بسم الله الرحمن الرحیم

دانشکده ریاضی و علوم کامپیوتر

پایان‌نامه برای دریافت درجه کارشناسی ارشد در رشته علوم کامپیوتر

گرایش علوم تصمیم و دانش

پیش‌بینی ترافیک شهری به کمک یادگیری عمیق

استادان راهنما

دکتر محمود امین طوسی و دکتر علیرضا قدسی

استاد مشاور

دکتر مهدی مهدی‌زاده

پژوهشگر:

مرتضی جلمبادانی

آذر ۱۴۰۰

بسمه تعالی

فرم ارزشیابی و صورتجلسه دفاع از پایان نامه کارشناسی ارشد

(این فرم الزاما باید به صورت تایپ شده تهیه، ارسال و در پایان نامه درج شود)

جلسه دفاع از پایان نامه آقای **مرتضی جلمبادانی** دانشجوی رشته **علوم کامپیوتر** گرایش **علوم تصمیم و دانش** به شماره دانشجویی **9713185026** با عنوان **پیش‌بینی ترافیک شهری با استفاده از یادگیری عمیق** در تاریخ **1400/09/29** در **دانشکده ریاضی و علوم کامپیوتر** برگزار و توسط هیات داوران مورد ارزشیابی قرار گرفت و نمره.....۱۸,۵..... برابر درجه.....بسیار خوب..... برای آن تعیین گردید.

به این ترتیب از این تاریخ نامبرده به عنوان کارشناس ارشد در رشته مذکور شناخته می شود.

موارد ارزشیابی	موارد	حداکثر نمره	نمره کسب شده
1- کیفیت نگارش	رعایت اصول نگارش انسجام در تنظیم بخش های مختلف، کیفیت تصاویر، جداول و اشکال ،تنظیم فهرست ها، منابع و ماخذ	4.25	۴,۱-
2- کیفیت علمی	بررسی تاریخچه و سابقه تجربی و نظری موضوع انسجام منطقی دربخش های مختلف پایان نامه ،ابتکار و نوآوری ،اهمیت و ارزش علمی پایان نامه ،استفاده از منابع معتبر و جدید ،کیفیت تجزیه و تحلیل یافته ها و نتیجه گیری ،روشن بودن روش کار ،هدف ها و فرضیه های تحقیق ،جدید بودن روش تحقیق	11.5	۱۰,۵
	امتیاز دستاوردهای پایان نامه : مقاله مستخرج از پایان نامه ، ساخت دستگاه ، اجرای طرح ارتباط با صنعت و جامعه ، کسب مقام در استارت آپ ، کاربردی بودن پایان نامه یا برگزاری کرسی علمی و . . . ؛ نمره این قسمت بر اساس فرم 103 اختصاص می یابد .		
3-کیفیت ارایه در جلسه دفاع	تسلط بر موضوع و بیان واضح و تفهیم آن ،توانایی در پاسخگویی به سوالات مطرح شده در جلسه ، رعایت زمان ارایه , روش ارایه	4.25	۴,۱-
	جمع		۱۸,۵

درجه معادل کسب شده: (از 19تا 20 عالی☐ از18تا 18/99بسیار خوب ☑از 16تا 17/99خوب ☐از 14 تا 15/99قابل قبول ☐کمتر از 14 غیر قابل قبول☐)

مشخصات هیات دوران

ردیف	نام و نام خانوادگی	سمت	مرتبه علمی	محل کار	امضا
1	دکتر محمود امین‌طوسی	استاد راهنما اول	استادیار	دانشگاه حکیم سبزواری	
2	دکتر علیرضا قدسی	استاد راهنما دوم	استادیار	دانشگاه حکیم سبزواری	
3	دکتر مهدی مهدی‌زاده	استاد مشاور اول	دانشیار	دانشگاه حکیم سبزواری	
4	دکتر مهدی زعفرانیه	استاد داور	دانشیار	دانشگاه حکیم سبزواری	
5	دکتر محمد بلبلیان قالیباف	نماینده تحصیلات تکمیلی	استادیار	دانشگاه حکیم سبزواری	

امضای
مدیر گروه

امضای
رییس دانشکده

سوگند نامه دانش آموختگان دانشگاه حکیم سبزواری

به نام خداوند جان و خرد کزین برتر اندیشه بر نگذرد

اینک که به خواست آفریدگار پاک، کوشش خویش و بهره گیری از دانش استادان و سرمایه‌های مادی و معنوی این مرز و بوم، توشه‌ای از دانش و خرد گردآورده‌ام، در پیشگاه خداوند بزرگ سوگند یاد می کنم که در به کارگیری دانش خویش، همواره بر راه راست ودرست گام بردارم. خداوند بزرگ، شما شاهدان، دانشجویان و دیگر حاضران را به عنوان داورانی امین گواه می گیرم که ازهمه دانش و توان خود برای گسترش مرزهای دانش بهره گیرم و از هیچ کوششی برای تبدیل جهان به جایی بهتر برای زیستن، دریغ نورزم. پیمان می‌بندم که همواره کرامت انسانی را در نظر داشته باشم و همنوعان خود را در هر زمان و مکان تا سر حد امکان یاری دهم. سوگند می خورم که در به کارگیری دانش خویش به کاری که با راه و رسم انسانی، آیین پرهیزگاری، شرافت و اصول اخلاقی برخاسته از ادیان بزرگ الهی، به ویژه دین مبین اسلام، مباینت دارد دست نیازم. همچنین در سایه اصول جهان شمول انسانی و اسلامی، پیمان می‌بندم از هیچ کوششی برای آبادانی و سرافرازی میهن و هم میهنانم فروگذاری نکنم و خداوند بزرگ را به یاری طلبم تا همواره در پیشگاه او و در برابر وجدان بیدار خویش و ملت سرافراز، بر این پیمان تا ابد استوار بمانم.

نام و نام خانوادگی: مرتضی جلمبادانی

تاریخ و امضا:

تأییدیه‌ی صحت و اصالت نتایج

باسمه تعالی

اینجانب مرتضی جلمبادانی به شماره دانشجویی ۹۷۱۳۱۸۵۰۲۶ دانشجوی رشته علوم کامپیوتر مقطع تحصیلی کارشناسی ارشد تأیید می‌نمایم که کلیه‌ی نتایج این پایان‌نامه حاصل کار اینجانب و بدون هرگونه دخل و تصرف است و موارد نسخه برداری شده از آثار دیگران را با ذکر کامل مشخصات منبع ذکر کرده‌ام. درصورت اثبات خلاف مندرجات فوق، به تشخیص دانشگاه مطابق با ضوابط و مقررات حاکم (قانون حمایت از حقوق مؤلفان و مصنفان و قانون ترجمه و تکثیر کتب و نشریات و آثار صوتی، ضوابط و مقررات آموزشی، پژوهشی و انضباطی ...) با اینجانب رفتار خواهد شد و حق هرگونه اعتراض درخصوص احقاق حقوق مکتسب و تشخیص و تعیین تخلف و مجازات را از خویش سلب می‌نمایم. در ضمن، مسؤولیت هرگونه پاسخگویی به اشخاص اعم از حقیقی و حقوقی و مراجع ذی صلاح (اعم از اداری و قضایی) به عهده ی اینجانب خواهد بود و دانشگاه هیچ گونه مسؤولیتی در این خصوص نخواهد داشت.

نام و نام خانوادگی: مرتضی جلمبادانی

تاریخ و امضا:

مجوز بهره برداری از پایان‌نامه

بهره برداری از این پایان‌نامه در چهارچوب مقررات کتابخانه و با توجه به محدودیتی که توسط استاد راهنما به شرح زیر تعیین می‌شود، بلامانع است:

☐ بهره برداری از این پایان‌نامه برای همگان بلامانع است.

☐ بهره برداری از این پایان‌نامه با اخذ مجوز از استاد راهنما، بلامانع است.

☐ بهره برداری از این پایان‌نامه تا تاریخ ..... ممنوع است.

استادان راهنما: دکتر محمود امین طوسی

دکتر علیرضا قدسی

تاریخ و امضا:

تقدیم به:

پدر و مادرم

سپاس خداوندگار حکیم را که با لطف بی کران خود، آدمی را زیور عقل آراست.
در آغاز وظیفه خود می دانم از زحمات بی دریغ اساتید راهنمای خود، دکتر امین طوسی و دکتر قدسی صمیمانه تشکر و قدردانی کنم که قطعاً بدون راهنمایی‌های ارزنده ایشان، این مجموعه به انجام نمی رسید.
از جناب آقای دکتر مهدی‌زاده که زحمت مطالعه و مشاوره این رساله را تقبل فرمودند و در آماده سازی این رساله، به نحو احسن اینجانب را مورد راهنمایی قرار دادند، کمال امتنان را دارم.
همچنین لازم می دانم از گروه پارسی‌لاتک در پاسخگویی به مشکلات کاربران کمال قدردانی را داشته باشم.
در پایان، بوسه می زنم بر دستان خداوندگاران مهر و مهربانی، پدر و مادر عزیزم و بعد از خدا، ستایش می‌کنم وجود مقدس شان را و تشکر می‌کنم از خانواده عزیزم به پاس عاطفه سرشار و گرمای امیدبخش وجودشان، که بهترین پشتیبان من بودند.

مرتضی جلمبادانی
آذر ۱۴۰۰

فهرست مطالب

فهرست جداول

فهرست تصاویر

فرم چکیده ی پایان‌نامه ی دوره ی تحصیلات تکمیلی

مدیریت تحصیلات تکمیلی

دانشگاه حکیم سبزواری

نام خانوادگی دانشجو:جلمبادانی	نام:مرتضی	ش. دانشجویی:۹۷۱۳۱۸۵۰۲۶
استادان راهنما: دکتر محمود امین طوسی و دکتر علیرضا قدسی		
استاد مشاور: دکتر مهدی مهدی‌زاده		
دانشکده ریاضی و علوم کامپیوتر	رشته: علوم کامپیوتر	گرایش: علوم تصمیم و دانش
مقطع: کارشناسی ارشد	تاریخ دفاع:آذر ۱۴۰۰	تعداد صفحات: ۸۸
عنوان پایان‌نامه: پیش‌بینی ترافیک شهری به کمک یادگیری عمیق		
کلید واژه‌ها:شبکه‌های عصبی، یادگیری عمیق، شبکه پیچشی گراف، شبکه عصبی بازگشتی، پیش‌بینی ترافیک، ترافیک شهری		

چکیده:

تخمین دقیق و پیش‌بینی در لحظه ترافیک نقش پر اهمیتی در سیستم‌های هوشمند کنترل ترافیک شهری و حمل و نقل درون شهری دارد، پیش‌بینی ترافیک شهری همواره از مسائل مورد بحث است که نتایج این مسئله به مدیریت ترافیک و کنترل بهتر طراحی معابر شهری کمک می‌کند. به علت ساختار شبکه معابر شهری و ساختار توپولوژی این شبکه، وابستگی زمانی و وابستگی مکانی عوامل دخیل در پیش‌بینی ترافیک شهری هستند. در این نوشتار به پیش‌بینی ترافیک شهری با استفاده از شبکه عصبی بازگشتی مبتنی بر شبکه پیچشی گراف پرداخته شده است. این روش با ترکیب دو روش شبکه عصبی بازگشتی و شبکه پیچشی گراف طراحی شده است، از شبکه پیچشی گراف در حل وابستگی مکانی ترافیک معابر شهری و از شبکه عصبی بازگشتی در حل وابستگی زمانی ترافیک لحظه ای استفاده شده است. در این تحقیق دو مجموعه داده تاکسی‌های شهر شنزن کشور چین با حدود ۱۵۶ معبر و مجموعه داده حسگرهای بزرگراهی شهر لس‌آنجلس شامل بیش از ۲۰۰ معبر بررسی شده است. نتایج پیرامون آزمایش روش ارائه شده بر روی این دو مجموعه داده‌ها با نتایج برگرفته از روش‌های دیگر مقایسه شده است. میزان تابع‌خطا بر روی نتایج آزمایش روش پیش‌بینی ارائه شده در حدود ٪۵۰ میزان خطا در روش پیش‌بینی ARIMA می‌باشد.

پیش‌گفتار

اطلاعات ترافیک شهری از اهمیت بالایی برخوردارمی‌باشند، از این داده‌ها در مشخص کردن وضعیت روزمره خیابان‌ها، محل نصب چراغ قرمزها، مدت زمان سبز و قرمز بودن هر چراغ، جهت حرکت مسیرهای واصل به هم، طول و عرض ایجاد معبر جدید شهری، افتتاح مراکز درمانی و خدماتی، بهره برداری از خیابان یا میدان‌ها و تقاطع‌ها استفاده می‌شود. هدف از این نوشتار پیش‌بینی ترافیک شهری معابر شهری می‌باشد. برای این پیش‌بینی از ویژگی‌های ترافیک که در جریان هستند استفاده می‌شود تا مدل (شبکه) ارائه شده آموزش ببیند و قادر به پیش‌بینی وضعیت ترافیکی با توجه به شرایط موجود در معابر شهری باشد.

برای تحلیل ترافیک شهری از ویژگی‌های ترافیک از قبیل سرعت لحظه‌ای معبر، سرعت پایه‌ای معبر، سرعت جریان آزاد معبر استفاده می‌شود. اصطلاحا یک معبر پرترافیک نامیده می‌شود اگر سرعت لحظه‌ای عبور مرور از آن کمتر از سرعت پایه‌ای آن باشد. به میزانی که نسبت سرعت لحظه‌ای به سرعت پایه آن معبر بیشتر باشد آن معبر **ترافیک روان** تری را دارد. **سرعت لحظه‌ای** سرعتی که از دستگاه‌های هوشمند برداشت می‌شود می‌باشد و همین‌طور **سرعت پایه** یک معبر حداقل میزان توان یک معبر برای جریان ترافیک می‌باشد. **سرعت جریان آزاد** تعریفی از سرعتی است که اگر در یک معبر به حد آن سرعت رسیده شود اصطلاحا ترافیک را روان می‌گویند.

در این پایان‌نامه از روش‌های حوزه یادگیری عمیق برای تخمین وضعیت ترافیکی در شرایط مختلف استفاده شده است. برای آموزش مدل پیش‌بینی ترافیک از شبکه عصبی پیچشی گراف و شبکه عصبی بازگشتی استفاده شده است که بر اساس داده‌های موقعیت مکانی دستگاه‌های هوشمند (حسگرها، دوربین‌های کنترل ترافیک شهری و ...) جمع آوری گردیده است. این دسته از داده‌ها موقعیت مکانی دستگاه‌هایی می‌باشد که حسگرهای مورد بررسی در اختیار دارند، این داده‌ها شامل سرعت، موقعیت مکانی، زمان برداشت موقعیت به‌صورت تاریخچه‌ای می‌باشند.

در این تحقیق دو مجموعه داده مورد ارزیابی قرار گرفته‌است، مجموعه داده اول مربوط به تاکسی‌های شهر شنزن کشور چین می‌باشد که در آن ۱۵۶ معبر مورد ارزیابی قرار گرفته است و مجموعه داده دوم مربوط به حسگرهای بزرگراهی شهر لس‌آنجلس شامل بیش از ۲۰۰ معبر بررسی شده است. نتایج پیرامون آزمایش روش ارائه شده بر روی این دو مجموعه داده‌ها در این نوشتار مورد بحث قرار گرفته است.

این پایان‌نامه شامل ۴ فصل می‌باشد که در فصل اول به شرحی از حوزه تحقیق، مقدمه‌ای بر مفاهیم و آشنایی با مفاهیم پایه شبکه‌ای عصبی پرداخته شده. فصل دوم مربوط به انواع شبکه‌های عصبی بکار رفته و یا مورد بحث در این پایان‌نامه

به‌طور مختصر می‌باشد. در فصل سوم روش پیش‌بینی ترافیک شهری با استفاده از یادگیری عمیق و تعریف مسئله و داده‌های ورودی تشریح شده است. در نهایت فصل انتهایی این نوشتار یعنی فصل چهارم به بیان نتایج تحقیقات و پیاده‌سازی انجام شده روش پیش‌بینی ترافیک شهری پرداخته است.

مطالب این پایان‌نامه برگرفته از منابع اصلی زیر می‌باشد:

1. Liu, Zhidan, et al. Urban traffic prediction from mobility data using deep learning. IEEE network, 32.4 (2018): 40-46

2. Ling Zhao, et al. T-GCN: A temporal graph convolutional network for traffic prediction IEEE Transactions on Intelligent Transportation Systems, (2019): 3848–3858

فصل ۱

مقدمه، کارهای مرتبط و مفاهیم اولیه

۱-۱ مقدمه

با افزایش نرخ تولید و مهاجرت به مناطق شهری، از شبکه‌های معابر شهری اغلب بیش از ظرفیت مورد نظر استفاده می‌شود که باعث به اجرا در آوردن قوانین و اقدامات خاصی در شهرها می‌گردد. به‌عنوان مثال محدود کردن تردد وسایل نقلیه براساس شماره پلاک خودرو، منجر به کنترل و مدیریت حجم ترافیک در معابر پر تردد خواهد شد. در نتیجه نیاز فوری به پیش‌بینی ترافیک در شهرها برای کاهش تراکم ترافیک، نیازهای زیست محیطی، اقتصادی و اجتماعی موثر بر کیفیت زندگی شهروندان برای حمایت از آینده پایدار مطرح می‌شود.

با در اختیار قراردادن اطلاعات دقیق و به‌موقع وضعیت ترافیکی در معابر شهری، می‌توان مسئله تراکم ترافیکی را بهبود بخشید و از ازدحام ترافیکی در معابر شهری جلوگیری کرد. بر این اساس پیش‌بینی ترافیکی یکی از نکات کلیدی برای آگاه‌سازی در کنترل و مدیریت ترافیک معابر شهری می‌باشد.

همچنین حل مسئله پیش‌بینی بهترین مسیر بین مبدا و مقصد وابسته به پیش‌بینی ترافیکی معابر گوناگون بین نقطه مبدا و نقطه مقصد می‌باشد. پارامتر اصلی حل مسئله بهترین مسیر از بین تمامی مسیرهای پیشنهادی بین مبدا و مقصد، میزان ترافیک فعلی معابر شهری می‌باشد.

این نوشتار بر رویکرد ارائه مدلی در جهت پیش‌بینی ترافیک شهری با استفاده از داده‌های ترافیکی تاریخچه‌ای و شبکه معابر شهری می‌پردازد. در این فصل ابتدا به بررسی کارهای مرتبط صورت گرفته در رابطه با پیش‌بینی ترافیک شهری پرداخته شده و سپس برخی تعاریف و مفاهیم مورد بحث در فصول دیگر نوشتار بیان شده است.

۲-۱ کارهای‌مرتبط

با توجه به اهمیت پیش‌بینی ترافیک و سیستم‌های ترافیک هوشمند معابر شهری، امروزه بررسی‌های متعددی بر روی این سیستم‌ها شکل گرفته است. به‌صورت کلی می‌توان رویکرد پیش‌بینی ترافیک موجود را به دو دسته مدل محور و داده محور تقسیم کرد:

- رویکرد مدل محور: رویکرد مدل محور عمدتا روابط لحظه‌ای و حالت پایدار بین حجم، سرعت و تراکم ترافیک را مورد ارزیابی قرار می‌دهد. روش‌هایی که این رویکرد را دنبال می‌کنند مستلزم مدل سازی جامع و دقیق سیستم بر اساس دانش قبلی می‌باشند. از این دسته روش‌ها میتوان به روش‌های نظریه صف [1] [۱] ، مدل انتقال سلول [2] [۲] ، مدل سرعت ترافیک [3] [۳] ، مدل نمودار بنیادی میکروسکوپی [4] [۴] و دیگر نمونه‌های مشابه اشاره کرد. در حقیقت داده‌های ترافیکی تحت تأثیر عوامل زیادی قرار دارند و بدست آوردن یک مدل کاملا دقیق برای پیش‌بینی ترافیک معابر شهری بسیار دشوار است. مدل‌های موجود نمی‌توانند تغییرات داده‌های ترافیکی را در محیط‌های پیچیده واقعی توصیف کنند. علاوه بر این، ساخت این مدل‌ها به محاسبات بسیار زیاد و پیچیده نیازمند است. [۵]

- رویکرد داده محور: رویکردهای داده محور مبتنی بر رفتار مختلف داده‌ها و استنباط روابط آماری می‌باشند و از این اطلاعات دریافت شده برای پیش‌بینی و ارزیابی وضعیت ترافیکی و ارائه روش پیش‌بینی ترافیک معابر شهری استفاده می‌کنند. این دسته از روش‌ها ویژگی‌های فیزیکی و رفتار پویای سیستم را تحلیل نمی‌کنند و مبتنی بر داده‌های دریافتی می‌باشند، از این رو انعطاف پذیری بالایی دارند. یکی از روش‌های پیشین که می‌توان آن را در دسته رویکرد داده محور قرار داد، روش میانگین تاریخیچه‌ای [5] [۶] می‌باشد که در آن مقدار متوسط حجم ترافیک در دوره‌های قبلی تاریخی یک معبر شهری به عنوان مقدار پیش‌بینی شده برای آن مورد استفاده قرار می‌گیرد. این روش نیاز به فرضی ندارد و محاسبات ساده و بسیار سریعی را دارد، اما نمی‌تواند به خوبی با ویژگی‌های زمانی سازگار باشد و دقت پیش‌بینی پایینی را نتیجه می‌دهد.

با پیشرفت تحقیقات در زمینه پیش‌بینی و توسعه ارزیابی‌های صورت گرفته در روش‌های پیش‌بینی ترافیک معابر شهری، تحول‌هایی صورت گرفت و در نتیجه روش‌های زیادی با دقت پیش‌بینی مناسب‌تر از روش‌های پیشین حاصل شد. این روش‌ها را می‌توان به دو دسته عمده روش‌های پارامتری [6] و روش ناپارامتری [7] تقسیم کرد.

در دسته روش پیش‌بینی پارامتری می‌توان به روش‌های مدل سری زمانی [8] ،مدل رگرسیون خطی [9] و مدل فیلتر کالمن [10] اشاره کرد که از رایج‌ترین روش‌های پیش‌بینی در این دسته می‌باشند. در سال ۱۹۷۶، جورج ادوارد باکس [11] و

[1]Queuing Theory model [2]Cell Transmission model [3]Traffic Velocity model [4]Microscopic Fundamental Diagram Model [5]Historical Average Model [6]parametric model [7]nonparametric mode [8]time series model [9]Linear Regression model [10]Kalman filtering model [11]George E. P. Box

گوئیلیم میریون جنکینز [1] پژوهشگران انگلیسی مدل اتورگرسیو-میانگین متحرک جمع بسته (ARIMA) [2] [۷] را پیشنهاد کردند که پرکاربردترین مدل سری زمانی است.

در سال ۱۹۹۵ پرفسور حامد [3] پژوهشگر اردنی-آلمانی به همراه همکارانش از مدل اتورگرسیو-میانگین متحرک جمع بسته برای پیش‌بینی حجم تردد در شریان‌های شهری استفاده کردند [۸] . آن‌ها برای بهبود دقت پیش‌بینی این مسئله، به تحقیق پیرامون روش‌های مختلفی پرداختند. از جمله ترکیب مدل اتورگرسیو-میانگین متحرک جمع بسته با مدل کوهن [4] [۹] ، زیر مجموعه مدل اتورگرسیو-میانگین متحرک جمع بسته [5] [۱۰] ، اتورگرسیو-میانگین متحرک جمع بسته فصلی [6] [۱۱] و چندین روش دیگر که نتایج دقیق‌تری را بدست آوردند.

مارکو لیپی [7] و همکارانش مدل رگرسیون بردار پشتیبان [8] را با مدل اتورگرسیو-میانگین متحرک جمع بسته مقایسه کردند و دریافتند یک مدل جدید از ترکیب این دو روش [9] در شرایط ازدحام ترافیکی نتایج بهتری را به همراه دارد [۱۲] .

در سال ۲۰۰۴ هوانگ سان [10] و همکارانش مسئله پیش‌بینی فاصله با استفاده از مدل خطی محلی [11] را حل می‌کنند و نتایج بهتری در پیش‌بینی ترافیک معابر شهری با استفاده از مجموعه داده ترافیک معابر شهری در شرایط واقعی به دست می‌آورند [۱۳] .

در سال ۱۹۸۴ ایوائو اوکوتانی [12] و همکارانش از نظریه فیلتر کالمن برای ایجاد مدل پیش‌بینی جریان ترافیک معابر شهری استفاده کردند [۱۴] ، مدل فیلتر کالمن شرایط ترافیکی آینده را بر اساس وضعیت ترافیک لحظه قبل و لحظه فعلی پیش‌بینی می‌کند.

مدل پیش‌بینی‌های روش‌های پارامتری ذکر شده دارای الگوریتم ساده و محاسبات متعادلی می‌باشند، اما نمی‌توانند در شرایطی مانند حوادث راهنمایی رانندگی، تصادفات پیش آمده در معبر و یا وضعیت آب و هوایی تصمیم‌گیری مناسبی داشته باشند و پیش‌بینی دقیقی را نتیجه دهند [۵] .

در روش‌های ناپارامتری مشکلات تداخل رویدادهایی مانند رخداد تصادفات و شرایط آب و هوایی قابل حل می‌باشد. روش‌های ناپارامتری نسبت به روش‌های پارامتری در حل مسئله رخداد وقایع، نتیجه بهتری را منعکس می‌کنند. مدل ناپارامتری تنها به داده‌های تاریخچه‌ای کافی نیاز دارد. مانند مدل چند نزدیک‌ترین همسایگی [13] [۱۵] ، مدل رگرسیون بردار پشتیبان [۱۶، ۱۷] ، مدل منطق فازی [14] [۱۸] ، مدل شبکه بیز [15] [۱۹] و مدل‌های متعدد شبکه عصبی [16] که نتیجه مطلوب‌تری را به همراه دارد [۵].

در سال‌های اخیر با توجه به توسعه سریع یادگیری عمیق و تعدد ابزارهای پیاده‌سازی روش‌های مبتنی بر یادگیری عمیق، مدل‌های شبکه عصبی عمیق مورد توجه بیشتری قرار گرفته است [۲۰، ۲۱، ۲۲] . زیرا این مدل‌ها می‌توانند ویژگی‌های پویای داده‌های ترافیک را بهتر ثبت کرده و بهترین نتایج را در حال حاضر بدست آورند.

تحقیقات و روش‌های فعلی مبتنی بر شبکه عصبی را با توجه به موضوع وابستگی فضایی داده‌های معابر شهری به

---

[1]Gwilym Jenkins [2]Autoregressive Integrate Moving Average Model [3]Mohammad M. Hamed [4]Kohonen-ARIMA [5]subset ARIMA [6]seasonal ARIMA [7]Marco Lippi [8]Support Vector Regression model [9]SARIMA: support vector regression model-ARIMA [10]Hongyu Sun [11]local linear model [12]Iwao Okutani [13]K-Nearest Neighbor model [14]Fuzzy Logic model [15]Bayesian network model [16]Neural Network model

یکدیگر، به دو دسته می‌توان تقسیم کرد. روش‌هایی که فقط مبتنی بر وابستگی زمانی می‌باشند و روش‌هایی که نه‌تنها از وابستگی زمانی استفاده می‌کنند، بلکه از وابستگی فضایی (توپولوژی شبکه معابر شهری) نیز استفاده می‌کنند.

از دسته روش‌هایی که تنها محوریت بررسی وابستگی زمانی داده‌های ترافیک معابر شهری در آن استفاده شده است، می‌توان به تحقیقات بر روی روش‌های شبکه عصبی [۱] برای پیش‌بینی جریان ترافیکی اشاره کرد [۲۳] .

از طرفی شبکه‌های عصبی بازگشتی (RNN) [۲] و انواع آن مانند شبکه عصبی بازگشتی دارای حافظه کوتاه مدت ماندگار (LSTM) [۳] و شبکه بازگشتی با واحدهای دروازه‌ای (GRU) [۴] با توجه به ساختار حافظه آن می‌توانند وابستگی زمانی را یاد گرفته و به نتایج مناسبی برسند [۲۲، ۲۴] .

شبکه‌های عصبی که تنها با استفاده از وابستگی زمانی داده‌ها، آموزش داده می‌شوند و وابستگی مکانی را نادیده می‌گیرند، نمی‌توانند پیش‌بینی دقیقی از ترافیک در تمامی معابر شهری ارائه دهند. استفاده کامل از وابستگی زمانی-مکانی کلید حل مشکلات مسئله پیش‌بینی ترافیک معابر شهری می‌باشد. برای حل مسائل پیش‌بینی ترافیک معابر شهری بسیاری از مطالعات بر روی وابستگی زمانی-مکانی تمرکز کرده‌اند که باعث حاصل شدن نتایج بسیار بهتری شده است. [۵]

از جمله روش‌هایی که از این روش استفاده کرده‌اند می‌توان به روش پشته‌ای خود رمزنگار [۵] [۲۵] ، شبکه باقی‌مانده مکانی-زمانی [۶] [۲۶] ، ترکیب شبکه عصبی پیچشی عمیق [۷] با شبکه عصبی بازگشتی دارای حافظه کوتاه مدت ماندگار [۲۷] ، شبکه حافظه کوتاه مدت پیچشی [۸] [۲۸] و ترکیب شبکه عصبی پیچشی (CNN) [۹] با شبکه عصبی بازگشتی دارای حافظه کوتاه مدت ماندگار (بخش شبکه عصبی پیچشی وظیفه درک وابستگی مکانی و شبکه عصبی بازگشتی برای یادگیری وابستگی زمانی استفاده می‌شود) [۲۹] اشاره کرد.

برای حل مسئله وابستگی مکانی در روش‌های ذکر شده شبکه عصبی پیچشی به عنوان راه حل و بازیگر اصلی رفع این مشکل معرفی شده و باعث پیشرفت بسیار پیش‌بینی ترافیک معابر شهری می‌شود. اما شبکه عصبی پیچشی اساسا برای فضای اقلیدسی مانند تصاویر، شبکه‌های منظم و دارای چارچوب مناسب است و محدودیت‌های آن برای شبکه معابر شهری که ساختار آن مانند گراف می‌باشد و توپولوژی بسیار پیچیده‌ای دارد مناسب نیست و امکان دریافت ویژگی‌های مکانی به صورت مطلوب برای آن میسر نمی‌باشد.

شبکه عصبی پیچشی گراف (GCN) [۱۰] ساختار بسیار مطلوب‌تری نسبت به شبکه عصبی پیچشی دارد که ویژگی مکانی را به عنوان مدل یادگیرنده آموزش می‌بیند [۳۰، ۳۱] .

بر اساس این پیش‌زمینه در این نوشتار یک رویکرد مبتنی بر نگرش داده محور و ناپارامتری مورد ارزیابی قرار می‌گیرد. در این رویکرد با استفاده از دو ویژگی موثر در پیش‌بینی ترافیک شهری، یعنی وابستگی مکانی و وابستگی زمانی، مدلی برای پیش‌بینی دقیق‌تر ترافیک در معابر شهری مورد بررسی قرار گرفته است.

---

[1]Feed Forward Neural Network [2]Recurrent Neural Network [3]Long Short-Term Memory [4]Gated Recurrent Unit [5]SAEs: Stacked AutoEncoders [6]ST-ResNet: Spatio-Temporal Residual Network [7]DCNN: Deep Convolutional Neural Network [8]FCL-Net: Fusion Convolutional Long short-term memory Network [9]Convolutional Neural Network [10]Graph Convolutional Neural network

۱-۳ مفاهیم اولیه

این بخش از نوشتار به تشریح مفاهیم مورد استفاده و عبارات مورد کاربرد پرداخته است. ابتدا توضیحی در مورد مفاهیم اولیه ترافیکی، سپس برخی عبارات، تعاریف و ادبیات مورد استفاده از نظریه گراف در این پایان‌نامه به‌طور مختصر بیان شده است.

۱-۳-۱ ترافیک

به صورت کلی به عبور و مرور جانداران، انسان، وسایل نقلیه و اطلاعات در مسیرهای پیش‌بینی شده، **ترافیک** می‌گویند. ترافیک‌های شهری سه عامل اصلی دارند که این سه عامل عبارت‌اند از:

- انسان
- راه
- وسایل نقلیه

در صورت عدم وجود هر یک از عوامل اصلی ذکرشده، ترافیک شهری تعریف نمی‌شود. ترافیک تبدیل به یکی از مهم‌ترین عوامل مختل کننده حمل و نقل شهری در زندگی ساکنین شهر شده است. زمانی‌که عبور و مرور روان، سریع و بدون خسارت به محیط‌زیست باشد می‌توان گفت که آن شهر دارای ترافیک مطلوبی می‌باشد.

۱-۳-۱-۱ علت ایجاد ترافیک

ترافیک‌ها معمولاً با عوامل مختلفی مانند کمبود شبکه‌های بزرگراهی، تعدد خودروهای تک سرنشین، کمبود شبکه‌های حمل و نقل عمومی مانند مترو و اتوبوس، هوشمند نبودن ترافیک و عدم رانندگی صحیح ایجاد می‌شوند. در برخی از کلان شهرها عدم توزیع مناسب امکانات نیز از دیگر عوامل ایجاد ترافیک است. در بسیاری از شهرهای پیشرفته و پرجمعیت دنیا در نقاط مختلف شهر مراکزی برای برطرف کردن نیازهای گوناگون شهروندان ایجاد شده است. این مراکز شامل فروشگاه‌های مختلف، ادارات، مراکز درمانی، مراکز آموزشی و ارائه سایر خدمات می‌شوند. زمانی که شهروندان به مراکز و موارد موردنیاز خود دسترسی نداشته باشند برای تهیه آن نیاز به سفرهای درون شهری دارند که همین امر می‌تواند باعث افزایش ترافیک شود.

تولید بیش از حد اتومبیل و عدم احداث راه‌های مناسب با آن می‌تواند باعث افزایش ترافیک شهری شود. همچنین مواردی مانند خودروهای تک سرنشین، عدم از رده خارج کردن خودروهای فرسوده و نحوه نادرست استفاده از خودرو و رانندگی با آن از دیگر مواردی است که می‌تواند باعث شود ترافیک در شهرها افزایش یابد.

۱-۳-۱-۲ هزینه‌های تحمیلی ناشی از ترافیک شهری

ترافیک به‌وجود آمده در معابر شهری باعث ایجاد اختلال در زندگی روزمره افراد می‌شود. این اختلال در عبور و مرور وسایل نقلیه هزینه‌هایی پیدا و پنهان را به مردم و دولت‌ها تحمیل می‌کند. برخی از این هزینه‌ها عبارتند از:

- **زمان:** روزانه بیش از میلیون‌ها نفر زمان بسیاری را در ترافیک می‌گذارنند و می‌توان گفت که جمع زمان هدر رفته در ترافیک به میلیون‌ها ساعت می‌رسد. زمان بسیاری که در ترافیک هدر می‌شود، می‌تواند صرف پرداختن به کار و یا استراحت شود.
- **سوخت:** افزایش زمان رانندگی و ماندن در ترافیک به معنی استفاده از میزان بیشتری سوخت است. هر چه ترافیک سنگین‌تر باشد میزان مصرف سوخت بالاتر می‌رود که همین امر می‌تواند باعث شود تا هزینه پرداخت شده برای سوخت افزایش یابد.
- **هزینه روانی:** ماندن در ترافیک برای هیچ فردی خوشایند نیست. ترافیک‌های سنگین باعث می‌شود تا افراد بسیاری با تنش‌های روانی مواجه شوند و همین امر سبب بیشتر شدن میزان درگیری‌های اجتماعی و تنش‌های خانوادگی و کاری می‌شود.
- **هزینه‌های بهداشت و درمان:** از دیگر مضرات ترافیک، آلودگی هوا است. آلودگی هوا به طور مستقیم یا غیرمستقیم هر ساله جان انسان‌های بسیاری را می‌گیرد و باعث بیمار شدن افراد زیادی می‌شود. درمان افراد بیمار می‌تواند هزینه‌های سنگینی را برای فرد و دولت‌ها داشته باشد.

۱-۳-۱-۳ عوامل بررسی ترافیک

- **معبر ترافیک شهری:** ساختار شبکه شهری دارای بزرگراه‌ها، خیابان‌ها، بلوارها و دیگر تقسیم‌بندی‌های موجود می‌شود. تمامی مسیرها و نقاطی که امکان تردد وسایل نقلیه در آن امکان پذیر می‌باشد، به عنوان معبر ترافیکی شهری در این نوشتار مورد ارزیابی قرارگرفته است.
- **حجم ترافیک:** حجم ترافیک عبارت است از تعداد وسایل نقلیه‌ای که در مدت زمان معینی در جهت یا جهات مشخصی از معبر شهری عبور می‌کنند. حجم ترافیک ممکن است در مسیر دو نقطه از شبکه شهری و یا نقطه مشخصی مورد اندازه‌گیری قرار گیرد. مطالعات حجم ترافیک برای جمع آوری داده‌ها در مورد تعداد وسایل نقلیه و یا عابران پیاده‌ای که در بازه زمانی مشخص از نقطه‌ای از شبکه معابر شهری عبور می‌کنند، انجام می‌شود. واحد متداول برای اندازه‌گیری حجم ترافیک تعداد وسیله نقلیه بر ساعت می‌باشد.
- **تراکم (چگالی) ترافیک:** تراکم ترافیک و یا چگالی ترافیک عبارت است از تعداد وسایل نقلیه (عابر پیاده) که در حجم خاصی از معبر شهری فضای جاده را اشغال می‌کنند. واحدهای متداول برای اندازه گیری تراکم ترافیکی مورد

اندازه‌گیری وابسته به واحد حجم در نظر گرفته شده در معبر شهری می‌باشد. واحد متداول تراکم ترافیکی، تعداد وسیله نقلیه بر مایل و یا تعداد وسیله نقلیه بر کیلومتر تعریف می‌شود.

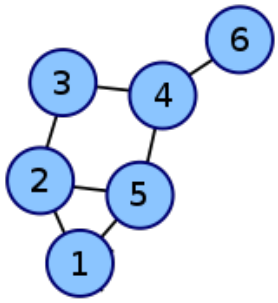
- **جریان ترافیک**: جریان ترافیک تعریفی از سرعت عبور وسایل نقلیه و یا عابر پیاده از نقطه مشخصی در معبر شهر می‌باشد. واحد متداول برای اندازه گیری جریان ترافیک تعداد بر ساعت می‌باشد. یکی از روش‌های اندازه‌گیری جریان ترافیکی، سنجش تعداد عبور و مرور طی ۱۵ دقیقه می‌باشد که با تناسب صحیح، جریان ترافیکی معادل ۴ برابر حجم اندازه گیری در زمان ذکر شده حاصل می‌شود.

- **ظرفیت معبر**: حداکثر میزان وسایل نقلیه که امکان تردد آن‌ها در واحد زمان از معبر میسر است، به‌عنوان ظرفیت ترافیکی معبر تعریف می‌شود. بنابراین تعریف واحد ظرفیت ترافیکی تعداد می‌باشد.

- **سرعت ترافیکی معبر**: سرعت یک مقدار مقیاس پذیر برای سنجش میزان حرکت یک جسم با واحد مسافت بر زمان است. سرعت ترافیک معبر برابر سرعت ممکن برای تردد در این معبر بیان می‌شود. سرعت یک معبر با جریان ترافیک یک معبر رابطه مستقیم دارد.

۱-۳-۲ گراف

ساختار **گراف** برای نمایش اجزا و ارتباط بین آن‌ها می‌باشد. یک گراف تشکیل شده از مجموعه‌ای از اجزا و مجموعه‌ای از ارتباطات بین این اجزا می‌باشد.



شکل ۱-۱: گراف بدون جهت همبند

یک گراف با n رأس [1] (اجزا) و m یال [2] (ارتباط بین اجزا) به‌صورت ذیل تعریف می‌شود:

$$G = (V, E) \tag{۱-۱}$$

که در آن G گراف تعریف شده، $V = \{V_۱, V_۲, ..., V_n\}$ مجموعه‌ای از رأس‌ها و $E = \{E_۱, E_۲, ..., E_m\}$ مجموعه‌ای از یال‌ها می‌باشد. یک یال در گراف باعث اتصال دو رأس به یکدیگر می‌شود.

یک یال را با نمایش زوج مرتب نشان می‌دهیم که شامل دو رأسی‌هایی است که این گراف آن‌ها را به یکدیگر متصل می‌کند. اگر دو رأس گراف با یک یال به هم متصل شوند، اصلاحا این دو رأس را مجاور گویند.

شکل ۱-۱ یک گراف نمونه بدون جهت همبند می‌باشد.این گراف دارای ۶ راس و ۷ یال است. درباره این گراف نمونه دو ویژگی راس‌ها و یال‌ها طبق تعریف برابرند با:

- مجموعه راس‌ها

$$E = \{۱, ۲, ۳, ۴, ۵, ۶\}$$

- مجموعه یال‌ها

$$V = \{(۱, ۲), (۱, ۵), (۲, ۳), (۲, ۵), (۳, ۴), (۴, ۵), (۴, ۶)\}$$

[1] Vertex [2] Edge

جهت گراف: گراف‌ها به‌طور کلی به دو دسته جهت‌دار و بدون جهت تقسیم می‌شوند. یک گراف جهت دار گرافی است که یال‌های آن دارای جهت باشند و به ازای هر یال E_x که گره V_s را به V_d متصل می‌کند الزاما این یال V_d را به V_s متصل نمی‌کند. در مقابل گراف فاقد جهت دارای یال‌های بدون جهت می‌باشد که هر یال آن هر دو گره را به هم متصل می‌کند.

مسیر در گراف: اگر S و D دو رأس در گراف باشند، یک مسیر از S به D، شامل دنباله‌ای از رئوسی است که به ترتیب از S شروع و به D ختم می‌شود. هر دو رأس متوالی در این دنباله می‌بایست مجاور باشند. طول یک مسیر برابر است با تعداد یال‌های موجود در آن مسیر.

دور در گراف: یک گراف دارای دور یا حلقه گفته می‌شود درصورتی‌که حداقل یک مسیر را شامل شود که نقطه ابتدایی و نقطه انتهایی یکسان باشد.

همبندی در گراف: گراف G را همبند می نامیم هرگاه بین هر دو رأس متمایز در این گراف حداقل یک مسیر وجود داشته باشد. یعنی تمامی رئوس برای یکدیگر قابل دسترس باشند. در صورت عدم وجود مسیر بین حداقل دو رأس متمایز گراف ناهمبند نامیده می‌شوند.

درجه رأس‌های گراف: به تعداد یال‌های متصل به یک رأس، درجه آن رأس گفته می‌شود. به عبارت دیگر درجه یک رأس تعداد همسایگی‌های مستقیم یک رأس را بیان می‌کند.

گراف وزن‌دار: گرافی است که به هر یال آن عددی نسبت داده شده‌است که وزن آن یال می‌باشد. یا به رأس آن عددی نسبت داده شده است. وزن یال می‌تواند نشان دهنده هزینه، مسافت، زمان و یا هر مشخصه دیگری از یال باشد. بعضی از نویسنده‌ها به آن گراف شبکه‌ای می‌گویند. از گراف وزن‌دار برای حل مسائلی چون فروشنده دوره گرد استفاده می‌شود. در گراف‌های ساده و بدون وزن وزن تمامی یال‌ها برابر یک می‌باشد.

ماتریس درجات گراف: ماتریس درجات گراف [1] D ، ماتریسی قطری (فقط در قطر اصلی ماتریس مقادیر غیر صفر یافت می‌شود) با ابعاد $N \times N$ می‌باشد که N تعداد راس‌های این گراف است و عنصر $D_{i,i}$ نمایانگر تعداد یال‌های متصل به این رأس می‌باشد. به عنوان مثال شکل ۱-۱ حاوی ماتریس درجات زیر می‌باشد:

[1] Degree matrix

$$D = \begin{pmatrix} ۲ & ۰ & ۰ & ۰ & ۰ & ۰ \\ ۰ & ۳ & ۰ & ۰ & ۰ & ۰ \\ ۰ & ۰ & ۲ & ۰ & ۰ & ۰ \\ ۰ & ۰ & ۰ & ۳ & ۰ & ۰ \\ ۰ & ۰ & ۰ & ۰ & ۳ & ۰ \\ ۰ & ۰ & ۰ & ۰ & ۰ & ۱ \end{pmatrix}$$

۱-۳-۲-۱ انواع‌گراف

گراف‌ها به خاطر داشتن ویژگی‌های مربوط به رأس‌ها و یال‌های آن به انواع مختلفی تقسیم می‌شوند. در ادامه به توضیح چندین مورد مطرح از تمامی انواع گراف‌ها پرداخته شده‌است.

گراف تهی: گراف G، گراف تهی گفته می‌شود درصورتی‌که هیچ‌گونه یالی در آن وجود نداشته باشد. به تعریف دیگر هیچ دو گره‌ای در آن مجاور نباشند.

گراف کامل: گراف G یک گراف کامل است، درصورتی‌که تمامی رأس‌های آن با یکدیگر مجاور باشند. در گراف کامل تعداد یال‌ها در حداکثر تعداد خود می‌باشد، یعنی اگر m تعداد رأس‌ها باشد،برای n تعداد یال‌ها داریم

$$m = \frac{n \times (n - ۱)}{۲} \qquad (۱-۲)$$

درخت: از کاربردی‌ترین انواع گراف می‌توان به درخت اشاره کرد. درخت‌ها یک گراف ساده، همبند، بدون جهت و فاقد دور می‌باشند. خاصیت همبندی و فاقد دور بودن درخت باعث بوجود آمدن رابطه دقیق بین تعداد یال‌ها و تعداد رأس‌ها می‌شود. تعداد یال‌ها در درخت یک واحد کمتر از تعداد رئوس می‌باشد که داریم:

$$m = n - ۱ \qquad (۱-۳)$$

که در آن m تعداد یال‌های گراف و n تعداد رأس‌های گراف می‌باشد.

۱-۳-۲-۲ نمایش و ذخیره‌سازی گراف

نمایش گراف در واقع همان روش‌های ذخیره‌سازی گراف در کامپیوتر است، به عبارت دیگر با توجه به محدودیت‌هایی که در کامپیوتر وجود دارد، امکان ذخیره‌سازی و نمایش گراف را به شکلی که بر روی کاغذ رسم شده وجود ندارد. همچنین نمی‌توان با گفتن ویژگی‌های تصویری، آن را به راحتی ذخیره و شبیه سازی کرد.

از این‌رو دو روش مطرح در ذخیره‌سازی گراف به‌وجود آمده است که به ذخیره سازی گراف و استخراج ویژگی‌های این گراف کمک کند. در هر دو روش ذخیره‌سازی ابتدا می‌بایست فرآیند شماره گذاری راس‌ها را در پیش گرفت. یعنی به هر رأس یک شماره منحصر به فرد نسبت داده شود تا امکان متمایز کردن رأس‌ها از یکدیگر برقرار باشد. از این‌رو گراف‌های یک شکل، نمایشی متفاوت دارند. به‌دلیل این‌که آرایه‌ها و آدرس‌ها در کامپیوتر از صفر شروع می‌شوند معمولاً شماره‌گذاری راس‌ها از صفر [1] آغاز می‌شود. به این فرآیند شماره‌گذاری ماتریس یا برچسب زدن ماتریس می‌گویند.

این دو روش ذخیره سازی عبارت‌اند از:

ماتریس مجاورت [2] : در ریاضی گسسته و دانش رایانه، ماتریس مجاورت یال‌های میان گره‌های گراف را نشان می‌دهد. به سخنی دیگر، ماتریس مجاورت نشان می‌دهد که آیا جفت‌گره‌ها با یالی مجاور یکدیگرند یا خیر. در گراف‌های وزن دار، این ماتریس وزن یال میان جفت‌گره‌ها را نمایش می‌دهد. برای گراف G با n گره، اندازه ماتریس مجاورت $n \times n$ است. درایه a_{ij} ماتریس مجاورت برای گرافی ساده نشان دهنده وجود و یا عدم وجود یال میان دو گره v_i و v_j است برای وجود یال از عدد ۱ و وقتی دو رأس مجاور نباشند از عدد ۰ استفاده می‌شود. در گرافی وزن‌دار، درایه a_{ij} برابر است با وزن یالی که دو گره v_i و v_j را به هم پیوند می‌زند. دو گراف متفاوت دارای ماتریس مجاورت متفاوت می‌باشند.

لیست مجاورت [3] : لیست مجاورت نحوه دیگری از نمایش و ذخیره‌سازی یک گراف در کامپیوتر می‌باشد. برای ماتریس G با n رأس و m یال، لیستی با اندازه m است. در گراف ساده هر آیتم از این لیست مجاورت شامل یک زوج مرتب که نشان‌دهنده دو راسی است که با یکدیگر مجاور هستند. در گراف وزن‌دار هر آیتم از این لیست شامل سه آیتمی می‌باشد. دو آیتم از آن نشان دهنده دو یالی است که با هم مجاورند و آیتم سوم نمایانگر وزن یال متصل کننده آن‌ها می‌باشد.

۱-۳-۲-۳ پیمایش گراف

پیمایش گراف به معنی بازدید از تمامی رأس‌های گراف به روشی مشخص می‌باشد. الگوریتم‌های پیمایش متفاوتی نسبت به کارایی آن وجود دارد که در ادامه به طور مختصر به تشریح چند نمونه از این پیمایش‌ها پرداخته شده‌است.

---

[1] zero base [2] Adjacency Matrix [3] Adjacency List

جستجوی اول سطح (BFS) [1] **:** جستجوی اول سطح یکی از ساده‌ترین الگوریتم‌ها برای جست و جوی گراف و نوعی پایه برای بسیاری از الگوریتم‌های مهم گراف است. گراف $G = (V, E)$ و رأس مبدأ مشخص s داده شده‌است، جستجوی اول سطح به صورت اصولی یال‌هایG را برای پیدا کردن رئوسی که از s قابل دستیابی اند، مرور می‌کند. این الگوریتم فاصله (کمترین تعداد یال‌ها) هر رأس قابل دستیابی از s را محاسبه می‌کند. برای هر رأس v قابل دستیابی از s مسیر واقع در درخت جستجوی اول سطح از s به v متناظر با کوتاه‌ترین مسیر [2] از s به v در G می‌باشد، به عبارت دیگر مسیری شامل کمترین تعداد یال. الگوریتم برای هر دو گراف جهت دار و بدون جهت اعمال می‌شود.

علت این که این الگوریتم، جستجوی اول سطح نامیده شده، این است که یک مرز بین رئوس کشف شده و رئوس کشف نشده به طور یکنواخت در راستای سطح مرز عبوری توسعه می‌دهد. به عبارت دیگر همه رأس‌ها ی با فاصله k از s را قبل از این که رئوس با فاصله $k + l$ را کشف کند، کشف می‌کند.

الگوریتم ۱-۱ الگوریتم جستجوی اول سطح BFS

Input: $G = (V, E)$, v(a vertex of G).
Output: depends on the application compute result of BFS.

```
1: begin
2: mark v
3: put v in the queue q
4: while q is not empty do
5:    remove first vertex w from the q
6:    for all edge (w, x) such that x is unmarked do
7:       mark x
8:       put x in the q
9:    end for
10: end while
11: end
```

به عنوان نمونه برای گراف مشخص شده در شکل ۱-۱ با اجرای عملیات پیمایش اول سطح از رأس شماره ۱، ترتیب پیمایش به صورت $1, 2, 5, 3, 4, 6$ خواهد بود.

جستجوی اول عمق (DFS) [3] **:** جستجوی اول عمق همانند جستجوی اول سطح یکی از کاربردی‌ترین الگوریتم‌ها برای پیماش گراف و نوعی پایه برای بسیاری از الگوریتم‌های مهم گراف است. گراف $G = (V, E)$ و رأس مبدأ مشخص

[1] Breadth First Search [2] Shortest Path [3] Depth First Search

s داده شده‌است، جستجوی اول عمق به صورت اصولی یال‌هایG را برای پیدا کردن رئوسی که از s قابل دستیابی اند، مرور می‌کند. این الگوریتم بر خلاف الگوریتم اول سطح جستجو را عمقی انجام داده و تا عمیق ترین گره قابل دسترس برای گره ممکن پیش‌روی و پس از مشاهده و پیمایش در عمق نسبت به بازگشت به گره‌های بالاتر اقدام می‌کند. این پیمایش به صورت بازگشتی این جستجوی عمیق را انجام می‌دهد.

الگوریتم ۱-۲ الگوریتم جستجوی اول عمق DFS

Input: $G = (V, E)$, v(a vertex of G).
Output: depends on the application compute result of DFS.

```
1: mark v
2: add v in the stack s
3: while l is not empty do
4:     remove first vertex w from s
5:     for all edge (w, x) such that x is unmarked do
6:         mark x
7:         put x in the s
8:     end for
9: end while
10: end
```

به عنوان نمونه برای گراف مشخص شده در شکل ۱-۱ با اجرای عملیات پیمایش اول عمق از رأس شماره ۱، ترتیب پیمایش به صورت $1, 2, 3, 4, 5, 6$ خواهد بود.

۱-۴ شبکه‌عصبی

شبکه‌های عصبی مصنوعی یا به زبان ساده‌تر **شبکه‌های عصبی**، سیستم‌ها و روش‌های محاسباتی نوین برای یادگیری ماشینی، نمایش دانش و در انتها اعمال دانش به دست آمده در جهت بیش‌بینی پاسخ‌های خروجی از سامانه‌های پیچیده هستند. ایده اصلی این گونه شبکه‌ها تا حدودی الهام‌گرفته از شیوه کارکرد سیستم عصبی زیستی برای پردازش داده‌ها و اطلاعات به منظور یادگیری و ایجاد دانش می‌باشد. عنصر کلیدی این ایده، ایجاد ساختارهایی جدید برای سامانه پردازش اطلاعات است.

این سیستم از شمار زیادی عناصر پردازشی فوق‌العاده به‌هم پیوسته با نام نورون تشکیل شده است، که برای حل یک مسئله با هم هماهنگ عمل می‌کنند و توسط سیناپس‌ها (ارتباطات الکترومغناطیسی) اطلاعات را منتقل می‌کنند. در این شبکه‌ها اگر یک سلول آسیب ببیند بقیه سلول‌ها می‌توانند نبود آن را جبران کرده، و نیز در بازسازی آن سهیم باشند. این شبکه‌ها قادر به یادگیری‌اند. یادگیری در این سیستم‌ها به صورت تطبیقی صورت می‌گیرد، یعنی با استفاده از مثال‌ها وزن سیناپس‌ها به گونه‌ای تغییر می‌کند که در صورت دادن ورودی‌های جدید، سیستم پاسخ درستی تولید کند.

شبکه عصبی یک مدل یادگیری ماشین است به این معنا که با مرور مثال‌ها، بدون نیاز به الگوریتم‌های پیچیده آن را می‌آموزد. به طور معمول، یک شبکه عصبی از چندین لایه تشکیل شده که هر لایه در درون خود از چندین نورون (گره، راس) تشکیل شده است. ارتباطاتی میان نورون‌های یک لایه با نورون‌های لایه‌های دیگر وجود دارد؛ که به دو دسته ارتباطات ورودی و خروجی تقسیم می‌شوند. ارتباطات ورودی به نورون بر مقدار خروجی نورون تأثیر گذار است. مقادیر ارتباطات ورودی یک نورون پس از محاسبات اوزان و بایاس به تابع فعال‌ساز داده می‌شود و موثر بر مقدار خروجی یک نورون و لایه‌های بعدی در ارتباط با نورون می‌باشد.

۱-۴-۱ لایه‌های شبکه عصبی

شبکه عصبی از سه لایه اصلی تشکیل شده است. هر لایه دارای نورون‌های مشخص می‌باشد که وزن و بایاس خاص خود را دارند. در فرآیند آموزش این شبکه این ویژگی‌ها (وزن‌ها و بایاس‌ها) بروزرسانی می‌شوند تا دقت تشخیص به بیشترین مقدار خود برسد.

۱-۴-۱-۱ لایه ورودی

لایه ورودی یک شبکه عصبی از نورون‌های مصنوعی ورودی تشکیل شده است و داده‌های اولیه را برای پردازش بیشتر توسط لایه‌های بعدی نورون‌های مصنوعی وارد سیستم می‌کند. لایه ورودی آغاز جریان کار شبکه عصبی مصنوعی است. هیچ محاسباتی در هیچ یک از گره‌های ورودی انجام نمی‌شود بلکه آن‌ها فقط اطلاعات را به گره‌های لایه پنهان بعدی منتقل می‌کنند. تعداد نورون‌ها در لایه ورودی وابسته به تعداد ویژگی‌های موجود در مجموعه ویژگی‌ها می‌باشد. به عنوان مثال در

یک تصویر با ابعاد ۱۵ × ۱۵ پیکسل، با فرض سیاه و سفید بودن تصویر(مقدار هر پیکسل برابر عددی بین ۰ تا ۲۵۵)، تعداد نورون‌های لایه ورودی برابر ۲۲۵ = ۱۵ × ۱۵ می‌باشد.

۱-۴-۱-۲ لایه پنهان

در ساختار شبکه‌های عصبی، لایه‌های پنهان (مخفی) لایه‌هایی هستند که بین لایه ورودی و لایه خروجی قرار دارند. این لایه با اعمال وزن و بایاس مخصوص، خروجی نورون‌ها را به تابع فعال‌ساز هدایت می‌کند. در واقع لایه‌های پنهان تغییرات غیر خطی شبکه عصبی را بر ورودی‌های وارد شده به یک شبکه عصبی انجام می‌دهد.

۱-۴-۱-۳ لایه خروجی

آخرین لایه یک شبکه عصبی مصنوعی یک لایه خروجی می‌باشد که وظیفه محاسبه نتیجه محاسبات تک تک اجزای یک شبکه عصبی به نورون‌های این لایه داده‌شده است. تعداد نورون‌ها در لایه خروجی وابسته به صورت مسئله مطرح شده می‌باشد. در مسائل طبقه‌بندی، تعداد نورون‌های لایه خروجی برابر تعداد دسته‌های موجود می‌باشد، هر نورون خروجی معادل یک دسته است. در مسئله‌های پیش‌بینی، لایه خروجی تنها یک نورون دارد که این نورون وظیفه محاسبه به عنوان آخرین نورون این شبکه عصبی را بر عهده دارد.

۱-۴-۲ توابع فعال‌ساز

تابع فعال‌سازی [1] تابعی است که به یک شبکه عصبی مصنوعی اضافه می شود تا به شبکه کمک کند الگوهای پیچیده را در داده ها بیاموزد. هنگام مقایسه با یک مدل مبتنی بر نورون که در مغز ما وجود دارد ، عملکرد فعال‌سازی در پایان تصمیم می‌گیرد که چه چیزی به نورون بعدی منتقل شود. ورودی و خروجی این تابع یک اسکالرمی‌باشد. فرض شود خروجی تابع فعال‌ساز برای یک نورون مقدار y باشد، با توجه به مقادیر وزن و بایاس مربوط به نورون در لایه داریم:

$$y = F\Big(\sum_i W_i x_i + b\Big) \qquad (۴-۱)$$

که در آن W وزن نورون‌های ورودی، بردار x ورودی نورون‌های متصل، b مقدار بایاس نورون در لایه و F تابع فعال‌ساز است.

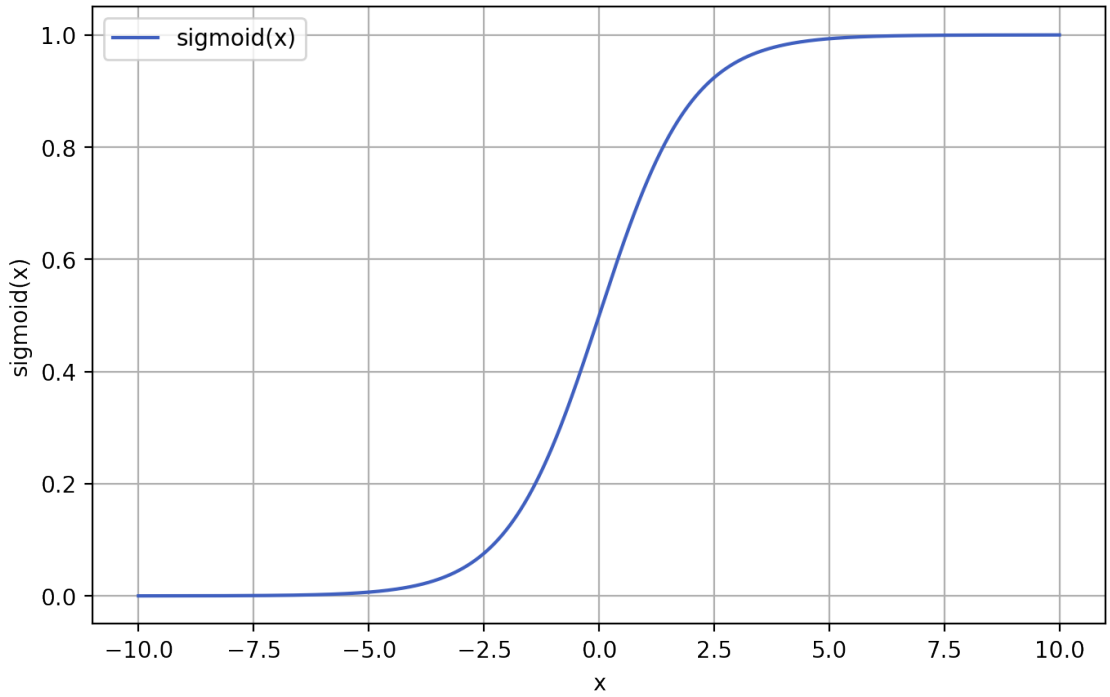
تابع فعال‌ساز با توجه به اینکه خروجی آن در محدوده مشخصی است، از بزرگ شدن خیلی زیاد و یا کوچک شدن بیش‌از حد جلوگیری می‌کند. در ادامه به برررسی برخی توابع فعال‌ساز معروف پرداخته‌شده است.

[1] Activation Function

۱-۲-۴-۱ تابع فعال‌ساز سیگموئید (sigmoid)

سیگموئید یک تابع فعال‌ساز غیر خطی می‌باشد که در لایه‌های خروجی شبکه‌های عصبی بسیار پرکاربرد است. برد این تابع محدود در بازه (۰, ۱) و دامنه آن شامل تمامی اعداد حقیقی است. به‌طور متداول تابع سیگموید با نماد S نشان داده می‌شود:

$$sigmoid(x) = \frac{۱}{۱ + e^{-x}} \quad (۵-۱)$$



شکل ۱-۲: تابع فعال‌ساز سیگموئید (sigmoid)

شکل ۱-۲ نمایشی از تابع فعال‌ساز غیرخطی سیگموئید (sigmoid) است که در بازه دامنه محدود نمایش داده شده است. این تابع در اعداد مثبت بزرگ خروجی نزدیک به ۱ و در اعداد منفی بزرگ خروجی تقریباً ۰ را نتیجه می‌دهد.

۱-۲-۴-۲ تابع فعال‌ساز بیشینه‌هموار (softmax)

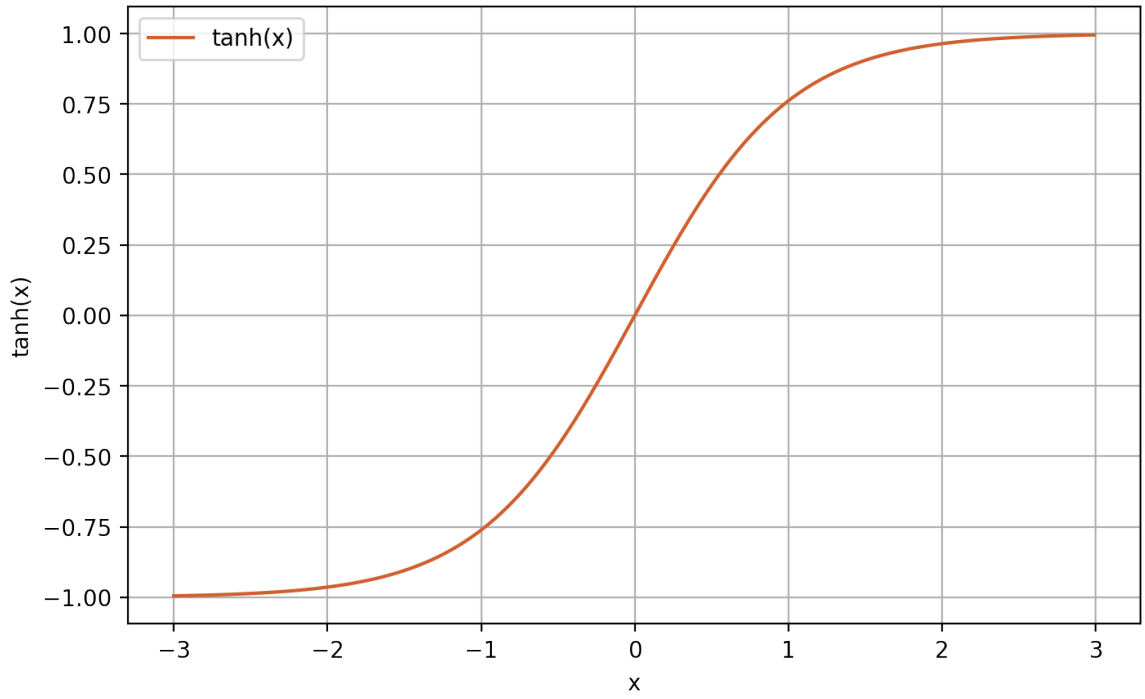
تابع فعال‌ساز بیشینه‌هموار نوع دیگری از توابع فعال‌ساز است که در شبکه عصبی مصنوعی استفاده می‌شود. تابع بیشینه‌هموار همواره یک بردار K تایی از اعداد حقیقی مانند X را به عنوان ورودی دریافت و خروجی آن در بازه اعداد حقیقی با برد محدود بین (۰, ۱) تولید می‌کند.

$$softmax(X_j) = \frac{e^{X_j}}{\sum_{j=1}^{K} e^{X_j}} \tag{۶-۱}$$

۱-۴-۲-۳ تابع فعال‌ساز تانژانت هذلولی (tanh)

ساختار تانژانت هذلولی (مماس هذلولی) به سبب برد محدود آن دارای ویژگی خاصی می‌باشد که در شبکه‌های عصبی مصنوعی کاربر زیادی دارد. این تابع با دامنه تمامی اعداد حقیقی مانند تابع سیگموئید بردی در بازه $(-۱, ۱)$ دارد.

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{۷-۱}$$



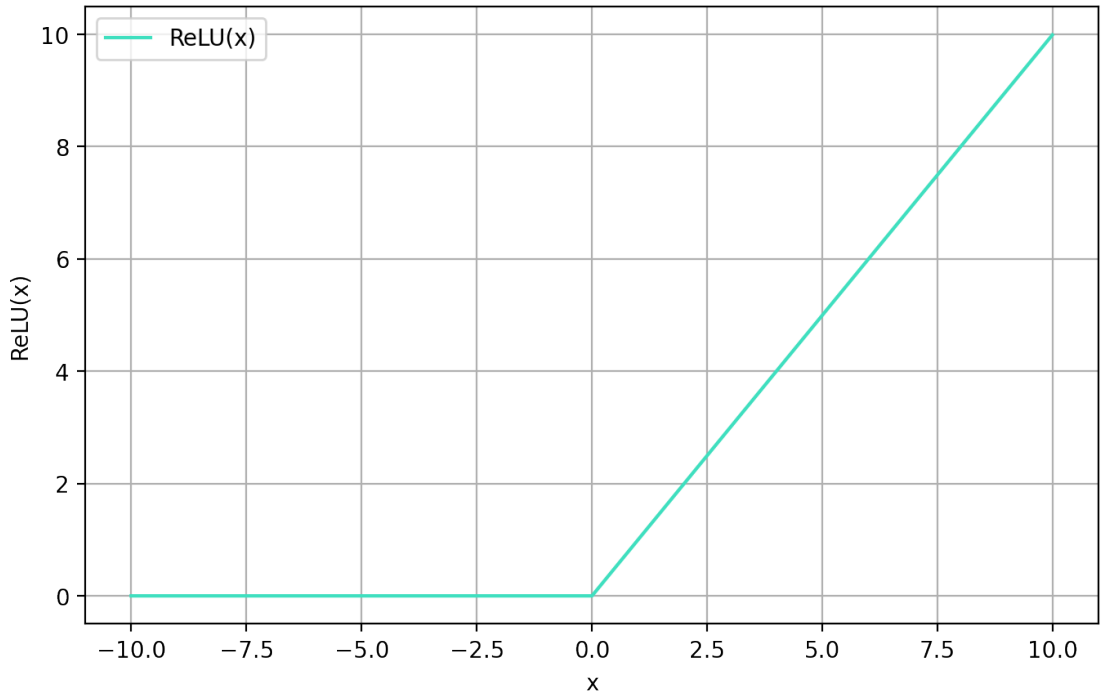
شکل ۱-۳: تابع فعال‌ساز تانژانت هذلولی (tanh)

شکل ۱-۳ نمایشی از تابع فعال‌ساز غیرخطی (tanh) است که در بازه دامنه محدود نمایش داده شده است. مطابق رابطه ۱-۷ ، تابع تانژانت هذلولی در اعداد مثبت بزرگ $(x \to +\infty)$ خروجی نزدیک به ۱ و در اعداد منفی بزرگ $(x \to -\infty)$ خروجی تقریباً -۱ را نتیجه می‌دهد.

۱-۴-۲-۴ تابع فعال‌ساز خطی اصلاح‌شده (ReLU)

تابع فعال‌ساز ReLU تابعی تقریباً خطی است که ویژگی‌های مدل‌های خطی را حفظ و بهینه‌سازی مدل‌های خطی را با روش‌های گرادیان کاهشی [1] آسان می‌کند. شکل ۱-۴ نمایشی از تابع فعال‌ساز خطی ReLU است که در بازه دامنه محدود نمایش داده شده است.

$$ReLU(x) = \begin{cases} ۰ & x < ۰ \\ x & x \geq ۰ \end{cases}$$



شکل ۱-۴: تابع فعال‌ساز ReLU

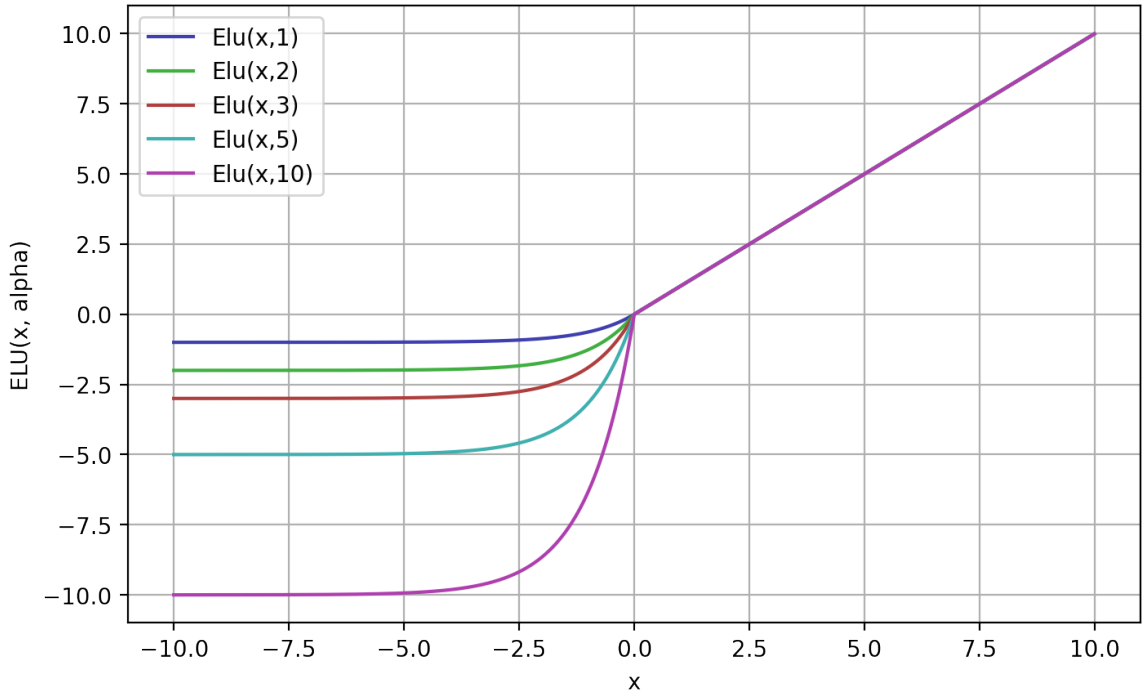
۱-۴-۲-۵ تابع فعال‌ساز (ELU)

تابع فعال‌ساز ELU [2] تابعی اصلاح شده از ساختار تابع فعال‌ساز ReLU می‌باشد. در ساختار تابع فعال‌ساز ReLU در قسمت منفی دارای شیب ۰ است و اصطلاحا مرگ نورون‌ها را به همراه دارد. در تابع ELU به اعداد منفی شیب غیر صفر را نسبت می‌دهد. با تنظیم پارامتر α شیب صفر در اعداد منفی حذف خواهد شد.

[1] Gradient Descent [2] Exponential Linear Units

$$ELU(x,\alpha) = \begin{cases} \alpha(e^x - ۱) & \text{x}< ۰ \\ x & \text{x}\geq ۰ \end{cases}$$

شکل ۱-۵ نمایشی از تابع فعال‌ساز ELU با پارامتر α برابر ۱، ۲، ۳، ۵ و ۱۰ است که در بازه دامنه محدود نمایش داده شده است.



شکل ۱-۵: تابع فعال‌ساز ELU

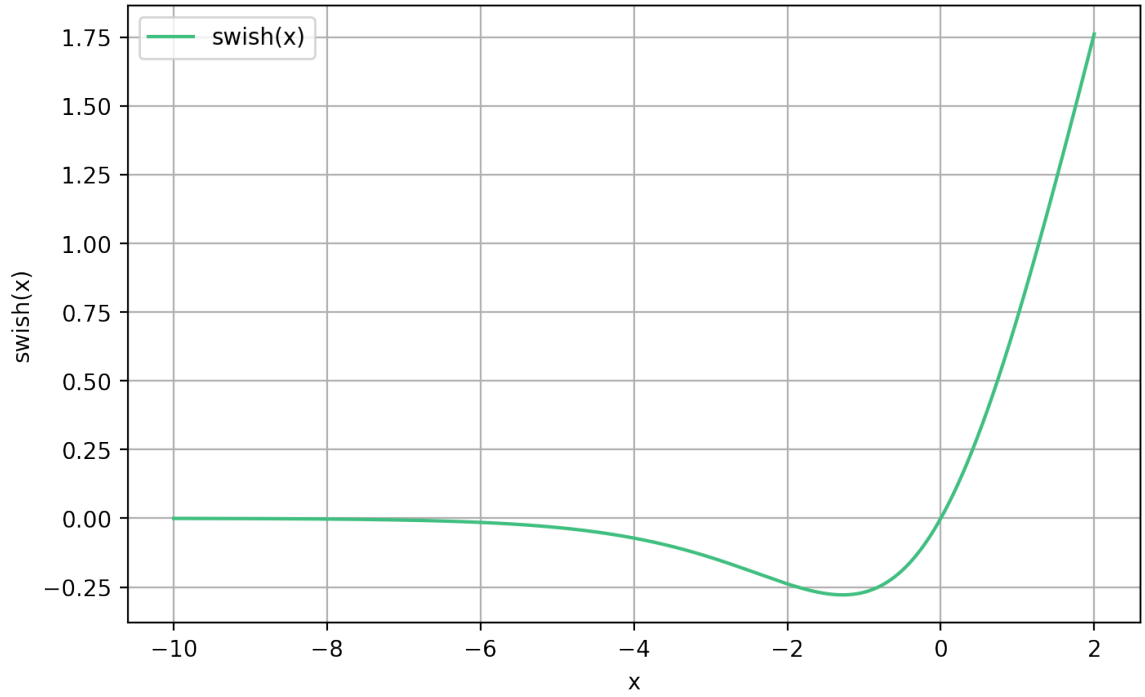
۱-۴-۲-۶ تابع فعال‌ساز (swish)

تابع فعال‌ساز swish نیز مانند تابع فعال‌ساز ELU تابعی اصلاح شده از ساختار تابع فعال‌ساز ReLU می‌باشد. این تابع فعال‌ساز اولین بار توسط محققان داده شرکت گوگل با هدف اصلاح ساختار ReLU ارائه شد. تابع فعال‌ساز swish دارای کران پایین است اما کران بالا ندارد و در رابطه آن از تابع فعال ساز سیگموئید (۱-۵) استفاده می‌شود.

$$swish(x) = x \times sigmoid(x) \qquad (۱-۸)$$

شکل ۱-۶ نمایشی از تابع فعال‌ساز swish در دامنه محدود می‌باشد. تابع به ازای مقادیر منفی بزرگ مقدار صفر دارد

اما در بازه تغییرات بین صفر و اعداد منفی آهنگ تغییرات کمی دارد که از مزایا این تابع فعال‌ساز محسوب می‌شود.



شکل ۱-۶: تابع فعال‌ساز swish

۱-۴-۳ مفاهیم شبکه‌عصبی

در این بخش به تعریف برخی مفاهیم ذکر نشده در بخش‌های دیگر پرداخته‌شده است:

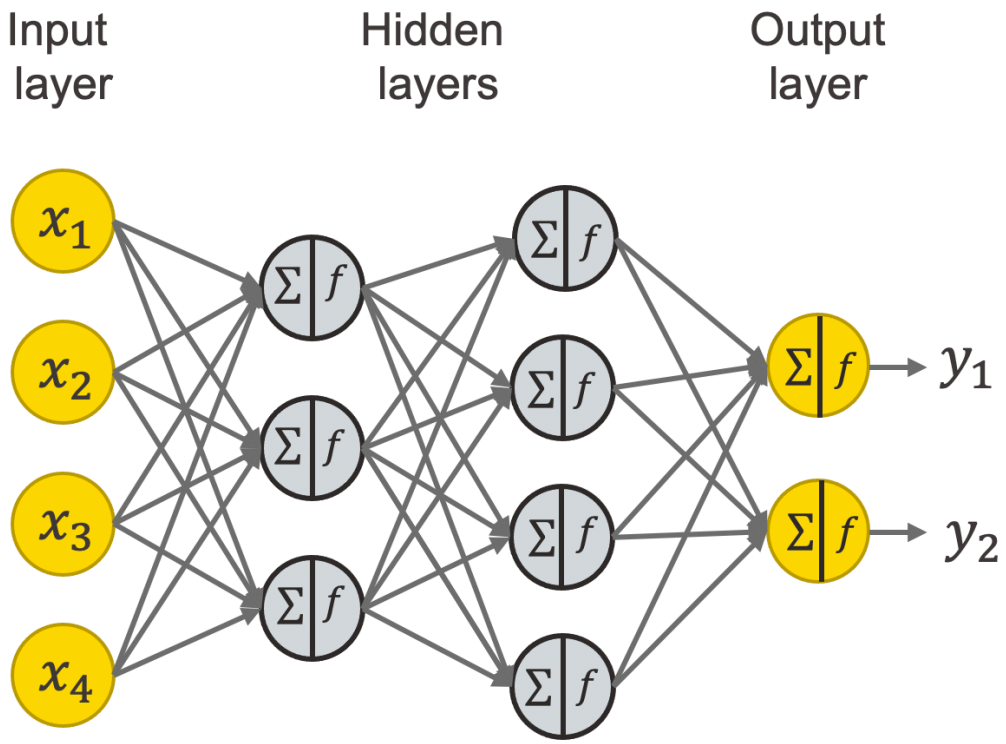
۱-۴-۳-۱ شبکه چند لایه پرسپترون (MLP)

شبکه عصبی با چندین لایه پنهان، شبکه عصبی چند لایه **پرسپترون (MLP)** [1] گفته می‌شود. شبکه عصبی چند لایه پرسپترون یک راه‌حل یادگیری ماشین عمومی در دهه ۱۹۸۰ بود، که کار آن پیدا کردن کاربرد در زمینه‌های مختلف مانند بازشناسی گفتار، بینایی رایانه ای، و نرم‌افزار ترجمه ماشینی بود اما از آن پس ازآن با رقابت قوی‌تری از ماشین‌های بردار پشتیبان بسیار ساده‌تر (و مرتبط) روبرو شدند.

این شبکه عصبی با داشتن ساختار چند لایه، نتیجه دسته‌بندی و یا پیش‌بینی ورودی را محاسبه می‌کند. در نورون‌های هر لایه یک فعال‌ساز ورودی‌ها را محاسبه و سپس به لایه بعدی انتقال می‌دهد. لایه اول این شبکه ورودی را دریافت و لایه آخر این شبکه حاصل را نتیجه می‌دهد. شکل ۱-۷ یک شبکه چند لایه پرسپترون است که در لایه اول ۴ نورون ورودی و در

[1] Multilayer Perceptron

لایه‌های پنهان دوم و سوم به ترتیب ۳ و ۴ نورون موجود می‌باشد. لایه آخر این شبکه که حاصل را مورد ارزیابی قرار می‌دهد دارای ۲ نورون خروجی می‌باشد.



شکل ۱-۷: شبکه چند لایه پرسپترون

۱-۴-۳-۲ تابع هزینه

تابع هزینه [1] یا **تابع خطا** [2] تابعی برای اندازه‌گیری میزان خطای خروجی شبکه‌عصبی می‌باشد. برخی از معروف‌ترین توابع هزینه عبارت اند از :

- میانگین مجموع مربعات خطا (MSE) :

$$F_{MSE}(y, \hat{y}) = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n} \qquad (۹-۱)$$

- میانگین قدر مطلق خطا (MAE) :

$$F_{MAE}(y, \hat{y}) = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n} \qquad (۱۰-۱)$$

[1] cost function [2] loss function

- جذر میانگین مربعات خطا (RMSE) :

$$F_{RMSE}(y, \hat{y}) = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}} \qquad (۱۱-۱)$$

۱-۴-۳-۳ تابع بهینه‌ساز

هنگامی‌که خروجی مدل توسط تابع هزینه مشخص می‌شود، با توجه به هدف مسئله که کم کردن خطا می‌باشد، این تابع بهینه‌ساز مقادیر را به‌گونه‌ای اصلاح می‌کند که مقدار خط نسبت به مرتبه قبل کمتر شود. پس از محاسبه وزن‌ها و بایاس برای نورون‌های شبکه، تابع بهینه ساز نسبت به به‌روزرسانی این اوزان و بایاس برای کم کردن خطای ناشی از شبکه می‌پردازد.

۱-۴-۳-۴ نرخ یادگیری

نرخ یادگیری [1] بیانگر سرعت (گام) بروزرسانی وزن‌ها است که می‌تواند به مقداری ثابت یا سازگارشونده تغییر کند. محبوب ترین روش حال حاضر آدام [2] نام دارد، متدی که نرخ یادگیری را در حین فرآیند آموزش تنظیم می‌کند. باید میزان یادگیری را با دقت زیادی انتخاب کرد زیرا نه باید خیلی زیاد باشد که موجب عدم همگرایی شبکه شود و نه خیلی کم باشد که همگرایی شبکه برای مدت زیادی طول بکشد. آدام یکی از الگوریتم‌های بهینه‌سازی است که می‌تواند به جای روش کلاسیک گرادیان نزولی تصادفی برای به‌روزرسانی وزن‌های شبکه بر اساس داده‌های آموزشی استفاده شود [۳۲].

۱-۴-۳-۵ پس انتشار خطا

پس انتشار خطا [3] یا انتشار معکوس روشی برای بروزرسانی وزن ها با توجه به خروجی واقعی و خروجی مورد انتظار در شبکه عصبی است. انتشار معکوس، حرکت شبکه به سمت عقب است، خطا همراه با گرادیان از لایه بیرون به داخل لایه‌های پنهان برگشت می‌یابد و وزن‌ها به روز می‌شوند.

۱-۴-۳-۶ انتشار روبه‌جلو

انتشار رو به جلو [4] به حرکت ورودی از طریق لایه‌های پنهان به لایه‌های خروجی اشاره دارد. در انتشار به جلو، اطلاعات در یک جهت واحد به جلو حرکت می‌کنند. لایه ورودی، ورودی لایه‌های مخفی را تأمین می‌کند و سپس خروجی تولید می‌شود.

[1] Learning rate [2] Adam [3] Back Propagation [4] Forward Propagation

۱-۴-۳-۷ محو/انفجار گرادیان

محو شدگی گرادیان [1] و همینطور انفجار گرادیان [2] ، مشکلی است که حین آموزش شبکه های عصبی مصنوعی با استفاده از روش های یادگیری مبتنی بر گرادیان رخ می‌دهد. در روش‌های آموزش شبکه عصبی به منظور بروزرسانی پارامترهای شبکه عصبی از گرادیان استفاده می‌شود. هر پارامتر با توجه به میزان اثری که در نتیجه نهایی شبکه داشته است مورد تغییر قرار می‌گیرد. این مهم با استفاده از مشتق جزئی تابع خطا نسبت به هر پارامتر در هر تکرار فرآیند آموزش صورت می‌پذیرد. مشکل محو شدگی اشاره به این مساله دارد که مقادیر گرادیان ها با حرکت به سمت ابتدای شبکه رفته رفته به حدی کوچک می‌شوند که تغییرات وزن بصورت ناچیزی صورت می‌گیرد و به این علت فرآیند آموزش بشدت کند می‌شود و در حالات شدیدتر این مساله باعث متوقف شدن فرآیند آموزش می‌گردد. این مساله عموماً به‌واسطه تعداد لایه‌های زیاد شبکه رخ می‌دهد.

۱-۴-۳-۸ بیش برازش

بیش‌برازش [3] به پدیده نامطلوبی است که باعث می‌شود مدل بر روی داده‌های آموزش دیده به خوبی نتیجه دهد اما با ورود داده جدید این پاسخ‌گویی دارای خطای زیاد شود. به عبارت دیگر بیش‌برازش زمانی اتفاق می‌افتد که مدل در هنگام برازش به جای یادگیری، شروع به حفظ کردن داده‌های ورودی کند.

---

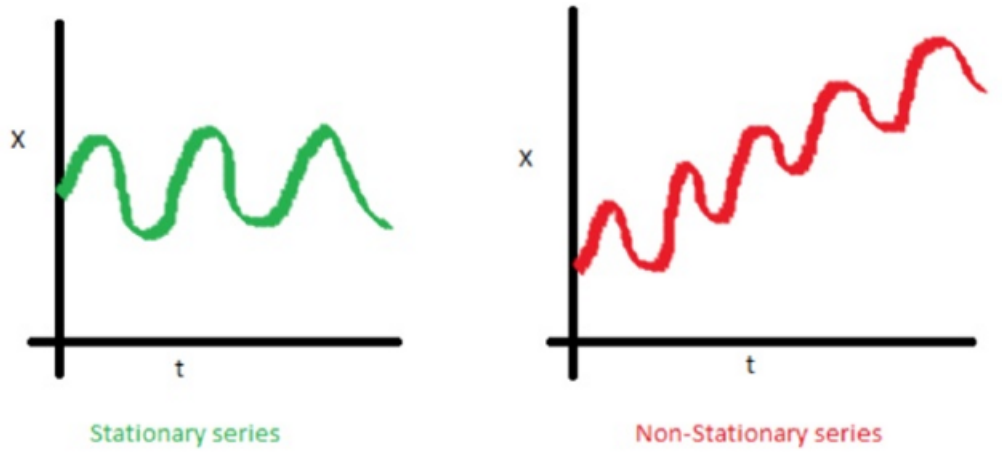
[1] Exploding Gradient [2] Vanishing Gradient [3] Overfitting

۱-۵ روش اتورگرسیو-میانگین متحرک جمع‌بسته (ARIMA)

روش **اتورگرسیو-میانگین متحرک جمع‌بسته**، یکی از مدل‌های برپایه سری زمانی می‌باشد. در مدل‌های بر پایه سری زمانی روش‌های پیش‌بینی و تعیین ساختار داده، داده‌ها وابسته به زمان می‌باشد. اتورگرسیو-میانگین متحرک جمع‌بسته از سه بخش تشکیل شده است که بخش اول اتورگرسیو [۱] بودن داده‌ها را مشخص می‌کند. بخش دوم این روش یکپارچگی [۲] و بخش سوم نیز میانگین متحرک [۳] را تشکیل می‌دهند که برای هموارسازی مقادیر سری زمانی با یکدیگر ترکیب می‌شوند.

۱-۵-۱ سری زمانی ایستا

سری زمانی ایستا [۴]، سری زمانی می‌باشد که خصوصیات آماری آن مانند میانگین و واریانس در طول زمان ثابت باشد. در واقع سری زمانی ایستا، دنباله‌ای از مقادیر وابسته به زمان است که میانگین و واریانس آن به زمان وابسته نباشد.



شکل ۱-۸: سری زمانی ایستا و غیرایستا

در شکل ۱-۸ دو نمودار در طول زمان مشخص شده است. نمودار سمت چپ (سبز رنگ) یک داده ایستا در طول زمان را نمایش می‌دهد و نمودار سمت راست (قرمز رنگ) نشان دهنده یک نمونه از نمودار غیر ایستا در طول زمان می‌باشد.

[1]Autoregressive [2]Integrated [3]Moving Average [4]Stationary

۱-۵-۲ اجزای مدل اتورگرسیو-میانگین متحرک جمع بسته

مدل اتورگرسیو-میانگین متحرک جمع بسته از سه بخش اصلی تشکیل می‌شود. در واقع این روش با سه پارامتر p، d و q کنترل می‌شود تا قابلیت پاسخگویی بهتر بر روی داده‌هایی با ساختار سری زمانی را انجام دهد. پارامتر p مربوط به بخش اتورگرسیو بودن مدل، پارامتر d مربوط به بخش یکپارچه سازی سری زمانی (ایستایی) و در نهایت پارامتر q تنظیم کننده درجه میانگین متحرک می‌باشد.

۱-۵-۲-۱ بخش اتورگرسیو

فرآیند اتورگرسیو متشکل از یک تابع خطی مشاهدات قبلی به علاوه میزان نویزی تصادفی می‌باشد. در هر لحظه از سری زمانی، یک اتورگرسیو تصمیم می‌گیرد با توجه به اطلاعات قبلی اطلاعات جدید چه مقداری را دارد. یک تابع اتورگرسیو با درجه p به صورت $AR(p, X)$ نمایش داده می‌شود که محاسبات برای لحظه t از این سری زمانی به شکل زیر می‌باشد:

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + ... + \phi_p X_{t-p} = \sum_{i=1}^{p} \phi_i X_{t-i} \qquad (۱۲-۱)$$

که در آن X سری زمانی، $X_t, X_{t-1}, ...X_{t-p}$ مقدار سری زمانی در لحظات مختلف، $\phi_1, \phi_2, ..., \phi_p$ پارامترهای تصادفی نابرابر با صفر مدل اتورگرسیو و p درجه تابع اتورگرسیو $AR(p, X)$ است.

با افزایش درجه یک مدل اتورگرسیو میزان وابستگی مقدار فعلی تابع اتورگرسیو به مقادیر پیشین بیشتر می‌شود. اصطلاحاً با افزایش پارامتر p تابع اتورگرسیو حافظه بیشتری را جهت محاسبه مقدار فعلی دارا می‌شود. به عنوان مثال تابع با داشتن درجه $p = 2$ از دو جمله قبلی و تابع با درجه $p = 8$ از ۸ جمله قبلی آن سری زمانی برای محاسبه مقدار فعلی تابع اتورگرسیو استفاده می‌کند.

۱-۵-۲-۲ بخش میانگین متحرک

روش میانگین متحرک، تابعی خطی با پارامتر q می‌باشد که با نماد $MA(q, X)$ معرفی می‌شود. این تابع خطی نشان دهنده مقدار سری زمانی در لحظه t با q مقدار خطا (نویز) است که تشکیل تابعی خطی می‌دهد. رابطه میانگین متحرک مشابه رابطه اتورگرسیو در بخش ۱-۵-۲-۱ می‌باشد با این تفاوت که که ترکیبات خطی برحسب خطاهایی بر اساس داده‌های زمانی است. این خطاها با ε_t نمایش داده می‌شود که دارای دو ویژگی اصلی زیر هستند:

- میانگین نویزها در زمان t برابر صفر است.
- واریانس نویزها در زمان t برابر ۱ است.

با توجه به نویزهای تعریف شده در مدل، مقدار پیش‌بینی برای تابع میانگین متحرک برابر حاصل ضرب ضرایب غیرصفر در نویزها تشکیل یک رابطه خطی می‌دهد و مفروض است:

$$X_t = \varepsilon_t + \beta_1\varepsilon_{t-1} + \beta_2\varepsilon_{t-2} + ... + \beta_q\varepsilon_{t-q} = \varepsilon_t + \sum_{i=1}^{q}\beta_i\varepsilon_{t-i} \qquad (۱۳-۱)$$

که در آن X سری زمانی، $\varepsilon_t, \varepsilon_{t-1}, ...\varepsilon_{t-q}$ مقدار نویز سری زمانی در لحظات مختلف، $\beta_1, \beta_2, ..., \beta_q$ پارامترهای تصادفی نابرابر با صفر مدل میانگین متحرک و q درجه تابع میانگین متحرک $MA(q, X)$ است.

۱-۵-۲-۳ بخش یکپارچه‌سازی

روش یکپارچه‌سازی یا جمع‌بسته، روشی برای ایستاسازی یک سری زمانی می‌باشد. این روش یک سری زمانی غیرایستا را به سری زمانی ایستا تبدیل می‌کند. فرآیند ایستاسازی (یکپارچه‌سازی) در مدل اتورگرسیو-میانگین متحرک جمع بسته با روش تفاضل‌گیری انجام می‌گیرد که با پارامتر d تعریف می‌شود. در روش تفاضل‌گیری d نشان دهنده درجه تفاضل در روش یکپارچه‌سازی می‌باشد که به شکل زیر تعریف می‌شود:

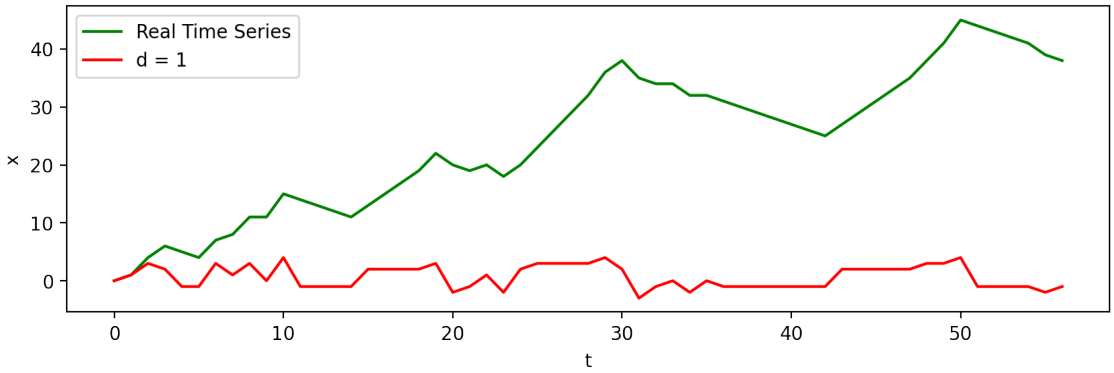
$$d = n \quad \Rightarrow \quad \hat{X}_t = (1 - L)^n X_t \qquad (۱۴-۱)$$

$$d = 0 \quad \Rightarrow \quad \hat{X}_t = (1 - L)^0 X_t = X_t \qquad (۱۵-۱)$$

$$d = 1 \quad \Rightarrow \quad \hat{X}_t = (1 - L)^1 X_t = X_t - LX_t = X_t - X_{t-1} \qquad (۱۶-۱)$$

$$d = 2 \quad \Rightarrow \quad \hat{X}_t = (1 - L)^2 X_t = (1 - 2L + L^2)X_t = X_t - 2X_{t-1} + X_{t-2} \qquad (۱۷-۱)$$

که در آن $..., X_t, X_{t-1}, ...$ مقادیر سری زمانی در زمان‌های مختلف، $\hat{X}_t$ مقدار خروجی سری زمانی در فرآیند ایستاسازی (یکپارچه‌سازی)، L نشان دهنده درجه تاخیر توانی مربوط به X_t و d درجه تابع یکپارچه‌سازی است. لازم به ذکر است پارامتر L در نقش درجه تاخیر در رابطه جبری استفاده می‌شود و عبارت AL^nX_t با ساده سازی به رابطه AX_{t-n} تبدیل می‌شود.



شکل ۱-۹: یکپارچه‌سازی درجه اول سری زمانی غیر ایستا

شکل ۱-۹ نمایشی از یک سری زمانی در حالت غیرایستا می‌باشد (رنگ سبز)، که با یکپارچه‌سازی درجه اول این سری زمانی تبدیل به یک سری زمانی ایستا می‌شود. پس از اعمال تابع یکپارچه‌سازی درجه اول سری زمانی جدید حاصل (رنگ قرمز)، خاصیت ایستایی را کسب می‌کند.

۱-۵-۳ ترکیب اجزا و تشکیل روش اتورگرسیو-میانگین متحرک جمع‌بسته

همان‌طور که در بخش ۱-۵-۲ اشاره شد، سه قسمت اصلی مدل اتورگرسیو-میانگین متحرک جمع‌بسته یعنی توابع $AR(p, X)$، $MA(q, X)$ و تابع یکپارچه‌سازی با پارامتر d بر روی یک سری زمانی تشکیل می‌شوند. با ترکیب اجزا این روش ، مدل $ARIMA(p, d, q)$ حاصل می‌شود که مفروض است:

$$ARIMA(p,d,q) = AR(p,\hat{X}) + MA(q,\hat{X})$$

$$\Rightarrow \hat{X}_t = \phi_1\hat{X}_{t-1} + \phi_2\hat{X}_{t-2} + ... + a_p\hat{X}_{t-p}$$

$$+\varepsilon_t + \beta_1\varepsilon_{t-1} + \beta_2\varepsilon_{t-2} + ... + \beta_q\varepsilon_{t-q}$$

$$= \sum_{i=1}^{p} \phi_i\hat{X}_{t-i} + \varepsilon_t + \sum_{i=1}^{q} \beta_i\varepsilon_{t-i}$$

که در آن p درجه اتورگرسیو، q درجه میانگین متحرک، d درجه ایستاسازی (یکپارچه‌سازی)، $\hat{X}$ سری ایستا شده از سری زمانی اصلی X با درجه d، $\hat{X}_t, \hat{X}_{t-1}, ...$ جملات سری یکپارچه‌سازی شده، $\phi_1, \phi_2, ...$ پارامترهای تصادفی نابرابر با صفر مدل اتورگرسیو $\varepsilon_t, \varepsilon_{t-1}, ...$ مقدار نویز سری زمانی در لحظات مختلف و $\beta_1, \beta_2, ...$ پارامترهای تصادفی نابرابر با صفر مدل میانگین متحرک است.

۱-۵-۴ پیش‌بینی با استفاده از اتورگرسیو-میانگین متحرک جمع‌بسته

به کمک روش اتورگرسیو-میانگین متحرک جمع‌بسته امکان پیش‌بینی سری زمانی برقرار می‌شود. مطابق روش ارائه شده در بخش ۱-۵-۳ امکان محاسبه مقدار سری زمانی در لحظه t ام وجود دارد. فرآیند پیش‌بینی با با داشتن $t-1$ جمله از سری زمانی صورت می‌گیرد. سپس پارامترهای اتورگرسیو (p)، درجه میانگین متحرک (q) و درجه یکپارچه‌سازی (d) تنظیم و انتخاب می‌شوند. با استفاده از داشتن پارامترهای ذکر شده و همچنین $t-1$ مقدار از سری زمانی، می‌توان عملیات پیش‌بینی مقدار t ام سری زمانی را با رابطه شرح داده شده در بخش ۱-۵-۳ ، از روش اتورگرسیو-میانگین متحرک جمع‌بسته محاسبه کرد. پس از محاسبه جمله t ام این سری زمانی، می‌توان مقادیر بعدی این سری زمانی یعنی $t+1, t+2, \ldots$ را با انجام عملیات تکراری بدست آورد.

فصل ۲

انواع شبکه‌های عصبی

شبکه‌های عصبی مصنوعی بر اساس چینش لایه‌های، ساختار درونی هرلایه، نحوه ارتباط لایه‌ها به یکدیگر و عوامل مختلف دیگر دارای تنوع‌های مختلفی هستند. هرکدام از این نوع شبکه‌ها نسبت به ورودی و صورت مسئله بازخورد متفاوت به خود را دارند. برخی شبکه‌های عصبی برای حل مسائل شناسایی و برخی دیگر برای حل مسائل پیش‌بینی با استفاده از داده تاریخچه‌ای نتیجه بهتری را حاصل می‌کنند. در این فصل به تعریف برخی از انواع شبکه‌های عصبی که در ادامه نوشتار مورد استفاده قرار گرفته‌اند پرداخته شده است. شبکه‌های عصبی بر خلاف ساختار ساده و الگوریتم قابل درک آن دارای تنوع و پارامترهای گوناگونی می‌باشند.

۲-۱ شبکه عصبی پیچشی CNN

شبکه عصبی کانولوشنال یا پیچشی (CNN) نوعی شبکه عصبی مصنوعی است که برای شناسایی و پردازش تصویر استفاده می‌شود. این شبکه‌ها اختصاصا برای پردازش داده‌های پیکسلی طراحی شده‌اند. شبکه‌های عصبی پیچشی در پردازش تصویر و هوش مصنوعی، شبکه‌های کارآمدی هستند که با استفاده از یادگیری عمیق کارهای تولیدی و توصیفی را انجام می‌دهند. این شبکه‌ها اغلب از دید ماشینی که شامل تشخیص تصویر و ویدئو به همراه سیستم‌های توصیه‌گر و پردازش زبان طبیعی [1] می‌شود، استفاده می‌کنند.

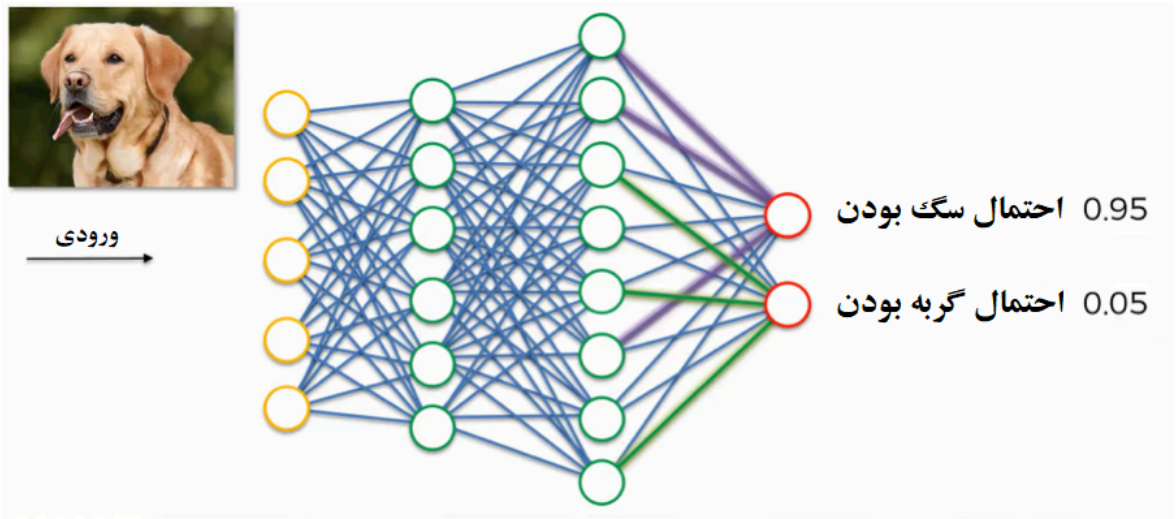
۲-۱-۱ ورودی و خروجی شبکه پیچشی

وقتی یک کامپیوتر تصویری را به عنوان ورودی دریافت می‌کند، آن را به صورت آرایه‌ای از اعداد می‌بیند. تعداد آرایه‌ها به سایز تصویر (بر اساس پیکسل) بستگی دارد. برای مثال اگر یک تصویر رنگی با فرمت JPG و اندازه ۴۸۰ × ۴۸۰ پیکسل

[1] Natural Language Processing

به کامپیوتر داده شود، آرایه جانشین آن دارای ۴۸۰ × ۴۸۰ × ۳ پیکسل عنصر خواهد بود (عدد ۳ به RGB اشاره دارد). هر کدام از عناصر آرایه نیز شامل عددی در بازه ۰ تا ۲۵۵ می‌باشد. این عدد شدت پیکسلی را نشان می‌دهد. این اعداد هر چند برای ما بی‌معنی به نظر می‌رسند، اما در دسته‌بندی تصاویر با استفاده از شبکه عصبی کانولوشن، تنها ابزار در دسترس، چنین اعدادی هستند. ایده اصلی آن است که به رایانه آرایه‌ای از اعداد شبیه آن چه توضیح ، می‌دهیم و کامپیوتر نیز در خروجی چنین چیزی را مشخص می‌کند: این تصویر با احتمال ٪۸۰ گربه، با احتمال ٪۱۵ سگ و با احتمال ٪۵، پرنده است [1].

در شکل ۲-۱ یک شبکه عصبی پیچشی با دو لایه می‌باشد. این شبکه یک عکس به عنوان ورودی دریافت و و پس از عملیات‌های شبکه عصبی پیچشی، در لایه آخر دو نورون خروجی مشخص کننده برچسب داده ورودی می‌باشد. در شکل تصویر یک سگ به عنوان ورودی داده شده است و در خروجی احتمال برچسب سگ برابر ۹۵ درصد و احتمال برچسب گربه برابر ۵ درصد محاسبه شده، که با احتمال بسیار قوی‌تر برچسب سگ به عنوان حاصل دسته‌بندی از شبکه عصبی پیچشی می‌باشد.



شکل ۲-۱: شبکه عصبی پیچشی گراف در مسائل دسته‌بندی

۲-۱-۲ لایه‌های شبکه عصبی پیچشی

شبکه‌های عصبی پیچشی شامل لایه‌هایی برای پردازش و انجام فرآیند یادگیری است. این لایه‌ها به طور کلی به دو سطح پیچش و تجمع دسته‌بندی می‌شود. حاصل عملیات پیچش و تجمع به لایه طبقه‌بندی منتقل شده تا نسبت به تصمیم‌گیری اقدام شود. در ادامه به تفصیل لایه‌های شبکه عصبی پیچشی پرداخته شده است.

[1] https://fanology.ir/convolutional-neural-network

۲-۱-۲-۱ لایه پیچش

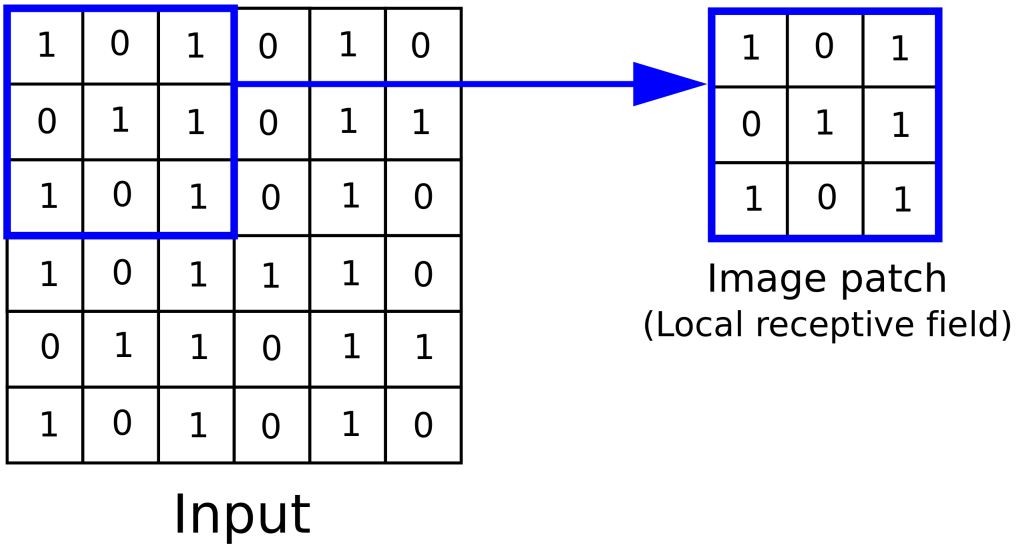
لایه کانولوشن یا لایه پیچشی در یک شبکه عصبی دارای ورودی از جنس آرایه‌ای از اعداد است. لایه اول در شبکه عصبی مانند یک چراغ قوه بر روی قسمت‌های تاریک یک تصویر عمل می‌کند. به عنوان نمونه در یک اتاق تاریک، چراغ قوه‌ای تصور شده که بر گوشه‌ی بالا و سمت چپ تصویر انداخته شده است، محدوده‌ای از تصویر که با نور چراغ قوه روشن می‌شود قابل مشاهده است. فرض شود چراغ قوه را بر روی قسمت‌های دیگر تصویر حرکت کند تا تمامی قسمت‌های تصویر روشن شود. این رویکرد معادل عملکرد شبکه‌های عصبی پیچشی در یادگیری ماشین می‌باشد.

در شبکه عصبی کانولوشن، این چراغ قوه، فیلتر [۱] (یا نورون یا کرنل) نام دارد. قسمتی از تصویر که چراغ قوه به آن نور می‌تاباند، محدوده پذیرش [۲] نامیده می‌شود. لازم به ذکر است، فیلترها نیز خود آرایه‌هایی از اعداد هستند. به اعداد موجود در فیلتر، وزن [۳] یا پارامتر گفته می‌شود. هم‌چنین عمق این فیلتر باید با عمق تصویر برابر باشد.

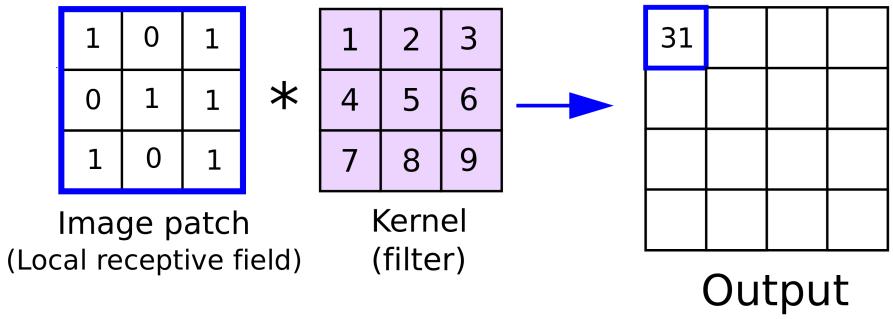
به عنوان مثال اگر تصویر یک آرایه عددی ۳ × ۵ × ۵ (عمق ۳) است، فیلتر نیز باید مطابق این تصویر با عمق ۳ تعریف شود. فیلتر در هر نگاه، یک قسمت از تصویر را بررسی و سپس بر روی تصویر حرکت می‌کند تا قسمت‌های دیگر تصویر نیز مورد ارزیابی قرار گیرد. حرکت فیلتر بر روی تصویر به عنوان عمل پیچش [۴] تعریف می‌شود. در گذر فیلتر بر روی تصویر، اعداد موجود در فیلتر با آرایه عددی پیکسل‌های تصویر ضرب و در نهایت نیز تمام حاصل ضرب‌ها با یکدیگر جمع شده و عدد نهایی حاصل می‌شود. به فرض یک تصویر با ابعاد ۳ × ۳۲ × ۳۲ را با استفاده از یک فیلتر ۳ × ۵ × ۵ مورد ارزیابی قرار داده شود. این فیلتر با عملیات توضیح داده شده در نهایت یک آرایه عددی با ابعاد ۱ × ۲۸ × ۲۸ تولید می‌کند (علت ۲۸ × ۲۸ بودن آن است که به ۷۸۴ حالت می‌توان یک تصویر ۳۲ × ۳۲ را با استفاده از فیلتر ۵ × ۵ گذر داد). به ماتریس ۱ × ۲۸ × ۲۸ که در نهایت به دست می‌آید نقشه فعال‌سازی [۵] یا نقشه ویژگی [۶] گفته می‌شود. اگر به عنوان جایگزین یک فیلتر، از دو فیلتر استفاده شود، در نهایت ماتریسی با ابعاد ۲ × ۲۸ × ۲۸ حاصل می‌شود. این کار می‌تواند دقت را در ابعاد بالاتر افزایش دهد.

---

[1] filter [2] receptive field [3] weight [4] convolve [5] activation map [6] feature map



شکل ۲-۲: انتخاب محدوده پذیرش عملیات پیچش در شبکه عصبی پیچشی



شکل ۲-۳: حاصل یک گذر از کرنل عملیات پیچش در شبکه عصبی پیچشی

در شکل ۲-۲ ورودی با اندازه ۶ × ۶ و محدوده پذیرش ۳ × ۳ انتخاب شده است. این محدوده پذیرش مطابق با شکل ۲-۳ در ماتریس ویژگی با اندازه‌ای مشابه خود ضرب و حاصل به عنوان نتیجه عملیات پیچش در خروجی قرارمی‌گیرد.

۲-۱-۲-۲ لایه تجمعی

لایه تجمعی [1] در شبکه عصبی پیچشی عملیات کاهش اندازه مکانی (عرض و ارتفاع) فضای ویژگی پیچانده [2] شده را انجام می‌دهد. عملکرد این لایه با هدف کاهش قدرت محاسباتی مورد نیاز برای پردازش داده‌ها از طریق کاهش ابعاد، انجام می‌شود. عملیات تجمع شامل راه حلی برای کاهش ابعاد می‌باشد، دو نوع رایج در مبحث لایه تجمع وجود دارد:

[1] Pooling layer [2] Convolved

- تجمع بیشینه Max Pooling

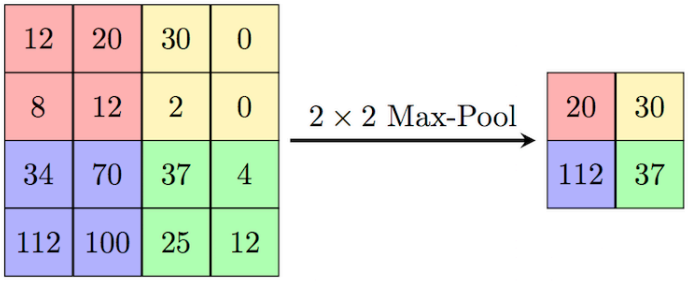
تجمع بیشینه، مقدار بیشینه را از قسمتی از تصویر بازمی‌گرداند که توسط کرنل پوشش داده شده است. تجمع بیشینه، کار حذف نویز [1] را نیز انجام می‌دهد.

شکل ۲-۴ یک نمایش از عملیات تجمع بیشینه می‌باشد که بر روی ورودی با اندازه ۴ × ۴ و فیلتری با ابعاد ۲ × ۲ انجام شده است. حاصل ابعاد کاهش یافته به اندازه ۲ × ۲ می‌باشد.

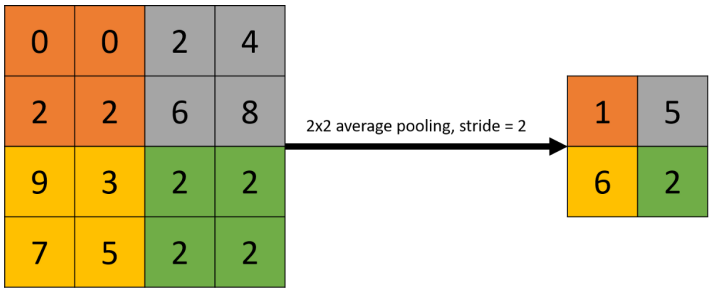
- تجمع میانگین Average Pooling

تجمع میانگین، میانگین همه مقادیر را از قسمتی از تصویر باز می‌گرداند که توسط کرنل پوشش داده شده است. در این عملیات، تمامی اعداد موجود در بازه فیلتر مورد میانگین‌گیری قرار گرفته و نتیجه محاسبه به عنوان حاصل خروجی روش تجمع میانگین می‌باشد.

شکل ۲-۵ یک نمایش از عملکرد روش تجمع میانگین با اندازه ورودی ۴ × ۴ و فیلتری با ابعاد ۲ × ۲ انجام شده است. حاصل ابعاد کاهش یافته به اندازه ۲ × ۲ می‌باشد.



شکل ۲-۴: عملیات تجمع بیشینه در شبکه عصبی پیچشی



شکل ۲-۵: عملیات تجمع میانگین در شبکه عصبی پیچشی

[1] Noise Suppressant

۲-۱-۲-۳ لایه دسته‌بندی- کاملا متصل

لایه کاملا متصل [1] به عنوان آخرین لایه یک شبکه عصبی پیچشی مورد استفاده قرار می‌گیرد. در معماری شبکه عصبی پیچشی پس از پیچش و تجمع‌های متعدد نیاز است تا حاصل نهایی برای تصمیم‌گیری به یک لایه تماماً متصل داده شود تا عمل دسته‌بندی انجام پذیرد. این لایه با استفاده از ماتریس ضرایب و میزان بایاس مخصوص خود وظیفه امتیاز دهی به ورودی حاصل را انجام و نسبت به انتخاب دسته بندی ورودی مورد نظر اقدام می‌کند.

۲-۲ شبکه عصبی پیچشی گراف GCN

در یادگیری ماشین مجموعه داده‌هایی با ساختار گراف (مانند شبکه معابر شهری) یادگیری و آموزش دشواری را به همراه دارند. ساختار شبکه پیچشی گراف این امکان را فراهم می‌سازد تا شبکه عصبی عمل یادگیری، آموزش، تشخیص یا پیش‌بینی را انجام دهد. شبکه عصبی پیچشی به‌گونه‌ای طراحی شده است تا با استفاده از ساختار خود مسائل با ساختار گراف را حل کند. شبکه عصبی پیچشی گراف ایده گرفته از شبکه‌پیچشی می‌باشد. معمولاً شبکه پیچشی دارای ورودی با مشخصه تصویر است، پیکسل‌ها و المان‌های تصاویر بخاطر مجاورتی که با یکدیگر دارند در عمل کانولوشن و تجمع ویژگی‌ها حفظ شده و پارامترها استخراج می‌شوند. در شبکه عصبی پیچشی گراف نیز یک رأس به علت مجاورت با راس‌های دیگر نیاز به حفظ این ویژگی همسایگی دارد. شبکه‌های عصبی پیچشی گراف یک معماری مبتنی بر شبکه عصبی پیچشی برای یادگیری بر روی گراف‌ها هستند.

شبکه‌های عصبی پیچشی گراف الگوریتم مشابه شبکه‌های پیچشی دارند با این تفاوت که در شبکه‌های پیچشی ساختار داده‌های ورودی منظم اقلیدسی [2] می‌باشد، در حالی که در شبکه‌های پیچشی گراف داده ورودی غیر ساختار یافته است و روابط اقلیدسی در آن برقرار نیست. اتصالات گره‌ها در این مدل پیش‌بینی متفاوت است که باعث می‌شود با استفاده از روابط ویژه گراف برای تحلیل داده‌های ورودی استفاده شود [۵] .

۲-۲-۱ ورودی شبکه پیچشی گراف

شبکه عصبی پیچشی گراف همانند شبکه‌های عصبی پرسپترون چند لایه، با الگوریتم پس انتشار خطا آموزش می‌بینند. شبکه، برای یادگیری و آموزش دیدن از مجموعه داده‌ها و همچنین وزن دهی مناسب به نورون‌ها نیاز به دو دسته ورودی دارد:

- **ماتریس مجاورت گراف** ($A \in R^{N \times N}$) :

برای انتقال دانش و ساختار ورودی یک گراف به شبکه عصبی، نیازمند ساختاری منظم و قابل درک برای انتقال به کامپیوتر می‌باشد. تصویر یک گراف ساختار مناسبی را برای استخراج داده از آن در اختیار نورون‌های شبکه عصبی

[1] Fully-Connected layer [2] Euclidean

قرار نمی‌دهد. از این‌رو نیاز است تا ساختاری یکپارچه تحت عنوان ماتریس مجاورت گراف به شبکه عصبی داده شود. این ماتریس با ابعاد $N \times N$ می‌باشد که در آن N تعداد راس‌های یک گراف است.

- **ماتریس ویژگی** ($X \in R^{N \times D}$):

هر گره از این گراف تعداد ویژگی‌های متعددی را شامل می‌شود که نیاز به پردازش و یادگیری آن توسط شبکه عصبی پیچشی گراف است. این ویژگی‌ها در قالب یک ماتریس که عموماً با X نشانه‌گذاری می‌شود، یک ماتریس با ابعاد $N \times D$ است که N مطابق تعریف پیشین، تعداد راس‌های این گراف و D تعداد ویژگی‌های موجود برای هر رأس در ماتریس ویژگی می‌باشد.

۲-۲-۲ لایه پنهان شبکه عصبی پیچشی گراف

شبکه عصبی پیچشی گراف دارای ساختار چند لایه می‌باشد، لایه‌های شبکه، ورودی را از اولین لایه دریافت و مقادیر نورون‌ها را برای ارسال به نورون‌های لایه بعدی مهیا می‌کنند. هر لایه از این GCN با توجه به ماتریس مجاورت گراف، ماتریس درجات گراف و وزن‌های لایه قبل ساخته می‌شود و با عملیات پس‌انتشار (backpropagation) این لایه در جهت بهینه شدن بروزرسانی می‌شود.

هر لایه از شبکه عصبی پیچشی گراف دارای وزن‌های مشخصی برای نورون‌های مربوطه می‌باشد که با $W^{(l-1)}$ نشانه گذاری می‌شود و ورودی هر لایه از شبکه، خروجی لایه قبل $H^{(l-1)}$ می‌باشد. لازم به ذکر است که ورودی لایه اول ماتریس ویژگی گراف X می‌باشد.

در شبکه عصبی پیچشی گراف از حاصل ضربی از ماتریس مجاورت و ماتریس درجات گراف A استفاده می‌شود. در ماتریس حاصل ذکر شده ابتدا گراف را تبدیل به یک گراف با ویژگی تمامی گره‌ها خود متصل [1] تبدیل می‌شود، ماتریس مجاورت گراف حاصل با $\widetilde{A}$ نشانه‌گذاری می‌شود و مفروض است:

$$\widetilde{A} = A + I_N \tag{۱-۲}$$

سپس ماتریس درجات گراف حاصل $\widetilde{D}$ مورد محاسبه قرار می‌گیرد:

$$\widetilde{D}_{i,i} = \sum_j \widetilde{A}_{i,j} \tag{۲-۲}$$

با تعاریف ذکر شده می‌توان ماتریس مجاورت گراف را به شکل حاصل ضربی از ماتریس تماماً خود متصل و ماتریس

[1] Self-Connected

درجات نمایش داد:

$$\hat{A} = \widetilde{D}^{-\frac{1}{2}}\ \widetilde{A}\ \widetilde{D}^{-\frac{1}{2}} \qquad (۳-۲)$$

که در آن $\hat{A}$ ماتریس حاصل ضرب ماتریس مجاورت و ماتریس درجات گراف می‌باشد.

برای هر لایه از این شبکه عصبی پیچشی گراف $H^{(l+1)}$ ، با وابسته بودن به ماتریس مجاورت و لایه پیشین خود برابر است با:

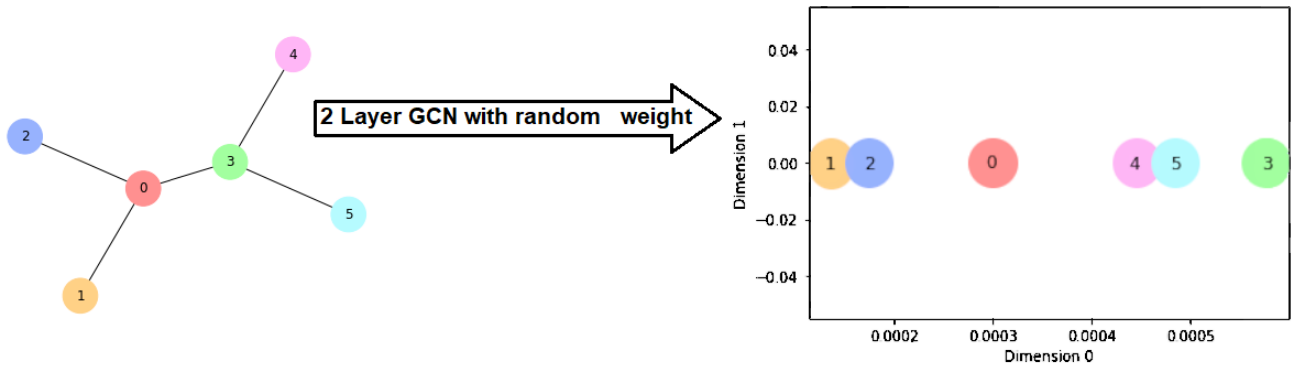
$$H^{(l+1)} = \sigma\left(\hat{A}\ H^{(l)}\ W^{(l)}\right) \qquad (۴-۲)$$

که در آن $H^{(l)}$ خروجی لایه l ام شبکه، $H^{(l+1)}$ خروجی لایه $l+1$ ام شبکه، $\sigma(.)$ تابع فعال‌ساز غیر خطی، $\hat{A}$ ماتریس حاصل ضرب ماتریس مجاورت گراف و ماتریس درجات گراف و $W^{(l)}$ وزن لایه l ام شبکه می‌باشد.

لازم به ذکر است، مقادیر لایه اول برابر ماتریس ویژگی می‌باشد:

$$H^{(0)} = X \qquad (۵-۲)$$

که در آن X ماتریس ویژگی راس‌ها، $H^{(0)} \in R^{N\times F}$ لایه اول شبکه، برابر ماتریس ویژگی است که N نشان‌دهنده تعداد راس‌ها و F تعداد ویژگی‌های هر رأس می‌باشد.



شکل ۲-۶: نمونه‌ای از عملکرد شبکه عصبی پیچشی گراف

در شکل ۲-۶ به‌صورت شماتیک، عملکردی از یک شبکه عصبی پیچشی گراف با داشتن دو لایه نشان داده شده است. ورودی شبکه مفروض(شکل سمت چپ)، گرافی با ۶ راس می‌باشد. هر رأس با رنگ مختص خود مشخص شده و اعداد

نمایش داده شده در شکل برچسب این راس‌ها است. در شبکه پیچشی گراف برای ورودی از ماتریس ویژگی $X \in R^{N\times ۱}$ و ماتریس مجاورت $A \in R^{N\times N}$ استفاده می‌شود، که در آن N تعداد راس‌های، X ماتریس ویژگی گراف و A ماتریس مجاورت گراف می‌باشند.

این شبکه نمونه به ترتیب دارای یک لایه پنهان و یک لایه خروجی نیز می‌باشد. لایه پنهان از ۴ نورون و لایه خروجی از ۲ نورون تشکیل شده است. شکل حاصل در سمت راست نمایانگر خروجی این شبکه از نتیجه حاصل در لایه خروجی می‌باشد. از مشاهده نتایج مشخص است، به‌صورت کلی می‌توان راس‌های این گراف را به دو دسته تقسیم کرد. راس‌های ۰ ، ۱ و ۲ یک دسته و راس‌های ۳ ، ۴ و ۵ دسته دیگر را تشکیل می‌دهند که با توجه به گراف تعریف شده قابل برداشت است [۳۰] . این عملیات بدون شامل شدن عملیات پس‌انتشار و اصلاح وزن‌ها می‌باشد که قطعه برنامه مربوط به آن در پیوند [۱] قابل مشاهده می‌باشد.

در مثال ۲-۶ ، ماتریس مجاورت مفروض است:

$$A = \begin{bmatrix} ۰ & ۱ & ۱ & ۱ & ۰ & ۰ \\ ۱ & ۰ & ۰ & ۰ & ۰ & ۰ \\ ۱ & ۰ & ۰ & ۰ & ۰ & ۰ \\ ۱ & ۰ & ۰ & ۰ & ۱ & ۱ \\ ۰ & ۰ & ۰ & ۱ & ۰ & ۰ \\ ۰ & ۰ & ۰ & ۱ & ۰ & ۰ \end{bmatrix}$$

پس از اتصال هر رأس به خود، ماتریس مجاورت خود متصل حاصل می‌شود:

$$\widetilde{A} = A + I_N = \begin{bmatrix} ۱ & ۱ & ۱ & ۱ & ۰ & ۰ \\ ۱ & ۱ & ۰ & ۰ & ۰ & ۰ \\ ۱ & ۰ & ۱ & ۰ & ۰ & ۰ \\ ۱ & ۰ & ۰ & ۱ & ۱ & ۱ \\ ۰ & ۰ & ۰ & ۱ & ۱ & ۰ \\ ۰ & ۰ & ۰ & ۱ & ۰ & ۱ \end{bmatrix}$$

[1] https://github.com/Morteza-j8/GCN-without-backpropagation

پس از محاسبه ماتریس مجاورت گراف خود متصل، ماتریس درجات گراف خود متصل طبق رابطه زیر مفروض است:

$$\widetilde{D} = \begin{bmatrix} ۴ & ۰ & ۰ & ۰ & ۰ & ۰ \\ ۰ & ۲ & ۰ & ۰ & ۰ & ۰ \\ ۰ & ۰ & ۲ & ۰ & ۰ & ۰ \\ ۰ & ۰ & ۰ & ۴ & ۰ & ۰ \\ ۰ & ۰ & ۰ & ۰ & ۲ & ۰ \\ ۰ & ۰ & ۰ & ۰ & ۰ & ۲ \end{bmatrix}$$

طبق رابطه ۲-۳ ماتریس $\hat{A}$ برابر است با:

$$\hat{A} = \widetilde{D}^{-\frac{۱}{۲}} \widetilde{A} \widetilde{D}^{-\frac{۱}{۲}}$$

$$= \begin{bmatrix} ۴ & ۰ & ۰ & ۰ & ۰ & ۰ \\ ۰ & ۲ & ۰ & ۰ & ۰ & ۰ \\ ۰ & ۰ & ۲ & ۰ & ۰ & ۰ \\ ۰ & ۰ & ۰ & ۴ & ۰ & ۰ \\ ۰ & ۰ & ۰ & ۰ & ۲ & ۰ \\ ۰ & ۰ & ۰ & ۰ & ۰ & ۲ \end{bmatrix}^{-\frac{۱}{۲}} \begin{bmatrix} ۱ & ۱ & ۱ & ۱ & ۰ & ۰ \\ ۱ & ۱ & ۰ & ۰ & ۰ & ۰ \\ ۱ & ۰ & ۱ & ۰ & ۰ & ۰ \\ ۱ & ۰ & ۰ & ۱ & ۱ & ۱ \\ ۰ & ۰ & ۰ & ۱ & ۱ & ۰ \\ ۰ & ۰ & ۰ & ۱ & ۰ & ۱ \end{bmatrix} \begin{bmatrix} ۴ & ۰ & ۰ & ۰ & ۰ & ۰ \\ ۰ & ۲ & ۰ & ۰ & ۰ & ۰ \\ ۰ & ۰ & ۲ & ۰ & ۰ & ۰ \\ ۰ & ۰ & ۰ & ۴ & ۰ & ۰ \\ ۰ & ۰ & ۰ & ۰ & ۲ & ۰ \\ ۰ & ۰ & ۰ & ۰ & ۰ & ۲ \end{bmatrix}^{-\frac{۱}{۲}}$$

$$= \begin{bmatrix} ۰/۲۵ & ۰/۳۵۳۵ & ۰/۳۵۳۵ & ۰/۲۵ & ۰ & ۰ \\ ۰/۳۵۳۵ & ۰/۵ & ۰ & ۰ & ۰ & ۰ \\ ۰/۳۵۳۵ & ۰ & ۰/۵ & ۰ & ۰ & ۰ \\ ۰/۲۵ & ۰ & ۰ & ۰/۲۵ & ۰/۳۵۳۵ & ۰/۳۵۳۵ \\ ۰ & ۰ & ۰ & ۰/۳۵۳۵ & ۰/۵ & ۰ \\ ۰ & ۰ & ۰ & ۰/۳۵۳۵ & ۰ & ۰/۵ \end{bmatrix}$$

با محاسبه $\hat{A}$ و انتخاب وزن‌های مربوط به لایه l ام شبکه عصبی پیچشی گراف، خروجی لایه l ام شبکه عصبی پیچشی

گراف از رابطه ۲-۴ قابل محاسبه می‌باشد. با فرض ماتریس ویژگی گراف (X) به عنوان لایه ۰ ام مفروض است:

$$H^{(۰)} = X = \begin{bmatrix} ۰ \\ ۱ \\ ۲ \\ ۳ \\ ۴ \\ ۵ \end{bmatrix}$$

برای محاسبه خروجی هر لایه با توجه به رابطه ۲-۴ عمل می‌شود. خروجی لایه ۱ ام با توجه به انتخاب وزن‌های تصادفی برای $W^{(۰)} \in R^{۱\times۴}$ و تابع فعال‌ساز $ReLU$ برابر است با:

$$H^{(۱)} = \sigma\big(\hat{A}\, H^{(۰)}\, W^{(۰)}\big)$$

$$= \sigma\left(\begin{bmatrix} ۰/۲۵ & ۰/۳۵۳۵ & ۰/۳۵۳۵ & ۰/۲۵ & ۰ & ۰ \\ ۰/۳۵۳۵ & ۰/۵ & ۰ & ۰ & ۰ & ۰ \\ ۰/۳۵۳۵ & ۰ & ۰/۵ & ۰ & ۰ & ۰ \\ ۰/۲۵ & ۰ & ۰ & ۰/۲۵ & ۰/۳۵۳۵ & ۰/۳۵۳۵ \\ ۰ & ۰ & ۰ & ۰/۳۵۳۵ & ۰/۵ & ۰ \\ ۰ & ۰ & ۰ & ۰/۳۵۳۵ & ۰ & ۰/۵ \end{bmatrix} \begin{bmatrix} ۰ \\ ۱ \\ ۲ \\ ۳ \\ ۴ \\ ۵ \end{bmatrix} \begin{bmatrix} ۰/۰۰۲ & -۰/۰۰۶ & ۰/۰۰۸ & ۰/۰۰۰۶ \end{bmatrix} \right)$$

$$= \sigma\left(\begin{bmatrix} ۰/۰۰۳۸ & -۰/۰۱۱۷ & ۰/۰۱۵۴ & ۰/۰۰۱۲ \\ ۰/۰۰۱۰ & -۰/۰۰۳۲ & ۰/۰۰۴۲ & ۰/۰۰۰۳ \\ ۰/۰۰۲۱ & -۰/۰۰۶۵ & ۰/۰۰۸۵ & ۰/۰۰۰۷ \\ ۰/۰۰۸۲ & -۰/۰۲۵۵ & ۰/۰۳۳۴ & ۰/۰۰۲۶ \\ ۰/۰۰۶۴ & -۰/۰۱۹۸ & ۰/۰۲۶۰ & ۰/۰۰۲۰ \\ ۰/۰۰۷۵ & -۰/۰۲۳۱ & ۰/۰۳۰۳ & ۰/۰۰۲۳ \end{bmatrix} \right)$$

با توجه به محاسبات صورت گرفته نتیجه حاصل خروجی لایه ۱ ام برابر است با:

$$H^{(1)} = \begin{bmatrix} ۰/۰۰۳۸ & ۰ & ۰/۰۱۵۴ & ۰/۰۰۱۲ \\ ۰/۰۰۱۰ & ۰ & ۰/۰۰۴۲ & ۰/۰۰۰۳ \\ ۰/۰۰۲۱ & ۰ & ۰/۰۰۸۵ & ۰/۰۰۰۷ \\ ۰/۰۰۸۲ & ۰ & ۰/۰۳۳۴ & ۰/۰۰۲۶ \\ ۰/۰۰۶۴ & ۰ & ۰/۰۲۶۰ & ۰/۰۰۲۰ \\ ۰/۰۰۷۵ & ۰ & ۰/۰۳۰۳ & ۰/۰۰۲۳ \end{bmatrix}$$

همچنین خروجی لایه ۲ ام با توجه به انتخاب وزن‌های تصادفی برای $W^{(1)} \in R^{۴\times۲}$ ، تابع فعال‌ساز $ReLU$ و خروجی لایه اول برابر است با:

$$H^{(۲)} = \sigma\left(\hat{A}\, H^{(1)}\, W^{(1)}\right)$$

$$= \sigma\left(\begin{bmatrix} ۰/۲۵ & ۰/۳۵۳۵ & ۰/۳۵۳۵ & ۰/۲۵ & ۰ & ۰ \\ ۰/۳۵۳۵ & ۰/۵ & ۰ & ۰ & ۰ & ۰ \\ ۰/۳۵۳۵ & ۰ & ۰/۵ & ۰ & ۰ & ۰ \\ ۰/۲۵ & ۰ & ۰ & ۰/۲۵ & ۰/۳۵۳۵ & ۰/۳۵۳۵ \\ ۰ & ۰ & ۰ & ۰/۳۵۳۵ & ۰/۵ & ۰ \\ ۰ & ۰ & ۰ & ۰/۳۵۳۵ & ۰ & ۰/۵ \end{bmatrix} \begin{bmatrix} ۰/۰۰۳۸ & ۰ & ۰/۰۱۵۴ & ۰/۰۰۱۲ \\ ۰/۰۰۱۰ & ۰ & ۰/۰۰۴۲ & ۰/۰۰۰۳ \\ ۰/۰۰۲۱ & ۰ & ۰/۰۰۸۵ & ۰/۰۰۰۷ \\ ۰/۰۰۸۲ & ۰ & ۰/۰۳۳۴ & ۰/۰۰۲۶ \\ ۰/۰۰۶۴ & ۰ & ۰/۰۲۶۰ & ۰/۰۰۲۰ \\ ۰/۰۰۷۵ & ۰ & ۰/۰۳۰۳ & ۰/۰۰۲۳ \end{bmatrix} \begin{bmatrix} ۰/۰۰۹۷ & -۰/۰۰۴۸ \\ -۰/۰۱۱۸ & -۰/۰۱۲۰ \\ ۰/۰۱۴۵ & -۰/۰۱۰۱ \\ ۰/۰۱۴۵ & ۰/۰۰۷۳ \end{bmatrix} \right)$$

$$= \sigma\left(\begin{bmatrix} ۰/۰۰۰۳ & -۰/۰۰۰۲ \\ ۰/۰۰۰۱ & -۰/۰۰۰۱ \\ ۰/۰۰۰۲ & -۰/۰۰۰۱ \\ ۰/۰۰۰۶ & -۰/۰۰۰۳ \\ ۰/۰۰۰۴ & -۰/۰۰۰۳ \\ ۰/۰۰۰۵ & -۰/۰۰۰۳ \end{bmatrix} \right)$$

$$= H^{(۲)} = \begin{bmatrix} ۰/۰۰۰۳۰۰۱ & ۰/۰۰۰۰ \\ ۰/۰۰۰۱۳۵۹ & ۰/۰۰۰۰ \\ ۰/۰۰۰۱۷۴۰ & ۰/۰۰۰۰ \\ ۰/۰۰۰۵۷۶۵ & ۰/۰۰۰۰ \\ ۰/۰۰۰۴۴۵۸ & ۰/۰۰۰۰ \\ ۰/۰۰۰۴۸۴۰ & ۰/۰۰۰۰ \end{bmatrix}$$

نتایج خروجی لایه آخر در شکل ۲-۶ در دو بعد نمایش داده شده است.

۲-۳ شبکه عصبی بازگشتی RNN

شبکه عصبی بازگشتی که به آن شبکه عصبی مکرر نیز گفته می شود، نوعی از شبکه عصبی مصنوعی است که بیشتر در پردازش داده‌های ترتیبی [1] استفاده می شود. بسیاری از شبکه‌های عمیق مانند CNN شبکه‌های پیش خور [2] هستند یعنی سیگنال در این شبکه‌ها فقط در یک جهت از لایه ورودی، به لایه‌های مخفی و سپس به لایه خروجی حرکت می کند و داده‌های قبلی به حافظه سپرده نمی شوند. اما شبکه‌های عصبی بازگشتی یک لایه بازخورد دارند که در آن خروجی شبکه به همراه ورودی بعدی، به شبکه بازگردانده می شود. RNN می تواند به علت داشتن حافظه داخلی، ورودی قبلی خود را به خاطر بسپارد و از این حافظه برای پردازش دنباله ایی از ورودی‌ها استفاده کند. به بیان ساده، شبکه‌های عصبی بازگشتی شامل یک حلقه بازگشتی هستند که موجب می شود اطلاعاتی را که از لحظات قبلی بدست آورده ایم از بین نروند و در شبکه باقی بمانند.

۲-۳-۱ ورودی شبکه عصبی بازگشتی

شبکه عصبی بازگشتی برای یادگیری نیاز به داده‌های تاریخچه‌ای (دنباله‌ای) می‌باشد. لایه‌ها و نورون‌های این شبکه ساختار مناسبی برای به حافظه سپردن جملات قبلی دنباله دارند. از سری داده‌های دنباله‌ای معروف می‌توان وضعیت آب و هوایی شهرها و به طور خاص در این نوشتار وضعیت ترافیکی معابر شهری را نام برد. داده‌های عنوان شده با توجه به توالی زمانی ساختار تاریخچه‌ای دارند.

۲-۳-۲ ساختار لایه‌های میانی شبکه‌عصبی بازگشتی

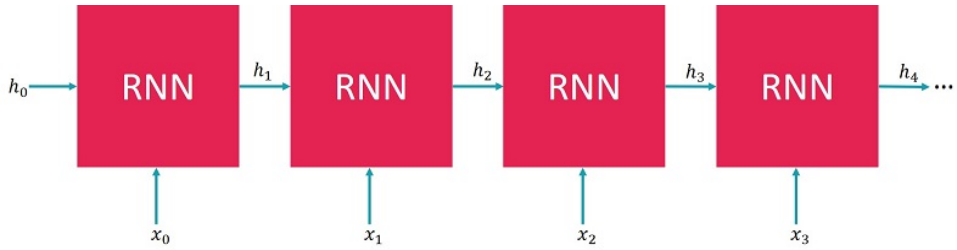
با توجه به ساختار داده‌های شبکه عصبی بازگشتی و خاصیت دنباله‌ای بودن این داده‌ها این شبکه به‌گونه‌ای طراحی شده است که حافظه‌ای را در خود جای دهد. هر لایه شبکه عصبی از دو ورودی استفاده می‌کند و خروجی را نتیجه می‌دهد.

ورودی‌های هر لایه (سلول) شبکه عصبی بازگشتی:

- ورودی اصلی: ماتریس ویژگی ورودی اصلی هر سلول می‌باشد و معمولاً با نماد X نشان داده می‌شود.
- وروردی حافظه: حافظه لحظات قبل را شامل می‌شود و در اصلاح به آن حالت پنهان گفته می‌شود. برای نماد گذاری آن در نوشتارهای مختلف معمولاً از h استفاده می‌شود. این ورودی حاصل پردازش لایه قبلی شبکه می‌باشد مطابق شکل ۲-۷ که نشان‌دهنده یک شبکه عصبی بازگشتی می‌باشد، ورودی و حاصل پردازش هر سلول در مراحل قبل به

[1] Sequential data [2] Feed Forward

سلول بعدی منتقل می‌شود تا در عمل پیش‌بینی نتیجه، هر دو عامل حافظه و ویژگی‌های همان ورودی دخیل باشند و جواب مطلوب‌تری محاسبه شود.



شکل ۲-۷: نمایش زنجیره‌ای از شبکه عصبی بازگشتی در طول زمان

خروجی‌های هر لایه شبکه عصبی بازگشتی: ورودی‌های ذکر شده پس از پردازش در خروجی این شبکه عصبی حاصل نتیجه می‌دهند، این نتیجه برای استفاده در لایه بعد مورد استفاده قرار می‌گیرد.برای محاسبه این خروجی معمولاً پس از محاسبات و اعمال جبری مربوط به نورون‌های لایه و بایاس از یک تابع فعال ساز غیر خطی عموماً تانژانت هذلولی استفاده می‌شود. استفاده از توابع فعال‌ساز غیر خطی با توجه به جنس مجموعه داده ورودی و نتیجه مورد نظر وابسته است.

$$h_t = tanh\left(W_{ht}h_{t-1} + W_{xt}X_t + b_t\right) \qquad (۶-۲)$$

که در آن h_t خروجی لایه t ام شبکه عصبی، W_{ht} و W_{xt} وزن‌های مربوط به نورون لایه t ام، b_t بایاس مربوط به لایه t ام و X ماتریس ویژگی گره‌ها می‌باشد.

۲-۳-۳ لایه انتهایی شبکه‌عصبی بازگشتی

پیش‌بینی و یا دسته‌بندی حاصل از این شبکه عصبی پیچشی بر عهده لایه آخر شبکه‌عصبی می‌باشد. این لایه ساختاری مشابه لایه‌های میانی دارد. در این لایه تابع فعال‌ساز و همین‌طور ماتریس ویژگی مورد نیاز نیست. این لایه نتیجه حاصل شده در شبکه عصبی بازگشتی را خروجی می‌دهد.

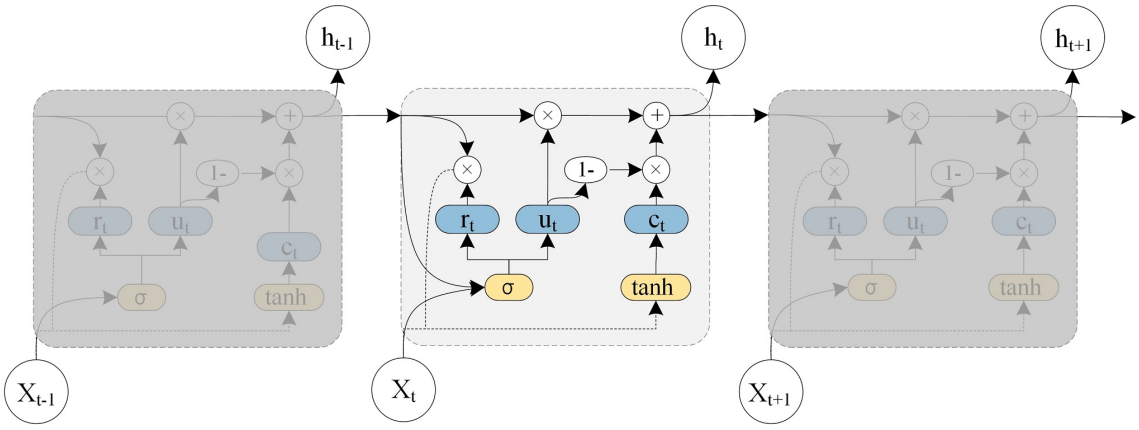
$$y_t = W_{hy}h_t + b_t \qquad (۷-۲)$$

که در آن y_t پیشبینی حالت t ام مجموعه داده می‌باشد.

۲-۴ شبکه عصبی بازگشتی با واحدهای دروازه‌ای GRU

حل مشکل وابستگی زمانی یکی‌دیگر از چالش‌های پیش‌بینی ترافیک معابر است، برای حل این مشکل وابستگی زمانی ترافیک هر معبر به ترافیک لحظات پیشین همان معبر، به‌طور متداول از شبکه‌های عصبی بازگشتی استفاده می‌شود اما این شبکه‌های عصبی دارای مشکل انفجار گرادیان [۱] هستند که باعث از بین رفتن یا تأثیر زیاد داده‌های تاریخچه‌ای لحظات قبل در تصمیم گیری فعلی می‌شوند [۳۳] .

برای حل مشکل انفجار گرادیان از شبکه عصبی بازگشتی با واحدهای حافظه‌ای استفاده می‌شود مانند شبکه عصبی بازگشتی حافظه طولانی کوتاه مدت ماندگار (LSTM) [۲] [۳۴] و شبکه عصبی بازگشتی با واحدهای دروازه‌ای (GRU) [۳۵] که در عمل ثابت شده است که مشکل نداشتن حافظه و انفجار گرادیان را برطرف می‌کنند. در این رویکرد از روش شبکه عصبی بازگشتی با واحدهای دروازه‌ای استفاده شده است، این دو شبکه هر دو کارآمد می‌باشند اما به علت کمتر بودن پارامترهای یادگیری ، کمتر بودن زمان یادگیری مدل و پیچیدگی کمتر مدل شبکه عصبی بازگشتی با واحدهای دروازه‌ای از این شبکه عصبی بازگشتی مورد استفاده قرار گرفته است (شکل ۲-۸).



شکل ۲-۸: نمایش چند سلول از شبکه عصبی بازگشتی

در شکل ۲-۸ سه سلول از شبکه عصبی بازگشتی با واحدهای دروازه‌ای نمایش داده شده است که در آن X_t ورودی در لحظه t و h_t خروجی مدل در لحظه t می‌باشد. در هر سلول t پارامترهای u_t، r_t و c_t باعث حافظه‌دار شدن آن سلول و بهره‌گیری یا از خاطر بردن آن سلول از سلول‌های سلول‌های پیشین شود.

پارامتر u_t به عنوان دروازه بروزرسانی [۳] استفاده می‌شود. دروازه بروزرسانی پارامتری است که تصمیم می‌گیرد در ورودی یک سلول، حالت قبلی مورد استفاده قرار گیرد و یا از ورودی آن استفاده شود. با تنظیم این دروازه قابلیت ترکیبی از حالت قبلی و ورودی نیز برقرار است:

[1] Vanishing Gradient [2] Long Short-Term Memory [3] Update Gate

$$u_t = \sigma\big(W_u[X_t, h_{t-۱}] + b_u\big) \tag{۸-۲}$$

که در آن W_u و b_u به ترتیب وزن و بایاس مربوط به دروازه سلول، X_t ورودی و $h_{t-۱}$ خروجی سلول پیشین می‌باشد.

پارامتر r_t به عنوان دروازه بازنشانی [1] استفاده می‌شود. دروازه بازنشانی پارامتری است که برای فراموشی مدل از داده‌های پیشین مورد استفاده قرار می‌گیرد. اصلاحا این دروازه مشخص می‌کند که چه میزان از اطلاعات گذشته در گام فعلی مورد استفاده نیست.

$$r_t = \sigma\big(W_r[X_t, h_{t-۱}] + b_r\big) \tag{۹-۲}$$

که در آن W_r و b_r به ترتیب وزن و بایاس مربوط به دروازه سلول، X_t ورودی و $h_{t-۱}$ خروجی سلول پیشین می‌باشد. در نهایت پارامتر c_t میزان خروجی متناسب از دروازه بازنشانی را محاسبه می‌کند:

$$c_t = \tanh\big(W_c[X_t, (r_t \times h_{t-۱})] + b_c\big) \tag{۱۰-۲}$$

که در آن W_c و b_c به ترتیب وزن و بایاس مربوط به دروازه سلول، r_t مقدار خروجی دروازه بازنشانی، X_t ورودی و $h_{t-۱}$ خروجی سلول پیشین می‌باشد.

با داشتن مقادیر به ازای دروازه‌های بروزرسانی و بازنشانی حاصل خروجی سلول مورد محاسبه قرار می‌گیرد:

$$h_t = u_t \times h_{t-۱} + (۱ - u_t) \times c_t \tag{۱۱-۲}$$

که در آن u_t مقدار خروجی دروازه بروزرسانی، c_t مقدار دروازه بازنشانی و $h_{t-۱}$ خروجی سلول پیشین می‌باشد.

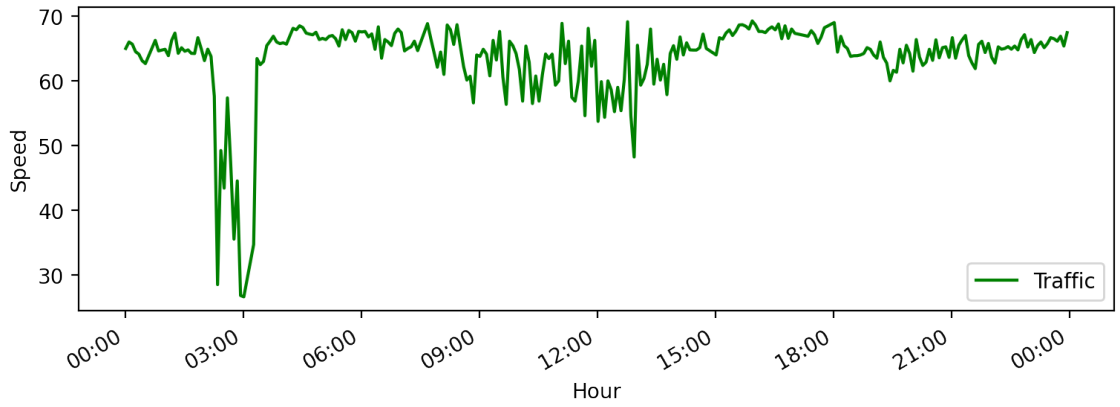
[1] Reset Gate

فصل ۳

روش پیش‌بینی ترافیک شهری با استفاده از یادگیری عمیق

در این فصل ابتدا به تعریف مسئله پیش‌بینی ترافیک شهری با استفاده از یادگیری عمیق پرداخته شده است و همچنین مدل پیشنهادی برای حل چالش‌های مورد بحث در حل این مسئله بیان شده که ترکیبی از شبکه‌های عصبی بازگشتی با واحدهای دروازه‌ای و شبکه پیچشی گراف است پرداخته شده است. در انتهای این فصل داده‌های آموزش و آزمایش نیز مورد بحث قرار گرفته است. مفاهیم فوق در بخش ۱-۳ تعریف شده‌اند.

هدف مدل مورد بررسی در این فصل، پیش‌بینی اطلاعات ترافیکی در بازه زمانی مشخصی است. منظور از اطلاعات ترافیکی، سرعت ترافیک برداشت شده در معبر شهری که متغیر تصمیم گیری در سنجش ترافیک یک معبر شهری است می‌باشد.این اطلاعات به‌شکل سرعت بازه‌ای از زمان مانند روز یا هفته در طول مدت زمان مشخص اندازه گیری شده است. به عنوان مثال شکل ۳-۱ نشان دهنده سرعت یک معبر در طول یک روز، با فاصله زمانی اندازه گیری هر ۵ دقیقه می‌باشد.



شکل ۳-۱: تغییرات سرعت ترافیک معبر شهری در طول یک روز

۳-۱ شبکه‌معابر شهری به عنوان داده ورودی مسئله

شبکه معابر شهری G یک گراف بدون وزن می‌باشد که گره‌های این گراف معابر شهری می‌باشند که در آن حسگرهای سنجش و اندازه گیری سرعت نصب شده است و یال‌های این گراف مسیرهای بین معابر شهری ذکر شده می‌باشند. یک معبر با معبری مجاور است که مسیر مستقیمی بدون معبر میانی بین این دو معبر شهری وجود داشته باشد.

برای تشریح ساختار توپولوژی شهری از نمایش گراف استفاده شده است، در گراف $G = (V, E)$ ، V نشانگر گره‌های این گراف یعنی مجموعه‌ای از معابر شهری که در آن حسگر نصب شده است می‌باشد:

$$V = \{v_1, v_2, ..., v_N\} \tag{۳-۱}$$

که در آن N نشان‌دهنده تعداد معابر شهری در شبکه معابر است.

E مجموعه ای از مسیرهای اتصال این معابر شهری است. ماتریس مجاورت A بیان‌گر نحوه اتصال معابر مختلف به یکدیگر می‌باشد، $A \in R^{N \times N}$. ماتریس مجاورت این گراف بدون وزن تنها دارای عناصر ۰ و ۱ می‌باشد، مقدار ۰ برای معابری که اتصال مستقیمی به یکدیگر ندارند و ۱ برای معابر دارای اتصال مستقیم به یکدیگر مورد استفاده قرار گرفته است.

۳-۲ سرعت ترافیک معابر به عنوان ماتریس ویژگی مسئله

ماتریس ویژگی (ویژگی ترافیکی) X به عنوان یک ماتریس ویژگی تعریف می‌شود که شامل ویژگی‌های معابر می‌باشد. ماتریس ویژگی دارای ابعاد $X \in R^{T \times N \times P}$ ، که در آن N نشان‌دهنده تعداد معابر، T تعداد لحظات تاریخچه‌ای و P تعداد ویژگی‌های موجود هر معبر در بررسی انجام شده می‌باشد. به‌عنوان مثال $X_t \in R^{N \times P}$ که نمایان‌گر ویژگی‌های معابر در لحظه t می‌باشد. در این ماتریس ویژگی دارای $T = 1$ است که سرعت ترافیک در لحظه فعلی تعریف شده است. داده‌های ماتریس ویژگی ساختار تاریخچه‌ای دارند که در بازه‌های زمانی مشخصی این داده‌ها اندازه‌گیری شده و نتایج در این

مجموعه داده ذکر شده‌است. بنابراین برای ماتریس ویژگی در لحظه t می‌توان $X(t)$ به شکل ذیل نمایش داد:

$$X(t)_{N\times P} = \begin{pmatrix} X_{۱,۱} & X_{۱,۲} & \cdots & X_{۱,P} \\ X_{۲,۱} & X_{۲,۲} & \cdots & X_{۲,P} \\ \vdots & \vdots & \ddots & \vdots \\ X_{N,۱} & X_{N,۲} & \cdots & X_{N,P} \end{pmatrix}$$

با توجه به تعریف ماتریس مجاورت شبکه معابر شهری (توپولوژی شبکه معابر شهری) و داده‌های تاریخچه‌ای معابر شهری (ماتریس ویژگی) تعریف مسئله ذکر شده با توجه با ساختار موجود دارای دو ورودی داده‌های توپولوژی معابر شهری A و ماتریس ویژگی داده‌های تاریخچه‌ای هر معبر X می‌باشد، بنابراین برای تابع پیش‌بینی ویژگی‌های ترافیکی معابر شهری در T لحظه بعد مفروض است:

$$[X_{t+۱}, ..., X_{t+T}] = f\Big(G, (X_{t-n}, ..., X_{t-۱}, X_t)\Big) \qquad (۲-۳)$$

که در آن n نشان‌دهنده طول داده‌های تاریخچه‌ای ترافیکی می‌باشد و T تعداد لحظات بعدی مورد نیاز برای پیش‌بینی است.

۳-۳ مدل ترکیبی پیش‌بینی ترافیک معابر شهری

شبکه ترکیبی پیش‌بینی ترافیک معابر شهری، از ترکیب دو شبکه عصبی پیچشی گراف و شبکه بازگشتی با واحدهای دروازه‌ای تشکیل شده می‌شود. از شبکه پیچشی گراف برای حل وابستگی مکانی و از شبکه بازگشتی با واحدهای دروازه‌ای به منظور حل وابستگی زمانی داده‌های ترافیکی معبر شهری استفاده شده است. با ترکیب شبکه عصبی بازگشتی با واحدهای دروازه‌ای و شبکه پیچشی گراف وابستگی مکانی-زمانی داده‌های مسئله قابل درک و پیش‌بینی می‌باشد.

۳-۳-۱ شبکه پیچشی گراف در حل وابستگی مکانی

همان‌طور که در بخش ۲-۲ به تفصیل بیان شد، شبکه‌های پیچشی گراف قابلیت درک وابستگی مکانی یک شبکه معابر شهری که به صورت ساختار گراف می‌باشد را دارد. برای استفاده از شبکه پیچشی گراف ترتیب فیلتر را بر اساس پیمایش اول سطح [1] هر گره اعمال شده انجام می‌گیرد. این ترتیب فیلتر با انجام عملیات بر روی ماتریس حاصل ضرب ماتریس مجاورت گراف و ماتریس درجات گراف صورت می‌پذیرد. در رابطه ۲-۴ با قراردادن تابع غیرخطی $ReLu(.)$ در لایه اول و

[1] first-order

تابع غیر خطی $\sigma(.)$ در لایه خروجی (دوم) نسبت به انجام عملیات پیچشی گراف اقدام می‌شود و در حاصل مفروض است:

$$H^{(l+1)} = \sigma\Big(\widehat{A}H^{(l)}W^{(l)}\Big) \tag{۳-۳}$$

$$H^{(0)} = X \tag{۴-۳}$$

$$H^{(1)} = ReLu\Big(\widehat{A}XW^{(0)}\Big) \tag{۵-۳}$$

$$H^{(2)} = \sigma\Big(\widehat{A}\ ReLu\big(\widehat{A}XW^{(0)}\big)W^{(1)}\Big) \tag{۶-۳}$$

که در آن X ماتریس ویژگی، $\widehat{A}$ ماتریس حاصل ضرب ماتریس مجاورت گراف و ماتریس درجات گراف، W_0, W_1 وزن ماتریس در لایه اول و لایه دوم و $\sigma(.), Relu(.)$ توابع فعال‌ساز است.

با استفاده از محاسبات انجام گرفته در ۳-۶ می‌توان تابع شبکه پیچشی گراف را بازنویسی کرد:

$$f(X, A) = \sigma(\widehat{A}Relu(\widehat{A}XW_0)W_1) \tag{۷-۳}$$

که در آن X ماتریس ویژگی، $\widehat{A}$ حاصل شرب ماتریس مجاورت گراف و ماتریس درجات گراف، W_0, W_1 وزن ماتریس در لایه اول و لایه دوم و $\sigma(.), Relu(.)$ توابع فعال‌ساز است.

با اعمال این روش، شبکه پیچشی گراف ویژگی‌های محلی یک گراف را مانند شبکه پیچشی در اختیار قرار خواهد گرفت. در واقع تابع $f(X, A)$ محاسبه شده در ۳-۷ با دریافت دو ورودی، ماتریس مجاورت گراف A و ماتریس ویژگی گراف، X وابستگی مکانی را برآورده می‌سازد. با استفاده از شبکه عصبی پیچشی گراف وابستگی مکانی را به مدل پیش‌بینی دریافت اعمال شده است و هر نقطه محلی به نقاط اطراف خود وابسته می‌شود [۳۰] .

۳-۳-۲ شبکه بازگشتی در حل وابستگی زمانی

مطابق تعریف در بخش ۲-۴ شبکه‌های عصبی بازگشتی با آموزش دیدن از داده‌های تاریخچه‌ای، قابلیت پیش‌بینی داده‌های آینده را دارند. شبکه‌های عصبی بازگشتی با داشتن حافظه، پیش‌بینی مطلوب‌تری بر روی داده‌هایی با خاصیت تاریخچه‌ای انجام می‌دهد. در مسئله پیش‌بینی ترافیک معابر شهری از شبکه عصبی بازگشتی با واحدهای دروازه‌ای استفاده شده است. این شبکه عصبی به جهت داشتن حافظه و همچنین پارامترهای آموزش نسبتا کمتر (نسبت به شبکه عصبی بازگشتی کوتاه مدت ماندگار(LSTM)) ، به عنوان بخشی از مدل ترکیبی پیش‌بینی ترافیک معابر انتخاب گردیده است.

شبکه‌عصبی بازگشتی با واحدهای دروازه‌ای خروجی شبکه عصبی پیچشی گراف را به عنوان ماتریس ویژگی دریافت و عملیات یادگیری شبکه را بر روی آن اعمال می‌کند. با توجه به روابط دروازه‌های سلول‌های شبکه عصبی بازگشتی دروازه‌ای بر حسب ماتریس ویژگی ساخته شده، روابط شبکه حاصل را می‌توان بازنویسی کرد.

برای دروازه بروزرسانی u_t با توجه به رابطه ۲-۸ و فرض ماتریس ویژگی برابر ۳-۷ مفروض است:

$$u_t = \sigma\big(W_u[f(X_t, A), h_{t-1}] + b_u\big) \tag{۳-۸}$$

که در آن W_u و b_u به ترتیب وزن و بایاس مربوط به دروازه بروزرسانی سلول، r_u مقدار خروجی دروازه بروزرسانی، $f(X_t, A)$ خروجی حاصل از شبکه عصبی پیچشی گراف و h_{t-1} خروجی سلول پیشین می‌باشد.

همچنین برای دروازه بازنشانی r_t با توجه به رابطه ۲-۹ و فرض ماتریس ویژگی برابر ۳-۷ روابط جدید مورد محاسبه قرار می‌گیرد:

$$r_t = \sigma\big(W_r[f(X_t, A), h_{t-1}] + b_r\big) \tag{۳-۹}$$

که در آن W_r و b_r به ترتیب وزن و بایاس مربوط به دروازه سلول، $f(X_t, A)$ خروجی حاصل از شبکه عصبی پیچشی گراف و h_{t-1} خروجی سلول پیشین می‌باشد.

و در نهایت نسبت به محاسبه آخرین دروازه سلول یعنی c_t که در رابطه ۲-۱۰ شرح داده شد، اقدام می‌شود:

$$c_t = \tanh\big(W_c[f(X_t, A), (r_t \times h_{t-1})] + b_c\big) \tag{۳-۱۰}$$

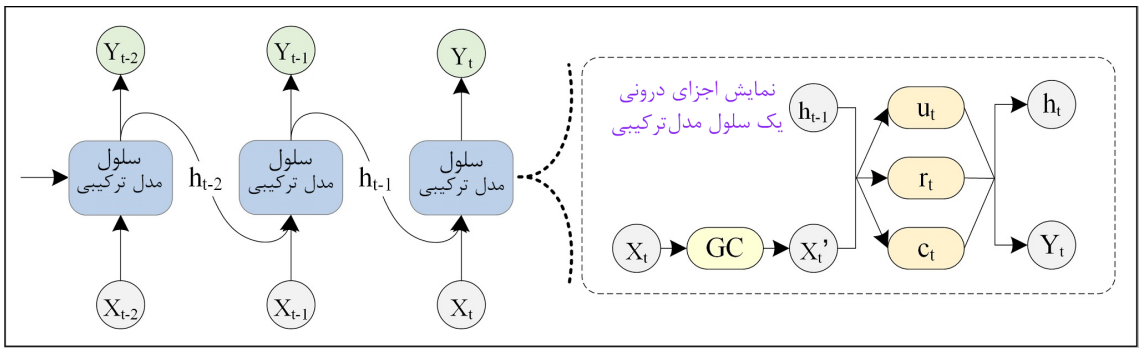
که در آن W_c و b_c به ترتیب وزن و بایاس مربوط به دروازه سلول، r_t مقدار خروجی دروازه بازنشانی، $f(X_t, A)$ خروجی حاصل از شبکه عصبی پیچشی گراف و h_{t-1} خروجی سلول پیشین می‌باشد.

با محاسبه مقادیر دروازه‌های شبکه عصبی بازگشتی دروازه‌ای، خروجی سلول t مطابق با رابطه ۲-۱۱ قابل محاسبه می‌باشد.

۳-۳-۳ ساختار سلول مدل ترکیبی

بنابر توصیفات ارائه شده در بخش‌های ۳-۳-۱ و ۳-۳-۲ نتیجه‌می‌دهد که از ترکیب دو شبکه پیچشی گراف و شبکه عصبی بازگشتی با واحدهای دروازه‌ای به مدلی با قابلیت پاسخگویی به وابستگی‌های زمانی ترافیک معابر و وابستگی مکانی بین معابر شهری حاصل می‌شود. همان‌طور که در شکل ۳-۲ نمایش داده شده در بخش سمت چپ هر سلول شبکه ترکیبی با داشتن ورودی هر لحظه و پاسخ‌های لحظات قبل نسبت به تصمیم گیری اقدام می‌کند. در هر سلول از مدل ترکیبی، بخش شبکه پیچشی گراف، وابستگی‌های مکانی را مورد ارزیابی قرار می‌دهد و خروجی این داده کد شده را به ورودی شبکه عصبی بازگشتی با سلول‌های دروازه‌ای که دارای حافظه هستند انتقال می‌دهد تا وابستگی زمانی نیز اعمال شود. بدین ترتیب با ورود داده‌های و پردازش آن توسط این شبکه ترکیبی، مدل آموزش دیده وابستگی زمانی و وابستگی مکانی را دریافت کرده و قابلیت

تشخیص ترافیک هر معبر را وابسته به مکان و ترافیک لحظات قبل آن دارد.



شکل ۳-۲: ساختار درونی سلول‌های مدل ترکیبی

۳-۳-۴ تابع هزینه

هدف این روش افزایش دقت پیش‌بینی ترافیک معابر است، کم کردن اختلاف بین میزان ترافیک پیش‌بینی شده $\widehat{y}$ و میزان واقعی ترافیک هدف y این مسئله می‌باشد. برای اندازه‌گیری دقت در درجه اول این میزان اختلاف مطلوب است $\|y - \widehat{y}\|$ و در درجه اهمیت بعدی باید از بیش برازش داده مدل جلوگیری شود، بدین ترتیب داریم:

$$loss = \|y - \widehat{y}\| + \lambda L_{reg} \qquad (۱۱-۳)$$

که در آن λ ابرپارامتر(فراپارامتر) [۱] و L_{reg} نرم مرتبه دوم [۲] است. از پارامتر L_{reg} به منظور جلوگیری از بیش‌برازش مدل استفاده می‌شود.

[1] Hyperparameter [2] L_2 Regularization

فصل ۴

نتایج پیاده‌سازی

پیاده‌سازی روش مدل ترکیبی پیشنهادی در این نوشتار به نتایج مطلوبی حاصل شده است. در این بخش به گزارشی از نتایج حاصل شده و همچنین مقایسه نتایج با دیگر روش‌های مطرح مورد بحث پرداخته شده است. محیط اجرا قطعه برنامه پیاده سازی شده و پارامترهای مدل پیش‌بینی ترافیک معابر شهری و همچنین داده مورد نیاز ورودی برای آموزش و سنجش روش ارائه شده به تفصیل در این بخش بیان می‌شود.

۴-۱ محیط اجرای برنامه

برنامه پیش‌بینی ترافیک شهری در یک دستگاه لپ‌تاپ با سیستم عامل [۱] لینوکس [۲] و توزیع اوبونتو (Ubuntu) نسخه 20.04.3 LTS از سری ۶۴ بیت اجرا شده است. میزان حافظه رام دستگاه اجرا کننده ۱۲ گیگابایت و میزان حافظه رام ثانویه [۳] با حجم گرفته شده ۱۲ گیگابایت می‌باشد. همچنین پردازنده مرکزی (CPU) دستگاه از مدل Intel Code i5-4200M با فرکانس ۲٫۵ گیگاهرتز و دارای ۴ هسته می‌باشد.

این برنامه در محیط زبان پایتون [۴] با استفاده از ابزار Tensorflow با نسخه 1.13.2 و به کمک برخی ابزارهای کتابخانه‌ای دیگر زبان پایتون شامل موارد زیر پیاده‌سازی شده است:

- numpy برای محاسبات ماتریسی و عملیات جبری
- matplotlib به‌منظور رسم نمودارها
- pandas جهت خواندن اطلاعات از مجموعه داده
- sklearn برای کمک گرفتن در اندازه گیری نتایج برنامه

[1]Operating System [2]Linux [3]Swapping Memory [4]Python

سورس و داده‌های آموزشی این برنامه در پیوند [1] موجود می‌باشد.

۴-۲ مجموعه داده

داده‌های مورد استفاده در برنامه، ساختار مکانی-زمانی دارند. مطابق شکل ۴-۱ یک شبکه معابر شهری با داشتن تاریخچه تغییرات سرعت در لحظه‌های مختلف به عنوان مجموعه داده ورودی استفاده می‌شود. در شکل ۴-۱ قسمتی از یک شبکه شهری نمونه نمایش داده شده است و همچنین داده یک روز از این معبر به بر روی نمودار سرعت - زمان رسم شده است.

شکل ۴-۱: طرح کلی از ساختار مجموعه داده

در پیاده سازی و آزمایش روش ارائه شده از دو مجموعه داده واقعی موجود استفاده شده است، که در نتایج پیش‌بینی

[1] https://github.com/Morteza-j8/TGCN-Urban-Traffic-Prediction-Tenssorflow

ویژگی سرعت ترافیک مورد آزمایش قرار گرفته است:

- **مجموعه داده تاکسی‌های شهر شنزن** [1]:

این مجموعه داده از یک شبکه شهری با ۱۵۶ معبر تشکیل شده است که داده‌های ترافیکی در هر معبر در ماه ژانویه سال ۲۰۱۵ میلادی هر ۱۵ دقیقه برداشته شده است. مجموعه داده ذکر شده در هر ساعت ۴ رکورد را شامل می‌شود که برای ۳۱ روز ماه ژانویه دارای $۲۹۷۶ = ۴ \times ۲۴ \times ۳۱$ داده تاریخچه‌ای سرعت می‌باشد.

$$A \in R^{۱۵۶ \times ۱۵۶} \quad (۱-۴)$$

$$X \in R^{۱۵۶ \times ۲۹۷۶ \times ۱} \quad (۲-۴)$$

که در آن A ماتریس مجاورت گراف و X ماتریس ویژگی راس‌ها می‌باشد.

- **مجموعه داده حسگرهای بزرگراهی شهر لس آنجلس** [2]:

حسگرهای کارگذاشته شده در این بزرگراه یک شبکه ۲۰۷ معبر را شامل می‌شوند که هر معبر در مدت ۵ دقیقه یک مرتبه داده سرعت ترافیکی را در آن ذخیره شده است. این داده‌های در بازه زمانی اول تا هفتم ماه مارس سال ۲۰۱۲ میلادی برداشت شده است. مجموعه داده حسگرهای بزرگراهی شهر لس آنجلس در هر ساعت ۱۲ رکورد را شامل می‌شود که در مجموع یک هفته موجود در این مجموعه داده $۲۰۱۶ = ۱۲ \times ۲۴ \times ۷$ داده تاریخچه‌ای سرعت وجود دارد. مشخصه‌های ورودی این مجموعه داده به شکل زیر می‌باشد:

$$A \in R^{۲۰۷ \times ۲۰۷} \quad (۳-۴)$$

$$X \in R^{۲۰۷ \times ۲۰۱۶ \times ۱} \quad (۴-۴)$$

که در آن A ماتریس مجاورت گراف و X ماتریس ویژگی راس‌ها می‌باشد.

ساختار ورودی داده‌ها به برنامه پیاده‌سازی شده به دو قسمت تبدیل می‌شود، قسمت اول مربوط به ماتریس مجاورت گراف و قسمت دوم مربوط به ماتریس ویژگی‌های گراف می‌باشد .

ماتریس مجاورت گراف در قالب ورودی به شکل یک فایل با فرمت csv شامل تعداد سطر و ستون برابر با تعداد راس‌ها می‌باشد. جدول ۴-۱ نمایش بخش کوچکی از این فایل می‌باشد. لازم به ذکر است که قالب ورودی نسبت به نوع مجموعه داده اندازه متفاوتی دارد. در مجموعه داده تاکسی‌های شهر شنزن اندازه جدول برابر ۱۵۶×۱۵۶ و در مجموعه داده حسگرهای بزرگراهی شهر لس آنجلس برابر ۲۰۷×۲۰۷ است.

[1] SZ-taxi [2] Los-Loop

راس	۰	۱	۲	۳	۴	۵	۶
۰	۰	۰	۱	۱	۰	۰	۰
۱	۰	۱	۰	۱	۰	۰	۰
۲	۰	۱	۱	۰	۱	۰	۰
۳	۰	۰	۰	۱	۰	۰	۰
۴	۰	۰	۰	۰	۰	۰	۱
۵	۰	۰	۰	۰	۰	۱	۰
۶	۰	۰	۰	۰	۰	۱	۱

جدول ۴-۱: نمونه داده ورودی ماتریس مجاورت شبکه معابر

ساختار دومین داده ورودی از این مجموعه داده‌ها مربوط به ماتریس ویژگی‌ها می‌باشد. که به منظور ماتریس ویژگی مورد استفاده قرار می‌گیرد. این ورودی نیز به فرمت یک فایل csv با اندازه $T \times N$ است که در آن T سطر برابر تعداد لحظات مورد سنجش و N ستون تعداد معابر مورد اندازه‌گیری می‌باشند.

معبر / لحظه	۱	۲	...	N
۱	۶۴٫۳۷۵	۶۷٫۶۲۵	...	۶۱٫۸۷۵
۲	۶۲٫۶۶۶	۶۸٫۵۵۵	...	۶۲٫۸۷۵
⋮	⋮	⋮	⋱	⋮
T	۶۶	۶۷٫۱۲۵	...	۵۸٫۸۷۵

جدول ۴-۲: ساختار ماتریس ویژگی ورودی برای آموزش مدل

در حل مسئله سرعت هر معبر به عنوان ویژگی معبر مورد نظر در لحظات مختلف ثبت شده است. مطابق شکل ۴-۲ برای هر معبر در T لحظه اندازه‌گیری سرعت انجام گرفته است. در مجموعه داده حسگرهای بزرگراهی شهر لس آنجلس ۷ روز در هر ۵ دقیقه یک مرتبه (هر ساعت ۱۲ مرتبه) داده سرعت معابر ثبت شده که تشکیل تاریخچه دنباله ای با اندازه $T = ۷ \times ۲۴ \times ۱۲ = ۲۰۱۶$ و مجموعه داده تاکسی‌های شهر شنزن با اندازه هر ۱۵ دقیقه یگ مرتبه (هر ساعت ۴ مرتبه) در ۳۱ روز از ماه ژانویه، تاریخچه‌ای با اندازه $T = ۳۱ \times ۲۴ \times ۴ = ۲۹۷۶$ وجود دارد.

از ۸۰ درصد داده‌های هر مجموعه برای آموزش و از ۲۰ درصد باقی‌مانده هر مجموعه داده برای آزمایش مدل ترکیبی

پیش‌بینی ترافیک معابر شهری استفاده می‌شود. این داده‌های در حالت‌های مختلف تجمیع بازه‌های زمانی ۱۵، ۳۰، ۴۵ و ۶۰ دقیقه مورد آموزش و پیش‌بینی قرار داده شده است.

۴-۳ پارامترهای مدل GRU در پیش‌بینی ترافیک معابر شهری

در اجرای برنامه پیش‌بینی ترافیک معابر شهری با استفاده از شبکه عصبی بازگشتی با واحدهای دروازه‌ای (GRU) از نرخ یادگیری ۰٫۰۰۱ ، اندازه دسته (Batch Size) ۳۲ ، ۲۰۰۰ مرحله آموزش (Epoch) با ۶۴ سلول در شبکه عصبی بازگشتی دروازه‌ای (GRU) استفاده شده است.

مطابق روش‌های یادگیری عمیق و شبکه عصبی، وزن‌ها و بایاس‌های موجود در مدل با استفاده از توابع بهینه‌سازی بروزرسانی شده‌است. در فرآیند آموزش مدل پیش‌بینی ترافیک معابر شهری پارامترها طی رفتار عقب‌گرد و بهینه‌سازی بهبود یافته و مقدار آن‌ها محاسبه گردیده است.

داده‌های تاریخچه‌ای در بازه‌های ۱۵، ۳۰، ۴۵ و ۶۰ تجمیع و نتایج آزمایش به تشریح در جدول‌های ۴-۳ و ۴-۴ نمایش داده شده است. در پیاده‌سازی مدل پیش‌بینی ترافیک معابر شهری از تاریخچه یک ساعت گذشته یک معبر، برای آموزش و تأثیر بر روی ترافیک فعلی آن معبر استفاده می‌شود. درمجموعه داده حسگرهای بزرگراهی لس آنجلس(۴-۲) با توجه به وجود داده تاریخچه‌ای در هر ۵ دقیقه از ۱۲ سرعت ترافیک قبلی در عملیات پیش‌بینی استفاده شده است و در مجموعه داده تاکسی‌های شهر شنزن چین(۴-۲) با توجه به وجود داده تاریخچه‌ای در هر ۱۵ دقیقه از ۴ جمله از دنباله تاریخچه سرعت در معبر مورد استفاده قرار می‌گیرد.

۴-۴ پارامترهای مدل ARIMA در پیش‌بینی ترافیک معابر شهری

در اجرای برنامه پیش‌بینی ترافیک معابر شهری با استفاده از مدل اتورگرسیو-میانگین متحرک جمع‌بسته (ARIMA) با اتورگرسیو درجه اول ($p = ۱$)، درجه یکپارچه‌سازی صفر ($d = ۰$) و میانگین متحرک درجه صفر ($q = ۰$) استفاده شده است. تابع اتورگرسیو-میانگین متحرک جمع‌بسته استفاده شده به شکل $ARIMA(۱, ۰, ۰)$ تعریف می‌شود.

داده‌های تاریخچه‌ای در بازه‌های ۱۵، ۳۰، ۴۵ و ۶۰ تجمیع و نتایج آزمایش به تشریح در جدول‌های ۴-۳ و ۴-۴ نمایش داده شده است. در پیاده‌سازی مدل پیش‌بینی ترافیک معابر شهری با استفاده از اتورگرسیو-میانگین متحرک جمع‌بسته، هر معبر بدون وابستگی به معابر دیگر مورد ارزیابی قرارگرفته است و مجموعه داده تاریخچه‌ای هر معبر به عنوان سری زمانی مورد استفاده در مدل ARIMA به کار گرفته شده است.

۴-۵ پارامترهای مدل ترکیبی پیش‌بینی ترافیک معابر شهری

در اجرای برنامه پیش‌بینی ترافیک معابر شهری از نرخ یادگیری ۰/۰۰۱ ، اندازه دسته (Batch Size) ۳۲ ، ۲۰۰۰ مرحله آموزش (Epoch) با ۶۴ سلول در شبکه عصبی بازگشتی دروازه‌ای (GRU) و ۲ لایه پنهان در شبکه عصبی پیچشی گراف (GCN) استفاده شده است.

مطابق روش‌های یادگیری عمیق و شبکه عصبی، وزن‌ها و بایاس‌های موجود در مدل با استفاده از توابع بهینه‌سازی بروزرسانی شده‌است. در فرآیند آموزش مدل پیش‌بینی ترافیک معابر شهری ۱۲۹۳۲ پارامتر طی رفتار عقب‌گرد و بهینه‌سازی بهبود یافته و مقدار آن‌ها محاسبه گردیده است.

داده‌های تاریخچه‌ای در بازه‌های ۱۵، ۳۰، ۴۵ و ۶۰ تجمیع و نتایج آزمایش به تشریح در جدول‌های ۴-۳ و ۴-۴ نمایش داده شده است. در پیاده‌سازی مدل پیش‌بینی ترافیک معابر شهری از تاریخچه یک ساعت گذشته یک معبر، برای آموزش و تأثیر بر روی ترافیک فعلی آن معبر استفاده می‌شود. درمجموعه داده حسگرهای بزرگراهی لس آنجلس(۴-۲) با توجه به وجود داده تاریخچه‌ای در هر ۵ دقیقه از ۱۲ سرعت ترافیک قبلی در عملیات پیش‌بینی استفاده شده است و در مجموعه داده تاکسی‌های شهر شنزن چین(۴-۲) با توجه به وجود داده تاریخچه‌ای در هر ۱۵ دقیقه از ۴ جمله از دنباله تاریخچه سرعت در معبر مورد استفاده قرار می‌گیرد.

۴-۶ نتایج پیاده سازی

نتایج مدل ترکیبی بیان شده بر روی هر دو مجموعه داده ذکر شده در بازه های تجمیع شده ۱۵، ۳۰، ۴۵و ۶۰ دقیقه با دو معیار جذر میانگین مربعات خطا و میانگین خطای مطلق در مقایسه با دو روش میانگین متحرک خودهمبسته یکپارچه و شبکه عصبی بازگشتی با واحدهای دروازه‌ای محاسبه شده است.

		$RMSE$
	$ARIMA$	۸/۲۱۵۱
۱۵min	GRU	۴/۰۴۸۳
	مدل ترکیبی	۳/۹۱۶۲
	$ARIMA$	۸/۲۱۲۳
۳۰min	GRU	۴/۰۷۶۹
	مدل ترکیبی	۳/۹۶۱۷
	$ARIMA$	۸/۲۱۳۲
۴۵min	GRU	۴/۱۰۰۲
	مدل ترکیبی	۳/۹۹۵
	$ARIMA$	۸/۲۰۶۳
۶۰min	GRU	۴/۱۲۴۱
	مدل ترکیبی	۴/۰۱۴۱

جدول ۴-۳: مقایسه نتایج مدل ترکیبی برو روی داده‌های شهر شنزن

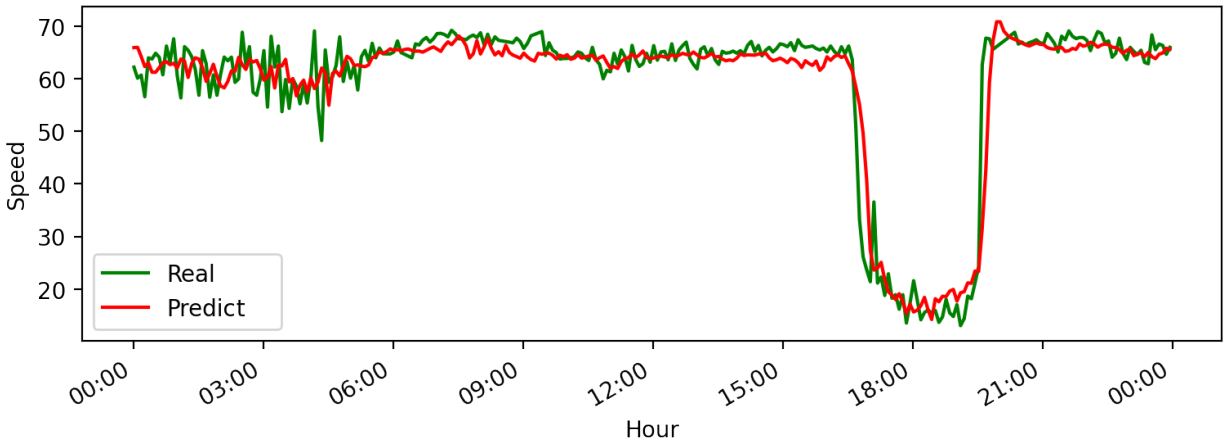
در جدول نتایج ۴-۳ نتایج برو روی مجموعه داده تاکسی‌های شهر شنزن چین انجام گرفته است. در این نتایج مقایسه مدل ترکیبی با دو مدل شبکه عصبی بازگشتی دروازه‌ای GRU و مدل اتورگرسیو-میانگین متحرک جمع بسته ARIMA نمایش داده شده است. در این بررسی در ۱۵۶ معبر شهری موجود در مجموعه داده ، ۸۰ درصد از داده‌های ورودی به عنوان آموزشی و ۲۰ درصد از داده‌های ورودی مسئله به عنوان داده آموزش مورد ارزیابی قرار گرفته است. درنتایج ثبت شده، با توجه به پیش‌بینی ترافیک هر معبر به‌صورت مجزا، منظور از مقدار $RMSE$ ، میانگین مقادیر $RMSE$ برای تمامی معابر شهری (۱۵۶ معبر) مورد بررسی در مجموعه داده است.

		$RMSE$
	$ARIMA$	۱۰٫۰۴۳۹
۱۵min	GRU	۵٫۲۱۸۲
	مدل ترکیبی	۵٫۱۲۶۴
	$ARIMA$	۹٫۳۴۵
۳۰min	GRU	۶٫۲۸۰۲
	مدل ترکیبی	۶٫۰۵۹۸
	$ARIMA$	۹٫۳۴۵
۴۵min	GRU	۷٫۰۳۴۳
	مدل ترکیبی	۶٫۷۰۶۵
	$ARIMA$	۱۰٫۰۵۳۸
۶۰min	GRU	۷٫۶۶۲۱
	مدل ترکیبی	۷٫۲۶۷۷

جدول ۴-۴: مقایسه نتایج مدل ترکیبی برو روی داده‌های بزرگراهی لس آنجلس

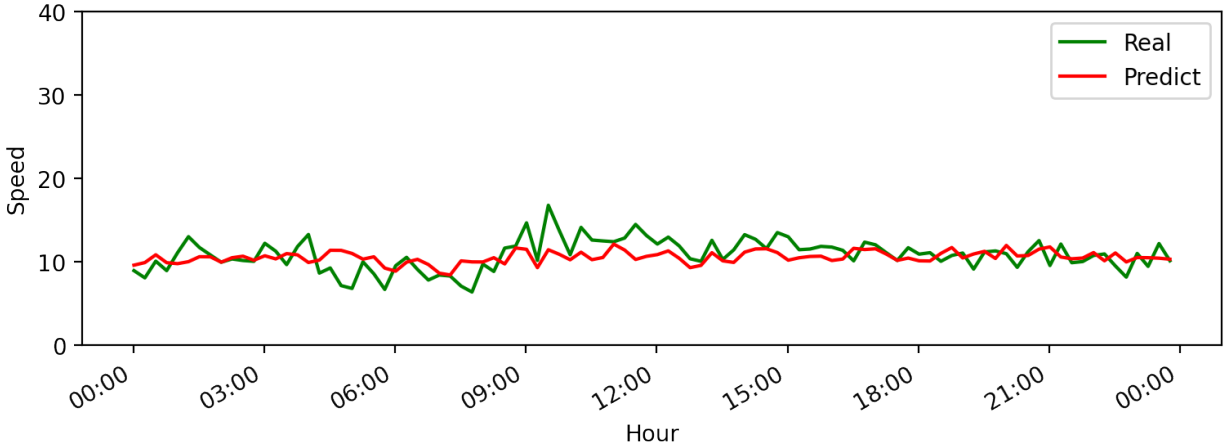
در جدول نتایج ۴-۴ نتایج برو روی مجموعه داده حسگرهای بزرگراهی شهر لس آنجلس انجام گرفته است. در این نتایج مقایسه مدل ترکیبی با دو مدل شبکه عصبی بازگشتی دروازه‌ای GRU و مدل اتورگرسیو-میانگین متحرک جمع بسته ARIMA نمایش داده شده است. در این بررسی در ۲۰۷ معبر شهری موجود در مجموعه داده ، ۸۰ درصد از داده‌های ورودی به عنوان آموزشی و ۲۰ درصد از داده‌های ورودی مسئله به عنوان داده آموزش مورد ارزیابی قرار گرفته است. درنتایج ثبت شـده، با توجه به پیش‌بینی ترافیک هر معبر به‌صورت مجزا، منظور از مقدار $RMSE$ ، میانگین مقادیر $RMSE$ برای تمامی معابر شهری (۲۰۷ معبر) مورد بررسی در مجموعه داده است.

برای هر معبر امکان پیش‌بینی لحظات مختلف برقرار می‌باشد، با داشتن مدلی با قابلیت پاسخ‌گویی به لحظات آینده، امکان پیش‌بینی ترافیک در مدت زمان‌های مختلف برقرار است .شکل ۴-۲ نمایش پیش‌بینی ترافیک در معبر شهری نمونه‌ای مجموعه داده حسگرهای بزرگراهی شهر لس آنجلس و شکل ۴-۳ نمونه‌ای از پیش‌بینی مدل ترکیبی از ترافیک یک روز از مجموعه داده تاکسی‌های شهر شنزن می‌باشد.



شکل ۴-۲: ترافیک پیش‌بینی شده معبر شهری از مجموعه داده بزرگراهی لس آنجلس در طول یک روز

در شکل‌های ۴-۲ و ۴-۳ دو مقدار ترافیک واقعی معبر که موجود در مجموعه داده می‌باشد و میزان ترافیک پیش‌بینی شده توسط مدل ترکیبی، در طول مدت یک روز نمایش داده شده است.

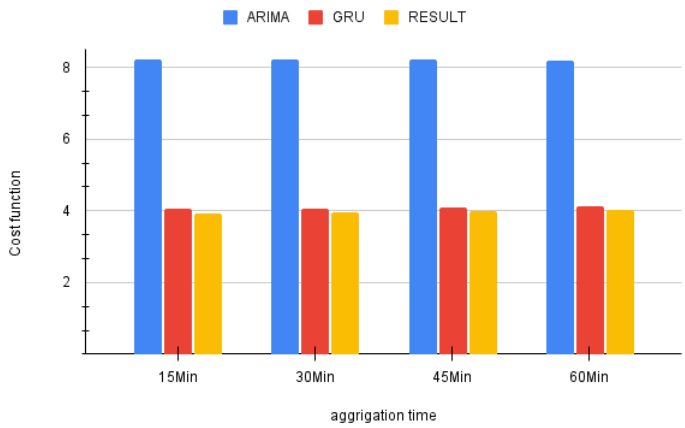


شکل ۴-۳: ترافیک پیش‌بینی شده معبر شهری از مجموعه داده تاکسی‌های شهر شنزن در طول یک روز

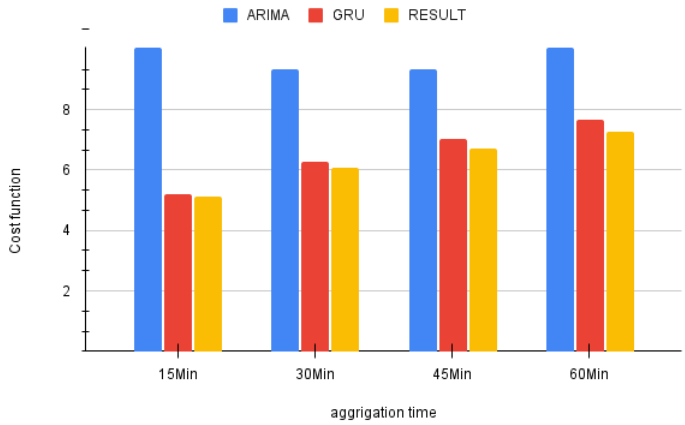
علت تراکم کم شکل ۴-۲ نسبت به شکل ۴-۳ به خاطر تفاوت مدت نمونه برداری در مجموعه داده‌ها می‌باشد. در مجموعه داده تاکسی‌های شهر شنزن چین در هر معبر شهری ترافیک ۱۵ دقیقه یک مرتبه برداشت شده است، در حالی که در مجموعه داده حسگرهای بزرگراهی شهر لس آنجلس داده‌ها هر ۵ دقیقه یک مرتبه در دسترس می‌باشد. در شکل‌های نمایش داده شده از این دو مجموعه داده برای مجموعه داده تاکسی‌های شهر شنزن چین ۹۶ (۴ × ۲۴) پیش‌بینی و برای مجموعه داده حسگرهای بزرگراهی شهر لس آنجلس ۲۸۸ (۱۲ × ۲۴) پیش‌بینی انجام گرفته است که در شکل‌های ۴-۲ و ۴-۳ مشاهده می‌شود.

پس از پیاده‌سازی و اندازه گیری نتایج مدل ارائه شده با تابع سنجش خطا RMSE بر روی دو مجموعه داده تاکسی‌های شهر شنزن و حسگرهای بزرگراهی شهر لس‌آنجلس مقایسه انجام شده است. شکل ۴-۴ مقایسه نتایج مدل ارائه شده با

معیار خطای RMSE بر روی مجموعه داده تاکسی‌های شهر شنزن و شکل ۴-۵ با در نظر گرفتن معیار خطای RMSE بر روی مجموعه داده حسگرهای بزرگراهی شهر لس آنجلس است.



شکل ۴-۴: مقایسه مدل ترکیبی با داده شهر شنزن با استفاده از تابع خطا RMSE



شکل ۴-۵: مقایسه مدل ترکیبی با داده شهر لس‌آنجلس با استفاده از تابع خطا RMSE

۴-۷ جمع بندی

در روش ارائه شده، شبکه‌ای مبتنی بر دو ارزیابی زمانی – مکانی مورد بررسی قرار گرفت که ابتدا داده‌ها به یک شبکه پیچشی گرافی داده شد و سپس توسط یک شبکه عصبی بازگشتی با واحدهای دروازه‌ای مورد یادگیری قرار گرفته است. برخلاف دیگر روش ها که یک شبکه وابسته به زمان را مورد ارزیابی قرار می‌دهند این شبکه ساختار توپولوژی و ترافیک لحظه‌ای را همزمان در مدل آموزشی اعمال می‌کند. این روش محدود به پیش‌بینی ترافیکی نیست و می‌تواند در سایر زمینه‌هایی که دو ساختار داده زمان و مکان در آن دخیل است نیز پاسخ دهی بسیار خوبی داشته باشد.

فهرست منابع

[1] Xu, Xin-yue, Liu, Jun, Li, Hai-ying, and Hu, Jian-Qiang. Analysis of subway station capacity with the use of queueing theory. Transportation research part C: emerging technologies, 38:28–43, 2014.

[2] Wei, P, Cao, Y, and Sun, D. Total unimodularity and decomposition method for large-scale air traffic cell transmission model. Transportation Research Part B: Methodological, 53:1–16, 2013.

[3] Chen, Wentian, Fang, Jie, Liu, Zhijia, and Xu, Mengyun. A key path-based deep learning approach for urban traffic speed prediction. in Journal of Physics: Conference Series, vol. 1972, p. 012095. IOP Publishing, 2021.

[4] Xu, Fei-Fei, He, Zhao-Cheng, and Sha, ZR. Impacts of traffic management measures on urban network microscopic fundamental diagram. Journal of Transportation Systems Engineering and Information Technology, 13(2):185–190, 2013.

[5] Zhao, Ling, Song, Yujiao, Zhang, Chao, Liu, Yu, Wang, Pu, Lin, Tao, Deng, Min, and Li, Haifeng. T-gcn: A temporal graph convolutional network for traffic prediction. IEEE Transactions on Intelligent Transportation Systems, 21(9):3848–3858, 2019.

[6] Liu, Jing and Guan, Wei. A summary of traffic flow forecasting methods [j]. Journal of highway and transportation research and development, 3:82–85, 2004.

[7] Ahmed, Mohammed S and Cook, Allen R. Analysis of freeway traffic time-series data by using Box-Jenkins techniques. no. 722. 1979.

[8] Hamed, Mohammad M, Al-Masaeid, Hashem R, and Said, Zahi M Bani. Short-term prediction of traffic volume in urban arterials. Journal of Transportation Engineering, 121(3):249–254, 1995.

[9] Van Der Voort, Mascha, Dougherty, Mark, and Watson, Susan. Combining kohonen maps with arima time series models to forecast traffic flow. Transportation Research Part C: Emerging Technologies, 4(5):307–318, 1996.

[10] Lee, Sangsoo and Fambro, Daniel B. Application of subset autoregressive integrated moving average model for short-term freeway traffic volume forecasting. Transportation Research Record, 1678(1):179–188, 1999.

[11] Williams, Billy M and Hoel, Lester A. Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results. Journal of transportation engineering, 129(6):664–672, 2003.

[12] Lippi, Marco, Bertini, Matteo, and Frasconi, Paolo. Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning. IEEE Transactions on Intelligent Transportation Systems, 14(2):871–882, 2013.

[13] Sun, Hongyu, Zhang, Chunming, and Ran, Bin. Interval prediction for traffic time series using local linear predictor. in Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No. 04TH8749), pp. 410–415. IEEE, 2004.

[14] Okutani, Iwao and Stephanedes, Yorgos J. Dynamic prediction of traffic volume through kalman filtering theory. Transportation Research Part B: Methodological, 18(1):1–11, 1984.

[15] Zhang, Xiao-li, He, Guo-guang, and Lu, Hua-pu. Short-term traffic flow forecasting based on k-nearest neighbors non-parametric regression. Journal of Systems Engineering, 24(2):178–183, 2009.

[16] Wu, Chun-Hsin, Ho, Jan-Ming, and Lee, Der-Tsai. Travel-time prediction with support vector regression. IEEE transactions on intelligent transportation systems, 5(4):276–281, 2004.

[17] Smola, Alex J and Schölkopf, Bernhard. A tutorial on support vector regression. Statistics and computing, 14(3):199–222, 2004.

[18] Yin, Hongbin, Wong, S_C, Xu, Jianmin, and Wong, CK. Urban traffic flow prediction using a fuzzy-neural approach. Transportation Research Part C: Emerging Technologies, 10(2):85–98, 2002.

[19] Sun, Shiliang, Zhang, Changshui, and Yu, Guoqiang. A bayesian network approach to traffic flow forecasting. IEEE Transactions on intelligent transportation systems, 7(1):124–132, 2006.

[20] Silver, David, Schrittwieser, Julian, Simonyan, Karen, Antonoglou, Ioannis, Huang, Aja, Guez, Arthur, Hubert, Thomas, Baker, Lucas, Lai, Matthew, Bolton, Adrian, et al. Mastering the game of go without human knowledge. nature, 550(7676):354–359, 2017.

[21] Moravčík, Matej, Schmid, Martin, Burch, Neil, Lisý, Viliam, Morrill, Dustin, Bard, Nolan, Davis, Trevor, Waugh, Kevin, Johanson, Michael, and Bowling, Michael. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. Science, 356(6337):508–513, 2017.

[22] Fu, Rui, Zhang, Zuo, and Li, Li. Using lstm and gru neural network methods for traffic flow prediction. in 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC), pp. 324–328. IEEE, 2016.

[23] Park, Dongjoo and Rilett, Laurence R. Forecasting freeway link travel times with a multilayer feedforward neural network. Computer-Aided Civil and Infrastructure Engineering, 14(5):357–367, 1999.

[24] Van Lint, JWC, Hoogendoorn, SP, and van Zuylen, Henk J. Freeway travel time prediction with state-space neural networks: modeling state-space dynamics with recurrent neural networks. Transportation Research Record, 1811(1):30–39, 2002.

[25] Lv, Yisheng, Duan, Yanjie, Kang, Wenwen, Li, Zhengxi, and Wang, Fei-Yue. Traffic flow prediction with big data: a deep learning approach. IEEE Transactions on Intelligent Transportation Systems, 16(2):865–873, 2014.

[26] Zhang, Junbo, Zheng, Yu, and Qi, Dekang. Deep spatio-temporal residual networks for citywide crowd flows prediction. in Thirty-first AAAI conference on artificial intelligence, 2017.

[27] Yu, Haiyang, Wu, Zhihai, Wang, Shuqin, Wang, Yunpeng, and Ma, Xiaolei. Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks. Sensors, 17(7):1501, 2017.

[28] Ke, Jintao, Zheng, Hongyu, Yang, Hai, and Chen, Xiqun Michael. Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach. Transportation Research Part C: Emerging Technologies, 85:591–608, 2017.

[29] Wu, Yuankai and Tan, Huachun. Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework. arXiv preprint arXiv:1612.01022, 2016.

[30] Kipf, Thomas N and Welling, Max. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907, 2016.

[31] Li, Yaguang, Yu, Rose, Shahabi, Cyrus, and Liu, Yan. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. arXiv preprint arXiv:1707.01926, 2017.

[32] Kingma, Diederik P and Ba, Jimmy. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[33] Bengio, Yoshua, Simard, Patrice, and Frasconi, Paolo. Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks, 5(2):157–166, 1994.

[34] Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[35] Cho, Kyunghyun, Van Merriënboer, Bart, Bahdanau, Dzmitry, and Bengio, Yoshua. On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259, 2014.

پیوست آ

برنامه‌های استفاده شده در این پایان‌نامه

در این بخش برنامه‌های استفاده شده در این پایان‌نامه آورده شده است. برخی برنامه‌های موجود برای ارزیابی نتایج روش پیش‌بینی ترافیک معابر شهری و بعضی دیگر برای نمایش بخش‌های نموداری و زیر برنامه‌های استفاده شده در این نوشتار می‌باشد.

آ-۱ برنامه نمایش توابع فعال‌سازی

این برنامه برای نمایش توابع فعال‌سازی با زبان پایتون نوشته شده است. که در آن توابع زیر پیاده‌سازی و نمودار هریک در بازه مشخص و محدودی نمایش داده می‌شود:

- sigmoid
- tanh
- ReLU
- ELU
- swish

برنامه آ-۱: نمایش توابع فعال‌سازی

```
                                                                    ۱
                                                                    ۲
import matplotlib.pyplot as plt                                     ۳
import numpy as np                                                  ۴
from math import e                                                  ۵
import math                                                         ۶
                                                                    ۷
                                                                    ۸
                                                                    ۹
def relu(x):                                                        ۱۰
  ans = [];                                                         ۱۱
  for i in x:                                                       ۱۲
        cur = max(0, i)                                             ۱۳
        ans.append(cur)                                             ۱۴
  return np.array(ans)                                              ۱۵
                                                                    ۱۶
                                                                    ۱۷
                                                                    ۱۸
def sigmoid_part(x):                                                ۱۹
  return 1 / (1 + e**(-x))                                          ۲۰
                                                                    ۲۱
def sigmoid(x):                                                     ۲۲
  ans = [];                                                         ۲۳
  for i in x:                                                       ۲۴
        ans.append(sigmoid_part(i))                                 ۲۵
  return np.array(ans)                                              ۲۶
                                                                    ۲۷
                                                                    ۲۸
def tanh(x):                                                        ۲۹
  ans = [];                                                         ۳۰
  for i in x:                                                       ۳۱
        cur = (e**i - e**(-i) ) / (e**i + e**(-i))                  ۳۲
        ans.append(cur)                                             ۳۳
  return np.array(ans)                                              ۳۴
                                                                    ۳۵
                                                                    ۳۶
def elu(x, alpha):                                                  ۳۷
  ans = [];                                                         ۳۸
  for i in x:                                                       ۳۹
        cur = i                                                     ۴۰
        if i <= 0:                                                  ۴۱
          cur = alpha*(e**i - 1)                                    ۴۲
        ans.append(cur)                                             ۴۳
  return np.array(ans)                                              ۴۴
                                                                    ۴۵
                                                                    ۴۶
def swish(x):                                                       ۴۷
  ans = [];                                                         ۴۸
  for i in x:                                                       ۴۹
        cur = i * sigmoid_part(i)                                   ۵۰
        ans.append(cur)                                             ۵۱
  return np.array(ans)                                              ۵۲
                                                                    ۵۳
                                                                    ۵۴
```

```
### Relu (x)
x = np.arange(-10, +10, 0.01)
plt.figure(figsize=(8, 5), dpi=200)
plt.xlabel('x', )
plt.ylabel('ReLU(x)')
plt.plot(x, relu(x), '#3FDFBF' , label="ReLU(x)")
plt.legend(loc="best")
plt.grid()
plt.show()


### sigmoid (x)
x = np.arange(-10, +10, 0.01)
plt.figure(figsize=(8, 5), dpi=200)
plt.xlabel('x', )
plt.ylabel('sigmoid(x)')
plt.plot(x, sigmoid(x), '#3F5FBF' , label="sigmoid(x)")
plt.legend(loc="best")
plt.grid()
plt.show()


### tanh (x)
x = np.arange(-3, +3, 0.01)
plt.figure(figsize=(8, 5), dpi=200)
plt.xlabel('x', )
plt.ylabel('tanh(x)')
plt.plot(x, tanh(x), '#CF5F2F' , label="tanh(x)")
plt.legend(loc="best")
plt.grid()
plt.show()


### ELU (x)
x = np.arange(-10, +10, 0.01)
plt.figure(figsize=(8, 5), dpi=200)
y = elu(x, 1)
plt.xlabel('x', )
plt.ylabel('ELU(x, alpha)')
plt.plot(x, elu(x,  1), '#3F3FAF' , label="Elu(x,1)")
plt.plot(x, elu(x,  2), '#3FAF3F' , label="Elu(x,2)")
plt.plot(x, elu(x,  3), '#AF3F3F' , label="Elu(x,3)")
plt.plot(x, elu(x,  5), '#3FAFAF' , label="Elu(x,5)")
plt.plot(x, elu(x, 10), '#AF3FAF' , label="Elu(x,10)")
plt.legend(loc="best")
plt.grid()
plt.show()


### swish (x)
x = np.arange(-10, +10, 0.01)
t=swish(x)
x = np.arange(-10, +2, 0.00001)
y = swish(x);
plt.figure(figsize=(8, 5), dpi=200)
plt.xlabel('x', )
```

```
plt.ylabel('swish(x)')
plt.plot(x, y, '#3FBF7F' , label="swish(x)")
plt.legend(loc="best")
plt.grid()
plt.show()
```

آ-۲ برنامه شبکه پیچشی گراف بدون عملکرد عقبگرد

این برنامه جهت معرفی یک شبکه پیچشی گراف (GCN) بدون عملکرد عقبگرد برای اصلاح وزن‌های با زبان پایتون با استفاده از روابط شبکه پیچشی گراف پیاده‌سازی شده است.

برنامه آ-۲: برنامه شبکه پیچشی گراف بدون عملکرد عقبگرد

```

import numpy as np
from scipy.linalg import fractional_matrix_power
import networkx as nx
import matplotlib.pyplot as plt

#defuine graph
G = nx.Graph(name='G')
for i in range(6):
  G.add_node(i, name=i)

#Define the edges and the edges to the graph
#edges = [(0,1),(0,2),(1,2),(0,3),(3,4),(3,5),(4,5)]
edges = [(0,1),(0,2),(0,3),(3,4),(3,5)]


G.add_edges_from(edges)


#Plot the graph
randomColors = ['#ff9090', '#ffd186', '#97b2fe', '#a2ff9f', '#ffb6f6
    ', '#b6fbff']
color_map = []
for iColor, node in enumerate(G):
    color_map.append(randomColors[iColor % len(randomColors)])
nx.draw(G, node_color=color_map, with_labels=True, node_size=1000)
plt.show()


A = np.array(nx.attr_matrix(G, node_attr='name')[0])
X = np.array(nx.attr_matrix(G, node_attr='name')[1])



X = np.expand_dims(X,axis=1)

#Implement ReLu as activation function
```

```
def relu(x):
    return np.maximum(0,x)



#Initialize the weights
np.random.seed(77777)
n_h = 4 #number of neurons in the hidden layer
n_y = 2 #number of neurons in the output layer
W0 = np.random.randn(X.shape[1],n_h) * 0.01
W1 = np.random.randn(n_h,n_y) * 0.01



#Build GCN layer
#In this function, we implement numpy to simplify
def gcn(A,H,W):
    I = np.identity(A.shape[0]) #create Identity Matrix of A
    A_hat = A + I #add self-loop to A
    D = np.diag(np.sum(A_hat, axis=0)) #create Degree Matrix of A
    D_half_norm = fractional_matrix_power(D, -0.5) #calculate D to
        the power of -0.5
    eq1 = D_half_norm.dot(A_hat)
    eq2 = eq1.dot(D_half_norm)
    eq3 = eq2.dot(H)
    eq4 = eq3.dot(W)

    return relu(eq4)


#Do forward propagation
H1 = gcn(A,X,W0)
H2 = gcn(A,H1,W1)
print('Features Representation from GCN output:\n', H2)


def plot_features(H2):
    #Plot the features representation
    x = H2[:,0]
    y = H2[:,1]


    size = 1000

    plt.scatter(x,y,size)
    #plt.xlim([np.min(x)*0.85, np.max(x)*1.1])
    #plt.ylim([-1, 1])
    plt.xlabel('Dimension 0')
    plt.ylabel('Dimension 1')

    for i,row in enumerate(H2):
        str = "{}".format(i)
        plt.annotate(str, (row[0],row[1]),fontsize=18, fontweight='
            bold', color = 'w'  )


```

```
    plt.show()
plot_features(H2)
```

آ-۳ برنامه مدل ترکیبی پیش‌بینی ترافیک معابر شهری

این برنامه حاصل پیاده‌سازی مدل ترکیبی پیش‌بینی ترافیک معابر شهری شرح داده شده در این پایان‌نامه می‌باشد. برنامه ذکر شده در محیط زبان برنامه نویسی پایتون با استفاده از ابزار Tenssorflow نوشته شده است.

برنامه آ-۳: برنامه مدل ترکیبی پیش‌بینی ترافیک معابر شهری

```
# -*- coding: utf-8 -*-
import pickle as pkl
import tensorflow as tf
import pandas as pd
import numpy as np
import math
import os
import numpy.linalg as la

from sklearn.metrics import mean_squared_error,mean_absolute_error
#import matplotlib.pyplot as plt
import time

import tensorflow as tf
from tensorflow.contrib.rnn import RNNCell

import scipy.sparse as sp

import matplotlib.pyplot as plt


########## visualization.py

def plot_result(test_result,test_label1,path):
    ##all test result visualization
    fig1 = plt.figure(figsize=(7,1.5))
#    ax1 = fig1.add_subplot(1,1,1)
    a_pred = test_result[:,0]
    a_true = test_label1[:,0]
    plt.plot(a_pred,'r-',label='prediction')
    plt.plot(a_true,'b-',label='true')
    plt.legend(loc='best',fontsize=10)
    plt.savefig(path+'/test_all.jpg')
    plt.show()
    ## oneday test result visualization
    fig1 = plt.figure(figsize=(7,1.5))
```

```
#     ax1 = fig1.add_subplot(1,1,1)
    a_pred = test_result[0:96,0]
    a_true = test_label1[0:96,0]
    plt.plot(a_pred,'r-',label="prediction")
    plt.plot(a_true,'b-',label="true")
    plt.legend(loc='best',fontsize=10)
    plt.savefig(path+'/test_oneday.jpg')
    plt.show()

def plot_error(train_rmse,train_loss,test_rmse,test_acc,test_mae,
    path):
    ###train_rmse & test_rmse
    fig1 = plt.figure(figsize=(5,3))
    plt.plot(train_rmse, 'r-', label="train_rmse")
    plt.plot(test_rmse, 'b-', label="test_rmse")
    plt.legend(loc='best',fontsize=10)
    plt.savefig(path+'/rmse.jpg')
    plt.show()
    #### train_loss & train_rmse
    fig1 = plt.figure(figsize=(5,3))
    plt.plot(train_loss,'b-', label='train_loss')
    plt.legend(loc='best',fontsize=10)
    plt.savefig(path+'/train_loss.jpg')
    plt.show()

    fig1 = plt.figure(figsize=(5,3))
    plt.plot(train_rmse,'b-', label='train_rmse')
    plt.legend(loc='best',fontsize=10)
    plt.savefig(path+'/train_rmse.jpg')
    plt.show()

    ### accuracy
    fig1 = plt.figure(figsize=(5,3))
    plt.plot(test_acc, 'b-', label="test_acc")
    plt.legend(loc='best',fontsize=10)
    plt.savefig(path+'/test_acc.jpg')
    plt.show()
    ### rmse
    fig1 = plt.figure(figsize=(5,3))
    plt.plot(test_rmse, 'b-', label="test_rmse")
    plt.legend(loc='best',fontsize=10)
    plt.savefig(path+'/test_rmse.jpg')
    plt.show()
    ### mae
    fig1 = plt.figure(figsize=(5,3))
    plt.plot(test_mae, 'b-', label="test_mae")
    plt.legend(loc='best',fontsize=10)
    plt.savefig(path+'/test_mae.jpg')
    plt.show()



########## END visualization.py

########## utils.py
```

```
                                                                    ۹۴
def normalized_adj(adj):                                            ۹۵
    adj = sp.coo_matrix(adj)                                        ۹۶
    rowsum = np.array(adj.sum(1))                                   ۹۷
    d_inv_sqrt = np.power(rowsum, -0.5).flatten()                   ۹۸
    d_inv_sqrt[np.isinf(d_inv_sqrt)] = 0.                           ۹۹
    d_mat_inv_sqrt = sp.diags(d_inv_sqrt)                           ۱۰۰
    normalized_adj = adj.dot(d_mat_inv_sqrt).transpose().dot(       ۱۰۱
        d_mat_inv_sqrt).tocoo()
    normalized_adj = normalized_adj.astype(np.float32)              ۱۰۲
    return normalized_adj                                           ۱۰۳
                                                                    ۱۰۴
def sparse_to_tuple(mx):                                            ۱۰۵
    mx = mx.tocoo()                                                 ۱۰۶
    coords = np.vstack((mx.row, mx.col)).transpose()                ۱۰۷
    L = tf.SparseTensor(coords, mx.data, mx.shape)                  ۱۰۸
    return tf.sparse_reorder(L)                                     ۱۰۹
                                                                    ۱۱۰
def calculate_laplacian(adj, lambda_max=1):                         ۱۱۱
    adj = normalized_adj(adj + sp.eye(adj.shape[0]))                ۱۱۲
    adj = sp.csr_matrix(adj)                                        ۱۱۳
    adj = adj.astype(np.float32)                                    ۱۱۴
    return sparse_to_tuple(adj)                                     ۱۱۵
                                                                    ۱۱۶
def weight_variable_glorot(input_dim, output_dim, name=""):         ۱۱۷
    init_range = np.sqrt(6.0 / (input_dim + output_dim))            ۱۱۸
    initial = tf.random_uniform([input_dim, output_dim], minval=-   ۱۱۹
        init_range,
                                maxval=init_range, dtype=tf.float32)۱۲۰
                                                                    ۱۲۱
    return tf.Variable(initial,name=name)                           ۱۲۲
                                                                    ۱۲۳
########## END utils.py                                            ۱۲۴
                                                                    ۱۲۵
                                                                    ۱۲۶
########## tgcn.py                                                 ۱۲۷
                                                                    ۱۲۸
                                                                    ۱۲۹
                                                                    ۱۳۰
class tgcnCell(RNNCell):                                            ۱۳۱
    """Temporal Graph Convolutional Network """                     ۱۳۲
                                                                    ۱۳۳
    def call(self, inputs, **kwargs):                               ۱۳۴
        pass                                                        ۱۳۵
                                                                    ۱۳۶
    def __init__(self, num_units, adj, num_nodes, input_size=None,  ۱۳۷
                 act=tf.nn.tanh, reuse=None):                       ۱۳۸
                                                                    ۱۳۹
        super(tgcnCell, self).__init__(_reuse=reuse)                ۱۴۰
        self._act = act                                             ۱۴۱
        self._nodes = num_nodes                                     ۱۴۲
        self._units = num_units                                     ۱۴۳
        self._adj = []                                              ۱۴۴
        self._adj.append(calculate_laplacian(adj))                  ۱۴۵
                                                                    ۱۴۶
                                                                    ۱۴۷
```

```
    @property                                                          ۱۴۸
    def state_size(self):                                              ۱۴۹
        return self._nodes * self._units                               ۱۵۰
                                                                       ۱۵۱
    @property                                                          ۱۵۲
    def output_size(self):                                             ۱۵۳
        return self._units                                             ۱۵۴
                                                                       ۱۵۵
    def __call__(self, inputs, state, scope=None):                     ۱۵۶
                                                                       ۱۵۷
        with tf.variable_scope(scope or "tgcn"):                       ۱۵۸
            with tf.variable_scope("gates"):                           ۱۵۹
                value = tf.nn.sigmoid(                                 ۱۶۰
                    self._gc(inputs, state, 2 * self._units, bias      ۱۶۱
                       =1.0, scope=scope))
                r, u = tf.split(value=value, num_or_size_splits=2,     ۱۶۲
                   axis=1)
            with tf.variable_scope("candidate"):                       ۱۶۳
                r_state = r * state                                    ۱۶۴
                c = self._act(self._gc(inputs, r_state, self._units,   ۱۶۵
                    scope=scope))
            new_h = u * state + (1 - u) * c                            ۱۶۶
        return new_h, new_h                                            ۱۶۷
                                                                       ۱۶۸
                                                                       ۱۶۹
    def _gc(self, inputs, state, output_size, bias=0.0, scope=None):   ۱۷۰
        ## inputs:(-1,num_nodes)                                       ۱۷۱
        inputs = tf.expand_dims(inputs, 2)                             ۱۷۲
        ## state:(batch,num_node,gru_units)                            ۱۷۳
        state = tf.reshape(state, (-1, self._nodes, self._units))      ۱۷۴
        ## concat                                                      ۱۷۵
        x_s = tf.concat([inputs, state], axis=2)                       ۱۷۶
        input_size = x_s.get_shape()[2].value                          ۱۷۷
        ## (num_node,input_size,-1)                                    ۱۷۸
        x0 = tf.transpose(x_s, perm=[1, 2, 0])                         ۱۷۹
        x0 = tf.reshape(x0, shape=[self._nodes, -1])                   ۱۸۰
                                                                       ۱۸۱
        scope = tf.get_variable_scope()                                ۱۸۲
        with tf.variable_scope(scope):                                 ۱۸۳
            for m in self._adj:                                        ۱۸۴
                x1 = tf.sparse_tensor_dense_matmul(m, x0)              ۱۸۵
#                print(x1)                                             ۱۸۶
            x = tf.reshape(x1, shape=[self._nodes, input_size,-1])     ۱۸۷
            x = tf.transpose(x,perm=[2,0,1])                           ۱۸۸
            x = tf.reshape(x, shape=[-1, input_size])                  ۱۸۹
            weights = tf.get_variable(                                 ۱۹۰
                'weights', [input_size, output_size], initializer=tf   ۱۹۱
                   .contrib.layers.xavier_initializer())
            x = tf.matmul(x, weights)  # (batch_size * self._nodes,    ۱۹۲
                output_size)
            biases = tf.get_variable(                                  ۱۹۳
                "biases", [output_size], initializer=tf.               ۱۹۴
                   constant_initializer(bias, dtype=tf.float32))
            x = tf.nn.bias_add(x, biases)                              ۱۹۵
            x = tf.reshape(x, shape=[-1, self._nodes, output_size])    ۱۹۶
            x = tf.reshape(x, shape=[-1, self._nodes * output_size])   ۱۹۷
```

```
        return x

########## END tgcn.py

########## input_data.py


def load_sz_data(dataset):
    sz_adj = pd.read_csv(r'data/sz_adj.csv',header=None)
    adj = np.mat(sz_adj)
    sz_tf = pd.read_csv(r'data/sz_speed.csv')
    return sz_tf, adj

def load_los_data(dataset):
    los_adj = pd.read_csv(r'data/los_adj.csv',header=None)
    adj = np.mat(los_adj)
    los_tf = pd.read_csv(r'data/los_speed.csv')
    return los_tf, adj


def preprocess_data(data, time_len, rate, seq_len, pre_len):
    train_size = int(time_len * rate)
    train_data = data[0:train_size]
    test_data = data[train_size:time_len]

    trainX, trainY, testX, testY = [], [], [], []
    for i in range(len(train_data) - seq_len - pre_len):
        a = train_data[i: i + seq_len + pre_len]
        trainX.append(a[0 : seq_len])
        trainY.append(a[seq_len : seq_len + pre_len])
    for i in range(len(test_data) - seq_len -pre_len):
        b = test_data[i: i + seq_len + pre_len]
        testX.append(b[0 : seq_len])
        testY.append(b[seq_len : seq_len + pre_len])

    trainX1 = np.array(trainX)
    trainY1 = np.array(trainY)
    testX1 = np.array(testX)
    testY1 = np.array(testY)
    return trainX1, trainY1, testX1, testY1

########## END input_data.py

time_start = time.time()
###### Settings ######
flags = tf.app.flags
FLAGS = flags.FLAGS
flags.DEFINE_float('learning_rate', 0.001, 'Initial learning rate.')
flags.DEFINE_integer('training_epoch', 1, 'Number of epochs to train
    .')
flags.DEFINE_integer('gru_units', 64, 'hidden units of gru.')
flags.DEFINE_integer('seq_len',12 , '  time length of inputs.')
flags.DEFINE_integer('pre_len', 3, 'time length of prediction.')
flags.DEFINE_float('train_rate', 0.8, 'rate of training set.')
flags.DEFINE_integer('batch_size', 32, 'batch size.')
flags.DEFINE_string('dataset', 'los', 'sz or los.')
```

```
flags.DEFINE_string('model_name', 'tgcn', 'tgcn')
model_name = FLAGS.model_name
data_name = FLAGS.dataset
train_rate =  FLAGS.train_rate
seq_len = FLAGS.seq_len
output_dim = pre_len = FLAGS.pre_len
batch_size = FLAGS.batch_size
lr = FLAGS.learning_rate
training_epoch = FLAGS.training_epoch
gru_units = FLAGS.gru_units

###### load data ######
if data_name == 'sz':
    data, adj = load_sz_data('sz')
if data_name == 'los':
    data, adj = load_los_data('los')

time_len = data.shape[0]
num_nodes = data.shape[1]
data1 =np.mat(data,dtype=np.float32)

#### normalization
max_value = np.max(data1)
data1  = data1/max_value
trainX, trainY, testX, testY = preprocess_data(data1, time_len,
   train_rate, seq_len, pre_len)

totalbatch = int(trainX.shape[0]/batch_size)
training_data_count = len(trainX)

def TGCN(_X, _weights, _biases):
    ###
    cell_1 = tgcnCell(gru_units, adj, num_nodes=num_nodes)
    cell = tf.nn.rnn_cell.MultiRNNCell([cell_1], state_is_tuple=True
       )
    _X = tf.unstack(_X, axis=1)
    outputs, states = tf.nn.static_rnn(cell, _X, dtype=tf.float32)
    m = []
    for i in outputs:
        o = tf.reshape(i,shape=[-1,num_nodes,gru_units])
        o = tf.reshape(o,shape=[-1,gru_units])
        m.append(o)
    last_output = m[-1]
    output = tf.matmul(last_output, _weights['out']) + _biases['out
       ']
    output = tf.reshape(output,shape=[-1,num_nodes,pre_len])
    output = tf.transpose(output, perm=[0,2,1])
    output = tf.reshape(output, shape=[-1,num_nodes])
    return output, m, states

###### placeholders ######
inputs = tf.placeholder(tf.float32, shape=[None, seq_len, num_nodes
   ])
labels = tf.placeholder(tf.float32, shape=[None, pre_len, num_nodes
   ])
```

```
# Graph weights
weights = {
    'out': tf.Variable(tf.random_normal([gru_units, pre_len], mean
       =1.0), name='weight_o')}
biases = {
    'out': tf.Variable(tf.random_normal([pre_len]),name='bias_o')}

if model_name == 'tgcn':
    pred,ttts,ttto = TGCN(inputs, weights, biases)

y_pred = pred


###### optimizer ######
lambda_loss = 0.0015
Lreg = lambda_loss * sum(tf.nn.l2_loss(tf_var) for tf_var in tf.
   trainable_variables())
label = tf.reshape(labels, [-1,num_nodes])
##loss
loss = tf.reduce_mean(tf.nn.l2_loss(y_pred-label) + Lreg)
##rmse
error = tf.sqrt(tf.reduce_mean(tf.square(y_pred-label)))
optimizer = tf.train.AdamOptimizer(lr).minimize(loss)

###### Initialize session ######
variables = tf.global_variables()
saver = tf.train.Saver(tf.global_variables())
#sess = tf.Session()
gpu_options = tf.GPUOptions(per_process_gpu_memory_fraction=0.333)
sess = tf.Session(config=tf.ConfigProto(gpu_options=gpu_options))
sess.run(tf.global_variables_initializer())

out = 'out/%s'%(model_name)
#out = 'out/%s_%s'%(model_name,'perturbation')
path1 = '%s_%s_lr%r_batch%r_unit%r_seq%r_pre%r_epoch%r'%(model_name,
   data_name,lr,batch_size,gru_units,seq_len,pre_len,training_epoch
   )
path = os.path.join(out,path1)
if not os.path.exists(path):
    os.makedirs(path)

###### evaluation ######
def evaluation(a,b):
    rmse = math.sqrt(mean_squared_error(a,b))
    mae = mean_absolute_error(a, b)
    F_norm = la.norm(a-b,'fro')/la.norm(a,'fro')
    r2 = 1-((a-b)**2).sum()/((a-a.mean())**2).sum()
    var = 1-(np.var(a-b))/np.var(a)
    return rmse, mae, 1-F_norm, r2, var


x_axe,batch_loss,batch_rmse,batch_pred = [], [], [], []
test_loss,test_rmse,test_mae,test_acc,test_r2,test_var,test_pred =
   [],[],[],[],[],[],[]

for epoch in range(training_epoch):
```

```
    for m in range(totalbatch):
        mini_batch = trainX[m * batch_size : (m+1) * batch_size]
        mini_label = trainY[m * batch_size : (m+1) * batch_size]
        _, loss1, rmse1, train_output = sess.run([optimizer, loss,
            error, y_pred],
                                                 feed_dict = {inputs
                                                    :mini_batch,
                                                    labels:
                                                    mini_label})
        batch_loss.append(loss1)
        batch_rmse.append(rmse1 * max_value)

     # Test completely at every epoch
    loss2, rmse2, test_output = sess.run([loss, error, y_pred],
                                         feed_dict = {inputs:testX,
                                            labels:testY})
    test_label = np.reshape(testY,[-1,num_nodes])
    rmse, mae, acc, r2_score, var_score = evaluation(test_label,
       test_output)
    test_label1 = test_label * max_value
    test_output1 = test_output * max_value
    test_loss.append(loss2)
    test_rmse.append(rmse * max_value)
    test_mae.append(mae * max_value)
    test_acc.append(acc)
    test_r2.append(r2_score)
    test_var.append(var_score)
    test_pred.append(test_output1)

    print('Iter:{}'.format(epoch),
          'train_rmse:{:.4}'.format(batch_rmse[-1]),
          'test_loss:{:.4}'.format(loss2),
          'test_rmse:{:.4}'.format(rmse),
          'test_acc:{:.4}'.format(acc))

    if (epoch % 500 == 0):
        saver.save(sess, path+'/model_100/TGCN_pre_%r'%epoch,
           global_step = epoch)

time_end = time.time()
print(time_end-time_start,'s')

############## visualization ###############
b = int(len(batch_rmse)/totalbatch)
batch_rmse1 = [i for i in batch_rmse]
train_rmse = [(sum(batch_rmse1[i*totalbatch:(i+1)*totalbatch])/
   totalbatch) for i in range(b)]
batch_loss1 = [i for i in batch_loss]
train_loss = [(sum(batch_loss1[i*totalbatch:(i+1)*totalbatch])/
   totalbatch) for i in range(b)]

index = test_rmse.index(np.min(test_rmse))
test_result = test_pred[index]
var = pd.DataFrame(test_result)
var.to_csv(path+'/test_result.csv',index = False,header = False)
plot_result(test_result,test_label1,path)
```

```
plot_error(train_rmse,train_loss,test_rmse,test_acc,test_mae,path)
print('min_rmse:%r'%(np.min(test_rmse)),
      'min_mae:%r'%(test_mae[index]),
      'max_acc:%r'%(test_acc[index]),
      'r2:%r'%(test_r2[index]),
      'var:%r'%test_var[index])
```

آ-۴ برنامه پیش‌بینی ترافیک معابر شهری به وسیله مدل ARIMA

این برنامه حاصل پیاده‌سازی پیش‌بینی ترافیک معابر شهری به کمک مدل رگرسیو-میانگین متحرک جمع‌بسته در این پایان‌نامه می‌باشد. برنامه ذکر شده در محیط زبان برنامه نویسی پایتون با استفاده از ابزار statsmodels نوشته شده است.

برنامه آ-۴: برنامه پیش‌بینی ترافیک معابر شهری به وسیله مدل ARIMA

```

data_path = r'data/los_speed.csv'

# -*- coding: utf-8 -*-
import pandas as pd
import numpy as np
from sklearn.metrics import mean_squared_error,mean_absolute_error
import numpy.linalg as la
import math
from statsmodels.tsa.arima_model import ARIMA

def preprocess_data(data, time_len, rate, seq_len, pre_len):
data1 = np.mat(data)
train_size = int(time_len * rate)
train_data = data1[0:train_size]
test_data = data1[train_size:time_len]

trainX, trainY, testX, testY = [], [], [], []
for i in range(len(train_data) - seq_len - pre_len):
a = train_data[i: i + seq_len + pre_len]
trainX.append(a[0 : seq_len])
trainY.append(a[seq_len : seq_len + pre_len])
for i in range(len(test_data) - seq_len -pre_len):
b = test_data[i: i + seq_len + pre_len]
testX.append(b[0 : seq_len])
testY.append(b[seq_len : seq_len + pre_len])
return trainX, trainY, testX, testY

###### evaluation ######
def evaluation(a,b):
rmse = math.sqrt(mean_squared_error(a,b))
mae = mean_absolute_error(a, b)
F_norm = la.norm(a-b)/la.norm(a)
r2 = 1-((a-b)**2).sum()/((a-a.mean())**2).sum()
```

```
var = 1-(np.var(a - b))/np.var(a)
return rmse, mae, 1-F_norm, r2, var

data = pd.read_csv(data_path)
data.shape

time_len = data.shape[0]
num_nodes = data.shape[1]
train_rate = 0.8
seq_len = 12
pre_len = 3

trainX,trainY,testX,testY = preprocess_data(data, time_len,
    train_rate, seq_len, pre_len)
print(
'length Train : ' , len(trainX),
'length Test: ' , len(testX),

)

rng = pd.date_range('1/3/2012', periods=time_len, freq='5min')
a1 = pd.DatetimeIndex(rng)
a1.shape

data.index = a1

p = 1
d = 0
q = 0

predicSpeed,rmse,mae,acc,r2,var,pred,ori = [],[],[],[],[],[],[],[]
for i in range(num_nodes):
ts = data.iloc[:,i]
ts_log=np.log(ts)
ts_log=np.array(ts_log,dtype=np.float)
where_are_inf = np.isinf(ts_log)
ts_log[where_are_inf] = 0
ts_log = pd.Series(ts_log)
ts_log.index = a1
model = ARIMA(ts_log,order=[p,d,q])
properModel = model.fit()
predict_ts = properModel.predict(4, dynamic=True)
log_recover = np.exp(predict_ts)
ts = ts[log_recover.index]

predicSpeed.append(log_recover);

er_rmse,er_mae,er_acc,r2_score,var_score = evaluation(ts,log_recover
    )
rmse.append(er_rmse)
mae.append(er_mae)
acc.append(er_acc)
r2.append(r2_score)
var.append(var_score)

acc1 = np.mat(acc)
```

```
acc1[acc1 < 0] = 0

print('arima_rmse:%r'%(np.mean(rmse)),
'arima_mae:%r'%(np.mean(mae)),
'arima_acc:%r'%(np.mean(acc1)),
'arima_r2:%r'%(np.mean(r2)),
'arima_var:%r'%(np.mean(var)))
```

واژه‌نامه فارسی به انگلیسی

پیش‌بینی ترافیک شهری به کمک یادگیری عمیق . . . . . Urban Traffic Predication using Deep Learning

بعد . . . . . Dimesion

توزیعی . . . . . Distributed

ماتریس . . . . . Matrix

گراف . . . . . Graph

حافظه . . . . . Memory

کانولوشن . . . . . Convolution

شبکه عصبی بازگشتی . . . . . Recurrent neural network

شبکه عصبی بازگشتی با واحدهای دروازه‌ای . . . . . Gated recurrent unit

شبکه‌عصبی بازگشتی حافظه کوتاه مدت مانگار . . . . . Long short-term memory

سری . . . . . Sequential

پایه . . . . . Base

شبکه عصبی پیچشی . . . . . Shared

شبکه عصبی پیچشی گراف . . . . . Graph Convolutional Neural Network

واژه‌نامه انگلیسی به فارسی

پیش‌بینی ترافیک شهری به کمک یادگیری عمیق . . . . . Urban Traffic Predication using Deep Learning

بعد . . . . . Dimesion

توزیعی . . . . . Distributed

ماتریس . . . . . Matrix

گراف . . . . . Graph

حافظه . . . . . Memory

کانولوشن . . . . . Convolution

شبکه عصبی بازگشتی . . . . . Recurrent Neural Network

شبکه عصبی بازگشتی با واحدهای دروازه‌ای . . . . . Gated recurrent unit

شبکه‌عصبی بازگشتی حافظه کوتاه مدت ماندگار . . . . . Long Short-Term Memory

سری . . . . . Sequential

پایه . . . . . Base

شبکه عصبی پیچشی . . . . . Convolution Neural Network

شبکه عصبی پیچشی گراف . . . . . Graph Convolution Neural Network

Hakim Sabzevari University

An Outline of MSc. Thesis

Surname:Jalambadani	Name:Morteza	Student No.:9713185026

Supervisors: Dr. Mahmood Amintoosi and Dr. Alireza Ghodsi

Advisor: Dr. Mahdi Mahdizadeh

Faculty of Mathematics and Computer Science

Program: Computer Science Field:Decision Science and Knowledge

Title of thesis: Urban Traffic Prediction Using Deep Learning

Keywords:Neural Network, Deep Learning, Graph Convolutional Networks, Recurrent Neural Networks, Traffic Prediction, Urban Traffic

Abstract:

Accurate and real-time traffic forecasting plays an important role in the Intelligent Traffic System and is of great significance for urban traffic planning, traffic management, and traffic control. However, traffic forecasting has always been considered an open scientific issue, owing to the constraints of urban road network topological structure and the law of dynamic change with time, namely, spatial dependence and temporal dependence. To capture the spatial and temporal dependence simultaneously, that the results of this issue help to better manage traffic and control the design of urban thoroughfares. Due to the structure of the urban road network and the topological structure of this network, temporal dependence and spatial dependence are factors involved in predicting urban traffic. In this thesis, urban traffic is predicted using recurrent neural network based on graph convolution network. This method is combined two methods of recurrent neural network and graph convolution network, graph convolution network is used to solve the spatial dependence of urban traffic and recurrent neural network is used to solve the temporal dependence of traffic. The study examined two taxi datasets in Shenzhen, China, with about 156 urban road network, and the Los Angeles Highway Sensor dataset, which includes more than 200 urban road network. The results of testing the method presented on these two datasets are compared with the results obtained from GRU and ARIMA methods. The lost function on the test results of the proposed forecasting method is about 50% of the lost function in the ARIMA forecasting method.

Hakim Sabzevari University
Faculty of Mathematics and Computer Science

A Thesis Submitted in Partial Fulfilment of the Requirement for the Degree of Master of Science in Computer Science

Urban Traffic Prediction Using Deep Learning

Supervisors:
Dr. Mahmood Amintoosi and Dr. Alireza Ghodsi

Advisor:
Dr. Mahdi Mahdizadeh

By:
Morteza Jalambadani

December 2021