز PID پیشنهادی در مقایسه با الگوریتم‌های بهینه‌سازی موجود، عملکرد بهتری در بهینه‌سازی شبکه‌های عمیق دارد. بهینه‌ساز PID پیشنهادی سرعت یادگیری را افزایش می‌دهد و نرخ خطای شبکه را کاهش می‌دهد.

منابع:

[1] Russakovsky Olga, Deng Jia, Su Hao, Krause Jonathan, Satheesh Sanjeev, Ma Sean, et al., "ImageNet large scale visual recognition challenge", IEEE International Journal of Computer Vision (IJCV), 2015.

[2] Alex Krizhevsky, Ilya Sutskever and Geoffrey E Hinton, "ImageNet classification with deep convolutional neural networks", Advances in neural information processing systems (NIPS), 2012.

[3] Shaoqing Ren, Kaiming He, Ross Girshick and Jian Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks", Advances in neural information processing systems (NIPS), 2015.

[4] Florian Schroff, Dmitry Kalenichenko and James Philbin, "FaceNet: A unified embedding for face recognition and clustering", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[5] Ian Goodfellow, Yoshua Bengio and Aaron Courville, "Deep Learning", MIT Press, 2016.

[6] Léon Bottou, "Online learning in neural networks" in chapter Online Learning and Stochastic Approximations, Cambridge University Press, pp. 9-42, 1998.

[7] Ilya Sutskever, James Martens, George Dahl and Geoffrey Hinton, "On the importance of initialization and momentum in deep learning", International Conference on Machine Learning, 2013.

[8] John Duchi, Elad Hazan and Yoram Singer, "Adaptive subgradient methods for online learning and stochastic optimization", Journal of Machine Learning Research, vol. 12, pp. 2121-2159, Jul 2011.

[9] Geoffrey Hinton, N Srivastava and Kevin Swersky, Lecture 6a overview of mini–batch gradient descent.

[10] Diederik Kingma and Jimmy Ba, "Adam: A method for stochastic optimization", International Conference for Learning Representations (ICLR), 2014.

[11] Wangpeng An, Haoqian Wang, Qingyun Sun "A PID Controller Approach for Stochastic Optimization of Deep Networks", 2018