به نام خدا

طراحی سیستم‌های دیجیتال کم‌توان

سحر توکلی

گزارش بازتولید مقاله: تخمین دقیق وضعیت شارژ باتری لیتیوم-یونی با رویکرد یادگیری عمیق و بهینه‌سازی بیزین

تابستان ۱۴۰۴

فهرست

توضیح مقاله 4

چکیده (Abstract) 4

مقدمه (Introduction) 4

مرور ادبیات (Literature Review) 5

الگوریتم‌های یادگیری عمیق (Deep Learning Algorithms) 6

نقش ابرپارامترها در الگوریتم‌های یادگیری عمیق (Role of Hyperparameters) 7

روش پیشنهادی (Proposed Methodology) 7

نتایج و بحث (Results and Discussion) 8

نتیجه‌گیری (Conclusion) 8

مقدمه و هدف 9

مروری بر مقاله اصلی 9

مراحل بازتولید مقاله 9

دریافت داده 9

پیش‌پردازش اولیه 10

حذف متادیتا و تعریف ستون‌ها 10

حذف مقادیر غیرمفید 10

محاسبه وضعیت شارژ(SoC) 10

مهندسی ویژگی‌ها و نرمال‌سازی 11

ایجاد ویژگی‌های مشتق‌شده 11

نرمال‌سازی داده‌ها 11

تولید توالی‌های زمانی برای مدل‌هایRNN 11

طراحی مدل‌های یادگیری عمیقLSTM ، GRU، BiLSTM 12

طراحی مدلLSTM 12

طراحی مدلGRU 13

طراحی مدلBiLSTM 13

انتخاب معیارهای آموزش و ارزیابی 13
آماده‌سازی برای تنظیم پارامترها..... 13
بهینه‌سازی ابرپارامترها با الگوریتم بیزین (Bayesian Optimization) وOptuna 13
بررسی و تحلیل ابرپارامترهای مدل 14
فرآیند پیاده‌سازی بهینه‌سازی باOptuna 15
آموزش نهایی مدل‌ها و ارزیابی دقیق روی داده‌های آزمون 16
ارزیابی عملکرد مدل‌ها 16
یافتن ابرپارامترها و ارزیابی نهایی..... 17
ابرپارامترهای بهینه و تنظیمات آموزش..... 17
نتایج نهایی و مقایسه عملکرد مدل‌ها 18
تحلیل نتایج و مقایسه با مقاله اصلی..... 18
عملکرد مدل‌ها 18
مقایسه با مقاله 19
دلایل احتمالی تفاوت نتایج..... 19

توضیح مقاله

یک رویکرد یادگیری عمیق بهینه‌شده با بیزین برای تخمین دقیق وضعیت شارژ باتری‌های لیتیوم-یون مورد استفاده در کاربرد وسایل نقلیه الکتریکی

چکیده (Abstract)

این مقاله یک روش خودکار و نظام‌مند برای انتخاب ابرپارامترها (hyperparameters) با استفاده از الگوریتم بهینه‌سازی بیزی (Bayesian Optimization) برای تخمین دقیق وضعیت شارژ (SOC) باتری‌های لیتیوم-یون در وسایل نقلیه الکتریکی (EV) پیشنهاد می‌دهد. اهمیت انتخاب صحیح ابرپارامترها برای دستیابی به بهترین عملکرد از مدل‌های یادگیری عمیق (DL) مورد تأکید قرار گرفته است. علاوه بر پارامترهای معمول باتری (جریان، ولتاژ و دما)، ولتاژ متوسط و جریان متوسط به عنوان پارامترهای ورودی مهم برای پیش‌بینی دقیق SOC استفاده می‌شوند، زیرا شامل سابقه گذشته و حال ولتاژها و شرایط بار هستند.

روش‌های پیشنهادی با چهار مجموعه داده مختلف که شامل دماهای متفاوت (۱۰-، ۰، ۱۰ و ۲۵ درجه سانتی‌گراد) می‌شوند، اعتبارسنجی و آزمایش شده‌اند. نتایج نشان می‌دهند که برای هر سه نوع شبکه عصبی بازگشتی (RNN) شامل LSTM، GRU و BiLSTM، پیکربندی بهینه منجر به تخمین SOC با خطای ریشه میانگین مربعات (RMSE) کمتر از ۲٪ و خطای حداکثر (Max Error) کمتر از ۵٪ می‌شود. در میان این سه، BiLSTM با ۷۰ نورون پنهان، SOC را با خطای تخمین کاهش‌یافته نسبت به سایر روش‌ها تخمین می‌زند. با استفاده از رویکرد پیشنهادی، سیستم‌های مدیریت باتری (BMS) که وضعیت باتری‌ها را در شرایط محیطی مختلف نظارت می‌کنند، می‌توانند قابل اطمینان‌تر شوند.

مقدمه (Introduction)

افزایش فروش وسایل نقلیه الکتریکی در سال‌های اخیر به دلیل کاهش انتشار گازهای گلخانه‌ای و ذخیره‌سازی انرژی‌های تجدیدپذیر، اهمیت فناوری باتری در این وسایل را افزایش داده است. باتری‌های لیتیوم-یون به دلیل چگالی توان و انرژی بالاتر، طول عمر بیشتر و نرخ خودتخلیه کمتر، گزینه ترجیحی برای ذخیره‌سازی انرژی در وسایل نقلیه هستند.

یکی از وظایف حیاتی سیستم مدیریت باتری (BMS) در وسایل نقلیه الکتریکی، نظارت بر وضعیت باتری، به ویژه وضعیت شارژ (SOC) است تا عملکرد ایمن و قابل اطمینان باتری تضمین شود. SOC به عنوان نسبت ظرفیت باقی‌مانده باتری به ظرفیت نامی آن در یک شرایط خاص تعریف می‌شود. با این حال، تخمین دقیق SOC دشوار است زیرا ظرفیت نامی باتری با افزایش سن، دما و شرایط محیطی تغییر می‌کند. همچنین، محدودیت‌های حسگرهای پتانسیومتری، آمپرمتری و هدایت‌سنجی بر تخمین SOC تأثیر می‌گذارند.

روش‌های سنتی تخمین SOC مانند روش ولتاژ مدار باز و روش آمپر-ساعت دارای محدودیت‌هایی هستند. روش‌های مبتنی بر فیلتر کالمن، اگرچه دارای ماهیت بازگشتی هستند، اما نیازمند مدل‌سازی دقیق و شناسایی پارامترهای باتری هستند. در

مقابل، رویکردهای یادگیری ماشین (ML) و به ویژه یادگیری عمیق (DL) نیازی به مدل‌های شیمیایی یا الکتریکی ندارند و راه حلی جایگزین برای پیش‌بینی دقیق SOC ارائه می‌دهند.

یکی از نگرانی‌های اصلی در روش‌های مبتنی بر داده، انتخاب ابرپارامترهای مدل مانند نرخ یادگیری، تعداد واحدهای پنهان، تعداد نورون‌های پنهان، اندازه بچ (batch size)، اپوک‌ها، تابع فعال‌سازی و نرخ دراپ‌اوت است. انتخاب نامناسب ابرپارامترها منجر به کاهش دقت پیش‌بینی می‌شود. محققان معمولاً از رویکردهای آزمون و خطا یا تکنیک‌های جستجوی جامع مانند جستجوی شبکه‌ای (grid search) و جستجوی تصادفی (random search) برای انتخاب ابرپارامترها استفاده می‌کنند که ناکارآمد و زمان‌بر است.

این مقاله، الگوریتم بهینه‌سازی بیزی را برای تنظیم ابرپارامترهای الگوریتم‌های RNN (LSTM, BiLSTM, GRU) معرفی می‌کند تا بر نقص رویکرد آزمون و خطا غلبه کند. علاوه بر پارامترهای ورودی باتری (جریان، ولتاژ و دما)، ولتاژ متوسط و جریان متوسط نیز به عنوان پارامترهای ورودی برای تخمین دقیق SOC در نظر گرفته می‌شوند. تأثیر تغییر تعداد نورون‌های پنهان بر دقت تخمین برای LSTM، BiLSTM و GRU نیز تحلیل می‌شود.

مرور ادبیات (Literature Review)

این بخش به بررسی روش‌های مختلف تخمین SOC می‌پردازد. روش‌های سنتی مانند شمارش کولن و روش جدول جستجو به دلیل محدودیت‌هایشان برای وسایل نقلیه الکتریکی ناکارآمد هستند. برای غلبه بر این کاستی‌ها، روش‌های مبتنی بر مدل، مشاهده‌گر (observer-based) و فیلتر (filter-based) پیشنهاد شده‌اند. فیلتر کالمن، یکی از این تکنیک‌ها، برای تعیین SOC و SOH باتری اهمیت زیادی پیدا کرده است. با این حال، این روش‌ها به مدل‌سازی دقیق باتری نیاز دارند که زمان و پیچیدگی محاسباتی را افزایش می‌دهد.

در دوران مدرن، تکنیک‌های تخمین SOC مبتنی بر داده به شدت توسط محققان برای پیش‌بینی دقیق SOC ترجیح داده می‌شوند. روش‌های مبتنی بر یادگیری عمیق (مانند LSTM، GRU، BiLSTM) می‌توانند به طور مستقیم سیگنال‌های عملیاتی باتری (مانند جریان و ولتاژ) را به SOC نگاشت کرده و نیاز به مدل‌سازی پرزحمت باتری یا مهندسی ویژگی را از بین ببرند.

با این حال، چالش اصلی در روش‌های مبتنی بر داده، انتخاب ابرپارامترها است. بسیاری از مطالعات قبلی از رویکردهای آزمون و خطا برای یافتن پارامترهای بهینه استفاده کرده‌اند. این رویکردها زمان‌بر و دشوار هستند، زیرا فضای جستجو برای ابرپارامترهای یادگیری عمیق به صورت نمایی بزرگ است.

این مقاله برای حل این مشکل، از بهینه‌سازی بیزی استفاده می‌کند. همچنین، پژوهش‌های موجود عمدتاً فقط ولتاژ، جریان و دما را به عنوان پارامترهای ورودی در نظر گرفته‌اند. این در حالی است که عوامل محیطی و شرایط گذشته و حال باتری (مانند ولتاژ و جریان متوسط) نیز بر مصرف انرژی و دقت تخمین SOC تأثیر می‌گذارند. این مقاله با در نظر گرفتن ولتاژ و جریان متوسط به عنوان پارامترهای ورودی، این شکاف تحقیقاتی را پر می‌کند.

الگوریتم‌های یادگیری عمیق (Deep Learning Algorithms)

در این مطالعه، سه نوع شبکه عصبی بازگشتی (RNN) برای تخمین SOC باتری EV مورد بررسی قرار گرفته‌اند:

- Long Short-Term Memory - LSTM:

شبکه‌های عصبی بازگشتی سنتی (RNN) قادر به مدیریت دنباله‌های ورودی طولانی به دلیل مشکلات انفجار گرادیان و ناپدید شدن گرادیان نیستند. LSTM به عنوان یک RNN پیشرفته با مکانیزم دروازه‌ای (gated RNN) پیشنهاد شد تا دنباله‌های ورودی طولانی را مدیریت کند. LSTM از سه دروازه اصلی تشکیل شده است: دروازه ورودی (input gate)، دروازه فراموشی (forget gate) و دروازه خروجی (output gate).

  - دروازه فراموشی (Forget Gate): تصمیم می‌گیرد که کدام اطلاعات از وضعیت سلول (cell state) حذف شود.
  - دروازه ورودی (Input Gate): وضعیت سلول را به‌روزرسانی کرده و تصمیم می‌گیرد که داده‌ها باید در وضعیت سلول نوشته شوند یا خیر.
  - دروازه خروجی (Output Gate): خروجی وضعیت سلول (c(t)) را تولید کرده و تعیین می‌کند که کدام داده به عنوان خروجی وضعیت پنهان (hidden state output) منتقل شود.
  - وضعیت کاندید (Candidate State): برای تصمیم‌گیری در مورد اطلاعاتی که در وضعیت سلول نوشته می‌شوند، استفاده می‌شود. این دروازه‌ها جریان اطلاعات را در سلول تنظیم می‌کنند. وضعیت پنهان (h(t)) نوعی از مشخصات داده‌های مرحله زمانی قبلی را کدگذاری می‌کند، در حالی که وضعیت سلول (c(t)) مجموعه‌ای از داده‌ها را از تمام مراحل زمانی پردازش شده قبلی کدگذاری می‌کند.

- Gated Recurrent Unit - GRU:

GRU که در سال ۲۰۱۴ پیشنهاد شد، می‌تواند وابستگی‌های دنباله‌ای بلندمدت را مدیریت کند و دارای ساختار داخلی ساده‌ای نسبت به LSTM است. GRU حافظه کمتری نیاز دارد و در حذف گرادیان‌ها مؤثرتر است. این شبکه از دو دروازه اصلی تشکیل شده است:

  - دروازه به‌روزرسانی (Update Gate - z(t)): تعیین می‌کند که چه مقدار از ورودی جدید برای به‌روزرسانی وضعیت پنهان استفاده شود.
  - دروازه بازنشانی (Reset Gate - r(t)): مشخص می‌کند که وضعیت پنهان قبلی تا چه حد باید فراموش شود.

- LSTM دولایه (Bi-layered LSTM - BiLSTM):

BiLSTM یک مدل دنباله‌ای است که شامل دو لایه LSTM است: یکی برای پردازش ورودی به جلو (forward) و دیگری برای پردازش ورودی به عقب (backward). این معماری می‌تواند دو شبکه LSTM متمایز را مشاهده کند، که یکی دنباله

توکن‌ها را به ترتیب اصلی و دیگری به صورت معکوس دریافت می‌کند. خروجی نهایی، جمع احتمالات هر شبکه LSTM است. این رویکرد به مدل اجازه می‌دهد تا هم اطلاعات گذشته و هم اطلاعات آینده را برای پیش‌بینی‌های خود در نظر بگیرد.

نقش ابرپارامترها در الگوریتم‌های یادگیری عمیق (Role of Hyperparameters)

تنظیم ابرپارامترها، یا بهینه‌سازی ابرپارامتر، فرآیند تعیین بهینه‌ترین ابرپارامترها برای استفاده در یک مدل است. مهم‌ترین ابرپارامترهای الگوریتم‌های یادگیری عمیق عبارتند از:

- نرخ یادگیری (Learning Rate): اندازه گام بهینه‌ساز را در هر تکرار آموزش کنترل می‌کند. نرخ یادگیری بیش از حد بالا می‌تواند باعث بی‌ثباتی و واگرایی شود، در حالی که نرخ یادگیری بسیار پایین می‌تواند منجر به همگرایی کند شود.
- اپوک‌ها (Epochs): تعداد دفعاتی که مدل با کل مجموعه داده آموزشی آموزش می‌بینند. افزایش اپوک‌ها می‌تواند عملکرد مدل را افزایش دهد، اما اگر با بی‌دقتی انجام شود، می‌تواند منجر به بیش‌برازش (overfitting) شود.
- تعداد لایه‌ها (Number of Layers): عمق مدل را تعیین می‌کند که می‌تواند به شدت بر پیچیدگی و ظرفیت یادگیری آن تأثیر بگذارد.
- تعداد گره‌ها در هر لایه (Number of Nodes per Layer) / نورون‌های پنهان (Hidden Neurons): این ابرپارامتر عرض مدل را کنترل کرده و بر توانایی آن در نمایش روابط پیچیده در داده‌ها تأثیر می‌گذارد.

روش پیشنهادی (Proposed Methodology)

مقاله برای غلبه بر معایب روش‌های دستی یا آزمون و خطا در انتخاب ابرپارامترها، از الگوریتم بهینه‌سازی بیزی استفاده می‌کند. این الگوریتم به یافتن بهترین پارامترهای شبکه کمک می‌کند تا عملکرد شبکه افزایش یابد.

- پارامترهای ورودی: علاوه بر پارامترهای معمول باتری (جریان، ولتاژ و دما)، ولتاژ متوسط و جریان متوسط نیز به عنوان پارامترهای ورودی برای تخمین دقیق SOC در این مطالعه در نظر گرفته شده‌اند. ولتاژ متوسط اطلاعات بیشتری در مورد وضعیت SOC قبلی فراهم می‌کند و جریان متوسط اطلاعاتی در مورد بار متصل به باتری (فعلی و گذشته) ارائه می‌دهد.
- مجموعه داده: برای اعتبارسنجی دقت مدل پیشنهادی در دماهای مختلف (۱۰-، ۰، ۱۰ و ۲۵ درجه سانتی‌گراد)، از مجموعه داده‌ای که از دانشگاه مک‌مستر همیلتون به دست آمده است، استفاده می‌شود.
- پیکربندی مدل: سه الگوریتم یادگیری عمیق (LSTM، GRU و BiLSTM) با تعداد نورون‌های پنهان متفاوت مورد بررسی قرار می‌گیرند.

نتایج و بحث (Results and Discussion)

یافته‌های این مطالعه نشان می‌دهد که برای هر سه نوع RNN (LSTM، GRU و BiLSTM)، پیکربندی بهینه (که توسط بهینه‌سازی بیزی یافت شده است) منجر به تخمین SOC با خطای ریشه میانگین مربعات (RMSE) کمتر از ۲٪ و خطای حداکثر (Max Error) کمتر از ۵٪ می‌شود.

- عملکرد در دماهای مختلف: مدل پیشنهادی در چهار دمای مختلف (۱۰-، ۰، ۱۰ و ۲۵ درجه سانتی‌گراد) آزمایش و نتایج نشان می‌دهد که BiLSTM عملکرد بهتری نسبت به LSTM و GRU برای پارامترهای ورودی انتخاب شده (ولتاژ، جریان، دما، ولتاژ متوسط و جریان متوسط) در تعداد متغیر نورون‌های پنهان دارد.
- تأثیر تعداد نورون‌های پنهان:
  - BiLSTM با ۷۰ نورون پنهان در مقایسه با سایر روش‌ها و تعداد نورون‌ها، SOC را با خطای تخمین کاهش‌یافته‌ای تخمین می‌زند.

نتیجه‌گیری (Conclusion)

این مقاله یک روش بهینه‌سازی بیزی برای تنظیم ابرپارامترهای شبکه‌های عصبی بازگشتی (LSTM, GRU, BiLSTM) جهت تخمین دقیق وضعیت شارژ (SOC) باتری‌های لیتیوم-یون در وسایل نقلیه الکتریکی ارائه می‌دهد. این رویکرد به طور خودکار بهترین پارامترها را تعیین می‌کند و بر معایب روش‌های دستی غلبه می‌یابد.

با استفاده از ولتاژ متوسط و جریان متوسط به عنوان پارامترهای ورودی اضافی، مدل پیشنهادی قادر به پیش‌بینی دقیق‌تر SOC تحت شرایط محیطی مختلف است. نتایج تجربی بر روی مجموعه داده‌های باتری تحت دماهای مختلف (۱۰-، ۰، ۱۰ و ۲۵ درجه سانتی‌گراد) نشان می‌دهد که هر سه مدل RNN عملکرد قابل قبولی دارند. با این حال، در میان آن‌ها، BiLSTM با ۷۰ نورون پنهان بهترین عملکرد را از نظر کاهش خطای تخمین SOC نشان می‌دهد. این تحقیق می‌تواند به توسعه سیستم‌های مدیریت باتری قابل اطمینان‌تر در شرایط عملیاتی متنوع کمک کند.

مقدمه و هدف

این گزارش به بررسی گام‌به‌گام بازتولید مقاله‌ای تحت عنوان « A Bayesian Optimized Deep Learning Approach for Accurate State of Charge Estimation of Lithium Ion Batteries Used for Electric Vehicle Application » می‌پردازد. هدف اصلی بازسازی روش پیشنهادی مقاله شامل پیش‌پردازش، طراحی مدل‌های یادگیری عمیق، تنظیم ابرپارامترها و ارزیابی نهایی دقت پیش‌بینی است.

مروری بر مقاله اصلی

در مقاله، مدل‌های یادگیری عمیق شامل LSTM ، GRU و BiLSTM برای تخمین SOC به‌کار گرفته شده‌اند. ویژگی‌های کلیدی مقاله به شرح زیر است:

- داده‌ها :استفاده از مجموعه داده باتری لیتیوم-یونی مدل LG HG2 مربوط به دانشگاه مک‌مستر در دسترس در Kaggle
- ویژگی‌های ورودی :ولتاژ، جریان، دما، میانگین ولتاژ و میانگین جریان
- مدل‌ها :شبکه‌های بازگشتی شامل LSTM ، GRU، BiLSTM
- تنظیم پارامترها :بهینه‌سازی ابرپارامترها با الگوریتم Bayesian Optimization
- معیار ارزیابی: RMSE و Max Error

مراحل بازتولید مقاله

دریافت داده

نقطه شروع این پروژه، دریافت داده‌هایی است که مقاله نیز بر پایه آن‌ها طراحی شده است. این داده‌ها از آزمایش روی سلول‌های باتری لیتیوم-یون مدل LG HG2 در دانشگاه مک‌مستر (Hamilton's McMaster University) به‌دست آمده‌اند. مقاله ذکر می‌کند که آزمایش‌ها در دماهای متنوعی انجام شده‌اند تا عملکرد مدل‌ها در شرایط محیطی مختلف سنجیده شود.

در پیاده‌سازی ما، فایل داده ابتدا از سایت Kaggle به صورت دستی دانلود شد. سپس محتوای فایل ZIP استخراج و در مسیر محلی قرار گرفت. در کد، این مسیر با متغیر dataset_base_path مشخص شد. این متغیر پایه تمام مسیرهای دسترسی به فایل‌های CSV است.

```
dataset_base_path =
'C:/Users/sahar/Documents/sut/lpd/project/Dataset_Li-ion'
```

فایل‌های آموزش و آزمون به صورت مجموعه‌ای از مسیرهای نسبی در لیست‌هایی جداگانه ذخیره شدند. این فایل‌ها مربوط به داده‌های جمع‌آوری‌شده در دماهای مختلف و چرخه‌های مختلف رانندگی بودند.

پیش‌پردازش اولیه

در این گام، هدف این بود که فایل‌های CSV که شامل خروجی مستقیم آزمایشگاه بودند، به داده‌های ساختارمند و تمیز قابل استفاده برای یادگیری عمیق تبدیل شوند. این فرآیند شامل چند زیرمرحله بود:

حذف متادیتا و تعریف ستون‌ها

هر فایل شامل ۳۰ ردیف ابتدایی (metadata) بود که ربطی به داده‌های عددی نداشت. بنابراین، هنگام خواندن فایل، از

```
pd.read_csv(..., skiprows=30)
```

استفاده شد تا این ردیف‌ها نادیده گرفته شوند.

سپس ستون‌ها با نام‌هایی استاندارد و معنادار مانند'Voltage' ، 'Current'، 'Temperature' تعریف شدند. این نام‌گذاری برای استفاده در مراحل بعدی و کاهش ابهام بود.

حذف مقادیر غیرمفید

بر اساس بررسی مقاله و ساختار فایل‌ها، تنها داده‌هایی با وضعیت 'TABLE' یا 'DCH' (تخلیه باتری) برای تخمین وضعیت شارژ (SOC) مفید هستند. بنابراین، داده‌های غیرمرتبط مانند 'CHG' (شارژ) حذف شدند.

محاسبه وضعیت شارژ(SoC)

از آنجا که ستون مستقیم وضعیت شارژ در فایل‌ها وجود نداشت، باید با استفاده از ظرفیت تخلیه (Capacity) مقدار SoC محاسبه می‌شد. این کار با استفاده از رابطه زیر انجام شد:

```
df["SoC Capacity"] = max_discharge + df["Capacity"]
df["SoC Percentage"] = df["SoC Capacity"] / df["SoC Capacity"].max()
```

این محاسبه در واقع فرض می‌کند که باتری از حالت کاملاً شارژ به‌سمت تخلیه کامل حرکت می‌کند و با دانستن بیشترین تخلیه ثبت‌شده، درصد شارژ فعلی قابل تخمین است. در پایان این مرحله، خروجی‌های پردازش‌شده در قالب DataFrame های train_df و test_df تجمیع شدند.

مهندسی ویژگی‌ها و نرمال‌سازی

در این مرحله تمرکز اصلی بر ایجاد ویژگی‌های جدید و آماده‌سازی آن‌ها برای مدل‌های یادگیری عمیق بود.

ایجاد ویژگی‌های مشتق‌شده

مقاله بیان کرده که استفاده از "میانگین جریان" و "میانگین ولتاژ" اطلاعات مفیدتری درباره روند شارژ و تخلیه باتری فراهم می‌کند. برای محاسبه این ویژگی‌ها از میانگین‌گیری متحرک (Rolling Mean) استفاده شد:

```
df["I_avg"] = df["Current"].rolling(window=20).mean()
df["V_avg"] = df["Voltage"].rolling(window=20).mean()
```

همچنین ویژگی جدیدی به نام Power نیز افزوده شد که حاصل‌ضرب ولتاژ و جریان است و شاخص مناسبی برای وضعیت بار روی باتری به‌شمار می‌رود.

نرمال‌سازی داده‌ها

مدل‌های یادگیری عمیق معمولاً در صورتی عملکرد مناسبی دارند که داده‌های ورودی در بازه‌ی مشخص (مثلاً ۰ تا ۱) قرار داشته باشند. برای این منظور، از MinMaxScaler استفاده شد. ابتدا این مقیاس‌گذار فقط روی داده‌های آموزش fit شد، سپس بر روی داده‌های آموزش و تست اعمال شد.

این کار باعث شد از نشت اطلاعات از تست به مدل جلوگیری شود و مقایسه مدل‌ها روی داده‌های دیده‌نشده معتبر بماند.

تولید توالی‌های زمانی برای مدل‌های RNN

از آنجا که مدل‌های انتخاب‌شده در مقاله LSTM ، GRU و BiLSTM برای تحلیل داده‌های سری‌زمانی طراحی شده‌اند، لازم بود که داده‌ها به‌جای حالت مستقل و دوبعدی، به شکل توالی‌هایی از مشاهدات تبدیل شوند.

برای این منظور پارامتری به‌نام sequence_length تعریف شد که مشخص می‌کرد مدل برای پیش‌بینی وضعیت شارژ در لحظه t ، چه تعداد گام زمانی قبلی را در نظر بگیرد. مقدار اولیه آن برابر با 60 تعیین شد. سپس تابعی طراحی شد که با پیمایش داده‌های ورودی، این توالی‌ها را ایجاد می‌کرد:

```python
def create_sequences(X, y, seq_length):
    xs, ys = [], []
    for i in range(len(X) - seq_length):
        x = X[i:i+seq_length]
        y_val = y[i+seq_length]
        xs.append(x)
        ys.append(y_val)
    return np.array(xs), np.array(ys)
```

خروجی این تابع شامل دو آرایه‌ی X_train_sequences و y_train_sequences برای آموزش و X_test_sequences و y_test_sequences برای ارزیابی مدل‌ها بود که آماده ورود به مرحله بعدی یعنی طراحی مدل‌ها بودند.

طراحی مدل‌های یادگیری عمیقLSTM ، GRU، BiLSTM

یکی از مهم‌ترین و کلیدی‌ترین تصمیمات در این پروژه، انتخاب معماری شبکه‌های یادگیری عمیق مناسب برای پیش‌بینی وضعیت شارژ باتری است. در مقاله، سه نوع معماری بازگشتی (Recurrent) معرفی شده‌اند که قابلیت یادگیری از داده‌های سری‌زمانی را دارند:

1. LSTM (Long Short-Term Memory):حافظه بلندمدت با توانایی نگهداری وابستگی‌های طولانی‌مدت در توالی‌ها.
2. GRU (Gated Recurrent Unit): معماری ساده‌تر و سبک‌تر با سرعت آموزش بالاتر.
3. BiLSTM (Bidirectional LSTM): شبکه‌ای که اطلاعات را هم از گذشته و هم از آینده در توالی یاد می‌گیرد.

طراحی مدلLSTM

مدل LSTM طراحی شده شامل لایه‌های زیر است:

- یک لایه LSTM که ورودی با شکل (sequence_length, num_features) دریافت می‌کند. تعداد نورون‌های این لایه به‌صورت پارامتر قابل تنظیم است.
- لایه Dropout برای جلوگیری از بیش‌برازش .این لایه در هر بار آموزش، برخی از نورون‌ها را به‌صورت تصادفی غیرفعال می‌کند.
- یک یا چند لایه Dense (تمام‌اتصال) که پس از لایه بازگشتی قرار می‌گیرد و وظیفه نگاشت ویژگی‌های استخراج‌شده به خروجی را بر عهده دارد.

- لایه خروجی Dense با یک نورون که خروجی نهایی مدل یعنی مقدار پیش‌بینی شده‌ی SoC را تولید می‌کند. از آنجا که مسأله پیش‌بینی SoC یک مسأله رگرسیون است، در این لایه از هیچ تابع فعال‌سازی استفاده نمی‌شود.

طراحی مدل GRU

ساختار مدل GRU بسیار مشابه LSTM است، با این تفاوت که از لایه GRU استفاده می‌شود که نسبت به LSTM ساختار ساده‌تری دارد و تعداد پارامترهای کمتری را شامل می‌شود. این ویژگی باعث می‌شود GRU در شرایطی با داده‌های کمتر یا زمان آموزش محدود، گزینه مناسب‌تری باشد.

طراحی مدل BiLSTM

مدل BiLSTM برای بهره‌گیری از اطلاعات گذشته و آینده در توالی‌ها طراحی شده است. این مدل از یک لایه Bidirectional استفاده می‌کند که یک LSTM را در دو جهت forward و backward اجرا می‌کند و خروجی‌های آن‌ها را ترکیب می‌کند.

انتخاب معیارهای آموزش و ارزیابی

در تمامی مدل‌ها از تابع هزینه MSE (mean_squared_error) برای آموزش استفاده شد، چرا که SoC یک مقدار پیوسته است و MSE معیار مناسبی برای رگرسیون به‌شمار می‌رود. همچنین برای ارزیابی عملکرد در طول آموزش، معیار RMSE (RootMeanSquaredError)مورد استفاده قرار گرفت که مقاله نیز از آن بهره گرفته است.

آماده‌سازی برای تنظیم پارامترها

در انتهای این گام، تمامی مدل‌ها به‌گونه‌ای طراحی شدند که قابلیت دریافت مقادیر دلخواه برای ابرپارامترهایی نظیر تعداد نورون، dropout، و غیره را داشته باشند. این طراحی ما را برای گام بعدی که شامل بهینه‌سازی پارامترها با استفاده از الگوریتم بیزین است، آماده می‌کند.

بهینه‌سازی ابرپارامترها با الگوریتم بیزین (Bayesian Optimization) و Optuna

مدل‌های یادگیری عمیق دارای ابرپارامترهای متعددی هستند که انتخاب مقدار مناسب برای آن‌ها تأثیر مستقیمی بر عملکرد مدل دارد. از جمله این پارامترها می‌توان به موارد زیر اشاره کرد:

- تعداد نورون‌ها در لایه‌های مخفی(Hidden Units)
- نرخ یادگیری(Learning Rate)

- نرخDropout
- تعداد نورون‌ها در لایه‌هایDense
- اندازهBatch
- تعداد Epoch برای آموزش

مقاله به‌جای انتخاب دستی این پارامترها یا استفاده از روش‌های پرهزینه‌ای مانندGrid Search ، از الگوریتم بهینه‌سازی بیزین استفاده کرده است. این الگوریتم با یادگیری الگوی عملکرد مدل در فضای پارامترها، بهترین ترکیب ممکن را پیشنهاد می‌دهد. در پیاده‌سازی این پروژه، از کتابخانه Optuna برای این منظور استفاده شد که یکی از محبوب‌ترین و مدرن‌ترین پیاده‌سازی‌های این الگوریتم است.

بررسی و تحلیل ابرپارامترهای مدل

در فرآیند آموزش مدل‌هایLSTM ، GRU وBiLSTM ، مجموعه‌ای از ابرپارامترها با استفاده از الگوریتم بهینه‌سازی بیزین (Optuna) تنظیم شدند. این ابرپارامترها نقش حیاتی در دقت، پایداری و کارایی نهایی مدل ایفا می‌کنند. در ادامه، هریک از آن‌ها را شرح می‌دهیم:

- تعداد واحدهای لایه بازگشتی(gru_units / lstm_units / bilstm_units)

این پارامتر مشخص می‌کند که چند نورون یا واحد پردازشی در لایه بازگشتی (Recurrent Layer) مدل وجود داشته باشد. هر یک از این نورون‌ها، در واقع به‌عنوان یک حافظه عمل کرده و در هر گام زمانی اطلاعات دریافتی را نگهداری، به‌روزرسانی و منتقل می‌کند. در شبکه‌هایRNN ، این نورون‌ها مسئول یادگیری وابستگی‌های زمانی و توالی رفتار داده‌ها هستند.

- تعداد نورون‌های لایه Fully Connected (dense_units)

پس از لایه بازگشتی، اطلاعات خروجی به لایه‌های تمام‌متصل (Dense) منتقل می‌شوند. این لایه‌ها وظیفه استخراج الگوهای سطح بالا و نگاشت اطلاعات به خروجی نهایی را بر عهده دارند. پارامتر dense_units مشخص می‌کند چند نورون در این لایه قرار گیرد.

- نرخ Dropout (dropout_rate)

Dropout در طول آموزش، به‌صورت تصادفی برخی نورون‌ها را "خاموش" می‌کند. این کار باعث می‌شود مدل از وابستگی بیش‌ازحد به نورون‌های خاص جلوگیری کرده و از پدیده بیش‌برازش (Overfitting) پیشگیری شود .

dropout_rate درصدی بین ۰ تا ۱ است که احتمال خاموش شدن هر نورون را نشان می‌دهد.

- نرخ یادگیری(learning_rate)

نرخ یادگیری یکی از مهم‌ترین ابرپارامترها در آموزش مدل‌های یادگیری عمیق است. این مقدار تعیین می‌کند که در هر به‌روزرسانی وزن‌ها، چه مقدار تغییر اعمال شود. نرخ زیاد ممکن است باعث نوسان یا واگرایی شود، در حالی که نرخ کم ممکن است آموزش را بسیار کند یا مدل را در نقاط محلی گیر بیندازد.

- اندازه Batch (batch_size)

در الگوریتم‌های یادگیری، داده‌ها به بخش‌هایی با اندازه مشخص تقسیم می‌شوند و پس از هر batch ، گرادیان‌ها محاسبه و وزن‌ها به‌روزرسانی می‌شوند. این پارامتر تأثیر زیادی بر مصرف حافظه، پایداری آموزش، و سرعت اجرای الگوریتم دارد.

- تعداد Epochs (epochs)

Epoch به معنای یک بار عبور کامل از کل مجموعه داده آموزشی است. در عمل، مدل‌ها ممکن است با ۲۰ تا ۱۰۰ اپوک یا بیشتر آموزش ببینند. با استفاده از مکانیزم EarlyStopping ، اگر عملکرد مدل در اعتبارسنجی بهبود نیابد، آموزش پیش از رسیدن به حداکثر epoch متوقف خواهد شد.

فرآیند پیاده‌سازی بهینه‌سازی با Optuna

- تقسیم‌بندی داده‌های آموزشی

داده‌های آموزشی اولیه (که قبلاً به شکل توالی درآمده بودند) به دو بخش تقسیم شدند:

- X_train_tune, y_train_tune: برای آموزش در هر تکرار از بهینه‌سازی
- X_val_tune, y_val_tune: برای ارزیابی عملکرد مدل با پارامترهای پیشنهادی

```
from sklearn.model_selection import train_test_split
X_train_tune, X_val_tune, y_train_tune, y_val_tune =
train_test_split(
    X_train_sequences, y_train_sequences, test_size=0.2,
shuffle=True)
```

- تعریف تابع هدف(Objective Function)

برای هر مدل LSTM ، GRU، BiLSTM، یک تابع هدف تعریف شد که توسط Optuna بارها فراخوانی می‌شود. این تابع:

- مقادیر پارامترها را از trial دریافت می‌کند.
- یک مدل با این پارامترها می‌سازد.
- آن را آموزش می‌دهد.
- عملکرد آن را روی داده‌های اعتبارسنجی ارزیابی می‌کند.
- مقدار RMSE اعتبارسنجی را باز می‌گرداند (که هدف، کمینه‌سازی آن است).

- اجرای فرآیند بهینه‌سازی

برای هر مدل یک Study در Optuna ایجاد و بهینه‌سازی با تعدادی تکرار (مثلاً 50) اجرا شد:

```
import optuna

study_lstm = optuna.create_study(direction='minimize')
study_lstm.optimize(objective_lstm, n_trials=50)
```

در پایان هر فرآیند، بهترین پارامترها از طریق study.best_params استخراج میشوند .

در پایان این گام، سه مجموعه ابرپارامتر بهینه برای هر یک از مدل‌ها به‌دست آمدند. این مجموعه‌ها به‌عنوان ورودی به مرحله آموزش نهایی داده شدند.

آموزش نهایی مدل‌ها و ارزیابی دقیق روی داده‌های آزمون

پس از آن‌که در گام قبلی بهترین ابرپارامترها برای مدل‌هایLSTM ، GRUو BiLSTM با استفاده از Optuna به‌دست آمد، حال نوبت آن است که:

- مدل‌ها را با این پارامترهای بهینه ساخته و آموزش دهیم.
- عملکرد هر مدل را روی داده‌های آزمون X_test_sequences و y_test_sequences بسنجیم.

ارزیابی عملکرد مدل‌ها

پس از آموزش مدل‌ها، از دو معیار زیر برای ارزیابی استفاده شد:

- RMSE (Root Mean Squared Error) که متوسط خطای پیش‌بینی در مقایسه با مقدار واقعی را نشان می‌دهد.
- Max Error که بزرگ‌ترین اختلاف مطلق بین مقدار پیش‌بینی‌شده و مقدار واقعی را ثبت می‌کند. این معیار حساسیت مدل به بدترین سناریوها را بررسی می‌کند.

این دو مقدار برای هر سه مدل محاسبه و در جدولی ثبت شدند.

یافتن ابرپارامترها و ارزیابی نهایی

در این بخش، فرآیند یافتن بهترین ابرپارامترها برای مدل‌های یادگیری عمیق و ارزیابی نهایی آن‌ها بر روی داده‌های آزمون شرح داده می‌شود.

با توجه به اینکه انتخاب ابرپارامترها تأثیر قابل توجهی بر عملکرد نهایی مدل دارد، از رویکردی نظام‌مند به جای آزمون و خطا استفاده شد.

1. **بهینه‌سازی ابرپارامترها با Optuna:** برای یافتن بهترین ترکیب ابرپارامترها (مانند نرخ یادگیری، تعداد نورون‌ها، نرخ Dropout و...) از کتابخانه Optunaکه الگوریتم بهینه‌سازی بیزین را پیاده‌سازی می‌کند، استفاده شد. این الگوریتم با جستجوی هوشمند در فضای پارامترها، به سمت یافتن مقادیر بهینه حرکت می‌کند.
2. **تفکیک فرآیند آموزش :**با توجه به زمان‌بر بودن فرآیند بهینه‌سازی و محدودیت‌های محاسباتی در محیط Google Colab، تصمیم گرفته شد که فرآیند برای هر مدل در یک نوت‌بوک جداگانه اجرا شود. به این ترتیب، سه نوت‌بوک مجزا برای مدل‌هایLSTM ، GRUو BiLSTM ایجاد و اجرا گردید.
3. **آموزش و ارزیابی نهایی :**پس از یافتن بهترین پارامترها برای هر مدل، یک بار دیگر هر مدل با استفاده از کل داده‌های آموزشی و پارامترهای بهینه‌شده، آموزش داده شد. در نهایت، عملکرد مدل نهایی روی مجموعه داده آزمون که تا به حال مشاهده نکرده بود، با معیارهای **RMSE**و **Max Error**سنجیده شد.

ابرپارامترهای بهینه و تنظیمات آموزش

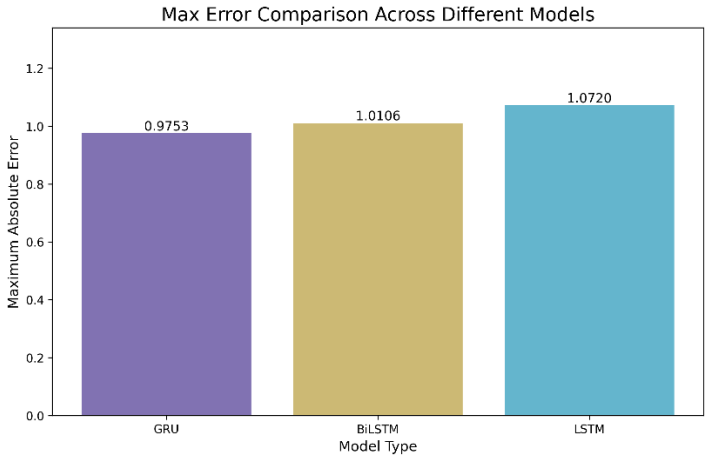
پس از اجرای فرآیند بهینه‌سازی با Optuna، بهترین ابرپارامترهای یافت‌شده برای هر مدل در جدول زیر خلاصه شده است. این پارامترها در مرحله آموزش نهایی مدل‌ها به کار گرفته شدند. سایر پارامترهای ثابت مانند بهینه‌ساز (Adam) و تابع هزینه (MSE) در تمام مدل‌ها یکسان بودند.

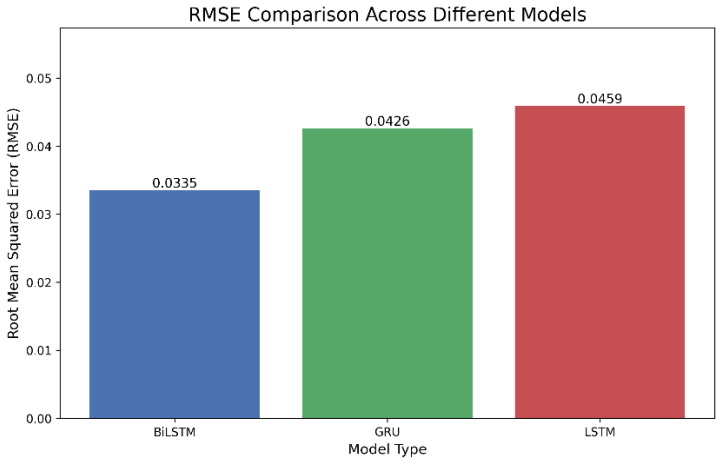
پارامتر	مقدار بهینه برای LSTM	مقدار بهینه برای GRU	مقدار بهینه برای BiLSTM
واحدهای لایه بازگشتی	70	30	100
نورون‌های لایهDense	50	50	50
نرخDropout	0.103	0.103	0.149
نرخ یادگیری	1.13e-05	4.09e-04	2.03e-04
اندازهBatch	256	256	128
تعداد Epoch بهینه‌سازی	25	25	76

نتایج نهایی و مقایسه عملکرد مدل‌ها

عملکرد نهایی هر یک از مدل‌ها پس از آموزش با پارامترهای بهینه بر روی مجموعه داده آزمون، در جدول زیر ارائه شده است. معیارها همان‌هایی هستند که در مقاله اصلی استفاده شده‌اند: خطای ریشه میانگین مربعات (RMSE) و حداکثر خطای مطلق (Max Error).

مدل	RMSEتست	Max Errorتست
LSTM	0.0459	1.0720
GRU	0.0426	0.9753
BiLSTM	**0.0335**	1.0106





تحلیل نتایج و مقایسه با مقاله اصلی

با توجه به نتایج به‌دست‌آمده در جدول بالا، می‌توان تحلیل زیر را ارائه داد:

عملکرد مدل‌ها

در میان سه مدل پیاده‌سازی شده، مدل BiLSTMبا کمترین مقدار RMSE (برابر با 0.0335) بهترین عملکرد را در تخمین وضعیت شارژ داشته است. پس از آن، مدل GRU و در نهایت LSTM قرار می‌گیرند. این نتیجه با یافته اصلی مقاله که مدل BiLSTM را به عنوان دقیق‌ترین مدل معرفی کرده بود، همخوانی دارد.

مقایسه با مقاله

- در مقاله مدل BiLSTM به‌عنوان دقیق‌ترین مدل معرفی شده بود که در تمام دماها توانست RMSE کمتر از ۲٪ و Max Error کمتر از ۵٪ ارائه دهد.
- **RMSE:** مقادیر RMSE به‌دست‌آمده در این بازتولید (بین 3.35٪ تا 4.59٪) بالاتر از مقادیر گزارش‌شده در مقاله (کمتر از 2٪) است.
- **Max Error:** حداکثر خطای مشاهده‌شده در مدل‌های ما (حدود 1.0) به طور قابل توجهی بیشتر از خطای زیر 5٪ گزارش‌شده در مقاله است. این مقدار خطا نشان می‌دهد که در برخی نقاط خاص، پیش‌بینی مدل اختلاف بسیار زیادی با مقدار واقعی داشته است.

دلایل احتمالی تفاوت نتایج

این اختلاف می‌تواند به دلایل متعددی رخ داده باشد:

1. تفاوت در تقسیم داده :ممکن است تقسیم داده‌های آموزشی و آزمون در بازتولید ما دقیقاً مشابه مقاله نباشد.
2. مقدار اولیه وزن‌ها (Random Seed): مقداردهی اولیه تصادفی وزن‌ها در شبکه‌های عصبی می‌تواند منجر به نتایج متفاوتی در اجراهای مختلف شود.
3. تفاوت‌های جزئی در پیاده‌سازی :تفاوت در نسخه‌های کتابخانه‌ها مانند TensorFlow/Keras یا تفاوت‌های جزئی در مراحل پیش‌پردازش می‌تواند بر نتایج نهایی تأثیر بگذارد.
4. فضای جستجوی پارامترها :ممکن است بازه‌ای که برای جستجوی ابرپارامترها در Optunaتعریف شده، با بازه استفاده‌شده در مقاله اصلی متفاوت باشد.

با این حال، روند کلی نتایج برتری BiLSTM با مقاله اصلی مطابقت دارد.