پژوهشگاه نیرو

NRI-ER-841-PPUSPN01-00-F

بررسی روش‌های یادگیری عمیق در پیش‌بینی بار کوتاه مدت

مدیر و مجری پروژه : نیکی مسلمی

DOI: 10.30503/nripress.2020.321


بسم الله الرحمن الرحيم

بررسی روش‌های یادگیری عمیق در پیش‌بینی بار کوتاه مدت

مدیر پروژه:

نیکی مسلمی

همکاران پروژه:

سید جواد موسوی، آرمان اشنویی

مجری: نیکی مسلمی

انتشارات پژوهشگاه نیرو

۱۴۰۰

کد گزارش NRI-ER-841-PPUSPN01-00-F

DOI: 10.30503/nripress.2020.321

ایران. وزارت نیرو. پژوهشگاه نیرو. گروه پژوهشی برنامه‌ریزی و بهره‌برداری در سیستم‌های قدرت

بررسی روش‌های یادگیری عمیق در پیش‌بینی بار کوتاه مدت/ مدیر پروژه: نیکی مسلمی. ویراستار: نازنین محمدی‌نام. کارفرما: پژوهشگاه نیرو، ۱۴۰۰.

۱۸۲ ص.: مصور، نمودار، جدول

۱. موسوی، جواد؛ همکار پروژه. ۲. اشنویی، آرمان؛ همکار پروژه. ۳. مسلمی، نیکی، مجری.

انتشارات پژوهشگاه نیرو

گروه پژوهشی برنامه‌ریزی و بهره‌برداری در سیستم‌های قدرت

نشانی: تهران، شهرک‌غرب، انتهای‌بلوارشهیددادمان، پژوهشگاه‌نیرو، کدپستی: ۱۴۶۸۶۱۳۱۱۳

تلفن: ۹-۸۸۰۷۹۴۰۱

نمابر: ۸۸۰۷۸۲۹۶

وبگاه: *www.nri.ac.ir*

پست الکترونیک انتشارات: *publications@nri.ac.ir*

DOI: 10.30503/nripress.2020.321

فرم تعهدنامه اصالت اثر

اینجانب نیکی مسلمی متعهد می‌شوم که مطالب مندرج در این گزارش حاصل کار پژوهشی اینجانب و همکاران به شرح زیر است و به دستاوردهای پژوهشی دیگران که در این پژوهش از آن استفاده شده است، مطابق مقررات ارجاع و در فهرست منابع و مآخذ ذکر گردیده است.

همکاران :

۱. امیر مشاری

۲. مازیار کریمی

۳. حسین کرمی

۴. سید جواد موسوی

۵. آرمان اشنویی

DOI: 10.30503/nripress.2020.321

پیشگفتار

در راستای پیاده‌سازی یادگیری عمیق به منظور پیش‌بینی یا کاربردهای دیگر، ابتدا لازم است به صورت مقدماتی با مفاهیم یادگیری عمیق آشنا شویم. متعاقبا و درجهت پیاده‌سازی عملیلتی در پروژه‌های مرتبط با یاد گیری ماشینی، گام مهم بعدی شناخت انواع معروف و پرکاربرد شبکه‌ها و ساختارهای موجود است. بنابراین لازم است انواع مختلف شبکه‌ها و لایه‌ها را در شبکه‌های عمیق بشناسیم تا سپس بتوانیم براساس شناخت ایجاد شده از نواع لایه و ساختار موجود، یک ترکیب مناسب را برای مسئله خودمان استخراج کنیم.

مفاهیم اولیه و ساختارهای موجود مستقل از نرم‌افزار، زبان برنامه نویسی و کتابخانه‌های نرم‌افزاری است. به این مفهوم که با ایجاد این شناخت اولیه می‌توان از هر زبان برنامه‌نویسی یا هر کتابخانه نرم‌افزاری استفاده کرد به شرطی که زیرساخت‌های لازم برای کار با لایه‌ها، توابع و در نهایت ایجاد شبکه‌های عمیق در کتابخانه مذکور فراهم شده باشد. این موضوع در مورد انواع موجود شبکه‌ها و لایه‌ها در یادگیری عمیق صادق است زیرا تمامی آنها دارای تعاریف مشخص و سیقل یافته در طول زمان هستند همچنین اسامی و رابطه‌های ریاضی پذیرفته شده‌ای دارند. بدین صورت با استفاده از کتابخانه‌های معروف در هرزبان عیناً می‌توان یک شبکه را در هر زبان برنامه نویسی ایجاد کرده و نتایج مشابه از آن دریافت کرد. این موضوع مستقل از خطای توسعه کتاب خانه (که در هر بروزرسانی ترمیم می‌گردد) و توانایی‌های ذاتی هر زبان برنامه نویسی مثل مدیریت حافظه و پردازهای CPU می‌باشد.

ذخیره انرژی الکتریکی در مقیاس بزرگ به دلیل تلفات تبدیل انرژی و هزینه سیستم‌های ذخیره‌ساز انرژی بسیار گران تمام می‌شود. در نتیجه ایجاد تعادل در عرضه و تقاضای انرژی الکتریکی از مهمترین فعالیت‌ها در صنعت برق است، زیرا تولید بیش از نیاز انرژی الکتریکی می‌تواند باعث اتلاف انرژی شود که هزینه‌های بالایی در پی خواهد داشت، در حالیکه تولید کمتر از میزان مورد نیاز نیز منجر به قطع بار مصرف‌کنندگان خواهد شد. بنابراین، پیش‌بینی‌های با دقت بالا از بار شبکه قدرت برای برنامه‌ریزی بهینه عملیات نیروگاه‌ها بسیار ارزشمند است به طوری‌که می‌توان هزینه‌های عملیاتی را کاهش داده و قابلیت اطمینان عملکرد شبکه را افزایش دهد. مقوله پیش‌بینی بار به چند دسته طبقه‌بندی می‌شود : کوتاه مدت (از چند ساعت تا چند روز)، میان مدت (از چند روز تا چند ماه) و بلند مدت (چند سال آینده). به طور معمول، پیش‌بینی بار بلند مدت و میان مدت برای برنامه‌ریزی‌های توسعه زیرساخت و همچنین برنامه‌ریزی تعمیرات استفاده می‌شود، در حالی‌که پیش‌بینی بار کوتاه مدت در بهره‌برداری روزانه سیستم‌های قدرت مورد استفاده قرار می‌گیرد. پیش‌بینی بار کوتاه مدت یکی از مسائل چالش برانگیز در صنعت برق است که طیف گسترده‌ای از علایق تحقیقاتی را به خود جلب کرده است. افزایش توسعه‌ی هوش مصنوعی و الگوریتم‌های داده کاوی، فرصت بی سابقه‌ای را برای مطالعه‌ی مدل‌های پیش‌بینی بار ایجاد کرده است. الگوریتم‌های یادگیری ماشین به عنوان زیر مجموعه‌ای از هوش مصنوعی، یک مدل ریاضی بر اساس داده‌های نمونه یا داده‌های آموزش به منظور پیش‌بینی ایجاد می‌کنند. در سال‌های اخیر، رویکردهای یادگیری عمیق به عنوان یک زیرشاخه از یادگیری ماشین در پیش‌بینی بار بسیار مورد استفاده قرار گرفته است. در این گزارش سعی بر آن است که منابع موجود در زمینه کاربرد یادگیری عمیق در پیش‌بینی بار مورد مطالعه و دسته‌بندی قرار گیرد.

DOI: 10.30503/nripress.2020.321

فهرست مطالب

عنوان **صفحه**

پیشگفتار VI
مقدمه XVI

فصل ۱- مقدمه ۱

۱-۱- یادگیری عمیق چیست؟ ۲
۱-۲- شبکه عمیق چیست؟ ۲
۱-۳- یادگیری تقویتی عمیق ۲
۱-۴- پیشرفت‌های ایجاد شده در معماری شبکه‌ها ۳
۱-۴-۱- پیشرفت‌ها در انواع لایه ۳
۱-۴-۲- پیشرفت در انواع نورون ۳
۱-۴-۳- معماری هیبریدی ۳
۱-۵- از مهندسی ویژگی‌ها تا یادگیری خودکار ویژگی ۳
۱-۵-۱- مهندسی ویژگی‌ها ۴
۱-۵-۲- یادگیری ویژگی ۴

فصل ۲- اصول پایه در یاد گیری عمیق ۵

مقدمه ۶
۲-۱- اصول مشترک معماری در شبکه های عمیق ۶
۲-۱-۱- پارامترها ۶
۲-۱-۲- لایه‌ها ۷
۲-۱-۳- توابع فعال‌سازی ۷
۲-۱-۳-۱- توابع فعال‌سازی برای یک معماری عمومی ۷
۲-۱-۳-۲- توابع فعال‌سازی در لایه پنهان ۷
۲-۱-۳-۳- لایه خروجی با هدف رگرسیون ۸
۲-۱-۳-۴- لایه خروجی برای طبقه بندی باینری ۸
۲-۱-۳-۵- لایه خروجی برای طبقه بندی چند کلاسه ۸
۲-۱-۴- تابع فاصله از هدف ۸
۲-۱-۴-۱- بازسازی *Cross-Entropy* ۹
۲-۱-۵- الگوریتم‌های بهینه‌یابی ۹
۲-۱-۵-۱- روش‌های مرتبه اول ۱۰
۲-۱-۵-۲- روش‌های مرتبه دوم ۱۰
۲-۱-۶- ابرپارامترها ۱۱
۲-۱-۶-۱- اندازه لایه ۱۱

DOI: 10.30503/nripress.2020.321

۲-۱-۶-۲-*Magnitude* ۱۲
2-1-6-3- Regularization ۱۳
۲-۱-۶-۴-*Mini-Batching* ۱۴
۲-۱-۶-۵- خلاصه ۱۴
۲-۲- بلوک‌های ساختمان شبکه عمیق ۱۴
۲-۲-۱-*RBM* ۱۵
۲-۲-۱-۱- طرح بندی شبکه ۱۵
۲-۲-۱-۲-*Training* ۱۶
۲-۲-۱-۳-*Reconstruction* ۱۷
۲-۲-۱-۴- کاربردهای دیگر *RBM*ها ۱۸
۲-۲-۲-*Autoencoders* ۱۹
۲-۲-۲-۱- تشابه با پرسپترون چند لایه ۱۹
۲-۲-۲-۲- ویژگی‌های *Autoencoder* ها ۱۹
۲-۲-۲-۳- آموزش *Autoencoder* ۱۹
۲-۲-۲-۴- نسخه‌های عمومی *Autoencoder* ها ۱۹
۲-۲-۲-۵- سایر کابردهای *Autoencoder* ها ۲۰
۲-۲-۳-*Variational Autoencoders* ۲۰

فصل ۳- معماری‌های عمده در شبکه عمیق ۲۲

مقدمه ۲۳
۳-۱- شبکه پیش آموزش غیر نظارتی (*UPN*) ۲۳
۳-۱-۱- شبکه باور عمیق (*DBN*) ۲۳
۳-۱-۱-۱- استخراج ویژگی با استفاده از لایه‌های *RBM* ۲۴
۳-۱-۱-۲- تنظیم دقیق *DBN* با استفاده از شبکه فیدفوروارد ۲۵
۳-۱-۱-۳- شبکه مقایسه‌ای مولد (*GAN*) ۲۵
۳-۱-۱-۴- آموزش مدل‌های مولد، یادگیری غیر نظارتی و *GAN* ۲۵
۳-۱-۱-۵-*GAN* های شرطی ۲۷
۳-۱-۱-۶- مقایسه *GAN* و *VAE* ۲۷
۳-۲- شبکه عصبی کانولوشنی (*CNN*) ۲۸
۳-۲-۱- زمینه بیولوژیکی ۲۹
۳-۲-۲- درک مستقیم ۲۹
۳-۲-۳- نگاهی کلی به معماری *CNN* ۲۹
۳-۲-۳-۱- قرارگیری فضایی نورون‌ها ۳۰
۳-۲-۳-۲- تحول در اتصال بین لایه‌ها ۳۱
۳-۲-۴- لایه‌های ورودی ۳۱
۳-۲-۵- لایه‌های کانولوشنی ۳۱

DOI: 10.30503/nripress.2020.321

۳-۲-۵-۱- کانولوشن ۳۲
۳-۲-۵-۲- فیلترها ۳۳
۳-۲-۵-۳- نگاشت‌های فعال‌سازی ۳۳
۳-۲-۵-۴- به اشتراک‌گذاری پارامتر ۳۵
۳-۲-۵-۵- فیلترها و رندرهای یاد گرفته شده ۳۵
۳-۲-۵-۶- توابع فعال‌سازی *ReLU* به عنوان لایه ۳۵
۳-۲-۵-۷- ابرپارامترهای لایه کانولوشن ۳۵
۳-۲-۵-۸-*Batch Normalization* ۳۷
۳-۲-۶- لایه ادغام ۳۷
۳-۲-۷- لایه‌های کاملا متصل ۳۸
۳-۲-۸- کاربردهای دیگر شبکه‌های *CNN* ۳۸
۳-۲-۹-*CNN* های مشهور ۳۸
۳-۲-۱۰- خلاصه ۳۹
۳-۳- شبکه‌های عصبی بازرخداد ۳۹
۳-۳-۱- مدل‌سازی زمان ۴۰
۳-۳-۱-۱- گم شدن در زمان ۴۰
۳-۳-۱-۲- بازخورد موقتی و حلقه‌ها در اتصالات ۴۰
۳-۳-۱-۳- توالی‌ها و داده‌های سری زمانی ۴۰
۳-۳-۱-۴- فهم ورودی و خروجی مدل ۴۱
۳-۳-۲- ورودی حجمی *D*۳ ..... ۴۱
۳-۳-۲-۱- سری‌های زمانی غیر هموار و پوشاندن ۴۲
۳-۳-۳- چرا نمی‌توان از مدل‌های مارکوو استفاده کرد؟ ۴۳
۳-۳-۴- معماری عمومی شبکه عصبی بازرخداد ۴۳
۳-۳-۴-۱- معماری و گام‌های زمانی ۴۳
۳-۳-۵- شبکه‌های *LSTM* ۴۴
۳-۳-۵-۱- ویژگی‌های شبکه‌های *LSTM* ۴۴
۳-۳-۵-۲- معماری شبکه *LSTM* ۴۵
۳-۳-۵-۳- واحدهای *LSTM* ۴۶
۳-۳-۵-۴-*GRU* ۴۸
۳-۳-۵-۵- لایه‌های *LSTM* ۴۸
۳-۳-۵-۶- آموزش ۴۸
۳-۳-۵-۷-*BPTT* و *BPTT* بریده ۴۸
۳-۳-۶- دامنه‌های کاربرد و شبکه‌های ترکیبی ۵۰
۳-۴- شبکه‌های عصبی بازگشتی ۵۰
۳-۴-۱- معماری شبکه ۵۰
۳-۴-۲- تنوع شبکه‌های عصبی بازگشتی ۵۰

DOI: 10.30503/nripress.2020.321

۳-۴-۳- کاربردهای شبکه‌های عصبی بازگشتی ۵۱
۳-۴-۴- مثالی از کاربرد شبکه‌های عصبی بازگشتی در *NLP* ۵۲
۳-۵- پیش‌بینی بار با استفاده از یاد گیری عمیق ۵۳
۳-۶- مطالعه آینده ۵۳
مراجع ۵۳

فصل ۴- مفاهیم پایه یادگیری ماشین و یادگیری عمیق ۵۸

۴-۱- معرفی هوش مصنوعی، یادگیری ماشین ۵۹
4-2- شبکه‌های عصبی ۶۱
۴-۳- یادگیری عمیق (*DL*) ۶۲
۴-۴- محبوبیت، مزایا و چالش‌های یادگیری عمیق ۶۴
۴-۵- روش‌های یادگیری عمیق ۶۴
۴-۵-۱- یادگیری بدون نظارت ۶۵
۴-۵-۱-۱- شبکه های عصبی خودرمزنگار ۶۵
۴-۵-۱-۲- شبکه های عصبی مولدی ۶۶
۴-۵-۲- یادگیری با نظارت ۶۷
4-۵-۲-۱- شبکه‌های عصبی بازگشتی ۶۸
۴-۵-۲-۲- شبکه‌های عصبی کانولوشنی ۶۸
4-۵-۳- یادگیری نیمه نظارتی ۶۹
۴-۵-۴- یادگیری تقویتی ۷۰

فصل ۵- بررسی مراجع در زمینه استفاده از یادگیری ماشین در پیش بینی بار سیستم‌های قدرت ۷۴

۵-۱- مقدمه ۷۵
۵-۲- مطالعات صورت گرفته بر روی پیش‌بینی بار با استفاده از شبکه‌های عصبی ۷۵
۵-۳- مطالعات صورت گرفته بر روی پیش‌بینی بار با استفاده از شبکه‌های عمیق ۷۷
۵-۴- دسته‌بندی مقالات مرور شده ۸۸
مراجع ۴۵

فصل ۶- پایتون ۵۱

۶-۱- پایتون ۵۲
۶-۱-۱- مزایای پایتون ۵۲
۶-۱-۲- معایب پایتون ۵۳
۶-۲- کتابخانه تنسورفلو ۵۴

فصل ۷- ساختار *LSTM* برای ورودی بار، ویژگی‌های آب و هوایی و ویژگی‌های تقویمی ۵۵

7-۱- فراخوانی کتابخانه‌های مورد نیاز ۵۶
۷-۱-۱- کتابخانه پانداس ۵۶
۷-۱-۲- کتابخانه *Numpy* ۵۶

DOI: 10.30503/nripress.2020.321

۷-۱-۳- کتابخانه *Matplotlib* ۵۷
۷-۱-۴- کتابخانه *joblib* ۵۷
۷-۱-۵- تعریف ماژول ۵۷
۷-۱-۶- کتابخانه *math* ۵۸
۷-۱-۷- کتابخانه *Keras* ۵۸
۷-۲- فراخوانی داده ۵۸
۷-۳- تعیین متغیرهای آموزش و تست ۵۹
۷-۴- پردازش موازی ۶۰
۷-۵- تبدیل داده‌ها به صورت ساعتی ۶۲
۷-۶- جابه‌جایی داده‌ها ۶۳
۷-۷- ورودی و خروجی برای آموزش ۶۴
۷-۸- تعریف مدل شبکه عصبی ۶۴
۷-۹- آموزش شبکه *LSTM* ۶۵
۷-۱۰- آماده‌سازی مجموعه داده برای پیش‌بینی ۶۵
۷-۱۱- ورودی و خروجی تست ۶۷
۷-۱۲- انجام پیش‌بینی ۶۷
۷-۱۳- رسم نمودارهای خروجی ۶۷
۷-۱۴- نحوه کد نویسی توابع استفاده شده ۶۸

فصل ۸- ساختار *LSTM* برای ورودی فقط بار (تفکیک بارها براساس نوع روز) ۷۰

۸-۱- ساختار کلی ۷۱
۸-۲- نحوه عملکرد کد نوشته شده ۷۱
۸-۳- شبه کد ۷۴
۸-۴- خروجی‌های مسئله ۷۶
۸-۴-۱- نتایج اجراء کد ۷۶

فصل ۹- ترکیب دو *LSTM* و *MLP* برای ورودی بار، ویژگی‌های آب و هوایی و ویژگی‌های تقویمی ۸۲

۹-۱- ساختار کلی ۸۳
۹-۲- نحوه عملکرد کد نوشته شده ۸۴
۹-۳- شبه کد ۸۵
۹-۴- خروجی‌های مسئله ۸۷
۹-۴-۱- نتایج اجراء کد ۸۷
۹-۵- نتیجه‌گیری ۹۴
مراجع ۹۵
پیوست الف: کد کامل مدل‌های پیش‌بینی ۹۵

DOI: 10.30503/nripress.2020.321

فهرست اشکال

عنوان **صفحه**

شکل ۲–۱ : شبکه RBM ۱۶
شکل ۲–۲ : فرایند باز سازی در شبکه RBM ۱۷
شکل ۲–۳ : نمونه از اعداد دست نوشته در دیتاست MNIST (تعدادی از تصاویر جدا شده از این دیتاست و تبدیل شده به فرمت قابل نمایش [24]) ۱۸
شکل ۲–۴ : ساخت اعداد MNIST با استفاده از RBM [26] ۱۸
شکل ۲–۵ : معماری شبکه Autoencoder ۱۹
شکل ۲–۶ : معماری شبکه VAE ۲۰
شکل ۳–۱ : معاری DBN ۲۴
شکل ۳–۲ : خروجی توابع فعال‌سازی در ابتدای آموزش شبکه [30] ۲۴
شکل ۳–۳ : ظاهر شدن ویژگی‌ها خروجی توابع فعال‌سازی در هنگام آموزش شبکه [30] ۲۴
شکل ۳–۴ : یادگیری ویژگی‌های مربوط به اعداد MNIST به سمت انتهای آموزش شبکه [30] ۲۴
شکل ۳–۵ : عملکرد لایه‌های معکوس–کانلوشنی [34] ۲۶
شکل ۳–۶ : تصاویر تولید شده اتاق خواب با استفاده از DCGAN [35] ۲۷
شکل ۳–۷ : CNN و بینایی کامپیوتری [38] ۲۸
شکل ۳–۸ : معماری سطح بالا و عمومی CNN [41] ۳۰
شکل ۳–۹ : حجم سه بعدی ورودی لایه ۳۱
شکل ۳–۱۰ : لایه کانولوشن همراه با حجم ورودی و خروجی [42] ۳۱
شکل ۳–۱۱ : عملیات کانولوشن [43] ۳۲
شکل ۳–۱۲ : کانولوشن و نگاشت فعال‌سازی ۳۳
شکل ۳–۱۳ : حجم فعال‌سازی خروجی مربوط به لایه کانولوشن ۳۴
شکل ۳–۱۴: اتصال نرون‌های لایه کانولوشن به حجم ورودی (مفهوم برگرفته از [44]) ۳۴
شکل ۳–۱۵ : مثالی از فیلترهای آموزش دیده [45] ۳۵
شکل ۳–۱۶ : گام ۲ پیکسلی [46] ۳۶
شکل ۳–۱۷ : یک نمونه Zero-padding شده با یک پیکسل [47] ۳۶
شکل ۳–۱۸ : اتساع کانولوشن: a) اتساع برابر ۱، b) اتساع برابر ۲ و c) اتساع برابر ۴ [48] ۳۷
شکل ۳–۱۹ : مثالی از اپراتور max pooling با گام ۲ [51] ۳۸
شکل ۳–۲۰ : ورودی شبکه‌های معمولی و ورودی شبکه‌های بازرخداد [65] ۴۲
شکل ۳–۲۱ : نمایش گام‌های زمانی در ورودی شبکه‌های بازرخداد، شبکه پیش–خورد در سمت چپ و شبکه بازرخداد در سمت راست ۴۲
شکل ۳–۲۲ : ایجاد ماسک برای گام‌های زمانی خاص ۴۳
شکل ۳–۲۳ : معماری شبکه عصبی چند لایه فیدفوروارد ۴۵
شکل ۳–۲۴ : شبکه چند لایه فید فوروارد با نمایی تسطیح شده ۴۵
شکل ۳–۲۵ : اتصالات بازرخداد روی لایه‌های پنهان در شبکه LSTM ۴۵
شکل ۳–۲۶ : باز کردن طومار شبکه بازرخداد در طول محور زمان ۴۶
شکل ۳–۲۷ : بلوک دیاگرام LSTM [69] ۴۶

DOI: 10.30503/nripress.2020.321

شکل ۳–۲۸ : BPTT استاندارد [65]..... ۴۹
شکل ۳–۲۹ : نحوه عملکرد truncated BPTT [65]..... ۴۹
شکل ۳–۳۰: ساختار یک شبکه عصبی بازگشتی که بخش‌هایی از یک تصویر و جملات یک زبان طبیعی را پارس کرده است [75]..... ۵۱
شکل ۴–۱ : مراحل یادگیری ماشین..... ۵۹
شکل ۴–۲: جزئیات مرحله یادگیری..... ۶۰
شکل ۴–۳: شبکه عصبی انسان..... ۶۱
شکل ۴–۴ : شماتیک یک لایه از یک شبکه عصبی مصنوعی..... ۶۱
شکل ۴–۵: شماتیک یک شبکه عصبی دو لایه..... ۶۲
شکل ۴–۶: شماتیک یک شبکه عصبی عمیق..... ۶۳
شکل ۴–۷: شماتیک یک شبکه عصبی کانولوشنی عمیق..... ۶۴
شکل ۴–۸: انواع روش‌های یادگیری عمیق..... ۶۵
شکل ۴–۹: ساختار یک شبکه خودرمزنگار..... ۶۶
شکل ۴–۱۰: ساختار یک شبکه خودرمزنگار با یک ورودی عدد دست نویس..... ۶۶
شکل ۴–۱۱: پیشرفت توانایی‌های شبکه های عصبی مولدی در طول ۴ سال [۷]..... ۶۷
شکل ۴–۱۲: شماتیک یک شبکه عصبی بازگشتی..... ۶۸
شکل ۴–۱۳: لایه‌های شبکه عصبی کانولوشنی..... ۶۹
شکل ۴–۱۴: ساختار کلی یک RL..... ۷۰
شکل ۴–۱۵: شماتیک بازی پانگ [۱۰]..... ۷۱
شکل ۴–۱۶ : بیان تصویری روش گرادیان سیاست [۱۰]..... ۷۲
شکل ۴–۱۷: دیاگرام کارتونی ۴ بازی پانگ [۱۰]..... ۷۳
شکل ۵–۱: فلوچارت پیش بینی بار با استفاده از LSTM [۳۳]..... ۷۸
شکل ۵–۲: نتایج پیش‌بینی بار متوسط برای سه روز (Sumr: Summer) [۳۳]..... ۷۹
شکل ۵–۳: نتایج پیش‌بینی بار پیک برای سه روز (Sumr: Summer) [۳۳]..... ۷۹
شکل ۵–۴: مدل شبکه عصبی عمیق EMD-GRU [۴۵]..... ۸۱
شکل ۵–۵: نتایج تجزیه بار اصلی به چند زیرسری[۴۵]..... ۸۲
شکل ۵–۶: مقایسه نتایج پیش‌بینی بار بدست آمده [۴۵]..... ۸۲
شکل ۵–۷: شماتیک مدل پیشنهادی [۶۰]..... ۸۵
شکل ۵–۸: مقایسه نتایج پیش‌بینی بار بدست آمده [۶۰]..... ۸۵
شکل ۵–۹: آنالیز آماری از مقالات دسته‌بندی شده در زمینه نوع دستاورد..... ۴۴
شکل ۵–۱۰: توزیع مدل‌های مختلف مبتنی بر یادگیری عمیق در پیش‌بینی بار..... ۴۵
شکل ۵–۱۱: توزیع ویژگی‌های استفاده شده در مقالات متمرکز بر کاربرد یادگیری عمیق در پیش‌بینی بار..... ۴۵
شکل ۷–۱: داده‌های بار کلی برای آموزش..... ۵۹
شکل ۷–۲: داده‌های بار کلی برای تست..... ۶۰
شکل ۷–۳: ساختار کل شبکه عمیق بکار رفته برای پیش‌بینی بار..... ۶۵
شکل ۷–۴: نتایج پیش‌بینی بدست آمده برای روز اول..... ۶۸

DOI: 10.30503/nripress.2020.321

شکل ۸–۱: فرایند پیش‌بینی در یک حلقه برای ۴۸ ساعت آینده با استفاده از بار و واقعی و خروجی خود شبکه..... ۷۲
شکل ۸–۲: حلقه‌های متوالی پیش‌بینی برای پیش‌بینی روزهای متوالی..... ۷۳
شکل ۸–۳: پیش‌بینی تعدادی از روزهای دسته اول در ۲۴ ساعت اول و دوم پیش‌بینی؛ تصاویر (الف)، (ج) و (ه) مربوط به ۲۴ ساعت اول و تصاویر (ب)، (د) و (ی) مربوط به ۲۴ ساعت دوم..... ۷۸
شکل ۸–۴: مقدار MAPE مربوط به ۳۱ روز ابتدایی بازه پیش‌بینی از نوع دسته اول در ۲۴ ساعت اول پیش‌بینی..... ۷۸
شکل ۸–۵: مقدار MAPE مربوط به ۹ روز انتهایی بازه پیش‌بینی از نوع دسته اول در ۲۴ ساعت اول پیش‌بینی..... ۷۸
شکل ۸–۶: مقدار MAPE مربوط به ۳۱ روز ابتدایی بازه پیش‌بینی از نوع دسته اول در ۲۴ ساعت دوم پیش‌بینی ۷۹
شکل ۸–۷: مقدار MAPE مربوط به ۹ روز انتهایی بازه پیش‌بینی از نوع دسته اول در ۲۴ ساعت دوم پیش‌بینی ۷۹
شکل ۸–۸: پیش‌بینی تعدادی از روزهای دسته دوم در ۲۴ ساعت اول و دوم پیش‌بینی؛ تصاویر (الف)، (ج) و (ه) مربوط به ۲۴ ساعت اول و تصاویر (ب)، (د) و (ی) مربوط به ۲۴ ساعت دوم..... ۸۰
شکل ۸–۹: مقدار MAPE مربوط به ۳۱ روز ابتدایی بازه پیش‌بینی از نوع دسته دوم در ۲۴ ساعت اول پیش‌بینی ۸۰
شکل ۸–۱۰: مقدار MAPE مربوط به ۳۱ روز انتهایی بازه پیش‌بینی از نوع دسته دوم در ۲۴ ساعت دوم پیش‌بینی ۸۱
شکل ۹–۱: ساختار کل شبکه عمیق بکار رفته برای پیش‌بینی که ترکیبی از LSTM و MLP است..... ۸۴
شکل ۹–۲: پیش‌بینی تعدادی از روزهای دسته اول در ۲۴ ساعت اول و دوم پیش‌بینی؛ تصاویر (الف)، (ج) و (ه) مربوط به ۲۴ ساعت اول و تصاویر (ب)، (د) و (ی) مربوط به ۲۴ ساعت دوم..... ۹۰
شکل ۹–۳: مقدار MAPE مربوط به ۳۱ روز ابتدایی بازه پیش‌بینی از نوع دسته اول در ۲۴ ساعت اول پیش‌بینی..... ۹۱
شکل ۹–۴: مقدار MAPE مربوط به ۹ روز انتهایی بازه پیش‌بینی از نوع دسته اول ۲۴ ساعت اول پیش‌بینی..... ۹۱
شکل ۹–۵: مقدار MAPE مربوط به ۳۱ روز ابتدایی بازه پیش‌بینی از نوع دسته اول در ۲۴ ساعت دوم پیش‌بینی ۹۲
شکل ۹–۶: مقدار MAPE مربوط به ۹ روز انتهایی بازه پیش‌بینی از نوع دسته اول در ۲۴ ساعت دوم پیش‌بینی ۹۲
شکل ۹–۷: پیش‌بینی تعدادی از روزهای دسته دوم در ۲۴ ساعت اول و دوم پیش‌بینی؛ تصاویر (الف)، (ج) و (ه) مربوط به ۲۴ ساعت اول و تصاویر (ب)، (د) و (ی) مربوط به ۲۴ ساعت دوم..... ۹۳
شکل ۹–۸: مقدار MAPE مربوط به ۳۱ روز ابتدایی بازه پیش‌بینی از نوع دسته دوم در ۲۴ ساعت اول پیش‌بینی ۹۳
شکل ۹–۹: مقدار MAPE مربوط به ۳۱ روز انتهایی بازه پیش‌بینی از نوع دسته دوم در ۲۴ ساعت دوم پیش‌بینی..... ۹۴
شکل ۹–۱۰: درصد روزهای با مقادیر MAPE در بازه ۰ تا ۲ درصد، ۲ تا ۵ درصد و بیشتر از ۵ درصد..... ۹۵

DOI: 10.30503/nripress.2020.321

فهرست جداول

عنوان **صفحه**

جدول ۵–۱: خلاصه‌ای از تحقیقات صورت گرفته شده در زمینه‌ی پیش‌بینی بار با استفاده از یادگیری عمیق..... ۳۳
جدول ۵–۲: دسته بندی مقالات از نقطه نظر نوع مدل و نوآوری مدل..... ۳۹
جدول ۵–۳: دسته بندی مقالات از نقطه نظر ویژگی‌ها..... ۴۱
جدول ۵–۴: دسته بندی مقالات از نقطه نظر پیش پردازش، انتخاب و استخراج ویژگی..... ۴۳

DOI: 10.30503/nripress.2020.321

مقدمه

پیش‌بینی پارامترهای مهم سیستم قدرت از جمله بار و قیمت یکی از مهمترین مسائل در بهره‌برداری بهینه از سیستم‌های قدرت در بازار رقابتی است. پیش‌بینی بار روزانه در شرکت‌های توزیع که به منظور ارائه به شرکت مدیریت شبکه می‌باشد امری جدا ناپذیر در شبکه‌های توزیع محسوب می‌گردد. پیش‌بینی دقیق بار، قیمیت تولید برق را درشبکه قدرت کاهش خواهد داد و در بحث بهره‌برداری و قابلیت اطمینان شبکه قدرت موثر و امری ضروری می‌باشد. پیش‌بینی بار با روش‌های آماری معمول از دقت لازم برخوردار نبوده و کاهش خطای این پیش‌بینی تاثیر بسزایی در کاهش هزینه تولید، خاموشی‌های ناخواسته و جریمه‌های اقتصادی دارد. همچنین، با توجه به تاثیرپذیری الگوهای بار از عوامل مختلفی مانند عوامل آب و هوایی، اقتصادی، اجتماعی و ...؛ پیش‌بینی دقیق بار امر دشواری می‌باشد. همچنین، سری زمانی بار به طور معمول رفتار پیچیده‌ای دارد. از جمله پیچیدگی‌های نمودار بار روزانه، رفتار غیر خطی و متغیر آن است.

روش‌های مختلفی برای پیش‌بینی بار ارائه شده تا دقت پیش‌بینی بار را بهبود ببخشند و نرم‌افزارهای صنعتی متعددی نیز بر پایه این روش‌ها تهیه شده‌اند. از جمله این روش‌ها می‌توان به انواع سری‌های زمانی، هموارسازی نمایی، فیلتر کالمن، شبکه‌های عصبی و شبکه‌های فازی عصبی اشاره نمود. سه روش اول جزو روش های استاتیکی هستند و قابلیت تفسیر مشخصات فیزیکی اجرا را دارند. اما توانایی تشخیص رفتارهای غیر خطی سیگنال‌های بار را ندارند. این در حالیست که روش‌های عصبی و فازی عصبی در این زمینه موفق‌تر عمل می‌کنند و قابلیت بهتری در مدل‌سازی داده‌های ورودی با رفتارهای غیرخطی بسیار دارند. اما نیاز به مدل‌های قوی‌تر برای پیش‌بینی بار هنوز هم در اولویت بالایی قرار دارد. در این راستا، رویکردهای یادگیری عمیق اخیراً مورد توجه بسیاری از محققان قرار گرفته و پیشرفت‌های چشمگیری را در بسیاری از حوزه‌ها مانند مدل‌سازی صوتی، پردازش زبان طبیعی و تشخیص تصویر نشان داده‌اند. در یک تعریف کلی، یادگیری عمیق یک زیر شاخه از یادگیری ماشین و بر مبنای مجموعه‌ای از الگوریتم‌ها است که برای یادگیری نمایشی سلسله‌مراتبی از داده‌ها به کار می‌روند. شبکه‌های عصبی عمیق، یک اصطلاح کلی برای مجموعه‌ای از شبکه‌های عصبی با معماری چند لایه است که نشان می‌دهند چگونه شبکه‌های عصبی با تعداد زیادی از لایه‌ها می‌توانند در ایجاد ساختارهای بازنمایی مورد نیاز در یادگیری عمیق موفق عمل کنند. این سازه‌های لایه‌لایه‌ای عمیق قابلیت انتزاع ویژگی را افزایش داده و مدل‌سازی الگوهای غیرخطی پیچیده را امکان‌پذیر می‌کند.

DOI: 10.30503/nripress.2020.321

فصل ۱- مقدمه

DOI: 10.30503/nripress.2020.321

مفاهیم مطرح شده در این گزارش به درک عمیق از معماری‌های متفاوت شبکه‌های یادگیری عمیق و در نهایت پیاده سازی آنها کمک می‌کند. قبل از معرفی معماری‌های عمده مربوط به شبکه‌های باید به دو سوال پاسخ دهیم: یادگیری عمیق چسیت؟ و شبکه عمیق چیست؟ که در ادامه این فصل به این دو سوال خواهیم پرداخت.

۱-۱- یادگیری عمیق چیست؟

چهار تفاوت اصلی بین شبکه یادگیری عمیق و شبکه‌های چند لایه پیشخورد شامل موارد زیر است:

۱. تعداد بیشتری نرون نسبت به گذشته
۲. روش‌های پیچیده‌تری برای اتصال لایه‌ها
۳. نیاز به توان پردازشی فوق‌العاده برای آموزش شبکه
۴. استخراج خودکار ویژگی

اتصال لایه‌ها از حالت تماماً متصل در شبکه‌های قدیمی به تکه‌های متصل محلی در شبکه CNN و اتصالات بازرخداد در شبکه‌های RNN تغییر کرده است.

اتصالات بیشتر به معنی داشتن پارامترهای بیشتر برای بهینه سازی است که این خود نیاز به توان پردازشی بسیار بیشتری را ایجاد می‌کند. این توان پردازشی در طول ۲۰ سال گذشته بوجود آمده است. این پیشرفت‌ها شالوده‌ای را ایجاد می‌کند برای ساخت نسل بعدی شبکه‌های عصبی که قابلیت استخراج ویژگی، مدل کردن فضاهای پیچیده‌تر و عملکرد هوشمندانه تری را دارد.

۱-۲- شبکه عمیق چیست؟

برای تعریف شبکه عمیق از چهار معماری شبکه‌های عمیق که در زیر آمده است شروع می‌کنیم.

۱. شبکه پیش آموزش غیر نظارتی
۲. شبکه عصبی کانولوشنی
۳. شبکه عصبی بازرخداد
۴. شبکه عصبی بازگشتی

این ساختارها در طول ۲۰ سال گذشته توسعه یافته‌اند. بنابراین به زبان ساده ساختارهای بالا هرجایی مورد استفاده قرار گیرند، معرف شبکه‌های عمیق هستند.

۱-۳- یادگیری تقویتی عمیق

در یادگیری تقویتی به جای روش‌های آموزش، مسئله ای که باید حل شود مورد توجه قرار می‌گیرد. به یاد گیرنده گفته نمی‌شود که چه عملی را انجام دهد در عوض به عامل اجازه داده می‌شود تا عملی را که منجر به بیشترین پاداش می‌گردد کشف کند که از طریق تجربه در شبیه‌سازی بدست می‌آید. در یادگیری تقویتی عامل یک مدل از پیش آموزش دیده را در اختیار ندارد.

تابع هدف براساس پاداش و هدفی که عامل به دنبال آن باید باشد تعریف می‌گردد. سیستم آموزش ورودهای محیطی را به عامل می‌دهد همچنین زمانی که خروجی شبیه‌سازی بازی شده مثبت باشد پاداش‌ها را به عامل خواهد داد. بعضی اوقات عمل انجام شده منجر به پاداش در لحظه نمی‌گردد بلکه می‌توان در آینده پاداشی را در بر داشته باشد. بنابراین مکانیزم آزمون و خطا همراه با پاداش‌های با تأخیر ویژگی‌های کلیدی یادگیری تقویتی هستند.

DOI: 10.30503/nripress.2020.321

یادگیری تقویتی عمیق نسخه‌ای از یادگیری عمیق است که در آن شبکه عصبی به عنوان یک تقریب زننده‌ی عمومی تابع به کار می‌رود. در حالت معمول رفتار شبکه عصبی مرزبندی نشده است و اثبات همگرایی برای آن وجود ندارد. ولی استفاده از شبکه عصبی به عنوان تقریب زننده‌ی عمومی تابع نتایج خوبی می‌دهد.
در سال ۲۰۱۳ مقاله‌ای توسط تیم DeepMind در Deep Learning Workshop ۲۰۱۳NIPS منتشر شد که در آن از Q Learning عمیق برای بازی آتاری استفاده شده بود. در آن از الگوریتم استاندارد Q Learning استفاده شده بود و تقریب زننده‌ی تابع آنها CNN بود که در آن عامل آتاری بازی می‌کند و پیکسل‌های تصویر صفحه نمایش به عنوان ورودی و CNN به عنوان مدل داخلی استفاده شده بود [۱].

۱-٤- پیشرفت‌های ایجاد شده در معماری شبکه‌ها

با حرکت از شبکه‌های چند لایه فیدفوروارد به شبکه‌هایی با ساختاری مثل CNN و RNN تغییرات شامل نحوه برپا کردن لایه‌ها، چگونگی ساخت نرون‌ها و چگونگی اتصال لایه‌ها می‌شود. معماری‌های مختلف شبکه، با توجه به مزیت‌های انواع مختلف داده ایجاد گردیده‌اند.

۱-۴-۱- پیشرفت‌ها در انواع لایه

باتوجه به معماری نوع لایه‌ها به شدت متفاوت هستند. شبکه باور عمیق (DBN) موفقیت خوبی را با استفاده از لایه‌های ماشین بولتزمن محدود شده (RBM) برای استخراج ویژگی‌ها در پیش آموزش، بدست آورده است. شبکه CNN از تابع فعال‌سازی متفاوتی استفاده می‌کند همچنین اتصال لایه‌ها را نیز ، از لایه‌های کاملاً متصل به مسیرهای متصل محلی تغییر داده است. شبکه عصبی بازرخداد نیز از ارتباطاتی استفاده می‌کند که بعد زمان را در سری‌های زمانی بهتر مدل کند.

۱-٤-۲- پیشرفت در انواع نورون

شبکه عصبی بازرخداد به طور ویژه‌ای یک پیشرفت چشمگیر در انواع نورون (یاخته عصبی) ایجاد کرده است که در شبکه LSTM استفاده می‌گردد. این نورون‌های جدید مخصوص RNN به عنوان واحد حافظه در LSTM یا به عنوان GRU استفاده می‌گردد. توضیحات بیشتر در مورد هریک در بخش‌های بعدی آمده است.

۱-٤-۳- معماری هیبریدی

در ادامه تطبیق داده ورودی به نوع معماری، معماری‌های نو ظهوری را می‌بینیم که هم بعد زمان و هم داده تصویری را شامل می‌شود. برای مثال طبقه بندی اشیاء در ویدئو با ترکیب CNN و شبکه عصبی بازرخداد در یک شبکه هیبریدی واحد موفقیت خوبی را کسب کرده است [۲]–[۵]. معماری‌های هیبریدی به ما اجازه می‌دهد مزایای خوب هر دو جهان را در یک مورد به خصوص استفاده کنیم.

۱-۵- از مهندسی ویژگی‌ها تا یادگیری خودکار ویژگی

علاوه بر لایه‌ها و نورون‌های جدید که در شبکه‌ای عمیق مورد استفاده قرار می‌گیرد، یک کلاسبند نیز در انتها وجود دارد که با استفاده از ویژگی‌های استخراج شده در لایه قبل به عنوان ورودی کار طبقه بندی را انجام می‌دهد. استخراج ویژگی یک روند کلی

DOI: 10.30503/nripress.2020.321

در معماری‌های مختلف شبکه‌های عمیق می‌باشد. هر معماری به صورت متفاوتی ویژگی‌ها را می‌سازد که در آن خبره شده و برای نوع مشخصی از داده‌ها کاربرد دارد [۶].

۱-۵-۱- مهندسی ویژگی‌ها

هنر ساخت ویژگی‌ها برای مدت‌ها نشانه بارز یادگیری ماشینی بود. کسانی که در رقابت یادگیری ماشینی پیروز بودند اغلب پایگاه داده را به خوبی مطالعه کرده و فوت و فن‌های محرمانه‌ای را استفاده می‌کردند تا فرایند آموزش تا جای ممکن برای الگوریتم آنها ساده گردد. اغلب داده‌ها، متن‌های ستونی یا جدولی هستند که می‌توان حوزه‌ای از دانش را روی یک ستون مشخص به کار گرفت. بنابراین ساخت ویژگی‌ها بسیار سرراست هستند. این نوع از ویژگی‌های دست‌ساز، مدل بسیار دقیقی را ایجاد می‌کند، ولی نیاز به تجربه فراوان دارد و به شدت زمانبر است.
طبقه بندی تصاویر یک مثال جالب است که ایجاد ویژگی‌های دست‌ساز برای آن بسیار سختر از ساخت ویژگی برای داده‌های جدولی است. اطلاعات موجود در تصاویر تحت تاثیر نورپردازی، زاویه و ... قرار دارد. استخراج و ساخت ویژگی برای تصاویر نیاز به یک رویکرد جدید داشت که منجر به انقلاب CNN شد.

۱-۵-۲- یادگیری ویژگی

بر خلاف اطلاعات حساب افراد که در یک جدول قرار دارد، بینی در یک تصویر چهره می‌تواند در هر پیکسلی قرار بگیرد. با استفاده از CNN شبکه را آموزش می‌دهیم تا لبه‌های تصویر بینی را تشخیص دهد و سپس در سطح پایین شکل بینی را با استفاده از این لبه درک کند. ممکن است لایه اول مسئول استخراج این ویژگی باشد. این ویژگی‌ها پس از استخراج در لایه‌های بالاتر با یکدیگر ترکیب شده و ویژگی به اسم صورت را به وجود می‌آوردند. بنابراین CNN بر اساس وظیفه‌اش دائما این سوال را می‌پرسد: این یک صورت است؟
گرفتن داده‌های خام پیچیده و ایجاد ویژگی‌های سطح بالا به منظور داشتن یک کلاسبند ساده از ویژگی‌های بارز یادگیری عمیق است.

DOI: 10.30503/nripress.2020.321

فصل ۲- اصول پایه در یاد گیری عمیق

DOI: 10.30503/nripress.2020.321

مقدمه

در این بخش از گزارش تلاش می‌کنیم مفاهیم پایه یادگیری عمیق را بیان کنیم. این کمک خواهد کرد که درک عمیقتری از آنچه در ساختارهای مختلف در شبکه‌های عمیق وجود دارد بدست آوریم. در بخش بعد به تشریح چند ساختار معروف شبکه عمیق خواهیم پرداخت.

۲-۱- اصول مشترک معماری در شبکه های عمیق

در جهت عمق بخشیدن به دانش پایه در مورد شبکه‌های عمیق این بخش اضافه گردید. مفاهیم پایه در تمام شبکه‌های عصبی شامل موارد زیر است که در اینجا به صورت ویژه توسعه آن‌ها برای شبکه‌های عمیق بررسی خواهد شد.

- پارامترها
- لایه‌ها
- توابع فعال‌سازی
- تابع فاصله از هدف[۱]
- روش بهینه سازی
- پارامترهای خارجی[۲]

با استفاده از این مفاهیم بلوک‌های ساختمانی شبکه عمیق مثل موارد زیر قابل توصیف هستند.

- RBM[3] ها
- اتوانکودرها[۴]

با استفاده از مفاهیم بالا انواع معماری شبکه‌های عمیق مثل موارد زیر را می‌توان ایجاد کرد.

- UPN
- CNN
- شبکه عصبی بازرخداد
- شبکه عصبی بازگشتی

در ادامه هریک از مفاهیم پایه را توضح خواهیم داد.

۲-۱-۱- پارامترها

پارامترها به طور مستقیم در ارتباط با وزن‌های روی ارتباطات بین نرون‌ها می‌باشد. در شبکه عمیق نیز به همین صورت است و تلاش برای بهینه کردن این وزن‌ها انجام می‌شود. مسئله اصلی در شبکه عمیق چگونگی اتصال لایه‌ها در معماری‌های مختلف است. برای مثال در شبکه DBN دو دسته موازی از اتصالات فیدفوروارد به دو شبکه مجزا وجود دارد که این خود بر اساس

[1] Loss Function
[2] Hyperparameters
[3] Restricted Boltzmann Machine
[4] Autoencoder

DOI: 10.30503/nripress.2020.321

پارامترهای شبکه تنظیم می‌گردد. یکی از این اتصالات مربوط به لایه‌های شبکه RBM بوده که کاربرد آن استخراج ویژگی برای شبکه دیگر است و اتصالات دیگر مربوط به شبکه فیدفوروارد معمولی می‌باشد. این مورد یکی از تعداد زیادی مثال، مربوط به تنظیم پارامتر در شبکه‌های عمیق است.

۲-۱-۲- لایه‌ها

در شبکه‌های فیدفوروارد سه نوع لایه ورودی، پنهان و خروجی وجود دارد. در شبکه‌های عمیق انواع بیشتری از این لایه‌ها وجود دارد که در ادامه ارتباط هریک را با معماری مشخصی از شبکه عمیق بررسی خواهیم کرد. لایه‌ها همچنین می‌توانند توسط زیر شبکه‌ها در بعضی از معماری‌های عمیق بیان گردند. لایه‌ها در شبکه‌های عمیق واحدهای بنیادی معماری هستند. لایه‌ها را می‌توان با تغییر توابع فعال‌سازی به صورت دلخواه شخصی‌سازی کرد. امکان ترکیب لایه‌های مختلف برای رسیدن به یک هدف مشخص وجود دارد. لایه‌های متفاوت دارای ابرپارامترهای[1] متفاوتی برای آغاز آموزش شبکه هستند که در ادامه بحث خواهد شد. تنظیم این ابرپارامترها برای جلوگیری از overfit شدن بسیار مفید است.

۲-۱-۳- توابع فعال‌سازی

توابع فعال‌سازی شبکه‌های عمیق با نوع سنتی آن در شبکه‌های فیدفوروارد متفاوت است. یادگیری ویژگی‌های سطح بالا در شبکه‌های عمیق یک تغییرشکل[2] غیر خطی است که به خروجی لایه قبل اعمال می‌گردد و اجازه می‌دهد تا یادگیری شبکه روی داده‌ها در یک فضای محدود شده صورت گیرد.

۲-۱-۳-۱- توابع فعال‌سازی برای یک معماری عمومی

بسته به اینکه چه تابع فعال‌سازی را انتخاب می‌شود، نتایج نشان می‌دهد که برخی توابع فعال‌سازی برای انواعی از داده مناسب‌تر هستند (منظور داده‌های چگال و داده‌های کم پشت می‌باشد). استفاده از توابع فعال‌سازی در طراحی شبکه عمیق را می‌توان در دو دسته قرار داد.

- لایه‌های پنهان
- لایه خروجی

لایه‌های پنهان به دنبال استخراج ویژگی‌های سطح بالا هستند که همواره در حال پیشرفت می‌باشند. بسته به معماری که با آن سروکار داریم یک زیرمجموعه مشخص از توابع فعال‌سازی مورد استفاده قرار می‌گیرد. در هنگام تنظیم دقیق شبکه‌های عمیق با توابع فعال‌سازی بیشتر سروکار خواهیم داشت.

از آنجایی که در لایه ورودی فقط نیاز به عبور بردار داده‌های خام است، نیازی به وجود توابع فعال‌سازی نخواهد بود.

۲-۱-۳-۲- توابع فعال‌سازی در لایه پنهان

توابعی که به طور معمول استفاده می‌شود شامل موارد زیر است:

- Sigmoid
- Tanh

[1] Hyperparameter
[2] Transform

DOI: 10.30503/nripress.2020.321

- Hard tanh
- ReLU[1] و جایگزین‌های آن

بیشتر توزیع‌های پیوسته از داده‌های ورودی به خوبی توسط ReLU مدل می‌شوند. جایی که ReLU نتیجه خوبی نمی‌دهد، استفاده از Tanh مناسب است (درصورتی که شبکه خیلی عمیق نباشد). البته باید توجه داشت که ممکن است ابرپارامترهای دیگری نیز در ارتباط با شبکه وجود داشته باشد. در سال‌های اخیر چه در عرصه کاربرد و چه در عرصه تحقیقات، Sigmoid ها کمتر مورد توجه هستند. توابع فعال‌سازی در معماری‌های مختلف در آرایش متفاوتی قرار خواهند گرفت. در صورت ادامه کار در حوزه یادگیری عمیق متوجه خواهید شد که در عمل خانواده توابع ReLU انقلابی را در تحقیقات شبکه‌های عمیق ایجاد کرده‌اند.

۲-۱-۳-۳- لایه خروجی با هدف رگرسیون

بسته به این‌که چه خروجی از مدل خود انتظار داریم، در صورتی که خروجی مدنظر یک عدد حقیقی است از توابع فعال‌سازی خطی استفاده خواهیم کرد.

۲-۱-۳-٤- لایه خروجی برای طبقه بندی باینری

در این مورد که فقط یک کلاس وجود دارد، از تابع فعال‌سازی Sigmoid استفاده خواهیم کرد که خروجی آن بین ۰.۰ تا ۱.۰ است به استثناء خود مقادیر ۰.۰ و ۱.۰ این مقادیر حقیقی بدست آمده به عنوان توزیع احتمالاتی تفسیر می‌شوند.

۲-۱-۳-٥- لایه خروجی برای طبقه بندی چند کلاسه

اگر مسئله مورد نظر یک مدل‌سازی چند کلاسه باشد، باید بیشترین رتبه در هر کلاس را بدست آوریم. برای این کار می‌توان لایه خروجی softmax همراه با یک تابع argmax() را به کار برد تا بیشترین رتبه را در تمام کلاس‌ها بدست آورد. تابع softmax توزیع احتمال روی تمام کلاس‌ها را می‌دهد.

اگه بخواهیم به ازاء هر خروجی به صورت جداگانه‌ای کلاس‌بندی داشته باشیم. به جای softmax از لایه خروجی sigmoid با تعداد n نرون استفاده خواهیم کرد. بنابراین یک توزیع احتمال جداگانه‌ای بین ۰.۰ و ۱.۰ برای هر کلاس خواهیم داشت.

۲-۱-٤- تابع فاصله از هدف

تابع فاصله از هدف میزان توافق بین خروجی پیش‌بینی شده و خروجی واقعی را کمی‌سازی می‌کند. از این تابع برای جریمه کردن کلاس‌بندی نادرست یک بردار ورودی استفاده می‌گردد. برای این کار می‌توان از توابع فاصله از هدف زیر استفاده کرد.

- باخت مربع[2]
- باخت لگاریتم[3]
- باخت هینژ[4]

[1] Rectified Linear Unit
[2] Squared loss
[3] Logistic loss
[4] Hinge loss

DOI: 10.30503/nripress.2020.321

- احتمال لوگاریتم منفی[1]

توابع فاصله از هدف در سه دسته زیر قرار می‌گیرند.

- رگرسیون
- کلاس‌بندی
- بازسازی[2]

دو دسته اول واضح هستند ولی دسته سوم در استخراج ویژگی غیر نظارتی کاربرد دارد و یکی از دلایلی است که شبکه‌های عمیق به نتایج فوق العاده دقیقی دست یافتند. در معماری‌های خاصی از شبکه‌های عمیق وقتی توابع فاصله از هدف از نوع بازسازی با توابع فعال‌سازی مناسب جفت می‌شوند، بطور موثری ویژگی‌ها را استخراج می‌کنند. برای مثال تابع فاصله از هدف cross-entropy چند کلاسه در ترکیب با توابع فعال‌سازی softmax در یک لایه برای کلاس‌بندی نتیجه بسیار خوبی می‌دهد.

۲-۱-٤-۱- بازسازی Cross-Entropy

نحوه استفاده از این تابع فاصله از هدف به این صورت است که ابتدا یک نویز گوسی (که نوعی از نویز سفید است) اعمال می‌گردد سپس تابع فاصله از هدف شبکه را به ازاء نتایجی که کمتر مشابه ورودی واقعی هستند جریمه می‌کند. این کار باعث می‌شود شبکه ویژگی‌های مختلف را در تلاش برای ساخت مجدد ورودی به طور موثری یاد بگیرد و خطا را کمینه کند. این تابع فاصله از هدف در شبکه‌های شامل RBM و در فاز پیش آموزش استفاده می‌شود.

۲-۱-۵- الگوریتم‌های بهینه‌یابی

آموزش مدل در یادگیری ماشینی شامل یافتن بهترین مجموعه مقادیر برای بردار پارامترها است. می‌توان به یادگیری ماشینی به عنوان یک مسئله بهینه‌یابی نگاه کرد به این صورت که هدف کمینه کردن تابع فاصله از هدف با بدست آوردن پارامترهای تابع پیش‌بینی برپایه مدل مدنظر ما است. الگوریتم‌های بهینه‌یابی را می‌توان به دو دسته زیر تقسیم کرد.

۱. مرتبه اول[3]
۲. مرتبه دوم[4]

الگوریتم‌های مرتبه اول ماتریس جاکوبیان[5] را محاسبه می‌کنند. جاکوبیان ماتریسی از مشتقات جزئی مقادیر تابع فاصله از هدف با توجه به هر پارامتر است. الگوریتم یک قدم درجهتی که توسط جاکوبیان تعیین شده برمی‌دارد.

در الگوریتم مرتبه دوم مشتق جاکوبیان محاسبه می‌گردد (مشتق ماتریس مشتقات جزئی). اینکار با محاسبه تقریب هسیان[6] آن انجام می‌گیرد. روش مرتبه دوم وابستگی متقابل بین پارامترها را در زمانی که تعیین می‌گردد هر پارامتر چقدر تغییر کند، درنظر می‌گیرد. الگوریتم مرتبه دوم گام‌های بهتری را انتخاب می‌کند ولی محاسبه آن زمان بسیار بیشتری می‌گیرد. جزئیات ارائه شده

[1] Negative log likelihood
[2] Reconstruction
[3] First-Order
[4] Second-Order
[5] Jacobian Matrix
[6] Hessian

DOI: 10.30503/nripress.2020.321

در مورد الگوریتم بهینه‌یابی به این دلیل است که بتوان انتخاب کرد چه الگوریتمی و در چه زمینه‌ای[1] مناسب است. همچنین توجه داشته باشید که الگوریتم‌های بهینه‌یابی دیگری نیز وجود دارد که در زیر آمده است:

- الگوریتم ژنتیک[2]
- بهینه‌یابی توده ذرات[3]
- بهینه‌یابی کلنی مورچه‌گان[4]
- گداختگی شبیه‌سازی شده[5]

۲-۱-۵-۱- روش‌های مرتبه اول

درواقع وقتی یک گام را در یک زمان برای رسیدن به هدف برمی‌داریم، روش مرتبه اول گرادیان (جاکوبیان) را در هر گام محاسبه می‌کند تا مشخص شود در چه جهتی در گام بعدی حرکت کنیم. یعنی در هر گام محاسبه می‌کنیم که بهترین جهت ممکن بعدی برای حرکت کدام است. به همین جهت بهینه‌یابی یک جستجو است زیرا بهترین مسیر به سمت کمترین مقدار تابع فاصله از هدف را بدست می‌دهد.

Gradient descent یکی از دسته الگوریتم‌های یافتن مسیر است. نسخه‌های مختلفی از این الگوریتم وجود دارد. SGD[6] یکی از الگوریتم‌های پر کاربرد یادگیری ماشینی است و چندین مرتبه سریعتر از BGD[7] است. قدرت SGD پیاده‌سازی سریع و پردازش سریع دیتاست‌های بزرگ است [12]–[7]. همچنین می‌توان آن را از طریق نرخ یادگیری یا اطلاعات مربوط به الگورتم مرتبه دوم تنظیم کرد. یکی از دلایل مشهور بودن این الگوریتم، پایداری[8] آن درمقابله با بروزرسانی‌های نویزی است. توجه داشته باشید که تکنیک‌ها دیگری مثل momentum و RMSProp می‌توانند روی نرخ یادگیری اثر بگذارند.

۲-۱-۵-۲- روش‌های مرتبه دوم

تمام روش‌های مرتبه دوم مربوط به محاسبه یا تقریب زدن هسیان است. همانطور که گفته شد هسیان را می‌توان مشتق جاکوبیان در نظر گرفت. یعنی ماتریسی از مشتقات مرتبه دوم مثل شتاب که از مشتق سرعت بدست می‌آید. وظیفه هسیان توصیف انحناء در هر نقطه است که به روش‌های زیر انجام می‌گیرد.

- Limited-memory BFGS (L-BFGS) [13]
- نزول گرادیان[9] [14]
- Hessian-free [15]

به این الگوریتم‌های بهینه‌یابی به عنوان یک روش جستجوی جعبه سیاه[10] نگاه کنید که خطا را کمینه کرده و گرادیان نسبی برای هر لایه را تعیین می‌کند.

L-BFGS:

[1] Context
[2] Genetic algorithms
[3] Particle swarm optimization
[4] Ant colony optimization
[5] Simulated annealing
[6] Stochastic Gradient Descent
[7] Batch Gradient Decent
[8] Robustness
[9] Conjugate gradient
[10] Black box

DOI: 10.30503/nripress.2020.321

یک الگوریتم بهینه‌یابی است که روش شبکه نیوتن نیز نامیده می‌شود. همانطور که از نام آن مشخص است یک نسخه از الگوریتم Broyden-Fletcher-Goldfarb-Shanno می‌باشد در این نسخه تعداد گرادیان‌های مرتب شده در حافظه محدود شده است. بنابراین این الگوریتم ماتریس هسیان کامل را محاسبه نمی‌کند زیرا از لحاظ پردازشی بسیار سنگین است.

الگوریتم L-BFGS معکوس ماتریسی هسیان را تقریب می‌زند تا جستجوی برای تنظیم وزن‌ها را در محدوده‌ای مناسب از فضای پارامترها انجام دهد. با وجود اینکه BFGS کل ماتریس معکوس n×n را ذخیره می‌کند ولی L-BFGS فقط بخشی از ماتریس را که حاوی بردارهای محدوده تقریب زده شده است را ذخیره می‌کند. L-BFGS سریعتر است زیرا فقط از اطلاعات مرتبه دوم تقریب زده شده، استفاده می‌کند. L-BFGS و conjugate gradient در عمل از روش‌های SGD سریعتر و پایدارتر هستند. توجه داشته باشید که L-BFGS به صورت عمومی در شبکه‌های عمیق استفاده نمی‌شود.

گرادیان همیوغ[1]:

این روش خط جستجو را برپایه اطلاعات conjugacy هدایت می‌کند. تمرکز در این روش بر کمینه کردن نرم conjugate L2 است. اگر در روش gradient descent جستجو به صورت خطی انجام شود، بسیار مشابه conjugate gradient خواهد بود. با این تفاوت که هر گام متوالی در فرایند جستجوی خطی و با توجه به جهت جستجو با یکدیگر conjugate باشند.

Hessian-free:

این روش مرتبط با روش نیوتن است با این تفاوت که تابع درجه دوم را بهتر کمینه می‌کند.

۲-۱-٦- ابرپارامترها

ابرپارامترها مانند سایر پیکربندی‌ها توسط کاربر انتخاب می‌گردد و کارایی شبکه را تحت تاثیر قرار خواهد داد. ابرپارامترها در دسته‌های زیر قرار می‌گیرند:

- اندازه لایه
- Magnitude (momentum, learning rate)
- Regularization (dropout, drop connect, L1, L2)
- فعال‌سازی‌ها (و خانواده توابع فعال‌سازی)
- استراتژی وزندهی اولیه
- توابع فاصله از هدف
- تطیمات epochها در طول آموزش (mini-batch size)
- نحوه نرمال‌سازی برای داده‌های ورودی (برداری کردن)

توجه داشته باشید که بعضی از ابرپارامترها فقط بعضی اوقات اعمال می‌گردند. ضمناً تغییر یک ابرپارامتر ممکن است بهترین تنظیم برای سایر ابرپارامترها را تغییر دهد و بعضی از پارامترها ناسازگار با سایر پارامترها هستن (مثل Adagrad و momentum).

۲-۱-٦-۱- اندازه لایه

[1] Conjugate gradient

DOI: 10.30503/nripress.2020.321

منظور از اندازه لایه، تعداد نرون‌ها در یک لایه است. لایه‌های ورودی و خروجی به سادگی قابل ایجاد هستند زیرا مستقیما در از روی ورودی و خروجی‌های مدل به دست می‌آیند. برای لایه‌های پنهان می‌توان هر تعدادی از نورون را انتخاب کرد، هیچ قاعده‌ای برای آن وجود ندارد. تنها می‌توان گفت تعداد نورون‌های موجود در لایه پنهان ارتباط مستقیمی با پیچیدگی مدل دارد. این موضوع ممکن است ما را وادار کند که تعداد زیادی نرون را برای شروع انتخاب کنیم ولی این انتخاب هزینه زیادی دارد.

تعداد ارتباطات بین لایه‌ها در شبکه‌های عمیق می‌تواند متفاوت باشد ولی وزن‌ها در فرایند آموزش شبکه باید بدست آیند. هرچه تعداد پارامترهای مدل بیشتر باشد تلاش لازم برای آموزش شبکه و در نتیجه هزینه پردازش یا زمان لازم برای همگرایی مدل بیشتر خواهد بود.

Magnitude -۲-۶-۱-۲

این دسته از ابر پارامترها شامل گرادیان[1]، اندازه گام[2] و شتاب[3] می‌باشد.

Learning rate:

در یادگیری ماشینی به معنی سرعت تغییر بردار پارامترها در هنگام حرکت در فضای جستجو است. اگر سرعت تغییر خیلی زیاد باشد می‌توانیم خیلی سریع به هدف می‌رسیم ولی گام‌های بزرگ ممکن است باعث شود از روی بهترین جواب عبور کنیم و جوابی کمی ضعیفتر را انتخاب کنیم و مشکل دیگری که ممکن است ایجاد کند عدم پایداری آموزش و درنتیجه عدم همگرایی در طول زمان می‌باشد.

اگر نرخ یادگیری خیلی کوچک انتخاب شود ممکن است در زمان مورد نظر ما به جواب نرسد و زمان یادگیری بسیار طولانی گردد. تنظیم نرخ یادگیری ریسکی است زیرا هم به نوع دیتاست بستگی دارد و هم به سایر ابرپارامترها در نتیجه سربار زیادی را برای پیدا کردن ابرپارامترهای درست تحمیل می‌کند.

امکان تنظیم نرخ یادگیری به طوری که در طول زمان و به طور منظم کاهش یابد نیز وجود دارد.

توجه داشته باشید که نرخ یادگیری یکی از ابرپارامترهای کلیدی شبکه‌های عصبی می‌باشد.

Nesterov's momentum

نسخه vanilla (به عنوان ساده ترین نسخه) از SGD[4] مستقیما از گرادیان استفاده می‌کند که مقدار آن برای هر پارامتر تقریبا صفر است و این یک مسئله اساسی می‌باشد. این باعث می‌گردد تا SGD گام‌های بسیار کوچکی بردارد و زمانی که گرادیان بزگ است گام‌های بزرگی برمی‌دارد. در جهت کاهش اثر این موضوع از تکنیک‌های زیر استفاده می‌گردد.

- Nesterov's momentum
- RMSProp
- Adam
- AdaDelta

توجه داشته باشید که در بعضی از چارچوب‌های نرم‌افزای از تکنیک‌ها به عنوان روش updater نام برده می‌شود.

با افزایش momentum می‌توان سرعت آموزش را افزایش داد ولی شانس به دست آوردن خطای کمینه در تابع فاصله از هدف کاهش می‌یابد زیرا ممکن است پارامترهای بهینه نادیده گرفته شود و از روی آن‌ها عبور کنیم. momentum فاکتوری است با مقدار بین ۰.۰ تا ۱.۰ که روی نرخ تغییر وزن‌ها در طول زمان اعمال می‌گردد. معمولا مقدار آن بین ۰.۹ تا ۰.۹۹ تغییر می‌کند.

AdaGrad:

[1] Gradient
[2] Step Size
[3] Momentum
[4] Stochastic Gradient Descent

DOI: 10.30503/nripress.2020.321

این تکنیک به عنوان یک جستجوی اضافه توسعه داده شد تا نرخ یادگیری درست را بدست آورد که به صورت تطبیق پذیر از روش subgradient برای کنترل پویای نرخ آموزش در الگوریتم بهینه‌یابی استفاده می‌کند. AdaGrad به صورت یکنواخت نرخ آموزش را کاهش داده و هرگز آن را به بالاتر از مقدار تنظیم شده‌ی اولیه نمی‌برد. AdaGrad ریشه دوم مجموع مربعات تعدادی از گرادیان‌های محاسبه شده است. در ابتدا سرعت آموزش را افزایش می‌دهد و سپس در هنگام حرکت به سمت همگرایی به طور مناسبی آهسته می‌شود تا فرایند آموزش هموارتری را موجب گردد [16].

RMSProp:

یک روش بسیار مؤثر برای تعیین نرخ آموزش به صورت تطبیقی است. این روش منتشر شده، توسط Geoff Hinton در یکی از کلاس‌های آن در Coursera ارائه شده است. جزئیات روش در [17] آمده است.

AdaDelta:

یک نسخه از AdaGrad است که به جای انباشتن سابقه تغییرات نرخ آموزش فقط تعدادی از تازه‌ترین تغییرات را نگه می‌دارد [18].

ADAM

یک تکنیک بروزرسانی تطبیق پذیر نرخ آموزش که توسط دانشگاه تورنتو ارائه شده است. در این روش نرخ آموزش با استفاده از momentهای اول و دوم گرادیان‌های تخمین زده می‌شود.

۲-۱-۶-۳- Regularization

Regularization سنجه‌ای است که در روش جلوگیری از overfitting استفاده می‌گردد. overfitting زمانی اتفاق می‌افتد که مجموعه آموزش را به خوب توصیف می‌کند ولی نمی‌توان برای داده‌های جدید عمومیت ببخشد. مدل overfit شده قابلیت پیش‌بینی برای داده‌هایی که هنوز ندیده است را ندارد.

انجام Regularization به تصحیح گرادیان به نحوی کمک می‌کند تا در جهتی که منجر به overfitting می‌شود گام برندارد. Regularization شامل موارد زیر است:

- Dropout
- DropConnect
- L1 penalty
- L2 penalty

Dropout و DropConnect بخش‌هایی از ورودی به هر لایه را خاموش می‌کنند، به طوری که سایر بخش‌ها شبکه یاد بگیرد. Regularization با اضافه کردن یک ترم اضافه به گرادیان معمولی محاسبه می‌گردد.

Dropout:

مکانیزمی برای بهبود فرایند آموزش شبکه عصبی با حذف کردن تدریجی واحدهای مخفی (نرون های یا واحدهای موجود در لایه پنهان) می‌باشد. همچنین باعث افزایش سرعت آموزش شبکه می‌شود. Dropout تعدادی از نورون‌ها را به صورت تصادفی حذف می‌کند بنابراین نورون‌های حذف شده در فرایند حرکت به سمت جلو و پس انتشار[1] شرکت نخواهند کرد [19]. Dropout به نوعی میانگین چندین مدل می‌باشد. اگر ضریب dropout را برابر ۰.۵ درنظر بگیریم دقیقا یک مدل میانگین خواهیم داشت. انتخاب تصادفی Dropout، درواقع همان انتخاب تصادفی بین 2^N معماری ممکن است که در آن N تعداد پارامترهای قابل تنظیم است.

DropConnect:

[1] Backpropagation

DOI: 10.30503/nripress.2020.321

همانند Dropout می‌باشد با این تفاوت که به جای انتخاب واحدهای پنهان (نورون‌ها) اتصالات بین دو نورون را انتخاب می‌کند [19].

:L1

روش‌های جریمه کردن L1 و L2 جلوی بزرگ شدن فضای پارامتر شبکه عصبی در یک جهت را می‌گیرد و وزن‌های بسیار بزرگ را کوچک می‌کنند.

روش L1 در مورد داده‌هایی که کم‌پشت نیستند غیر کارآمد است و خروجی‌های کم پشت دارد و همچنین شامل روش انتخاب ویژگی از پیش ساخته است (یعنی ویژگی‌هایی که به هر طریقی استخراج شده به شبکه داده می‌شود و شبکه با روشی مشخص آن‌ها را انتخاب می‌کند). L1 مقدار واقعی وزن‌ها را بجای مربع آن‌ها ضرب می‌کند. این روش تعداد زیادی وزن را صفر می‌کند و عده کمی را اجازه می‌دهد تا بزرگ شوند. البته این موضوع باعث می‌گردد تا وزن‌ها آسان‌تر تفسیر گردند.

:L2

L2 از لحاظ پردازشی به صرفه است زیرا دارای روش حل تحلیلی بوده و خروجی آن کم‌پشت نیست، ولی انتخاب ویژگی را به صورت اتوماتیک انجام نمی‌دهد. L2 یک ابرپارامتر ساده و معمول است که یک ترم را به تابع هدف اضافه کرده تا مربعات وزن‌ها را کاهش دهد. نصف مربع وزن‌ها با ضریبی به نام weight-cost ضرب می‌شود. L2 عمومیت‌بخشی مدل را بهبود می‌دهد همچنین خروجی هموارتری در برابر تغییرات ورودی بدست می‌دهد و وزن‌های بلا استفاده را حذف می‌کند.

۲-۱-٦-٤- Mini-Batching

با استفاده از Mini-Batching بیشتر از یک بردار ورودی را برای آموزش به شبکه اعمال می‌کنیم (یک گروه از بردارها) تا سیستم آموش داده شود. این باعث بهبود کارایی در استفاده از سخت افزار می‌گردد. همچنین امکان محاسبه عملیات جبر خطی معین به سبک برداری شده را فراهم می‌کند (بویژه ضرب ماتریسی در ماتریسی). بنابراین امکان ارسال محاسبات برداری شده به GPU نیز فراهم می‌گردد [20].

۲-۱-٦-۵- خلاصه

تعداد زیادی از ابرپارامترها بین شبکه‌های عصبی فیدفوروارد و شبکه عمیق مشترک است که در اینجا بخشی را که به طور عمده مربوط به شبکه عمیق است بررسی کردیم.

۲-۲- بلوک‌های ساختمان شبکه عمیق

ساخت شبکه عمیق فراتر از یک شبکه فیدفوروارد اولیه چند لایه است. در بعضی موارد ترکیبی از شبکه‌های کوچکتر به عنوان بلوک ساختمانی شبکه عمیق در ساختن آن به کار می‌روند ولی در موارد دیگر یک دسته از لایه‌های تخصصی به کار می‌روند. در ادامه تعدادی مهم از بلوک‌های ساختمان شبکه عمیق آمده است.

- شبکه‌های فیدفوروارد چند لایه
- RBM ها
- Autoencoder ها

شبکه‌های فیدفوروارد ساده‌ترین نوع شبکه‌های عصبی هستند. RBM ها و Autoencoder ها هر دو نیاز به گام‌های بیشتری مرتبط با لایه در آموزش دارند. هر دو مورد برای پیش‌آموزش در شبکه‌های عمیق بزرگ به کار می‌روند. در طول زمان روش‌های بهتر بهینه‌یابی، توابع فعال‌سازی بهتر و روش‌های وزن‌دهی اولیه بهتر اهمیت شبکه‌های عمیق برپایه پیش‌آموزش را

DOI: 10.30503/nripress.2020.321

نشان دادند. در مواردی که تعداد زیادی داده برچسب نخورده داریم درحالی که تعداد داده‌های برچسب خورده بسیار کم است، این موضوع اهمیت بیشتری خواهد یافت. پیش آموزش سربار پردازشی بیشتری را همراه خواهد داشت. پیش آموزش مبتنی بر لایه کار خود را با آموزش غیر نظارتی اولین لایه با استفاده از داده‌های ورودی شروع می‌کند. به این ترتیب وزن‌های اولین لایه از شبکه اصلی را بدست می‌آوریم. این فرایند به طور پیش رونده‌ای انجام می‌شود و خروجی لایه قبل به عنوان ورودی در آموزش لایه بعد استفاده می‌گردد. این فرایند منجر به بدست آوردن مقادیر اولیه مناسب برای پارامتر شبکه خواهد شد [21].

RBM ها احتمال را مدل کرده و در استخراج ویژگی عالی هستند. هر واحد RBM همانند یک شبکه فیدفوروارد است که به جای یک بایاس، از دو بایاس استفاده می‌کند.

Autoencoder ها همان شبکه‌های فیدفوروارد هستند که دارای بایاس اضافه برای محاسبه خطا در ساخت مجدد ورودی هستند. بعد از آموزش، Autoencoder را می‌توان همانند یک شبکه فیدفوروارد برای ورود به تابع فعال‌سازی استفاده کرد. این یک شکل از استخراج ویژگی غیر نظارتی است که برای وزن‌های یادگیری فقط از داده‌های ورودی برچسب خورده به جای فرایند پس انتشار[1] استفاده می‌کند. شبکه عمیق هریک از دو مورد RBM یا Autoencoder را می‌تواند به عنوان بلوک ساختمان معماری خود برای شبکه‌های بزرگ استفاده کند. استفاده از هر دو مورد در یک شبکه به ندرت اتفاق می‌افتد.

RBM -۱-۲-۲

RBMها برای مقاصد زیر استفاده می‌گردند.

- استخراج ویژگی
- کاهش ابعاد[2]

کلمه «محدود شده» در «ماشین محدود شده بولتزمن (RBM)» به این دلیل است که ارتباط بین نودها در یک لایه ممنوع است. Geoff Hinton پیشگام یادگیری عمیق، ماشین بولتزمن عمومی را به این صورت توصیف می‌کند [22]: *یک شبکه از اتصالات متقارن دارای واحدهایی شبیه به نرون که تصمیماتی دارای تغییر را در مورد اینکه روشن یا خاموش باشند می‌گیرند.*

RBM ها درواقع یک نوع Autoencoder هستند و در شبکه‌هایی مثل DBN[3] بسیار بزرگ برای پیش آموزش استفاده می‌گردند.

۲-۲-۱-۱- طرح بندی شبکه

RBM دارای ۵ بخش اصلی می‌باشد.

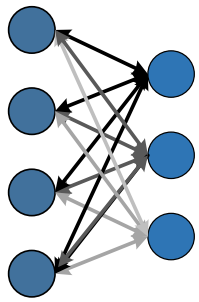
۱. واحدهای آشکار[4]
۲. واحدهای پنهان[5]
۳. وزن‌ها
۴. واحدهای بایاس آشکار[6]

[1] Backpropagation
[2] Dimensionality Reduction
[3] Deep Belief Network
[4] Visible units
[5] Hidden units
[6] Visible bias units

DOI: 10.30503/nripress.2020.321

۵. واحدهای بایاس مخفی[1]

همانطور که در شکل ۲–۱ مشخص است، یک واحد استاندارد RBM دارای یک لایه پنهان و یک لایه آشکار است. همچنین خطوط نشان دهنده وزن‌های شبکه می‌باشد.



لأن لقّهم لف لأن لا يعضايه

شکل ۲–۱ : شبکه RBM

در RBM هر واحد آشکار به هر واحد پنهان همراه با یک وزن متصل است و واحدهای موجود در یک لایه هیچ اتصالی به هم ندارند.

لایه‌های پنهان و آشکار:

واحدهای لایه پنهان ویژگی را تشخیص داده و با استفاده از داده‌های ورودی یاد می‌گیرند. نودهای موجود در هر لایه در واقع همان نودهای موجود در شبکه فیدفوروارد چند لایه هستند. واحدهای موجود در لایه آشکار مسئول دریافت بردارهای ورودی برای آموزش هستند. هر لایه دارای یک واحد بایاس است که وضعیت آن همیشه on است.

هر نود محاسبه‌ای را برپایه ورودی انجام داده و خروجی یک تصمیم تغییر پذیر[2] است. این موضوع در هر دو حالت عبور داده از طریق تابع فعال‌سازی یا عدم عبور آن از تابع فعال سازی برقرار است. وزن‌های اولیه به صورت تصادفی تولید می‌گردند.

وزن ها و اتصالات:

تمام اتصالات از نوع آشکار–پنهان هستند و هیچ نوعی آشکار–آشکار یا پنهان–پنهان نیست. یال‌ها در شکل ۲–۱ نشان‌دهنده‌ی مسیرهایی است که از آن‌ها سیگنال عبور می‌کند و نودها واحدهای تصمیم گیری هستند. آن‌ها تصمیم می‌گیرند که روشن باشند یا خاموش که روشن به معنی عبور سیگنال از آن‌ها در شبکه و خاموش به معنی عدم عبور سیگنال است.

به طور معمول داده‌ی عبور کرده از نود ارزشمند است. یعنی حاوی اطلاعاتی است که به شبکه کمک می‌کند تا تصمیم بگیرد. خاموش به این معنی است که داده ورودی یک نویز بی ربط است. از طریق آموزش، شبکه ارتباط بین برچسب‌ها و سیگنال‌ها را متوجه می‌شود. از طریق آموزش، شبکه کلاس‌بندی دقیق داده‌های ورودی را یاد می‌گیرد.

بایاس‌ها[3]:

یک دسته از وزن‌های بایاس وجود دارد که ارتباط بین واحد بایاس هر لایه را با واحدهای درون آن لایه برقرار می‌کند. بایاس به شبکه کمک می‌کند تا داده‌ها را بهتر ارزیابی یا مدل کند مخصوصا زمانی که یک نود ورودی همیشه on یا همیشه off است.

Training -۲-۱-۲-۲

[1] Hidden bias units
[2] Stochastic
[3] Biases

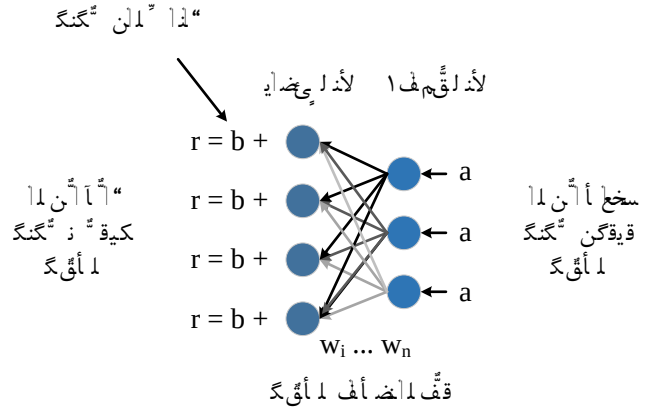
DOI: 10.30503/nripress.2020.321

تکنیک پیش آموزش با استفاده از RBMها به معنی آموزش شبکه برای ساخت مجدد داده اصلی ورودی با استفاده از تعدادی نمونه محدود می‌باشد. RPM ها یاد می‌گیرند که دیتاست ورودی را بازسازی کنند. RBM ها گرادیان را با استفاده از الگوریتمی به نام «واگرایی تمایز[1]» محاسبه می‌کنند. این الگوریتم برای نمونه گیری در پیش آموزش شبکه استفاده می‌شود. این الگوریتم واگرایی KL (دلتای بین توزیع داده واقعی و داده حدسی) را کمینه می‌کند، این کار با نمونه گیری k گام از یک زنجیره مارکوو[2] انجام می‌شود تا یک حدس را محاسبه کند [23].

Reconstruction -۳-۱-۲-۲

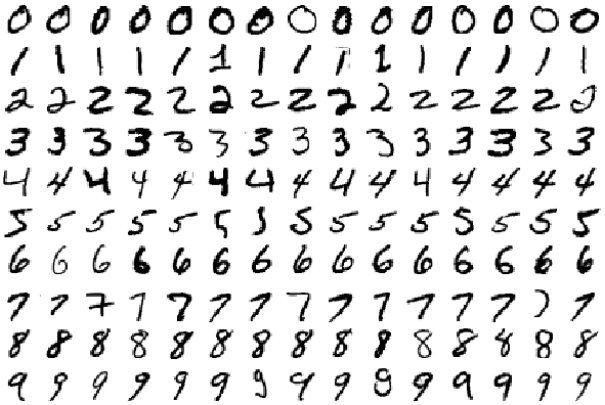
شبکه‌های عمیق با پیش آموزش غیر نظارتی (RBM ها و Autoencoder ها) از طریق بازسازی داده یک فرایند مهندسی ویژگی را روی داده‌های برچسب نخورده انجام می‌دهند. در شبکه‌هایی مثل DBN وزن‌های بدست آمده در پیش آموزش به عنوان وزن‌های اولیه در آموزش شبکه استفاده می‌گردد. بازسازی درواقع یک مسئله فاکتورگیری[3] ماتریسی است که تجزیه ماتریسی نیز نامیده می‌شود.

شکل ۲–۲ شبکه RBM در فرایند بازسازی را نشان می‌دهد.



شکل ۲–۲ : فرایند باز سازی در شبکه RBM

به صورت تصویری می‌توان بازسازی توسط شبکه RBM را نشان داد. برای این کار می‌توان از MNIST[4] استفاده کرد. MNIST یک دیتاست مربوط به موسسه ملی تکنولوژی و استاندارد در آمریکا است. این دیتاست یک مجموعه از اعداد به صورت دست نوشته است که نمونه آن در شکل ۲–۳ نشان داده شده است.

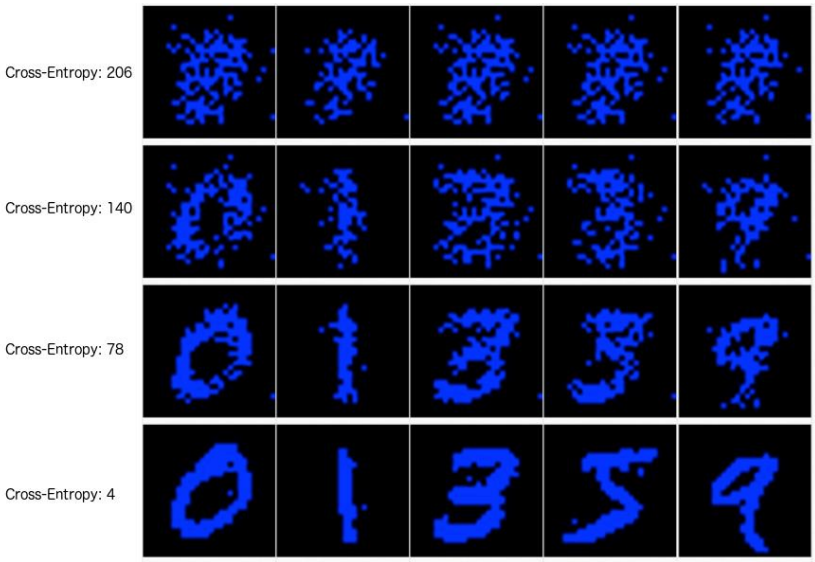


[1] Contrastive Divergence
[2] Markov
[3] Factorization
[4] http://yann.lecun.com/exdb/mnist/

DOI: 10.30503/nripress.2020.321

شکل ۲–۳ : نمونه از اعداد دست نوشته در دیتاست MNIST **(تعدادی از تصاویر جدا شده از این دیتاست و تبدیل شده به فرمت قابل نمایش** [24]**)**

مجموعه آموزش MNIST دارای ۶۰۰۰۰ رکورد و مجموعه تست آن شامل ۱۰۰۰۰ رکورد است. اگر از RBM برای یادگیری MNIST استفاده کنیم باید از مجموعه آموزش آن نمونه گیری کنیم [25]. شکل ۲–۴ فرایند ساخت مجدد اعداد MNIST را به صورت رو به جلو با استفاده از یک RBM نشان می‌دهد.



شکل ۲–٤ : ساخت اعداد MNIST **با استفاده از** RBM [26]

تابع هدف در این مورد به صورت معمول cross-entropy یا واگرایی KL می‌باشد. آنتروپی بر اساس تئوری اطلاعات به معنی عدم قطعیت[1] است. بنابراین cross-entropy به معنی مقایسه بین عدم قطعیت (آنتروپی) دو توزیع مختلف است. برای مثال یک توزیع نرمال با منحنی پهن به معنی عدم قطعیت یا آنتروپی بالا است.

۲-۲-۱-٤- کاربردهای دیگر RBMها

جاهای دیگری که ممکن است RBM ها را مشاهده کنید.

- کاهش ابعاد داده
- کلاس‌بندی
- رگرسیون
- فیلترینگ همکارانه[2]
- مدل کردن موضوع[3]

[1] Uncertainty
[2] Collaborative Filtering
[3] Topic Modeling

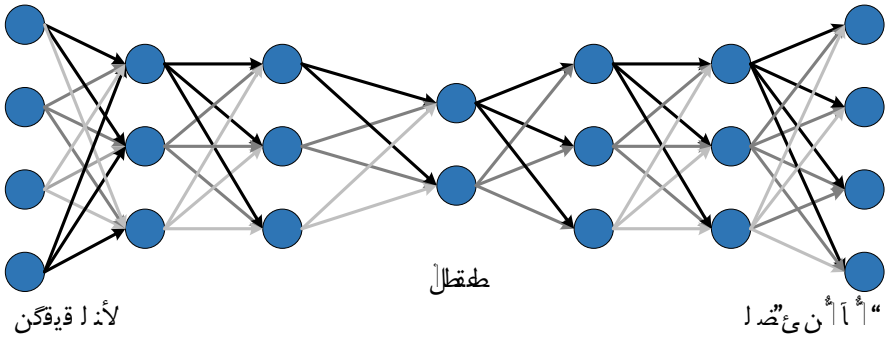
DOI: 10.30503/nripress.2020.321

۲-۲-۲- Autoencoders

از Autoencoderها برای یادگیری داده‌های فشرده شده استفاده می‌گردد. معمولا کاربرد آن کاهش ابعاد دیتاست است. خروجی شبکه Autoencoder داده‌های بازسازی شده به روش بسیار بهینه است.

۲-۲-۲-۱- تشابه با پرسپترون چند لایه

مشترکات زیادی با شبکه‌های پرسپترون چند لایه وجود دارد، همانند پرسپترون دارای لایه ورودی، لایه پنهان و لایه خروجی است. تفاوت مشخص با پرسپترون معمولی در داشتن تعداد واحدهای ورودی برابر با تعداد واحدهای خروجی می‌باشد. شکل ۲–۵ یک نمونه از شبکه Autoencoder را نشان می‌دهد.



شکل ۲–۵ : معماری شبکه Autoencoder

غیر از تفاوت در لایه خروجی مواردی دیگری نیز وجود دارد که در ادامه آمده است.

۲-۲-۲-۲- ویژگی‌های Autoencoder ها

دو ویژگی متمایز کننده Autoencoder شامل موارد زیر است.

- از داده‌های بدون برچسب در یک یادگیری غیر نظارتی استفاده می‌کند.
- یک شکل فشرده از داده‌های ورودی را ایجاد می‌کنند.

۲-۲-۲-۳- آموزش Autoencoder

Autoencoderها از پس انتشار[1] برای بروزرسانی وزن‌های شبکه استفاده می‌کنند. تفاوت عمده بین کلاس عمومی Autoencoder ها و RBM ها در نحوه محاسبه گرادیان می‌باشد.

۲-۲-۲-٤- نسخه‌های عمومی Autoencoder ها

دو نسخه مهم Autoencoder ها شامل موارد زیر است.

فشرده سازها:

معماری این دسته در شکل ۲–۵ نشان داده شده است که ویژگی اصلی آن وجود گلوگاه در شبکه است.

حذف کننده‌های نویز:

[1] Backpropagation

DOI: 10.30503/nripress.2020.321

در این نوع، ورودی تخریب شده از داده به Autoencoder اعمال می‌شود. تخریب شامل حذف تصادفی بعضی از ویژگی‌ها است. سپس شبکه یاد می‌گیرد تا داده‌ی کامل را در خروجی ارائه کند [27].

۲-۲-۲-۵- سایر کابردهای Autoencoder ها

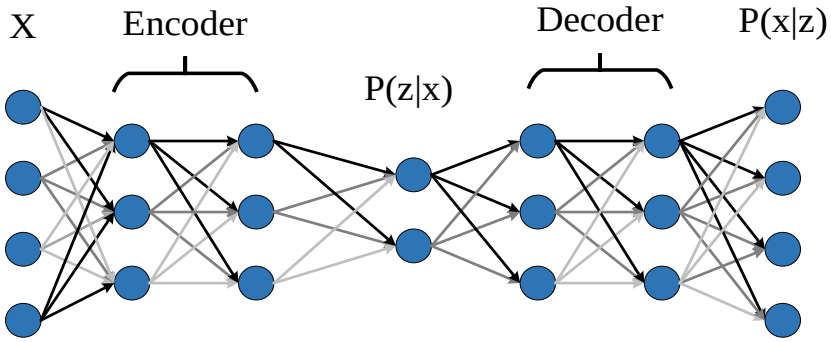
موضوع قابل توجه در مورد Autoencoder ها تفاوت بین ورودی و خروجی است. اگر به یک شبکه فقط یاد بدهیم که داده‌های ورودی را یاد بگیرد بنابراین شبکه قادر خواهد بود که تفاوت بین داده‌های معمول و داده‌های غیر متعارف [1] را تشخیص دهد.

Anomaly Detector

Autoencoder به صورت عمومی در سیستم‌هایی که شکل کلی داده‌ها مشخص است ولی تشخیص داده‌های غیر متعارف مشکل است، استفاده می‌گردد. Autoencoder ها در افزایش قدرت سیستم‌های تشخیصِ داده غیر متعارف بسیار مناسب هستند.

۲-۲-۳- Variational Autoencoders

این نوع جدیدی از Autoencoder ها است با مخفف VAE نمایش داده می‌شود. همانند حذف کننده‌های نویز و فشرده‌سازها هستند با این تفاوت که آموزش آن‌ها برای ساختن مجدد داده‌های ورودی به روش غیر نظارتی انجام می‌گیرد [28]. ساختار یک VAE در شکل ۲–۶ نشان داده شده است.



شکل ۲–٦ : معماری شبکه VAE

مکانیزم آموزش شبکه در VAE کاملا با دو نوع قبل متفاوت است. در دو نوع قبلی توابع فعال‌سازی همانند شبکه‌های عصبی معمولی است ولی شبکه‌های VAE از رویکرد احتمالاتی استفاده می‌کند.

در مدل VAE فرض بر این است که x به دو صورت تولید می‌شود.

۱. مقدار $z^i \sim p(z)$ با استفاده از توزیع اولیه تولید می‌شود.

۲. هر نمونه داده بر طبق بعضی توزیع‌های شرطی $x^i \sim p(x \mid z)$ تولید می‌گردد.

البته ما مقدار واقعی z را نمی‌دانیم و استنتاج $p(z \mid x)$ به صورت دقیق غیر قابل دنبال کردن است. برای به انجام رساندن آن اجازه می‌دهیم که هر دو توزیع $p(z \mid x)$ و $p(x \mid z)$ با استفاده از شبکه عصبی تخمین زده شود یعنی به ترتیب هم encoder و هم decoder. برای مثال اگر $p(z \mid x)$ یک توزیع گوسی باشد توابع فعال‌سازی عبور داده از شبکه پارامترهای توزیع گوسی یعنی μ و σ^2 فراهم می‌کند.

[1] Anomalous

DOI: 10.30503/nripress.2020.321

به همین صورت پارامترهای توزیع گوسی $p(x|z)$ در عبور داده از شبکه decoder فراهم می‌شود. توجه داشته باشید پارامترها نباید با پارمترهای آموزش شبکه اشتباه گرفته شود. آن‌ها فقط توابع فعال‌سازی هستند که برای مثال می‌تواند برای تعیین مقادیر میانه و واریانس یک توزیع گوسی یا تعیین مقدار میانه برای توزیع برنولی استفاده شود.

به طور کلی شبکه با استفاده از پس انتشار[1] آموزش داده می‌شود تا حد پایین احتمال حاشیه‌ای[2] داده‌های آموزش $\log p\left(x^1,\ldots,x^N\right)$ را بیشینه کند. همچنین VAE برای سری‌های زمانی نیز توسعه داده شده است [29].

[1] Backpropagation
[2] Marginal likelihood

DOI: 10.30503/nripress.2020.321

فصل ۳- معماری‌های عمده در شبکه عمیق

DOI: 10.30503/nripress.2020.321

مقدمه

در این بخش معماری‌های اصلی شبکه‌های عمیق و نحوه استفاده از شبکه‌های کوچکتر برای ساخت آن‌ها را توضیح خواهیم داد. شبکه‌های عمیق دارای چهار معماری اصلی به صورت زیر هستند.

- شبکه پیش آموزش غیر نظارتی (UPN)
  - شبکه باور عمیق (DBN)
  - شبکه های مقایسه‌ای مولد (GAN)
- شبکه عصبی کانولوشنی (پیچشی) (CNN)
- شبکه عصبی بازرخداد (RNN[1])
  - حافظه کوتاه مدت گذشته (LSTM)/ واحدهای بازرخداد دروازه‌ای (GRU)
- شبکه عصبی بازگشتی (RNN[2])

دسته شبکه‌هایی که زیر هر معماری ذکر شدند انواع شبکه پرکاربرد از معماری مورد نظر هستند. در ادامه جزئیات هریک از این معماری‌ها بیان خواهد شد. تمرکز اصلی روی معماری‌های سطح بالا می‌باشد تا بتوان در کاربرد مناسب از آن استفاده کرد.

۳-۱- شبکه پیش آموزش غیر نظارتی (UPN)

سه نوع شبکه پر کاربرد در این معماری شامل انواع زیر است.

- Autoencoder ها
- شبکه های باور عمیق (DBN[3])
- شبکه های مقایسه‌ای مولد (GAN[4])

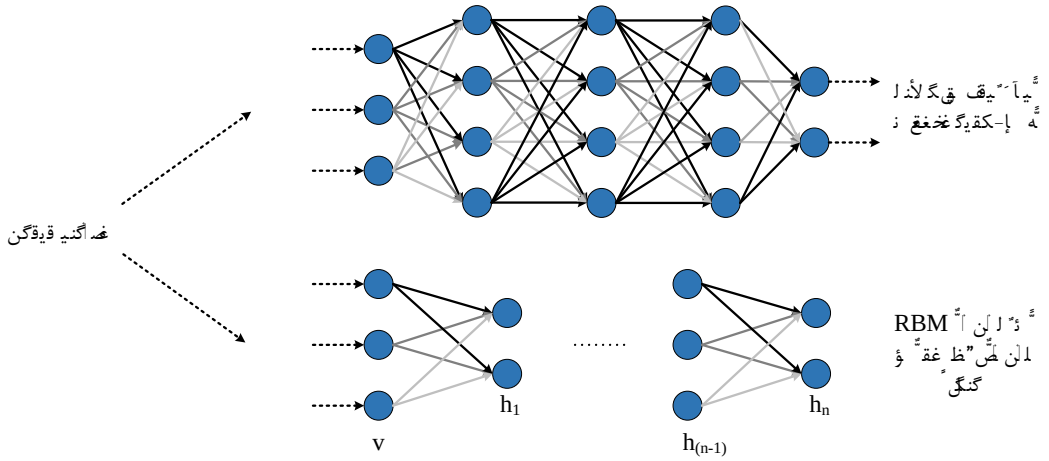
Autoencoder ها ساختارهای پایه در شبکه‌های عمیق هستند و معمولا به عنوان بخشی از یک شبکه بزرگتر استفاده می‌شوند. همچنین به عنوان یک شبکه مستقل نیز به کار می‌روند. Autoencoder ها در بخش قبل به صورت کامل پوشش داده شدند در ادامه شبکه‌های DBN و GAN را بررسی می‌کنیم.

۳-۱-۱- شبکه باور عمیق (DBN)

DBN ها از ترکیب لایه‌های ماشین بولتمن محدود شده (RBM) برای فاز پیش آموزش و سپس یک شبکه فیدفوروارد برای فاز تنظیم دقیق ایجاد گردیده‌اند. شکل ۳—۱ معماری این شبکه را نشان می‌دهد.

[1] Recurrent Neural Networks
[2] Recursive Neural Networks
[3] Deep Belief Networks
[4] Generative Adversarial Networks

DOI: 10.30503/nripress.2020.321



شکل ۳–۱ : معاری DBN

۳-۱-۱-۱- استخراج ویژگی با استفاده از لایه‌های RBM

با استفاده از RBM ها ویژگی‌های سطح بالا از داده خام استخراج می‌گردد. در اینجا نیز RBM ها کار بازسازی داده ورودی را انجام می‌دهند. که برای این کار وزن‌ها و حالت‌های RBM ها در فرایند آموزش بدست می‌آید. تعدادی از لایه‌های RBM مسئول استخراج ویژگی‌ها سطح پایین و تعدادی مسئول استخراج ویژگی‌های سطح بالا است. این روش باعث یادگیری بهتر در شبکه عصبی می‌شود.

یادگیری ویژگی‌های سطح بالا به صورت اتوماتیک:

یادگیری این ویژگی‌ها به صورت غیر نظارتی فاز پیش آموزش در شبکه DBN است. هر لایه از RBM ویژگی‌هایی را یاد می‌گیرد که در نهایت این ویژگی‌های سطح بالا به روش غیر خطی به هم ترکیب می‌شوند. در شکل ۳–۲، شکل ۳–۳ و شکل ۳–۴ ساخت ویژگی توسط لایه‌های RBM را به صورت پیش رونده برای اعداد MNIST نشان می‌دهد.

شکل ۳–۲ : خروجی توابع فعال‌سازی در ابتدای آموزش شبکه [30]

شکل ۳–۳ : ظاهر شدن ویژگی‌ها خروجی توابع فعال‌سازی در هنگام آموزش شبکه [30]

شکل ۳–٤ : یادگیری ویژگی‌های مربوط به اعداد MNIST **به سمت انتهای آموزش شبکه** [30]

همانطور که در اشکال مشخص است RBM ها تکه‌هایی از اعداد را یاد می‌گیرند. این ویژگی‌های در لایه‌های سطح بالا با یکدیگر ترکیب شده و ویژگی‌های پیچیده تر و غیر خطی را می‌سازند. بنابراین سیستم قادر خواهد بود ویژگی‌های سطح بالا را از داده‌های خام استخراج کند.

DOI: 10.30503/nripress.2020.321

آغاز به کار شبکه فیدفوروارد:

لایه‌های ویژگی بدست آمده به عنوان وزن‌های آغازین در شبکه فیدفوروارد استفاده می‌گردد. این وزن دهی اولیه به الگوریتم آموزش کمک می‌کند تا پارامترهای شبکه عصبی را به ناحیه بهتری از فضای جستجوی پارامترها هدایت کند. این فرایند « فاز تنظیم دقیق» شبکه DBN نامیده می‌شود.

۳-۱-۱-۲- تنظیم دقیق DBN با استفاده از شبکه فیدفوروارد

در این فاز فرایند پس انتشار[1] با نرخ یادگیری پایین انجام می‌گیرد تا وزن‌ها به صورت دقیق برای مدل نهایی به دست آید. درواقع در این فاز، شبکه به صورت تخصصی روی موضوع مدنظر ما (مثل کلاس‌بندی) کار خواهد کرد.

لایه اول یاد می‌گیرد که داده‌ها را چگونه بازسازی کند. لایه‌های بعدی یاد می‌گیرند که چگونه توزیع احتمالاتی توابع فعال‌سازی لایه قبل را بازسازی کنند. لایه خروجی انجام هدف نهایی را به عهده دارد.

۳-۱-۱-۳- شبکه مقایسه‌ای مولد (GAN)

شبکه‌های GAN قابلیت خوبی را در سنتز کردن تصاویر جدید برپایه تصاویر آموزش دیده از خود نشان داده‌اند [31]. این مفهوم قابلیت گسترش به محدوده‌های دیگر مثل موارد زیر را دارد.

- صدا
- ویدئو [32]
- تولید تصویر بر اساس توصیف متنی [33]

GAN ها مثالی از یک شبکه غیر نظارتی هستند که دو مدل را به صورت موازی آموزش می‌دهند. نکته کلیدی در مورد GAN نحوه استفاده از پارامترهایی است که به صورت واضحی مشابهت دارند. درحالی که در شبکه‌های معمولی نسبت به حجم داده این تشابه‌ها خیلی کمتر است. شبکه به صورت کارامدی مجبور به نمایش داده‌های آموزش داده شده می‌گردد و این باعث می‌گردد تا در تولید داده‌هایی مشابه داده‌های ورودی کارآمد باشد.

۳-۱-۱-٤- آموزش مدل‌های مولد، یادگیری غیر نظارتی و GAN

اگر مجموعه بزرگی از تصاویر (مثل دیتاست ImageNet) را برای آموزش شبکه داشته باشیم، می‌توان یک مدل مولد ایجاد کرد تا با استفاده از آن تصویر تولید کنیم. تصاویر تولید شده با استفاده از این شبکه، نمونه‌های خروجی مدل ما هستند. تصاویر تولید شده توسط مدل مولد در GAN با استفاده از شبکه متمایز کننده[2] کلاس‌بندی می‌گردد.

شبکه متمایز کننده ورودی را به دو دسته تصاویر واقعی و تصاویر سنتز شده تقسیم می‌کند. پارامترها در این شبکه براساس تصاویر ورودی به نحوی بروز رسانی می‌گردند که تصاویر تولید شده قابل باور باشند. درواقع هدف تولید تصاویری است که توسط شبکه متمایز کننده از تصاویر واقعی قابل تشخیص نباشد.

یک مدل بهینه در GAN برای مدل کردن ImageNet دارای حدود ۱۰۰ میلیون پارامتر است. یک دیتاست ورودی در ImageNet با حجم معادل 200GB به پارامترهایی با حجم 100MB تبدیل می‌شود. فرایند آموزش در GAN تلاش می‌کند تا بهینه ترین راه ممکن برای نمایش ویژگی‌ها در داده‌ها را مانند گروه پیکسل‌های مشابه، لبه‌ها و الگوهای دیگر بیابد.

[1] Backpropagation
[2] Discriminator

DOI: 10.30503/nripress.2020.321

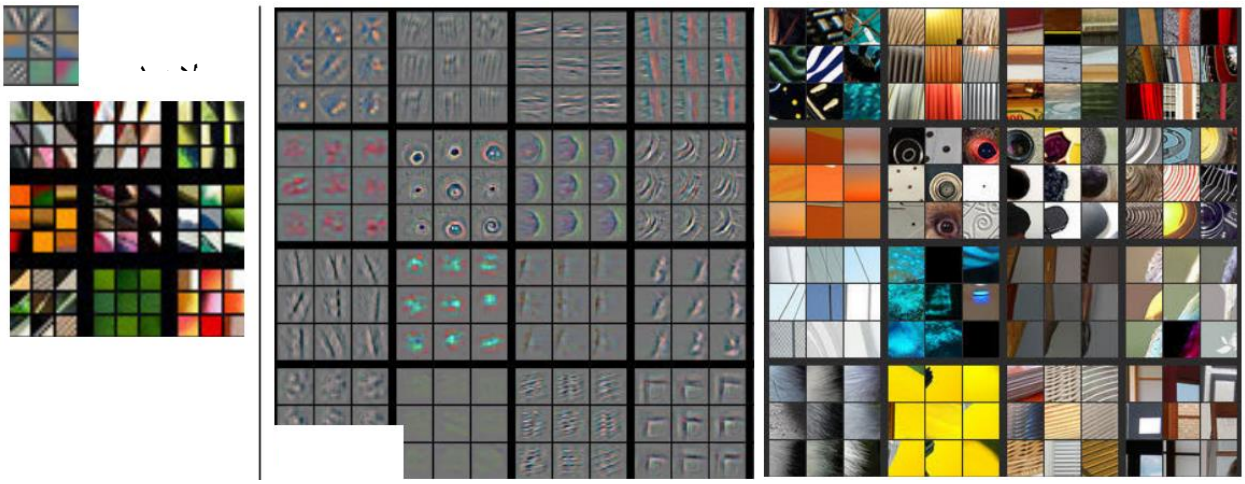
شبکه متمایز کننده:

شبکه متمایز کننده معمولا یک CNN می‌باشد. این شبکه متمایز کننده تصاویر را گرفته کلاس‌بندی انجام می‌دهد. گرادیان خروجی شبکه متمایز کننده با توجه به تصاویر سنتز شده‌ی ورودی چگونگی انجام تغییرات کوچک روی تصاویر سنتز شده به طوری که واقعی‌تر به نظر برسند را ارائه می‌کند.

شبکه مولد:

این شبکه با استفاده از لایه‌های خاصی به نام معکوس-کانلوشنی[1] تصاویر را تولید می‌کند. در طول آموزش شبکه با استفاده از فرایند پس انتشار روی هر دو شبکه، پارامترهای شبکه مولد نیز بروز می‌شود تا تصاویر واقعی‌تری تولید کند. هدف در این جا به روز رسانی شبکه مولد تا نقطه‌ای است که به اندازه کافی شبکه متمایز کننده از خروجی شبکه مولد به اشتباه بیوفتد و آن را واقعی درنظر بگیرد.

شبکه‌های معکوس-کانولوشنی وقتی تصاویر را مدل کردند ویژگی‌ها را به پیکسل‌ها نگاشت می‌کنند. این موضوع در شکل ۳–۵ نشان داده شده است که عکس عملکرد شبکه‌های کانلوشنی می‌باشد. این ویژگی شبکه‌های معکوس-کانلوشنی اجازه تولید تصاویر را به عنوان خروجی از شبکه عصبی می‌دهد. این شبکه‌ها نیز غیر نظارتی بوده و به سبک مبتنی برلایه آموزش داده می‌شوند مشابه آنچه در DBN وجود دارد. شبکه دارای چندین لایه معکوس-کانلوشنی پشته‌ای است که هر لایه بر اساس خروجی لایه قبل آموزش می‌بیند [34].



شکل ۳–۵ : عملکرد لایه‌های معکوس-کانلوشنی [34]

یک نسخه از GAN شبکه مقایسه‌ای مولد کانلوشنی عمیق (DCGAN[2]) که نمونه‌ای از خروجی این شبکه برای تصاویر یاد گرفته شده از اتاق خواب در شکل ۳–۶ نشان داده شده است.

[1] Deconvolutional
[2] Deep Convolutional Generative Adversarial Network

DOI: 10.30503/nripress.2020.321

شکل ۳-٦ : تصاویر تولید شده اتاق خواب با استفاده از DCGAN [35]

شبکه DCGAN اعداد تصادفی با توزیع یکنواخت را دریافت کرده و تصاویر را با استفاده از مدل یاد گرفته شده شبکه تولید می‌کند. وقتی عدد تصادفی ورودی تغییر می‌کند نوع تصاویر خروجی نیز عوض می‌شود.

۳-۱-۱-۵- GAN های شرطی

در شبکه‌های GAN شرطی از داده‌های برچسب خورده برای آموزش استفاده می‌گردد. بنابراین شبکه‌های مذکور به طور خاص می‌توانند داده‌های مربوط به یک کلاس را تولید کنند [36].

۳-۱-۱-٦- مقایسه GAN و VAE

تمرکز GAN روی تشخیص این است که آیا تصویر تولید شده مربوط توسط مدل ایجاد شده یا واقعی است. وقتی مدل شبکه متمایز کننده پیش بینی انجام می‌دهد درصورتی که تفاوتی بین توزیع داده‌های واقعی و داده‌های تولید شده باشد شبکه مولد پارامترهای خود را تصحیح می‌کند. بنابراین شبکه مولد روی پارامترهایی همگرا می‌شود که داده‌هایی با توزیع واقعی تولید کند به طوری که شبکه متمایز کننده قادر به تشخیص تفاوت نباشد.

با شبکه VAE مسئله‌ای مشابه با استفاده از مدل‌های گرافیکی تصادفی[1] حل می‌گردد تا ورودی به سبک غیر نظارتی بازسازی شود. VAE ها تلاش می‌کنند که حد پایینی احتمال لگاریتمی داده‌ها را بیشینه کنند تا تصاویر تولید شده بیشتر و بیشتر واقعی به نظر برسند.

تفاوت دیگر GAN و VAE در چگونگی تولید تصاویر است. در GAN پایه تصاویر بر اساس کدهای دلخواه تولید می‌گردند و راهی برای تولید تصویر با ویژگی‌های دلخواه وجود ندارد. برعکس VAE ها که دارای رویه encode/decode هستند که براساس آن می‌توان تصویر تولید شده را تصویر اصلی مقایسه کرد. اثر جانبی این موضوع ایجاد کد برای تولید هر نوع خاص از تصاویر است.

[1] Probabilistic Graphical Model

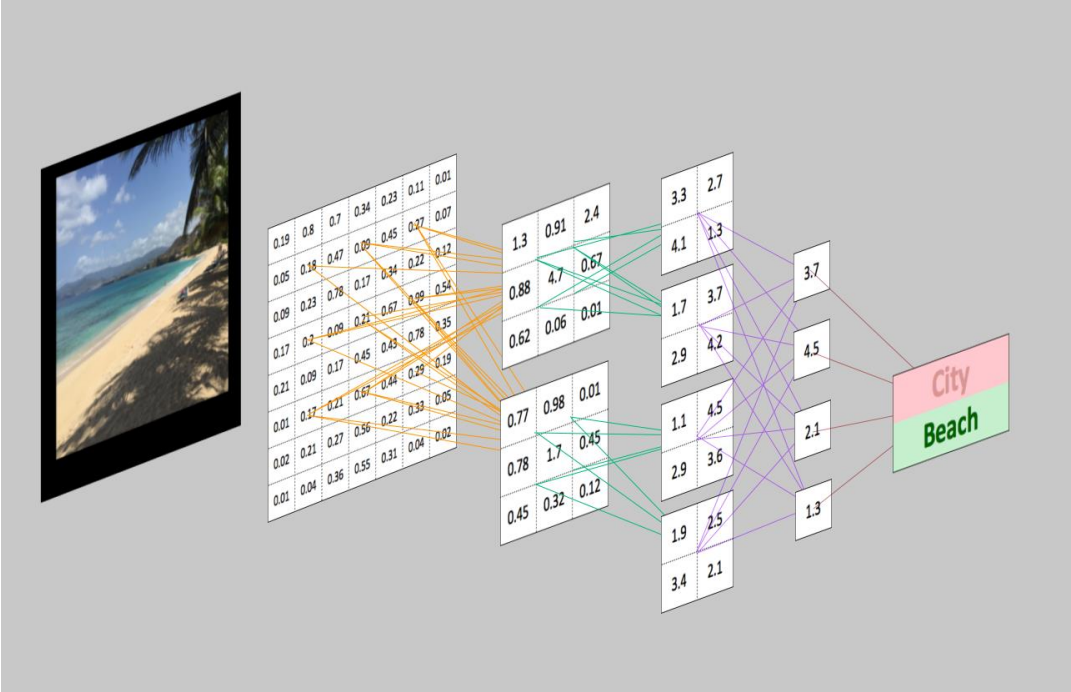
DOI: 10.30503/nripress.2020.321

دو نوع نویز متفاوت نیز در این شبکه‌ها وجود دارد. تصاویر تولید شده توسط VAE بعضی اوقات شفاف نیست. ازطرفی GAN ها تلاش می‌کنند تا استایل کلی تصاویر را استخراج کنند. بنابراین بعضی از تصاویر تولید شده دارای ترکیب غیر منطقی هستند (مثلا تصویر مربوط به سگ است ولی به صورت کلی درست به نظر نمی‌رسد).

۳-۲- شبکه عصبی کانولوشنی (CNN)

هدف از ایجاد CNN یادگیری ویژگی‌های سطح بالا در داده‌ی خام با استفاده از کانولوشن می‌باشد. این نوع از شبکه‌ها برای تشخیص اشیاء در تصاویر و کلاس‌بندی بدون تناقض و رقابتی بسیار مناسب هستند. این نوع از شبکه‌ها می‌توانند صورت، افراد، علائم جاده و جنبه‌های دیگر داده‌های بصری را تشخیص دهند. CNN ها با تحلیل متن توسط روش «تشخیص کاراکتر نوری» (OCR[1]) همپوشانی دارند، ولی برای تحلیل کلمات به عنوان واحدهای متنی مجزا مفید هستند [37]. همچنین این نوع از شبکه‌ها در تحلیل صدا نیز مفید هستند.

این کارامدی CNN باعث محبوبیت شبکه‌های عمیق شد. همانطور که در شکل ۳–۷ نشان داده شده، ساختن موقعیت و ویژگی به صورت تغییر ناپذیر در مقابل چرخش، از داده‌ی خام (یک تصویر پردازش نشده) عالی هستند.



شکل ۳–۷ : CNN **و بینایی کامپیوتری** [38]

CNN ها زمانی که ساختارهایی در داده‌های ورودی وجود داشته باشد نیز مفید هستند. برای مثال الگوهای تکراری در تصویر (عکس) یا صدا که پشت سر هم قرار دارند با یکدیگر به طور ویژه‌ای ارتباط دارند. همچنین CNN در ترجمه یا تولید زبان طبیعی [16] و تحلیل احساسات [39] نیز به کار می‌رود. کانولوشن یک مفهوم بسیار قدرتمند برای ساخت فضای ویژگی پایدار برپایه یک سیگنال می‌باشد.

[1] Optical Character Recognition

DOI: 10.30503/nripress.2020.321

۳-۲-۱- زمینه بیولوژیکی

زمینه بیولوژیکی شبکه‌های CNN کرتکس بینایی در حیوانات می‌باشد [40]. سلول‌های موجود در کرتکس بینایی به زیر نواحی کوچک ورودی حساس هستند که نواحی بینایی یا نواحی پذیرنده نامیده می‌شوند. این زیرنواحی کوچک به صورت موزاییکی کل میدان بینایی را پوشش می‌دهند. این سلول‌های برای پیدا کردن همبستگی‌های محلی قوی می‌توانند به کار گرفته شوند و به عنوان فیلتر محلی روی فضای ورودی عمل کنند. دو دسته سلول در این ناحیه از مغز وجود دارد. سلول‌های ساده زمانی که لبه‌ها را تشخیص می‌دهند فعال می‌گردد و سلول‌های پیچیده، در زمان تشخیص نواحی پذیرنده‌ی بزرگتر فعال می‌گردند و نسبت به موقعیت الگو حساس نیستند.

۳-۲-۲- درک مستقیم

مسئله اصلی در شبکه‌های فیدفورارد چند لایه برای کار با تصاویر عدم مقیاس پذیری آن‌ها برای داده ورودی تصویر است. برای مثال دیتاست CIFAR-10 را در نظر بگیرید تصاویر دارای ابعاد ۳۲ در ۳۲ پیکسل و ۳ کانال رنگی RGB هستند. این منجر به ایجاد ۳۰۷۲ وزن فقط در لایه ورودی می‌شود و به احتمال زیاد تعداد بیشتری نورون در لایه پنهان نیاز خواهد داشت همچنین در بسیار موارد تعداد لایه‌های پنهان بیشتر نیز خواهند بود.

حال یک تصویر معمولی که ۳۰۰ در ۳۰۰ پیکسل است و ۳ کانال رنگی RGB دارد را درنظر بگیرید. تعداد ۲۷۰۰۰۰ وزن در اتصالات هر نورون لایه داخلی خواهیم داشت. این موضوع نشان می‌دهد که به تغییر اندازه ورودی به چه سرعتی تعداد اتصالات و درنتیجه تعداد وزن‌ها در شبکه عصبی چند لایه رشد می‌کند. تغییر ساختار تصاویر نیاز به تغییر ساختار شبکه عصبی دارد تا بتوان ساختار بهینه برای آن پیدا کرد.

در CNN می‌توان نورون‌ها را در ساختار سه بعدی مرتب کرد و به این ترتیب داریم:

- طول
- عرض
- ارتفاع

این ساختار با ساختار تصاویر موجود تطبیق دارد:

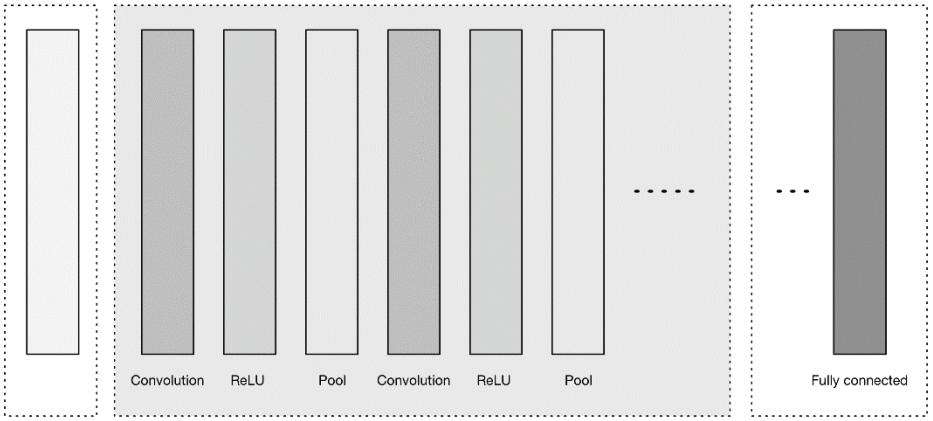
- طول برحسب پیکسل
- عرض برحسب پیکسل
- عمق تصویر برحسب کانال‌های رنگ

این ساختار را می‌توان یک حجم سه بعدی از نورون‌های درنظر گرفت. پیشرفت قابل توجه CNN ها نسبت به شبکه‌های فیدفورارد قبلی، بهینه بودن در محاسبات با توجه به استفاده از نوع جدید لایه‌ها می‌باشد.

۳-۲-۳- نگاهی کلی به معماری CNN

CNN ها با عبور دادن داده‌های ورودی از لایه‌ها، نهایتا در لایه پایانی آن‌ها را به تعدادی امتیاز برای هر کلاس تبدیل می‌کنند. معماری CNN دارای نسخه‌های متعددی است ولی از الگوی نشان داده شده در شکل ۳—۸ پیروی می‌کنند.

DOI: 10.30503/nripress.2020.321



شکل ۳–۸ : معماری سطح بالا و عمومی CNN [41]

در شکل ۳–۸ سه گروه اصلی از لایه‌ها نشان داده شده است.

- لایه ورودی
- لایه استخراج ویژگی (لایه یادگیری)
- لایه کلاس‌بندی

لایه ورودی به طور عمومی داده سه بعدی را دریافت می‌کند. توجه داشته باشید در صورت استفاده از mini-batch یعنی دسته‌بندی کردن داده‌های ورودی بایکدیگر برای انجام آموزش، بعد چهارم نیز به داده‌های ورودی اضافه می‌گردد. بعد چهارم مربوط به اندیس هر نمونه از داده‌ها در دسته‌های ایجاد شده است.

لایه‌های استخراج ویژگی دارای یکی الگوی تکراری عمومی با توالی زیر است:

۱. لایه کانولوشن
۲. لایه مربوط به توابع فعال‌سازی واحدهای خطی یکسویه
۳. لایه ادغام

این لایه‌ها تعدادی از ویژگی‌ها را از تصاویر استخراج می‌کنند و به طور پیشرونده‌ای ویژگی‌های سطح بالا می‌سازند. این همان حرکت عمومی در یادگیری عمیق است که ویژگی‌ها به طور خودکار یاد گرفته می‌شوند در مقابل روش‌های دستی سنتی.

در نهایت لایه‌های کلاس‌بندی را داریم که می‌تواند یک یا چند لایه کاملا متصل باشد که ویژگی‌های سطح بالا را دریافت کرده و امتیاز یا احتمال کلاس‌ها را در خروجی می‌دهد. این لایه‌ها به صورت کامل با تمام نورون‌های لایه قبل از خود متصل هستند و خروجی این لایه‌ها معمولا دو بعدی است یعنی تعداد کلاس‌ها (N) و تعداد نمونه‌ها (b) در هر mini-batch به صورت [b×N].

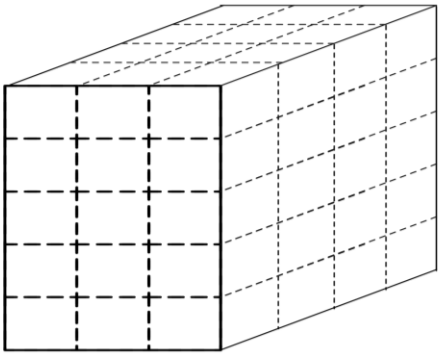
۳-۲-۳-۱- قرارگیری فضایی نورون‌ها

در شبکه‌های فیدفوروارد چند لایه، این لایه‌ها هستند که در بعد سوم پشت سر هم قرار می‌گیرند. در CNN به دلیل وجود mini-batch خود نورون‌های هر لایه در سه بعد قرار خواهند گرفت. توجه به این موضوع مهم است زیرا خود لایه‌ها سه بعدی هستند و اگر همانند فیدفورواردهای سنتی به صورت کامل به لایه بعدی متصل شوند کل نورون‌های قرارگرفته در سه بعد در یک لایه به کل نورون‌های قرار گرفته در سه بعد در لایه بعدی متصل می‌گردند.

DOI: 10.30503/nripress.2020.321

۳-۲-۳-۲- تحول در اتصال بین لایه‌ها

تحول دیگر نحوه اتصال بین لایه‌های کانولوشن است. نورون‌های یک لایه با ناحیه کوچکی از نورون‌های لایه قبل متصل هستند. CNN ها معماری لایه‌ای را همانند شبکه‌های چند لایه قبلی ابقاء کرده‌اند ولی دارای انواع لایه‌ها متفاوت می‌باشد. همانطور که در شکل ۳–۹ نشان داده شده است، هر لایه یک ورودی سه بعدی از لایه قبل دریافت می‌کند و سپس به خروجی توابع فعال‌سازی لایه خود تبدیل می‌کند. این کار را توسط توابعی تفکیک‌پذیر که ممکن است پارامترهایی داشته باشد انجام می‌دهد.



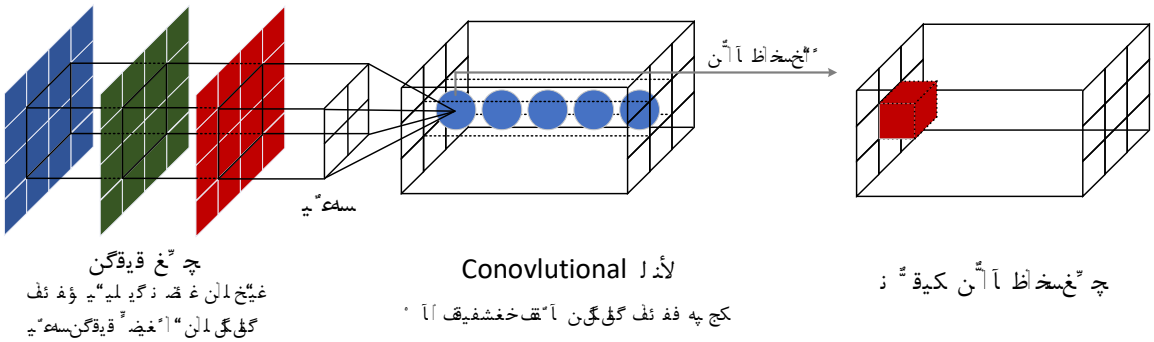
شکل ۳–۹ : حجم سه بعدی ورودی لایه

۳-۲-۴- لایه‌های ورودی

لایه‌های ورودی جایی است که داده‌های خام تصاویر برای پردازش در شبکه ذخیره می‌گردند. این ورودی‌ها دارای طول، عرض و تعداد کانال رنگ هستند. که معمولا برای RGB برابر ۳ است.

۳-۲-۵- لایه‌های کانولوشنی

لایه‌های کانولوشنی اجزاء ساختمانی اصلی در شبکه CNN هستند. همانطور که در شکل ۳–۱۰ نشان داده شده است، لایه‌های کانولوشنی داده‌های ورودی را با استفاده از یک تکه از نورون‌های متصل لایه قبلی تبدیل می‌کنند. لایه جاری ضرب نقطه‌ای بین ناحیه‌ای از نورون‌ها در لایه ورودی را با ماترس وزن‌ها محاسبه می‌کند تا مقدار آن در اتصال محلی لایه خروجی بدست آید.



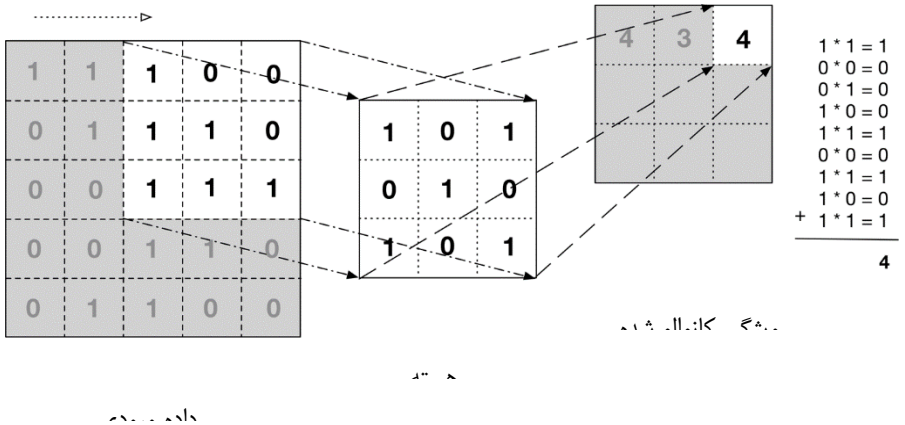
شکل ۳–۱۰ : لایه کانولوشن همراه با حجم ورودی و خروجی [42]

نتایج خروجی به طور عمومی دارای ابعادی مساوی یا کمتر از ابعاد ورودی است ولی در بعضی موارد عمق خروجی بیشتر از ورودی است.

DOI: 10.30503/nripress.2020.321

۳-۲-۵-۱- کانولوشن

کانولوشن یک عملیات ریاضی است که قواعد چگونگی ترکیب دو دسته از اطلاعات را بیان می‌کند. کانولوشن در دو زمینه فیزیک و ریاضی مهم است و به عنوان یک پل بین دو محدوده زمان/فضا و محدوده فرکانس از طریق تبدیلات فوریه عمل می‌کند. کانولوشن ورودی را گرفته، هسته کانولوشن را اعمال کرده و نگاشت ویژگی را به عنوان خروجی می‌دهد.



شکل ۳–۱۱ : عملیات کانولوشن [43]

عملیات کانولوشن همانطور که در شکل ۳–۱۱ نشان داده شده است به عنوان شناسایی‌کننده‌ی ویژگی در CNN شناخته شده است. ورودی کانولوشن می‌تواند داده خام یا یک نگاشت ویژگی خروجی از لایه دیگر باشد. کانولوشن معمولا به عنوان فیلتر تفسیر می‌گردد که در آن هسته نوع مشخصی از داده‌ها را فیلتر می‌کند. برای مثال فیلتر لبه فقط اجازه عبور داده‌های مربوط به لبه در تصاویر را می‌دهد.

شکل ۳–۱۱ نشان می‌دهد که چگونه هسته کانولوشن روی داده‌ها حرکت کرده (لیز خوردن) و داده‌های ویژگی‌های کانولوشن شده را تولید می‌کند. در هر گام از حرکت هسته روی داده‌ها، ماتریس هسته در داده‌های محدوده خود ضرب می‌شود و یک مقدار را در نگاشت ویژگی خروجی ایجاد می‌کند. در عمل مقدارهای ایجاد شده در خروجی، در صورتی که ویژگی مورد نظر شناسایی گردد، بزرگ خواهد بود.

معمولا فیلتر به مجموعه‌ای از وزن‌ها در لایه کانولوشن (هسته) گفته می‌شود. این فیلتر با داده‌های ورودی کانولوشن شده و نگاشت ویژگی‌ها را بوجود می‌آورد. لایه ورودی تبدیلاتی را روی حجم داده‌های ورودی انجام می‌دهد که تابعی از توابع فعال‌سازی لایه ورودی و پارامترهای خود لایه کانولوشن (وزن‌ها و بایاس‌های نورون‌ها) می‌باشد. نگاشت فعال‌سازی برای هر فیلتر به صورت پشته، عمق حجم خروجی را ایجاد می‌کند.

لایه کانولوشن دارای پارامترها و ابرپارامترهای اضافه‌ای می‌باشد که کاهش گرادیان[1] برای آموزش این پارامترها در این لایه استفاده می‌گردد و امتیاز کلاس‌ها، سازگار با برچسب‌های استفاده شده در مجموعه آموزش می‌باشد. اجزاء اصلی لایه کانولوشن درزیر آمده است.

- فیلترها
- نگاشت‌های فعال‌سازی[2]
- به اشتراک‌گذاری پارامتر
- ابرپارامترهای ویژ لایه[3]

[1] Gradient Descent
[2] Activation Maps
[3] Layer-specific hyperparameters

DOI: 10.30503/nripress.2020.321

در ادامه هریک توضیح داده شده است.

۳-۲-۵-۲- فیلترها

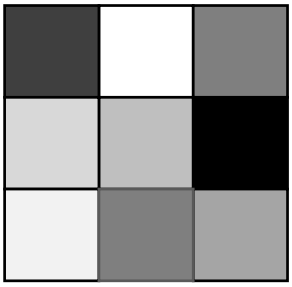
از پارامترهای قابل تنظیم در لایه‌های کانولوشنی پیکربندی مجموعه فیلترهای لایه می‌باشد. فیلترها دارای طول و عرض هستند که باید از طول و عرض ورودی کوچکتر باشند. می‌توان فیلترهایی هم سایز ورودی داشت ولی فقط در یک بعد یعنی فقط طول یا عرض. این مورد در بکارگیری CNN برای پردازش زبان طبیعی کاربرد دارد.

فیلترها به تمام عمق موجود در داده ورودی اعمال می‌گردد. خوروجی حاصل از اعمال فیلتر نگاشت فعال‌سازی (نگاشت ویژگی) نامیده می‌شود.

شمارنده فیلتر[1]:

یکی از ابرپارامترهای لایه کانولوشن می‌باشد. این ابرپارامتر تعداد نگاشت‌های فعال‌سازی تولید شده توسط لایه کانولوشن را که به عنوان ورودی برای لایه بعدی استفاده می‌گردد، کنترل می‌کند و به عنوان بعد سوم در حجم مربوط به نگاشت فعال‌سازی خروجی درنظر گرفته می‌شود. این ابرپارامتر می‌تواند به صورت آزادانه انتخاب گردد، زیرا بعضی از مقادیر ممکن است نتیجه بهتری حاصل کند.

فیلترها یاد می‌گیرند که فقط به ازاء الگوهای مشخصی فعال شوند. فیلترهایی وجود دارند که با ورود ترکیبی غیر خطی از الگوهای ورودی فعال می‌شوند. معماری‌های کانولوشن با کارایی بالا دارای شبکه عمیق تری هستند، این موضوع نشان دهنده اهمیت فاکتور عمق می‌باشد.



0.1	1	0.3
0.8	0.5	0.0
0.9	0.2	0.4

شکل ۳–۱۲ : کانولوشن و نگاشت فعال‌سازی

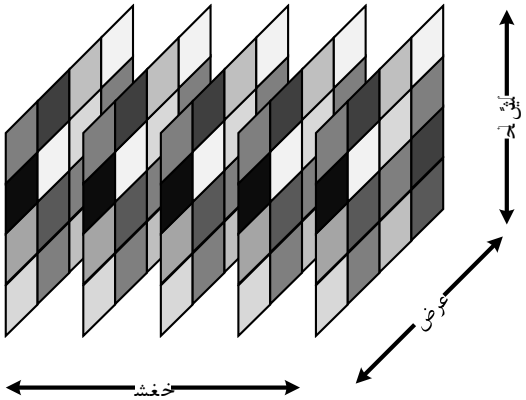
۳-۲-۵-۳- نگاشت‌های فعال‌سازی

وقتی یک فیلتر فعال است یعنی اجازه عبور اطلاعات را از حجم ورودی به حجم خروجی می‌دهد. با حرکت فیلتر روی داده در بعد طول و عرض یک خروجی دو بعدی به دست می‌آید که نگاشت فعال‌سازی نامیده می‌شود. شکل ۳–۱۲ نحوه ارتباط این نگاشت را با مفهوم ویژگی‌های کانولوشن شده که قبلا گفته شد نشان می‌دهد.

در بعضی از منابع از نگاشت فعال‌سازی با عنوان نگاشت ویژگی یاد می‌گردد. برای محاسبه نگاشت فعال‌سازی لازم است که فیلتر در عمق داده‌ها یعنی برش‌های موجود در حجم ورودی نیز حرکت کند. محاسبه به صورت ضرب نقطه‌ای بین فیلتر و حجم داده ورودی می‌باشد. فیلتر درواقع وزن‌هایی را نشان می‌دهد که باید با پنجره متحرک (زیرمجموعه) روی داده‌ها ضرب شود. فیلتر در موقعیت‌هایی که نوعی از ویژگی‌های مدنظر را ببیند فعال می‌گردد. همانطور که در شکل ۳–۱۳ نشان داده شده است، هر لایه

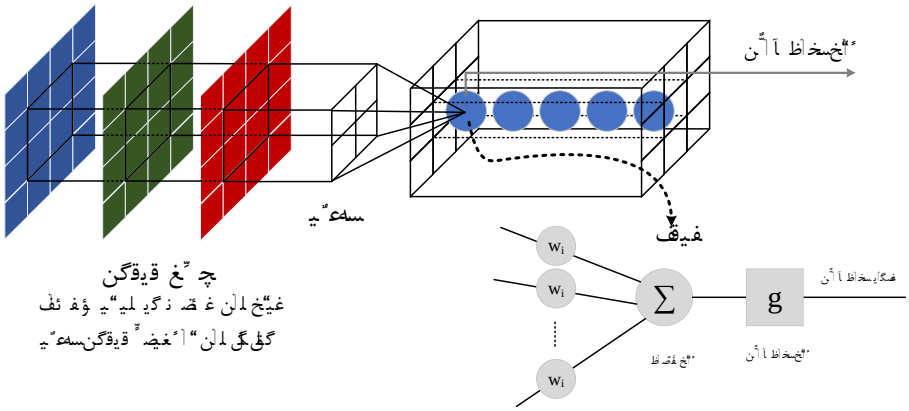
[1] Filter Count

کانولوشن با پشت سر هم قراردادن نگاشت‌های فعال‌سازی مربوط با حرکت در عمق یک حجم خروجی پشته‌ای را می‌سازد. هر المان از داده خروجی را می‌توان مربوط به یک نرون که در حال دیدن یک پنجره کوچک از داده است درنظر گفت.



شکل ۳–۱۳ : حجم فعال‌سازی خروجی مربوط به لایه کانولوشن

در بعضی از موارد این خروجی ناشی از اشتراک پارامترها با نورون‌ها در یک نگاشت فعال‌سازی می‌باشد. همانطور که در شکل نشان داده شده است. هر نورون که حجم خروجی را تولید می‌کند فقط به یک ناحیه محلی از حجم ورودی متصل است. اتصال محلی این فرایند توسط ابرپارامتری به نام receptive field کنترل می‌گردد. این ابرپارامتر مقدار عرض و طول حجم ورودی که باید توسط فیلتر نگاشت شود را کنترل می‌کند.



شکل ۳–۱۴: اتصال نرون‌های لایه کانولوشن به حجم ورودی (مفهوم برگرفته از [44])

نورون‌های در لایه کانولوشن به نواحی کوچکی از عرض و طول از حجم فضایی ورودی متصل هستند (شکل ۳–۱۴) درحالی که باید کاملا به تمام عمق ناحیه مورد نظر متصل باشند. برای مثال اگر از دیتاست CIFAR-10 استفاده کنیم که دارای تصایر RGB می‌باشد، با توجه به سایز تصاویر حجم ورودی برابر 32×32×3 خواهد بود. اگر ابرپارامتر receptive field را برابر 5×5×3 انتخاب کنیم، هر نورون در لایه کانولوشن به ناحیه‌ای با ابعاد 5×5×3 از حجم ورودی متصل خواهد بود. همانطور که مشخص است اتصال عمق ورودی برابر با عمق receptive field است یعنی اتصال در عمق به صورت کامل می‌باشد. بنابراین به تعداد 5×5×3=75 وزن برای هر نورون در لایه کانولوشن بدست می‌آید. همچنان ضرب نقطه‌ای بین ورودی و وزن‌ها توسط یک تابع غیر خطی محاسبه می‌گردد. بنابراین تنها تفاوت با شبکه‌های چند لایه سنتی تا اینجا این است که نرون‌ها به‌جای اتصال به تمام نرون‌های لایه قبل تنها به زیرمجموعه‌ای از آن‌ها متصل می‌شوند. بنابراین تعداد پارامترهایی که باید آموزش داده شوند کمتر خواهد بود و حتی با تکنیک دیگری به نام اشتراک پارامتر[1] می‌توان تعداد پارامترها را کمتر نیز کرد.

[1] Parameter Sharing

DOI: 10.30503/nripress.2020.321

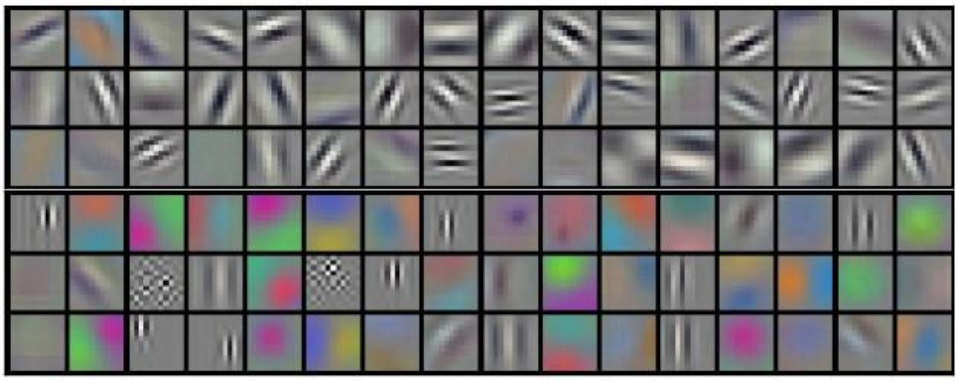
۳-۲-۵-٤- به اشتراک‌گذاری پارامتر

CNN از اشتراک‌گذاری پارامتر برای کنترل تعداد کلی پارامترها استفاده می‌کند که باعث کاهش زمان آموزش شبکه و استفاده کمتر از منابع خواهد شد. برای پیاده‌سازی به اشتراک‌گذاری پارامتر ابتدا هر برش موجود در عمق از حجم ورودی را در نظر بگیرد، این برش‌ها تکه هایی دو بعدی با عرض و طول برابر با حجم ورودی می‌باشند. نورون‌ها در هر برش وزن‌هایی برابر با بایاس در همان برش را استفاده می‌کنند.

اگر تصاویر دارای ساختارهای ویژه مرکزی باشند از اشتراک‌گذاری پارامتر نمی‌توان استفاده کرد. این موضوع در مورد صورت وجود دارد، وقتی می‌خواهیم یک ویژگی خاصی در مکان خاصی ظاهر گردد (برای تصویر صورت در مرکز). همچنین این ویژگی به CNN امکان عدم تغییر در موقعیت یا تبدیلات را می‌دهد.

۳-۲-۵-۵- فیلترها و رندر[1]های یاد گرفته شده

شکل ۳–۱۵ یک مثال از ۹۶ فیلتر آموزش داده شده با ابعاد $11\times11\times3$ را نشان می‌دهد. با برنامه به اشتراک‌گذاری پارامتر، می‌بینم که شناسایی لبه‌های افقی در بسیار از جاها به دلیل تغییر ناپذیری[2] انتقالی مفید است. به این معنی که لبه‌های افقی در یک جا یاد گرفته می‌شود و دیگر نگران یادگیری این ویژگی در تمام موقعیت‌های تصویر نیستیم.



شکل ۳–۱۵ : مثالی از فیلترهای آموزش دیده [45]

تغییر ناپذیری (ویژگی‌های بدست آمده) در مقابل چرخش به صورت عمومی وجود ندارد ولی (این تغییر ناپذیری) تا سطوحی با اضافه کردن داده‌های مناسب بدست می‌آید.

۳-۲-۵-٦- توابع فعال‌سازی ReLU به عنوان لایه

لایه ReLU تابع فعال‌سازی مبتنی بر المان‌های داده آستانه‌ای را اعمال می‌کند، برای مثال تابع $Max(0,x)$. بنابراین ابعاد داده ورودی با ابعاد داده خروجی یکسان است. با اعمال این لایه مقدار هر پیکسل می‌تواند تغییر کند ولی ابعاد ثابت است. همچنین ReLU پارامتری برای یادگیری و ابرپارامتری برای تنظیم ندارد.

۳-۲-۵-۷- ابرپارامترهای لایه کانولوشن

در زیر ابرپارامترهای لایه کانولوشن آمده است .

[1] Renders
[2] Invariant

DOI: 10.30503/nripress.2020.321

- اندازه هسته یا فیلتر
- عمق خروجی
- گام[1]
- Zero-padding
- اتساع[2]

اندازه فیلتر:

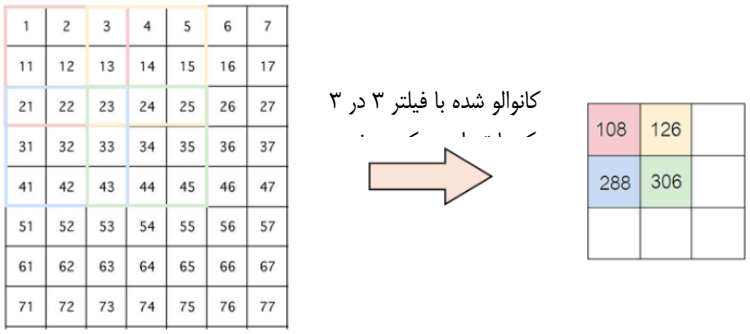
موضوعی است که قبلا به صورت کامل بحث شده است.

عمق خروجی:

تعداد نورون‌هایی را که به یک ناحیه مشخص از ورودی متصل هستند تعیین می‌کند. نورون‌های مختلفی در عمق یاد می‌گیرند تا در مقابل تحریک‌های مختلف ایجاد شده در ورودی فعال شوند (مثل رنگ، لبه و ...). درواقع می‌توان تصور کرد که یک دسته از نورون‌ها به یک ناحیه مشخص از ورودی نگاه می‌کنند و به این ترتیب عمق خروجی ایجاد می‌گردد.

گام:

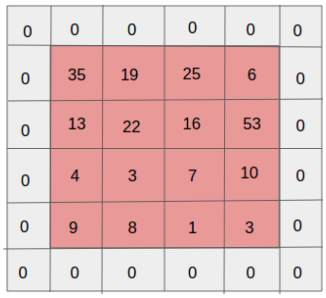
این ابر پارامتر تعیین می‌کند که پنجره فیلتر با گام‌های چند پیکسلی روی ورودی لیز بخورد. به این ترتیب حداقل گام ۱ پیکسل است که بیشترین میزان همپوشانی را دارد در شکل ۳–۱۶ یک نمونه از تنظم گام با ۲ پیکسل را مشاهده می‌کنید. توجه داشته باشید که در این شکل عمق داده ورودی ۱ درنظر گرفته شده برای تصاویر رنگی RGB این عمق برابر ۳ است.



شکل ۳–۱۶ : گام ۲ پیکسلی [46]

Zero-padding:

توسط این پارامتر می‌توان ابعاد طول و عرض حجم خروجی را کنترل کرد. این ابرپارامتر معمولا درجایی که می‌خواهیم طول و عرض ورودی با طول و عرض خروجی یکسان باشد کاربرد دارد. نمونه‌ای از در شکل ۳–۱۷ نشان داده شده است.



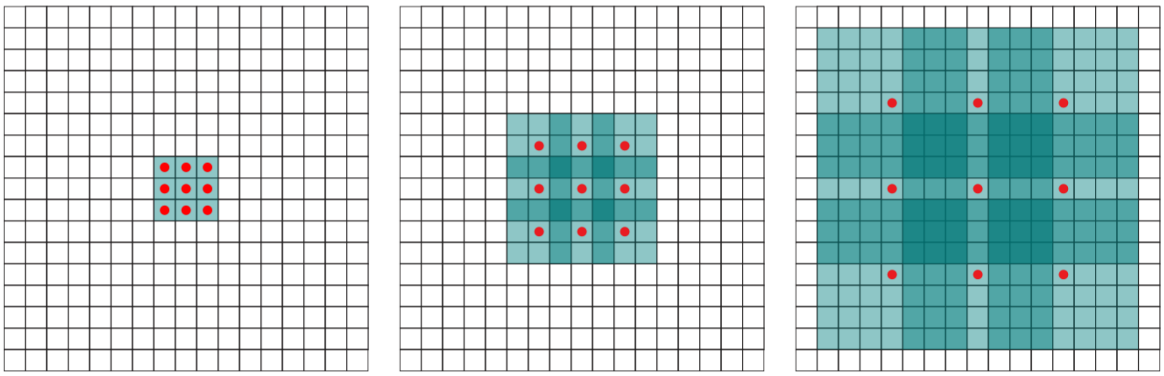
شکل ۳–۱۷ : یک نمونه Zero-padding **شده با یک پیکسل** [47]

[1] Stride
[2] Dilation

DOI: 10.30503/nripress.2020.321

اتساع[1]:

این پارامتر به صورت ساده می‌گوید هر المان ورودی به فیلتر چند پیکسل ورودی را شامل شود. به این ترتیب تعداد پیکسل‌های ورودی به فیلتر بدون تغییر ابعاد فیلتر و با تغییر پارامتر اتساع به صورت نمایی افزایش می‌یابد. برای مثال در شکل ۳–۱۸ بخش a (اتساع برابر ۱) هر المان ورودی به فیلتر فقط از یک پیکسل تاثیر می‌پذیرد، در بخش b (اتساع برابر ۲) هر المان ورودی از ۹ پیکسل در اطراف خود تاثیر می‌پذیرد و بخش c (اتساع برابر ۴) هر المان ورودی به فیلتر از ۴۹ پیکسل اطراف خود تاثیر می‌پذیرد. این ابرپارامتر روشی برای افزایش دید شبکه به صورت نمایی با یک پارامتر خطی می‌باشد. بنابراین شبکه می‌تواند اطلاعات بیشتری را با یک هزینه بسیار کم دریافت کند.



شکل ۳–۱۸ : اتساع کانولوشن: a) اتساع برابر ۱، b) اتساع برابر ۲ و c) اتساع برابر ٤ [48]

۳-۲-۵-۸- Batch Normalization

برای سرعت بخشیدن به آموزش شبکه می‌توان توابع فعال‌سازی لایه قبل را برای هر batch نرمال کرد [49]. این تکنیک تبدیلی را اعمال می‌کند که میانه توابع فعال‌سازی را نزدیک به ۰.۰ و انحراف معیار توابع فعال‌سازی را نزدیک به ۱.۰ نگه می‌دارد. این تکنیک با نرمال سازی بخشی از شبکه سرعت آموزش را افزایش می‌دهد. با اعمال این نرمال‌سازی به هر mini-batch می‌توان از نرخ یادگیری بالاتری استفاده کرد. همچنین حساسیت به وزن‌دهی اولیه را نیز کاهش می‌دهد. این نرمال سازی قابل اعمال به شبکه LSTM نیز می‌باشد [50].

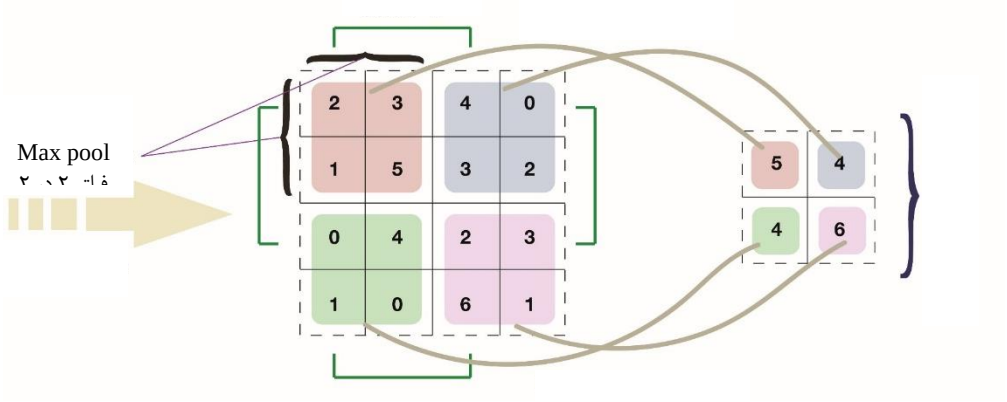
۳-۲-٦- لایه ادغام

لایه‌های ادغام بین لایه‌های متوالی کانولوشن قرار می‌گیرد. قرار گیری لایه‌های ادغام به بعد از لایه‌های کانولوشن باعث کاهش ابعاد طول و عرض نمایش داده خروجی هر لایه می‌گردد. کاهش ابعاد به کنترل overfitting کمک می‌کند. لایه ادغام به صورت مستقل روی هر برش از عمق داده ورودی عمل می‌کند.

یکی از اپراتورهای پرکاربرد برای ادغام Max() می‌باشد. برای مثال اگر اپراتور 2×2 باشد همانطور که در شکل ۳–۱۹ نشان داده شده است، تعداد چهار ورودی را دریافت کرده و بیشینه آنها را به خروجی می‌دهد. اپراتور پرکاربرد بعدی Average() می‌باشد که مانند اپراتور قبل ورودی را دریافت کرده و با این تفاوت که بین آن‌ها میانگین می‌گیرد. هیچ یک از اپراتورهای ادغام روی بعد عمق عمل نمی‌کنند. اعمال این اپراتورها نیاز به تنظیم گام نیز دارد. برای مثال اپراتور 2×2 با گام 2 هیچ همپوشانی روی ورودی

[1] Dilated

نخواهد داشت. لایه ادغام باعث افزایش ابرپارمترهای شبکه می‌گردد ولی تاثیر در تعداد پارامترهایی که باید یاد گرفته شوند ندارد. به طور معمول از Zero-padding برای لایه‌های ادغام استفاده نمی‌گردد.



شکل ۳–۱۹ : مثالی از اپراتور max pooling **با گام ۲** [51]

۳-۲-۷- لایه‌های کاملا متصل

از این لایه‌ها برای محاسبه امتیاز کلاس‌ها استفاده می‌گردد. مثلا برای دیتاست CIFAR تعداد این کلاس‌ها که همان خروجی‌های شبکه است برابر ۱۰ می‌باشد. لایه‌های کاملا متصل، تبدیلاتی را روی داده ورودی انجام می‌دهند که تابعی از توابع فعال‌سازی و پارامترهای ورودی (مثل وزن‌ها و بایاس‌های نورون‌ها) است.

تعدادی از شبکه‌های CNN از لایه‌های کاملا متصل در انتهای شبکه استفاده می‌کنند. AlexNet نمونه‌ای از این شبکه‌ها است که دارای دو لایه کاملا متصل و یک لایه softmax در انتها می‌باشد.

۳-۲-۸- کاربردهای دیگر شبکه‌های CNN

غیر از داده‌های تصاویر دو بعدی معمولی، CNN ها در دیتاست‌های سه بعدی نیز به کار می‌روند. مثال‌هایی از کاربرد آن در زیر آمده است.

- داده‌های MRI [52]
- داده‌های اشکال 3D [53]
- داده‌های گراف [54]
- پردازش زبان‌های طبیعی [55]

ویژگی تغییر ناپذیری در مقابل موقعیت، باعث کاربرد CNN در دامنه‌های ذکر شده در بالا شده است. زیرا محدودیتی در کدگذاری دستی ویژگی‌ها برای ظاهر شدن در موقعیت خاصی ندارد.

۳-۲-۹- CNN های مشهور

در زیر تعدادی از معماری‌های مشهور CNN آمده است.

- LeNet [56]
  - برای خواندن اعداد در تصاویر ایجاد گردیده است

DOI: 10.30503/nripress.2020.321

- [45] AlexNet
  - باعث محبوبیت CNN ها شد.
  - جایزه ILSVRC 2012 را از آن خود کرد.
- [34] ZF Net
  - جایزه ILSVRC 2013 را ازآن خود کرد.
  - مفهوم بصری‌سازی شبکه‌های عکس-کانولوشنی را معرفی کرد.
- [57] GoogleNet
  - جایزه ILSVRC 2014 را از آن خود کرد.
  - اسم کد آن Inception است و نسخه یک آن ۲۲ لایه دارد.
- [58] VGGNet
  - در ILSVRC 2014 به عنوان Runner-Up انتخاب شد.
  - نشان داد که عمق شبکه یکی از فاکتورهای مهم برای بدست آوردن کارایی خوب از شبکه است.
- [59] ResNet
  - به عنوان یک شبکه بسیار عمیق با ۱۲۰۰ لایه آموزش داده شده است.
  - جایزه اول را در کلاس‌بندی در ILSVRC 2015 دریافت کرد.

۳-۲-۱۰- خلاصه

CNN ها در استخراج ویژگی استاد هستند، مهم نیست این ویژگی در کجای تصویر باشد. همچنین دیدیم که لایه‌های کانولوشن، لایه‌ها ادغام و لایه‌های کاملا متصل چگونه کنار هم برای انجام کلاس‌بندی کار می‌کنند.

۳-۳- شبکه‌های عصبی بازرخداد

شبکه‌های بازرخداد در خانواده شبکه‌های عصبی فیدفوروارد قراردارد. تفاوت آنها با شبکه‌های معمولی فیدفوروارد در ارسال اطلاعات روی گام‌های زمانی است. برخلاف کامپیوترهای معمولی (پردازنده‌های رایج)، شبکه‌های عصبی بازرخداد همانند مغز انسان عمل می‌کند، به طوری که می‌تواند جریان ورودی یک حسگر را به دنباله‌ای از خروجی‌های مربوط به یک موتور تبدیل کند. این منجر به مدلی خواهد شد که قادر به حل مسائلی می‌باشد که ماشین‌های معمولی نمی‌توانند حل کنند.

همواره آموش این نوع از شبکه‌ها بسیار مشکل بوده است ولی پیشرفت‌های کنونی در بهینه‌یابی، ساختارهای شبکه، موازی‌سازی و واحدهای پردازش گرافیکی (GPU ها) استفاده از این مدل‌ها را دست‌یافتنی‌تر کرده‌اند.

شبکه‌های عصبی بازرخداد هر بردار از یک دنباله از بردارهای ورودی را به عنوان یک زمان مشخص مدل می‌کند. این‌کار باعث نگهداشتن حالت[1] شبکه می‌شود تا بتواند یک بردار ورودی را درمیان بردارهای یک پنجره ورودی مدل کند. نشانه بارز شبکه‌های بازرخداد مدل کردن بعد زمان می‌باشد.

[1] State

DOI: 10.30503/nripress.2020.321

۳-۳-۱- مدل‌سازی زمان

شبکه‌های عصبی بازرخداد را می‌توان به عنوان یک تورینگ[1] کامل در نظر گرفت که می‌تواند هر برنامه دلخواهی را (با استفاده از وزن‌ها) شبیه‌سازی کند. اگر شبکه‌های عصبی را به عنوان مسئله بهینه‌یابی روی توابع در نظر بگیریم، شبکه‌های عصبی بازرخداد را می‌توان به عنوان مسئله بهینه‌یابی روی برنامه‌ها درنظر گرفت. شبکه‌های عصبی بازرخداد برای مدل‌سازی توابعی بسیار مناسب است که ورودی و خروجی آن‌ها از بردار تشکیل شده و دارای وابستگی زمانی بین مقادیر می‌باشد. مدل‌سازی زمان در این شبکه‌ها با ساختن چرخه‌هایی صورت می‌گیرد.

۳-۳-۱-۱- گم شدن در زمان

بسیاری از ابزارهای کلاس‌بندی (ماشین بردار پشتیبان، رگرسیون منطقی[2] و شبکه‌های فیدفوروارد) بدون مدل‌سازی زمان با موفقیت بکار گرفته شده‌اند. نسخه‌هایی از این ابزارها با بکارگیری پنجره لیزخورنده دینامیک زمان را نیز مدل می‌کنند.

نسخه اولیه این ابزارها که فرض آن‌ها بر عدم وجود وابستگی زمانی است، برای مدل کردن وابستگی‌های بلند مدت بی‌فایده است. استفاده از تکنیک پنجره محدود در درک اثر وابستگی‌های بلندتر از اندازه پنجره شکست می‌خورند. یک مثال خوب برای این مورد مدل کردن نحوه عملکرد مکالمات و فهم نحوه پاسخگویی به صورت منسجم برای یک مکالمه در طول زمان می‌باشد. یک شبکه عصبی بازرخداد می‌تواند در تست تورینگ مشهور به رقابت بپردازد.

۳-۳-۱-۲- بازخورد موقتی و حلقه‌ها در اتصالات

شبکه‌های عصبی بازرخداد دارای حلقه در اتصالات هستند. این ویژگی امکان مدل کردن رفتارهای موقت و بدست آوردن دقت خوب در زمینه‌های سری‌های زمانی، زبان، صدا و متن را فراهم می‌کند. داده‌ها در این زمینه‌ها به صورت وراثتی مرتب می‌شوند یعنی مقادیر بعدی به مقادیر قبلی خود وابسته است و نسبت به زمینه حساس هستند. اتصالات و سیم‌کشی‌های شبکه‌های عصبی بازرخداد اجازه دریافت بازخورد را می‌دهد و درک این اثرهای زمانی را امکانپذیر می‌کند.

حلقه‌های بازخورد موجود در شبکه بازرخداد امکان یادگیری از توالی‌ها را فراهم می‌کند که شامل توالی‌هایی با طول متفاوت نیز می‌گردد. همچنین در این شبکه‌ها ماتریس پارامترهای اضافه‌ای وجود دارد که مربوط به اتصالات بین گام‌های زمانی است تا با استفاده از آن ارتباط‌های زمانی بین داده‌ها توسط شبکه درک شود.

این شبکه‌ها آموزش می‌بینند تا توالی‌ای از داده را تولید کنند که در آن داده‌ی تولید شده در هر گام مبتنی بر داده‌های ورودی همان گام و تمام ورودی‌های گام‌های قبلی می‌باشد. شبکه‌های عصبی بازرخداد معمولی گرادیان را با استفاده از الگوریتم پس انتشار در طول زمان (BPTT[3]) محاسبه می‌کنند.

۳-۳-۱-۳- توالی‌ها و داده‌های سری زمانی

در مسائل مختلفی در صنعت داده‌های دارای توالی یافت می‌گردد که خروجی مدل باید یک توالی از بردارها باشد.

- کپشن گذاری تصاویر [60]
- تحلیل گفتار [61]
- تولید موزیک [62]

[1] Turing
[2] Logistic Regression
[3] Backpropagation Through Time

DOI: 10.30503/nripress.2020.321

- مدل‌سازی زبان [63]
- مدل‌های تولید متن در سطح کاراکتر [64]

در زمینه‌های دیگر توالی از داده‌های ورودی نیز مورد نیاز است.

- پیش‌بینی سری زمانی
- تحلیل ویدئو
- بازیابی اطلاعات موسیقی

همچنین کاربردهایی وجود دارد که ورودی و خروجی به صورت سری هستند.

- ترجمه زبان طبیعی
- قرارگرفتن در گفتگو
- کنترل رباتیک

شبکه‌های بازرخداد در نوع داده‌های ورودی که می‌توانند مدل کنند با سایر شبکه‌های عصبی تفاوت واضحی دارند.

- گام‌های پردازشی غیر ثابت
- اندازه خروجی غیر ثابت
- می‌تواند عملیاتی را روی توالی از بردارها مثل فریم‌های ویدئو انجام دهد

وجه بسیار مهم شبکه‌های بازرخداد نحوه کار با ورودی و خروجی به صورت یکتا است.

۳-۳-۱-٤- فهم ورودی و خروجی مدل

در مدل‌های سنتی یک رابطه بین وروی و خروجی را با اندازه ثابت ورودی و خروجی می‌بینیم. این یک الگوی عمومی برای ایجاد یک کلاس‌بند برای کلاس‌بندی تصاویر با داده‌های ستونی می‌باشد. شبکه‌های بازرخداد این الگو را با ایجاد چندین ورودی پویا تغییر دادند، یک ورودی برداری برای هر گام زمانی. مثال‌هایی از نحوه عملکرد شبکه‌های بازرخداد روی توالی از بردارهای ورودی و خروجی در زیر آمده است.

- یک به چند: توالی در خروجی است. به عنوان مثال، برای کپشن گذاری تصاویر، یک تصویر دریافت می‌شود و توالی از کلمات در خروجی داده می‌شود.
- چند به یک: توالی در ورودی است. به عنوان مثال در تحلیل احساسات یک جمله یا توالی از کلمات به عنوان ورودی دریافت می‌کند.
- چند به چند: مثل کلاس‌بندی ویدئو. برچسب زدن به فریم‌های یک ویدئو.

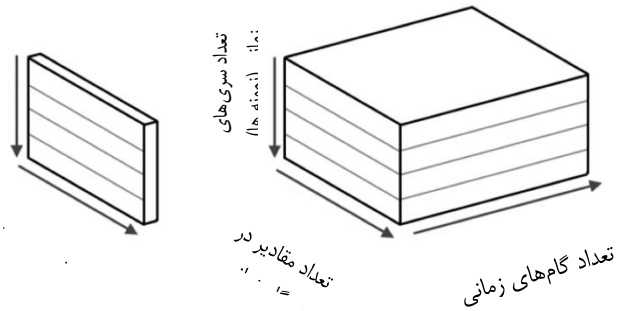
۳-۳-۲- ورودی حجمی D۳

همانطور که در شکل ۳—۲۰ مشخص است. ورودی شبکه‌های بازرخداد شامل ابعاد بزرگتری نسبت به شبکه‌های دیگر می‌باشد. از نظر مفهمومی مشابه CNN می‌باشد.

۱. اندازه Mini-batch
۲. تعداد ستون‌ها در بردار به ازاء هر گام زمانی

DOI: 10.30503/nripress.2020.321

۳. تعداد گام‌های زمانی



شکل ۳–۲۰ : ورودی شبکه‌های معمولی و ورودی شبکه‌های بازرخداد [65]

Mini-batch تعداد رکوردهای ورودی می‌باشد که میخواهیم در هر batch مدل شود، یعنی مکعب نشان داده شده در شکل ۳–۲۰. تعداد گام‌های زمانی نشان دهنده‌ی نحوه نمایش تغییرات ورودی در طول زمان است. این رویکرد سری‌زمان در داده‌ی ورودی می‌باشد. این نحوه نمایش داده‌های ورودی با گام‌های زمانی بزرگتر از یک برای مدل‌هایی است که از نوع Many-to- هستند.

۳-۳-۲-۱- سری‌های زمانی غیر هموار و پوشاندن[1]

نحوه اضافه کردن گام‌های زمانی در ویژگی‌های ورودی در شکل ۳–۲۱ نشان داده شده است.

گامهای زمانی

ستون‌های بردار ورودی	مقادیر					
	۰	۱	۲	۳	۴	...
بار	۱۲۰۰	۱۹۰۰	۱۵۰۰	۱۴۰۰	۱۶۵۰	...
دما	۲۵	۳۸	۳۱	۳۰	۳۱.۵	...
رطوبت	۰.۲	۰.۴	۰.۳	۰.۴	۰.۳	...
فشار	۰.۹۸	۰.۹۵	۰.۹۷	۰.۹۵	۰.۹۶	...
ساعت	۱۲	۱۲	۱۴	۱۵	۱۶	...

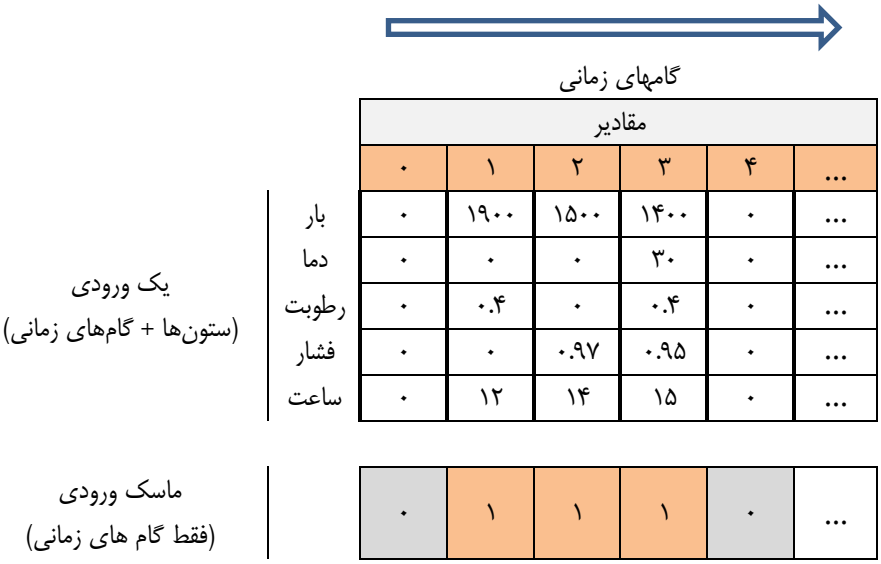
ستون‌های بردار ورودی	مقدار
	–
بار	۱۹۰۰
دما	۳۸
رطوبت	۰.۴
فشار	۰.۹۵
ساعت	۱۲

شکل ۳–۲۱ : نمایش گام‌های زمانی در ورودی شبکه‌های بازرخداد، شبکه پیش–خورد در سمت چپ و شبکه بازرخداد در سمت راست

تمام ستون‌ها ممکن است در تمام گام‌های زمانی مقدار نداشته باشند. مخصوصا زمانی که داده‌های توصیف کننده (ستون‌هایی از جدول پایگاه داده ایستا) را با داده‌های سری زمانی (مثل اندازه‌گیری‌های ICU از تعداد ضربان قلب بیمار در هر دقیقه) ترکیب کنیم. زمانی که داده‌های سری زمانی دندانه‌دار[2] هستند باید از ماسک (پوشش) استفاده کنیم. برای این کار همانند شکل ۳–۲۲ یک ماتریس دیگر اضافه می‌شود که به اندازه گام‌های زمانی بوده و هر گامی را که حداقل یکی از ویژگی‌های آن دارای مقدار باشد، مشخص می‌کند.

[1] Masking
[2] Jagged

DOI: 10.30503/nripress.2020.321



شکل ۳–۲۲ : ایجاد ماسک برای گام‌های زمانی خاص

۳-۳-۳- چرا نمی‌توان از مدل‌های مارکوو استفاده کرد؟

مدل‌های مارکوو کلاس دیگری از مدل‌های یادگری ماشینی است که می‌تواند توالی‌ها را مدل کند. فاکتور محدودکننده مدل مارکوو این است که با افزایش اندازه پنجره زمینه مدل‌سازی برای وابستگی‌های بلند مدت عملاً به لحاظ محاسباتی ناممکن می‌شود. بنابراین شبکه‌های عصبی بازرخداد که مدل‌های پیوندگرا هستند از مدل‌های مارکوو و سایر مدل‌های پنجره زمانی محدود بهتر هستند زیرا می‌توانند وابستگی‌های بلند مدت را در داده‌های ورودی درک کنند. شبکه‌های بازرخداد به دلیل استفاده از state پنهان می‌توانند اطلاعات را از پنجره با هر اندازه‌ای دریافت کنند، بدون محدودیتی که سایر تکنیک‌ها دارند. علاوه براین تعداد state هایی که می‌توانند مدل کنند توسط تعداد لایه‌های پنهان مشخص می‌گردد و تعداد آن با افزایش تعداد نودها به صورت نمایی افزایش می‌یابد. بنابراین به صورت نمایی می‌توانند اطلاعات مربوط به بعد زمان را در طول تعداد زیادی از بردارهای ورودی درک کنند. اگر ورودی فقط باینری باشد شبکه می‌تواند 2^N حالت را نمایش دهد که N تعداد نودها در لایه پنهان است. اگر خروجی عدد حقیقی ۶۴ بیتی باشد یک لایه پنهان با N نود می‌تواند 2^{64^N} حالت مختلف را نمایش دهد. سختی آموزش چنین شبکه‌ای با افزایش تعداد نودهای لایه پنهان تنها به صورت توان دوم افزایش می‌یابد درحالی که توان پاسخ‌گویی شبکه به صورت نمایی افزایش می‌یابد.

۳-۳-٤- معماری عمومی شبکه عصبی بازرخداد

این شبکه‌ها زیر مجموعه‌ای از شبکه‌های فیدفوروارد هستند با این تفاوت که مفهوم ارتباط بازرخداد را اضافه کردند. این ارتباط‌ها گام‌های زمانی مجاور (قبلی) را پوشش می‌دهند. به این ترتیب مفهوم زمان وارد مدل می‌گردد. شبکه‌های کانولوشن دارای چرخه در ارتباطات خود نیستند. اتصال بازرخداد باعث بوجود آمدن چرخه در ارتباطات می‌گردد شامل برگشت به خود نورون در گام زمانی بعدی.

۳-۳-٤-۱- معماری و گام‌های زمانی

DOI: 10.30503/nripress.2020.321

در هر گام زمانی نودهای دریافت‌کننده‌ی بازرخداد، هم از توابع فعال‌سازی ورودی داده را دریافت کرده و هم اطلاعات نودهای پنهان مربوط به state قبلی شبکه را دریافت می‌کنند. خروجی با استفاده از state پنهان در گام زمانی داده شده محاسبه می‌گردد. بردار ورودی قبلی در گام زمانی قبلی بر خروجی جاری در گام زمانی جاری از طریق ارتباطات بازرخداد اثر می‌گذارد.

می‌توان این لایه‌های نورون‌های بازرخداد را به یکدیگر زنجیره‌وار متصل کرد تا مدل بهتری بدست آید. این لایه‌ها باز هم همانند شبکه‌های فیدفوروارد به هم متصل می‌گردند.

شبکه‌های بازرخداد مشکل گرادیان ناپدید شونده[1] را دارند. این مسئله زمانی که گرادیان خیلی بزرگ یا خیلی کوچک می‌شود، اتفاق می‌افتد و مدل کردن وابستگی‌های برد بلند (۱۰ گام زمانی یا بیشتر) در ساختار دیتاست ورودی را مشکل می‌کند. این مسئله با استفاده از LSTM حل می‌گردد.

۳-۳-۵- شبکه‌های LSTM

LSTM یکی از رایج‌ترین و پراستفاده‌ترین نسخه‌های شبکه‌های عصبی بازرخداد است. این شبکه در سال ۱۹۹۷ میلادی معرفی گردید [66]. مؤلفه اساسی LSTM سلول حافظه[67] و گیت است که شامل گیت فراموشی[68] و گیت ورودی می‌باشد.

مفهوم سلول حافظه توسط گیت فراموشی و گیت ورودی مدوله شده است [67]. فرض کنید هر دو گیت بسته است، محتوای سلول بین گام‌های زمانی بدون تغییر می‌ماند. وجود ساختار گیت اجازه می‌دهد تا اطلاعات بین گام‌های زمانی ابقاء شود، در نتیجه اجازه می‌دهد گرادیان‌ها در تعداد زیادی گام زمانی جاری شود. همین موضوع باعث حل مسئله ناپدید شدن گرادیان که در اغلب شبکه‌های بازرخداد وجود دارد، می‌گردد.

۳-۳-۵-۱- ویژگی‌های شبکه‌های LSTM

LSTM ها به خاطر موارد زیر شناخته شده‌اند:

- بروزرسانی بهتر معادلات
- پس انتشار بهتر

مثال‌هایی از کاربرد LSTM:

- تولید جملات (مدل زبان در سطح کاراکتر)
- کلاس‌بندی سری زمانی
- بازشناخت گفتار
- بازشناخت دستخط
- مدل‌سازی موسیقی چند صدایی[2]

معماری LSTM و شبکه‌های بازرخداد دوسویه در بنچمارک‌های[3] صنعتی وظایف زیر را انجام می‌دهند.

- بازشناسی تصویر
- ترجمه زبان
- بازشناسی دستخط

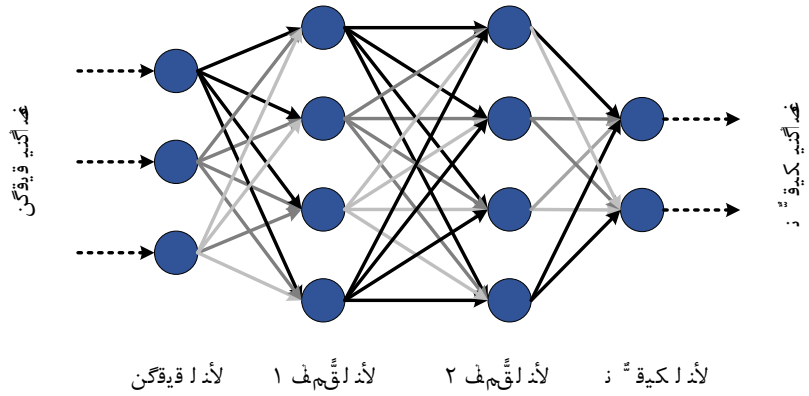
[1] Vanishing Gradient Problem
[2] Polyphonic
[3] Benchmark

DOI: 10.30503/nripress.2020.321

شبکه‌های LSTM از تعداد زیادی سلول LSTM ساخته می‌شود که در طول آموزش بسیار کارآمد عمل می‌کنند. پیچیدگی محاسباتی در عملیات پیشروی اطلاعات و پس‌انتشار آنها بر اساس تعداد گام‌های زمانی به صورت خطی مقیاس‌پذیر می‌باشد.

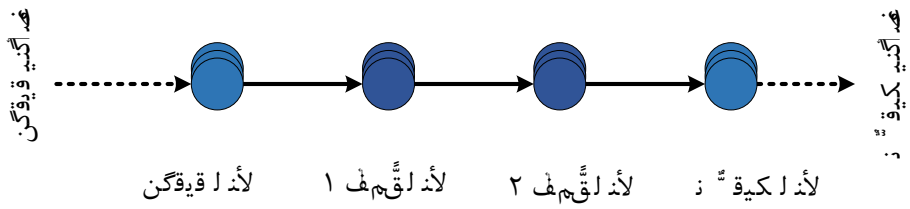
۳-۳-۵-۲- معماری شبکه LSTM

برای فهم بهتر ترتیب پیچیده اتصالات بین واحدها و لایه‌ها در شبکه LSTM باید ابتدا تعدادی از مفاهیم اولیه بیان گردد. یک شبکه فیدفوروارد ساده مانند آنچه می‌باشد که در شکل ۳–۲۳ نشان داده شده است



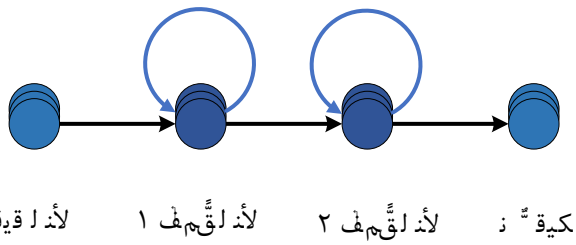
شکل ۳–۲۳ : معماری شبکه عصبی چند لایه فیدفوروارد

اگر هر لایه از این شبکه را به صورت یک نود تکی نشان دهیم تصوری مثل شکل ۳–۲۴ از این شبکه خواهیم داشت.



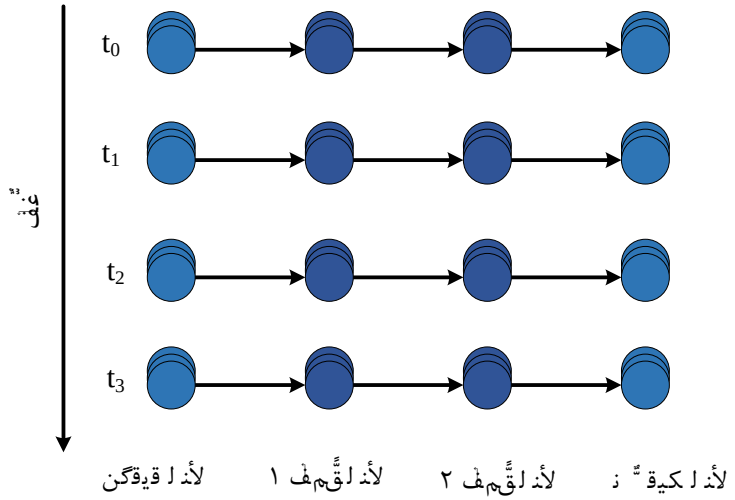
شکل ۳–۲۴ : شبکه چند لایه فید فوروارد با نمایی تسطیح شده

با استفاده از شبکه‌های بازرخداد نوعی از اتصال را می‌توان معرفی کرد که خروجی یک لایه پنهان را به ورودی همان لایه متصل می‌کند. با چنین ارتباط بازرخدادی می‌توان ورودی گام زمانی قبلی نورون را به عنوان بخشی از اطلاعات ورودی نورون در نظر گرفت. بنابراین با استفاده از تصوری در شکل ۳–۲۵ می‌توان اتصالات بازرخداد را در شبکه LSTM به صورت بصری نشان داد.



شکل ۳–۲۵ : اتصالات بازرخداد روی لایه‌های پنهان در شبکه LSTM

اگر فرض کنیم که شکل ۳–۲۵ تصویر یک طومار (توپ پارچه) باز نشده است، حال اگر آن را باز کنیم تصویر همانند شکل ۳–۲۶ خواهیم دید که جریان اطلاعات داخل شبکه به سبک فیدفوروارد و در طول زمان را نشان می‌دهد. شبکه‌های LSTM اطلاعات بیشتری را طول اتصالات بازرخداد نسبت به شبکه‌های بازرخداد سنتی عبور می‌دهد.



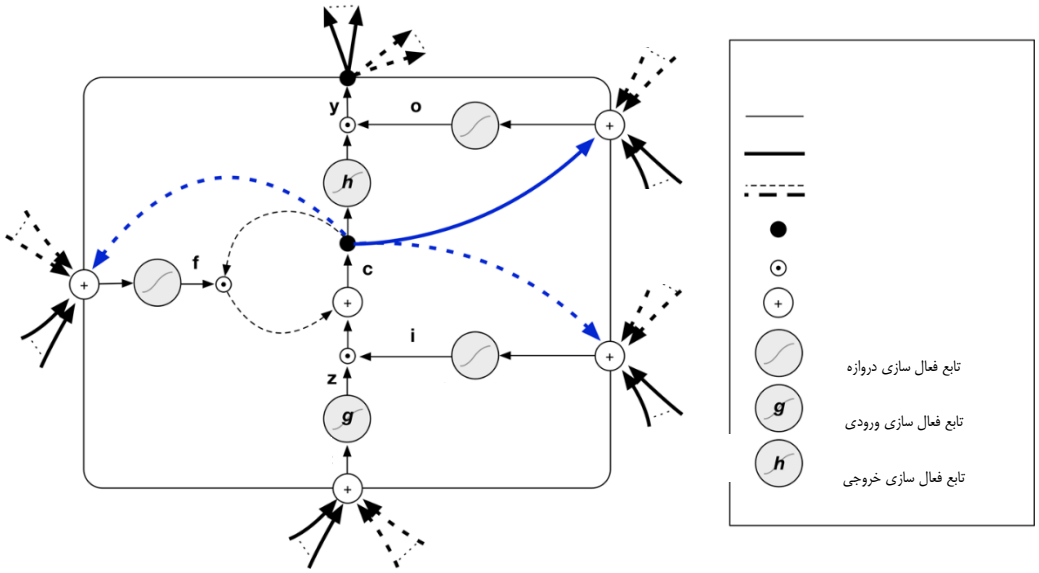
شکل ۳–۲۶ : باز کردن طومار شبکه بازرخداد در طول محور زمان

۳-۳-۵-۳- واحدهای LSTM

این واحدهای شبکه‌های بازرخداد متفاوت با نورون عصبی مصنوعی سنتی هستند. هر واحد LSTM دارای دو نوع ارتباط می‌باشد.

- اتصال مربوط به گام زمانی قبلی (خروجی واحدها در آن گام)
- اتصال مربوط به لایه قبل

سلول حافظه در شبکه‌های LSTM یک مفهوم مرکزی است که باعث نگهداشتن state در طول زمان می‌شود. بدنه اصلی واحد LSTM که به عنوان بلوک LSTM نیز شناخته می‌شود در شکل ۳–۲۷ نشان داده شده است.



شکل ۳–۲۷ : بلوک دیاگرام LSTM [69]

مؤلفه‌های واحد LSTM در زیر آمده است.

- سه عدد گیت

DOI: 10.30503/nripress.2020.321

- گیت ورودی (گیت تلفیق ورودی)
- گیت فراموشی
- گیت خروجی

- ورودی بلوک
- سلول حافظه (کاروسل[1] خطای پایدار)
- توابع فعال‌سازی خروجی
- اتصالات روزنه[2]

سه عدد گیت وجود دارد تا واحد خطی را در مقابل سیگنال‌های گمراه کننده محافظت کند.

- گیت ورودی واحد را درمقابل رویدادهای ورودی نامرتبط محافظت می‌کند.
- گیت فراموشی به واحد حافظه کمک می‌کند تا محتوای حافظه قبلی را فراموش کند.
- گیت خروجی محتوای سلول حافظه را در معرض خروجی واحد LSTM می‌گذارد.

خروجی بلوک LSTM یک اتصال بازرخداد به ورودی بلوک و تمام گیت‌های بلوک LSTM می‌باشد. گیت‌های ورودی، فراموشی و خروجی در واحد LSTM دارای توابع فعال‌سازی sigmoid با بازه بسته ۰ و ۱ می‌باشند. ورودی و خروجی بلوک LSTM معمولا یک تابع فعال‌سازی tanh می‌باشد.

تابع فعال‌سازی با مقدار ۱.۰ یعنی همه چیز را به خاطر بیاور و مقدار ۰.۰ یعنی همه چیز را فراموش کن. از دیدگاه دیگر می‌توان نام مناسب گیت یادآوری را برای گیت فراموشی آن انتخاب کرد. بنابراین با انتخاب مقادیر بزرگ برای گیت فراموشی به عنوان بایاس اولیه وابستگی‌های بلند مدت را یاد خواهد گرفت.

$$
\begin{aligned}
\mathbf{z}^t &= g(\mathbf{W}_z\mathbf{x}^t + \mathbf{R}_z\mathbf{y}^{t-1} + \mathbf{b}_z) \\
\mathbf{i}^t &= \sigma(\mathbf{W}_i\mathbf{x}^t + \mathbf{R}_i\mathbf{y}^{t-1} + \mathbf{p}_i \odot \mathbf{c}^{t-1} + \mathbf{b}_i) \\
\mathbf{f}^t &= \sigma(\mathbf{W}_f\mathbf{x}^t + \mathbf{R}_f\mathbf{y}^{t-1} + \mathbf{p}_f \odot \mathbf{c}^{t-1} + \mathbf{b}_f) \\
\mathbf{c}^t &= \mathbf{i}^t \odot \mathbf{z}^t + \mathbf{f}^t \odot \mathbf{c}^{t-1} \\
\mathbf{o}^t &= \sigma(\mathbf{W}_o\mathbf{x}^t + \mathbf{R}_o\mathbf{y}^{t-1} + \mathbf{p}_o \odot \mathbf{c}^t + \mathbf{b}_o) \\
\mathbf{y}^t &= \mathbf{o}^t \odot h(\mathbf{c}^t)
\end{aligned}
$$

با استفاده از نوتیشن Greff و همکارانش فرمول برداری برای عبور به سمت جلو در لایه LSTM به صورت معادله زیر خواهد بود.

لیست متغییرهای این رابطه به صورت زیر است.

x^t = بردار ورودی در زمان t است.

W = ماتریس‌های وزن‌های ورودی

R = مربع ماتریس‌های وزن‌های بازرخداد

P = بردارهای وزن روزنه

b = بردارهای بایاس

[1] Carousel
[2] Peephole

DOI: 10.30503/nripress.2020.321

اتصالات خود–بازرخداد دارای وزن ثابت ۱.۰ هستند (بجز زمانی که مدوله شده است یعنی مقادیری متفاوت بر اساس یک دامنه تابع دیگر دارد) تا بر مسئله ناپدید شدن گرادیان غلبه کند. این واحد هسته امکان کشف رویدادهای برد بلند را در دنباله می‌دهد. این رویدادها می‌توانند تا ۱۰۰۰ گام زمانی گسترش یابند که در مقایسه با شبکه‌های بازرخداد قدیمی که فقط تا ۱۰ گام امکانپذیر بود بسیار بهینه تر عمل می‌کنند.

۳-۳-۵-٤- GRU

واحد بازرخداد دیگری همانند LSTM واحد بازرخداد دروازه‌ای (GRU[1]) می‌باشد [70]. GRU دارای گیت reset و گیت update می‌باشد که مشابه گیت‌های فراموشی و ورود در واحد LSTM است. تفاوت عمده در GRU در معرض قرار دادن کامل محتوای حافظه از طریق انتگرال ناقص[2] است. این کار از طریق یک ثابت زمانی تطبیق‌پذیر انجام می‌گیرد که توسط گیت update کنترل می‌گردد. GRU از واحد LSTM الهام گرفته شده است که هدف آن محاسبات و پیاده‌سازی ساده‌تر می‌باشد.

۳-۳-۵-۵- لایه‌های LSTM

یک لایه به صورت پایه بردار ورودی با طول غیر ثابت x را دریافت کرده و y را در خروجی می‌دهد. خروجی y از x و تمام سابقه آن اثر می‌پذیرد. لایه LSTM با استفاده از اتصالات بازرخداد از سابقه x اثر می‌پذیرد. هر RNN دارای یک state داخلی است که هر زمان با ورود یک بردار به لایه به‌روز می‌گردد. یک بردار تکی پنهان state را تشکیل می‌دهد.

۳-۳-۵-٦- آموزش

شبکه LSTM از آموزش نظارتی برای بروزرسانی وزن‌ها استفاده می‌کند. آموزش با استفاده از یک بردار ورودی در یک زمان در دنباله‌ای از بردارها انجام می‌گیرد. بردارها دارای مقادیر حقیقی بوده و به دنباله‌ای از توابع فعال‌سازی نودهای ورودی تبدیل می‌شوند. هر واحد غیر ورودی فعال‌سازی جاری خود را در هر گام زمانی محاسبه می‌کنند. این مقادیر فعال‌سازی به عنوان توابع غیر خطی از جمع وزن‌دار فعال‌سازی‌های تمام واحدهایی که اتصال از آن‌ها دریافت می‌کند، محاسبه می‌گردد.

برای هر بردار ورودی در دنباله ورودی‌ها، خطا برابر با جمع انحرافات تمام سیگنال‌های هدف از فعال‌سازی‌های معادل محاسبه شده توسط شبکه می‌باشد. حال باید نگاهی به BPTT بیاندازیم، که نسخه‌ای از پس‌انتشار استفاده شده توسط شبکه‌های بازرخداد ازجمله LSTM می‌باشد.

۳-۳-۵-۷- BPTT و BPTT بریده[3]

آموزش شبکه‌های عصبی بازرخداد از نظر محاسباتی بسیار پرهزینه است. گزینه سنتی استفاده از BPTT می‌باشد. اساس BPTT همانند پس‌انتشار استاندارد است. قاعده‌ی زنجیره برآن اعمال می‌گردد تا گرادیان‌ها را برپایه ساختار اتصالات شبکه محاسبه کند. درواقع تشخیص در طول زمان وجود دارد، یعنی بعضی از سیگنال‌های خطا یا گرادیان‌ها از گام‌های زمانی آینده به سمت عقب جاری می‌شوند، نه فقط از لایه‌های بالاتر که در پس‌انتشار استاندارد وجود دارد.

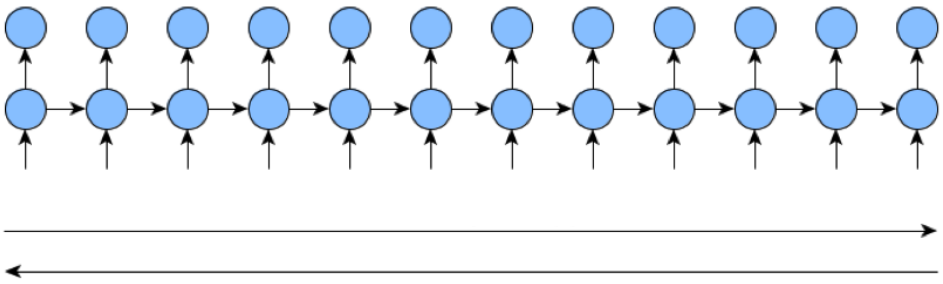
[1] Gated Recurrent Unit
[2] Leaky Integration
[3] Truncated

DOI: 10.30503/nripress.2020.321

وقتی شبکه بازرخداد با دنباله‌های بلندی با تعداد زیادی گام زمانی (بیشتر از چند صد تا) سروکار دارد استفاده از truncated BPTT بهتر است. با استفاده از truncated BPTT پیچیدگی محاسباتی بروزرسانی پارامترها کمتر می‌شود. انجام زود به زود بروزرسانی پارامترها سرعت شبکه را بالاتر می‌برد.

محاسبه گرادیان برای یک شبکه بازرخداد با دنباله‌ای از ۱۰۰۰ گام زمانی دارای هزینه محاسباتی برابر با عبور داده‌های به سمت جلو و برگشت آن‌ها روی شبکه پرسپترون با ۱۰۰۰ لایه می‌باشد.

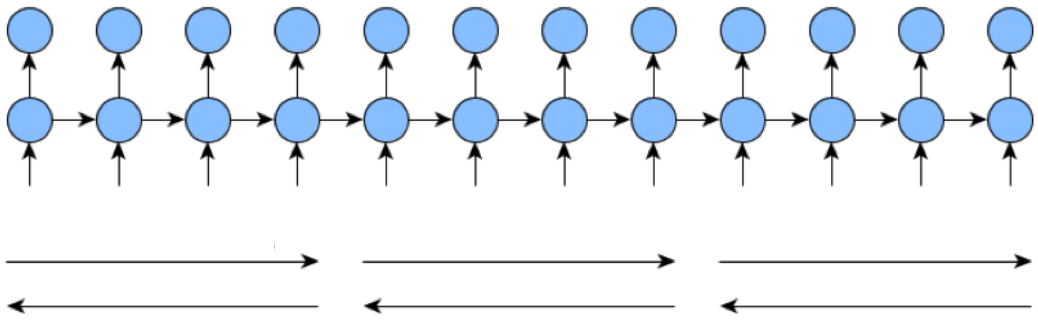
برای فهم بهتر truncated BPTT آموزش یک شبکه با ورودی سری زمانی با ۱۲ گام زمانی را درنظر بگیرد. همانطور که در شکل ۳–۲۸ نشان داده شده است، در این سناریو عبور ۱۲ گام و سپس محاسبه خطای شبکه انجام می‌شود. پس از آن برگشت ۱۲ گام زمانی انجام می‌گیرد.



شکل ۳–۲۸ : BPTT **استاندارد** [65]

۱۲ گام زمانی برای فرایند آموزش خیلی دشوار نیست، وقتی تعداد گام‌های زمانی بیشتر از صد یا چند صد باشد آموزش شبکه بسیار دشوار خواهد بود. اگر تعداد گام‌های زمانی ۱۰۰۰ عدد باشد پس‌انتشار استاندارد باید برای بروزرسانی هر پارامتر ۱۰۰۰ گام زمانی را به سمت جلو حرکت کرده و سپس برگردد. بنابراین می‌بینید که به سرعت از نظر محاسباتی هزینه‌بر می‌شود به همین دلیل به دنبال روش‌هایی مثل BPTT و truncated BPTT می‌گردیم.

truncated BPTT عبور به سمت جلو و عقب را در عملیات کوچک جداگانه انجام می‌دهد. همانطور که در شکل ۳–۲۹ نشان داده شده است، هر عبور به سمت جلو معادل خود، یک عبور به سمت عقب را ایجاد می‌کند و پس از آن پارامترهای نامزد را به‌روز می‌کند. اندازه این رفت و برگشت‌ها یک ابرپارامتر است که توسط کاربر مشخص می‌گردد. در این تصویر truncated BPTT از چهار گام زمانی عبور می‌کند.



شکل ۳–۲۹ : نحوه عملکرد truncated BPTT [65]

روش truncated BPTT در حال حاضر عملی‌ترین روش آموزش شبکه‌های بازرخداد است. با استفاده از truncated BPTT می‌توان با هزینه پردازش کمتری نسبت به BPTT وابستگی‌های بلند مدت را استخراج کرد. گاهاً وابستگی‌های بدست آمده توسط truncated BPTT کوتاه‌تر از BPTT است که این را می‌توان تا حدودی از یک جنبه منفی درنظر گرفت ولی

DOI: 10.30503/nripress.2020.321

افزایش سرعت بدست آمده توسط truncated BPTT در صورتی که تعداد گام‌های زمانی مناسب انتخاب شود، بسیار ارزشمند است.

۳-۳-٦- دامنه‌های کاربرد و شبکه‌های ترکیبی

شبکه‌های بازرخداد تطبیق‌پذیری خود را در بینایی ماشین ثابت کرده‌اند. این کاربردها شامل موارد زیر است.

- تحلیل ویدئو در سطح فریم [71]
- کپشن گذاری ویدئو [72]
- پاسخ‌گویی سوالات تصویری [73]

حوزه مرتبط جدیدی با RNN در بینایی ماشینی شبکه‌هایی هستند که اطلاعات بخش کوچکی از تصویر را استخراج می‌کنند که مدل بازرخداد توجه بینایی نامیده می‌شوند [74]. این شبکه‌ها که ترکیبی از CNN و RNN هستند، برای تصاویری که شامل تعداد زیادی شیء است، مناسب می‌باشند. کاربرد دیگری از ترکیب CNN و BRNN شبکه‌ای است توصیف زبان طبیعی برای تصویر و نواحی آن ایجاد می‌کند.

۳-٤- شبکه‌های عصبی بازگشتی

این شبکه‌ها نیز همانند شبکه‌های بازرخداد با رشته‌ای از متغییرهای ورودی سروکار دارد. تفاوت عمده در شبکه‌های بازگشتی قابلیت مدل کردن داده‌های ورودی با ساختار سلسله مراتبی است. تصاویر دارای صحنه‌هایی ترکیب‌یافته از اشیاء مختلف می‌باشد. تجزیه صحنه‌ها یکی از زمینه‌های تحقیقاتی هست که جای زیادی برای کار دارد. شبکه‌های بازگشتی نه‌تنها در تجزیه اشیاء صحنه ما را به چالش می‌کشند بلکه ارتباط بین اشیاء با صحنه را نیز دربر می‌گیرند.

۳-٤-۱- معماری شبکه

معماری شبکه عصبی بازگشتی از ماتریسی از وزن‌های اشتراکی و ساختار درخت باینری ترکیب یافته است که باعث می‌شود شبکه دنباله‌هایی متفاوت از کلمات یا بخش‌های مختلفی از تصویر را یاد بگیرد. این شبکه به عنوان تجزیه‌کننده جمله یا صحنه مناسب است. شبکه بازگشتی نسخه‌ای از پس انتشار به نام پس‌انتشار در طول ساختار (BPTS[1]) را استفاده می‌کند. فیدفوروارد به صورت بالا به پایین عبور کرده و پس‌انتشار پایین به بالا است (یک ساختار درختی معکوس را درنظر بگیرد). هدف در نوک درخت و ورودی‌ها در پایین درخت است.

۳-٤-۲- تنوع شبکه‌های عصبی بازگشتی

این شبکه‌ها دارای چند نوع هستند یک نوع autoencoder بازگشتی است. همانند autoencoder معمولی تلاش می‌کند تا ورودی را بازسازی کند. در مورد NLP تلاش می‌کند تا زمینه موضوع را بسازد. یک autoencoder بازگشتی نیمه نظارتی تلاش می‌کند تا احتمال برچسب‌های معین در هر زمینه را یاد بگیرد.

[1] Backpropagation Through Structure

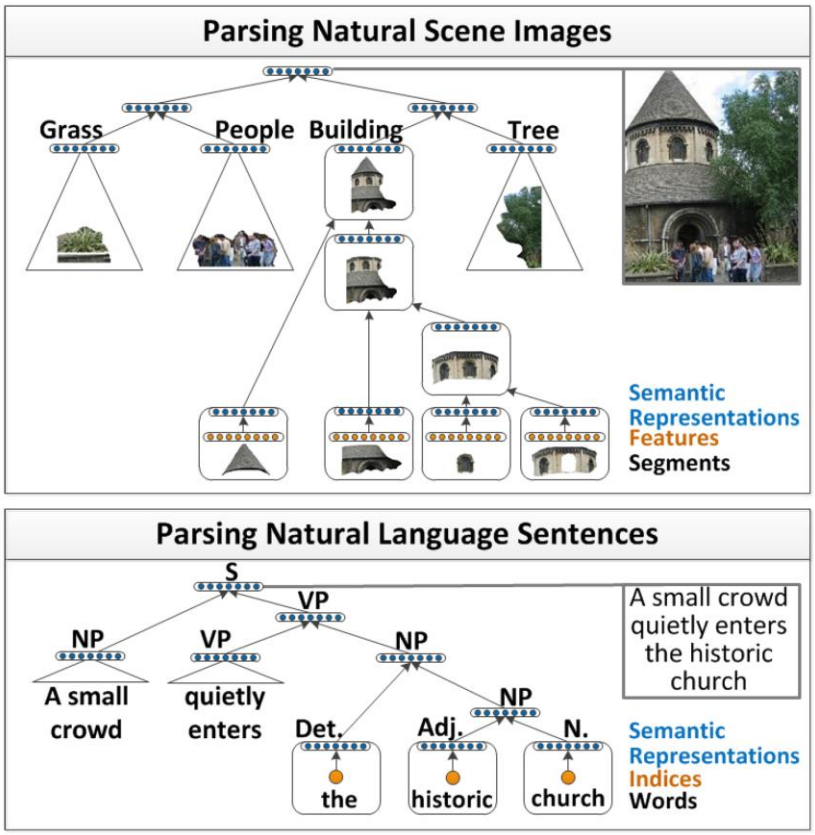
DOI: 10.30503/nripress.2020.321

نسخه دیگر آن شبکه عصبی نظارتی است که شبکه تنسور عصبی بازگشتی نامیده می‌شود که یک هدف نظارتی را در هر نود از درخت محاسبه می‌کند. وجود تنسور در نام آن به معنی محاسبه گرادیان به روشی کمی متفاوت است. فاکتورگیری از اطلاعات بیشتر در هر نود با استفاده از مزیت ابعاد دیگر اطلاعات که با استفاده از تنسور انجام می‌شود (ماترسی با ابعاد سه و یا بالاتر).

۳-۴-۳- کاربردهای شبکه‌های عصبی بازگشتی

هردوی شبکه‌های عصبی بازگشتی و بازرخداد دارای کاربردهای مشترک زیادی هستند. شبکه‌های بازگشتی به صورت سنتی برای NLP به کار می‌رفت زیرا از درخت باینری، context ها و تجزیه‌کننده‌های[1] مبتنی بر زبان طبیعی استفاده می‌کند. شبکه‌های بازگشتی هم ساختارهای دانه‌ای[2] و هم ساختار دیتاست‌های سلسله مراتبی سطح بالا را، همانند تصاویر و جملات، می‌توانند بازیابی کنند. کاربردها شامل موارد زیر می‌گردد.

- تجزیه صحنه در تصاویر
- NLP
- تبدیل صدا به نوشته



شکل ۳–۳۰: ساختار یک شبکه عصبی بازگشتی که بخش‌هایی از یک تصویر و جملات یک زبان طبیعی را پارس کرده است [75].

[1] Parser
[2] Granular

DOI: 10.30503/nripress.2020.321

۳-٤-٤- مثالی از کاربرد شبکه‌های عصبی بازگشتی در NLP

همانطور که می‌دانید قواعد نحوی زبانی طبیعی به عنوان یک فرایند بازگشتی شناخته شده است. عبارت‌های اسمی[1] در دل خود دارای عبارت‌های اسمی دیگری است. مثلا : he church which has nice windows. چنین ساختار تودرتو[2] را می‌توان در تصاویر مربوط به صحنه‌های طبیعی نیز یافت که در آن رابطه part-of و proximity وجود دارد. برای مثال cars are often on top of street regions. بنابراین یک ناحیه بزرگ مربوط به ماشین‌ها را می‌توان به صورت بازگشتی به نواحی کوچکتر مربوط به ماشین ها تقسیم کرد. همانطور که در شکل ۳—۳۰ با استفاده از این ساختار بازگشتی می‌توان اجزاء تصاویر مربوط به صحنه‌های طبیعی و بخش‌های اصلی یک جمله را پارس کرد از خروجی این مرحله بعدا می‌توان برای کلاس‌بنی تصاویر استفاده کرد. در جهت فهم بهتر این نوع از شبکه‌ها، به صورت خیلی ساده شده‌ای می‌توان گفت ساختار نودها و شبکه بر اساس ورودی و به صورت تودرتو ایجاد می‌گردد.

به طور کلی بر اساس بررسی صورت گرفته روی معماری‌های اصلی درشبکه‌های عمیق، این معماری‌ها را بر اساس کاربرد می‌توان به صورت زیر خلاصه کرد.

- تولید داده مثل تصویر، صدا و متن
  - GAN
  - VAE
  - شبکه‌های عصبی بازرخداد
- مدل‌سازی تصاویر
  - CNN
  - DBN
- مدل‌سازی داده‌های دارای توالی
  - شبکه‌های عصبی بازگشتی مخصوصا LSTM

در روش‌های پیش‌بینی بار معمولا از داده‌های بار گذشته، اطلاعات تقویمی و اطلاعات آب و هوایی استفاده میشود. که به صورت سنتی پژوهشگران با روش‌های آزمون و خطا و استدلال‌های خبرگی تلاش میکنند داده‌های ورودی برای شبکه را طوری بدست آورند که بیشترین ارتباط معنادار را با باری که قرار است پیش‌بینی شود داشته باشد. به این ترتیب دقت پیش‌بینی را افزایش می‌دهند. اینکار نیازمند بارها آموزش شبکه، تست ساختارهای مختلف شبکه و زمان زیادی است. حال اگر نخواهیم خودمان زمان زیادی برای بدست آوردن این رابطه معنادار بین داده‌ها صرف کنیم و این کار را به عهده یک هوش مصنوعی قرار دهیم. با توجه قابلیت‌های LSTM، بهترین گزینه استفاده از چنین شبکه‌ای است که حجم زیادی از داده‌های بار گذشته، داده‌های تقویمی گذشته و آینده و داده‌های آب و هوایی را در اختیار آن قرار می‌دهیم. انتخاب داده‌های مناسب و کشف رابطه‌های معنادار را به عهده شبکه می‌گذاریم.

[1] Noun phrases
[2] Nested

DOI: 10.30503/nripress.2020.321

۳-۵- پیش‌بینی بار با استفاده از یاد گیری عمیق

همنطور که در بالا اشاره شد زمانی که بخوایم بیشتر کار پیش‌بینی را به هوش مصنوعی واگذار کنیم استفاده از شبکه‌های عمیق با معماری LSTM مناسب خواهد بود. همانطور که در بازشناسی تصاویر، تشخیص چهره و کلاس‌بندی تصاویر کار استخراج ویژگی به عهده شبکه عمیق گذاشته می‌شود. در اینجا نیز یافتن ارتباط معنادار بین داده‌های موجود و بار قابل پیش‌بینی را می‌توان به عهده یک شبکه عمیق قرار داد. نکته مهم در این کار استفاده از سخت افزار مناسب می‌باشد، تا آموزش شبکه در زمان معقولی انجام گیرد.

معمولا انواع داده زیر در پیش‌بینی بار دردسترس می‌باشد.

۱. بار روزهای گذشته به صورت ساعتی
۲. داده‌های تقویمی روزهای گذشته، روز جاری و روزهای آینده
۳. داده دمای واقعی روزهای گذشته و جاری
۴. داده دمای پیش‌بینی شده روزهای آینده
۵. داده رطوبت واقعی روزهای گذشته و روز جاری
۶. داده رطوبت روزهای آینده
۷. داده وضعیت آسمان (ابری، بارندگی، برفی، آفتابی و ...) واقعی برای روزهای گذشته و جاری
۸. داده وضعیت آسمان (ابری، بارندگی، برفی، آفتابی و ...) پیش‌بینی برای روزهای آینده

چند آرایش پیش‌نهادی متفاوت را برای داده‌ها ورودی و خروجی شبکه LSTM به صورت زیر می‌توان درنظر گرفت.

۱. تک متغییره با یک ورودی (گام قبل) و یک خروجی (گام بعدی)
۲. چند متغییره با یک ورودی (گام قبل) و یک خروجی (گام بعدی)
۳. تک متغییره با ورودی پنجره زمانی و یک گام خروجی
۴. چند متغییره با ورودی پنجره زمانی و یک گام خروجی
۵. تک متغییره با ورودی پنجره زمانی و خروجی برداری (چندین گام بعدی)
۶. چند متغییره با ورودی پنجره زمانی و خروجی برداری (چندین گام بعدی)

آرایش ۱ و ۲ دارای زمان کمتری برای آموزش لازم دارند. زمان آموزش در آرایش‌های ۳و ۴ بسیار زیاد خواهد شد و در آرایش‌های ۵ و ۶ برای زمان آموزش روی یک دستگاه مشخص به شدت افزایش می‌یابد.

۳-۶- مطالعه آینده

در مطالعه بعدی در پروژه جاری روش LSTM به عنوان روش پیش‌نهاد شده در این تحقیق برای یکی از آرایش‌های ذکر شده در بالا و روی داده‌ی نمونه انجام خواهد شد. جزئیات بیشتر مربوط به آرایش مورد استفاده همراه با کد پیاده سازی شده به تفصیل در کار بعدی خواهد آمد.

مراجع

[1] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, Art. no. 7540, Feb. 2015, doi: 10.1038/nature14236.

[2] K. Dutta, P. Krishnan, M. Mathew, and C. V. Jawahar, “Improving cnn-rnn hybrid networks for handwriting recognition,” in *2018 16th international conference on frontiers in handwriting recognition (ICFHR)*, 2018, pp. 80–85.
[3] L. Guo, D. Zhang, L. Wang, H. Wang, and B. Cui, “CRAN: a hybrid CNN-RNN attention-based model for text classification,” in *International Conference on Conceptual Modeling*, 2018, pp. 571–585.
[4] Y. Hu, Y. Wong, W. Wei, Y. Du, M. Kankanhalli, and W. Geng, “A novel attention-based hybrid CNN-RNN architecture for sEMG-based gesture recognition,” *PloS one*, vol. 13, no. 10, p. e0206049, 2018.
[5] M. Jain, M. Mathew, and C. V. Jawahar, “Unconstrained OCR for Urdu using deep CNN-RNN hybrid networks,” in *2017 4th IAPR Asian Conference on Pattern Recognition (ACPR)*, 2017, pp. 747–752.
[6] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.
[7] G. Damaskinos, R. Guerraoui, R. Patra, and M. Taziki, “Asynchronous Byzantine machine learning (the case of SGD),” in *International Conference on Machine Learning*, 2018, pp. 1145–1154.
[8] X. Yang, X. Zheng, and H. Gao, “SGD-based adaptive NN control design for uncertain nonlinear systems,” *IEEE transactions on neural networks and learning systems*, vol. 29, no. 10, pp. 5071–5083, 2018.
[9] W. Shao, L. Yao, Z. Ge, and Z. Song, “Parallel computing and SGD-based DPMM for soft sensor development with large-scale semisupervised data,” *IEEE Transactions on Industrial Electronics*, vol. 66, no. 8, pp. 6362–6373, 2018.
[10] Y. Zhuang, W.-S. Chin, Y.-C. Juan, and C.-J. Lin, “A fast parallel sgd for matrix factorization in shared memory systems,” in *Proceedings of the 7th ACM conference on Recommender systems*, 2013, pp. 249–256.
[11] Z. Li, F. Zhou, F. Chen, and H. Li, “Meta-sgd: Learning to learn quickly for few-shot learning,” *arXiv preprint arXiv:1707.09835*, 2017.
[12] B. Neyshabur, R. Salakhutdinov, and N. Srebro, “Path-sgd: Path-normalized optimization in deep neural networks,” *arXiv preprint arXiv:1506.02617*, 2015.
[13] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, “On optimization methods for deep learning,” 2011.
[14] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, “Efficient backprop,” in *Neural networks: Tricks of the trade*, Springer, 2012, pp. 9–48.
[15] J. Martens, “Deep learning via hessian-free optimization.,” in *ICML*, 2010, vol. 27, pp. 735–742.
[16] J. Gehring, M. Auli, D. Grangier, and Y. N. Dauphin, “A convolutional encoder model for neural machine translation,” *arXiv preprint arXiv:1611.02344*, 2016.
[17] G. Hinton, “CS231n Convolutional Neural Networks for Visual Recognition,” *hithub*. http://cs231n.github.io/neural-networks-3/ (accessed Mar. 09, 2020).
[18] M. D. Zeiler, “Adadelta: an adaptive learning rate method,” *arXiv preprint arXiv:1212.5701*, 2012.
[19] Y. Bengio, I. Goodfellow, and A. Courville, *Deep learning*, vol. 1. MIT press, 2017.
[20] Y. Bengio, “Practical recommendations for gradient-based training of deep architectures,” in *Neural networks: Tricks of the trade*, Springer, 2012, pp. 437–478.
[21] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, “Greedy layer-wise training of deep networks,” in *Advances in neural information processing systems*, 2007, pp. 153–160.

[22] G. E. Hinton, T. J. Sejnowski, and D. H. Ackley, *Boltzmann machines: Constraint satisfaction networks that learn*. Carnegie-Mellon University, Department of Computer Science Pittsburgh, PA, 1984.
[23] G. E. Hinton, "A practical guide to training restricted Boltzmann machines," in *Neural networks: Tricks of the trade*, Springer, 2012, pp. 599–619.
[24] A. F. Özbek, "How to Train a Model with MNIST dataset," *Medium*, Jan. 28, 2019. https://medium.com/@afozbek_/how-to-train-a-model-with-mnist-dataset-d79f8123ba84 (accessed Feb. 03, 2021).
[25] J. Yosinski and H. Lipson, "Visually debugging restricted boltzmann machine training with a 3d example," 2012.
[26] RBM Visualizations, "Restricted Boltzmann Machine Visualizations," *github*. https://jpatanooga.github.io/Metronome/rbm20140306.html (accessed Jan. 28, 2021).
[27] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of machine learning research*, vol. 11, no. Dec, pp. 3371–3408, 2010.
[28] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.
[29] O. Fabius and J. R. van Amersfoort, "Variational Recurrent Auto-Encoders," *arXiv:1412.6581 [cs, stat]*, Jun. 2015, Accessed: Mar. 13, 2020. [Online]. Available: http://arxiv.org/abs/1412.6581.
[30] R. Calandra, T. Raiko, M. P. Deisenroth, and F. M. Pouzols, "Learning deep belief networks from non-stationary streams," in *International Conference on Artificial Neural Networks*, 2012, pp. 379–386.
[31] I. Goodfellow *et al.*, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.
[32] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," in *Advances in neural information processing systems*, 2016, pp. 613–621.
[33] H. Zhang *et al.*, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5907–5915.
[34] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, 2014, pp. 818–833.
[35] A. Radford, *Newmu/dcgan_code*. 2020.
[36] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.
[37] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. van den Oord, A. Graves, and K. Kavukcuoglu, "Neural machine translation in linear time," *arXiv preprint arXiv:1610.10099*, 2016.
[38] A. A. Helmy, "A multilingual encoding method for text classification and dialect identification using convolutional neural network," *arXiv preprint arXiv:1903.07588*, 2019.
[39] C. Dos Santos and M. Gatti, "Deep convolutional neural networks for sentiment analysis of short texts," in *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, 2014, pp. 69–78.
[40] M. Eickenberg, A. Gramfort, G. Varoquaux, and B. Thirion, "Seeing it all: Convolutional network layers map the function of the human visual system," *NeuroImage*, vol. 152, pp. 184–194, May 2017, doi: 10.1016/j.neuroimage.2016.10.001.
[41] R. Ngestrini, "Predicting Poverty of a Region from Satellite Imagery using CNNs," Master's Thesis, 2019.

[42] A. Saxena, “Convolutional Neural Networks (CNNs): An Illustrated Explanation,” *XRDS*, Jun. 29, 2016. https://blog.xrds.acm.org/2016/06/convolutional-neural-networks-cnns-illustrated-explanation/ (accessed Jan. 31, 2021).
[43] Jose Parreno Garcia, “Intro to CNN,” *github*, Mar. 2018. https://joparga3.github.io/ud_deep_learning_cnns/ (accessed Jan. 31, 2021).
[44] cs231n, “CS231n Convolutional Neural Networks for Visual Recognition,” *Github*. https://cs231n.github.io/convolutional-networks/ (accessed Feb. 03, 2021).
[45] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
[46] D. Kothawade, I. Kazi, J. Mhatre, K. Nair, and P. Nitnaware, “Recognition of Labels for Hand Drawn Images,” 2020.
[47] O. Zapletal, “IMAGE RECOGNITION BY CONVOLUTIONAL NEURAL NETWORKS - BASIC CONCEPTS,” Master’s Thesis, BRNO UNIVERSITY OF TECHNOLOGY, Brno, Czech Republic, 2017.
[48] F. Yu and V. Koltun, “Multi-scale context aggregation by dilated convolutions,” *arXiv preprint arXiv:1511.07122*, 2015.
[49] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, 2015.
[50] T. Cooijmans, N. Ballas, C. Laurent, Ç. Gülçehre, and A. Courville, “Recurrent batch normalization,” *arXiv preprint arXiv:1603.09025*, 2016.
[51] S. Sukrit, “Towards Deep Learning Approaches for Detecting, Ranking and Synthesizing Visual Attributes,” PhD, University of Cambridge, Cambridge, UK, 2017.
[52] F. Milletari, N. Navab, and S.-A. Ahmadi, “V-net: Fully convolutional neural networks for volumetric medical image segmentation,” in *2016 Fourth International Conference on 3D Vision (3DV)*, 2016, pp. 565–571.
[53] D. Maturana and S. Scherer, “Voxnet: A 3d convolutional neural network for real-time object recognition,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 922–928.
[54] M. Henaff, J. Bruna, and Y. LeCun, “Deep convolutional networks on graph-structured data,” *arXiv preprint arXiv:1506.05163*, 2015.
[55] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, “Very deep convolutional networks for text classification,” *arXiv preprint arXiv:1606.01781*, 2016.
[56] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[57] C. Szegedy *et al.*, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.
[58] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
[59] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
[60] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo, “Image captioning with semantic attention,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4651–4659.
[61] A. Graves and N. Jaitly, “Towards end-to-end speech recognition with recurrent neural networks,” in *International conference on machine learning*, 2014, pp. 1764–1772.
[62] A. Nayebi and M. Vitelli, “Gruv: Algorithmic music generation using recurrent neural networks,” *Course CS224D: Deep Learning for Natural Language Processing (Stanford)*, 2015.

[63] T. Mikolov, “Statistical language models based on neural networks,” *Presentation at Google, Mountain View, 2nd April*, vol. 80, 2012.
[64] Andrej Karpathy, “The Unreasonable Effectiveness of Recurrent Neural Networks,” *github*, May 21, 2015. https://karpathy.github.io/2015/05/21/rnn-effectiveness/ (accessed Feb. 17, 2020).
[65] J. Zhi loh and P. Dubs, “Recurrent Neural Network (RNN) implementations in DL4J.,” *deeplearning4j*, 2020. https://deeplearning4j.konduit.ai/models/recurrent (accessed Feb. 01, 2021).
[66] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[67] A. Graves, “Supervised Sequence Labelling with Recurrent Neural Networks [Ph. D. dissertation],” *Technical University of Munich, Germany*, 2008.
[68] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with LSTM,” 1999.
[69] R. Hasani, *raminmh/LSTM_dynamics_interpretability*. 2019.
[70] K. Cho *et al.*, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.
[71] N. Srivastava, E. Mansimov, and R. Salakhudinov, “Unsupervised learning of video representations using lstms,” in *International conference on machine learning*, 2015, pp. 843–852.
[72] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko, “Translating videos to natural language using deep recurrent neural networks,” *arXiv preprint arXiv:1412.4729*, 2014.
[73] Q. Wu, C. Shen, P. Wang, A. Dick, and A. van den Hengel, “Image captioning and visual question answering based on attributes and external knowledge,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1367–1381, 2017.
[74] V. Mnih, N. Heess, and A. Graves, “Recurrent models of visual attention,” in *Advances in neural information processing systems*, 2014, pp. 2204–2212.
[75] R. Socher, C. C. Lin, C. Manning, and A. Y. Ng, “Parsing natural scenes and natural language with recursive neural networks,” in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 129–136.

DOI: 10.30503/nripress.2020.321

فصل ٤- مفاهیم پایه یادگیری ماشین و یادگیری عمیق

DOI: 10.30503/nripress.2020.321

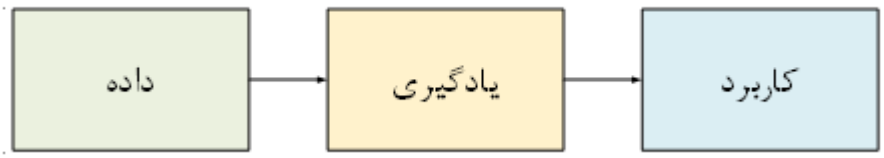
مقدمه

در این فصل در ابتدا مفاهیم پایه یادگیری ماشین بیان می‌شود. سپس مفهوم یادگیری عمیق و انواع ساختارهای مختلف در شبکه‌های عمیق به صورت خلاصه تشریح می‌شود.

٤-١- معرفی هوش مصنوعی، یادگیری ماشین

به طور کلی، قابلیت ادارک و یادگیری ماشین، هوش ماشین تلقی می‌شود. ماشین به معنای هر دستگاه هوشمندی است که دارای پردازنده باشد و بتواند داده‌های ورودی را از محیط پیرامون بگیرد و روی آن‌ها پردازش انجام دهد و تصمیم‌گیری نیز بکند. هدف یادگیری ماشین، شبیه سازی و درک رفتار انسان است. سیستم‌های تشخیص چهره، بازی‌های کامپیوتری، کورتانا[۱] ، نتفلیکس[۲] و بسیاری از سیستم‌هایی که همه روزه از آن‌ها استفاده می‌کنیم، مثال‌هایی از کاربردهای یادگیری ماشین هستند.

یادگیری ماشین، مطالعه‌ی علمی الگوریتم‌ها و مدل‌های آماری مورد استفاده‌ی سیستم‌های کامپیوتری است که بجای استفاده از دستورالعمل‌های واضح از الگوها و استنباط برای انجام وظایف سود می‌برند. به عنوان زیر مجموعه‌ای از هوش مصنوعی، الگوریتم‌های یادگیری ماشین یک مدل ریاضی بر اساس داده‌های نمونه یا "داده‌های آموزش" به منظور پیش‌بینی یا تصمیم‌گیری بدون برنامه‌ریزی آشکار، ایجاد می‌کنند. در تمام روش‌ها و الگوریتم‌های یادگیری ماشین سه مرحله مشترک وجود دارد که در شکل ۴-۱ نشان داده شده است [۱].



شکل ٤-١ : مراحل یادگیری ماشین

مجموعه داده[۳]ها دسته‌ای از اطلاعات با ویژگی‌های مشخص هستند که در یک حوزه‌ی مشخص گرداوری شده است. مجموعه داده به مجموعه‌ای از داده‌ها می‌گویند که با موضوعیت یکسان، جهت انجام تحلیل‌ها و پروژه‌های داده کاوی استفاده می‌شوند. یک کاربرد دیگر مجموعه داده‌ها نیز مقایسه بین روش‌های مختلف است، به این صورت که به طور نمونه بر روی یک مجموعه داده، دو روش(الگوریتم) مختلف را اجرا کرده و با توجه به نتایج بدست آمده و ارزیابی معیارهای دقت، می‌توان سرعت و پیچیدگی هریک از روش‌ها را مقایسه کرد.

به طور خلاصه می‌توان مجموعه داده‌ها را به موارد زیر تقسیم‌بندی کرد:

مجموعه داده متنی (متن مقاله، کتاب، نظرات شبکه‌های اجتماعی، توییت‌ها و ...)

[۱] Cortana

[۲] Netflix

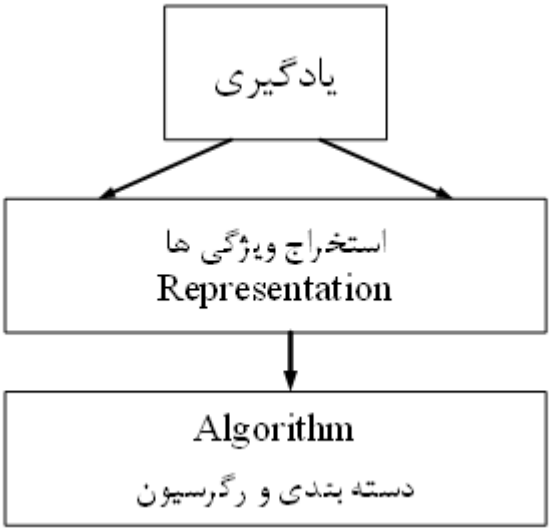
[۳] Data set

DOI: 10.30503/nripress.2020.321

مجموعه داده جدولی (داده‌های موجود در پایگاه داده، خوشه‌بندی، طبقه‌بندی، سری زمانی و ...)

مجموعه داده چندرسانه‌ای (عکس، ویدیو و صوت)

در مرحله‌ی بعدی با استفاده از داده‌های ورودی، الگوریتم یاد می‌گیرد و آموزش داده می‌شود که به هدف مورد نظر برسد. شکل ۴-۲ جزئیات مرحله یادگیری را به تصویر می‌کشد [۲].



شکل ٤–۲: جزئیات مرحله یادگیری

همانطور که از شکل ۴-۲ پیداست، در مرحله استخراج ویژگی‌ها، الگوریتم تلاش می‌کند اطلاعات و ویژگی‌های مفید را از داده‌ها استخراج کند. مثلا در یک تصویر برای تشخیص یک مربع، نیازی نیست که تمام نقاط یک تصویر را وارد الگوریتم کرد. همین که بتوان گوشه‌ها، لبه‌ها و یا زاویه‌ها را تشخیص داد کافی است. در مرحله بعدی، الگوریتم در نهایت با استفاده از اطلاعات و ویژگی‌های مفید ورودی، آموزش داده می‌شود. الگوریتم‌ها وظیفه دارند هر داده ورودی را به کلاس و دسته مربوط به خودش متعلق کنند. به طور کلی، استخراج ویژگی‌ها به دو صورت انجام می‌شود:

- روش دستی:

در این حالت، روش‌های از پیش تعیین شده‌ای که از قبل طراحی شده‌اند، بر روی داده‌ها اعمال می‌شوند و در نتیجه ویژگی‌های مدل بدست می‌آیند. در این قسمت، به صورت دستی، ویژگی‌ها توسط شخص تعیین می‌شوند و روش استخراج نیز توسط شخص تعیین می‌شود و در نهایت این ویژگی‌ها به داده‌های ورودی اعمال می‌شوند.

- روش اتوماتیک:

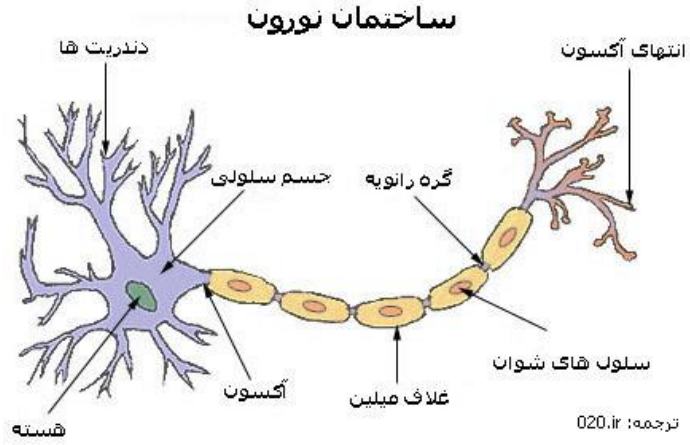
در این حالت، الگوریتم خودش یاد می‌گیرد که چه ویژگی‌هایی خوب هستند و خودش هم یاد می‌گیرد که چطور آنها را استخراج کند. مباحث موجود در یادگیری عمیق به استخراج اتوماتیک ویژگی‌ها ارتباط دارد.

در مرحله‌ی آخر، بعد از اتمام یادگیری و کامل شدن الگوریتم، از آن در کاربردهای مورد نظر استفاده می‌شود.

DOI: 10.30503/nripress.2020.321

٤-٢- شبکه‌های عصبی

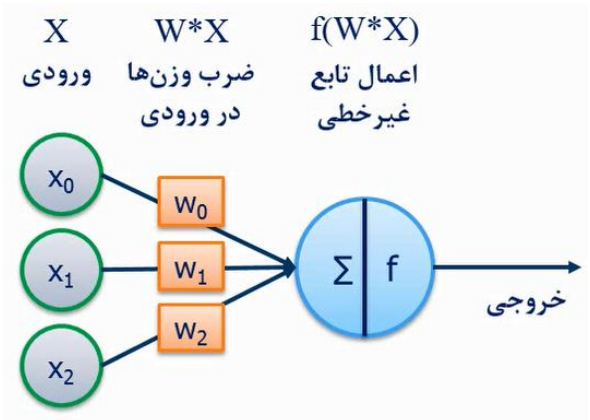
به طور کلی شبکه‌های عصبی مصنوعی برگرفته از شبکه عصبی انسان هستند. یک نمونه از شبکه عصبی انسان در شکل ۴-۳ نشان داده شده است.



شکل ٤-٣: شبکه عصبی انسان

در شبکه عصبی انسان، یکسری واحدهای عملیاتی به اسم سلول‌ها یا نورون‌های عصبی وجود دارند. در این سلول‌های عصبی، اطلاعات یا داده‌ها از طریق شاخه‌هایی به نام دندریت از سلول‌های مختلف وارد هسته سلول می‌شوند. در هسته سلول تمام ورودی‌ها با یکدیگر جمع می‌شوند و به اصلاح یک پردازشی روی آنها انجام می‌شود و پالس جدید تولید می‌شود که از طریق شاخه‌ای به اسم اکسون از هسته سلول خارج می‌شود و وارد سلول‌های دیگر می‌شود.

در یک شبکه مصنوعی، سه مرحله ورودی، وزن‌ها و تابع غیرخطی وجود دارند که به این سه مرحله یک لایه گفته می‌شود. وظیفه این است که این لایه آموزش داده شود. آموزش به این صورت است که وزن‌هایی پیدا شود که ورودی را به خروجی موردنظر تبدیل کند. پس می‌توان نتیجه گرفت که این وزن‌ها هستند که آموزش داده می‌شوند. شکل ۴-۴ شماتیک یک لایه از شبکه مصنوعی را نشان می‌دهد.

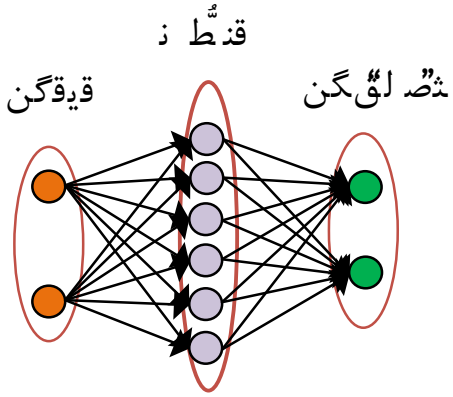


شکل ٤-٤ : شماتیک یک لایه از یک شبکه عصبی مصنوعی

یک ورودی سه بعدی می‌تواند با وزن‌های متفاوت به چهار نورون دیگر متصل شود که در شکل ۴-۵ نشان داده شده است. این فرآیند شبیه فرآیند استخراج ویژگی است. همانطور که نشان داده شده است، ورودی از یک فضای سه

DOI: 10.30503/nripress.2020.321

بعدی به یکسری ویژگی تبدیل می‌شود، سپس ویژگی‌ها به عنوان ورودی الگوریتم مورد استفاده قرار می‌گیرد. به عبارت دیگر، لایه اول، ورودی را به لایه میانی متصل می‌کند و ویژگی‌ها را مشخص می‌کند و لایه دوم، ویژگی‌ها را به دسته‌های مربوط به خودش متعلق می‌کند.



شکل ٤–٥: شماتیک یک شبکه عصبی دو لایه

٤-٣- یادگیری عمیق[1] (DL)

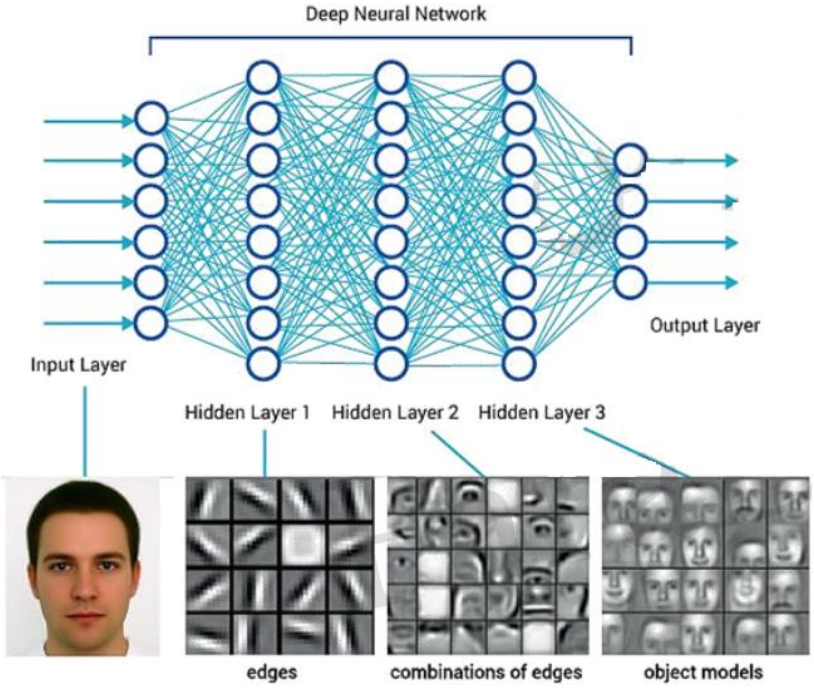
یادگیری عمیق عبارت است از ساخت مدل‌های یادگیری ماشین که برای یادگیری نمایشی سلسله‌مراتبی از داده‌ها به کار می‌روند. شبکه‌های عصبی عمیق، یک اصطلاح کلی برای مجموعه‌ای از شبکه‌های عصبی با معماری چند لایه است که نشان می‌دهند چگونه شبکه‌های عصبی با تعداد زیادی از لایه‌ها می‌توانند در ایجاد ساختارهای بازنمایی مورد نیاز در یادگیری عمیق موفق عمل کنند. الگوریتم‌های یادگیری ویژگی به صورت با ناظر و بدون ناظر می‌توانند در تنظیم وزنه‌ای این شبکه‌ها به کار گرفته شوند. شبکه‌های عصبی عمیق، با ترکیب چندین تبدیل غیرخطی شکل می‌گیرند که هدف آنها دستیابی به انتزاع بیشتر و در نهایت بازنمایی‌های سودمند از داده‌های موجود است. هر چه تعداد لایه‌ها در شبکه عصبی بیشتر شود، مسئله بهینه‌سازی پیچیده‌تر می‌شود. به همین دلیل یکی از روش‌های آموزش این نوع از شبکه‌ها با استفاده از الگوریتم‌های پیش‌آموزشی بدون ناظر انجام می‌گیرد. در این روش ابتدا هر لایه به صورت مجزا آموزش داده می‌شود و در نهایت روی کل شبکه تنظیم دقیق و یکپارچه‌ای انجام می‌گیرد. به طور کلی، شبکه عصبی همانند سلولهای عصبی در مغز انسان عمل می‌کند تا کارهای مختلف محاسباتی را سریعتر انجام دهد در حالی که یادگیری عمیق نوع خاصی از یادگیری ماشینی است که از روش یادگیری که انسان برای کسب دانش استفاده می کند ، تقلید می‌کند [۳].

به طور خلاصه، چهار تفاوت اصلی بین شبکه یادگیری عمیق و شبکه‌های غیر عمیق شامل موارد زیر است: ۱- تعداد بیشتری نرون نسبت به گذشته ۲- روش‌های پیچیده‌تری برای اتصال لایه‌ها ۳- نیاز به توان پردازشی فوق‌العاده برای آموزش شبکه ۴- استخراج اتوماتیک ویژگی. در شکل ۴-۶، یک شبکه عصبی عمیق نشان داده شده است که شامل یک لایه ورودی، سه لایه میانی و یک لایه خروجی است. همانطور که نشان داده شده است، ورودی یک تصویر

[1] Deep learning (DL)

DOI: 10.30503/nripress.2020.321

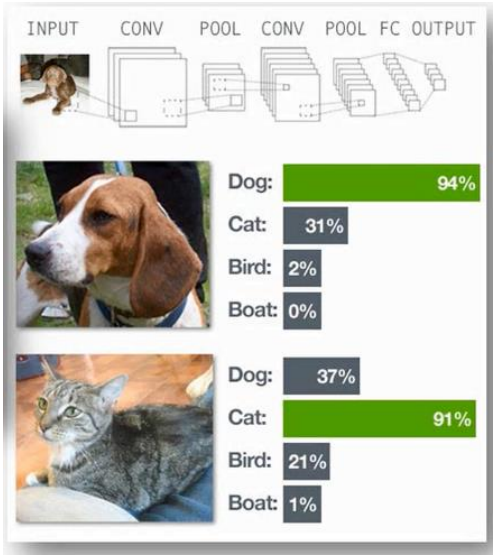
صورت انسان است و خروجی کلاس مربوط به تصویر است که می‌تواند چهار تا کلاس مختلف را نشان دهد. به عنوان مثال، کلاس‌ها می‌توانند تصویر مرد، زن، بچه و یا پیر باشند. تصویر ورودی به صورت پیسکل وارد شبکه می‌شود، به این معنی که تصویر به صورت برداری از پیکسل‌ها وارد شبکه می‌شود. در لایه اول شبکه یاد می‌گیرد که ویژگی‌های سطح پایین یعنی لبه‌ها و گوشه‌ها را تشخیص دهد. به عبارت دیگر، مدل استخراج ویژگی، یک شبکه عصبی عمیق است که با در نظر گرفتن یک تصویر قادر به استخراج ویژگی های برجسته، اغلب به شکل یک بردار با طول ثابت است. بعد این ویژگی‌ها به عنوان ورودی به لایه دوم استفاده می‌شوند. در لایه دوم، ویژگی‌های پیچیده‌تر یعنی ترکیب گوشه‌ها و لبه‌ها تشخیص داده می‌شوند (اجزای مختلف صورت). مجددا، ویژگی‌های جدید به عنوان ورودی به لایه سوم مورد استفاده قرار می‌گیرند. در لایه سوم، ویژگی‌های پیچیده‌تر از لایه دوم یعنی مدل‌های مختلف صورت آموزش داده می‌شوند. به این فرآیند، یادگیری چند لایه ویژگی‌ها گفته می‌شود. دسترسی همزمان به هر سه لایه ویژگی‌ها، یکی از مزیت‌های این طرح به شمار می رود. همچنین، طبقه‌بند قادر است که با قدرت خیلی بیشتری تشخیص بدهد که کلاس تصویر ورودی مربوط است به تصویر مرد یا زن یا بچه و یا پیر.



شکل ٤–٦: شماتیک یک شبکه عصبی عمیق

شکل ۴-۷ نمونه‌ای از شبکه عصبی کانولوشنی عمیق را به تصویر می‌کشد. این شبکه با داده‌های تصویری آموزش دیده است. همانطور که پیداست، ورودی این شبکه یک عکس و خروجی چهار تا کلاس مختلف است. کلاس‌ها شامل کلاس سگ، گربه، پرنده و قایق هستند. وظیفه‌ی این شبکه این است که تشخیص بدهد که تصویر ورودی با چه درصدی مربوط به این چهار تا کلاس است. هر کلاسی که بیشترین نمره و یا درصد را آورد به عنوان کلاس تصویر ورودی شناسایی می‌شود.

DOI: 10.30503/nripress.2020.321



شکل ٤-٧: شماتیک یک شبکه عصبی کانولوشنی عمیق

٤-٤- محبوبیت، مزایا و چالش‌های یادگیری عمیق

محبوبیت یادگیری عمیق از سال ۲۰۰۶ با شروع رفع مشکلات مربوط به یادگیری که در ذیل معرفی شده‌اند، آغاز شد.

- تولید میلیون‌ها داده برچسب‌دار در اینترنت
- پیشرفت سخت افزاری و استفاده از پردازنده‌های گرافیکی
- ابداع تکنیک‌های آموزش

همچنین، کار در حوزه یادگیری عمیق از سال ۲۰۱۲ در برخی شرکت‌ها و دانشگاهای بزرگ که در ذیل معرفی شده‌اند، آغاز شد.

- شرکت‌های بزرگ مانند: Google، Microsoft، Twitter، و غیره
- دانشگاه‌های بزرگ مانند: Berkeley، Oxford، Stanford، و غیره

به عنوان مزایای یادگیری عمیق می‌توان به موارد زیر اشاره کرد: ۱- یادگیری خودکار ویژگی‌ها ۲- یادگیری چند لایه ویژگی‌ها ۳- دقت بالا ۴- پشتیبانی سخت افزاری و نرم افزاری ۵- قدرت تعمیم بالا ۶- پتانسیل ایجاد قابلیت‌های بیشتر

همچنین، به عنوان چالش‌های یادگیری عمیق می توان به موارد زیر اشاره کرد: ۱- پشتوانه تئوری ضعیف ۲- هزینه محاسباتی بالا ۳- نیاز به تعداد زیاد داده ۴- مشکلات آموزش ۵- دشواری تنظیم پارامترها [۴].

٤-٥- روش‌های یادگیری عمیق

انواع روش‌های یادگیری عمیق در شکل ۴-۸ بیان شده است [۵].

DOI: 10.30503/nripress.2020.321

شکل ٤-٨: انواع روش‌های یادگیری عمیق

٤-٥-١- یادگیری بدون نظارت[١]

فرآیند یادگیری ماشین بدون راهنمایی انسان، به این صورت که یک مسئله به یک الگوریتم داده می‌شود، اما جواب اصلی مسئله به آن داده نمی‌شود و انتظار می‌رود که فرآیند آموزش و یادگیری برای حل مساله توسط ماشین انجام شود. الگوریتم‌های مصورسازی یکی از مثال‌های یادگیری بدون نظارت هستند. به عنوان مثال تعداد بسیار زیادی عکس به یک شبکه عصبی بدون نظارت داده می‌شود و هیچ برچسبی هم بر روی عکس‌ها وجود ندارد. الگوریتم مورد نظر تلاش می‌کند تا ساختاری میان آن‌ها پیدا کند و آن‌ها را خوشه‌بندی کند. درنهایت الگوریتم متوجه می‌شود که چطور داده را سازماندهی کند.

به عنوان روش‌های متعارف در یادگیری بدون نظارت می‌توان به موارد زیر اشاره کرد.

- خوشه‌بندی کامینز
- PCA

همچنین، انواع شبکه‌های عصبی در یادگیری بدون نظارت می‌توان به موارد زیر اشاره کرد.

- شبکه‌های عصبی خودرمزنگار[٢]
- شبکه‌های عصبی مولدی[٣]

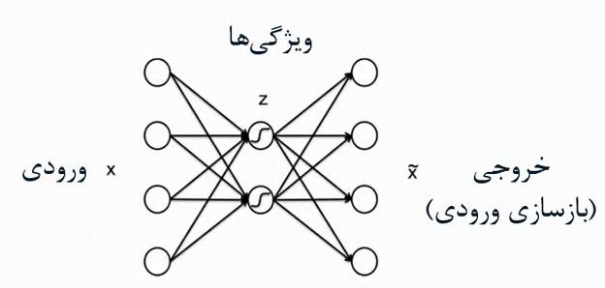
٤-٥-١-١- شبکه های عصبی خودرمزنگار

شبکه‌های خودرمزنگار همان شبکه‌های پیشخور هستند که دارای بایاس اضافه برای محاسبه خطا در ساخت مجدد ورودی هستند. بعد از آموزش، شبکه‌های خودرمزنگار را می‌توان همانند یک شبکه فیدفوروارد برای ورود به تابع

[١] Unsupervised learning
[٢] Autoencoder
[٣] Generative adversarial networks

DOI: 10.30503/nripress.2020.321

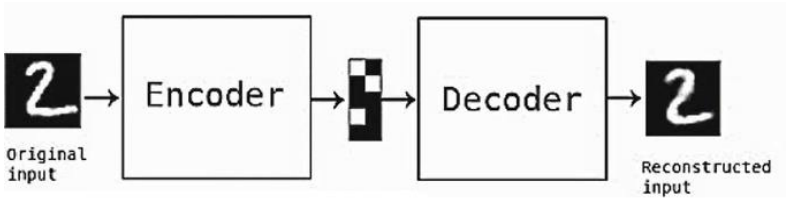
فعال‌سازی استفاده کرد. این یک شکل از استخراج ویژگی غیر نظارتی است که برای وزن‌های یادگیری فقط از داده‌های ورودی برچسب خورده استفاده می‌کند. شکل ۴-۹ ساختار کلی یک شبکه عصبی خود رمزنگار را نشان می‌دهد.



شکل ٤–۹: ساختار یک شبکه خودرمزنگار

همانطور که پیداست، خروجی همان بازسازی ورودی است. در مرحله اول، ورودی‌ها انکودر می‌شوند به چند ویژگی، بعد این ویژگی‌ها به داده اولیه دکودر می‌شوند.

شکل ۴-۱۰ نمونه‌ای دیگر از یک شبکه عصبی خودرمزنگار را نشان می‌دهد. همانطور که پیداست، تصویر یک عدد دست‌نویس به یک شبکه انکودر داده می‌شود و تبدیل می‌شود به یک سری کد و ویژگی و اگر این ویژگی‌ها به یک شبکه دکودر داده شود، باید همان تصویر را بازسازی کند.



شکل ٤–۱۰: ساختار یک شبکه خودرمزنگار با یک ورودی عدد دست نویس

٤-٥-١-٢- شبکه های عصبی مولدی

شبکه عصبی مولدی رویکردی برای مدل‌سازی مولد با استفاده از روش‌های «یادگیری عمیق» مانند شبکه‌های عصبی کانولوشنی است. مدل‌سازی مولد یک فعالیت نظارت نشده در یادگیری ماشین است که شامل اکتشاف خودکار و یادگیری قواعد یا الگوهای موجود در داده‌های ورودی می‌شود. این کار به صورتی انجام می‌شود که از مدل می‌توان برای تولید یا خروجی دادن نمونه‌های جدیدی که به شکل قابل باوری از مجموعه داده اصلی قابل برگرفته شدن هستند، استفاده کرد [۷]. شبکه‌های مولدی راهکاری هوشمندانه برای آموزش دادن یک مدل مولد هستند. آن‌ها این کار را با قاب‌بندی مساله به عنوان یک مساله یادگیری نظارت شده با دو زیر مدل انجام می‌دهند. این دو زیر مدل عبارتند از مدل مولد که برای تولید نمونه‌های جدید آموزش داده می‌شود و مدل متمایزگر که تلاش می‌کند تا نمونه‌ها را به عنوان نمونه واقعی (از دامنه) یا جعلی (تولید شده) دسته‌بندی کند. هر دو مدل با یکدیگر در یک بازی مجموع صفر تخاصمی آموزش داده می‌شوند و این کار تا جایی ادامه پیدا می‌کند که مدل متمایزگر بالغ بر نیمی از دفعات گول بخورد؛ بدین معنا که مدل مولد، نمونه‌های قابل باور تولید کرده است.

DOI: 10.30503/nripress.2020.321

شبکه‌های عصبی مولد، یک زمینه مهیج و به سرعت در حال رشد هستند که وعده خود مبنی بر داشتن توانایی تولید نمونه‌های واقعی در طیف گوناگونی از مسائل را محقق کرده‌اند. از جمله برجسته‌ترین این مسائل، «تبدیل تصویر به تصویر» (Image to Image Translation) است. به عنوان مثالی از تبدیل تصویر به تصویر، می‌توان به تبدیل تصاویر تابستان به زمستان، تصاویر روز به شب و تولید تصاویر فوتورئالیستیک از اشیا، صحنه‌ها و افراد اشاره کرد. این تصاویر به گونه‌ای تولید می‌شوند که حتی انسان‌ها نیز نمی‌توانند بگویند تصاویر جعلی هستند. متقاعد کننده‌ترین دلیل برای اینکه شبکه‌های عصبی مولدی به طور گسترده مورد مطالعه، توسعه و استفاده قرار می‌گیرند، موفقیت‌های آن‌ها است. آن‌ها قادر به تولید کردن تصاویر بسیار واقعی هستند که انسان‌ها نیز قادر به تشخیص و بیان آن نیستند که این تصاویر متعلق به اشیا، صحنه‌ها و انسان‌هایی هستند که در دنیای واقعی وجود خارجی ندارند. مثالی از پیشرفت توانایی‌های شبکه‌های عصبی مولدی از سال ۲۰۱۴ تا ۲۰۱۷ در شکل ۴-۱۱ نشان داده شده است.

شکل ٤–١١: پیشرفت توانایی‌های شبکه های عصبی مولدی در طول ۴ سال [۷]

٤-٥-٢- یادگیری با نظارت[1]

یادگیری با نظارت، فرآیند یادگیری ماشین با راهنمایی انسان تعریف می‌شود. یعنی علاوه بر دادن مسئله به الگوریتم، جواب صحیح نیز داده می‌شود و الگوریتم تلاش می‌کند که ویژگی‌ها را از داده‌های برچسب یاد بگیرد و به جواب تعیین شده برسد. یکی از مثال‌های مرسوم در یادگیری با نظارت تشخیص و فیلتر کردن هرزنامه‌ها میان پیام‌ها است. ابتدا تمامی داده‌ها به دو کلاس سالم و هرزنامه تقسیم می‌شوند، سپس ماشین آن‌ها را با مثال‌های موجود می‌آموزد در نهایت از او امتحان گرفته می‌شود و امتحان به این منظور تلقی می‌شود که شما ایمیل جدیدی که تا به حال ندیده است را به آن بدهید، سپس آن تشخیص دهد که سالم یا هرزنامه است.

به عنوان روش‌های متعارف در یادگیری با نظارت می‌توان به موارد زیر اشاره کرد.

- ماشین بردار پشتیبان[2]

همچنین، انواع شبکه های عصبی در یادگیری با نظارت می‌توان به موارد زیر اشاره کرد.

- شبکه‌های عصبی بازگشتی[3]
- شبکه‌های عصبی کانولوشنی

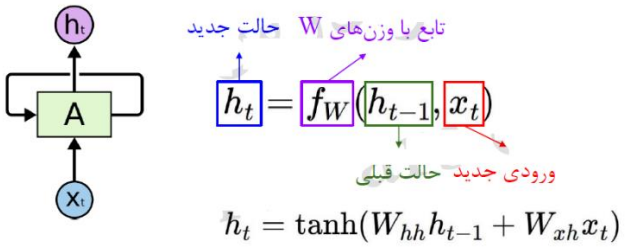
[1] Supervised learning
[1] Support vector machine
[2] Recurrent neural network

DOI: 10.30503/nripress.2020.321

٤-٥-٢-١- شبکه‌های عصبی بازگشتی

شبکه‌های عصبی بازگشتی در سال ۱۹۸۰ ایجاد شدند اما تنها در چند سال اخیر بوده است که این گونه از شبکه‌ها بطور گسترده مورد استفاده قرار گرفته‌اند. از دلایل عمده وقوع چنین رخدادی میتوان به پیشرفت‌های صورت گرفته در طراحی شبکه‌های عصبی بطور عام و بهبود چشمگیر قدرت محاسباتی و بطور ویژه بهره‌وری از قدرت واحدهای پردازش موازی کارتهای گرافیک اشاره نمود. این گونه از شبکه‌های عصبی بطور خاص برای پردازش داده‌های سری یا دنباله دار مفید هستند و در آن‌ها هر نورون یا واحد پردازشی قادر به حفظ حالت داخلی یا همان حافظه به منظور حفظ اطلاعات مرتبط با ورودی قبلی می‌باشد. این ویژگی بطور ویژه در کاربردهای مختلف مرتبط با داده‌های سری اهمیت اساسی پیدا می‌کند. بطور مثال در پردازش زبان طبیعی, جمله‌ای همانند "I had washed my car" معنای متفاوتی از جمله I had my car washed دارد. در این دو جمله همه کلمات یکسان هستند اما معنای مرتب با هرکدام کاملا متفاوت است. در جمله اول ما میگوییم "من خودروی سواری خود را شسته‌ام" در حالی که در جمله دوم ما با جمله "دستور دادم تا خودروی سواری‌ام شسته شود (فرد یا گروه دیگری با دستور من خودرو سواری را شستند). این ویژگی حفظ حالت درونی یا همان قابلیت حافظه به شبکه کمک می‌کند تا قادر به فهم و کشف ارتباط بین لغات مختلف در دنباله‌های طولانی تر باشد. نام این شبکه عصبی از این واقعیت بدست می‌آید که این نوع از شبکه‌ها بصورت بازگشتی عمل می‌کنند. یعنی یک عملیات برای تک تک المانهای یک دنباله (کلمه، جمله،...) انجام می‌گیرید و خروجی آن وابسته به ورودی فعلی و عملیات‌های قبلی است. این مهم از طریق تکرار یک خروجی از شبکه در زمان t با ورودی شبکه در زمان $t+1$ انجام می‌شود.(یعنی خروجی از مرحله قبل با ورودی تازه در مرحله جدید ترکیب می‌شوند.) این چرخه‌ها اجازه وجود اطلاعات از یک گام زمانی به گام زمانی بعدی را موجب می‌شوند. به عبارت بهتر این نوع شبکه‌ها دارای حلقه‌ای در درون خود هستند که بوسیله آن می‌توانند اطلاعات را در حین خواندن ورودی از نورون‌ها عبور دهند [۷].

ورودی‌ها و حالت‌ها در یک شبکه عصبی بازگشتی در شکل ۴-۱۲ نشان داده شده است.

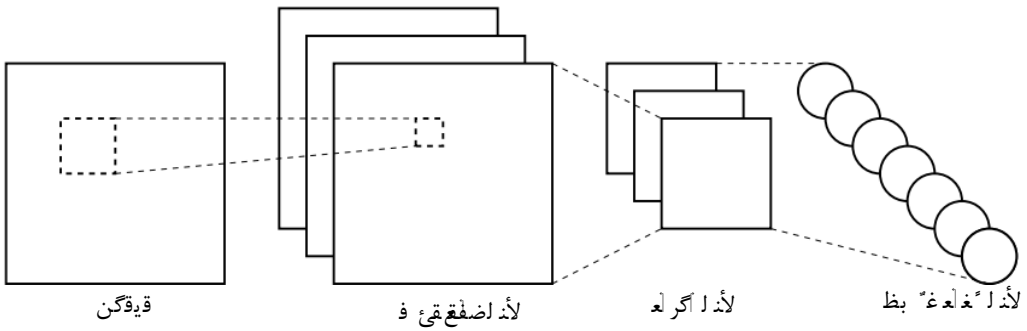


شکل ٤-۱۲: شماتیک یک شبکه عصبی بازگشتی

٤-٥-٢-٢- شبکه‌های عصبی کانولوشنی

DOI: 10.30503/nripress.2020.321

شبکه‌های عصبی کانولوشنی رده‌ای از شبکه‌های عصبی عمیق هستند که معمولاً برای انجام تحلیل‌های تصویری یا گفتاری در یادگیری ماشین استفاده می‌شوند [۸]. شبکه‌های عصبی کانولوشنی عموما از لایه‌های زیر که در شکل ۴-۱۳ نشان داده شده‌اند، تشکیل می‌شوند.



شکل ٤–۱۳: لایه‌های شبکه عصبی کانولوشنی

لایه کانولوشنی از فیلترهایی استفاده می‌کند که عملیات کانولوشنی را در هنگام پویش تصویر ورودی به نسبت ابعادش اجرا می‌کند و می‌تواند ویژگی‌های جدیدی را از تصویر ورودی استخراج کند. خروجی حاصل‌شده نگاشت ویژگی یا نگاشت فعال‌سازی نامیده می‌شود. دومین لایه، لایه ادغام[۱] است. یک لایه ادغام معمولا بعد از یک لایه کانولوشن قرار می‌گیرد و از آن برای کاهش اندازه‌ی ویژگی‌ها و پارامترهای شبکه می‌توان استفاده کرد. لایه‌ی تمام متصل بر روی یک ورودی مسطح به طوری که هر ورودی به تمامی نورون‌ها متصل است، عمل می‌کند. در صورت وجود، لایه‌های تمام متصل معمولا در انتهای معماری‌های شبکه عصبی کانولوشنی یافت می‌شوند و می‌توان آن‌ها را برای بهینه‌سازی اهدافی مثل امتیازات کلاس به کار برد.

٤-٥-۳- یادگیری نیمه نظارتی[۲]

در روش یادگیری نیمه نظارت شده، فرض بر این است که داده‌های آموزشی به دو دسته‌ی داده‌های برچسب خورده و داده‌های برچسب نخورده (یا بدون برچسب) تقسیم می‌شوند. به طور کلی، نمونه‌های برچسب‌دار غالبا مشکل، پرهزینه یا وقت‌گیر هستند. ضمن اینکه جمع‌آوری داده‌های بدون برچسب نسبتاً آسان است اما راه‌های استفاده از آنها محدود می‌باشند. یادگیری نیمه نظارتی این مشکل را با استفاده از مقادیر زیاد داده‌های بدون برچسب همراه با داده‌های برچسب دار برطرف می‌کند. از آن جایی که یادگیری نیمه نظارتی به تلاش انسانی کمتری نیاز دارد و در عین حال دقت بیشتری دارد از این رو از نظر تئوری و علمی بسیار قابل توجه است. روش‌های یادگیری نیمه‌نظارتی را در یک دسته‌بندی کلی به دسته‌های زیر می‌توان تقسیم کرد. ۱- روش‌های مولد ۲- روش‌های مبتنی بر فرض جداسازی کم چگالی ۳- روش‌های مبتنی برگراف.

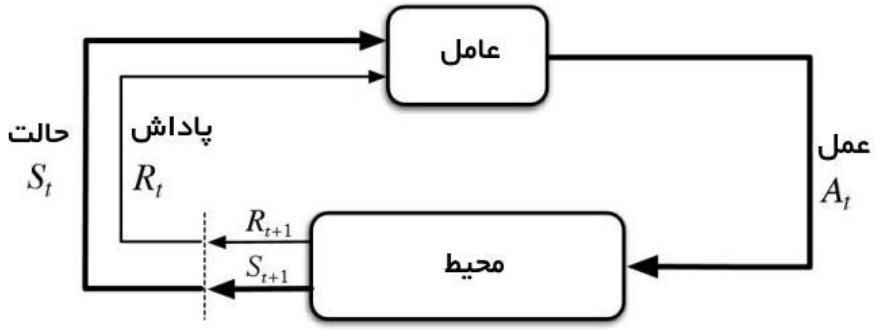
[۱] Pooling layer

[۱] Semi-supervised learning

DOI: 10.30503/nripress.2020.321

۴-۵-۴- یادگیری تقویتی[۱]

یادگیری تقویتی از یادگیری نظارتی یا غیر نظارتی کلی‌تر است. مولفه‌ای به نام عامل[۲] وجود دارد که عملا در حال یادگیری است و حاصل این یادگیری در رفتار مدل نمایان می‌شود. ساختار کلی یادگیری تقویتی در شکل ۲-۱۴ نشان داده شده است، همانطور که پیداست، یک عامل اقدامات را در یک محیط انجام می‌دهد، سپس این اقدام تولیدی توسط عامل، به صورت یک پاداش[۳] و یک حالت[۴] دوباره به عامل به عنوان ورودی داده می‌شود. برای پیاده‌سازی یادگیری تقویتی، الگوریتم‌های مختلفی مانند یادگیری Q، مونته کارلو و غیره وجود دارند [۹].



شکل ٤–١٤: ساختار کلی یک RL

یک شبکه‌ی یادگیری تقویتی عمیق اصول و رویکردهای یادگیری عمیق و یادگیری تقویتی را به کار می‌گیرد. روش‌های یادگیری عمیق Q و گرادیان سیاست[۵] جزو الگوریتم‌های پیاده‌سازی یادگیری تقویتی عمیق هستند. در روش گرادیان سیاست یک سیاست صریح و یک رویکرد اصولی وجود دارد که مستقیماً پاداش مورد انتظار را بهینه می‌کند. امروزه رایانه‌ها می‌توانند به طور خودکار یاد بگیرند که بازهای آتاری را انجام دهند. فرض کنیم می‌خواهیم یک شبکه عصبی را آموزش دهیم که بازی آتاری پانگ را انجام دهد.

بازی پانگ به شرح زیر عمل می‌کند: یک قاب تصویر (یک آرایه ۲۱۰*۱۶۰*۳ بایتی (مقادیر صحیح از ۰ تا ۲۵۵ که نشان دهنده‌ی مقادیر پیکسل هستند)) دریافت می‌کند و تصمیم می‌گیرد که آیا می‌خواهد عملگر بازی به سمت بالا برود یا به سمت پایین برود. بعد از هر انتخاب مجزا، شبیه‌ساز بازی اقدام به عمل می‌کند و یک پاداش تولید می‌شود: اگر توپ از حریف عبور کرد، پاداش است ۱+ و اگر حریف توپ را عبور دهد پاداش است ۱- . یک شماتیک از بازی پانگ در شکل ۴-۱۵ نشان داده شده است.

---

[۲] Reinforcement learning

[۳] Agent

[۴] Reward

[۵] State

[۱] Policy gradients

DOI: 10.30503/nripress.2020.321

شکل ٤–١٥: شماتیک بازی پانگ [١٠]

چیزی که در تنظیمات یادگیری با نظارت برای آموزش شبکه عصبی بازی پانگ نیاز است، این است که یک بازی‌کننده خوب انسانی برای چندین ساعت بازی کند و داده تولید کند. داده می‌تواند شامل اقدام جهت بالا رفتن یا اقدام جهت پایین رفتن باشد. سپس این داده وارد شبکه عصبی می‌شود و درنهایت این شبکه خروجی را تولید می‌کند که می‌تواند اقدام بالا یا اقدام پایین باشد. اما دو چالش اصلی در هنگام استفاده از یادگیری با نظارت وجود دارد: ١- باید یک مجموعه داده وجود داشته باشد که همیشه این یک کار آسان نیست. ٢- اگر می‌خواهیم که یک شبکه عصبی را آموزش دهیم که به سادگی اقدامات بالا رفتن یا پایین رفتن بازی‌کننده انسانی را تقلید کند، هیچگاه این شبکه نمی‌تواند بازی را از بازی‌کننده انسانی بهتر انجام دهد. روشی برای این مشکل وجود دارد که تحت آن عامل بتواند بازی را کاملا به وسیله‌ی خودش انجام دهد و آن روش یادگیری تقویتی است.

چهارچوب یادگیری تقویتی در واقع شبیه به چهارچوب یادگیری با نظارت است. ما هنوز یک قاب ورودی از تصویر را داریم که آن را به مدل شبکه عصبی می‌دهیم و خروجی که تولید می‌کند می‌تواند اقدام جهت بالا رفتن یا اقدام جهت پایین رفتن باشد. اما تنها اختلافی که وجود دارد این است که ما نمی‌دانیم که باید عملگر بازی به سمت بالا یا به سمت پایین حرکت کند، به این دلیل که مجموعه داده‌ای وجود ندارد که با توجه به این داده مدل را آموزش داد.

به این منظور، ما قصد داریم یک شبکه تحت عنوان شبکه‌ی سیاست[١] را تعریف کنیم که عامل یا بازیکن را پیاده‌سازی می‌کند. این شبکه هر بار یک حالت از بازی را می‌گیرد و تصمیم می‌گیرد که به سمت بالا خروجی یا به سمت پایین خروجی تولید کند، سپس آن اقدام تولیدی را دوباره به موتور بازی برمی‌گرداند و موتور بازی، قاب بعدی از تصویر را تولید می‌کند واین حلقه ادامه می‌یابد.

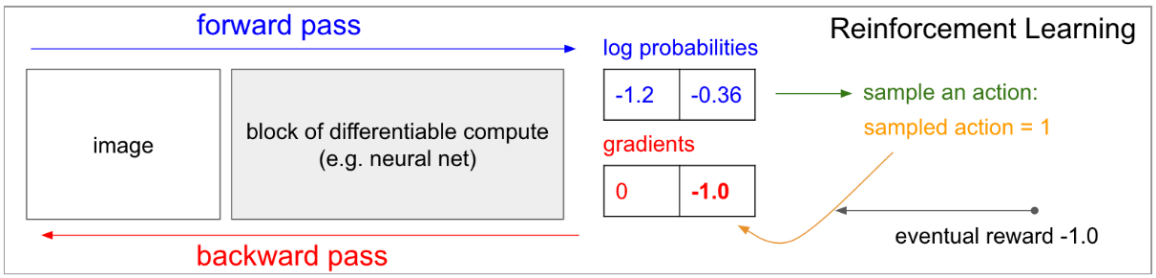
از آنجا که می‌خواهیم عامل تنها به وسیله‌ی خودش آموزش ببیند، علاوه بر قاب بازی، یک فیدبک دیگر نیز دریافت می‌کند و آن اسکوربورد بازی است، هر زمان که عامل مدیریت می‌کند که گل بزند یک پاداش ١+ دریافت می‌کند و اگر حریف مقابل یک گل بزند سپس عامل یک پاداش ١- دریافت می‌کند. به طور کلی هدف عامل این است که بتواند تا جای ممکن پاداش ١+ را دریافت کند.

خروجی یک شبکه سیاست از دو احتمال اقدام بالا و احتمال اقدام پایین تشکیل شده است. به عنوان مثال:

With 30% chance >>>>>> go up
With 70% chance >>>>>> go down

[١] Policy network

هدف از این کار این است که اقدامات دقیقا یکسان را در هر تکرار تولید نکند و این به عامل اجازه می‌دهد که کمی به صورت تصادفی از محیط جستجو کند و در نتیجه بتواند رفتار بهتری را به وجود آورد. فرض شود که شبکه اقدام خروجی پایین را تولید کرده است. سپس یک گرادیان عدد ۱ روی احتمال اقدام پایین وارد می‌شود. سپس با پیاده کردن الگوریتم پس انتشار خطا[۱] بردار گرادیان بدست می‌آید که شبکه را تشویق می‌کند که در آینده کمی به احتمال زیاد اقدام پایین را انجام دهد. اما مشکل این است که حداقل در حال حاضر ما هنوز نمی‌دانیم که آیا پایین رفتن خوب است یا خیر. اما نکته مهم این است که می توانیم کمی صبر کنیم و ببینیم! به عنوان مثال در بازی پانگ می‌توانیم منتظر پایان بازی باشیم و ببینیم که شبکه پاداش ۱+ را دریافت می‌کند یا خیر. به عنوان مثال، اگر اقدام پایین رفتن منجر به باختن بازی از سوی عامل شود، با وارد کردن گرادیان عدد ۱- روی احتمال اقدام پایین و پیاده کردن روش پس انتشار خطا و بدست آوردن بردار گرادیان، شبکه دلسرد می‌شود که در آینده اقدام پایین را برای آن ورودی در آینده انجام دهد (شکل ۴-۱۶). به این روش، روش گرادیان سیاست گفته می‌شود [۱۰].

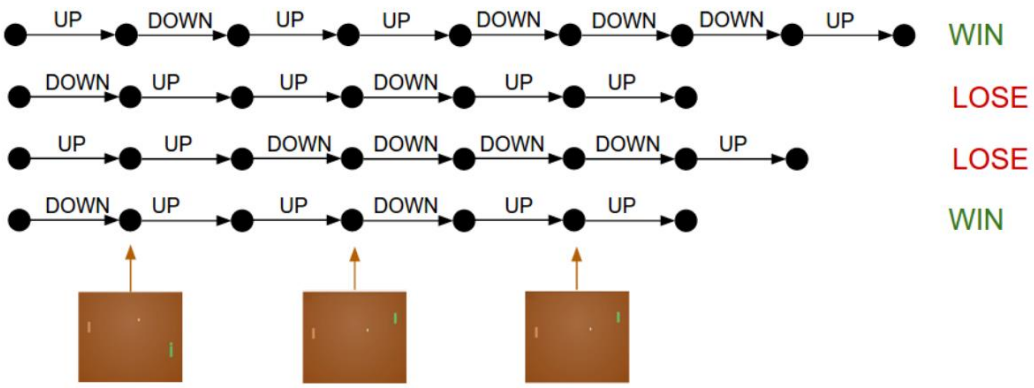


شکل ٤–١٦ : بیان تصویری روش گرادیان سیاست [۱۰]

فرض شود که ۱۰۰ بازی پانگ انجام شده است که هر بازی از ۲۰۰ قاب بازی تشکیل شده است. بنابراین می‌تواند نتیجه گرفت که ۲۰۰۰۰ تصمیم برای اقدام بالا و پایین از سوی شبکه سیاست گرفته شده است. در حال حاضر، چون عامل هنوز چیز مفیدی یاد نگرفته است، ممکن است بیشتر بازی‌ها را ببازد. اما گاهی اوقات عامل ممکن است که خوش شانس باشد، یعنی ممکن است به طور تصادفی یک مجموعه توالی اقدامات انتخاب کند که به گل منجر شود. به طور مثال، فرض شود که عامل، ۱۲ بازی را برده است و ۸۸ بازی را باخته است. در این صورت گرادیان سیاست همه ۲۰۰*۱۲ تصمیمی که در طول بازی‌هایی که برده، گرفته است را یک بروزرسانی مثبت انجام می‌دهد. بروزرسانی مثبت به این معنی است که یک گرادیان ۱+ را روی احتمال آن اقدامات وارد می‌کند، پس انتشار خطا را انجام می‌دهد، بردار گرادیان را بدست می‌آورد و در نهایت شبکه تشویق می‌شود که آن اقداماتی را که در طول ۱۲ بازی انجام داده است، در آینده برای همان قاب‌های تصویر ورودی تکرار کند. دیاگرام کارتونی ۴ بازی پانگ در شکل ۴-۱۷ نشان داده شده است. به عنوان یک نتیجه می‌توان گفت که روش گرادیان سیاست که به عنوان یکی از روش‌های پیاده‌سازی یادگیری تقویتی عمیق شناخته می‌شود این است که با پیاده کردن مسئله، یک دسته از تجربیات را بدست آورد، بعد ببیند که کدام اقدام‌ها منجر به دریافت بیشترین پاداش ۱+ می‌شود که در نهایت احتمال وقوع آن اقدامات را در آینده برای همان ورودی‌ها افزایش دهد.

[۱] Back propagation

DOI: 10.30503/nripress.2020.321



شکل ٤–١٧: دیاگرام کارتونی ۴ بازی پانگ [۱۰]

DOI: 10.30503/nripress.2020.321

فصل ۵- بررسی مراجع در زمینه استفاده از یادگیری ماشین در پیش بینی بار سیستم‌های قدرت

DOI: 10.30503/nripress.2020.321

۵-۱- مقدمه

انرژی الکتریکی باید به محض تولید مصرف شود زیرا با توجه به هزینه‌های بسیار زیاد سیستم‌های ذخیره انرژی نمی‌توان آن را به سادگی ذخیره کرد. بنابراین، باید با پیش‌بینی دقیق بار، میزان تقاضا و عرضه انرژی متعادل شود. پیش‌بینی بار یک ماژول اساسی سیستم مدیریت انرژی است که هدف آن تهیه یک سیستم مدیریت ایمن و مطمئن برای سیستم برق است و در دهه‌های اخیر بطور جدی توسعه یافته است. عملکرد ایمن و اقتصادی سیستم قدرت به شدت به پیش‌بینی بار متکی است. پیش‌بینی نادرست منجر به ضرر زیاد شرکت‌های برق می‌شود و بر پایداری سیستم نیرو تأثیر می‌گذارد. در مقیاس زمانی، پیش‌بینی‌ها می‌تواند هم کوتاه مدت باشد و هم بلند مدت. به عنوان مثال، پیش‌بینی‌های کوتاه مدت برای متعادل کردن منبع برق و پیش‌بینی‌های بلند مدت برای گسترش ظرفیت، مطالعات بازگشت سرمایه و تجزیه و تحلیل درآمد مورد استفاده قرار می‌گیرند. با افزایش سطح تولید برق توزیع‌شده، چالش‌های جدیدی برای بهره‌برداری از شبکه ایجاد می‌شود. بنابراین، پیش‌بینی بار کوتاه مدت[۱] (STLF) که از دقت و قابلیت اطمینان بالاتری نسبت به پیش‌بینی بار میان مدت و بلند مدت برخوردار است، تعداد زیادی از محققان را به خود جلب کرده است و روش‌های مختلفی برای آن نیز ارائه شده است. روشهای سنتی مبتنی بر آمار ریاضی شامل روش تحلیل رگرسیون [۱۱]، روش میانگین خودهمبسته متحرک[۲] (ARIMA) [۱۲]، روش هموار سازی نمایی[۳] [۱۳]، روش فیلتر کالمن[۴] [۱۴] و غیره هستند. اگرچه روش‌های سنتی از الگوریتم‌های ساده بهره می‌برند، اما مبتنی بر آنالیز خطی هستند، در نتیجه قادر به پیش‌بینی دقیق بارهای غیرخطی نیستند. روش دیگر، روش‌های هوشمند هستند که در مقایسه با روش‌های سنتی بهتر عمل می‌کنند و قادر به پیش‌بینی بارهای غیرخطی نیز هستند. از جمله روش‌های هوشمند می‌توان به سیستم خبره [۱۵]، شبکه‌های عصبی مصنوعی[۵] (ANN) [۱۶] و رگرسیون بردار پشتیبانی[۶] (SVR) [۱۷] اشاره کرد. در سال‌های اخیر، با افزایش هوش مصنوعی، برخی از الگوریتم‌های یادگیری ماشین جدید به سرعت توسعه یافته‌اند و از هوش مصنوعی در زمینه‌های مختلف استفاده شده است.

۵-۲- مطالعات صورت گرفته بر روی پیش‌بینی بار با استفاده از شبکه‌های عصبی

به عنوان بررسی منابع موجود در زمینه پیش‌بینی بار با استفاده از شبکه‌های عصبی، در [۱۸] یک روش پیش‌بینی بار مبتنی بر ترکیب شبکه عصبی با الگوریتم تکامل دیفرانسیل ارایه شده است. در این مطالعه، یک الگوریتم انتخاب ویژگی براساس تجزیه و تحلیل سری زمانی بی نظم[۷] ارائه شده است. مرجع [۱۹] یک رویکرد مبتنی بر شبکه عصبی

[1] Short-term load forecasting
[2] Autoregressive integrated moving average
[3] Exponential smoothing method
[4] Kalman filtering method
[5] Artificial neural network
[6] Support Vector Regression
[۷] Chaotic time series analysis

DOI: 10.30503/nripress.2020.321

جهت تنظیم صحیح پارامترهای شبکه برای حل مشکل پیش‌بینی بار کوتاه مدت ارایه می‌کند. در [۲۰]، شبکه عصبی بازگشتی[۱](RNN) به عنوان یکی از روش‌های هوش مصنوعی برای پردازش داده‌های دنباله‌دار مورد استفاده قرار گرفته است. با این حال، RNN تا حد زیادی از مسئله گرادیان ناپدید شده در برخورد با وابستگی طولانی مدت رنج می‌برد. مسئله گرادیان ناپدید شده مسئله‌ای است که در آموزش شبکه‌های عصبی RNN با روش‌های یادگیری مبتنی بر گرادیان وجود دارد. به همین دلیل، مفهوم حافظه کوتاه مدت طولانی[۲] (LSTM) ارایه شده است. تحقیقات نشان داده است که LSTM در مسئله‌های دنیای واقعی می‌تواند عملکرد بهتری نسبت به سایر روش‌های RNN داشته باشد. در مرجع [۲۱] یک مدل ترکیبی برای پیش‌بینی بار کوتاه مدت میکروگریدها پیشنهاد شده است، مدل پیشنهادی ترکیبی از روش تجزیه حالت تجربی[۳] و فیلتر کالمن است. در [۲۲]، یک استراتژی پیش بینی ترکیبی بر اساس شبکه عصبی آبشاری شده برای STLF پیشنهاد شده است. این استراتژی پیشنهادی ترکیبی است از تبدیل موجک[۴] (WT)، انتخاب ویژگی هوشمند و سه شبکه عصبی که با یگدیگر سری شده‌اند. در [۲۳]، یک الگوریتم دو مرحله‌ای برای پیش‌بینی بار کوتاه مدت پیشنهاد شده است، در مرحله اول روش، از WT و یک ANN برای پیش‌بینی اولیه بار در طی ۲۴ ساعت بعدی استفاده می‌شود. ورودی‌های این مرحله ویژگی‌های آب و هوا (شامل میانگین دما روزانه، حداکثر درجه حرارت، میانگین رطوبت و میانگین سرعت باد) و داده‌های بار روز قبل است. در مرحله دوم، یک WT و سیستم استنتاج فازی عصبی تطبیقی[۵] (ANFIS) برای بهبود نتایج پیش‌بینی بار مرحله اول استفاده می‌شود. در این مقاله، WT برای استخراج داده‌های هواشناسی استفاده شده است. در [۲۴]، یک مدل ترکیبی براساس الگوریتم بازسازی فضای فاز[۶] و مدل رگرسیون هسته توان دوم[۷] برای پیش‌بینی بار کوتاه مدت معرفی شده است. در این مقاله، از الگوریتم بازسازی فضای فاز برای بازسازی داده‌های بار الکتریکی استفاده شده است تا بتواند قابلیت اطمینان عملکردهای پیش‌بینی را بهبود ببخشد. مرجع [۲۵] یک روش برای پیش‌بینی بار کوتاه مدت براساس ترکیبی از روش‌های تبدیل موجک، شبکه عصبی پیشخور و رگرسیون حداقل مربعات جزئی ارایه داده است. در [۲۶]، یک ساختار داده برای پیش‌بینی بار کوتاه مدت با استفاده از خوشه‌بندی بار بر اساس داده‌های کنتور هوشمند و درخت تصمیم‌گیری ارایه شده است. مرجع [۲۷] یک رویکرد مشابه با مرجع قبلی برای پیش‌بینی بار روزانه ارایه کرده است. رویکرد پیشنهادی براساس خوشه‌بندی داده‌های کنتور هوشمند و شبکه عصبی است. به طور کلی، شبکه عصبی برای دستیابی به مشکلات یادگیری غیرخطی با موفقیت زیادی مورد استفاده قرار گرفته است، اما وقتی برای مشکلات پیش‌بینی بار با ویژگی‌های

[۲] Recurrent neural network
[۳] Long short-term memory
[۴] Empirical mode decomposition (EMD)
[۵] Wavelet transform
[۶] Adaptive neuro fuzzy inference system

[۷] Phase space reconstruction algorithm
[۸] Bi-square kernel regression mode

DOI: 10.30503/nripress.2020.321

ورودی متعدد استفاده می‌شود، منجر به ایجاد لایه‌های واحد پنهان از نظر محاسباتی می شود. به عبارت دیگر، افزایش پیچیدگی یک شبکه عصبی باعث ایجاد تاخیرهای بسیار زیاد در آموزش می‌شود. به همین دلیل، بسیاری از مطالعات در این زمینه مجبور شده‌اند تعداد ویژگی‌های ورودی را کاهش دهند تا زمان آموزش قابل کنترل شود. این در حالی است که یک مدل پیش‌بینی هوش مصنوعی باید ویژگی‌های حداکثر ممکن را برای افزایش عملکرد پبش‌بینی داشته باشد.

۵-۳- مطالعات صورت گرفته بر روی پیش‌بینی بار با استفاده از شبکه‌های عمیق

همانطور که بررسی منابع نشان داد، روش‌های مختلفی با استفاده از روش‌های آماری و یادگیری ماشین ارائه شده است تا پیش‌بینی بار را بهبود ببخشد، اما نیاز به مدل‌های دقیق‌تر برای پیش‌بینی بار هنوز هم در اولویت بالایی قرار دارد. در این راستا، رویکردهای یادگیری عمیق اخیراً مورد توجه بسیاری از محققان قرار گرفته و پیشرفت‌های چشمگیری را در بسیاری از حوزه‌ها مانند مدل‌سازی صوتی، پردازش زبان طبیعی و تشخیص تصویر نشان داده‌اند. این سازه‌های لایه‌لایه‌ای عمیق قابلیت انتزاع ویژگی را افزایش داده و مدل‌سازی الگوهای غیرخطی پیچیده را امکان‌پذیر می‌کند. بر این اساس، در [۲۸]، یک آنالیز سری زمانی براساس مدل یادگیری عمیق LSTM برای پیش‌بینی بار کوتاه مدت و میان مدت ارایه شده است. اگرچه، روش پیشنهادی بهتر از روش‌های یادگیری ماشین عمل کرد، اما بسیاری از ویژگی‌های بار پیچیده که توسط سری‌های زمانی نمایش داده می‌شوند را در نظر نمی‌گیرد. این ویژگی‌ها شامل فرکانس داده‌ها، روندها، سطوح، وقفه‌های ساختاری و اثرات تقویم هستند. به ویژه، الگوهای پیچیده روزانه، هفتگی، ماهانه و سالانه بار الکتریکی به عنوان ورودی به مدل پیش بینی LSTM داده نشده است و فقط یک توالی واحد از بارهای گذشته را به عنوان ورودی برای پیش‌بینی بارهای کوتاه مدت و میان مدت در نظر می‌گیرد.

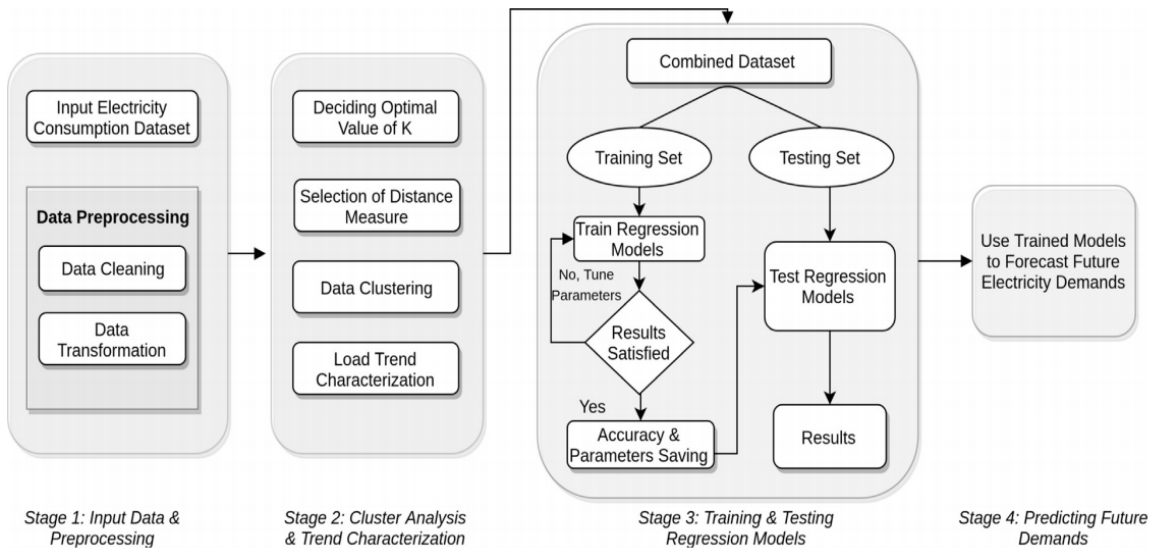
در [۲۹]، از شبکه عصبی عمیق برای پیش‌بینی بار کوتاه مدت استفاده شده است. این مقاله از ترکیبی از شبکه عصبی کانولوشنی برای استخراج ویژگی از داده‌های بار تاریخی و شبکه عصبی بازگشتی برای یادگیری الگوهای مصرف روزانه بار روز بعد استفاده کرده است. نتایج حاصله از روش پیشنهادی با روش‌های تحلیل رگرسیون خطی و SVR مقایسه شده است. مدل‌های یادگیری ماشین، به ویژه یادگیری عمیق، پتانسیل عملکرد بهتری از روش‌های تحلیل سری زمانی سنتی و رگرسیون دارند. با این حال، برای بهبود مدل‌های مصرف الگوهای غیرخطی انرژی، هنوز نیاز اساسی به پیشرفت دارند. این بهبود همراه است با دقت بالا و ثبات پیش‌بینی به ویژه برای پیش‌بینی بار میان مدت.

در [۳۰]، یک رویکرد یادگیری عمیق براساس LSTM و واحدهای بازگشتی دروازه‌ای[1] (GRU) ارایه شده است. نتایج بدست آمده نشان می‌دهد که مدل پیشنهادی برای هر دو پیش‌بینی بار کوتاه مدت و میان مدت با دقت بالا عمل می‌کند. در [۳۱]، از تکنیک‌های مبتنی بر یادگیری عمیق برای پیش‌بینی بار کوتاه مدت ساختمان‌های تجاری

[1] Gated recurrent unit (GRU)

استفاده کرده است. این مقاله، مدل‌های شبکه عصبی gate RNN و gate CNN را به کار برده است. عملکرد این مدل‌ها با روش ARIMA مقایسه شده است. در [۳۲]، از یادگیری عمیق بیزی [۱] برای پیش‌بینی بار خانگی کوتاه مدت استفاده شده است. روش پیشنهادی دو مفهوم تئوری احتمالاتی بیزی و شبکه‌ی LSTM عمیق را ترکیب می‌کند.

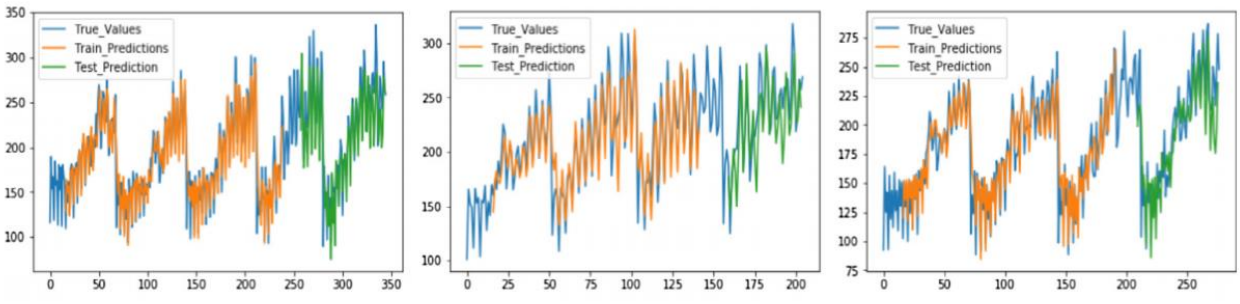
در [۳۳]، یک چارچوب مبتنی بر شبکه‌ی LSTM عمیق برای پیش‌بینی تقاضای برق کوتاه مدت و بلند مدت پیشنهاد شده است. در این مقاله، یک تجزیه و تحلیل خوشه‌ای روی داده‌های مصرف برق در تمام ماه‌ها انجام شده است تا داده‌های فصلی را تولید کند و در نتیجه مشخصه بار را بدست آورد. نتایج بدست آمده با نتایج حاصله از روش‌های RNN و ANN و SVR مقایسه شده و اثربخشی آن ثابت شده است. فرآیند چهارچوب پیشنهادی در شکل ۵-۱ نشان داده شده است. شکل ۵-۲ و ۵-۳ نتایج پیش بینی بار متوسط و پیک با استفاده از مدل پیشنهادی را به تصویر می‌کشند. محور x و محور y در شکل ۵-۲ و ۵-۳ به ترتیب تعداد نمونه‌ها یا نقاط داده و مصرف کل برق را نشان می‌دهد. خط آبی نشانگر داده‌های تجربی واقعی است در حالی که خطوط نارنجی و سبز نشان‌دهنده نتایج پیش‌بینی مدل LSTM پیشنهادی برای آموزش و آزمایش داده‌ها است.



شکل ۵-۱: فلوچارت پیش بینی بار با استفاده از LSTM [۳۳]

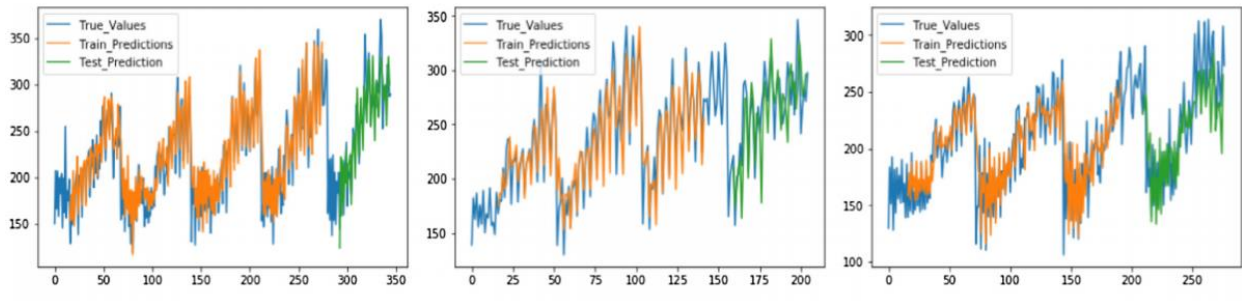
[۲] Bayesian deep learning

DOI: 10.30503/nripress.2020.321



(a) Sumr_Monday_Demand_Prediction_Results (b) Sumr_Wednesday_Demand_Prediction_Result (c) Sumr_Sunday_Demand_Prediction_Results

شکل ۵–۲: نتایج پیش‌بینی بار متوسط برای سه روز (Sumr: Summer) [۳۳]



(a) Sumr_Monday_Demand_Prediction_Results (b) Sumr_Wednesday_Demand_Prediction_Result (c) Sumr_Sunday_Demand_Prediction_Results

شکل ۵–۳: نتایج پیش‌بینی بار پیک برای سه روز (Sumr: Summer) [۳۳]

مرجع [۳۴] نیز به طور مشابه، از شبکه LSTM عمیق برای پیش بینی بار ساعتی استفاده می‌کند و نتایج بدست آمده از این روش را با نتایج حاصله از روش‌های ARMAX و ARMA مقایسه می‌کند.

در [۳۵]، یک روش یادگیری عمیق برای پیش‌بینی پاسخ‌های آینده توربین بادی پیشنهاد شده است. روش پیشنهادی ترکیبی از روش‌های CNN، SVR وLSTM -RNN است. روش پیشنهادی برای مدل‌سازی پیکربندی مکانی-زمانی جریان باد و همچنین مطالعه اثرات آن بر پاسخ‌های توربین بادی، تولید انرژی آن و همچنین لحظه خمش پره توربین ارایه شده است. همچنین در این مقاله، از رویکرد یادگیری چند کاره[۱] برای پیش‌بینی بار کوتاه مدت و بلند مدت استفاده شده است.

مرجع [۳۶]، از رویکرد انتخاب مدل دینامیکی مبتنی بر یادگیری Q[۲] که جزو روش‌های یادگیری تقویتی است، برای پیش‌بینی بار قطعی کوتاه مدت استفاده کرده است. نتایج بدست آمده نشان می‌دهد که رویکرد پیشنهادی به طور قابل ملاحظه‌ای دقت پیش‌بینی بار قطعی را با انتخاب کردن بهترین مدل پیش‌بینی در هر مرحله زمان پیش‌بینی بهبود می‌بخشد. همان نویسندگان در [۳۷] از یک رویکرد دو مرحله‌ای QMS برای پیش‌بینی بارهای قطعی و احتمالاتی کوتاه مدت استفاده کرده‌اند. در مرحله اول، یک عامل یادگیری Q بهترین مدل پیش‌بینی بار قطعی را انتخاب می‌کند و آن را به عنوان ورودی به مرحله دوم ارسال می‌کند. در نهایت، بهترین مدل پیش‌بینی بار احتمالی توسط یک عامل یادگیری Q دیگر انتخاب می‌شود.

[1] Multi-task learning
[2] Q-learning based dynamic model selection (QMS)

DOI: 10.30503/nripress.2020.321

در [۳۸]، یک روش یادگیری عمیق برای پیش‌بینی تقاضای بار کوتاه مدت و میان مدت ساختمان‌های مسکونی پیشنهاد شده است. رویکرد یادگیری عمیق پیشنهادی براساس RNN-GRU است. نتایج مدل پیشنهادی با نتایج بدست آمده از روش‌های SVR، ARIMA، RNN، RNN-LSTM مقایسه شده است. به طور کلی روش GRU نوعی از روش LSTM است. این روش در مقایسه با روش LSTM تعداد پارامترهای کمتر و مدل ساده‌تری دارد. در [۳۹] از روش یادگیری عمیق CNN برای پیش‌بینی بار کوتاه مدت خانگی ارایه شده است. نتایج بدست آمده با نتایج حاصله از روش‌های SVR و شبکه عصبی ساده مقایسه شده است.

شبکه عصبی توابع پایه شعاعی[۱] یک شبکه عصبی مصنوعی است که از توابع پایه شعاعی به عنوان توابع فعال‌سازی استفاده می‌کند. خروجی شبکه، ترکیبی از توابع پایه شعاعی ورودی‌ها و پارامترهای نورون است. شبکه‌های توابع پایه شعاعی کاربردهای زیادی دارند، از جمله تقریب عملکرد، پیش‌بینی سری زمانی، طبقه‌بندی و کنترل سیستم. به همین منظور، در [۴۰]، یک رویکرد جدید برای پیش‌بینی بار کوتاه مدت با استفاده از یک شبکه عصبی عمیق توابع پایه شعاعی چند ستونی ارائه شده است. مزیت این روش جدید نسبت به مدل‌های مشابه در سرعت و دقت مورد بحث قرار گرفته است. قابل ذکر است که در این مقاله از الگوریتم درخت k-d برای تقسیم کردن مجموعه داده‌های ورودی به زیرمجموعه‌های کوچکتر استفاده شده است.

ماشین‌های بردار پشتیبان حداقل مربعات[۲] (LS-SVM) نسخه‌ای از روش ماشین بردار پشتیبان هستند که از روش‌های یادگیری نظارتی محسوب می‌شوند و برای طبقه‌بندی و تحلیل رگرسیون استفاده می‌شوند. در این زمینه، در [۴۱]، یک روش ترکیبی براساس رویکرد یادگیری چند کاره و LS-SVM برای پیش‌بینی بار کوتاه مدت ۲۴ ساعته استفاده شده است. به این صورت که رویکرد یادگیری چند کاره به عنوان مدل پیش‌بینی در نظر گرفته شده است و الگوریتم LS-SVM برای انجام پیش‌بینی به کار برده شده است.

نصب کنتورهای هوشمند امکان جمع‌آوری داده‌های مصرف برق را فراهم می‌کند و پیش‌بینی میزان بار مصرف کننده فردی را ممکن می‌کند. در مقایسه با بارهای تجمیع شده، پیش‌بینی بار برای مصرف کنندگان منفرد از ویژگی‌های غیر ثابت و تصادفی برخوردار است. در این زمینه، در [۴۲]، از روش LSTM برای پیش‌بینی بار احتمالاتی کوتاه مدت برای مصرف کنندگان تکی استفاده شده است. در رویکرد پیشنهادی، تابع از دست دادن پین بال[۳] به جای میانگین مربعات خطا[۴] (MSE) برای هدایت آموزش شبکه LSTM استفاده شده است. در [۴۳] روش LSTM-GRU برای پیش‌بینی بار کوتاه مدت به کار برده شده است. اعتبارسنجی مدل پیشنهادی با انجام مقایسه با روش‌های LSTM، SVR و ANN حاصل شده است.

مدل LSTM دارای یک ظرفیت یادگیری قوی برای گرفتن وابستگی زمانی سری زمانی است و عملکرد پیشرفته‌ای را ارائه می‌دهد. با این حال، با افزایش همبستگی بخش‌های بار در مدت زمان طولانی، آموزش مدل LSTM سخت‌تر

---

۳ Radial basis function

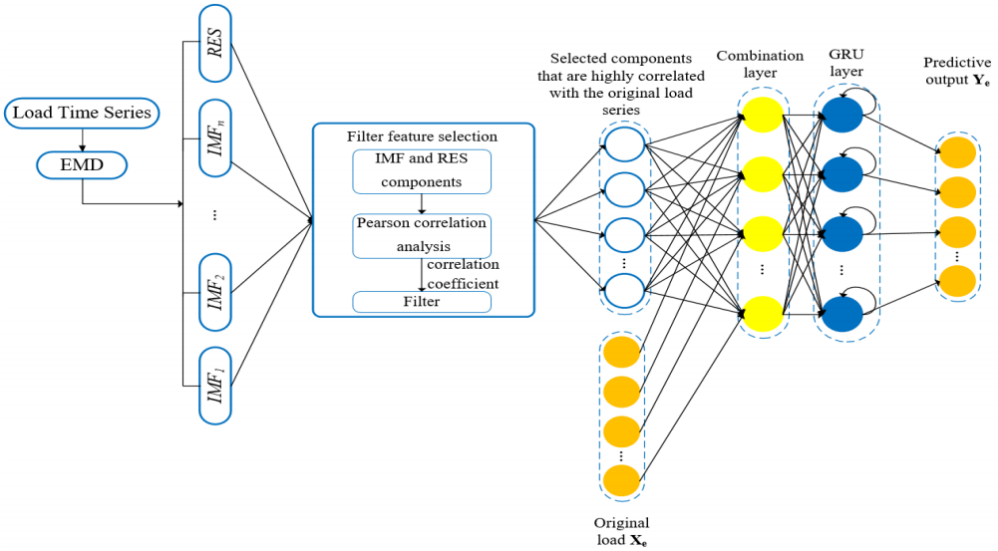
[۱] Least square support vector machine

[۲] Pinball loss

[۳] Mean square error

می‌شود زیرا نمی‌تواند به طور کامل از مسئله گرادیان ناپدید شونده جلوگیری کند. به همین منظور، در [۴۴]، یک الگوی یادگیری گروهی عمیق[۱] ارائه شده است، که شامل یک انتخاب کننده و یک پیش‌بینی کننده است. انتخاب کننده بطور فعال چندین بخش بار اصلی با شبیه‌ترین الگوی مشابه را به عنوان نمونه فعلی به منظور آموزش پیش‌بینی کننده، انتخاب می‌کند و پیش‌بینی کننده یک ماشین یادگیری عمیق مبتنی بر یادگیری گروهی است که از ادغام LSTM با شبکه پرسپترون[۲] چندلایه تشکیل شده است. شبکه پرسپترون چندلایه دسته‌ای از شبکه‌های عصبی مصنوعی پیشخور[۳] است.

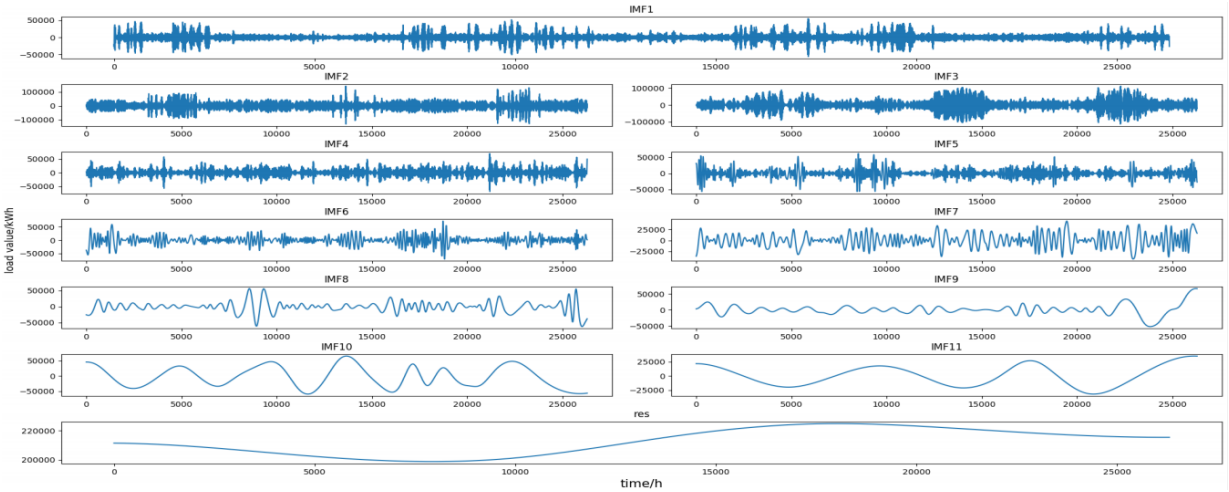
در مرجع [۴۵]، یک رویکرد یادگیری عمیق براساس EMD-GRU برای پیش‌بینی بار ساعتی ارایه شده است. این مقاله در ابتدا، سری اطلاعات بار اصلی را توسط EMD به چند زیر سری بار تبدیل می‌کند. سپس، با استفاده از روش ضریب همبستگی پیرسون[۴]، همبستگی بین زیر سری‌های بار و سری اصلی بار را تحلیل می‌کند. برخی از زیر سری‌های با همبستگی زیاد همراه با سری بار اصلی به عنوان ویژگی‌ها به عنوان ورودی به شبکه GRU وارد می‌شوند و در نهایت مدل پیش‌بینی ایجاد می‌شود. فرآیند مدل پیشنهادی در شکل ۵-۴ نشان داده شده است. نتایج یک مجموعه بار ورودی که به چند زیر سری بار تجزیه شده است، در شکل ۵-۵ به تصویر کشیده شده است. محور افقی نقطه زمان را نشان می‌دهد و محور عمودی مقدار تجزیه را نشان می‌دهد. شکل ۵-۶ مقایسه‌ی بین مقادیر پیش‌بینی شده و واقعی به دست آمده توسط هر مدل برای مجموعه داده ورودی را نشان می‌دهد. از نتایج پیداست که روش پیشنهادی بهتر می‌تواند روند تغییر بار را در مقایسه با روش‌های دیگر ثبت کند.



شکل ۵-٤: مدل شبکه عصبی عمیق EMD-GRU [۴۵]

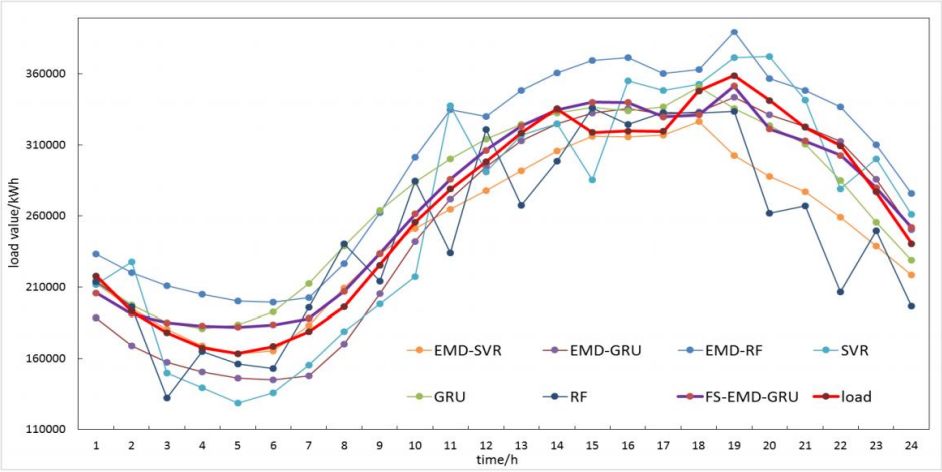
[۴] Deep ensemble learning model
[۵] Multilayer perceptron
[۶] Feedforward artificial neural network
[۷] Pearson correlation coefficient

DOI: 10.30503/nripress.2020.321



شکل ۵–۵: نتایج تجزیه بار اصلی به چند زیرسری[۴۵]

در مرجع [۴۶]، از روش یادگیری عمیق مبتنی بر CNN برای پیش‌بینی بار کوتاه مدت استفاده شده است. به این صورت که در ابتدا، از روش‌های درخت تصمیم‌گیری، حذف ویژگی بازگشتی[۱] و جنگل‌های تصمیم تصادفی[۲] برای انتخاب و استخراج ویژگی‌ها از اطلاعات بار ورودی استفاده می‌کند. سپس ویژگی‌های بدست آمده را به عنوان ورودی به شبکه CNN ارسال می‌کند تا عمل پیش‌بینی را انجام دهد.



شکل ۵–۶: مقایسه نتایج پیش‌بینی بار بدست آمده [۴۵]

در مرجع [۴۷]، یک رویکرد یادگیری عمیق براساس DTW-GRU[۳] برای پیش‌بینی بار روزانه ارائه شده است. در ابتدا، از ضریب همبستگی برای توصیف میزان همبستگی داده‌های ورودی استفاده شده است. در مرحله‌ی بعد، رویکرد DTW استفاده شده است که شبیه‌ترین منحنی بار را مطابقت دهد و شبیه‌ترین بار پیک روزانه را بدست آورد. علاوه

[۱] Recursive feature elimination
[۲] Random decision forests
[۳] Dynamic time warping

DOI: 10.30503/nripress.2020.321

بر این، از طرح رمزگذاری[1] بر روی اطلاعات تقویم برای گسترش ویژگی‌های پیش‌بینی استفاده کرده است که می‌تواند تأثیر آنها را بیشتر در پیش‌بینی بار مشخص کند.

مرجع [۴۸]، از یک شبکه ترکیبی یادگیری عمیق براساس LSTM-DRNN برای پیش‌بینی بار روزانه و توان خروجی فتوولتاییک در یک ریزشبکه استفاده کرده است. میکروگرید پیشنهادی متشکل از خودروهای برقی، آرایه‌های فتوولتاییک و سیستم‌های ذخیره‌ساز انرژی است. همچنین، عدم قطعیت‌های متناظر با توان خروجی فتوولتاییک و سطح بار نیز در نظر گرفته شده است. این مقاله، اثربخشی مدل پیشنهادی با انجام مقایسه با روش‌های MLP و SVM نشان داده است.

در مرجع [۴۹]، یک رویکرد یادگیری عمیق براساس EMD-LSTM برای پیش‌بینی بار متوسط و پیک ساعتی، روزانه و فصلی پیشنهاد شده است. در ابتدا بر روی داده‌های ورودی عمل پیش پردازش داده‌ها اعمال شده است سپس خروجی با استفاده از روش خوشه‌بندی داده‌ها، نرمال‌سازی شده‌اند. در ادامه، روش EMD به کار برده شده است تا سری اصلی بار را به چند زیر سری بار تبدیل کند. شبکه LSTM به طور جداگانه با استفاده از هر زیر سری آموزش داده می‌شود. عملکرد طرح پیشنهادی با مقایسه کردن با نتایج حاصل از روش‌های RNN، LSTM و EMD-LSTM ارزیابی شده است.

مرجع [۵۰] تلاش می‌کند مسئله پیش‌بینی بار ساعتی را برای خانه‌های مسکونی منفرد مورد بررسی قرار دهد. در ابتدا برای ارزیابی و مقایسه ناسازگاری بین بار جمع شده و بارهای فردی از یک روش خوشه‌بندی مبتنی بر تراکم استفاده می‌کند. سپس از رویکرد یادگیری LSTM برای پیش‌بینی بار خانواده‌های مسکونی منفرد استفاده می‌کند. اگر چه پیش‌بینی بار خانوارهای منفرد زیاد دقیق نیست، اما نتایج نشان می‌دهد که با جمع‌آوری تمام پیش‌بینی‌های منفرد، پیش‌بینی بهتری برای سطح تجمع در مقایسه با حالتی که مستقیما سطح بار تجمع بدست آید، حاصل می‌شود.

پیش‌بینی بار احتمالی مسکونی به دلیل تنوع و نوسانات زیاد که ترکیب شده با حجم زیادی از داده‌ها، یک مسئله چالش برانگیز است. برای پرداختن به این چالش، در مرجع [۵۱]، یک رویکرد یادگیری عمیق براساس MTL-Bayesian برای پیش‌بینی بار ساعتی بارهای خانگی پیشنهاد شده است. در ابتدا، روش خوشه‌بندی داده‌ها به منظور افزایش تنوع و حجم داده‌های آموزش استفاده شده است. هدف استفاده از یادگیری چند کاره در این مقاله این است که عدم قطعیت‌های مشترک میان گروه‌های مصرف کننده مجزا را تعیین کند.

در مرجع [۵۲]، از شبکه عصبی پسخور-پیشخور برای پیش‌بینی بار ساعتی ساختمان‌های خانگی استفاده شده است. در ابتدا، روش خوشه‌بندی داده‌ها برای دسته‌بندی داده‌های ورودی استفاده شده است. از الگوریتم Nesterov برای آموزش شبکه استفاده شده است. الگوریتم Nesterov به یک رویکرد کلی اشاره دارد که می‌تواند برای اصلاح کردن روش‌های مبتنی بر گرادیان کاهشی مورد استفاده قرار گیرد و باعث بهبود همگرایی اولیه مسئله شود.

همانطور که در مرجع [۵۱] عنوان شد، چالش اصلی پیش‌بینی بار خانگی در نوسانات زیاد و عدم اطمینان پروفیل‌های بار نهفته است. در مرجع [۵۳]، یک رویکرد یادگیری عمیق براساس RNN ارایه شده است. به منظور

[1] Encoding

DOI: 10.30503/nripress.2020.321

مقابله با مسئله ی overfit و عدم قطعیت پروفیل بارها، در ابتدا پروفیل بار خانوارها در یک استخر پروفیل بار جمع می‌شوند. نتایج بدست آمده نشان می‌دهد که روش پیشنهادی می‌تواند پیشرفت قابل توجهی را برای پیش‌بینی بار خانگی در حضور عدم قطعیت‌ها داشته باشد.

در مرجع [۵۴]، از روش یادگیری عمیق GRU برای پیش‌بینی بار روزانه استفاده شده است. در ابتدا، داده‌های ورودی با استفاده از روش خوشه‌بندی داده‌ها براساس ویژگی‌های ورودی به چندین دسته تقسیم‌بندی شده‌اند. شبکه GRU با استفاده از داده‌های سطح بار ورودی و همچنین ویژگی‌ها آموزش داده شده است. در این مقاله، داده‌های قیمت برق نیز به عنوان یکی از ویژگی‌ها به عنوان ورودی به شبکه در نظر گرفته شده است.

مرجع [۵۵] از یک روش یادگیری مبتنی بر SVR برای پیش‌بینی بار ساعتی استفاده می‌کند. این مقاله از روش خود رمزنگار[۱] برای استخراج ویژگی‌ها استفاده می‌کند. مهمترین نوآوری مدل پیشنهادی این است که ویژگی‌های استخراج شده از داده‌های بار الکتریکی اصلی توسط اتو انکودر خود رمزنگارها منجر به پیش‌بینی بار با دقت بیشتر و خطای کمتر می‌شود.

در مرجع [۵۶]، از روش یادگیری عمیق CNN مبتنی بر شبکه عصبی توابع پایه شعاعی برای پیش‌بینی بار روزانه استفاده شده است، به این صورت که که از توابع پایه شعاعی به عنوان توابع فعال‌سازی نورون‌ها استفاده می‌کند. مدل پیشنهادی در ابتدا داده‌های ورودی را با استفاده از یک روش خوشه‌بندی مبتنی بر منطق فازی در گروه‌های مختلف خوشه‌بندی می‌کند. برای هر دسته ایجاد شده، یک مدل شبکه عصبی در نظر گرفته شده است که با استفاده از داده‌های آن دسته آموزش داده می‌شوند. در نهایت پیش‌بینی بار بدست آمده از هر شبکه عصبی با یکدیگر جمع می‌شوند.

در مرجع [۵۷]، از روش‌های یادگیری تقویتی شامل روش گرادیان سیاست قطعیت بازگشتی و شبکه سیاست قطعیت عمیق برای پیش‌بینی بار ساعتی واحدهای مسکونی استفاده شده است. در ابتدا، یک روش تشخیص ناهنجاری به‌عنوان عمل پیش پردازش داده‌ها استفاده شده است که ناهنجاری در داده‌های ورودی را از بین ببرد. همچنین، بر روی داده‌های ورودی از ضریب همبستگی برای استخراج ویژگی‌ها استفاده شده است. نتایج نشان می‌دهد که مدل‌های پیشنهادی در پیش‌بینی مصرف انرژی واحدهای مسکونی دارای مزایای آشکاری هستند.

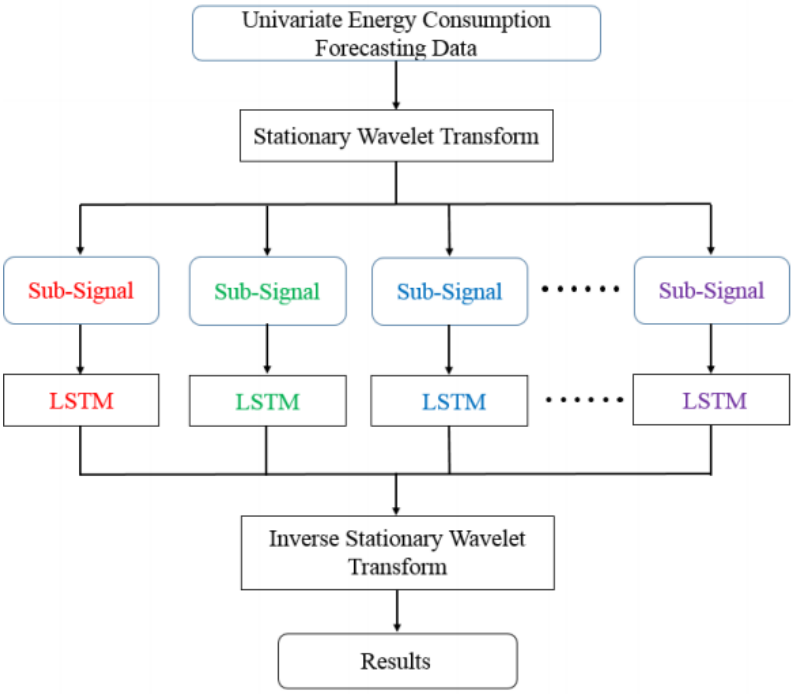
مرجع [۵۸]، از شبکه عصبی پسخور پیشخور عمیق برای پیش‌بینی میزان و چگالی سطح بار استفاده می‌کند. در این مقاله، عمل پیش پردازش داده‌ها شامل عمل پاکسازی داده‌ها نیز انجام شده است. پاکسازی داده‌ها فرایند تشخیص و تصحیح داده‌های نادرست از یک مجموعه داده است. در مرجع [۵۹]، از شبکه‌های عصبی و کانولوشنی عمیق برای پیش‌بینی بار ساعتی استفاده شده است. در این مقاله، روش greedy forward به عنوان یکی از روش‌های انتخاب ویژگی مبتنی بر wrapper برای انتخاب ویژگی‌ها استفاده شده است.

در مرجع [۶۰]، یک مدل یادگیری عمیق ترکیبی پیشنهاد شده است. مدل پیشنهادی ترکیبی از شبکه LSTM با روش تبدیل موجک ثابت است. روش تبدیل موجک ابعاد داده‌های ورودی را افزایش می‌دهد، که به طور بالقوه به

[۱] Auto Encoder

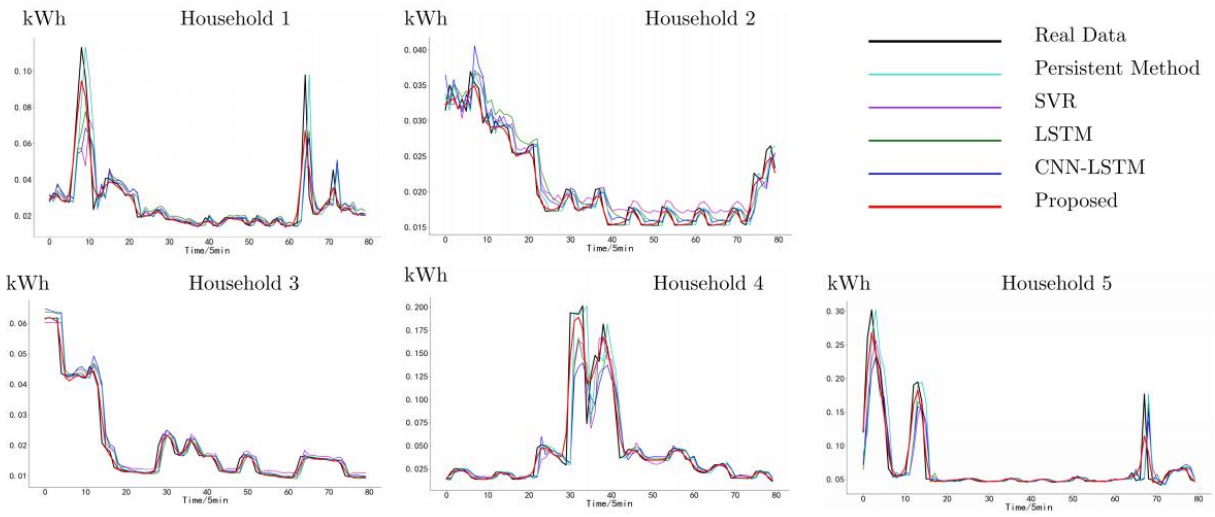
DOI: 10.30503/nripress.2020.321

بهبود دقت پیش بینی LSTM کمک می‌کند. عملکرد طرح پیشنهادی با مقایسه کردن با نتایج حاصله از روش‌های LSTM، SVR و CNN-LSTM ارزیابی شده است. فرآیند مدل پیشنهادی در شکل ۵-۷ نشان داده شده است.



شکل ۵–۷: شماتیک مدل پیشنهادی [۶۰]

نتایج پیش‌بینی بار براساس مدل‌های مختلف برای داده‌های مصرف برق پنج ساختمان خانگی در شکل ۵-۸ نشان داده شده است. همانطور که از نتایج پیداست، خطای نتایج بدست آمده با روش پیشنهادی نسبت به داده بار اصلی دارای حداقل خطا است.



شکل ۵–۸: مقایسه نتایج پیش‌بینی بار بدست آمده [۶۰]

DOI: 10.30503/nripress.2020.321

مرجع [۶۱]، یک الگوریتم شبکه عصبی عمیق برای تخمین تابع چگالی احتمال قیمت، پیش‌بینی سرعت باد و بار خانگی ارائه می‌دهد. مدل پیشنهادی ترکیبی از شبکه عصبی CNN و GRU است. در این مقاله، روش برآوردگر چگالی هسته تطبیقی به عنوان یک روش عددی استفاده شده است تا مشخصات احتمالی قیمت برق را تخمین بزند.

در مرجع [۶۲]، یک روش پیش‌بینی بار شبکه توان آفریقای جنوبی براساس یک روش هوش مصنوعی ترکیبی و یک روش یادگیری عمیق پیشنهاد شده است. روش هوش مصنوعی ترکیبی پیشنهاد شده براساس روش OP-ELM[1] است و روش یادگیری عمیق مورد استفاده شده براساس روش LSTM است. در این مقاله، در ابتدا بر روی داده‌های ورودی عمل پیش پردازش داده‌ها براساس روش ANFIS انجام شده است.

در مرجع [۶۳]، یک رویکرد یادگیری عمیق ترکیبی براساس LSTM و CNN برای پیش‌بینی بار روزانه سیستم قدرت پیشنهاد شده است. مدل پیشنهادی از نقطه نظر داده‌های مصرف انرژی سه شهرک صنعتی بزرگ در کره جنوبی تأیید شده است. اثربخشی مدل پیشنهادی با انجام مقایسه با روش‌های RNN، CNN و MLP اثبات شده است.

همانطور که قبلا بیان شد، RNNها قادر به استخراج رفتار زمانی دینامیک برای سری‌های زمانی هستند، اما در مورد مراحل طولانی مدت آموزش آن‌ها سخت می‌شود. به منظور مقابله با همین مسئله، مرجع [۶۴]، از رویکردهای یادگیری عمیق CNN و LSTM برای پیش‌بینی بارهای کوتاه مدت و میان مدت استفاده شده است. هنگام استفاده از CNN، با تبدیل همبستگی زمانی سری داده‌های بار به همبستگی مکانی، بارهای الکتریکی را به پیکسل‌های تصویر تبدیل کرده است و سری بارها را به صورت اکتشافی در قالب یک تصویر مرتب کرده است. همچنین، هنگام استفاده از LSTM، آن دنباله کوتاه‌تر جدیدی از سری‌های بار اصلی را بازسازی می‌کند که بر محدودیت‌های LSTM در مراحل طولانی مدت غلبه می‌کند، در حالی که بیشترین اطلاعات را از نسخه اصلی بار دارد.

شبکه باور عمیق[2] یک کلاس دیگر از شبکه‌های عصبی عمیق است. شبکه باور عمیق متشکل از چند لایه با اتصالات بین لایه‌ها است نه اتصالات بین واحدهای درون هر لایه.

مدیریت سمت تقاضا[3] (DSM) پیچیدگی محیط پیش‌بینی بار را افزایش می‌دهد، در این حالت، برآورده کردن یک پیش‌بینی بار دقیق با استفاده از روش‌های سنتی دشوار است. از آنجا که یادگیری عمیق می‌تواند عوامل مختلفی را برای بهبود نتایج پیش‌بینی بار در نظر بگیرد، مرجع [۶۵] شبکه باور عمیق[4] را از سه جنبه داده ورودی، مدل و عملکرد بهبود می‌بخشد و از آن برای حل مشکل پیش‌بینی بار کوتاه مدت در DSM استفاده می‌کند. شبکه باور عمیق پیشنهادی دراین مقاله از چند لایه ماشین بولتزمن محدود گاوسی[5] تشکیل شده است. اثربخشی مدل پیشنهادی با انجام مقایسه با روش‌های LSSVM و ARIMA اثبات شده است.

مرجع [۶۶]، یک رویکرد مبتنی بر شبکه باور عمیق را برای پیش‌بینی بار کوتاه مدت سیستم‌های تهویه هوا به کار برده است. در ابتدا، مشابه مرجع [۶۵]، سری اطلاعات بار اصلی خنک کننده توسط EMD به چند زیر سری

[1] Optimally pruned extreme learning machine (OP-ELM)
[2] Deep belief network
[3] Demand-side management
[4] Deep belief network
[5] Gauss-Bernoulli restricted Boltzmann machine

DOI: 10.30503/nripress.2020.321

اطلاعات بار با رفتارهای بهتر تبدیل شده است. سپس این اطلاعات به عنوان ورودی به شبکه باور عمیق مبتنی بر ماشین بولتزمن محدود چند لایه داده می‌شود. در نهایت اثربخشی مدل پیشنهادی با انجام مقایسه با روش‌های SVM و ANN اثبات شده است.

در مرجع [۶۷]، یک شبکه باور عمیق مبتنی بر ماشین بولتزمن برای پیش‌بینی بار ساعتی سیستم قدرت پیشنهاد شده است. داده‌های بار یک ساله که از مناطق شهری در تگزاس و آرکانزاس در ایالات متحده جمع‌آوری شده است، در مطالعات استفاده شده است. در این مقاله روش تبدیل توان Box-Cox به کار برده شده است که داده‌های سری زمانی ورودی را نرمالیزه کند و آن را به عنوان ورودی به شبکه باور عمق ارسال کند.

پیش‌بینی بار دقیق شبکه توزیع به منظور کنترل و عملکرد بهینه شبکه‌های توزیع یک امر مهم است. در این راستا، در مرجع [۶۸]، یک شبکه باور عمیق مبتنی بر ماشین بولتزمن برای پیش‌بینی بار احتمالی و قطعی شبکه توزیع پیشنهاد شده است. همچنین در این مقاله، تکنیک‌های bagging و boosting معرفی شده‌اند تا تطبیق‌پذیری شبکه باور عمیق را ارتقا دهند. در روش bagging یک زیر مجموعه از مجموعه داده‌ی اصلی به هر کدام از دسته‌بندها داده می‌شود. یعنی هر دسته‌بند یک قسمت از مجموعه‌ی داده را مشاهده کرده و باید مدل خود را بر اساس همان قسمت از داده‌ها که در اختیارش قرار گرفته است، بسازد. در روش boosting، آن نمونه‌هایی که دسته‌بند اول نتوانسته است به درستی دسته‌بندی کند، با احتمال بیشتری برای دسته‌بند دوم انتخاب می‌شود.

همچنین، در مرجع [۶۹]، یک روش دسته‌بندی گروهی متشکل از روش EMD و رویکرد یادگیری عمیق پیشنهاد شده است. در ابتدا، مشابه با مراجع [۶۵] و [۶۶]، سری اطلاعات بار اصلی توسط EMD به چند زیر سری اطلاعات بار تقسیم شده‌اند. سپس، هر یک از این زیر سری بارها به عنوان داده‌های ورودی به یک شبکه باور عمیق داده می‌شوند. در نهایت، نتایج پیش‌بینی هر یک از این زیر سری اطلاعات بار، با یکدیگر جمع شده تا خروجی کل برای پیش‌بینی بار بدست آید.

یادگیری انتقال[۱] یک مسئله تحقیق در یادگیری ماشین است که بر ذخیره دانش به دست آمده هنگام حل یک مسئله و استفاده از آن در یک مسئله متفاوت اما مرتبط متمرکز است. به عنوان مثال، دانش به دست آمده هنگام یادگیری تشخیص اتومبیل می‌تواند در هنگام تلاش برای تشخیص کامیون‌ها کاربرد داشته باشد. در زمینه پیش‌بینی بار با استفاده از یادگیری انتقال، در مرجع [۷۰]، یک رویکرد یادگیری انتقال زنجیره‌ای مبتنی بر شباهت برای ایجاد مدل‌های پیش‌بینی بار مبتنی بر شبکه عصبی اندازه‌گیرهای هوشمند ارایه شده است. در ابتدا، داده‌های بدست آمده از اولین اندازه‌گیر به عنوان ورودی به یک شبکه RNN داده می‌شود و آن شبکه با استفاده از این داده‌ها آموزش داده می‌شود. سپس، مدل با استفاده از داده‌های آن اندازه‌گیری که پروفیل مصرف انرژی آن بیشترین شباهت را با پروفیل مصرف اندازه‌گیر اول دارد ساخته می‌شود. اما باید توجه داشت که روند آموزش با مدل از قبل آموزش دیده از اندازه‌گیر داده اول شروع می‌شود. این روند با توجه به شباهت‌های موجود در الگوهای مصرف انرژی به صورت زنجیره‌ای ادامه می‌یابد.

---

[۱] Transfer learning

DOI: 10.30503/nripress.2020.321

۵-٤- دسته‌بندی مقالات مرور شده

جدول ۵-۱ مراجع بررسی شده در زمینه‌ی پیش‌بینی بار با استفاده از یادگیری عمیق را از نقطه نظر نوع مدل، انواع ویژگی‌ها، روش‌های استخراج و انتخاب ویژگی‌ها، انواع عملیات پیش پردازش روی داده‌ها، مشخصات ساختاری شبکه (شامل تعداد لایه‌های نهان، تعداد نورون و ...)، منبع و دوره داده و نوع دستاورد خلاصه می‌کند. در جدول ۵-۲، مقالات از نقطه نظر نوع مدل و نوع نوآوری پیشنهاد شده دسته‌بندی شده‌اند. در جدول ۵-۳، مقالات از نقطه نظر ویژگی‌ها دسته‌بندی شده‌اند. همچنین، در جدول ۵-۴، مقالات از نقطه نظر وجود عملیات پیش پردازش روی داده‌ها و روش‌های استخراج و انتخاب ویژگی‌ها دسته‌بندی شده‌اند.

DOI: 10.30503/nripress.2020.321

جدول ۵-۱: خلاصه‌ای از تحقیقات صورت گرفته شده در زمینه‌ی پیش‌بینی بار با استفاده از یادگیری عمیق

دستاورد	تعداد لایه نهان - تعداد نورون- بچ سایز- نرخ یادگیری	پیش پردازش	استخراج ویژگی	انتخاب ویژگی	منبع داده (دوره)	ویژگی‌ها	مدل	مرجع
پیش‌بینی بار هفتگی و ماهانه	۶ – -۱۲۵-۱۰۰ ۰/۰۰۵	Missing data	-	مدل Wrapper (درخت تصمیم گیری)	داده‌های مصرف متروپل فرانسه (۲۰۰۸-۲۰۱۶)	داده‌های بار، سرعت باد، رطوبت، دما، روز هفته، ماه سال	LSTM	[28]
پیش‌بینی بار روزانه	-	-	CNN	-	داده‌های یک شهر در چین شمالی	داده‌های بار، دما، روز هفته، ساعت روز	RNN	[29]
پیش‌بینی بار روزانه، هفتگی و ماهانه	۳ – -۱۲۵-۱۰۰ ۰/۰۰۵	Missing data	-	مدل Wrapper (درخت تصمیم گیری)	داده‌های مصرف متروپل فرانسه (۲۰۰۸-۲۰۱۶)	داده‌های بار، سرعت باد، رطوبت، دما، روز هفته، ماه سال	LSTM, GRU	[30]
پیش‌بینی بار روزانه	۲ - -۵۰-۳ ۰/۰۰۵	Missing data (روش list-wise deletion)	-	-	داده‌های سه ساختمان تجاری در شهرهای اسکندریه، شرلی، اوکسبریج (۱ سال)	داده‌های بار، سرعت باد، رطوبت، دما، فشار هوا، ساعت روز	gated RNN, gated CNN	[31]
پیش‌بینی بار روزانه	۲ - -۷۲۰-۲۰ ۰/۰۰۱	کلاستربندی داده‌های ورودی	-	-	داده‌های بار و فتوولتاییک سقفی در شبکه برق استرالیا (۳ سال)	داده‌های بار، ساعت روز، روز هفته، ماه سال، داده‌های تولید فتوولتاییک سقفی	Bayesian theory-LSTM	[32]
پیش‌بینی بار یک روز از هر فصل سال	-	Missing data, Data Aggregation, Data Transformation	-	-	داده‌های مصرف چندی‌گر مرکز دو ایالت پنجاب و هاریانا در کشور هند (روزهای انتخابی در هر فصل برای ۱ سال)	داده‌های بار	LSTM, RNN	[33]
پیش‌بینی بار ساعتی	-	-	-	-	داده‌های مصرف ۱۳ ساختمان در دانشگاه تگزاس در دالاس (۲ سال)	داده‌های بار، سرعت باد، رطوبت، دما، تابش افقی جهانی، ساعت روز، روز هفته ،ماه سال	یادگیری تقویتی (روش یادگیری Q)	[36]
پیش‌بینی بار ساعتی	-	-	-	-	داده‌های مصرف ۱۳ ساختمان در دانشگاه تگزاس در دالاس (۲ سال)	داده‌های بار، سرعت باد، رطوبت، دما، فشار هوا، تابش افقی جهانی، تابش افقی پراکنده، ساعت روز، روز هفته، ماه سال	یادگیری تقویتی (رویکرد دو مرحله‌ای با روش یادگیری Q)	[37]

DOI: 10.30503/nripress.2020.321

جدول ۵-۱: خلاصه‌ای از تحقیقات صورت گرفته شده در زمینه‌ی پیش‌بینی بار با استفاده از یادگیری عمیق (ادامه...)

دستاورد	تعداد لایه نهان - تعداد نورون- بچ سایز- نرخ یادگیری	پیش پردازش	استخراج ویژگی	انتخاب ویژگی	منبع داده (دوره)	ویژگی‌ها	مدل	مرجع
پیش‌بینی بار ساعتی	۴ (تعداد لایه نهان)	-	-	-	داده های مصرف ساختمان‌های مسکونی به دست آمده از وب سایت Dataport (۱ سال و ۲ ماه)	داده‌های بار، سرعت باد، رطوبت، دما، فشار هوا، ساعت روز، روز هفته، روز ماه، ماه سال	RNN, GRU	[38]
پیش‌بینی بار ساعتی	۵۰ (تعداد نورون)	-	الگوریتم درخت k-d	-	داده‌های مصرف ساعتی New England ISO (۱۳ سال)	داده‌های بار، سرعت باد، دما، فشار هوا، ساعت روز، روز سال	BFNN	[40]
پیش‌بینی بار ساعتی	-	نرمال سازی داده‌های ورودی	-	-	داده‌های مصرف سیستم انرژی یکپارچه شامل زیر سیستم‌های انرژی برق، گرما، سرمایش و گاز در پارک صنعتی سوژو (۱ سال)	سرعت باد، دما، فشار هوا، رطوبت، روز کاری، تعطیلات، قیمت برق، داده‌های بار الکتریکی، حرارتی، گازی و سرمایشی	MTL-LSSVM	[41]
پیش‌بینی بار نیم ساعت بعد	-	-	-	-	داده‌های بار مصرف کنندگان مسکونی و تجاری تهیه شده توسط کمیسیون تنظیم انرژی در ایرلند (۱ و نیم سال)	داده‌های بار، ساعت روز، روز هفته	LSTM	[42]
پیش‌بینی بار روزانه	-	نرمال سازی داده‌های ورودی، abnormal data، Missing data	-	-	داده‌های بار دریافت شده از شرکت انرژی اسلواکی (۱ سال)	داده‌های بار، دما، رطوبت، تعطیلات، ساعت روز، روز هفته	LSTM-GRU	[43]
پیش‌بینی بار روزانه	۲ (تعداد لایه)، ۸ (تعداد نورون)	نرمال سازی داده‌های ورودی	-	ضریب همبستگی پیرسون	بار الکتریکی روزانه شهر Hangzhou در شرق چین (۳ سال و ۲ ماه)	داده‌های بار، دما، روز هفته	LSTM-MLP	[44]

DOI: 10.30503/nripress.2020.321

جدول ۵-۱: خلاصه‌ای از تحقیقات صورت گرفته شده در زمینه‌ی پیش‌بینی بار با استفاده از یادگیری عمیق (ادامه...)

دستاورد	تعداد لایه نهان - تعداد نورون- بچ سایز- نرخ یادگیری	پیش پردازش	استخراج ویژگی	انتخاب ویژگی	منبع داده (دوره)	ویژگی‌ها	مدل	مرجع
پیش‌بینی بار ساعتی	رنج برای تعداد نورون [40,100]، رنج برای نرخ یادگیری [0.001,0.01]	-	-	ضریب همبستگی پیرسون	داده‌های بار روزانه از یک منطقه در شمال غرب چین و شرکت خدمات عمومی آمریکا (۳ سال)	داده‌های بار	EMD[1]-GRU	[45]
پیش‌بینی بار روزانه	۵۰ (بچ سایز)، ۸۰۰ (تعداد نورون)، ۰.۰۱(نرخ یادگیری)	۱- کاربرد ضریب همبستگی برای توصیف میزان همبستگی داده‌ها ۲-تطبیق منحنی بار با DTW ۳- انکودر کردن ویژگی‌ها به جز داده‌های بار و دما	-	-	مجموعه داده‌های شبکه اروپا در فناوری‌های هوشمند (۲ سال)	داده‌های بار، دما، روز کاری، تعطیلات، روز هفته	GRU−DTW	[47]
پیش‌بینی بار و توان خروجی فتوولتاییک روزانه	۲ (تعداد لایه برای RNN)،	نرمال سازی داده‌های ورودی	-	-	داده‌های مصرف و فتوولتاییک سقفی ساختمان‌های مسکونی به دست آمده در تگزاس از وب سایت Dataport (۱ ماه)	داده‌های بار، سرعت باد، رطوبت، دما، تابش افقی جهانی، تابش افقی پراکنده، ساعت روز، روز هفته، روز ماه، ماه سال	LSTM-DRNN	[48]
پیش‌بینی بار متوسط و پیک ساعتی، روزانه و فصلی	-	نرمال سازی داده‌های ورودی Data Cleaning, Data Aggregation, Data Transformation	-	-	بار الکتریکی شهر چندی‌گر در هند (۵ سال)	داده‌های بار،	EMD-LSTM	[49]

[1] این مقاله در ابتدا، سری اطلاعات بار اصلی را توسط EMD به چند زیر سری بار تبدیل میکند.

DOI: 10.30503/nripress.2020.321

جدول ۵-۱: خلاصه‌ای از تحقیقات صورت گرفته شده در زمینه‌ی پیش‌بینی بار با استفاده از یادگیری عمیق (ادامه...)

دستاورد	تعداد لایه نهان - تعداد نورون- بچ سایز- نرخ یادگیری	پیش پردازش	استخراج ویژگی	انتخاب ویژگی	منبع داده (دوره)	ویژگی‌ها	مدل	مرجع
پیش‌بینی بار ساعتی	۲ (تعداد لایه)، ۲۰ (تعداد نورون در هر لایه)	نرمال سازی داده‌های ورودی	-	-	داده‌های مصرف حدود ۱۰،۰۰۰ مصرف کننده در New South Wales استرالیا (۳ ماه)	داده‌های بار، تعطیلات، ساعت روز، روز هفته	LSTM	[50]
پیش‌بینی بار ساعتی	-	کلاستربندی داده‌های ورودی	-	-	داده‌های مصرف ارائه شده توسط کمیسیون تنظیم مقررات انرژی ایرلند (۱ سال و ۶ ماه) شبکه هوشمند استرالیا (۴ سال)	داده‌های بار، حرارت، تعطیلات، ساعت روز، روز هفته	MTL-Bayesian theory	[51]
پیش‌بینی بار ساعتی	۳ (تعداد لایه)، ۶۴ (تعداد نورون)، ۰.۰۱ (نرخ یادگیری)	کلاستربندی داده‌های ورودی	-	Univariate selection, Recursive Feature Elimination, Lasso Regularization	داده‌های آنلاین کنتورهای هوشمند ایرلند (۱ سال و ۵ ماه)	داده‌های بار، دما، سرعت باد، رطوبت، پوشش ابر، فشار هوا ساعت روز، روز هفته	FF-ANN-Nesterov learning[1]	[52]
پیش‌بینی بار ساعتی	۵ (تعداد لایه)، ۱۰۰ (تعداد نورون)، ۰.۰۱ (نرخ یادگیری)	Data cleaning	-	-	داده‌های مصرف ۵۰۰۰ مصرف‌کننده خانگی ارائه شده توسط کمیسیون تنظیم مقررات انرژی ایرلند (۱ سال و ۵ ماه)	داده‌های بار	pooling[2]-based RNN	[53]
پیش‌بینی بار روزانه	۳ (تعداد لایه)، ۳۰ (تعداد نورون در هر لایه)، ۰.۰۱ (نرخ یادگیری)	کلاستربندی داده‌های ورودی	-	-	داده‌های مصرف در استرالیا در New South Wales (۴ سال)	داده‌های بار، داده‌های قیمت برق، دما، روز هفته، تعطیلات	GRU	[54]

[1] در این مقاله از شبکه عصبی پسخور پیشخور استفاده شده است که از الگوریتم Nesterov برای آموزش شبکه استفاده شده است.

[2] به منظور مقابله با مسئله ی overfit و عدم قطعیت پروفیل بارها، در ابتدا پروفیل بار خانوارها در یک استخر پروفیل بار جمع می‌شوند

DOI: 10.30503/nripress.2020.321

جدول ۵-۱: خلاصه‌ای از تحقیقات صورت گرفته شده در زمینه‌ی پیش‌بینی بار با استفاده از یادگیری عمیق (ادامه...)

دستاورد	تعداد لایه نهان - تعداد نورون- بچ سایز- نرخ یادگیری	پیش پردازش	استخراج ویژگی	انتخاب ویژگی	منبع داده (دوره)	ویژگی‌ها	مدل	مرجع
پیش‌بینی بار ساعتی	۲ (تعداد لایه نهان)، (هر لایه، ۱۰۰ نورون)، (بچ سایز، ۶۴)	-	-	-	داده‌های بار مصرف ۵ خانواده در شهر لندن انگلستان (۱ سال)	داده‌های بار	LSTM-SWT	[60]
تخمین تابع چگالی احتمال قیمت، پیش بینی سرعت باد و بار ساعتی	-	-	-	-	داده‌های بار مصرف خانگی	داده‌های بار، سرعت باد، قیمت برق	CNN-GRU	[61]
پیش‌بینی بار روزانه	-	-	-	-	داده‌های بار شبکه توان آفریقای جنوبی (۴ سال)	داده‌های بار، دما، ساعت روز، رطوبت	LSTM, OP-ELM	[62]
پیش‌بینی بار روزانه	-	-	-	-	داده‌های مصرف انرژی سه مجتمع توزیع بزرگ در کره جنوبی (۱ سال)	داده‌های بار،رطوبت، دما، سرعت باد	LSTM-CNN	[63]
پیش‌بینی بار روزانه کوتاه مدت و میان مدت	-	-	-	-	داده‌های مصرف انرژی شهر Hanzhou برای پیش بینی کوتاه مدت (۱ هفته و دو هفته) و داده‌های مصرف انرژی شهر تورنتو برای پیش بینی میان مدت مدت (۱ سال، ۶ ماه و ۱ماه)	داده‌های بار، رطوبت، دما، سرعت باد	LSTM, CNN	[64]
پیش‌بینی بار ساعتی	-	-	-	-	داده‌های بار مصرف شبکه قدرت تیانجین در چین (۱ سال)	داده‌های بار، دما، سرعت باد، تابش خورشید	شبکه باور عمیق[1]	[65]

[1] Deep Belief network

DOI: 10.30503/nripress.2020.321

جدول ۵-۱: خلاصه‌ای از تحقیقات صورت گرفته شده در زمینه‌ی پیش‌بینی بار با استفاده از یادگیری عمیق (ادامه...)

دستاورد	تعداد لایه نهان - تعداد نورون- بچ سایز- نرخ یادگیری	پیش پردازش	استخراج ویژگی	انتخاب ویژگی	منبع داده (دوره)	ویژگی‌ها	مدل	مرجع
پیش‌بینی بار ساعتی	۸ (تعداد لایه نهان)، (لایه اول ۲۰ نورون، لایه دوم ۱۰نورون، لایه سوم ۱۵ نورون، لایه چهارم ۱۷ نورون، لایه پنجم ۳۵ نورون، لایه ششم ۱۵ نورون، لایه هفتم ۲۵ نورون، لایه هشتم ۳۵ نورون)	-	-	-	داده‌های مصرف خنک کننده‌ها در یک ساختمان در هنگ کنگ (۱ سال)	داده‌های بار، دمای بیرونی ساختمان، تابش خورشید، رطوبت	شبکه باور عمیق-EMD	[66]
پیش‌بینی بار ساعتی		نرمال‌سازی داده‌های ورودی با استفاده از روش Box-Cox	-	-	داده‌های بار مصرف شهری در تگزاس و آرکانزاس در ایالات متحده (۱ سال)	داده‌های بار، دما، قیمت برق، رطوبت، فشار هوا	شبکه باور عمیق	[67]
پیش‌بینی بار ولتاژ پایین احتمالی و قطعی	-	-	-	-	داده‌های بار مصرف یکی در شرق چین (۹ ماه) و دیگری در ساحل شرقی استرالیا (۱ سال)	داده‌های بار و دما	شبکه باور عمیق	[68]
پیش‌بینی بار روزانه	-	-	-	-	داده‌های بار بدست آمده توسط اپراتور بازار انرژی استرالیا (۱ سال)	داده‌های بار	شبکه باور عمیق	[69]
پیش‌بینی بار ساعتی	(تعداد لایه نهان، ۶۴)، (بچ سایز، ۲۵۶)، (نرخ یادگیری، ۰.۰۰۱)	-	-	-	یک مجموعه داده شامل ۷ اندازه‌گیر هوشمند- مجموعه داده دوم شامل ۹ اندازه‌گیر هوشمند- مجموعه داده سوم شامل ۴۵۶ اندازه‌گیر هوشمند	داده‌های بار، روز سال، ماه سال، روز هفته، آخر هفته، روز ماه، شاخص ساعت، شاخص فصل، تعطیلات	یادگیری انتقال-RNN	[70]

DOI: 10.30503/nripress.2020.321

جدول ۵-۲: دسته بندی مقالات از نقطه نظر نوع مدل و نوآوری مدل

نوآوری در مدل	مدل											مرجع
	یادگیری تقویتی	Gated CNN	Feedforward ANN	Deep ANN	BFNN	شبکه باور عمیق	SVM	GRU	LSTM	RNN	CNN	
LSTM+ Bayesian theory									✓			[28]
										✓		[29]
								✓	✓			[30]
		✓						✓				[31]
									✓			[32]
									✓	✓		[33]
روش یادگیریQ	✓											[36]
رویکرد دو مرحله‌ای با روش یادگیریQ	✓											[37]
								✓		✓		[38]
Radial Basis Function به عنوان تابع فعال سازی نورون					✓							[40]
ترکیب یادگیری چند کاره با SVM حداقل مربعات							✓					[41]
									✓			[42]
ترکیب LSTM با GRU								✓	✓			[43]
									✓			[44]
تبدیل اطلاعات بار اصلی به چند زیر سری بار توسط EMD								✓				[45]
تطبیق منحنی بار با DTW								✓				[47]
									✓			[48]
تبدیل اطلاعات بار اصلی به چند زیر سری بار توسط EMD									✓			[49]
									✓			[50]
LSTM+ Bayesian theory +یادگیری چند کاره									✓			[51]
الگوریتم Nesterov برای آموزش شبکه			✓									[52]

DOI: 10.30503/nripress.2020.321

به منظور مقابله با مسئله ی overfit و عدم قطعیت پروفیل بارها، در ابتدا پروفیل بار خانوارها در یک استخر پروفیل بار جمع می‌شوند										✓		[53]

جدول ۵-۲: دسته بندی مقالات از نقطه نظر نوع مدل و نوآوری مدل (ادامه...)

نوآوری در مدل	مدل											مرجع
	یادگیری تقویتی	Gated CNN	Feedforw ard ANN	Deep ANN	BFNN	شبکه باور عمیق	SVM	GRU	LSTM	RNN	CNN	
								✓				[54]
							✓					[55]
ترکیب BFNN با CNN					✓						✓	[56]
استفاده از روش‌های یادگیری تقویتی: Recurrent Deterministic Policy Gradient, Deep Deterministic Policy Gradient, Asynchronous Advantage Actor-Critic	✓											[57]
									✓			[54]
				✓								[58]
				✓							✓	[59]
ترکیب LSTM با روش تبدیل موجک ثابت				✓					✓			[60]
ترکیب GRU با CNN								✓			✓	[61]
ترکیب LSTM با OP-ELM									✓			[62]
ترکیب LSTM با CNN									✓		✓	[63]
									✓		✓	[64]
در نظر گرفتن DSM و ماشین بولتزمن محدود گاوسی						✓						[65]
تبدیل اطلاعات بار اصلی به چند زیر سری بار توسط EMD						✓						[66]
روش تبدیل توان Box-Cox برای نرمال سازی						✓						[67]
تکنیک‌های bagging و boosting برای افزایش تطبیق پذیری شبکه باور عمیق						✓						[68]
تبدیل اطلاعات بار اصلی به چند زیر سری بار توسط EMD						✓						[69]
استفاده از رویکرد یادگیری انتقال						✓						[70]

DOI: 10.30503/nripress.2020.321

جدول ۵-۳: دسته بندی مقالات از نقطه نظر ویژگی‌ها

مرجع	ویژگی‌ها															
	داده های بار	سرعت باد	دما	تابش خورشید	داده های قیمت برق	میزان بارش	رطوبت	فشار هوا	پوشش ابر	روز هفته	ساعت روز	ماه سال	روز ماه	روز سال	شاخص کیفیت هوا	داده های تولید PV
[28]	✓	✓	✓				✓			✓		✓				
[29]	✓		✓							✓	✓					
[30]	✓	✓	✓				✓			✓		✓				
[31]	✓	✓	✓				✓	✓			✓					
[32]	✓									✓	✓	✓				✓
[33]	✓															
[36]	✓	✓	✓	✓			✓			✓	✓	✓				
[37]	✓	✓	✓	✓			✓	✓		✓	✓	✓				
[38]	✓	✓	✓				✓	✓		✓	✓	✓	✓			
[40]	✓	✓	✓					✓			✓			✓		
[41]	✓	✓	✓		✓		✓	✓		✓						
[42]	✓									✓	✓					
[43]	✓		✓				✓			✓	✓					
[44]	✓		✓							✓						
[45]	✓															
[47]	✓		✓							✓						
[48]	✓	✓	✓	✓			✓			✓		✓	✓			
[49]	✓															
[50]	✓									✓	✓					
[51]	✓		✓							✓	✓					
[52]	✓	✓	✓				✓	✓	✓	✓	✓					
[53]	✓															
[54]	✓		✓		✓					✓						
[55]	✓		✓							✓	✓	✓				
[56]	✓	✓	✓	✓		✓	✓	✓								
[57]	✓	✓	✓				✓									
[34]	✓	✓	✓	✓												
[58]	✓		✓							✓	✓	✓			✓	
[59]	✓	✓	✓				✓	✓		✓	✓			✓		
[60]	✓															
[61]	✓	✓			✓											
[62]	✓		✓				✓				✓					
[63]	✓	✓	✓				✓									
[64]	✓	✓	✓				✓									
[65]	✓	✓	✓	✓												
[66]	✓		✓	✓			✓									

DOI: 10.30503/nripress.2020.321

جدول ۵-۳: دسته بندی مقالات از نقطه نظر ویژگی‌ها (ادامه...)

مرجع	ویژگی‌ها															
	داده های بار	سرعت باد	دما	تابش خورشید	داده های	میزان بارش	رطوبت	فشار هوا	پوشش ابر	روز هفته	ساعت روز	ماه سال	روز ماه	روز سال	شاخص	داده های تولید PV
[67]	✓		✓		✓		✓	✓								
[68]	✓		✓													
[69]	✓															
[70]	✓									✓		✓	✓	✓		

جدول ۵–٤: دسته بندی مقالات از نقطه نظر پیش پردازش، انتخاب و استخراج ویژگی

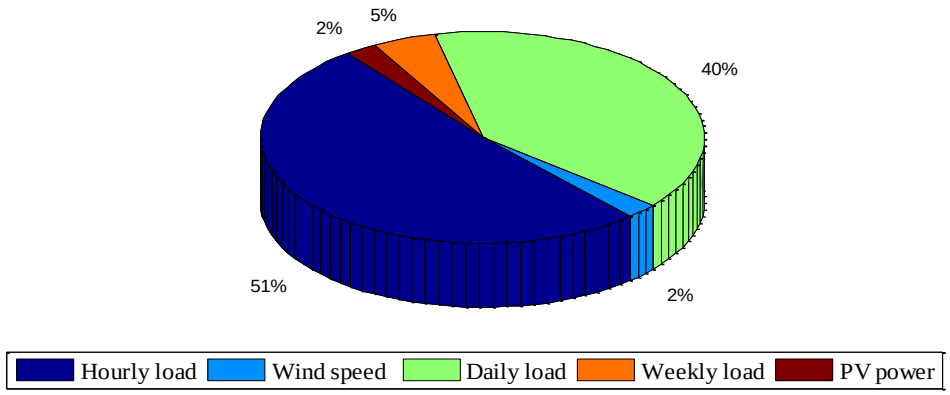
مرجع	دستاورد					استخراج ویژگی	انتخاب ویژگی	پیش پردازش
	پیش‌بینی توان PV	پیش‌بینی بار هفتگی	پیش‌بینی بار روزانه	پیش‌بینی سرعت باد	پیش‌بینی بار ساعتی			
[28]		✓					✓	✓
[29]			✓			✓		
[30]		✓	✓				✓	✓
[31]			✓					✓
[32]			✓					✓
[33]			✓					✓
[36]					✓			
[37]					✓			
[38]					✓			
[40]					✓	✓		
[41]					✓			✓
[42]					✓			
[43]			✓					✓
[44]			✓				✓	✓
[45]			✓				✓	
[47]			✓					✓
[48]	✓		✓					✓
[49]			✓		✓			✓
[50]					✓			✓
[51]					✓			✓
[52]					✓		✓	✓
[53]					✓			✓
[54]					✓			✓
[55]					✓	✓		
[56]			✓					✓
[57]					✓	✓		✓
[34]					✓			
[58]			✓			✓		✓
[59]					✓		✓	
[60]					✓			
[61]				✓	✓			
[62]			✓					
[63]			✓					
[64]			✓					
[65]					✓			

DOI: 10.30503/nripress.2020.321

جدول ۵-۴: دسته بندی مقالات از نقطه نظر پیش پردازش، انتخاب و استخراج ویژگی (ادامه...)

مرجع	دستاورد					استخراج ویژگی	انتخاب ویژگی	پیش پردازش
	پیش‌بینی توان PV	پیش‌بینی بار هفتگی	پیش‌بینی بار روزانه	پیش‌بینی سرعت باد	پیش‌بینی بار ساعتی			
[66]					✓			
[67]					✓			✓
[68]					✓			
[69]			✓					
[70]					✓			

یک آنالیز آماری از مقالات دسته‌بندی شده در زمینه نوع دستاورد در شکل ۵-۹ نشان داده شده است. همانطور که پیداست، ۵۱ درصد مقالات، از روش یادگیری عمیق برای پیش‌بینی ساعتی بار استفاده کرده‌اند. ۴۰ درصد مقالات نیز مفهوم یادگیری عمیق را برای پیش‌بینی روزانه بار به کار برده‌اند. همچنین، ۵ درصد مقالات از یادگیری عمیق برای پیش‌بینی بار هفتگی استفاده کرده‌اند. ۲ درصد مقالات هم برای پیش‌بینی سرعت باد و ۲ درصد دیگر برای پیش‌بینی توان خروجی فتوولتاییک از روش یادگیری عمیق بهره برده‌اند.

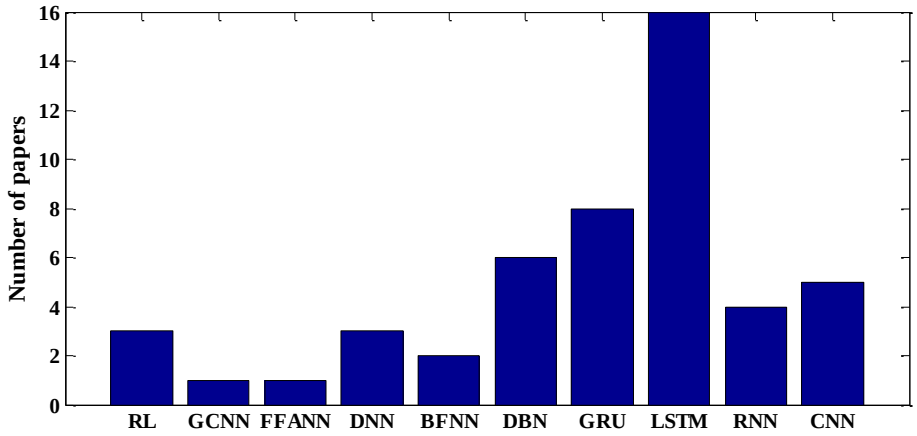


شکل ۵–۹: آنالیز آماری از مقالات دسته‌بندی شده در زمینه نوع دستاورد

شکل ۵-۱۰ توزیع مدل‌های مختلف مبتنی بر یادگیری عمیق در پیش‌بینی بار را نشان می‌دهد. از آنجا که یک مقاله می‌تواند به چندین کاربرد در یک مدل پیش‌بینی خاص اقدام کند، توزیع هر مدل پیش‌بینی از طریق مقالات بررسی شده بازنمایی بهتری را ارائه می‌دهد. بنابراین، نوع توزیع به عنوان تعدادی از مقالات در نظر گرفته می‌شود که در آن از یک مدل پیش‌بینی خاص استفاده می‌شود و نه تعداد کاربردها. همانطور که از نتیجه پیداست، مدل شبکه عصبی LSTM، GRU و DBN به ترتیب بیشترین سهم در تعداد مقاله را دارند. از نقطه نظر نوع نوآوری در مقالات که در جدول ۵-۲ خلاصه شده است، مشهود است که تعداد ۴ مقاله از روش EMD استفاده کرده است به این صورت

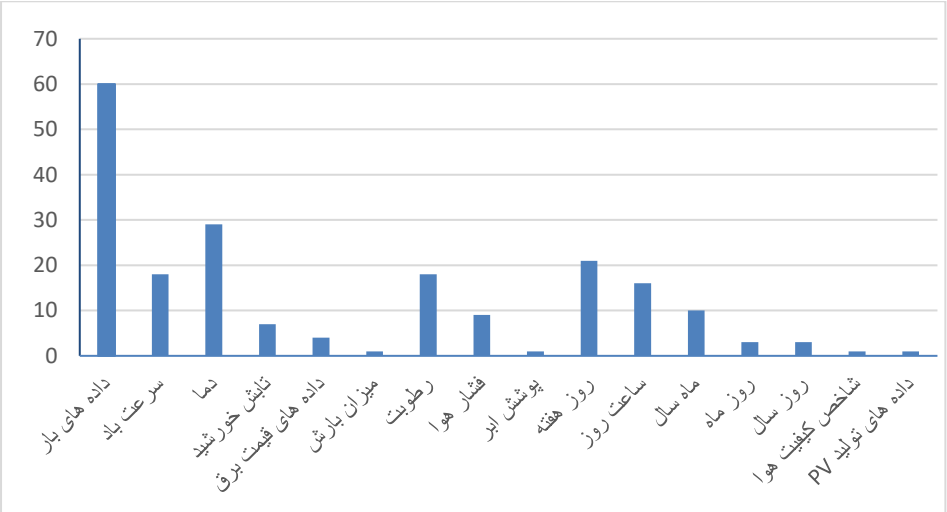
DOI: 10.30503/nripress.2020.321

که از روش ذکر شده برای تبدیل کردن اطلاعات بار اصلی به چند زیر سری بار استفاده شده است. همچنین، تعداد ۶ مقاله از ترکیب کردن روش LSTM با رویکردها و مدل‌های شبکه عصبی دیگر استفاده کرده است، برای مثال، تعداد دو مقاله از ترکیب کردن روش LSTM با روش تئوری بیضین برای پیش‌بینی بار کوتاه مدت استفاده کرده است. تعداد ۳ مقاله نیز رویکردهای یادگیری تقویتی مانند روش یادگیری Q و روش PG را برای هدف پیش‌بینی بار به کار گرفته‌اند.



شکل ۵–۱۰: توزیع مدل‌های مختلف مبتنی بر یادگیری عمیق در پیش‌بینی بار

به طور مشابه، ویژگی‌های مورد استفاده در مقالات متمرکز بر کاربرد یادگیری عمیق در پیش‌بینی بار در شکل ۵-۱۱ خلاصه شده‌اند. همانطور که از نتیجه پیداست، ویژگی دما بعد از ویژگی داده‌های مصرف بیشترین سهم در تعداد مقاله (یعنی حدود ۳۰ مقاله از ۶۰ مقاله) را دارد. بعد از آن، ویژگی‌های سرعت باد، رطوبت و روز هفته در تعداد بیشتری از مقالات به عنوان ورودی به مدل شبکه عصبی مبتنی بر یادگیری عمیق در نظر گرفته شده‌اند.



شکل ۵–۱۱: توزیع ویژگی‌های استفاده شده در مقالات متمرکز بر کاربرد یادگیری عمیق در پیش‌بینی بار

مراجع

[1] T. Mitchell, Machine Learning. New York: McGraw Hill. ISBN 0-07-042807-7. OCLC 36417892, 2009.

[2] E. Alpaydin, "Introduction to Machine Learning", 2009, MIT Press. p. 9. ISBN 978-0-262-01243-0.

[3] Y. Bengio, A. Courville, P. Vincent,. "Representation Learning: A Review and New Perspectives". *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 2003, 1798–1828. arXiv:1206.5538. doi:10.1109/tpami.2013.50. PMID 23787338. S2CID 393948.

[4] M. Najafabadi, F. Villanustre, T. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic. "Deep learning applications and challenges in big data analytics. Journal of Big Data", 2(1):1, 2015.

[5] J. Schmidhuber. "Deep Learning in Neural Networks: An Overview". *Neural Networks*, 2015, 61: 85–117. arXiv:1404.7828. doi:10.1016/j.neunet.2014.09.003. PMID 25462637. S2CID 11715509.

[6] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, 2014. Generative adversarial networks. arXiv preprint arXiv:1406.2661.

[7] Miljanovic, Milos (Feb–Mar 2012). "Comparative analysis of Recurrent and Finite Impulse Response Neural Networks in Time Series Prediction" (PDF). Indian Journal of Computer and Engineering. 3 (1).

[8] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. van den Oord, A. Graves, and K. Kavukcuoglu, "Neural machine translation in linear time," *arXiv preprint arXiv:1610.10099*, 2016.

[9] Kaelbling, Leslie P.; Littman, Michael L.; Moore, Andrew W. (1996). "Reinforcement Learning: A Survey". Journal of Artificial Intelligence Research. 4: 237–285.

[10] [Online].https://towardsdatascience.com/intro-to-reinforcement-learning-pong 92a94aa0f84d

[11] N. Amral, C. Ozveren, and D. King, "Short term load forecasting using multiple linear regression," in *2007 42nd International universities power engineering conference*, 2007, pp. 1192-1198: IEEE.

[12] G. E. Box and D. A. Pierce, "Distribution of residual autocorrelations in autoregressive-integrated moving average time series models," *Journal of the American statistical Association*, vol. 65, no. 332, pp. 1509-1526, 1970.

[13] W. Christiaanse, "Short-term load forecasting using general exponential smoothing," *IEEE Transactions on Power Apparatus and Systems*, no. 2, pp. 900-911, 1971.

[14] C. Wu, B. Li, and J. Zheng, "A speech enhancement method based on kalman filtering," *International Journal of Wireless and Microwave Technologies (IJWMT)*, vol. 1, no. 2, p. 55, 2011.

[15] M. Kandil, S. M. El-Debeiky, and N. Hasanien, "Long-term load forecasting for fast developing utility using a knowledge-based expert system," *IEEE transactions on Power Systems*, vol. 17, no. 2, pp. 491-496, 2002.

[16] Z. Dianmin, G. Xiaohong, S. Jie, and H. Yong, "A short-term load forecasting system based on BP artificial neural network," *POWER SYSTEM TECHNOLOGY-BEIJING-*, vol. 26, no. 2, pp. 10-13, 2002

[17] J. Che and J. Wang, "Short-term load forecasting using a kernel-based support vector regression combination model," *Applied energy*, vol. 132, pp. 602-609, 2014.

[18] S. Kouhi, F. Keynia, and S. N. Ravadanegh, "A new short-term load forecast method based on neuro-evolutionary algorithm and chaotic feature selection," *International Journal of Electrical Power & Energy Systems*, vol. 62, pp. 862-867, 2014.

[19] P. Singh and P. Dwivedi, "Integration of new evolutionary approach with artificial neural network for solving short term load forecast problem," *Applied energy*, vol. 217, pp. 537-549, 2018.

[20] J. Vermaak and E. Botha, "Recurrent neural networks for short-term load forecasting," *IEEE Transactions on Power Systems*, vol. 13, no. 1, pp. 126-132, 1998.

[21] N. Liu, Q. Tang, J. Zhang, W. Fan, and J. Liu, "A hybrid forecasting model with parameter optimization for short-term load forecasting of micro-grids," *Applied Energy*, vol. 129, pp. 336-345, 2014.

[22] S. Kouhi and F. Keynia, "A new cascade NN based method to short-term load forecast in deregulated electricity market," *Energy Conversion and Management*, vol. 71, pp. 76-83, 2013.

[23] R.-A. Hooshmand, H. Amooshahi, and M. Parastegari, "A hybrid intelligent algorithm based short-term load forecasting approach," *International Journal of Electrical Power & Energy Systems*, vol. 45, no. 1, pp. 313-324, 2013.

[24] G.-F. Fan, L.-L. Peng, and W.-C. Hong, "Short term load forecasting based on phase space reconstruction algorithm and bi-square kernel regression model," *Applied Energy*, vol. 224, pp. 13-33, 2018.

[25] S. Li, L. Goel, and P. Wang, "An ensemble approach for short-term load forecasting by extreme learning machine," *Applied Energy*, vol. 170, pp. 22-29, 2016.

[26] P. Zhang, X. Wu, X. Wang, and S. Bi, "Short-term load forecasting based on big data technologies," *CSEE Journal of Power and Energy Systems*, vol. 1, no. 3, pp. 59-67, 2015.

[27] F. L. Quilumba, W.-J. Lee, H. Huang, D. Y. Wang, and R. L. Szabados, "Using smart meter data to improve the accuracy of intraday load forecasting considering customer behavior similarities," *IEEE Transactions on Smart Grid*, vol. 6, no. 2, pp. 911-918, 2014.

[28] S. Bouktif, A. Fiaz, A. Ouni, and M. A. Serhani, "Optimal deep learning lstm model for electric load forecasting using feature selection and genetic algorithm: Comparison with machine learning approaches," *Energies*, vol. 11, no. 7, p. 1636, 2018.

[29] He,W. Load forecasting via deep neural networks. Procedia Comput. Sci., 122, 308–314, 2017.

[30] S. Bouktif, A. Fiaz, A. Ouni, and M. A. Serhani, "Single and multi-sequence deep learning models for short and medium term electric load forecasting," *Energies*, vol. 12, no. 1, p. 149, 2019.

[31] M. Cai, M. Pipattanasomporn, and S. Rahman, "Day-ahead building-level load forecasts using deep learning vs. traditional time-series techniques," *Applied energy,* vol. 236, pp. 1078-1088, 2019.

[32] M. Sun, T. Zhang, Y. Wang, G. Strbac, and C. Kang, "Using Bayesian deep learning to capture uncertainty for residential net load forecasting," *IEEE Transactions on Power Systems,* vol. 35, no. 1, pp. 188-201, 2019.

[33] J. Bedi and D. Toshniwal, "Deep learning framework to forecast electricity demand," *Applied energy,* vol. 238, pp. 1312-1326, 2019.

[34] S. Muzaffar and A. Afshari, "Short-term load forecasts using LSTM networks," *Energy Procedia,* vol. 158, pp. 2922-2927, 2019.

[35] Y. Qin *et al.*, "Hybrid forecasting model based on long short term memory network and deep learning neural network for wind signal," *Applied Energy,* vol. 236, pp. 262-272, 2019.

[36] Feng, Cong, and Jie Zhang. "Reinforcement learning based dynamic model selection for short-term load forecasting." 2019 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT). IEEE, 2019.

[37] C. Feng, M. Sun, and J. Zhang, "Reinforced deterministic and probabilistic load forecasting via Q-learning dynamic model selection," *IEEE Transactions on Smart Grid,* 2019.

[38] L. Wen, K. Zhou, and S. Yang, "Load demand forecasting of residential buildings using a deep learning model," *Electric Power Systems Research,* vol. 179, p. 106073, 2020.

[39] A. Estebsari and R. Rajabi, "Single Residential Load Forecasting Using Deep Learning and Image Encoding Techniques," *Electronics,* vol. 9, no. 1, p. 68, 2020.

[40] A. O. Hoori, A. Al Kazzaz, R. Khimani, Y. Motai, and A. J. Aved, "Electric Load Forecasting Model using a Multi-Column Deep Neural Networks," *IEEE Transactions on Industrial Electronics,* 2019.

[41] Z. Tan *et al.*, "Combined electricity-heat-cooling-gas load forecasting model for integrated energy system based on multi-task learning and least square support vector machine," *Journal of Cleaner Production,* vol. 248, p. 119-252, 2020.

[42] Y. Wang, D. Gan, M. Sun, N. Zhang, Z. Lu, and C. Kang, "Probabilistic individual load forecasting using pinball loss guided LSTM," *Applied Energy,* vol. 235, pp. 10-20, 2019.

[43] X. Tang, Y. Dai, T. Wang, and Y. Chen, "Short-term power load forecasting based on multi-layer bidirectional recurrent neural network," *IET Generation, Transmission & Distribution,* vol. 13, no. 17, pp. 3847-3854, 2019.

[44] Z. Wang, B. Zhao, H. Guo, L. Tang, and Y. Peng, "Deep Ensemble Learning Model for Short-Term Load Forecasting within Active Learning Framework," *Energies,* vol. 12, no. 20, p. 3809, 2019.

[45] X. Gao, X. Li, B. Zhao, W. Ji, X. Jing, and Y. He, "Short-term electricity load forecasting model based on EMD-GRU with feature selection," *Energies,* vol. 12, no. 6, p. 1140, 2019.

[46] M. Zahid *et al.*, "Electricity price and load forecasting using enhanced convolutional neural network and enhanced support vector regression in smart grids," *Electronics*, vol. 8, no. 2, p. 122, 2019.

[47] Z. Yu, Z. Niu, W. Tang, and Q. Wu, "Deep learning for daily peak load forecasting–a novel gated recurrent neural network combining dynamic time warping," *IEEE Access*, vol. 7, pp. 17184-17194, 2019.

[48] L. Wen, K. Zhou, S. Yang, and X. Lu, "Optimal load dispatch of community microgrid with deep learning based solar power and load forecasting," *Energy*, vol. 171, pp. 1053-1065, 2019.

[49] J. Bedi and D. Toshniwal, "Empirical mode decomposition based deep learning for electricity demand forecasting," *IEEE Access*, vol. 6, pp. 49144-49156, 2018.

[50] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-term residential load forecasting based on LSTM recurrent neural network," *IEEE Transactions on Smart Grid*, vol. 10, no. 1, pp. 841-851, 2017.

[51] Y. Yang, W. Li, T. A. Gulliver, and S. Li, "Bayesian Deep Learning Based Probabilistic Load Forecasting in Smart Grids," *IEEE Transactions on Industrial Informatics*, 2019.

[52] S.-V. Oprea and A. Bâra, "Machine Learning Algorithms for Short-Term Load Forecast in Residential Buildings Using Smart Meters, Sensors and Big Data Solutions," *IEEE Access*, vol. 7, pp. 177874-177889, 2019.

[53] H. Shi, M. Xu, and R. Li, "Deep learning for household load forecasting—A novel pooling deep RNN," *IEEE Transactions on Smart Grid*, vol. 9, no. 5, pp. 5271-5280, 2017.

[54] W. Wu, W. Liao, J. Miao, and G. Du, "Using Gated Recurrent Unit Network to Forecast Short-Term Load Considering Impact of Electricity Price," *Energy Procedia*, vol. 158, pp. 3369-3374, 2019.

[55] C. Tong, J. Li, C. Lang, F. Kong, J. Niu, and J. J. Rodrigues, "An efficient deep model for day-ahead electricity load forecasting with stacked denoising auto-encoders," *Journal of Parallel and Distributed Computing*, vol. 117, pp. 267-273, 2018.

[56] G. Sideratos, A. Ikonomopoulos, and N. D. Hatziargyriou, "A novel fuzzy-based ensemble model for load forecasting using hybrid deep neural networks," *Electric Power Systems Research*, vol. 178, p. 106025, 2020.

[57] T. Liu, Z. Tan, C. Xu, H. Chen, and Z. Li, "Study on deep reinforcement learning techniques for building energy consumption forecasting," *Energy and Buildings*, vol. 208, p. 109675, 2020.

[58] Z. Guo, K. Zhou, X. Zhang, and S. Yang, "A deep learning model for short-term power load and probability density forecasting," *Energy*, vol. 160, pp. 1186-1200, 2018.

[59] X. Liu, Z. Zhang, and Z. Song, "A comparative study of the data-driven day-ahead hourly provincial load forecasting methods: From classical data mining to deep learning," *Renewable and Sustainable Energy Reviews*, vol. 119, p. 109632, 2020.

[60] K. Yan, W. Li, Z. Ji, M. Qi, and Y. Du, "A Hybrid LSTM Neural Network for Energy Consumption Forecasting of Individual Households," *IEEE Access*, vol. 7, pp. 157633-157642, 2019.

[61] M. Afrasiabi, M. Mohammadi, M. Rastegar, and A. Kargarian, "Probabilistic deep neural network price forecasting based on residential load and wind speed predictions," *IET Renewable Power Generation*, vol. 13, no. 11, pp. 1840-1848, 2019.

[62] S. Motepe, A. N. Hasan, and R. Stopforth, "Improving Load Forecasting Process for a Power Distribution Network Using Hybrid AI and Deep Learning Algorithms," *IEEE Access*, vol. 7, pp. 82584-82598, 2019.

[63] J. Kim, J. Moon, E. Hwang, and P. Kang, "Recurrent inception convolution neural network for multi short-term load forecasting," *Energy and Buildings*, vol. 194, pp. 328-341, 2019.

[64] L. Han, Y. Peng, Y. Li, B. Yong, Q. Zhou, and L. Shu, "Enhanced deep networks for short-term and medium-term load forecasting," *IEEE Access*, vol. 7, pp. 4045-4055, 2018.

[65] X. Kong, C. Li, F. Zheng, and C. Wang, "Improved Deep Belief Network for Short-Term Load Forecasting Considering Demand-Side Management," *IEEE Transactions on Power Systems*, 2019.

[66] G. Fu, "Deep belief network based ensemble approach for cooling load forecasting of air-conditioning system," *Energy*, vol. 148, pp. 269 282, 2018.

[67] T. Ouyang, Y. He, H. Li, Z. Sun, and S. Baek, "Modeling and forecasting short-term power load with copula model and deep belief network," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 3, no. 2, pp. 127-136, 2019.

[68] Z. Cao, C. Wan, Z. Zhang, F. Li, and Y. Song, "Hybrid Ensemble Deep Learning for Deterministic and Probabilistic Low-voltage Load Forecasting," IEEE Transactions on Power Systems, 2019.

[69] X. Qiu, Y. Ren, P. N. Suganthan, and G. A. Amaratunga, "Empirical mode decomposition based ensemble deep learning for load demand time series forecasting," *Applied Soft Computing*, vol. 54, pp. 246-255, 2017.

[70] Y. Tian, L. Sehovac, and K. Grolinger, "Similarity-based chained transfer learning for energy forecasting with Big Data," *IEEE Access*, vol. 7, pp. 139895-139908, 2019.

DOI: 10.30503/nripress.2020.321

فصل ٦- پایتون

DOI: 10.30503/nripress.2020.321

هنگامی که صحبت از یادگیری ماشین[۱]، هوش مصنوعی[۲] ، یادگیری عمیق[۳] و وظایف علم داده[۴] می‌شود، نام پایتون[۵] پیشگام است. براساس آمار فرآهم شده توسط سایت «builtwith»، در حدود ٪۴۵ از شرکت‌های فناوری از زبان پایتون برای پیاده‌سازی هوش مصنوعی و یادگیری ماشین استفاده می‌کنند. از طرف دیگر، پیاده‌سازی شبکه‌های عصبی مصنوعی عمیق در پایتون نیاز به استفاده از یک سری کتابخانه‌های نرم افزاری دارد که از مهم‌ترین آنها می‌توان به کتابخانه تنسورفلو[۶] اشاره کرد. در این فصل مزایا و معایب پایتون و هچنین قابلیت‌های کتابخانه تنسورفلو ارائه گردیده است.

٦-١- پایتون

پایتون یک زبان برنامه نویسی سطح بالا تفسیر شده برای برنامه‌نویسی عمومی است. این زبان دارای یک فلسفه طراحی است که برای خواندن کد مبتنی بر استفاده از space و tab می‌باشد. پایتون دارای یک سیستم نوع پویا و مدیریت حافظه خودکار است و پارادایم‌های چندگانه برنامه‌نویسی را پشتیبانی می‌کند. در ذیل، مزایا و معایب نرم افزار پایتون معرفی شده‌اند [۲-۱]:

٦-١-١- مزایای پایتون

مزایای برنامه‌نویسی در نرم‌افزار پایتون را می‌توان به صورت زیر اشاره کرد:

- حضور ماژول‌های شخص ثالث

پایتون شامل چندین ماژول شخص ثالث است که باعث می‌شود پایتون بتواند با بسیاری از زبان‌ها و سیستم عامل‌های دیگر ارتباط برقرار کند.

- پشتیبانی کتابخانه‌های گسترده

پایتون کتابخانه استاندارد بزرگی را ارائه می‌دهد که شامل موضوعات مختلف مانند پروتکل اینترنت، عملیات رشته، ابزارها و سرویس‌های وب و رابط‌های سیستم عامل است. بسیاری از کارهای برنامه‌نویسی پر کاربرد قبلاً در کتابخانه استاندارد نگاشته شده‌اند که باعث می‌شود طول کد به طور قابل توجهی کاهش داده شود.

- منبع باز

---

[۱] Machine Learning

[۲] Artificial Intelligence

[۳] Deep Learning

[۴] Data Science

[۵] Python

[۶] Tensorflow

DOI: 10.30503/nripress.2020.321

زبان پایتون تحت مجوز OSI[1] تأیید شده است که استفاده و توزیع آن را آزاد می‌کند، از جمله برای اهداف تجاری. علاوه بر این، توسعه آن توسط جامعه‌ای انجام می‌شود که از طریق میزبانی کنفرانس‌ها، برای کد آن همکاری می‌کنند و ماژول های بی‌شماری را برای توسعه آن فراهم می‌کنند.

- یادگیری سریع و آسان

پایگاه گسترده کاربران و توسعه دهندگان فعال باعث شده است تا یک بانک منابع اینترنتی غنی برای ترغیب توسعه و ادامه پذیرش زبان ایجاد شود.

- ساختار داده‌های کاربر پسند

پایتون دارای ساختار داخلی داده‌ها و فرهنگ‌نامه‌ها است که می‌تواند برای ساخت سریع داده‌های زمان اجرا استفاده شود.

- بهره‌وری و سرعت

پایتون قابلیت‌های کنترل پیشرفته یک فرایند را فراهم می‌کند، و توانایی‌های ادغام و پردازش متن دارد، که همه این عوامل به افزایش سرعت و بهره‌وری آن کمک می‌کند. پایتون گزینه‌ای مناسب برای ساخت برنامه‌های پیچیده دارای چند پروتکل تحت شبکه محسوب می‌شود.

٦-١-٢- معایب پایتون

- سرعت

پایتون کندتر از C یا C++ است. پایتون یک زبان سطح بالا است ، برخلاف C یا C++ به سخت افزار نزدیک نیست.

- توسعه موبایل

پایتون یک زبان خیلی خوب برای توسعه موبایل نیست. این یک زبان ضعیف برای محاسبات موبایل است. به همین دلیل است که برنامه‌های اندکی در تلفن‌های همراه مانند Carbonnelle در آن ساخته شده‌اند.

- مصرف حافظه

پایتون برای کارهای فشرده حافظه گزینه مناسبی نیست. به دلیل انعطاف‌پذیری انواع داده‌ها، میزان مصرف حافظه پایتون نیز زیاد است.

- خطاهای زمان اجرا

[1] Open Simulation Interface

DOI: 10.30503/nripress.2020.321

برنامه‌نویسان پایتون در زمینه طراحی زبان چندین موضوع را ذکر کردند. از آنجا که این زبان یک تعریف نوع متغیر پویا می باشد به صورت پویا تایپ می‌شود (نیاز به تعریف نوع متغیر ندارد)، به آزمایش بیشتری نیاز دارد و دارای خطاهایی است که فقط در زمان اجرا مشخص می‌شود.

٦-٢- کتابخانه تنسورفلو

تنسورفلو، یک کتابخانه رایگان و متن‌باز[۱] قدرتمند برای محاسبات عددی است که کاربردهای گوناگونی در یادگیری ماشین دارد، که از آن جمله می‌توان به پیاده‌سازی شبکه‌های عصبی[۲] اشاره کرد. این کتابخانه توسط تیم گوگل برین[۳]، برای مصارف داخلی گوگل توسعه داده شده بود؛ ولی در نهم نوامبر سال ۲۰۱۵ با گواهینامه «آپاچی متن‌باز ۲» منتشر شد. در حال حاضر، کتابخانه تنسورفلو، در گوگل هم برای پروژه‌های تحقیقاتی و هم پروژه‌های عملیاتی مورد استفاده قرار می‌گیرد [۲].

تنسورفلو قابلیت اجرا روی cpuها و gpuهای مختلف را دارد و همچنین قادر است که در زبان‌های مختلف مانند C++، پایتون و یا جاوا مورد استفاده قرار بگیرد. نام تنسورفلو از عملیاتی گرفته شده است که شبکه‌های عصبی روی آرایه‌های داده چندبعدی که از آن‌ها با عنوان تنسور یاد می‌شود، انجام می‌دهند. معماری تنسورفلو به گونه‌ای است که دارای سه بخش پیش پردازش داده‌ها، ساخت مدل و تعلیم و تخمین مدل است. یک تنسور آرایه‌ای از ماتریس‌های N بعدی هست که می‌توانند انواع اطلاعات را نمایش دهند. تنسورها می‌توانند ورودی یا خروجی یک محاسبه باشند. در تنسورفلو همه عملیات‌ها داخل گراف انجام می‌شوند. هر گراف شامل مجوعه‌ای از محاسبات هست که پیوسته انجام می‌شوند. همه محاسبات در گراف با اتصال همه تنسورها به یکدیگر اجرا می‌شوند. در هر گراف، هر یال نشان دهنده‌ی یک مقدار (تنسور) است و هر گره بیانگر یک عملگر (مثلا جمع) می‌باشد.

---

[۱] Open Source

[۲] Neural Network

[۳] Google Brain

DOI: 10.30503/nripress.2020.321

فصل ۷- ساختار LSTM برای ورودی بار، ویژگی‌های آب و هوایی و ویژگی‌های تقویمی

DOI: 10.30503/nripress.2020.321

در این فصل، نحوه کدنویسی و به‌کارگیری کتابخانه‌ها، کلاس‌ها و توابع مورد نیاز جهت پیش‌بینی یک سری بار نمونه (داده‌های بار شبکه توزیع تهران) توسط شبکه عصبی LSTM فراهم شده است. همچنین، سعی شده است که کدها به ترتیب تعریف و توضیح داده شوند.

۷-۱- فراخوانی کتابخانه‌های مورد نیاز

در این بخش، کتابخانه‌های اصلی که نیاز هستند به کد وارد شوند، معرفی شده‌اند.

۷-۱-۱- کتابخانه پانداس[۱]

تحلیل داده یکی از مهم‌ترین مهارت‌های دنیای امروزی است. یکی از مهم‌ترین ابزارهایی که محققین داده با آن سروکار دارند، کتابخانه پانداس است. پانداس یک کتابخانه قدرتمند و متن‌باز[۲] است که برای زبان برنامه‌نویسی پایتون توسعه داده شده است و عمده کاربرد آن در دست‌کاری[۳] و تجزیه‌وتحلیل داده‌ها است. پانداس یک ابزار سریع، قدرتمند، انعطاف‌پذیر و ساده است. بخش‌های اصلی این کتابخانه در زبان پایتون یا زبان C نوشته شده است و به همین دلیل از نظر عملکرد، کارایی بالایی دارد. پانداس به محققین و مهندسان داده کمک می‌کند که داده‌های خود را در قالب جداول رابطه‌ای وارد حافظه کند و روی آن‌ها انواع پردازش‌ها، رسم نمودارها، تحلیل‌ها و به‌طور کلی هر کاری را که نیاز است، به بهترین شکل ممکن انجام دهد. به‌طور اختصاصی، پانداس ساختمان داده و عملیاتی را برای دست‌کاری جدول‌های عددی و سری‌های زمانی پیشنهاد می‌دهد. پانداس عموما برای یادگیری ماشین و به شکل فریم‌های داده به‌کار گرفته می‌شود. یکی از ویژگی‌های منحصربفرد این کتابخانه این است که می‌تواند با انواع داده‌ها با فرمت‌های متفاوتی مثل csv، excel و غیره کار کند.

نحوه فراخوانی کتابخانه پانداس در پایتون به صورت زیر است:

```
1- import pandas as pd
```

۷-۱-۲- کتابخانه Numpy

NumPy یک کتابخانه برای زبان برنامه نویسی پایتون است. با استفاده از این کتابخانه امکان استفاده از آرایه‌ها و ماتریس‌های بزرگ چند بعدی فراهم می‌شود. هدف اصلی NumPy فراهم ساختن امکان کار با آرایه‌های چندبعدی همگن است. این آرایه‌ها جدولی از عناصر (معمولاً اعداد) هستند که همگی از یک نوع می‌باشند و با یک چندتایی، از اعداد صحیح مثبت اندیس‌گذاری می‌شوند. کتابخانه NumPy برای ذخیره‌ی داده‌ها از حافظه‌ی خیلی کمتری استفاده

---

[۱] Pandas

[۲] Open-source

[۳] Manipulation

DOI: 10.30503/nripress.2020.321

می‌کند. در کتابخانه NumPy برای تمام محاسبات ماتریسی، اپراتور تعریف شده است که به سادگی می‌توان محاسبات را انجام داد و نیاز به تعریف کدهای طولانی و ریاضیاتی نیست.

نحوه فراخوانی کتابخانه NumPy در پایتون به صورت زیر است:

```
2-  import numpy as np
```

۷-۱-۳- کتابخانه Matplotlib

کتابخانه Matplotlib اولین کتابخانه مصورسازی داده‌ها بوسیله زبان پایتون می‌باشد که بسیار رایج و پرکاربرد است و بر اساس آرایه‌های NumPy ساخته شده است. کتابخانه Matplotlib یک کتابخانه جامع و قدرتمند برای طراحی نمودارهای ثابت، انیمیشنی و تعاملی در پایتون است.

نحوه فراخوانی کتابخانه Matplotlib در پایتون به صورت زیر است:

```
3-  import matplotlib.pyplot as plt
```

۷-۱-٤- کتابخانه joblib

کتابخانه joblib امکان استفاده از پردازش موازی را فرآهم می‌کند. دلایل مختلفی برای استفاده از کتابخانه joblib وجود دارد که می توان به نکات زیر اشاره کرد: ۱- توان پردازشی متغیرهای ورودی را افزایش می دهد. ۲- موجب صرفه جویی در زمان می شود. ۳- می تواند به راحتی یکپارچه شود.

در این پروژه، از کتابخانه joblib برای بالا بردن سرعت پردازش متغیرهای هواشناسی استفاده شده است.

نحوه فراخوانی کتابخانه joblib در پایتون به صورت زیر است:

```
4-  from joblib import Parallel, delayed
```

همانطور که مشاهده می گردد، با استفاده از کتابخانه joblib، می توان کلاس Parallel و متد delayed را تعریف کرد. در بخش ۲-۴ نحوه استفاده از این کتابخانه نمایش داده شده است.

۷-۱-۵- تعریف ماژول[1]

هر چه برنامه مورد نظر بزرگ‌تر شود بهترین کار این است که برنامه به بخش‌های کوچک‌تر تبدیل شود. این کار توسط ماژول‌های پایتون امکان‌پذیر است. به عبارت دیگر، به جای تعریف مکرر توابع پر کاربرد در برنامه، می‌توان یک بار آن‌ها را در یک ماژول نوشته و هرگاه که لازم باشد آن ماژول یا بخشی از آن در برنامه وارد شود.

نحوه تعریف ماژول در پایتون به صورت زیر است:

```
5-  from modules.preparing_module import varPreparing
6-  from modules.preparing_module import utilities
```

[1] Module

همانطور که مشاهده می‌شود، دو ماژول به نام modules.preparing_module با دو کلاس به اسم varPreparing و utilities تعریف شده‌اند. در بخش ۲-۴- نحوه استفاده از این دو ماژول نمایش داده شده است.

۷-۱-۶- کتابخانه math

تابع math برای عملیات ریاضی استفاده می‌شود.

نحوه فراخوانی کتابخانه math در پایتون به صورت زیر است:

```
7- import math
```

۷-۱-۷- کتابخانه Keras

Keras یک کتابخانه رایگان منبع باز قدرتمند و با کاربرد آسان برای توسعه و ارزیابی مدل‌های یادگیری عمیق است. keras دو کتابخانه یادگیری ماشین عددی تیانو[۱] و تنسورفلو را پوشش می‌دهد و این امکان را فرآهم می‌کند که مدل‌های شبکه عصبی در چند خط تعریف و آموزش داده شوند. بنابراین، وارد کردن کتابخانه Keras برای ایجاد شبکه عصبی مصنوعی لازم است. Keras به طور پیش‌فرض از تنسورفلو به عنوان بک‌اند خود استفاده می‌کند. در ادامه، نیاز به وارد کردن ماژول Sequential برای مقداردهی اولیه به شبکه عصبی و ماژول Dense برای ساخت لایه‌های شبکه عصبی مصنوعی است. در این پروژه، مدل شبکه عصبی استفاده شده، شبکه عصبی LSTM است.

نحوه فراخوانی کتابخانه Keras و کلاس‌های مورد نیاز به صورت زیر است:

```
8- from keras.models import Sequential
9- from keras.layers import Dense
10- from keras.layers import LSTM
11- from keras.layers import Flatten
```

۷-۲- فراخوانی داده

کتابخانه پانداس که در بخش ۷-۱-۱ معرفی شد، برای وارد کردن داده‌های ورودی مورد استفاده قرار می‌گیرد. نحوه فراخوانی داده‌های بار ورودی به صورت زیر است:

```
12- TehranLoadReal=pd.read_csv('data/tehran/TehranLoadReal_repaired.csv',
    index_col=False)
13- TehranDemandResponse=pd.read_excel('data/tehran/TehranDemandResponse.xlsx',
    sheet_name='Sheet1', header=None, index_col=False)
14- TehranTotalLoad=pd.read_csv('data/tehran/TehranTotalLoad_repaired.csv',
    index_col=False)
```

[۱] Theano

```
15- TehranLoadReal.columns=['year','month','day', 'col1', 'col2', 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]
16- TehranDemandResponse.columns=['year','month','day', 'col1', 'col2', 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]
```

همانطور که مشاهده می‌گردد، داده‌های بار که با فرمت‌های csv و xlsx ذخیره شده‌اند و در پوشه data قرار دارند، به راحتی با استفاده از کتابخانه پانداس قابل فراخوانی هستند. index_col=False برای حالتی است که اولین رکورد در فایل CSV به عنوان هدر در نظر گرفته نشود. همچنین، در کد خط شماره ۱۵ و ۱۶، عنوان برای هر یک از ستون‌های داده‌ها تعریف شده است.

به طور مشابه، داده‌های هواشناسی و تقویمی به صورت زیر به کد فراخوانی می‌شوند:

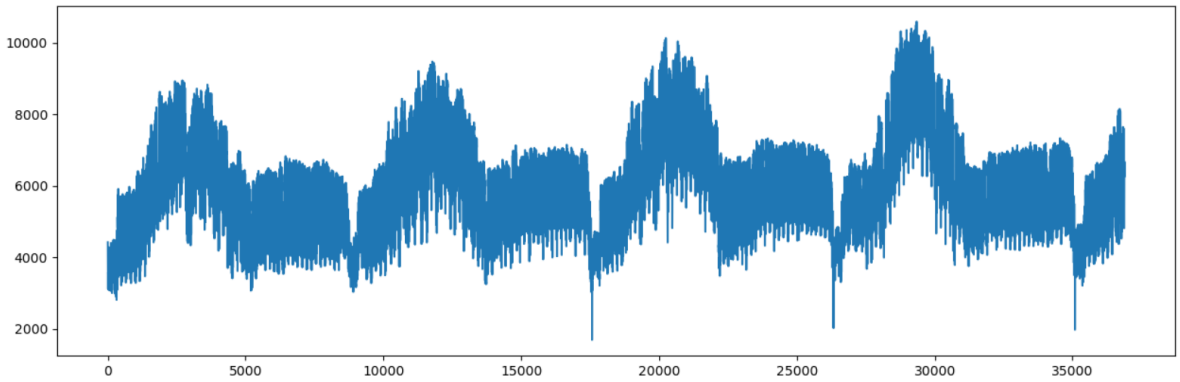
```
17- allWeatherData=pd.read_csv('data/allWeatherData_with_dayNum_weekNum_repaired.csv', index_col=False)
18- Holidaies_Ashora_Tasoaa=pd.read_excel('data/Holidaies_Ashora_Tasoaa_weekday.xlsx', sheet_name='calendar', index_col=False)
```

۷-۳- تعیین متغیرهای آموزش و تست

بعد از مرحله فراخوانی داده‌ها، نیاز هست که زمان آغاز و پایان داده‌های آموزش و تست برای هر دو متغیرهای ورودی و خروج تعیین شود، که به صورت زیر انجام می‌شود.

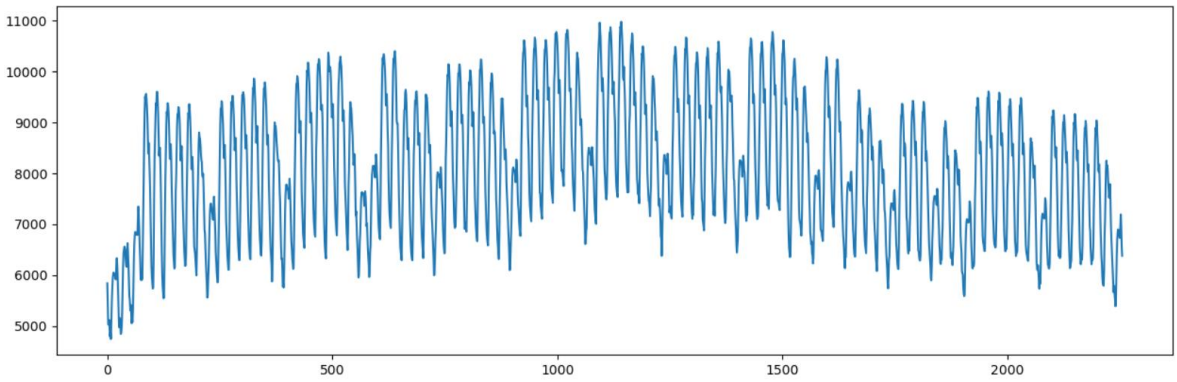
```
19- train_start_date = [1394, 1, 1]
20- train_end_date = [1398, 3, 14]
21- test_start_date = [1398, 3, 15]
22- test_end_date = [1398, 6, 15]
```

شکل ۷-۱ و ۷-۲ داده‌های بار کلی از یکم فروردین ۱۳۹۴ تا ۱۴ خرداد ۱۳۹۸ و داده‌های بار کلی از ۱۴ خرداد ۱۳۹۸ تا ۱۵ شهریور ۱۳۹۸ را به تصویر می‌کشد.



شکل ۷-۱: داده‌های بار کلی برای آموزش

DOI: 10.30503/nripress.2020.321



شکل ۷-۲: داده‌های بار کلی برای تست

۷-٤- پردازش موازی

بر اساس زمان‌های تعیین شده در مرحله قبل، متغیرها در آرایه‌های جداگانه‌ای قرار می‌گیرند که این کار با روش پردازش موازی که در بخش ۴-۱-۱ تعریف شد، انجام می‌شود. نحوه‌ی کدنویسی برای انجام پردازش موازی روی متغیرهای هواشناسی و تقویمی تست ورودی به صورت زیر است:

```
23- train_weather_arrs = Parallel(n_jobs=11)(delayed(varPreparing.cutData)(allWeatherData, train_start_date, train_end_date, col_name) for col_name in ['temperature', 'humidity' , 'dew Point', 'wind Speed', 'year', 'day in year', 'day in week', 'hour', 'icon'])
24- train_temperature_arr = train_weather_arrs[0]
25- train_humidity_arr = train_weather_arrs[1]
26- train_dewPoint_arr = train_weather_arrs[2]
27- train_windSpeed_arr = train_weather_arrs[3]
28- train_year_arr = train_weather_arrs[4]
29- train_day_in_year_arr = train_weather_arrs[5]
30- train_day_in_week_arr = train_weather_arrs[6]
31- train_hour_arr = train_weather_arrs[7]
32- train_icon_arr = train_weather_arrs[8]
```

همانطور که مشاهده می‌گردد، یک تابع به نام cutData در یک کلاس به نام varPreparing که در ماژول به نام modules.preparing_module قرار دارد، تعریف شده است. نحوه‌ی کدنویسی مربوط به تابع cutData در بخش ۷-۱۴ آورده شده است. متد delayed برای معرفی تابع تعریف شده به کلاس Parallel به کار می‌رود. متغیر col_name جزو متغیرهای نخ[۱] به شمار می‌رود که نیاز هست مقداردهی اولیه شود. همانطور که کد خط ۲۴ نشان می‌دهد، متغیر col_name از طریق آرایه ['temperature', 'humidity' , 'dew Point', 'wind Speed', 'year', 'day in year', 'day in week', 'hour', 'icon'] مقداردهی شده است. n_jobs به موتور می‌گوید که اجازه استفاده

[۱] thread

DOI: 10.30503/nripress.2020.321

از چه تعداد پردازنده را دارد. اگر مقدار این هایپرپارامتر برابر با ۱ باشد، می‌تواند تنها از یک پردازنده استفاده کند. در این پروژه مقدار n_jobs برابر با ۱۱ در نظر گرفته شده است. به عبارت دیگر، هر کدام از متغیرهای thread تعداد ۱۱ هسته را در حین اجرای کد استفاده خواهند کرد. به طور کلی، thread در محاسبات، یک فرآیند نمونه‌ای از یک برنامه‌ی کامپیوتری در حال اجرا است. همچنین، thread کوچکترین واحد پردازش است که می تواند در یک سیستم عامل انجام شود.

به طور مشابه، پردازش موازی روی متغیرهای هواشناسی و تقویمی آموزش به صورت زیر انجام می‌گیرد.

```
33- test_weather_arrs = Parallel(n_jobs=11)(delayed(varPreparing.cutData)(allWeatherData, test_start_date, test_end_date, col_name) for col_name in ['temperature', 'humidity' , 'dew Point', 'wind Speed', 'year', 'day in year', 'day in week', 'hour', 'icon'])
34- test_temperature_arr = test_weather_arrs[0]
35- test_humidity_arr = test_weather_arrs[1]
36- test_dewPoint_arr = test_weather_arrs[2]
37- test_windSpeed_arr = test_weather_arrs[3]
38- test_year_arr = test_weather_arrs[4]
39- test_day_in_year_arr = test_weather_arrs[5]
40- test_day_in_week_arr = test_weather_arrs[6]
41- test_hour_arr = test_weather_arrs[7]
42- test_icon_arr = test_weather_arrs[8]
```

نحوه‌ی کدنویسی برای انجام پردازش موازی روی متغیرهای بار تست و آموزش نیز مشابه بالا است با این تفاوت که تابع به نام cutDataColRange از کلاس varPreparing فراخوانی می‌شود. در داده‌های هواشناسی هر سطر شامل یک ساعت و چند متغیر متفاوت است. در حالیکه داده‌های مربوط به بار در هرسطر شامل یک روز کامل و فقط یک متغیر می‌باشد. بنابراین نیاز است که تابع دیگری به نام cutDataColRange تعریف شود. نحوه‌ی کدنویسی مربوط به تابع cutDataColRange در بخش ۱۴-۷ آورده شده است. نحوه‌ی نگارش کد برای انجام پردازش موازی روی متغیرهای بار آموزش به صورت زیر است:

```
43- train_loads_arrs=Parallel(n_jobs=11)(delayed(varPreparing.cutDataColRange)(allData, train_start_date, train_end_date, range(24)) for allData in [TehranLoadReal, TehranDemandResponse, TehranTotalLoad])
44- train_TehranLoadReal_arr = train_loads_arrs[0]
45- train_TehranDemandResponse_arr = train_loads_arrs[0]
46- train_TehranTotalLoad_arr = train_loads_arrs[0]
```

و به طور مشابه، برای متغیرهای بار تست خواهیم داشت:

```
47- test_loads_arrs=Parallel(n_jobs=11)(delayed(varPreparing. cutDataColRange)(allData, test_start_date, test_end_date, range(24)) for allData in [TehranLoadReal, TehranDemandResponse, TehranTotalLoad])
48- test _TehranLoadReal_arr = test _loads_arrs[0]
```

```
49- test _TehranDemandResponse_arr = test _loads_arrs[0]
50- test _TehranTotalLoad_arr = test _loads_arrs[0]
```

در مرحله پردازش موازی رو داده های تست و آموزش بار، متغیر thread از طریق آرایه [TehranLoadReal, TehranDemandResponse, TehranTotalLoad] مقداردهی شده است.

۷-۵- تبدیل داده‌ها به صورت ساعتی

برای تبدیل داده‌ها به صورت ساعتی، بایستی یک تابع به نام serialize از کلاس به نام varPreparing فراخوانی شود. از طریق این تابع، هر سطر شامل یک ساعت می‌شود. قابل توجه است که نحوه‌ی کدنویسی مربوط به تابع serialize در بخش ۷-۱۴ ارئه شده است.

از طریق کد زیر، داده‌های آموزش هواشناسی و تقویمی به صورت ساعتی تغییر داده شده‌اند.

```
51- train_serialize_vars = Parallel(n_jobs=11)(delayed(varPreparing.serialize)(var) for var in [train_temperature_arr, train_humidity_arr, train_dewPoint_arr, train_windSpeed_arr, train_year_arr, train_day_in_year_arr, train_day_in_week_arr, train_hour_arr, train_icon_arr])
52- train_temperature_ser = train_serialize_vars[0]
53- train_humidity_ser = train_serialize_vars[1]
54- train_dewPoint_ser = train_serialize_vars[2]
55- train_windSpeed_ser = train_serialize_vars[3]
56- train_year_ser = train_serialize_vars[4]
57- train_day_in_year_ser = train_serialize_vars[5]
58- train_day_in_week_ser = train_serialize_vars[6]
59- train_hour_ser = train_serialize_vars[7]
60- train_icon_ser = train_serialize_vars[8]
```

و به طور مشابه برای داده‌های تست آموزش هواشناسی و تقویمی داریم:

```
61- test_serialize_vars = Parallel(n_jobs=11)(delayed(varPreparing.serialize)(var) for var in [test_temperature_arr, test_humidity_arr, test_dewPoint_arr, test_windSpeed_arr, test_year_arr, test_day_in_year_arr, test_day_in_week_arr, test_hour_arr, test_icon_arr])
62- test_temperature_ser = test_serialize_vars[0]
63- test_humidity_ser = test_serialize_vars[1]
64- test_dewPoint_ser = test_serialize_vars[2]
65- test_windSpeed_ser = test_serialize_vars[3]
66- test_year_ser = test_serialize_vars[4]
67- train_day_in_year_ser = train_serialize_vars[5]
68- train_day_in_week_ser = train_serialize_vars[6]
69- train_hour_ser = train_serialize_vars[7]
70- train_icon_ser = train_serialize_vars[8]
```

DOI: 10.30503/nripress.2020.321

داده‌های آموزش بار نیز به صورت زیر به ساعتی تغییر داده شده‌اند.

```
train_serialize_vars = Parallel(n_jobs=11)(delayed(varPreparing.serialize)(var) for var in [train_TehranLoadReal_arr,train_TehranDemandResponse_arr,train_TehranTotalLoad_arr])
train_TehranLoadReal_ser = train_serialize_vars[0]
train_TehranDemandResponse_ser = train_serialize_vars[1]
train_TehranTotalLoad_ser = train_serialize_vars[2]
```

و به طور مشابه برای داده‌های تست بار داریم:

```
test_serialize_vars = Parallel(n_jobs=11)(delayed(varPreparing.serialize)(var) for var in [train_TehranLoadReal_arr,train_TehranDemandResponse_arr,train_TehranTotalLoad_arr])
test_TehranLoadReal_ser = test_serialize_vars[0]
test_TehranDemandResponse_ser = test_serialize_vars[1]
test_TehranTotalLoad_ser = test_serialize_vars[2]
```

۷-۶- جابه‌جایی داده‌ها

به طور کلی، شبکه عصبی طراحی شده پنجره زمانی شامل گام فعلی را دریافت می‌کند و براساس آن گام بعدی را پیش‌بینی می‌کند. بنابراین، داده‌های ورودی و خروجی نیز باید به همین صورت آماده شوند. ذکر این نکته مهم است که داده‌های آموزش و تست ورودی نباید شامل گام بعدی و یا خروجی باشد. زیرا موجب می‌گردد که در زمان تست نتایج بیش از حد معمول خوب باشد و در زمان پیش‌بینی نتایج مطلوب نباشند. با استفاده از تعدادی عملیات ریاضی که در کد مشخص است، داده ها شیفت داده شده‌اند.

```
train_TehranTotalLoad_ser_input_shift = train_TehranTotalLoad_ser[:-1]
train_TehranTotalLoad_ser_output_shift = train_TehranTotalLoad_ser[1:]
train_temperature_ser_shift = train_temperature_ser[1:]
train_humidity_ser_shift = train_humidity_ser[1:]
train_dewPoint_ser_shift = train_dewPoint_ser[1:]
train_windSpeed_ser_shift = train_windSpeed_ser[1:]
train_year_ser_shift = train_year_ser[1:]
train_day_in_year_ser_shift = train_day_in_year_ser[1:]
train_day_in_week_ser_shift = train_day_in_week_ser[1:]
train_hour_ser_shift = train_hour_ser[1:]
train_icon_ser_shift = train_icon_ser[1:]
test_TehranTotalLoad_ser_input_shift=np.hstack((train_TehranTotalLoad_ser[-1], test_TehranTotalLoad_ser[:-1]))
test_TehranTotalLoad_ser_output_shift = test_TehranTotalLoad_ser
test_temperature_ser_shift = test_temperature_ser
test_humidity_ser_shift = test_humidity_ser
```

DOI: 10.30503/nripress.2020.321

```
test_dewPoint_ser_shift = test_dewPoint_ser
test_windSpeed_ser_shift = test_windSpeed_ser
test_year_ser_shift = test_year_ser
test_day_in_year_ser_shift = test_day_in_year_ser
test_day_in_week_ser_shift = test_day_in_week_ser
test_hour_ser_shift = test_hour_ser
test_icon_ser_shift = test_icon_ser
```

در ادامه، تمام متغیرها را دریک آرایه قرار می‌دهیم. قابل ذکر است که از تابع hstack برای ترکیب کردن آرایه‌ها استفاده می‌شود. همچنین تابع len تعداد مولفه‌های یک لیست یا آرایه را بدست می‌آورد. همچنین، تابع reshape یک آرایه را بدون تغییر دادن محتوای آن به shape مناسب در می‌آورد. کد زیر داده‌های آماده شده (dataset) برای آموزش شبکه را بیان می‌کند.

```
dataset=np.hstack((train_year_ser_shift.reshape(len(train_year_ser_shift),1),
train_day_in_year_ser_shift.reshape(len(train_day_in_year_ser_shift),1),
train_day_in_week_ser_shift.reshape(len(train_day_in_week_ser_shift),1),
train_hour_ser_shift.reshape(len(train_hour_ser_shift),1),
train_icon_ser_shift.reshape(len(train_icon_ser_shift),1),
train_temperature_ser_shift.reshape(len(train_temperature_ser_shift),1),
train_TehranTotalLoad_ser_input_shift.reshape(len(train_TehranTotalLoad_ser_input_s
ift),1),
train_TehranTotalLoad_ser_output_shift.reshape(len(train_TehranTotalLoad_ser_outp
ut_shift),1)))
```

۷-۷- ورودی و خروجی برای آموزش

بعد از آماده‌سازی داده‌های آموزش، نیاز است با توجه به متغیر n_history_steps، تعدادی از رکوردهای dataset باهم به عنوان پنجره زمانی درنظر گرفته شوند. n_history_steps تعداد گامی که به عنوان ورودی به شبکه داده می‌شود را نشان می‌دهد. به عبارت دیگر، آرایه ورودی X و آرایه خروجی y باید به گونه‌ای تعیین شود که به ازای هر پنجره زمانی، یک داده خروجی یا یک گام وجود داشته باشد. به این ترتیب یک ارایه سه بعدی خواهیم داشت. نحوه کدنویسی برای این بخش، به صورت زیر انجام می‌گیرد.

```
n_history_steps = 48
X, y = varPreparing.split_sequences(dataset, n_history_steps)
```

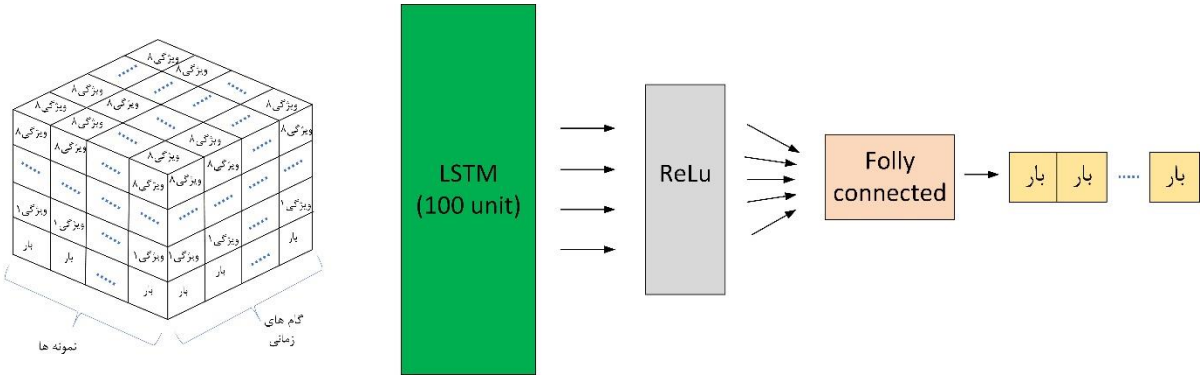
۸-۷- تعریف مدل شبکه عصبی

در این پروژه، مدل شبکه عصبی LSTM برای انجام پیش‌بینی در نظر گرفته شده است. نحوه کدنویسی برای تعریف مدل شبکه LSTM به صورت زیر است.

```
model = Sequential()
```

```
105- model.add(LSTM(100, activation='relu', input_shape=(n_history_steps, n_features)))
106- model.add(Dense(1))
107- model.compile(optimizer='adam', loss='mse')
```

همانطور که مشاهده می‌شود، تعداد نورون‌ها برابر ۱۰۰ در نظر گرفته شده است. نوع تابع فعال‌سازی relu است و نوع تابع بهینه‌سازی adam است. همچنین، نوع تابع جریمه در آموزش mse فرض شده است. input_shape نیز ساختار داده ورودی را نشان می‌دهد که شامل تعداد ویژگی‌ها (n_features) و تعداد گام‌های ورودی (n_history_steps) است. در این پژوه، تعداد لایه‌های شبکه LSTM برابر با ۹۶ در نظر گرفته شده است. ساختار شبکه پیشنهادی با ورودی‌های در نظر گرفته شده در شکل ۷-۳ نشان داده شده است. دما، تعطیلی، وضعیت آسمان، سال، روز در سال، روز در هفته، ساعت در روز، عاشورا-تاسوعا به عنوان ویژگی‌های ورودی به شبکه پیشنهادی در نظر گرفته شده‌اند.



شکل ۷-۳: ساختار کل شبکه عمیق بکار رفته برای پیش‌بینی بار

۷-۹- آموزش شبکه LSTM

آموزش شبکه طراحی شده با استفاده از دستور fit به صورت زیر انجام می‌گیرد.

```
108- model.fit(X, y, epochs=200, verbose=1)
```

۷-۱۰- آماده‌سازی مجموعه داده برای پیش‌بینی

در زمان پیش‌بینی نیز ورودی شبکه به طور هماهنگ با داده‌های آموزش تعیین گردد. برای این کار تعدادی از ورودی‌های آموزش به عنوان تعدادی از ورودهای مجموعه تست اضافه می گردد تا یک پنجره زمانی کامل تشکیل گردد. همانند داده‌های آموزش، جابه‌جایی داده‌ها باید در داده‌های تست نیز انجام گیرد. نحوه کد نویسی مرتبط با این بخش به صورت زیر است.

```
109- test_year_ser_shift_added=np.hstack((train_year_ser_shift[(len(train_year_ser_shift) - n_history_steps - 1):len(train_year_ser_shift)], test_year_ser_shift))
```

DOI: 10.30503/nripress.2020.321

```
test_day_in_year_ser_shift_added=np.hstack((train_day_in_year_ser_shift[(len(train_day_in_year_ser_shift)-n_history_steps-1):len(train_day_in_year_ser_shift)], test_day_in_year_ser_shift))
test_day_in_week_ser_shift_added=np.hstack((train_day_in_week_ser_shift[(len(train_day_in_week_ser_shift)-n_history_steps-1):len(train_day_in_week_ser_shift)], test_day_in_week_ser_shift))
test_hour_ser_shift_added=np.hstack((train_hour_ser_shift[(len(train_hour_ser_shift)-n_history_steps - 1):len(train_hour_ser_shift)], test_hour_ser_shift))
test_icon_ser_shift_added=np.hstack((train_icon_ser_shift[(len(train_icon_ser_shift) - n_history_steps - 1):len(train_icon_ser_shift)], test_icon_ser_shift))
test_temperature_ser_shift_added=np.hstack((train_temperature_ser_shift[(len(train_temperature_ser_shift)-n_history_steps-1):len(train_temperature_ser_shift)], test_temperature_ser_shift))
test_TehranTotalLoad_ser_input_added=np.hstack((train_TehranTotalLoad_ser_input_shift[(len(train_TehranTotalLoad_ser_input_shift)-n_history_steps-1):len(train_TehranTotalLoad_ser_input_shift)], test_TehranTotalLoad_ser_input_shift))
test_TehranTotalLoad_ser_output_added=np.hstack((train_TehranTotalLoad_ser_output_shift[(len(train_TehranTotalLoad_ser_output_shift)-n_history_steps-1):len(train_TehranTotalLoad_ser_output_shift)], test_TehranTotalLoad_ser_output_shift))
test_TehranTotalLoad_ser_input_added_0=test_TehranTotalLoad_ser_input_added
test_TehranTotalLoad_ser_input_added_0[(n_history_steps):] = 0
```

سپس مجموعه‌ی داده را که هر سطر آن حاوی یک رکورد ساده ورودی و خروجی است با استفاده از کد زیر تعریف می‌کنیم.

```
test_dataset=np.hstack((test_year_ser_shift_added.reshape(len(test_year_ser_shift_added),1),
test_day_in_year_ser_shift_added.reshape(len(test_day_in_year_ser_shift_added),1),
test_day_in_week_ser_shift_added.reshape(len(test_day_in_week_ser_shift_added),1),
test_hour_ser_shift_added.reshape(len(test_hour_ser_shift_added),1),
test_icon_ser_shift_added.reshape(len(test_icon_ser_shift_added),1),
test_temperature_ser_shift_added.reshape(len(test_temperature_ser_shift_added),1),
test_TehranTotalLoad_ser_input_added_0.reshape(len(test_TehranTotalLoad_ser_input_added_0),1),
test_TehranTotalLoad_ser_output_added.reshape(len(test_TehranTotalLoad_ser_output_added), 1)))
```

۷-۱۱- ورودی و خروجی تست

در این بخش مجموعه ورودی و خروجی تست را آماده می‌کنیم. باید توجه داشت که در زمان اجراء واقعی فقط مجموعه ورودی را خواهیم داشت که بر اساس آن پیش‌بینی انجام شده و بعدا خروجی را دریافت خواهیم کرد. با استفاده از کد زیر مجموعه ورودی و خروجی برای تست بدست می‌آید.

```
119- test_X, test_y = varPreparing.split_sequences(test_dataset, n_history_steps)
```

۷-۱۲- انجام پیش‌بینی

با استفاده از داده‌های تست که در بخش قبل آماده شد، اولین گام پیش‌بینی می‌شود. ولی در ادامه برای اینکه به حالت واقعی نزدیک باشد. مقدار بار پیش‌بینی شده برای هر ساعت را جایگزین بار واقعی کرده تا در پیش‌بینی ساعت بعدی استفاده گردد، یعنی دقیقا به همان صورت که در حالت واقعی اتفاق می‌افتد و بار واقعی وجود ندارد. به این ترتیب برای ۱۰ روز به همین صورت پیش‌بینی انجام می‌گیرد. در نهایت کل پیش‌بینی‌ها با داده‌های واقعی مقایسه می‌گردد. این کار با استفاده از کد زیر قابل انجام است.

```
120- prediction_days = 10
121- predicted_y = model.predict(test_X[0,:,:].reshape(1,n_history_steps,n_features), verbose=0) predicted_y_arr = predicted_y
122- test_X[1,(n_history_steps-1),(n_features-1)] = predicted_y
     for i in range(1,(prediction_days * 24)):
     predicted_y = model.predict(test_X[i,:,:].reshape(1,n_history_steps,n_features), verbose=0) predicted_y_arr = np.hstack((predicted_y_arr, predicted_y))
123- if i < n_history_steps:
     for j in range((n_history_steps - 1 - i),n_history_steps): test_X[(i + 1),j,(n_features-1)] = predicted_y_arr[0,(j - n_history_steps + 1 + i)] test_X[(i + 1),j,(n_features-1)] = predicted_y_arr[0,(j - n_history_steps + 1 + i)]
     else:
     for j in range(0,n_history_steps): test_X[(i + 1),j,(n_features-1)] = predicted_y_arr[0,(j - n_history_steps + 1 + i)] test_X[(i + 1),j,(n_features-1)] = predicted_y_arr[0,(j - n_history_steps + 1 + i)]
```

و در ادامه پیش‌بینی صورت گرفته شده را از طریق کد زیر به ساختار مناسب تبدیل می‌کنیم:

```
124- predicted_y_arr = predicted_y_arr.reshape(predicted_y_arr.shape[1])
```

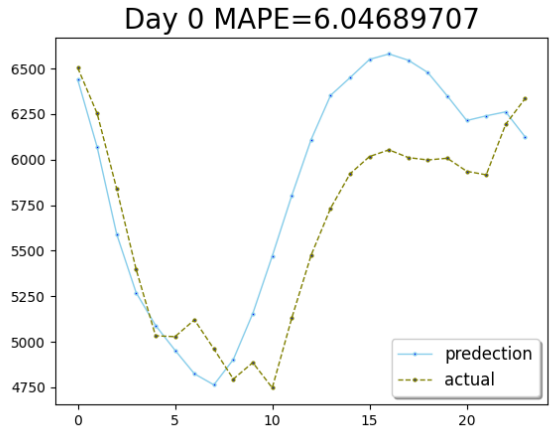
۷-۱۳- رسم نمودارهای خروجی

به منظور ارزیابی و انجام مقایسه داده‌های واقعی با داده‌های پیش‌بینی شده توسط شبکه عصبی طراحی شده، نتایج برای هر روز یا هر ۲۴ ساعت بدست می‌آیند. از طریق کد زیر می‌توان نمودارها را رسم کرد:

```
125- for i in range(0,prediction_days):
```

```
    predict_p = predicted_y_arr[(i * 24):((i * 24) + 24)]
    test_p = test_y[(i * 24):((i * 24) + 24)]
    mape = utilities.mape(test_p, predict_p)
    plt.figure(i)
    plt.plot(predict_p, marker='o', markerfacecolor='blue', markersize=2, color='skyblue',
    linewidth=1, label="predection")
    plt.plot(test_p, marker='o', markerfacecolor='blue', markersize=2, color='olive',
    linewidth=1, linestyle='dashed', label="actual")
    plt.title(r'Day '+ str(i) +' MAPE=' + np.array2string(mape) , fontsize=20)
    plt.draw()
    plt.pause(0.001)
    plt.show()
```

شکل ۷-۴ نتیجه پیش‌بینی بار مرتبط با روز اول را به تصویر می‌کشد.



شکل ۷-٤: نتایج پیش‌بینی بدست آمده برای روز اول

به طور کلی، همبستگی بین تغییرات مقادیر ورودی بار و ویژگی‌های هواشناسی و تقویمی خیلی مهم است چرا که شبکه تلاش می‌کند که همبستگی بین ورودی‌ها را یاد بگیرد. اما زمانی که به دلایلی مانند برنامه پاسخگویی بار، تغییرات جزئی در مقادیر بار ورودی اعمال شود، باعث می‌شود که همبستگی بین مقادیر بار و ویژگی‌ها از دست برود و شبکه نتواند به خوبی روند آموزش را طی کند که موجب می‌گردد نتایج نامطلوبی برای پیش‌بینی حاصل شود.

۷-۱٤- نحوه کد نویسی توابع استفاده شده

کد مربوط به تابع cutData از کلاس varPreparing به صورت زیر است:

```
def cutData(allData, start_date, end_date, col_name):
                allData_size = allData.shape
                first_falg = 1
                arr = [];
                # dataRangeWithBar = tqdm(range(0, allData_size[0]))
                for r in range(0, allData_size[0]):
                        if allData.loc[r,'hour'] == 23 and (allData.loc[r,'year'] > start_date[0] or
(allData.loc[r,'year'] == start_date[0] and allData.loc[r,'month'] > start_date[1]) or
```

```
(allData.loc[r,'year'] == start_date[0] and allData.loc[r,'month'] == start_date[1] and
allData.loc[r,'day'] >= start_date[2])) and (allData.loc[r,'year'] < end_date[0] or (allData.loc[r,'year']
== end_date[0] and allData.loc[r,'month'] < end_date[1]) or (allData.loc[r,'year'] == end_date[0]
and allData.loc[r,'month'] == end_date[1] and allData.loc[r,'day'] <= end_date[2])):
                                        if first_falg == 0:
                                                temp = allData.loc[r-23:r,col_name]
                                                arr = np.vstack((arr, temp))
                                        if first_falg == 1:
                                                arr = allData.loc[r-23:r,col_name]
                                                first_falg = 0
                                # dataRangeWithBar.set_description("Making array of " + col_name + " data
%s" % r)
                        return arr
```

کد مربوط به تابع cutDataColRange از کلاس varPreparing به صورت زیر است:

```
def cutDataColRange(allData, start_date, end_date, col_range):
                        allData_size = allData.shape
                        first_falg = 1
                        arr = [];
                        # dataRangeWithBar = tqdm(range(0, allData_size[0]))
                        for r in range(0, allData_size[0]):
                                if (allData.loc[r,'year'] > start_date[0] or (allData.loc[r,'year'] ==
start_date[0] and allData.loc[r,'month'] > start_date[1]) or (allData.loc[r,'year'] == start_date[0] and
allData.loc[r,'month'] == start_date[1] and allData.loc[r,'day'] >= start_date[2])) and
(allData.loc[r,'year'] < end_date[0] or (allData.loc[r,'year'] == end_date[0] and allData.loc[r,'month']
< end_date[1]) or (allData.loc[r,'year'] == end_date[0] and allData.loc[r,'month'] == end_date[1]
and allData.loc[r,'day'] <= end_date[2])):
                                        if first_falg == 0:
                                                arr = np.vstack((arr, allData.loc[r,col_range]))
                                        if first_falg == 1:
                                                arr = allData.loc[r,col_range]
                                                first_falg = 0
                                # dataRangeWithBar.set_description("Making array of Load data range %s"
% r)
                                                                        return arr
```

کد مربوط به تابع serialize از کلاس varPreparing به صورت زیر است:

```
def serialize(var):
                        rows_num = var.shape[0]
                        serialized_var = var[0 , :]
                        for i in range(1,rows_num):
                                serialized_var = np.hstack((serialized_var, var[i, :]))

                        return serialized_var
```

DOI: 10.30503/nripress.2020.321

فصل ۸- ساختار LSTM برای ورودی فقط بار (تفکیک بارها براساس نوع روز)

DOI: 10.30503/nripress.2020.321

در این فصل، فقط متغیر بار برای آموزش شبکه عصبی LSTM جهت پیش‌بینی بار استفاده شده است.

۸-۱- ساختار کلی

ساختار کلی پیش‌بینی با استفاده از داده‌های فقط بار برای آموزش شبکه LSTM به صورت شکل ۸-۱ می‌باشد. همانطور که نشان داده شده است، فقط متغیر بار با تعداد مشخصی از گام‌های قابل تنظیم از قبل به شبکه داده شده و خروجی شبکه نیز گام بعدی بار (یا همان ساعت بعدی) می‌باشد. این گام‌ها ساعت‌های مصرف بار هستند. این روند برای پیش‌بینی ۴۸ ساعت آینده که دو روز است (تعداد روزهای مورد پیش‌بینی برابر ۲ انتخاب شده) انجام شده است. روزهای متوالی با استفاده از حلقه نشان داده شده در شکل ۸-۲ پیش‌بینی می‌گردد.

۸-۲- نحوه عملکرد کد نوشته شده

متغیرهای زیر در کد قابل تنظیم هستند.

```
1- n_lockBack = 48
2- prediction_days = 2
3- shift_days = 1
4- BarChart_days = 31
```

متغیر موجود در کد بالا برای انجام تنظیمات اولیه در کد می‌باشد. توضیح مربوط به هریک به صورت جداگانه در زیر آمده است.

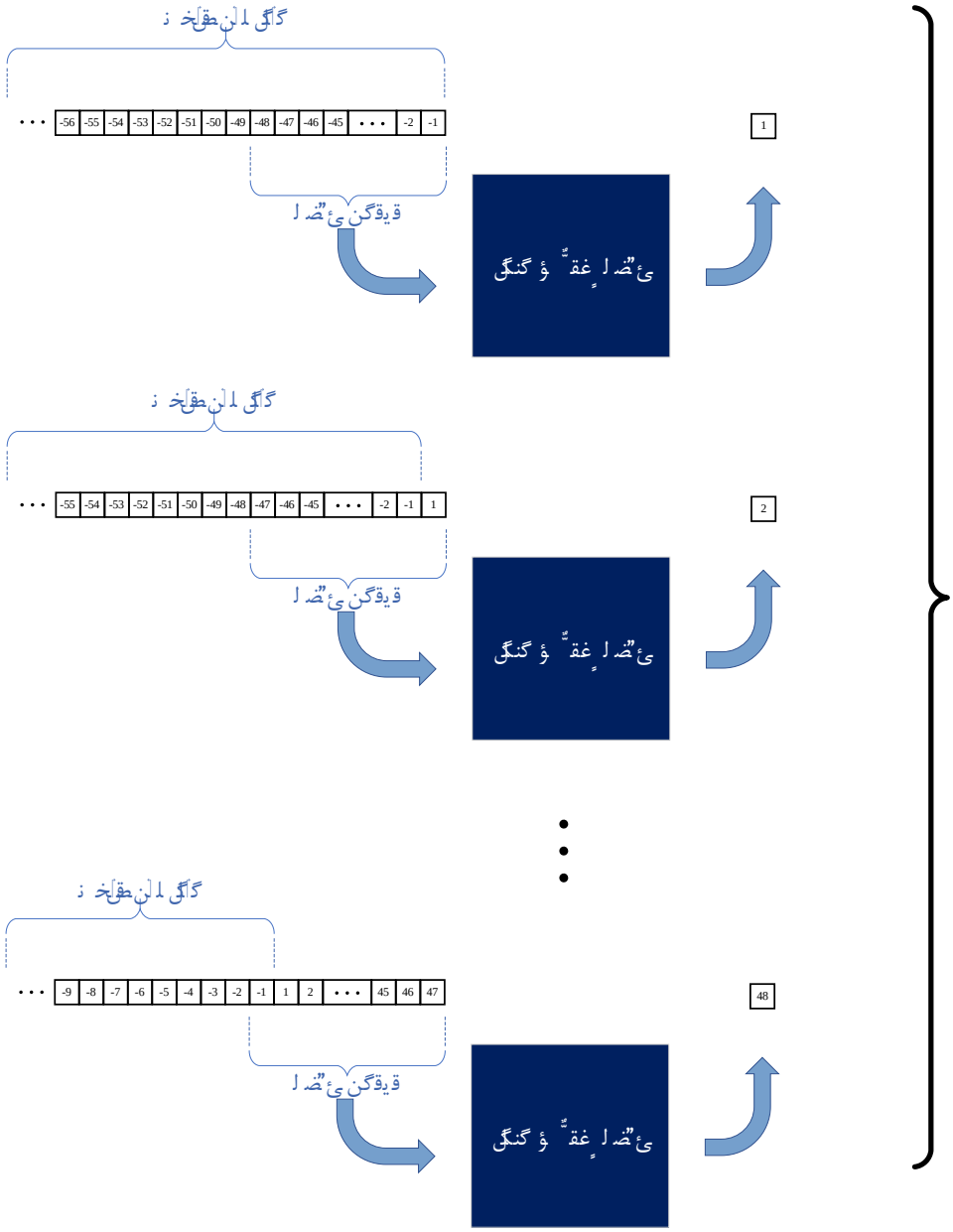
n_lockBack: تعیین تعداد گام‌های ورودی به شبکه

prediction_days: تعیین تعداد روزهای پیش‌بینی

shift_days: پس از هر بار پیش‌بینی بدون استفاده از دادههای واقعی بار چند روز داده واقعی به شبکه داده شود

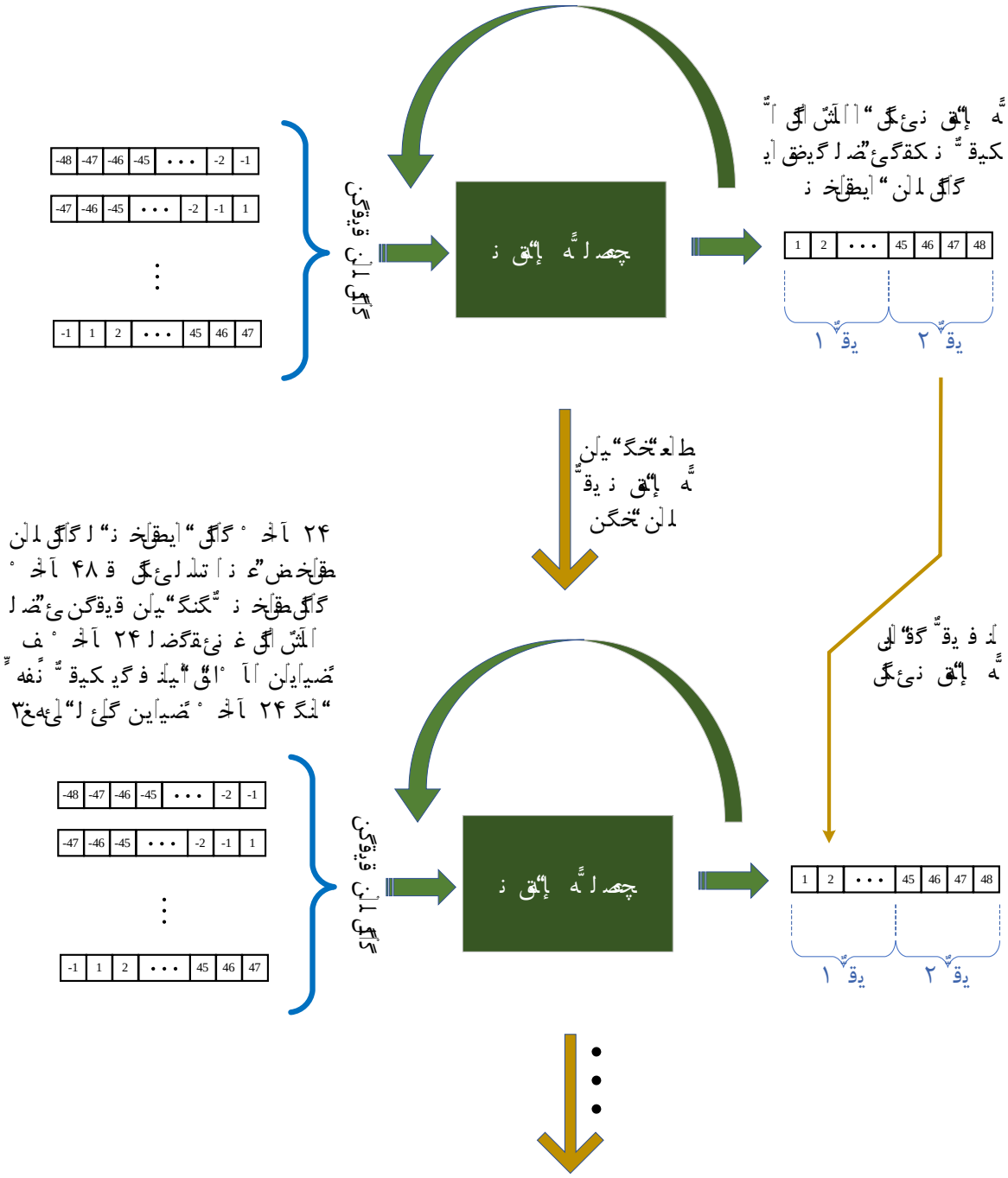
BarChart_days: نمودارهای حاوی میله‌ای شامل چند روز پیش‌بینی پشت سر هم باشند

DOI: 10.30503/nripress.2020.321



شکل ۸–۱: فرایند پیش‌بینی در یک حلقه برای ۴۸ ساعت آینده با استفاده از بار و واقعی و خروجی خود شبکه

DOI: 10.30503/nripress.2020.321



شکل ۸–۲: حلقه‌های متوالی پیش‌بینی برای پیش‌بینی روزهای متوالی

DOI: 10.30503/nripress.2020.321

۸-۳- شبه کد

تقریبا تمام بخش‌های کد بکار رفته در فصل دوم توضیح داده شده است. برای اینکه نگاه کلی به نحوه عمل کرد کد داشته باشیم بخش‌های مختلف آن را به صورت شبه‌کد[1] در ادامه قرار دادیم. الگوریتم مربوط به فرایند آموزش شبکه LSTM در شبه‌کد1 آمده است. خط ۱ مقادیر متغییرهایی که هر کدام ازآن‌ها در فصل دوم توضیح داده شده تنظیم می‌شوند. خط ۲ مقادیر داده‌های بار و تقویمی از فایل‌های مربوطه خوانده و بارگذاری می‌شود. خط ۳ بر اساس تاریخ های تعیین شده در متغیرهای خط ۱ بارها به دو دسته آموزش و تست تقسیم می‌گردد. خط ۴ براساس داده‌های بار جداشده برای آموزش، تعداد گام‌هایی که شبکه باید به عقب نگاه کند و اطلاعات تقویمی موجود (که برای مشخص کردن نوع روزها به کار می‌رود) متغییرهای آموزش اماده می‌گردد (توجه داشته باشید که برای هر دسته از روز ها نیاز به آموزش جداگانه است و درنتیجه نیاز است که در کدهای جدایی نوشته شوند که مهمترین تفاوت کدهای ایجاد شده برای روزهای مختلف در این قسمت است که نوع روز انتخاب می‌گردد). خط ۵ متغییرهای تست نیز همانند متغییرهای آموزش با همان ورودی‌ها ایجاد می‌گردد. خط ۶ رکوردهای موجود در متغیرهای آرایه‌ای مربوط به آموزش باید از حالت مرتب خارج شود و آمیخته شود فقط باید اینکار طوری انجام شود که خروجی مربوط به هر ورودی شبکه جابه‌جا نشود. خط ۷ مقادیر قبل از اعمالبه شبکه باید نرمال شود. خط ۸ همانطور که در فصل دوم گفته شد ورودی شبکه به صورت آریه سه بعدی با ابعاد مشخص است که یک بعد آن تعداد ویژگی‌ها می‌باشد که در اینجا فقط یک ویژگی وجود دارد. خط ۹ مدل براساس ساختار تعیین شده توسط پارامترهای ورودی آن ایجاد می‌گردد همچنین پارامترهای بهینه‌یابی در آموزش شبکه نیز تعیین می‌گردد. خط ۱۰ آموزش شبکه شروع تار رسیدن به یکی از حدود تعیین شده در پارامترهای بهینه‌یابی آن ادامه می‌یابد.

شبه‌کد1 : الگوریتم مروبط به آموزش شبکه LSTM

1	**Set** train_start_date, train_end_date, test_start_date, test_end_date, n_features, n_lockBack, prediction_days, shift_days, BarChart_days
2	LoadVar := ReadFile('FilePath'); CalenderVar := ReadFile('FilePath');
3	TrainLoadVar, TestLoadVar := split(LoadVar, train_start_date, train_end_date, test_start_date, test_end_date)
4	X_train, Y_train := MakeTrainSet(TrainLoadVar, n_lockBack, CalenderVar, TrainCalenderVar(LoadVar))
5	X_test, Y_test := MakeTestSet(TestLoadVar, n_lockBack, CalenderVar, TestCalenderVar(LoadVar))
6	X_train, Y_train := Shuffle(X_train, Y_train)
7	X_train, Y_train := Normalize(X_train, Y_train)
8	X_train := ReshapeForLSTM(X_train, n_features)
9	Model := DefineModel(LSTM(parameter), Optimization parameters)
10	Model := fit(Model, Training parameters)

الگوریتم مربوط به فرایند پیش‌بینی توسط شبکه LSTM در شبه‌کد2 آمده است. خط ۱ تعداد ساعت‌های هر بار پیشبینی تعیین می‌شود بهتر است ضریبی از ۲۴ باشد. خط ۲ پس از هر بار پیشبینی چه تعداد داده واقعی به انتهای دادهای تست اضافه شود که دوباره پیشبینی از انتهای داده های اضافه شده انجام گیرد. خط ۳ در مجموع پیشبینی‌ها برای چند روز انجام شود. خط ۴ داده‌های واقعی اضافه شده به انتهای داده‌های تست معادل چند روز است.

[1] Pseudocode

DOI: 10.30503/nripress.2020.321

خط ۵ ایجاد متغییر جدید برای داده‌های تست با توجه به اینکه در طول پیشبینی داده‌های بار پیشبینی شده خود شبکه به انتهای داده بار واقعی برای انجام پیشبینی های بعدی ساعتی اضافه می‌گردد. خط ۶ با توجه به اینکه شبکه با داده‌های نرمال آموزش دیده است داده‌های ورودی شبکه با همان مقیاس باید مجددا نرمال گردد. خط ۷ ابتدای حلقه پیشبینی برای کل تعداد روزها. خط ۸ ابتدای حلقه پیشبینی برای کل ساعات در هر بار پیشبینی. خط ۹ پیشبینی ساعت بعدی. خط ۱۰ خارج کردن مقدار پیشبینی شده از حالت نرمال و اضافه کردن آن به انتهای آرایه پیشبینی. خط ۱۱ اضافه کردن مقدار پیشبینی شده به انتهای دادهای واقعی برای انجام پیشبینی ساعت بعدی. خط ۱۳ اضافه کردن داده‌های واقعی به اندازه تعداد ساعات مشخص شده به انتهای داده‌های تست برای انجام پیشبینی روزهای بعد. خط ۱۴ قراردادن متغیر ورودی تست در یک متغییر جدید. خط ۱۵ نرمال کردن متغییر تست موقت که برای پیشبینی ساعات به کار می رود. خط ۱۶ اضافه کردن آرایه مربوط به ساعات پیشبینی شده به یک آرایه متشکل از این آرایه های پیشبینی.

شبه‌کد2 : الگوریتم مروبط به پیش بینی با شبکه LSTM

```
1  Predict_hours := 48
2  Shift_hours := 24
3  predction_days := 60
4  predction_shifts := floor (predction_days/(Shift_hours/24))
5  X_test_p = X_test
6  X_teat_normalize := Normalize(X_test_p)
7  For i = 1 to  predction_shifts {
8      For j = 1 to  Predict_hours{
9          Y_predict := predict(Model, X_test_p [(i * Predict_hours) +  j ])
10         Y_predicttions = append(Y_predicttions ,UnNormalize(Y_predict))
11         append Y_predict variables to X_test_p
12     }
13     append Y_test[i * (Shift_hours - 1) : i * Shift_hours] to  X_test
14     X_test_p = X_test
15     X_teat_normalize := Normalize(X_test_p)
16     Y_All_predicttions := append(Y_All_predicttions, Y_predicttions)
17 }
```

پس از آنجام پیش بینی نیاز به رسم نمودارها می‌باشد که اینکار با استفاده از الگوریتم نشان داده شده در شبه‌کد 3 انجام می‌گیرد. خط ۱ شروع حلقه برای کل روزهای پیشبینی شده که قرار است به تعداد روزهای پیشبینی نمودار برای هرروز ایجاد کند و مقدار MAPE هر روز را به صورت یک میله در نمودار میله‌ای ایجاد کند، خط۲ شروع حلقه برای کل ساعات پیشبینی شده که تمام آن برای هر روز یک نمودار می‌شود و مقدار MAPE به دست آمده از آن روز یک میله را در نمودار میله ای ایجاد می‌کند. خط ۳ رسم هر نمودار کامل مربوط به پیشبینی‌های یک روز در مقایسه با مقدار های واقعی همان روز در همان ساعات. خط ۴ محاسبه MAPE برای روز پیشبینی شده. خط ۵ اضافه کردن مقدار پیشبینی شده MAPE مربوط به همان روز به آرایه پیش‌بینی‌های MAPE. خط ۸ رسم نمودار میله‌ای مربوط به MAPE پیشبینی‌ها.

شبه‌کد3 : الگوریتم مروبط به نمایش نتایج پیش‌بینی با شبکه LSTM

```
1  For i = 1 to  predction_shifts {
2      For j = 1 to floor(Shift_hours/24){
3          Plot(Y_All_predicttions[i][(j-1) * 24: j * 24] , Y_test[i])
4          MAPE = MAPE_calculator(Y_All_predicttions[i][(j-1) * 24: j * 24], Y_test[i])
```

DOI: 10.30503/nripress.2020.321

```
5        Bar_array = append(Bar_array, MAPE)
6    }
7    }
8    Bar(Bar_array)
```

کد کامل مربوط به آموزش و پیش‌بینی شبکه LSTM و رسم نمودارهای مربوط به پیش‌بینی انجام شده توسط این شبکه در پیوست الف آمده است.

۸-٤- خروجی‌های مسئله

در این کد، نمودارهای خروجی به صورت فایل‌های تصویری در پوشه results_figures که در پوشه‌ی مسیر کد قرار دارد، ذخیره شده‌اند و حاوی اطلاعات MAPE و تاریخ روز مربوطه هستند. نمونه‌هایی از نمودارهای خروجی در ادامه آمده است. تنظیمات و ساختار شبکه به صورت زیر است.

مشخصات شبکه:

- تعداد ویژگی‌های ورودی : ۱ (فقط داده بار)
- تعداد گام‌های زمانی : ۴۸ گام
- تعداد خروجی‌ها: ۱ (فقط بار)
- تعداد لایه‌های LSTM : ۹۶

تنضیمات آموزش شبکه:

- تعداد حداکثر دوره‌ها (epochs) : ۸۰
- تابع optimizer: adam (تغییر نرخ آموزش در هر دوره با مقادیر پیشفرض)
- تابع loss : mse[1]

۸-٤-۱- نتایج اجراء کد

به منظور دستیابی به بهترین نتایج، دسته‌بندی‌های زیر برای روزهای مشابه بدست آمده است. روزهای ورودی به دو دسته زیر تقسیم شدند.

دسته اول: روزهای غیر از پنج‌شنبه، جمعه و تعطیلات رسمی.

دسته دوم: روزهای عادی شامل شنبه، یکشنبه، دوشنبه، سه شنبه و چهارشنبه که تعطیل نیستند.

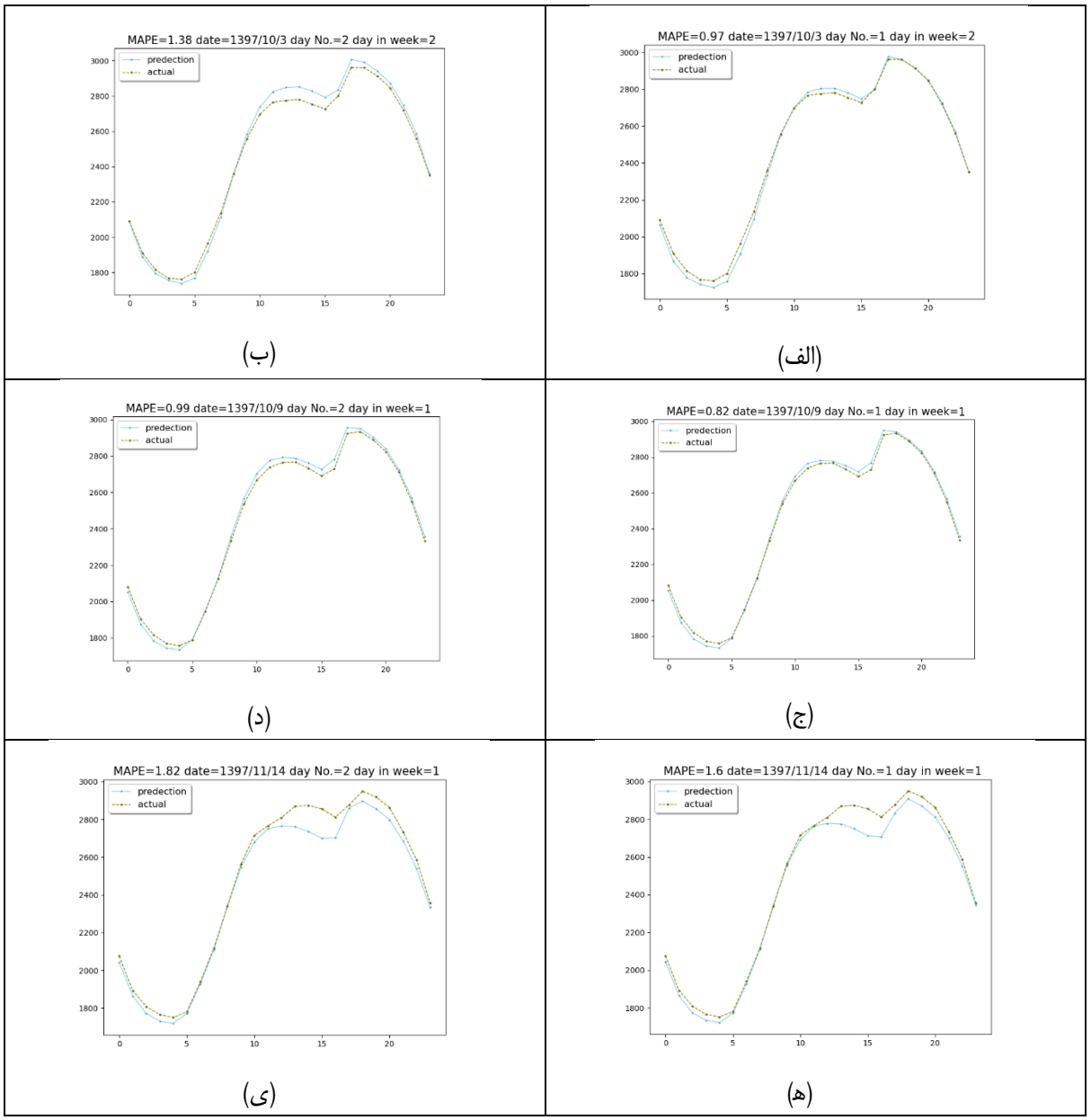
توجه کنید که پش‌بینی در هر روز دو بار انجام شده است، یکبار زمانی که روز مورد پیش‌بینی ابتدای ۴۸ ساعت است و یک بار زمانی که انتهای آن است. بنابراین روزهایی که در ابتدای ۴۸ ساعت قراردارند بدون فاصله از داده‌های واقعی هستند و روزهایی که در انتهای ۴۸ ساعت قراردارند به اندازه ۲۴ ساعت از داده‌های واقعی فاصله دارند. پس در پیش‌بینی آن‌ها از خروجی خود شبکه نیز به مقدار بسیار بیشتری استفاده شده است. بنابراین دو نوع پیش‌بینی زیر نیز وجود دارند که در ترکیب با دسته‌های بالا ۴ حالت ایجاد می‌شود.

[1] Mean Squared Error

DOI: 10.30503/nripress.2020.321

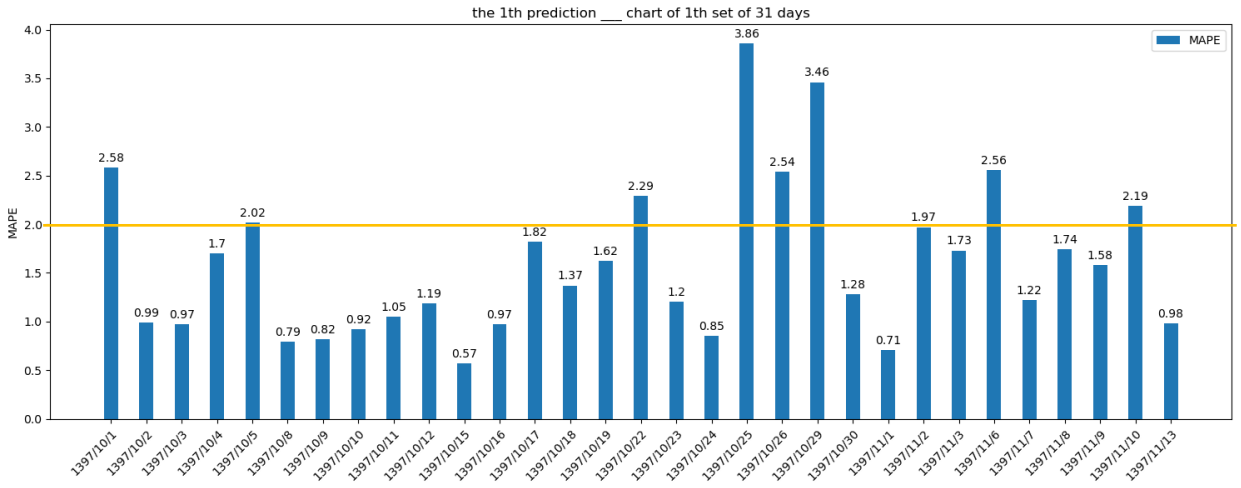
- ۲۴ ساعت اول پیش‌بینی
- ۲۴ ساعت دوم پیش‌بینی

نتایج به دست آمده برای تعدادی از روزهای دسته اول که در پیش‌بینی دسته‌ی ۲۴ ساعت اول و دسته ی ۲۴ ساعت دوم برای همان روز قرار دارد، به صورت نشان داده شده در شکل ۸–۳ می‌باشد. در این تصاویر مقدار روز در هفته از شنبه تا جمعه بین ۰ تا ۶ شماره گذاری شده است. همچنین شماره روز (که با .Day No در بالای نمودار درج شده) نشان‌دهنده روز چندم از تعداد کل پیش‌بینی‌های انجام شده بدون استفاده از داده‌های واقعی جدید بار است که با اضافه کردن خروجی خود شبکه پیش‌بینی شده است. مقدار MAPE و تاریخ مربوطه نیز در بالای هر نمودار آمده است. مقدار MAPE مربوط به کل روزهای دسته اول ۲۴ ساعت اول پیش‌بینی در نمودارهای شکل ۸-۴ و شکل ۸-۵ نشان داده شده است. همینطور مقدار MAPE مربوط به کل روزهای دسته اول و ۲۴ ساعت دوم پیش‌بینی در نمودارهای شکل ۸-۶ و شکل ۸-۷ نشان داده شده است.

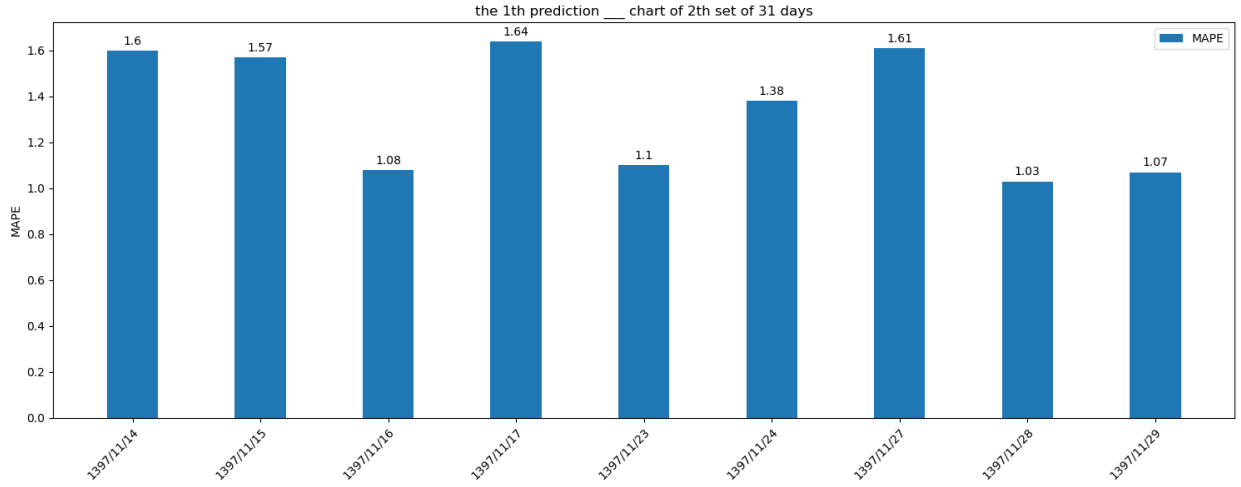


DOI: 10.30503/nripress.2020.321

شکل ۸–۳: پیش‌بینی تعدادی از روزهای دسته اول در ۲۴ ساعت اول و دوم پیش‌بینی؛ تصاویر (الف)، (ج) و (ه) مربوط به ۲۴ ساعت اول و تصاویر (ب)، (د) و (ی) مربوط به ۲۴ ساعت دوم



شکل ۸–۴: مقدار MAPE مربوط به ۳۱ روز ابتدایی بازه پیش‌بینی از نوع دسته اول در ۲۴ ساعت اول پیش‌بینی

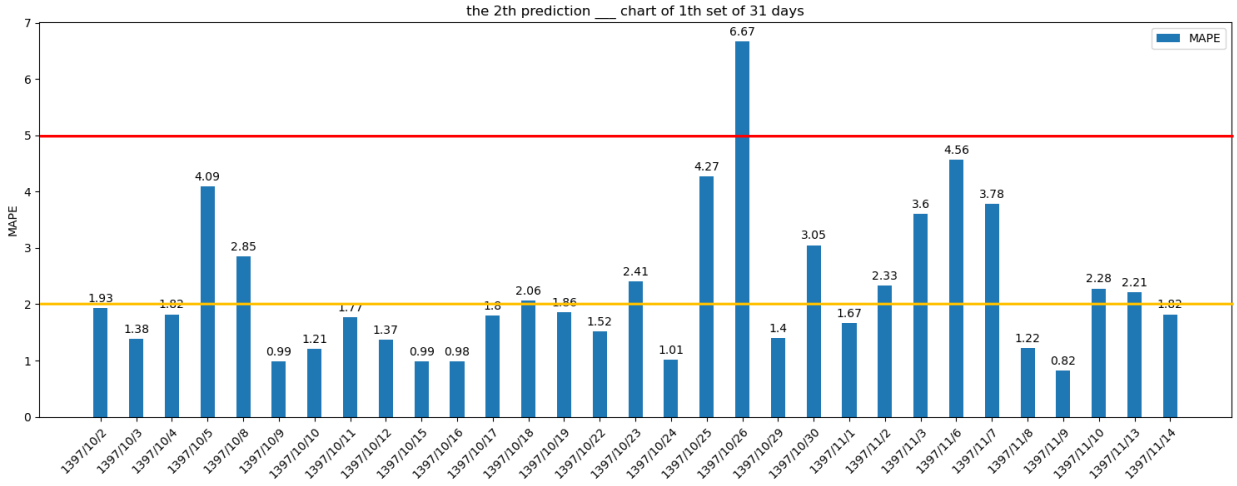


شکل ۸–۵: مقدار MAPE مربوط به ۹ روز انتهایی بازه پیش‌بینی از نوع دسته اول در ۲۴ ساعت اول پیش‌بینی

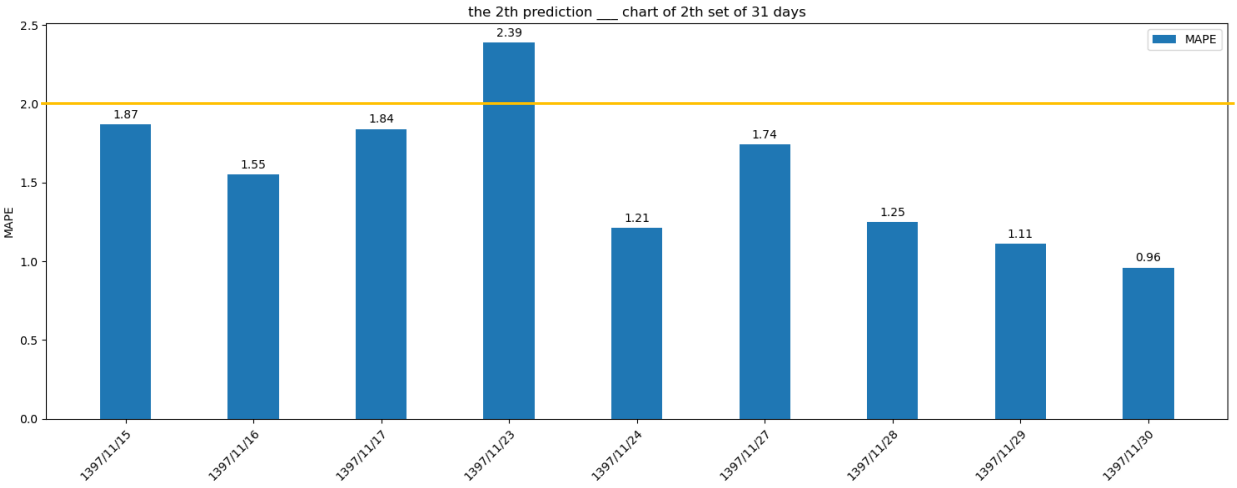
توجه داشته باشد که در نمودارهای موجود در شکل ۸–۴، شکل ۸–۵، شکل ۸–۶ و شکل ۸–۷ تمام روزهای بازه پیش‌بینی (۱۳۹۷/۱۰/۱ تا ۱۳۹۷/۱۱/۳۰) موجود نیست فقط روزهای از نوع دسته اول موجود هستند.

همانطور که با مقایسه شکل ۸–۴ و شکل ۸–۶ مشاهده می‌شود، با افزایش فاصله از داده‌های واقعی مقدار MAPE افزایش می‌یابد. همچنین، تعداد بیشتری بالای ۲ درصد قرار می‌گیرد که این موضوع با مقایسه نمودارهای شکل ۸–۵ و شکل ۸–۷ قابل استنباط است. از طرف دیگر، همانطور که با مقایسه نمودارهای موجود در شکل ۸–۴ و شکل ۸–۵ می‌توان فهمید، تعداد روزهای با MAPE کمتر از ۲ درصد در روزهای ابتدایی باز پیش‌بینی (۱۳۹۷/۱۰/۱ تا ۱۳۹۷/۱۱/۳۰) بیشتر از روزهای انتهایی است.

DOI: 10.30503/nripress.2020.321

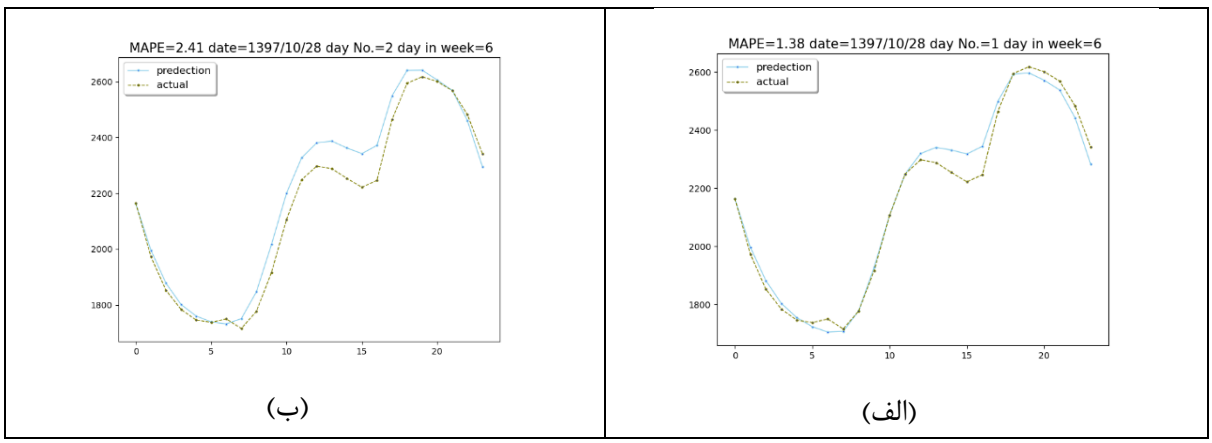


شکل ۸–۶: مقدار MAPE مربوط به ۳۱ روز ابتدایی بازه پیش‌بینی از نوع دسته اول در ۲۴ ساعت دوم پیش‌بینی



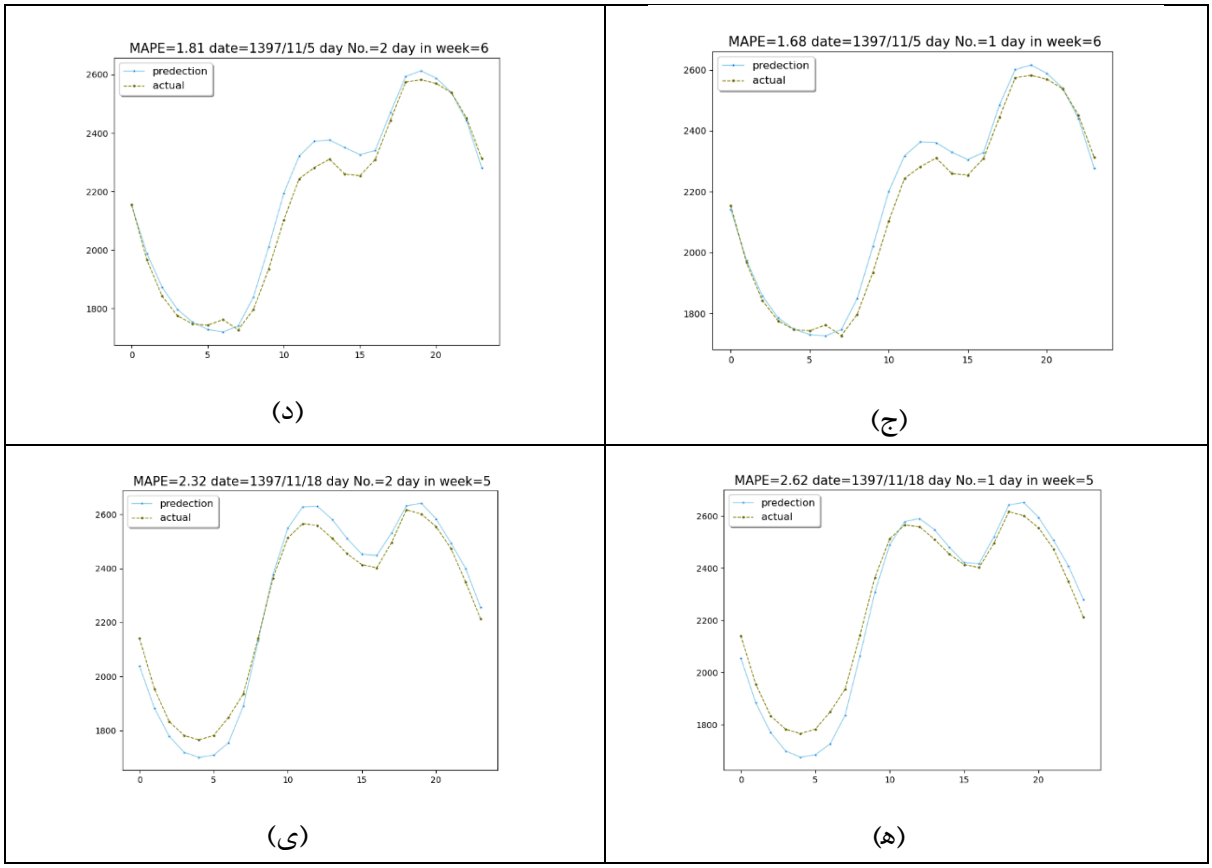
شکل ۸–۷: مقدار MAPE مربوط به ۹ روز انتهایی بازه پیش‌بینی از نوع دسته اول در ۲۴ ساعت دوم پیش‌بینی

در شکل ۸–۸ تعدادی از روزهای دسته دوم در ۲۴ ساعت اول و دوم پیش‌بینی برای هر روز مشخص آمده است. نمودارهای موجود در شکل ۸–۹ و شکل ۸–۱۰ مقدار MAPE دسته دوم از روزها در بازه پیش‌بینی (۱۳۹۷/۱۰/۱ تا ۱۳۹۷/۱۱/۳۰) را به تصویر می‌کشند.

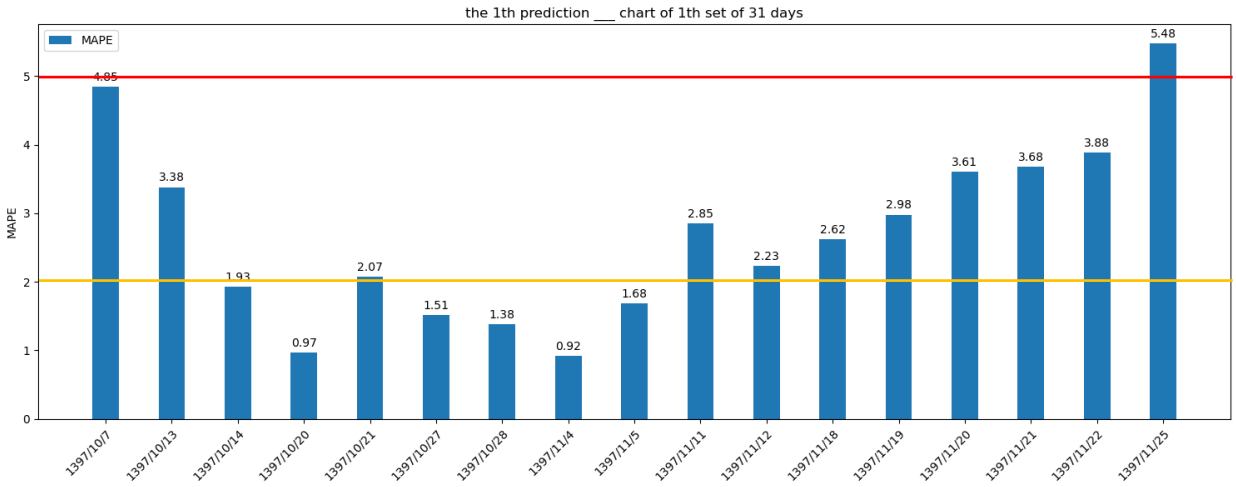


(ب) (الف)

DOI: 10.30503/nripress.2020.321

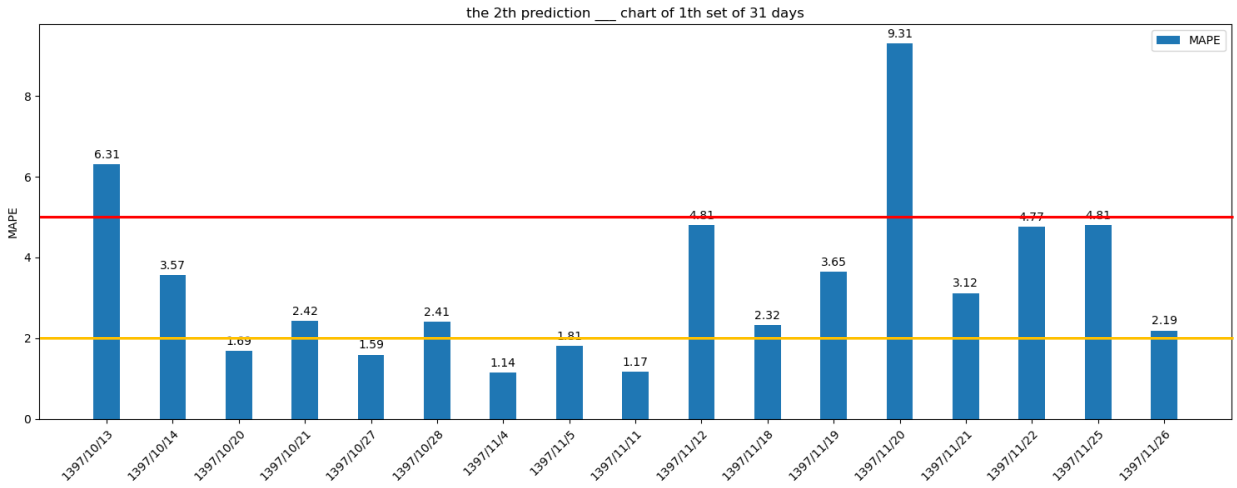


شکل ۸–۸: پیش‌بینی تعدادی از روزهای دسته دوم در ۲۴ ساعت اول و دوم پیش‌بینی؛ تصاویر (الف)، (ج) و (ه) مربوط به ۲۴ ساعت اول و تصاویر (ب)، (د) و (ی) مربوط به ۲۴ ساعت دوم



شکل ۸–۹: مقدار MAPE مربوط به ۳۱ روز ابتدایی بازه پیش‌بینی از نوع دسته دوم در ۲۴ ساعت اول پیش‌بینی

DOI: 10.30503/nripress.2020.321



شکل ۸–۱۰: مقدار MAPE مربوط به ۳۱ روز انتهایی بازه پیش‌بینی از نوع دسته دوم در ۲٤ ساعت دوم پیش‌بینی

همانطور که در شکل ۸–۹ و شکل ۸–۱۰ مشخص است همانند دسته اول داده‌ها، با افزایش فاصله از بارهای واقعی دقت پیش‌بینی کاهش می‌یابد.

DOI: 10.30503/nripress.2020.321

فصل ۹- ترکیب دو LSTM و MLP برای ورودی بار، ویژگی‌های آب و هوایی و ویژگی‌های تقویمی

DOI: 10.30503/nripress.2020.321

در این فصل از ترکیب دو شبکه LSTM و MLP برای پیش‌بینی بار استفاده شده است. ورودی‌های در نظر گرفته شده برای این شبکه ترکیبی جهت آموزش، متغیر بار، ویژگی‌های آب و هوایی و تقویمی هستند.

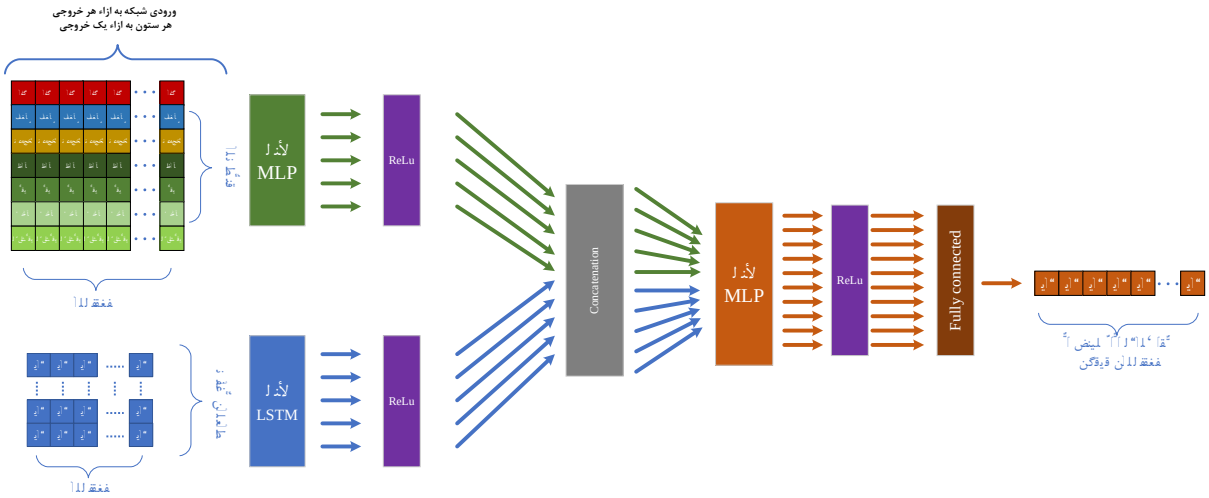
۹-۱- ساختار کلی

شبکه دارای ساختاری به صورت نشان داده شده در شکل ۹–۱ می‌باشد. یک بخش LSTM است که برای استخراج ویژگی بارهای ورودی به کار می‌رود که دارای گام‌های زمانی نیز می‌باشد و یک بخش شبکه تمام متصل MLP که اثر ویژگی‌های آب و هوایی را درکنار ویژگی‌های تقویمی روی بار ساعتی که قرار است پیش‌بینی شود، درنظر می‌گیرد. درنهایت یک لایه تک نرونی پیش‌خورد نیز در انتها قرار دارد که با استفاده از ویژگی‌های استخراج شده از هر دو نوع داده نتیجه‌گیری کرده و بار را پیش‌بینی می‌کند.

در ورودی بخش LSTM متغیر بار با تعداد مشخصی از گام‌های قابل تنظیم از قبل به شبکه داده شده و خروجی این لایه ویژگی‌های بار می‌باشد. همانند فصل سوم که فقط بار به شبکه داده می‌شد، این گام‌ها بار ساعتی هستند. برای پیش‌بینی هر دو روز یعنی ۴۸ ساعت (تعداد روزهای پیش‌بینی قابل تنظیم است) از خروجی خود شبکه در هر ساعت برای ورودی آن استفاده می‌شود.

پس از آموزش، برای هر بار پیش‌بینی باید همان تعداد مشخص شده از گام‌های بار که قبلا تعیین شده بود به شبکه داده شود تا شبکه گام بعدی را پیش‌بینی کند. سپس گام پیش‌بینی شده به انتهای آرایه گام‌های ورودی اضافه شده و دوباره آرایه به بخش LSTM شبکه داده می‌شود تا پیش‌بینی انجام شود، این روند برای پیش‌بینی ۴۸ ساعت آینده که دو روز است (تعداد روزهای مورد پیش‌بینی برابر ۲ انتخاب شده) انجام شده و سپس در انتهای آرایه ورودی ۲۴ گام (تعداد روز های شیفت یک روز تعیین شده) از بار واقعی قرار داده می‌شود. شبکه دوباره بار همان روال قبل ۴۸ ساعت بعدی را پیش‌بینی می‌کند. ورود داده به شبکه در بخش LSTM به صورت شماتیک همانند شکل ۸-۱ در فصل سوم می‌باشد. با استفاده از این حلقه که مربوط به بخش LSTM شبکه است، روزهای متوالی به همان صورت نشان داده شده در شکل ۸-۲ فصل سوم پیش‌بینی می‌گردد. در بخش MLP فقط کافی است برای هر ساعتی که قرار است پیش‌بینی انجام شود، ویژگی‌های تقویمی و آب و پیش‌بینی آب و هوایی را به صورت ساعتی به شبکه بدهیم. شکل ۹–۱ ساختار کل شبکه عمیق بکار رفته برای پیش‌بینی که ترکیبی از LSTM و MLP است را به تصویر می‌کشد.

DOI: 10.30503/nripress.2020.321



شکل ۹-۱: ساختار کل شبکه عمیق بکار رفته برای پیش‌بینی که ترکیبی از LSTM و MLP است

۹-۲- نحوه عملکرد کد نوشته شده

مهمترین متغیرهای قابل تنظیم در کد در زیر آمده است.

متغیرهای زیر در کد قابل تنظیم هستند.

1- n_lockBack = 48
2- prediction_days = 2
3- shift_days = 1
4- BarChart_days = 31
5- n_features = 8

متغیر موجود در کد بالا برای انجام تنظیمات اولیه در کد می‌باشد. توضیح مربوط به هریک به صورت جداگانه در زیر آمده است.

n_lockBack: تعیین تعداد گام‌های ورودی به شبکه

prediction_days: تعیین تعداد روزهای پیش‌بینی

shift_days: پس از هر بار پیش‌بینی بدون استفاده از داده های واقعی بار چند روز داده واقعی به شبکه داده شود

BarChart_days: نمودارهای حاوی میله‌ای شامل چند روز پیش‌بینی پشت سر هم باشند

n_features: تعداد ویژگی‌های ورودی به شبکه غیر از بار

DOI: 10.30503/nripress.2020.321

۹-۳- شبه کد

تقریبا تمام بخش‌های کد بکار رفته در شبکه ترکیبی LSTM و MLP نیز همانند آنچه می‌باشد که در فصل دوم توضیح داده شده است. برای اینکه نگاهی کلی به نحوه عمل کرد کد درفرایند آموزش شبکه داشته باشیم شبه‌کد[1] مربوطه در ادامه آمده است. شبه‌کد4 به طور کلی الگوریتم مربوط به فرایند آموزش شبکه ترکیبی LSTM و MLP را نشان می‌دهد.

شبه‌کد4 : الگوریتم مروبط به آموزش شبکه ترکیبی LSTM و MLP	
1	**Set** train_start_date, train_end_date, test_start_date, test_end_date, n_features, n_lockBack, prediction_days, shift_days, BarChart_days
2	LoadVar := ReadFile('FilePath')
3	WeatherVar := ReadFile('FilePath')
4	CalenderVar := ReadFile('FilePath')
5	TrainLoadVar, TestLoadVar := split(LoadVar, train_start_date, train_end_date, test_start_date, test_end_date)
6	TrainWeatherVar, TestWeatherVar := split(WeatherVar, train_start_date, train_end_date, test_start_date, test_end_date)
7	TrainCalenderVar, TestCalenderVar := split(CalenderVar, train_start_date, train_end_date, test_start_date, test_end_date)
8	X_train, Y_train := MakeTrainSet(TrainLoadVar, TrainWeatherVar, n_lockBack, TrainCalenderVar(WeatherVar))
9	X_test, Y_test := MakeTestSet(TestLoadVar, TestWeatherVar, n_lockBack, TestCalenderVar(WeatherVar))
10	X_train, Y_train := Shuffle(X_train, Y_train)
11	X_train, Y_train := Normalize(X_train, Y_train)
12	X_train := ReshapeForLSTM(X_train)
13	Model := DefineModel(LSTM(parameter), Optimization parameters)
14	Model := fit(Model, Training parameters)

در شبه‌کد4 خط ۱ مقادیر متغییرهایی که هر کدام ازآن‌ها در فصل دوم توضیح داده شده تنظیم می‌شوند. خط ۲ مقادیر داده‌های بار از فایل مربوطه خوانده و بارگذاری می‌شود. خط ۳ مقادیر داده‌های هواشناسی از فایل مربوطه خوانده و بارگذاری می‌شود. خط ۴ مقادیر داده‌های تقویمی از فایل مربوطه خوانده و بارگذاری می‌شود. خط ۵ ، ۶ و ۷ بر اساس تاریخ های تعیین شده در متغیرهای خط ۱ به ترتیب بارها، داده‌های هواشناسی و داده‌های تقویمی به دو دسته آموزش و تست تقسیم می‌گردد. خط ۸ براساس داده‌های بار جداشده برای آموزش، تعداد گام‌هایی که شبکه باید به عقب نگاه کند و اطلاعات تقویمی موجود (که برای مشخص کردن نوع روزها به کار می‌رود) متغییرهای آموزش آماده می‌گردد (توجه داشته باشید که برای هر دسته از روز ها نیاز به آموزش جداگانه است و درنتیجه نیاز است که در کدهای جدایی نوشته شوند که مهمترین تفاوت کدهای ایجاد شده برای روزهای مختلف در این قسمت است که نوع روز انتخاب می‌گردد). خط ۹ متغییرهای تست نیز همانند متغییرهای آموزش با همان ورودی‌ها ایجاد می‌گردد. خط ۱۰ رکوردهای موجود در متغیرهای آرایه‌ای مربوط به آموزش باید از حالت مرتب خارج شود و آمیخته شود فقط باید اینکار طوری انجام شود که خروجی مربوط به هر ورودی شبکه جابه‌جا نشود. خط ۱۱ مقادیر قبل از اعمالبه شبکه باید نرمال شود. خط ۱۲ همانطور که در فصل دوم گفته شد ورودی شبکه به صورت آریه سه بعدی با ابعاد مشخص است که یک بعد آن تعداد ویژگی‌ها می‌باشد که در اینجا فقط یک ویژگی وجود دارد. خط ۱۳ مدل براساس ساختار تعیین شده توسط پارامترهای ورودی آن ایجاد می‌گردد همچنین پارامترهای بهینه‌یابی در آموزش شبکه نیز تعیین

[1] Pseudocode

DOI: 10.30503/nripress.2020.321

می‌گردد. خط ۱۴ آموزش شبکه شروع تار رسیدن به یکی از حدود تعیین شده در پارامترهای بهینه‌یابی آن ادامه می‌یابد.

الگوریتم مربوط به فرایند پیش‌بینی توسط شبکه ترکیبی LSTM و MLP در شبه‌کد5 آمده است. خط ۱ تعداد ساعت‌های هر بار پیشبینی تعیین می‌شود بهتر است ضریبی از ۲۴ باشد. خط ۲ پس از هر بار پیشبینی چه تعداد داده واقعی به انتهای دادهای تست اضافه شود که دوباره پیشبینی از انتهای داده های اضافه شده انجام گیرد. خط ۳ در مجموع پیشبینی‌ها برای چند روز انجام شود. خط ۴ داده‌های واقعی اضافه شده به انتهای داده‌های تست معادل چند روز است. خط ۵ ایجاد متغییر جدید برای داده‌های تست با توجه به اینکه در طول پیشبینی داده‌های بار پیشبینی شده خود شبکه به انتهای داده بار واقعی برای انجام پیشبینی های بعدی ساعتی اضافه می‌گردد. خط ۶ با توجه به اینکه شبکه با داده‌های نرمال آموزش دیده است داده‌های ورودی شبکه با همان مقیاس باید مجددا نرمال گردد. خط ۷ ابتدای حلقه پیشبینی برای کل تعداد روزها. خط ۸ ابتدای حلقه پیشبینی برای کل ساعات در هر بار پیشبینی. خط ۹ پیشبینی ساعت بعدی. خط ۱۰ خارج کردن مقدار پیشبینی شده از حالت نرمال و اضافه کردن آن به انتهای آرایه پیشبینی. خط ۱۱ اضافه کردن مقدار پیشبینی شده به انتهای دادهای واقعی برای انجام پیشبینی ساعت بعدی. خط ۱۳ اضافه کردن داده‌های واقعی به اندازه تعداد ساعات مشخص شده به انتهای داده‌های تست برای انجام پیشبینی روزهای بعد. خط ۱۴ قراردادن متغیر ورودی تست در یک متغییر جدید. خط ۱۵ نرمال کردن متغییر تست موقت که برای پیشبینی ساعات به کار می رود. خط ۱۶ اضافه کردن آرایه مربوط به ساعات پیشبینی شده به یک آرایه متشکل از این آرایه های پیشبینی.

شبه‌کد5 : الگوریتم مروبط به پیش بینی با شبکه ترکیبی LSTM و MLP

```
1   Predict_hours := 48
2   Shift_hours := 24
3   predction_days := 60
4   predction_shifts := floor (predction_days/(Shift_hours/24))
5   X_test_p = X_test
6   X_teat_normalize := Normalize(X_test_p)
7   For i = 1 to  predction_shifts {
8       For j = 1 to  Predict_hours{
9           Y_predict := predict(Model, X_test_p [(i * Predict_hours) +  j ])
10          Y_predicttions = append(Y_predicttions ,UnNormalize(Y_predict))
11          append Y_predict and other related weather and calender variables to X_test_p
12      }
13      append Y_test[i * (Shift_hours - 1) : i * Shift_hours] to  X_test
14      X_test_p = X_test
15      X_teat_normalize := Normalize(X_test_p)
16      Y_All_predicttions := append(Y_All_predicttions, Y_predicttions)
17  }
```

پس از آنجام پیش بینی نیاز به رسم نمودارها می‌باشد که اینکار با استفاده از الگوریتم نشان داده شده در **Error! Reference source not found.**انجام می‌گیرد که کاملا مشابه شبه‌کد مربوط به رسم نمودار در ش بکه LSTM است. خط ۱ شروع حلقه برای کل روزهای پیشبینی شده که قرار است به تعداد روزهای پیشبینی نمودار برای هرروز ایجاد کند و مقدار MAPE هر روز را به صورت یک میله در نمودار میله‌ای ایجاد کند، خط۲ شروع حلقه برای کل ساعات پیشبینی شده که تمام آن برای هر روز یک نمودار می‌شود و مقدار MAPE به دست

DOI: 10.30503/nripress.2020.321

آمده از آن روز یک میله را در نمودار میله ای ایجاد می‌کند. خط ۳ رسم هر نمودار کامل مربوط به پیشبینی‌های یک روز در مقایسه با مقدار های واقعی همان روز در همان ساعات. خط ۴ محاسبه MAPE برای روز پیشبینی شده. خط ۵ اضافه کردن مقدار پیشبینی شده MAPE مربوط به همان روز به آرایه پیش‌بینی‌های MAPE. خط ۸ رسم نمودار میله‌ای مربوط به MAPE پیشبینی‌ها. کد کامل مربوط به آموزش و پیش‌بینی شبکه ترکیبی LSTM و MLP همراه با کد مربوط به رسم نمودارهای مربوط به پیش‌بینی انجام شده توسط این شبکه در پیوست الف آمده است.

شبه‌کد6 : الگوریتم مروبط به نمایش نتایج پیش‌بینی با شبکه ترکیبی LSTM و MLP

```
1  For i = 1 to  predction_shifts {
2      For j = 1 to floor(Shift_hours/24){
3          Plot(Y_All_predicttions[i][(j-1) * 24: j * 24] , Y_test[i])
4          MAPE = MAPE_calculator(Y_All_predicttions[i][(j-1) * 24: j * 24], Y_test[i])
5          Bar_array = append(Bar_array, MAPE)
6      }
7  }
8  Bar(Bar_array)
```

۹-٤- خروجی‌های مسئله

در این کد، نمودارهای خروجی‌های به صورت فایل‌های تصویری در پوشه results_figures که در پوشه مسیر کد قرار دارد ذخیره می‌شود و حاوی اطلاعات MAPE و تاریخ روز مربوطه در فایل تصویری است. نمونه‌هایی از آن در ادامه آمده است.

۹-٤-۱- نتایج اجراء کد

همانند فصل سوم بهترین نتایج با استفاده از انجام دسته‌بندی‌های زیر برای روزهای مشابه بدست آمده است. روزهای ورودی به دو دسته زیر تقسیم شده‌اند.

دسته اول: روزهای غیر از پنج‌شنبه، جمعه و تعطیلات رسمی

دوسته دوم: روزهای عادی شامل شنبه، یکشنبه، دوشنبه، سه شنبه و چهارشنبه که تعطیل نیستند

بهترین نتایج برای تنظیمات و ساختار شبکه برای دسته اول به صورت زیر است.

مشخصات شبکه:

- تعداد ویژگی‌های ورودی غیر از بار: ۷ ویژگی
  - سال
  - روز در سال
  - روز در هفته
  - ساعت در روز

DOI: 10.30503/nripress.2020.321

- وضعیت آسمان
- دما
- عاشورا-تاسوعا

- تعداد گام‌های زمانی مربوط به بار ورودی به شبکه: ۴۸ گام
- تعداد خروجی ها: ۱ (فقط بار)
- تعداد نودها در لایه LSTM: ۹۶
- تعداد نودها در لایه MLP: ۴۰
- تعداد نودها در لایه MLP بعد از تجمیع‌کننده: ۱۳۶
- درصدی از داده‌های آموزش برای اعتبار سنجی توسط خود شبکه: ۱۰٪

تنضیمات آموزش شبکه:

- تعداد حداکثر دوره‌ها (epochs): ۱۰۰
- توقف آموزش بعد از چند دوره (epochs): ۶۱
- تحمل‌پذیری عدم بهبود شبکه بعد از چند دوره (epochs): ۱۰
- تابع optimizer: adam (تغییر نرخ آموزش در هر دوره با مقادیر پیشفرض)
- تابع loss: mse

زمان لازم برای آموزش شبکه: ۳۰ دقیقه
بهترین نتایج برای تنظیمات و ساختار شبکه برای دسته دوم به صورت زیر است.
مشخصات شبکه:

- تعداد ویژگی‌های ورودی غیر از بار: ۷ ویژگی
  - سال
  - روز در سال
  - روز در هفته
  - ساعت در روز
  - وضعیت آسمان
  - دما

DOI: 10.30503/nripress.2020.321

- عاشورا-تاسوعا

- تعداد گام‌های زمانی مربوط به بار ورودی به شبکه: گام
- تعداد خروجی‌ها: ۱ (فقط بار)
- تعداد نودها در لایه LSTM: ۹۶
- تعداد لایه های MLP و تعداد نودها در هر لایه: ۳ لایه در هر لایه ۲۰ نود
- تعداد نودها در لایه MLP بعد از تجمیع‌کننده: ۱۳۶ نود
- درصدی از داده‌های آموزش برای اعتبار سنجی توسط خود شبکه: ۱۰٪

تنضیمات آموزش شبکه:

- تعداد حداکثر دوره‌ها (epochs): ۱۰۰
- توقف آموزش بعد از چند دوره (epochs): ۳۶
- تحمل پذیری عدم بهبود شبکه بعد از چند دوره (epochs): ۱۰
- تابع optimizer: adam (تغییر نرخ آموزش در هر دوره با مقادیر پیشفرض)
- تابع lose: mse

زمان لازم برای آموزش شبکه: ۱۸ دقیقه

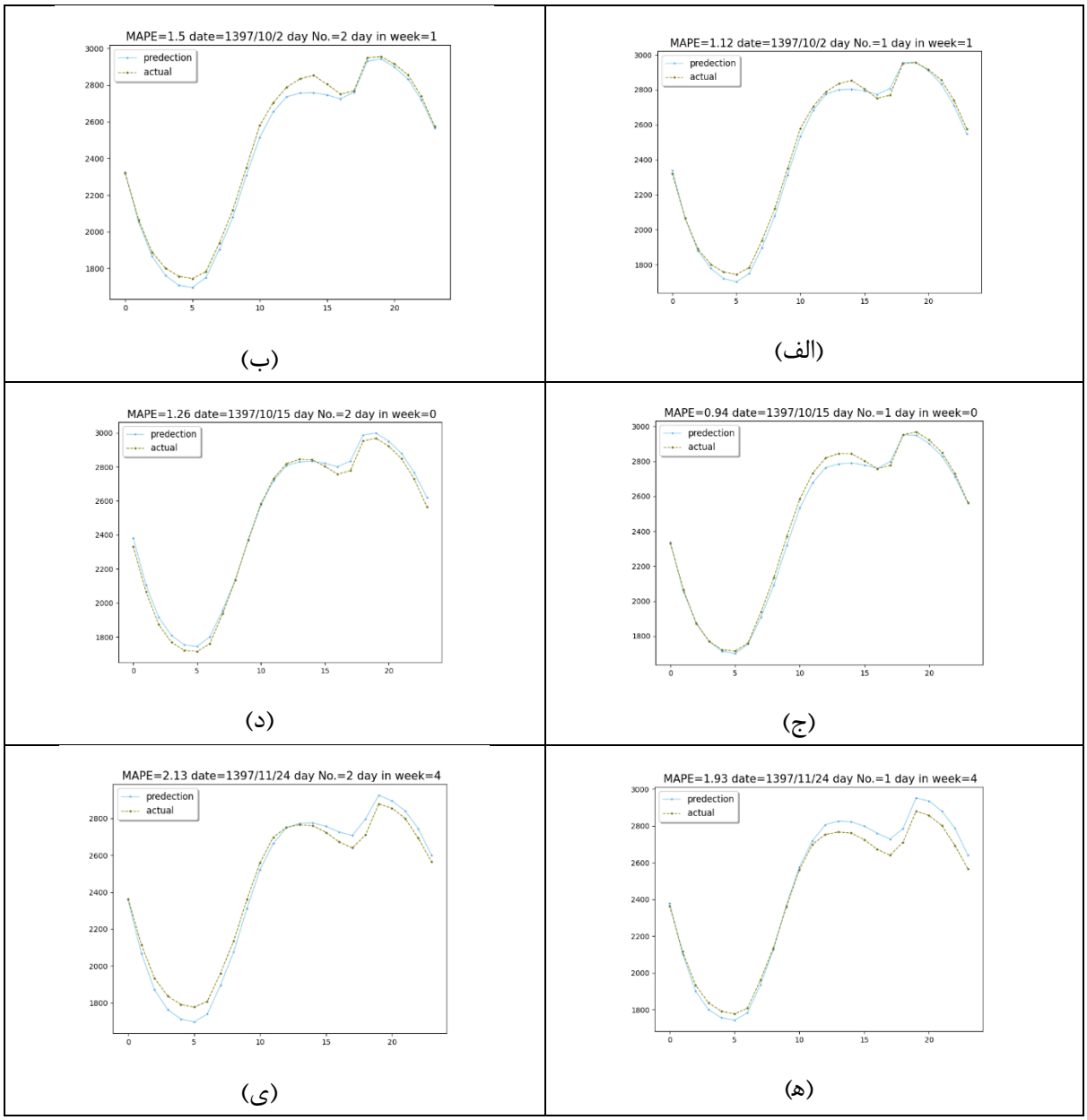
توجه کنید که پش‌بینی هر روز دو بار انجام شده است؛ یکبار زمانی که روز مورد پیش‌بینی ابتدای ۴۸ ساعت است و یک بار زمانی که انتهای آن است. بنابراین روزهایی که در ابتدای ۴۸ ساعت قراردارند بدون فاصله از داده‌های واقعی هستند و روزهایی که در انتهای ۴۸ ساعت قراردارند به اندازه ۲۴ ساعت از داده‌های واقعی فاصله دارند. پس در پیش‌بینی آن‌ها از خروجی خود شبکه نیز به مقدار بسیار بیشتری استفاده شده است. بنابراین دو نوع پیش‌بینی زیر در نظر گرفته شده است که در ترکیب با دسته‌های بالا ۴ حالت ایجاد می‌شود.

- ۲۴ ساعت اول پیش‌بینی
- ۲۴ ساعت دوم پیش‌بینی

نتایج به دست آمده برای تعدادی از روزها که در پیش‌بینی ۲۴ ساعت اول و دوم همان روز قرار دارد، به صورت نمودار روز کامل نشان داده شده در شکل ۹–۲ می‌باشد. در این تصاویر مقدار روز در هفته از شنبه تا جمعه بین ۰ تا ۶ شماره گذاری شده است. همچنین شماره روز (که با Day No. در بالای نمودار درج شده) نشان دهنده روز چندم از پیش‌بینی انجام شده بدون استفاده از داده‌های واقعی جدید بار است که با اضافه کردن خروجی خود شبکه به ورودی آن پیش‌بینی انجام شده است. مقدار MAPE و تاریخ مربوطه نیز در بالای هر نمودار آمده است.

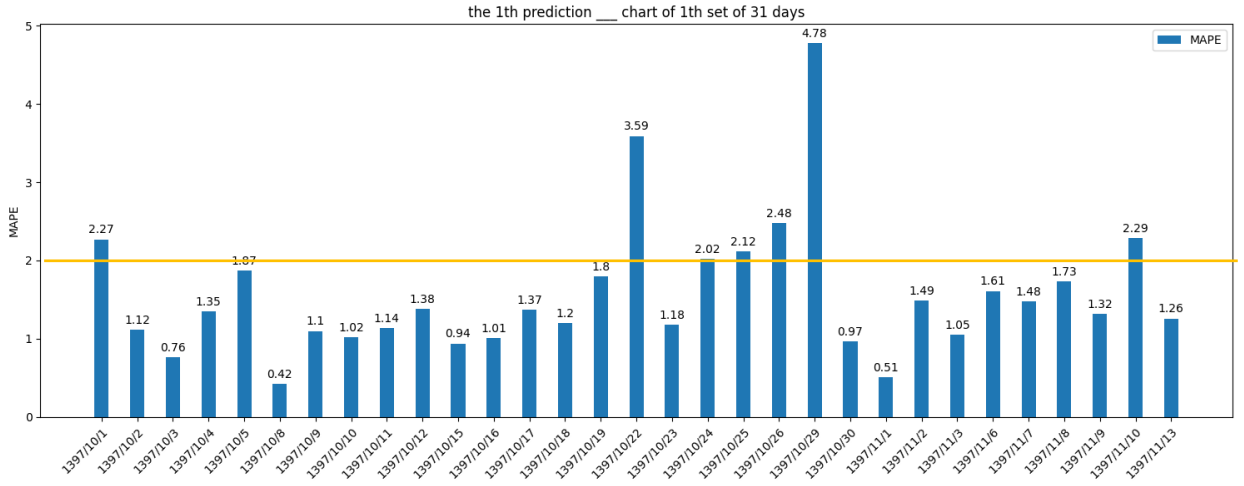
DOI: 10.30503/nripress.2020.321

مقدار MAPE مربوط به کل روزهای دسته اول در ۲۴ ساعت اول پیش‌بینی در نمودارهای شکل ۹–۳ و شکل ۹–۴ نشان داده شده است. همینطور مقدار MAPE مربوط به کل روزهای دسته اول در ۲۴ ساعت دوم پیش‌بینی در نمودارهای شکل ۹–۵ و شکل ۹–۶ نشان داده شده است. در بالای هر نمودار قبل از زیر خط، روز چندم پیش‌بینی مشخص شده یعنی پیش‌بینی مربوط به ۲۴ ساعت اول یا دوم می‌باشد. حداکثر تعداد روزهای قابل نمایش نیز بعد از زیرخط مشخص شده و تاریخ مربوط به هر روز در پایین هر میله آمده است.

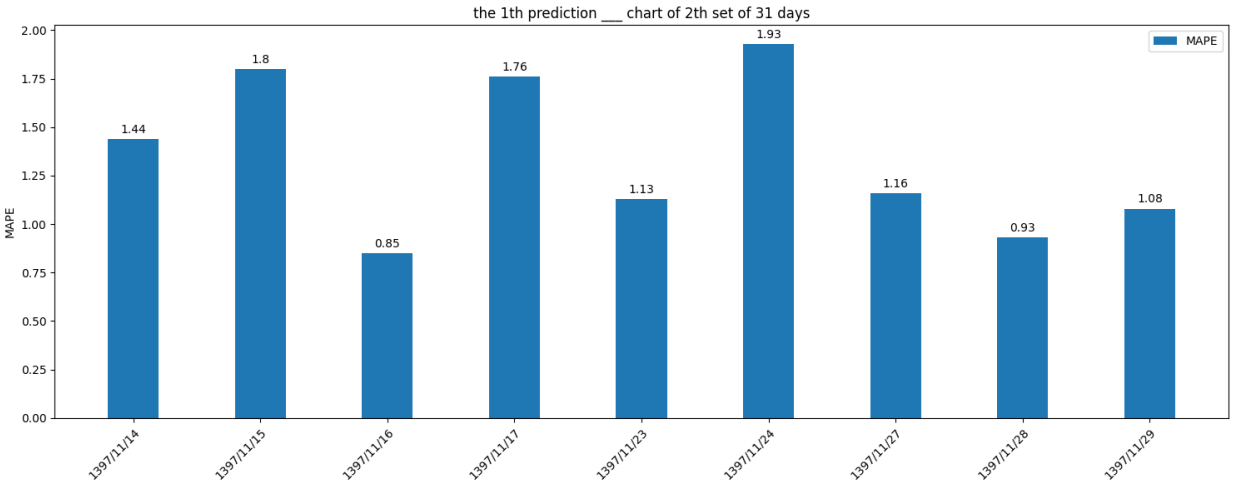


شکل ۹–۲: پیش‌بینی تعدادی از روزهای دسته اول در ۲٤ ساعت اول و دوم پیش‌بینی؛ تصاویر (الف)، (ج) و (ه) مربوط به ۲٤ ساعت اول و تصاویر (ب)، (د) و (ی) مربوط به ۲٤ ساعت دوم

DOI: 10.30503/nripress.2020.321



شکل ۹–۳: مقدار MAPE مربوط به ۳۱ روز ابتدایی بازه پیش‌بینی از نوع دسته اول در ۲٤ ساعت اول پیش‌بینی

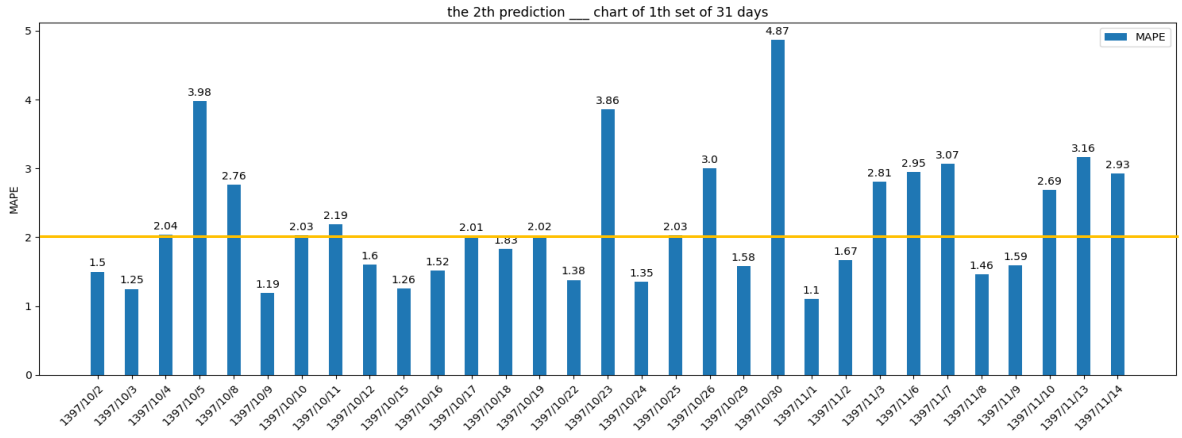


شکل ۹–٤: مقدار MAPE مربوط به ۹ روز انتهایی بازه پیش‌بینی از نوع دسته اول ۲٤ ساعت اول پیش‌بینی

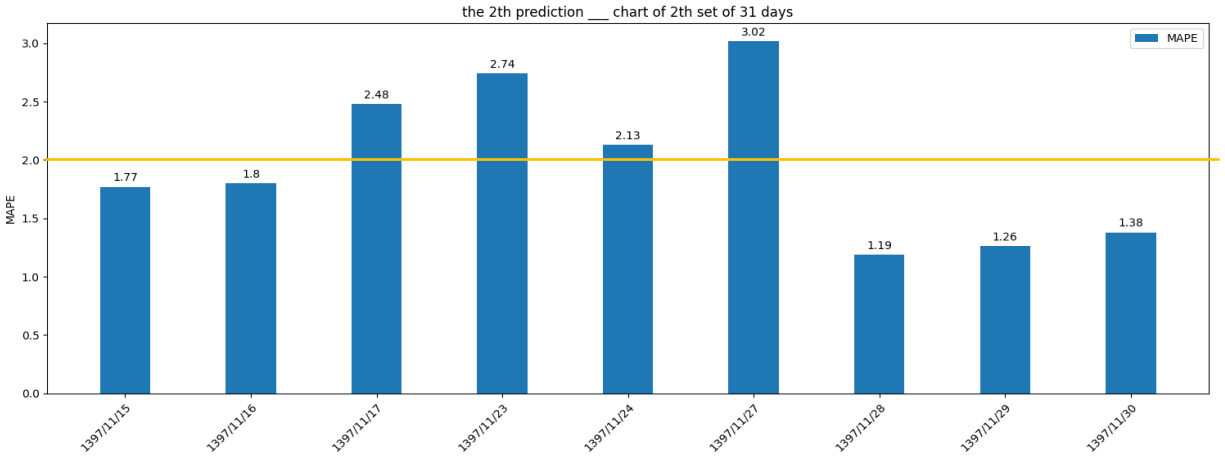
توجه داشته باشید که در نمودارهای موجود در شکل ۸–۴، شکل ۸–۵، شکل ۸–۴ و شکل ۸–۵ تمام روزهای بازه مورد پیش‌بینی (۱۳۹۷/۱۰/۱ تا ۱۳۹۷/۱۱/۳۰) می‌باشد.

همانطور که با مقایسه شکل ۸–۴ و شکل ۸–۶ مشاهده می‌شود، با افزایش فاصله از داده‌های واقعی مقدار MAPE افزایش می‌یابد. تعداد بیشتری از نتایج بالای ۵ درصد قرار می‌گیرند، همچنین این موضوع با مقایسه نمودارهای شکل ۸–۵ و شکل ۸–۷ قابل استنباط است. از طرف دیگر تعداد روزهای با MAPE کمتر از ۲ درصد در روزهای انتهایی باز پیش‌بینی (۱۳۹۷/۱۰/۱ تا ۱۳۹۷/۱۱/۳۰) بیشتر از روزهای ابتدایی است که این موضوع را می‌توان با مقایسه نمودارهای موجود در شکل ۸–۴ و شکل ۸–۵ تایید کرد.

DOI: 10.30503/nripress.2020.321

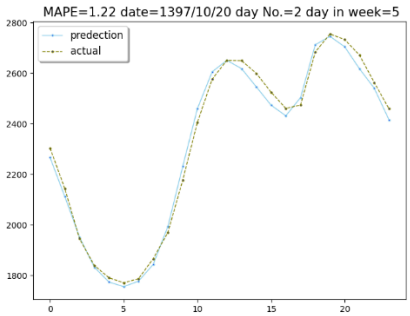


شکل ۹–۵: مقدار MAPE مربوط به ۳۱ روز ابتدایی بازه پیش‌بینی از نوع دسته اول در ۲۴ ساعت دوم پیش‌بینی



شکل ۹–۶: مقدار MAPE مربوط به ۹ روز انتهایی بازه پیش‌بینی از نوع دسته اول در ۲۴ ساعت دوم پیش‌بینی

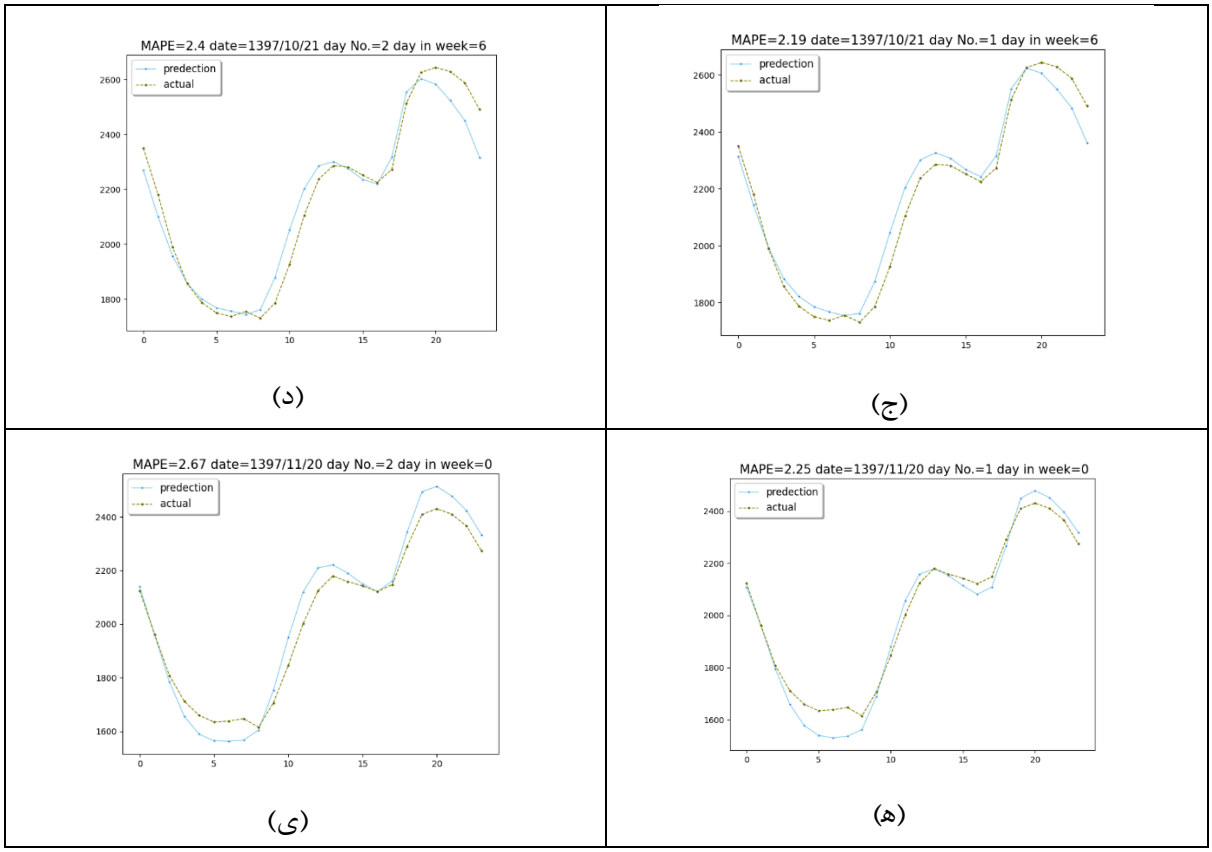
در شکل ۸–۸ نمودار روز کامل مربوط به تعدادی از روزهای دسته دوم در ۲۴ ساعت اول و دوم پیش‌بینی برای هر روز مشخص آمده است. نمودارهای موجود در شکل ۸–۹ و شکل ۸–۱۰ مقدار MAPE تمام روزهای دسته دوم در بازه پیش‌بینی (۱۳۹۷/۱۰/۱ تا ۱۳۹۷/۱۱/۳۰) را نشان می‌دهد.



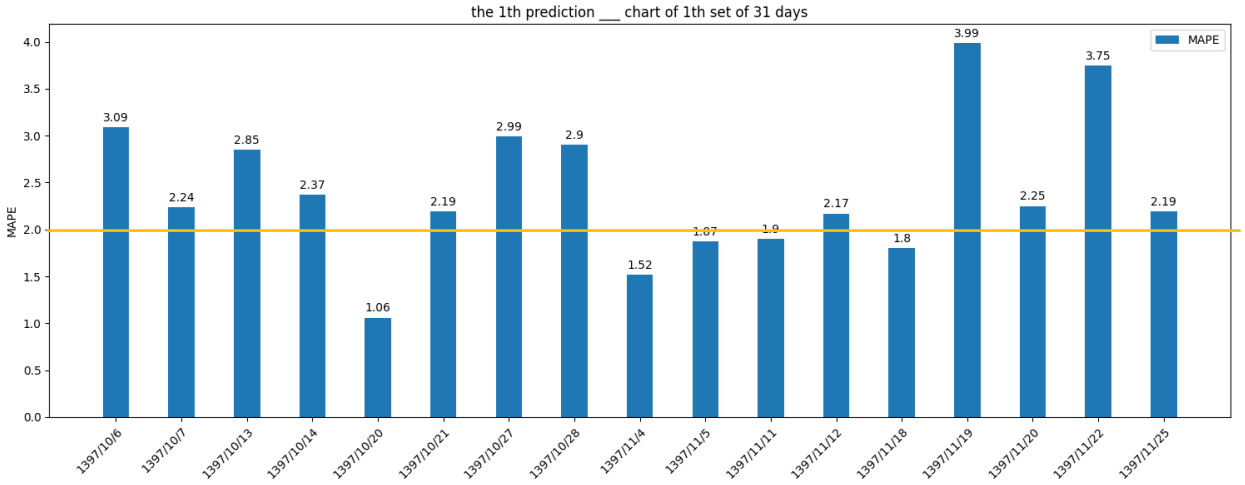
(الف)

(ب)

DOI: 10.30503/nripress.2020.321

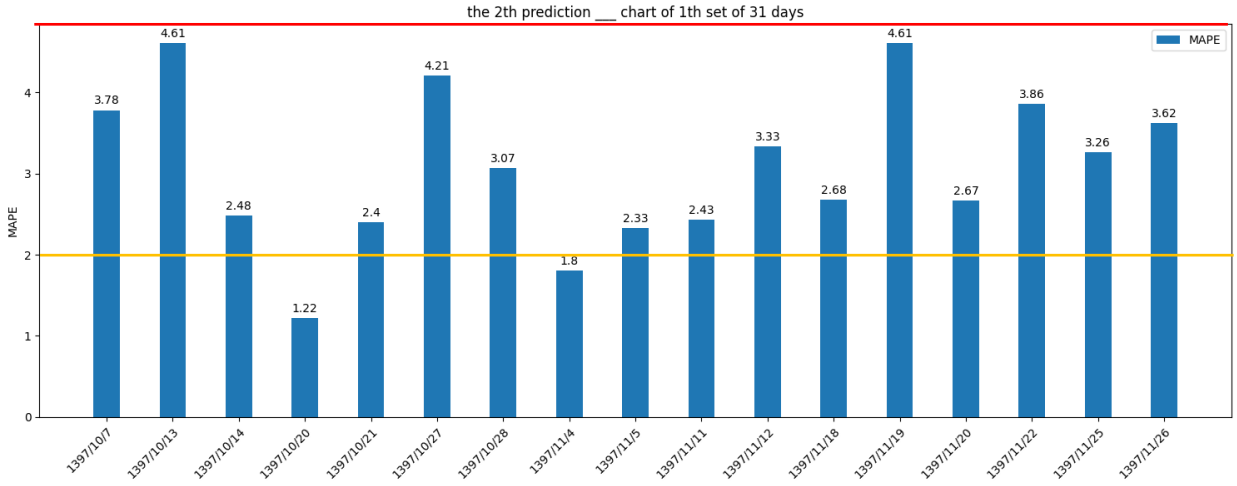


شکل ۹–۷: پیش‌بینی تعدادی از روزهای دسته دوم در ۲۴ ساعت اول و دوم پیش‌بینی؛ تصاویر (الف)، (ج) و (ه) مربوط به ۲۴ ساعت اول و تصاویر (ب)، (د) و (ی) مربوط به ۲۴ ساعت دوم



شکل ۹–۸: مقدار MAPE مربوط به ۳۱ روز ابتدایی بازه پیش‌بینی از نوع دسته دوم در ۲۴ ساعت اول پیش‌بینی

DOI: 10.30503/nripress.2020.321



شکل ۹–۹: مقدار MAPE مربوط به ۳۱ روز انتهایی بازه پیش‌بینی از نوع دسته دوم در ۲٤ ساعت دوم پیش‌بینی

همانطور که در شکل ۸–۹ و شکل ۸–۱۰ مشخص است همانند دسته اول داده‌ها، با افزایش فاصله از بارهای واقعی دقت کاهش می‌یابد.

همانطور که از نمودارهای موجود در شکل ۸–۴، شکل ۸–۵، شکل ۸–۴، شکل ۸–۵، شکل ۸–۹ و شکل ۸–۱۰ مشخص است، روزهای با MAPE بالای ۵ درصد در این روش (ترکیب MLP و LSTM) وجود ندارد. ترکیب MLP و LSTM در ۲۴ ساعت دوم پیش‌بینی بسیار بهتر از روش موجود در فصل قبل عمل کرده است، در فصل قبل از شبکه LSTM و فقط بار به عنوان ورودی استفاده شده بود. در اینجا ویژگی‌ها نیز به بخش دیگری از شبکه با ساختار متفاوت اعمال گردیده است. این کار زمان یادگیری و تعداد ابرپارامترهایی که توسط کاربر باید تنظیم گردد را افزایش می‌دهد. در بخش دوم که بار مربوط به روزهای دسته دوم می‌باشد، با توجه به کم بودن نوع داده‌ها و عدم قطعیت بیشتر نتایج بسیار بهتری حاصل گردید.

۹–۵– نتیجه‌گیری

در این گزارش، در فصل اول، نرم افزار پایتون و مفاهیم اولیه کتابخانه تنسورفلو معرفی شد. در فصل دوم، نحوه کدنویسی و به کارگیری کلاس‌ها و توابع مورد نیاز جهت پیش‌بینی یک سری بار نمونه توسط شبکه عصبی LSTM ارایه گردید. در کنار متغیرهای بار، ویژگی‌های آب و هوایی و ویژگی‌های تقویمی جهت آموزش شبکه مورد استفاده قرار گرفت. در فصل سوم، شبکه عصبی LSTM برای ورودی های فقط متغیر بار آموزش داده شد، به این صورت که تفکیک بارها براساس نوع روز انجام شده بود. نتایج نشان داد که تعداد روزهای با MAPE کمتر از ۲ درصد در روزهای ابتدایی بازه پیش‌بینی بیشتر از روزهای انتهایی است. در فصل چهارم، یک شبکه ترکیبی از دو شبکه عصبی LSTM و MLP جهت پیش‌بینی بار پیشنهاد داده شد. بخش LSTM برای استخراج ویژگی بارهای ورودی به کار برده شده که دارای گام‌های زمانی نیز می‌باشد و بخش دیگر یعنی شبکه MLP اثر ویژگی‌های آب و هوایی را درکنار ویژگی‌های تقویمی روی بار ساعتی که قرار است پیش‌بینی شود، درنظر می‌گیرد. در

این مطالعه نیز تفکیک بارها براساس نوع روز انجام شده بود. در این فصل نیز نشان داده شد که تعداد روزهای با MAPE کمتر از ۲ درصد در روزهای ابتدایی باز پیش‌بینی قرار می‌گیرند.

نتایج پیش‌بینی با MAPE های کمتر از ۲ درصد از اهمیت ویژه‌ای برخودار بوده، مقادیر MAPE بین ۲ تا ۵ درصد قابل قبول است و بیشتر از ۵ درصد جریمه درپی دارد. بنابراین مقادیر MAPE نتایج پیش‌بینی در شبکه ترکیبی MLP و LSTM برای کل بازه پیش‌بینی ۲ ماه و برای تمام انواع روز، براساس سه بازه فوق شمارش شده است و در نمودار پایچارت در شکل ۹–۱۰ نشان داده شده است.

شکل ۹–۱۰: درصد روزهای با مقادیر MAPE در بازه ۰ تا ۲ درصد، ۲ تا ۵ درصد و بیشتر از ۵ درصد

همانطور که در شکل ۹–۱۰ مشخص است هیچ مقدار MAPE بیشتر از ۵ درصد وجود ندارد بنابراین نمودارها فقط ار دو رنگ آبی (مربوط به MAPE کمتر از ۲ درصد) و رنگ نارنجی (مربوط به MAPE بین ۲ تا ۵ درصد) تشکیل شده‌اند و رنگ قرمز در آن وجود ندارد.

مراجع

[1] Guttag, John V. (12 August 2016). *Introduction to Computation and Programming Using Python: With Application to Understanding Data*. MIT Press. ISBN 978-0-262-52962-4.

[2] https://www.freecodecamp.org/news/an-a-z-of-useful-python-tricks-b467524ee747/

[3] Google, “Introduction to TensorFlow,” TensorFlow. https://www.tensorflow.org/learn (accessed Sep. 10, 2020).

پیوست الف: کد کامل مدل‌های پیش‌بینی

الف–۱ کد کامل شبکه LSTM با ورودی فقط بار

با توجه به این‌که تقریبا تمام بخش‌های این کد و توابع به کار رفته در فصل دوم توضیح داده شده است. در اینجا فقط کد کامل که پیش‌بینی با استفاده از آن انجام شده، آمده است.

```
import xlrd
```

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from tqdm import tqdm_gui
from tqdm import tqdm
from joblib import Parallel, delayed
# import xlsxwriter as xlsw


#--------- modules
from modules.preparing_module import varPreparing
from modules.preparing_module import utilities



#--------- ANN
import math
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error



# ---------------------------- load data (tehran)
TehranLoadReal = pd.read_csv('data/tehran/TehranLoadReal_repaired.csv',  index_col=False)
TehranDemandResponse = pd.read_excel('data/tehran/TehranDemandResponse.xlsx',
sheet_name='Sheet1', header=None, index_col=False)
TehranTotalLoad = pd.read_csv('data/tehran/TehranTotalLoad_repaired.csv',  index_col=False)
AllLoads88_to_97 = pd.read_csv('data/tehran/AllLoads88_to_97-11_repaired.csv',
index_col=False)
TehranLoadReal.columns=['year','month','day', 'col1', 'col2', 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]
TehranDemandResponse.columns=['year','month','day', 'col1', 'col2', 0, 1, 2, 3, 4, 5, 6, 7, 8,
9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]
TehranTotalLoad.columns=['year','month','day', 'col1', 'col2', 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]
AllLoads88_to_97.columns=['year','month','day', 'col1', 'col2', 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]


# ---------------------------- weather and calendar
allWeatherData = pd.read_csv('data/allWeatherData_with_dayNum_weekNum_repaired.csv',
index_col=False)
Holidaies_Ashora_Tasoaa =
pd.read_excel('data/Holidays_Ashora_Tasoaa_88_99_Bitween_Holidays_is_Holiday.xlsx',
sheet_name='calendar',  index_col=False)



print('Separating days ...')
headers = ['year','month','day', 'col1', 'col2', 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
14, 15, 16, 17, 18, 19, 20, 21, 22, 23]
headers1 = ['year', 'month', 'day', 'day in year', 'day in week', 'hour', 'icon',
'temperature', 'apparent Temperature', 'dew Point', 'humidity', 'wind Speed', 'wind Bearing',
'cloud Cover', 'uv Index', 'precipitation Intensity', 'precipitation Probability']
headers2 = ['year', 'month', 'day', 'week_day', 'vacation', 'ashora_tasoaa']

TehranLoadReal_selected = varPreparing.selectNormalDay(TehranLoadReal,
Holidaies_Ashora_Tasoaa, headers)
```

```
TehranDemandResponse_selected = varPreparing.selectNormalDay(TehranDemandResponse,
Holidaies_Ashora_Tasoaa, headers)
TehranTotalLoad_selected = varPreparing.selectNormalDay(TehranTotalLoad,
Holidaies_Ashora_Tasoaa, headers)
AllLoads88_to_97_selected = varPreparing.selectNormalDay(AllLoads88_to_97,
Holidaies_Ashora_Tasoaa, headers)

allWeatherData_selected = varPreparing.selectNormalDayColl(allWeatherData,
Holidaies_Ashora_Tasoaa, headers1)
Holidaies_Ashora_Tasoaa_selected = varPreparing.selectNormalDayColl1(Holidaies_Ashora_Tasoaa,
headers2)



# ---------------------------------------------- Initial paramers
train_start_date = [1390, 1, 1] #[1388, 1, 1]
train_end_date = [1397, 9, 30]

test_start_date = [1397, 10, 1]
test_end_date = [1397, 11, 30]




# ------------------------------------------- get the start day index for tests
test_weather_start_index = varPreparing.get_index_s(allWeatherData_selected, test_start_date)
test_load_start_index = varPreparing.get_col_index_s(AllLoads88_to_97_selected,
test_start_date)



# ------------------------------------------  Making weather variable
train_weather_arrs =
Parallel(n_jobs=11)(delayed(varPreparing.cutData)(allWeatherData_selected, train_start_date,
train_end_date, col_name) for col_name in ['temperature', 'humidity' , 'dew Point', 'wind
Speed'])

train_temperature_arr = train_weather_arrs[0]
train_humidity_arr = train_weather_arrs[1]
train_dewPoint_arr = train_weather_arrs[2]
train_windSpeed_arr = train_weather_arrs[3]

test_weather_arrs = Parallel(n_jobs=11)(delayed(varPreparing.cutData)(allWeatherData_selected,
test_start_date, test_end_date, col_name) for col_name in ['temperature', 'humidity' , 'dew
Point', 'wind Speed'])

test_temperature_arr = test_weather_arrs[0]
test_humidity_arr = test_weather_arrs[1]
test_dewPoint_arr = test_weather_arrs[2]
test_windSpeed_arr = test_weather_arrs[3]

print('Weather variables is ready!')



#---------------------------------------- Making power load variable
# --------
train_loads_88_97_arrs = Parallel(n_jobs=11)(delayed(varPreparing.cutDataColRange)(allData,
train_start_date, train_end_date, range(24)) for allData in [AllLoads88_to_97_selected])
train_AllLoads88_to_97_arr = train_loads_88_97_arrs[0]
```

DOI: 10.30503/nripress.2020.321

```
test_loads_88_97_arrs = Parallel(n_jobs=11)(delayed(varPreparing.cutDataColRange)(allData,
test_start_date, test_end_date, range(24)) for allData in [AllLoads88_to_97_selected])
test_AllLoads88_to_97_arr = test_loads_88_97_arrs[0]



print('Load variables is ready!')


#---------------------------------------- Serializing rows in variables
train_serialize_88_to_97_vars = Parallel(n_jobs=11)(delayed(varPreparing.serialize)(var) for
var in [train_AllLoads88_to_97_arr])
train_AllLoads88_to_97_ser = train_serialize_88_to_97_vars[0]


test_serialize_88_to_97_vars = Parallel(n_jobs=11)(delayed(varPreparing.serialize)(var) for
var in [test_AllLoads88_to_97_arr])
test_AllLoads88_to_97_ser = test_serialize_88_to_97_vars[0]









#------------------------------------------------------------------------ NN
# choose a number of time steps
n_lockBack = 48

prediction_days = 2
shift_days = 1
BarChart_days = 31

if prediction_days < shift_days:
    raise Exception('Error: "prediction_days" must be greater than or equal to "shift_days"')



# split into samples
X, y = varPreparing.split_sequence(train_AllLoads88_to_97_ser, n_lockBack)
X_norm, y_norm = varPreparing.normalize(X, X) ,varPreparing.normalize(y, y)

test = test_AllLoads88_to_97_ser
test = np.hstack((train_AllLoads88_to_97_ser[-n_lockBack:], test))
X_test, y_test = varPreparing.split_sequence(test, n_lockBack)
X_test_norm = varPreparing.normalize(X_test, X)







# reshape from [samples, timesteps] into [samples, timesteps, features]
n_features = 1
X_norm = X_norm.reshape((X_norm.shape[0], X_norm.shape[1], n_features))



# define model (first)
model = Sequential()
model.add(LSTM(240, activation='relu', input_shape=(n_lockBack, n_features)))
```

```
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')

# fit model
model.fit(X_norm, y_norm, epochs=60, verbose=1)


# predection
predicted_y_arr = np.array([])
predicted_arr_days = list()
testXNew = np.expand_dims(X_test_norm[0], axis=(0,2))


total_days = math.floor(X_test_norm.shape[0]/24)
total_number_of_shift = math.floor(total_days/shift_days)


total_number_of_shift_Bar = tqdm(range(0, total_number_of_shift))
for i in total_number_of_shift_Bar:
    if ((i * shift_days * 24) + (prediction_days * 24)) <= X_test_norm.shape[0]:
        for j in range(0,(prediction_days*24)):
                predicted_y = model.predict(testXNew, verbose=0)
                testXNew_temp = np.zeros((1,n_lockBack,1))
                testXNew_temp[0,0:-1,0] = testXNew[0,1:n_lockBack, 0]
                testXNew_temp[0,(n_lockBack-1),0] = predicted_y
                testXNew = testXNew_temp
                predicted_y_arr = np.append(predicted_y_arr,
varPreparing.unnormalize(predicted_y, y))
        predected_days = list()
        real_days = list()
        predected_MAPE_days = list()
        predected_date_days = list()
        predected_DiW_days = list()
        for k in range(0,prediction_days):
                predected_day_vals = predicted_y_arr[(k * 24) : ((k + 1) * 24)]
                predected_days.append(predected_day_vals)
                pointer = ((i * shift_days * 24) + (k * 24))
                real_days.append(test_AllLoads88_to_97_ser[pointer:(pointer + 24)])
                MAPE = utilities.mape(test_AllLoads88_to_97_ser[pointer:(pointer + 24)],
predected_day_vals)
                predected_MAPE_days.append(MAPE)
                load_idx = test_load_start_index + ((i * shift_days) + k)
                day_date = str(int(AllLoads88_to_97_selected.loc[load_idx,'year'])) + '/' +
str(int(AllLoads88_to_97_selected.loc[load_idx,'month'])) + '/' +
str(int(AllLoads88_to_97_selected.loc[load_idx,'day']))
                predected_date_days.append(day_date)
                predected_DiW_days.append(allWeatherData_selected['day in
week'][test_weather_start_index + pointer])
        predicted_arr_days.append([predected_days, real_days, predected_MAPE_days,
predected_date_days, predected_DiW_days])
        # Replacing last real load to make last time-step-set to prediction
        testXNew = np.expand_dims(X_test_norm[(i + 1) * shift_days * 24], axis=(0,2))
        # reset predection array
        predicted_y_arr = np.array([])
    # progress Bar
    total_number_of_shift_Bar.set_description("Predection progress %s" % i)



# ---  plot
predicted_arr_days_len = len(predicted_arr_days)
```

```
pltNum = 0
for i in range(0, predicted_arr_days_len):
    item_len = len(predicted_arr_days[i][0])
    for k in range(0, item_len):
        plt.figure(pltNum, figsize=(8, 6), dpi=100)
        plt.plot(predicted_arr_days[i][0][k], marker='o', markerfacecolor='blue',
markersize=2, color='skyblue', linewidth=1, label="predection")
        plt.plot(predicted_arr_days[i][1][k], marker='o', markerfacecolor='blue',
markersize=2, color='olive', linewidth=1, linestyle='dashed', label="actual")

        plt.title(' MAPE=' + np.array2string(round(predicted_arr_days[i][2][k], 2)) +' date='
+ predicted_arr_days[i][3][k] +' day No.=' + str(k+1) +' day in week=' +
str(int(predicted_arr_days[i][4][k])) , fontsize=15)
    plt.legend(loc='best', shadow=True, fontsize='large')
        plt.draw()
        plt.pause(0.001)
        plt.savefig('results_figures/day-' + str(k+1) + '_test'+str(i+1)+'.png')
        pltNum = pltNum + 1
        plt.close()
plt.show()




#------ bar
MAPE_list_arr = list()
item_len = len(predicted_arr_days[0][0])
for k in range(0,item_len):
    MAPE_arr = list()
    MAPE_date_arr = list()
    for i in range(0, predicted_arr_days_len):
        MAPE_arr.append(round(predicted_arr_days[i][2][k], 2))
        MAPE_date_arr.append(predicted_arr_days[i][3][k])
    MAPE_list_arr.append(np.vstack((np.array(MAPE_arr),np.array(MAPE_date_arr))))


for k in range(0,item_len):
    MAPE_len = len(MAPE_list_arr[k][0])
    MAPE_charts = math.floor(MAPE_len/BarChart_days)
    flag_edge = MAPE_len/BarChart_days != math.floor(MAPE_len/BarChart_days)
    if flag_edge:
        MAPE_charts = MAPE_charts + 1
    for i in range(0, MAPE_charts):
        idx_strst = i * BarChart_days
        idx_end = (i + 1) * BarChart_days

        x_bar = np.arange(len(MAPE_list_arr[k][0][idx_strst:idx_end]))
        width = 0.4
        fig_bar, ax = plt.subplots(figsize=(15, 6), dpi=100)
        rects1 = ax.bar(x_bar, MAPE_list_arr[k][0][idx_strst:idx_end].astype(np.float), width,
label='MAPE')
        ax.legend()
        ax.set_ylabel('MAPE')
        ax.set_title('the ' + str(k+1) + 'th prediction ___ chart of ' + str(i+1) + 'th set of
' + str(BarChart_days) + ' days')
        ax.set_xticks(x_bar)
        ax.set_xticklabels(MAPE_list_arr[k][1][idx_strst:idx_end])
        plt.setp(ax.get_xticklabels(), rotation=45, ha="right", rotation_mode="anchor")
        utilities.autolabel(rects1, ax)
        fig_bar.tight_layout()
        plt.draw()
```

DOI: 10.30503/nripress.2020.321

```
        plt.pause(0.001)
        plt.savefig('results_figures/day-'+str(k+1)+'_chart'+str(i+1)+'.png')
        plt.close()
plt.show()



print('\033[92m' + "Done!" + '\033[0m')
```

الف–۲ کد کامل شبکه ترکیبی LSTM و MLP

با توجه به این‌که تقریبا تمام بخش‌های این کد و توابع به کار رفته در فصل دوم توضیح داده شده است. در اینجا فقط کد کامل مروبط به آموزش شبکه ترکیبی LSTM و MLP و پیش‌بینی با استفاده از آن به همراه کد مربوط به رسم نمودار مروبط به نتایج پیش‌بینی آمده است، که با استفاده از آن پیش‌بینی برای یک دسته از روزها آمده است.

```
import xlrd
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from tqdm import tqdm_gui
from tqdm import tqdm
from joblib import Parallel, delayed
import math
import colored
from colored import stylize

#--------- modules
from modules.preparing_module import varPreparing
from modules.preparing_module import utilities

#--------- ANN
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from keras.models import load_model


# ---------------------------- load data (tehran)
TehranLoadReal = pd.read_csv('data/tehran/TehranLoadReal_repaired.csv',  index_col=False)
TehranDemandResponse = pd.read_excel('data/tehran/TehranDemandResponse.xlsx', sheet_name='Sheet1',
header=None, index_col=False)
TehranTotalLoad = pd.read_csv('data/tehran/TehranTotalLoad_repaired.csv',  index_col=False)
AllLoads88_to_97 = pd.read_csv('data/tehran/AllLoads88_to_97-11_repaired.csv',  index_col=False)
TehranLoadReal.columns=['year','month','day', 'col1', 'col2', 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]
TehranDemandResponse.columns=['year','month','day', 'col1', 'col2', 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]
TehranTotalLoad.columns=['year','month','day', 'col1', 'col2', 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]
AllLoads88_to_97.columns=['year','month','day', 'col1', 'col2', 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]


# ---------------------------- weather and calendar
allWeatherData = pd.read_csv('data/allWeatherData_with_dayNum_weekNum_repaired.csv', index_col=False)
Holidaies_Ashora_Tasoaa = pd.read_excel('data/Holidaies_Ashora_Tasoaa_88_99_repaired.xlsx',
sheet_name='calendar',  index_col=False)


print('Separating days ...')
headers = ['year','month','day', 'col1', 'col2', 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
16, 17, 18, 19, 20, 21, 22, 23]
headers1 = ['year', 'month', 'day', 'day in year', 'day in week', 'hour', 'icon', 'temperature',
'apparent Temperature', 'dew Point', 'humidity', 'wind Speed', 'wind Bearing', 'cloud Cover', 'uv
Index', 'precipitation Intensity', 'precipitation Probability']
```

DOI: 10.30503/nripress.2020.321

```
headers2 = ['year', 'month', 'day', 'week_day', 'holiday', 'ashora_tasoaa']

TehranLoadReal_selected = varPreparing.selectNormalDay(TehranLoadReal, Holidaies_Ashora_Tasoaa,
headers)
TehranDemandResponse_selected = varPreparing.selectNormalDay(TehranDemandResponse,
Holidaies_Ashora_Tasoaa, headers)
TehranTotalLoad_selected = varPreparing.selectNormalDay(TehranTotalLoad, Holidaies_Ashora_Tasoaa,
headers)
AllLoads88_to_97_selected = varPreparing.selectNormalDay(AllLoads88_to_97, Holidaies_Ashora_Tasoaa,
headers)

allWeatherData_selected = varPreparing.selectNormalDayColl(allWeatherData, Holidaies_Ashora_Tasoaa,
headers1)
Holidaies_Ashora_Tasoaa_selected = varPreparing.selectNormalDayColl1(Holidaies_Ashora_Tasoaa,
headers2)



# -------------------- Add calendar to weather data to hourly
allWeatherData = varPreparing.add_calendar(allWeatherData_selected, Holidaies_Ashora_Tasoaa_selected)


# ---------------------------------------------- Initial parameters
train_start_date = [1388, 1, 1]  #[1388, 1, 1]
train_end_date = [1397, 9, 30]

test_start_date = [1397, 10, 1]
test_end_date = [1397, 11, 30]


# ---------------------------------------- get the start day index for tests
test_weather_start_index = varPreparing.get_index(allWeatherData, test_start_date)
test_load_start_index = varPreparing.get_col_index(AllLoads88_to_97_selected, test_start_date)


# ----------------------------------------  Making weather variable
train_weather_arrs = Parallel(n_jobs=11)(delayed(varPreparing.cutData)(allWeatherData,
train_start_date, train_end_date, col_name) for col_name in ['temperature', 'humidity' , 'dew Point',
'wind Speed', 'year', 'day in year', 'day in week', 'hour', 'icon', 'holiday', 'ashora_tasoaa'])


train_temperature_arr = train_weather_arrs[0]
train_humidity_arr = train_weather_arrs[1]
train_dewPoint_arr = train_weather_arrs[2]
train_windSpeed_arr = train_weather_arrs[3]
train_year_arr = train_weather_arrs[4]
train_day_in_year_arr = train_weather_arrs[5]
train_day_in_week_arr = train_weather_arrs[6]
train_hour_arr = train_weather_arrs[7]
train_icon_arr = train_weather_arrs[8]
train_holiday_arr = train_weather_arrs[9]
train_Ashora_and_Tasoa_arr = train_weather_arrs[10]



test_weather_arrs = Parallel(n_jobs=11)(delayed(varPreparing.cutData)(allWeatherData, test_start_date,
test_end_date, col_name) for col_name in ['temperature', 'humidity' , 'dew Point', 'wind Speed',
'year', 'day in year', 'day in week', 'hour', 'icon', 'holiday', 'ashora_tasoaa'])

test_temperature_arr = test_weather_arrs[0]
test_humidity_arr = test_weather_arrs[1]
test_dewPoint_arr = test_weather_arrs[2]
test_windSpeed_arr = test_weather_arrs[3]
test_year_arr = test_weather_arrs[4]
test_day_in_year_arr = test_weather_arrs[5]
test_day_in_week_arr = test_weather_arrs[6]
test_hour_arr = test_weather_arrs[7]
test_icon_arr = test_weather_arrs[8]
test_holiday_arr = test_weather_arrs[9]
test_Ashora_and_Tasoa_arr = test_weather_arrs[10]


print('Weather variables is ready!')
```

```
# --------
train_loads_88_97_arrs = Parallel(n_jobs=11)(delayed(varPreparing.cutDataColRange)(allData,
train_start_date, train_end_date, range(24)) for allData in [AllLoads88_to_97_selected])
train_AllLoads88_to_97_arr = train_loads_88_97_arrs[0]


test_loads_88_97_arrs = Parallel(n_jobs=11)(delayed(varPreparing.cutDataColRange)(allData,
test_start_date, test_end_date, range(24)) for allData in [AllLoads88_to_97_selected])
test_AllLoads88_to_97_arr = test_loads_88_97_arrs[0]


print('Load variables is ready!')


#------------------------------------------ Serializing rows in variables weather data
train_serialize_vars = Parallel(n_jobs=11)(delayed(varPreparing.serialize)(var) for var in
[train_temperature_arr, train_humidity_arr, train_dewPoint_arr, train_windSpeed_arr, train_year_arr,
train_day_in_year_arr, train_day_in_week_arr, train_hour_arr, train_icon_arr, train_holiday_arr,
train_Ashora_and_Tasoa_arr])

train_temperature_ser = train_serialize_vars[0]
train_humidity_ser = train_serialize_vars[1]
train_dewPoint_ser = train_serialize_vars[2]
train_windSpeed_ser = train_serialize_vars[3]
train_year_ser = train_serialize_vars[4]
train_day_in_year_ser = train_serialize_vars[5]
train_day_in_week_ser = train_serialize_vars[6]
train_hour_ser = train_serialize_vars[7]
train_icon_ser = train_serialize_vars[8]
train_holiday_ser = train_serialize_vars[9]
train_Ashora_and_Tasoa_ser = train_serialize_vars[10]



test_serialize_vars = Parallel(n_jobs=11)(delayed(varPreparing.serialize)(var) for var in
[test_temperature_arr, test_humidity_arr, test_dewPoint_arr, test_windSpeed_arr, test_year_arr,
test_day_in_year_arr, test_day_in_week_arr, test_hour_arr, test_icon_arr, test_holiday_arr,
test_Ashora_and_Tasoa_arr])

test_temperature_ser = test_serialize_vars[0]
test_humidity_ser = test_serialize_vars[test_humidity_ser = test_serialize_vars[1]
test_dewPoint_ser = test_serialize_vars[2]
test_windSpeed_ser = test_serialize_vars[3]
test_year_ser = test_serialize_vars[4]
test_day_in_year_ser = test_serialize_vars[5]
test_day_in_week_ser = test_serialize_vars[6]
test_hour_ser = test_serialize_vars[7]
test_icon_ser = test_serialize_vars[8]
test_holiday_ser = test_serialize_vars[9]
test_Ashora_and_Tasoa_ser = test_serialize_vars[10]



train_serialize_88_to_97_vars = Parallel(n_jobs=11)(delayed(varPreparing.serialize)(var) for var in
[train_AllLoads88_to_97_arr])
train_AllLoads88_to_97_ser = train_serialize_88_to_97_vars[0]


test_serialize_88_to_97_vars = Parallel(n_jobs=11)(delayed(varPreparing.serialize)(var) for var in
[test_AllLoads88_to_97_arr])
test_AllLoads88_to_97_ser = test_serialize_88_to_97_vars[0]



#------- Define parameters
# choose a number of time steps
n_lockBack = 48

prediction_days = 2
shift_days = 1
BarChart_days = 31

#--- according to features
```

DOI: 10.30503/nripress.2020.321

```
n_features = 7


#------------------ Shift
# weather data must set for output
train_AllLoads88_to_97_ser_input_shift = train_AllLoads88_to_97_ser[:-1]
train_AllLoads88_to_97_ser_output_shift = train_AllLoads88_to_97_ser[1:]
train_temperature_ser_shift = train_temperature_ser[n_lockBack:]
train_humidity_ser_shift = train_humidity_ser[n_lockBack:]
train_dewPoint_ser_shift = train_dewPoint_ser[n_lockBack:]
train_windSpeed_ser_shift = train_windSpeed_ser[n_lockBack:]
train_year_ser_shift = train_year_ser[n_lockBack:]
train_day_in_year_ser_shift = train_day_in_year_ser[n_lockBack:]
train_day_in_week_ser_shift = train_day_in_week_ser[n_lockBack:]
train_hour_ser_shift = train_hour_ser[n_lockBack:]
train_icon_ser_shift = train_icon_ser[n_lockBack:]
train_holiday_ser_shift = train_holiday_ser[n_lockBack:]
train_Ashora_and_Tasoa_ser_shift = train_Ashora_and_Tasoa_ser[n_lockBack:]


test_AllLoads88_to_97_ser_input_shift = np.hstack((train_AllLoads88_to_97_ser[-1],
test_AllLoads88_to_97_ser[:-1]))
test_AllLoads88_to_97_ser_output_shift = test_AllLoads88_to_97_ser
test_temperature_ser_shift = test_temperature_ser
test_humidity_ser_shift = test_humidity_ser
test_dewPoint_ser_shift = test_dewPoint_ser
test_windSpeed_ser_shift = test_windSpeed_ser
test_year_ser_shift = test_year_ser
test_day_in_year_ser_shift = test_day_in_year_ser
test_day_in_week_ser_shift = test_day_in_week_ser
test_hour_ser_shift = test_hour_ser
test_icon_ser_shift = test_icon_ser
test_holiday_ser_shift = test_holiday_ser
test_Ashora_and_Tasoa_ser_shift = test_Ashora_and_Tasoa_ser
if prediction_days < shift_days:
    raise Exception('Error: "prediction_days" must be greater than or equal to "shift_days"')


# horizontally stack columns
Load_dataset =
np.hstack((train_AllLoads88_to_97_ser_input_shift.reshape(len(train_AllLoads88_to_97_ser_input_shift),
1), train_AllLoads88_to_97_ser_output_shift.reshape(len(train_AllLoads88_to_97_ser_output_shift),1)))

# convert into input/output
X_load, y_load = varPreparing.split_sequences(Load_dataset, n_lockBack)
X_load_norm, y_load_norm = varPreparing.normalize(X_load, X_load), varPreparing.normalize(y_load,
y_load)
X_Weather_Calendar = np.hstack((
    train_year_ser_shift.reshape(len(train_year_ser_shift),1),
    train_day_in_year_ser_shift.reshape(len(train_day_in_year_ser_shift),1),
    train_day_in_week_ser_shift.reshape(len(train_day_in_week_ser_shift),1),
    train_hour_ser_shift.reshape(len(train_hour_ser_shift),1),
    train_icon_ser_shift.reshape(len(train_icon_ser_shift),1),
    train_temperature_ser_shift.reshape(len(train_temperature_ser_shift),1),
    # train_holiday_ser_shift.reshape(len(train_holiday_ser_shift),1),
    train_Ashora_and_Tasoa_ser_shift.reshape(len(train_Ashora_and_Tasoa_ser_shift),1)
    ))
y_Weather_Calendar = y_load
X_Weather_Calendar_norm = varPreparing.normalize(X_Weather_Calendar, X_Weather_Calendar)
X_Weather_Calendar_norm = np.expand_dims(X_Weather_Calendar_norm, axis=(2))
y_Weather_Calendar_norm = y_load_norm


#----- for prediction
# #---------------- Var. prepering fpr prediction
test_year_ser_shift_added = test_year_ser_shift
test_day_in_year_ser_shift_added = test_day_in_year_ser_shift
test_day_in_week_ser_shift_added = test_day_in_week_ser_shift
test_hour_ser_shift_added = test_hour_ser_shift
test_icon_ser_shift_added = test_icon_ser_shift
test_temperature_ser_shift_added = test_temperature_ser_shift
test_holiday_ser_shift_added = test_holiday_ser_shift
test_Ashora_and_Tasoa_ser_shift_added = test_Ashora_and_Tasoa_ser_shift
```

DOI: 10.30503/nripress.2020.321

```
test_AllLoads88_to_97_ser_input_added = np.hstack((train_AllLoads88_to_97_ser_input_shift[( -
n_lockBack):], test_AllLoads88_to_97_ser_input_shift))
test_AllLoads88_to_97_ser_output_added = np.hstack((train_AllLoads88_to_97_ser_output_shift[( -
n_lockBack):], test_AllLoads88_to_97_ser_output_shift))


test_load_dataset = np.hstack((
test_AllLoads88_to_97_ser_input_added.reshape(len(test_AllLoads88_to_97_ser_input_added), 1),
test_AllLoads88_to_97_ser_output_added.reshape(len(test_AllLoads88_to_97_ser_output_added), 1)))


# ---- make var
test_X_load, test_y_load = varPreparing.split_sequences(test_load_dataset, n_lockBack)
test_X_load_norm, test_y_load_norm = varPreparing.normalize(test_X_load, X_load),
varPreparing.normalize(test_y_load, y_load)
test_X_Weather_Calendar = np.hstack((
    test_year_ser_shift_added.reshape(len(test_year_ser_shift_added), 1),
    test_day_in_year_ser_shift_added.reshape(len(test_day_in_year_ser_shift_added), 1),
    test_day_in_week_ser_shift_added.reshape(len(test_day_in_week_ser_shift_added), 1),
    test_hour_ser_shift_added.reshape(len(test_hour_ser_shift_added), 1),
    test_icon_ser_shift_added.reshape(len(test_icon_ser_shift_added), 1),
    test_temperature_ser_shift_added.reshape(len(test_temperature_ser_shift_added), 1),
    # test_holiday_ser_shift_added.reshape(len(test_holiday_ser_shift_added), 1),
    test_Ashora_and_Tasoa_ser_shift_added.reshape(len(test_Ashora_and_Tasoa_ser_shift_added), 1)
    ))
test_y_Weather_Calendar = test_y_load
test_X_Weather_Calendar_norm = varPreparing.normalize(test_X_Weather_Calendar, X_Weather_Calendar)
test_X_Weather_Calendar_norm = np.expand_dims(test_X_Weather_Calendar_norm, axis=(2))


#--- Model
Load_input = keras.Input(
    shape=(n_lockBack, 1), name="Load"
)
Weather_Calendar_input = keras.Input(
    shape=(n_features, ), name="Weather_Calendar"
)


Load_features = layers.LSTM(96)(Load_input)
Weather_Calendar_features = layers.Dense(40, activation='relu')(Weather_Calendar_input)


x = layers.concatenate([Load_features, Weather_Calendar_features])

x1 = layers.Dense(136, activation='relu')(x)
# x2 = layers.Dense(136, activation='relu')(x1)

Load_pred = layers.Dense(1, name="OutputLoad")(x1)


model = keras.Model(
    inputs=[Load_input, Weather_Calendar_input],
    outputs=[Load_pred],
)

model.summary()

keras.utils.plot_model(model, "results_figures/model.png", show_shapes=True, dpi=300)

model.compile(
    optimizer=keras.optimizers.Adam(
        learning_rate=0.001,
        beta_1=0.9,
        beta_2=0.999,
        epsilon=1e-07,
        amsgrad=False,
        name="Adam",
    ),
    loss="mse",
)
```

DOI: 10.30503/nripress.2020.321

```
#------- save best
path_checkpoint = "results_figures/model_checkpoint.h5"
es_callback = keras.callbacks.EarlyStopping(monitor="val_loss", min_delta=0, patience=10)

modelckpt_callback = keras.callbacks.ModelCheckpoint(
    monitor="val_loss",
    filepath=path_checkpoint,
    verbose=1,
    save_weights_only=False,
    save_best_only=True,
)


#--- train the model
Train_history = model.fit(
    {"Load": X_load_norm, "Weather_Calendar": X_Weather_Calendar_norm},
    {"OutputLoad": y_load_norm},
    epochs=100,
    # batch_size=1,  # Defult is 32. In some cases doesn't need to define
    validation_split=0.1,
    callbacks=[es_callback, modelckpt_callback],
)


#-----training curve
utilities.visualize_loss(Train_history, 'Training curve', 'results_figures/trining-curve.png')

#----- load best checkpoint
model = load_model('results_figures/model_checkpoint.h5')
```

```
# ----- predection
predicted_y_arr = np.array([])
predicted_arr_days = list()
testXNew = np.expand_dims(test_X_load_norm[0], axis=(0))


total_days = math.floor(test_X_load_norm.shape[0]/24)
total_number_of_shift = math.floor(total_days/shift_days)


total_number_of_shift_Bar = tqdm(range(0, total_number_of_shift))
for i in total_number_of_shift_Bar:
    if ((i * shift_days * 24) + (prediction_days * 24)) <= test_X_load_norm.shape[0]:
        for j in range(0,(prediction_days*24)):
                prediction_pointer = ((i * shift_days * 24) + j)
                predicted_y = model.predict({"Load": testXNew, "Weather_Calendar":
np.expand_dims(test_X_Weather_Calendar_norm[prediction_pointer], axis=(0))}, verbose=0)
                testXNew_temp = np.zeros((1,n_lockBack,1))
                testXNew_temp[0,0:-1,0] = testXNew[0,1:n_lockBack, 0]
                testXNew_temp[0,(n_lockBack-1),0] = predicted_y[0][0]
                testXNew = testXNew_temp
                predicted_y_arr = np.append(predicted_y_arr,
varPreparing.unnormalize(predicted_y[0][0], y_load))
        predected_days = list()
        real_days = list()
        predected_MAPE_days = list()
        predected_date_days = list()
        predected_DiW_days = list()
        for k in range(0,prediction_days):
                predected_day_vals = predicted_y_arr[(k * 24) : ((k + 1) * 24)]
                predected_days.append(predected_day_vals)
                pointer = ((i * shift_days * 24) + (k * 24))
                real_days.append(test_y_load[pointer:(pointer + 24)])
                MAPE = utilities.mape(test_y_load[pointer:(pointer + 24)], predected_day_vals)
                predected_MAPE_days.append(MAPE)
                load_idx = test_load_start_index + ((i * shift_days) + k)
                day_date = str(int(AllLoads88_to_97_selected.loc[load_idx,'year'])) + '/' +
str(int(AllLoads88_to_97_selected.loc[load_idx,'month'])) + '/' +
```

DOI: 10.30503/nripress.2020.321

```
str(int(AllLoads88_to_97_selected.loc[load_idx,'day']))
                predected_date_days.append(day_date)
                predected_DiW_days.append(allWeatherData['day in week'][test_weather_start_index +
pointer])
        predicted_arr_days.append([predected_days, real_days, predected_MAPE_days,
predected_date_days, predected_DiW_days])
        # Replacing last real load to make last time-step-set to prediction
        testXNew = np.expand_dims(test_X_load_norm[(i + 1) * shift_days * 24], axis=(0))
        # reset predection array
        predicted_y_arr = np.array([])
    # progress Bar
    total_number_of_shift_Bar.set_description("Predection progress %s" % i)



# ---  plot
predicted_arr_days_len = len(predicted_arr_days)
pltNum = 1
for i in range(0, predicted_arr_days_len):
    item_len = len(predicted_arr_days[i][0])
    for k in range(0, item_len):
        plt.figure(pltNum, figsize=(8, 6), dpi=100)
        plt.plot(predicted_arr_days[i][0][k], marker='o', markerfacecolor='blue', markersize=2,
color='skyblue', linewidth=1, label="predection")
        plt.plot(predicted_arr_days[i][1][k], marker='o', markerfacecolor='blue', markersize=2,
color='olive', linewidth=1, linestyle='dashed', label="actual")

        plt.title(' MAPE=' + np.array2string(round(predicted_arr_days[i][2][k], 2)) +' date=' +
predicted_arr_days[i][3][k] +' day No.=' + str(k+1) +' day in week=' +
str(int(predicted_arr_days[i][4][k])) , fontsize=15)
        plt.legend(loc='best', shadow=True, fontsize='large')
        plt.draw()
        plt.pause(0.001)
        plt.savefig('results_figures/day-' + str(k+1) + '_test'+str(i+1)+'.png')
        pltNum = pltNum + 1
        plt.close()
plt.show()


#------ bar
MAPE_list_arr = list()
item_len = len(predicted_arr_days[0][0])
for k in range(0,item_len):
    MAPE_arr = list()
    MAPE_date_arr = list()
    for i in range(0, predicted_arr_days_len):
        MAPE_arr.append(round(predicted_arr_days[i][2][k], 2))
        MAPE_date_arr.append(predicted_arr_days[i][3][k])
    MAPE_list_arr.append(np.vstack((np.array(MAPE_arr),np.array(MAPE_date_arr))))


for k in range(0,item_len):
    MAPE_len = len(MAPE_list_arr[k][0])
    MAPE_charts = math.floor(MAPE_len/BarChart_days)
    flag_edge = MAPE_len/BarChart_days != math.floor(MAPE_len/BarChart_days)
    if flag_edge:
        MAPE_charts = MAPE_charts + 1
    for i in range(0, MAPE_charts):
        idx_strst = i * BarChart_days
        idx_end = (i + 1) * BarChart_days
```

DOI: 10.30503/nripress.2020.321

```
        x_bar = np.arange(len(MAPE_list_arr[k][0][idx_strst:idx_end]))
        width = 0.4
        fig_bar, ax = plt.subplots(figsize=(15, 6), dpi=100)
        rects1 = ax.bar(x_bar, MAPE_list_arr[k][0][idx_strst:idx_end].astype(np.float), width,
label='MAPE')
        ax.legend()
        ax.set_ylabel('MAPE')
        ax.set_title('the ' + str(k+1) + 'th prediction ___ chart of ' + str(i+1) + 'th set of ' +
str(BarChart_days) + ' days')
        ax.set_xticks(x_bar)
        ax.set_xticklabels(MAPE_list_arr[k][1][idx_strst:idx_end])
        plt.setp(ax.get_xticklabels(), rotation=45, ha="right", rotation_mode="anchor")
        utilities.autolabel(rects1, ax)
        fig_bar.tight_layout()
        plt.draw()
        plt.pause(0.001)
        plt.savefig('results_figures/day-'+str(k+1)+'_chart'+str(i+1)+'.png')
        plt.close()
plt.show()



print(stylize("Done!", colored.fg(46)))
```