

فهرست

| | |
|----|--|
| ۶ | بخش اول : مقدمه |
| ۷ | بخش دوم : بررسی اهمیت تشخیص اخبار جعلی و ویژگی های آن |
| ۷ | ۱-۲ اهمیت تشخیص اخبار جعلی |
| ۷ | ۲-۲ تعریف اخبار جعلی |
| ۸ | ۳-۲ ویژگی های اخبار جعلی |
| ۸ | ۴-۲ منتشرکنندگان اخبار جعلی و کاربران هدف |
| ۹ | ۵-۲ انواع محتواهای خبری و اثر پژواک |
| ۱۰ | بخش سوم : روش های تشخیص اخبار جعلی |
| ۱۰ | ۱-۳ رویکرد های شناسایی اخبار جعلی |
| ۱۱ | ۲-۳ مدل های شناسایی اخبار جعلی |
| ۱۲ | ۳-۳ مقایسه روش های مبتنی بر یادگیری ماشین و یادگیری عمیق |
| ۱۳ | بخش چهارم : بررسی انواع مدل های یادگیری عمیق و پارامتر های آن ها |
| ۱۳ | ۱-۴ معرفی شبکه های عصبی |
| ۱۵ | ۲-۴ انواع شبکه های عصبی پرسپترون |
| ۱۹ | ۳-۴ معرفی چند تابع فعالسازی پرکاربرد |
| ۲۱ | ۴-۴ معرفی تابع خطا Cross-entropy |
| ۲۲ | ۵-۴ الگوریتم یادگیری و پس انتشار خطا |
| ۲۸ | ۶-۴ روش های بهینه سازی وزن ها |
| ۳۰ | ۷-۴ انواع داده ورودی به مدل |
| ۳۰ | ۸-۴ تعمیم پذیری در شبکه های عصبی عمیق |
| ۳۱ | ۹-۴ تجزیه بایاس و واریانس |
| ۳۲ | ۱۰-۴ چالش های آموزش شبکه های عصبی عمیق |
| ۳۲ | ۱۱-۴ بیش برازش مدل |
| ۳۳ | ۱۲-۴ روش های مقابله با بیش برازش |
| ۳۶ | ۱۳-۴ انفجار و محوشوندگی گرادیان |
| ۳۷ | ۱۴-۴ روش های مقابله با انفجار و محوشوندگی گرادیان |

| | |
|-----|--|
| ۴۳ | ۵-۱۰-۴ داده های محدود |
| ۴۴ | ۶-۱۰-۴ چالش زمان در آموزش شبکه های عصبی عمیق |
| ۴۶ | ۱۱-۴ معرفی شبکه های عصبی بازگشتی |
| ۵۱ | ۱۱-۴ انواع الگو های پردازش توالی |
| ۵۲ | ۴-۱۱-۴ معماری کدگذار و گدگشا در شبکه های عصبی بازگشتی |
| ۵۳ | ۴-۱۱-۴ نحوه آموزش شبکه های بازگشتی |
| ۶۲ | ۴-۱۱-۴ شبکه های عصبی بازگشتی دو طرفه |
| ۶۴ | ۴-۱۱-۴ پایداری شبکه های عصبی بازگشتی |
| ۷۰ | ۴-۱۲-۴ معرفی شبکه های عصبی با حافظه طولانی کوتاه مدت (LSTM) |
| ۷۵ | ۴-۱۳-۴ معرفی شبکه های عصبی با واحد های بازگشتی دروازه ای (GRU) |
| ۷۷ | ۴-۱۴-۴ شبکه های عصبی دو طرفه |
| ۷۸ | ۴-۱۵-۴ شبکه های عصبی کانولوشنی |
| ۸۵ | ۴-۱۶-۴ مدل های بر پایه توجه |
| ۸۹ | ۴-۱۶-۴ ۱-۱۶-۴ مدل های خود-توجه |
| ۹۲ | ۴-۱۶-۴ ۲-۱۶-۴ مدل های توجه چندسر |
| ۹۵ | ۴-۱۶-۴ ۳-۱۶-۴ شبکه های مبدل |
| ۹۸ | ۴-۱۶-۴ ۴-۱۶-۴ شبکه های کدگذار دوطرفه برمبنای مبدل ها (BERT) |
| ۱۰۱ | بخش پنجم : پیش پردازش داده ها و بازنمایی کلمات |
| ۱۰۱ | ۱-۵ پیش پردازش داده ها |
| ۱۰۲ | ۲-۵ بازنمایی کلمات |
| ۱۰۹ | ۳-۵ انتخاب ابرپارامتر های مناسب برای بازنمایی کلمات |
| ۱۱۰ | ۴-۵ شبکه های آموزش دیده دو طرفه مبتنی بر مبدل ها |
| ۱۱۳ | بخش ششم : پیاده سازی و نتایج |
| ۱۱۳ | ۱-۶ مجموعه داده |
| ۱۱۳ | ۲-۶ پیش پردازش داده ها |
| ۱۱۹ | ۳-۶ پیاده سازی مدل های یادگیری عمیق |
| ۱۱۹ | ۴-۳-۶ مدل های یادگیری عمیق مبتنی بر شبکه های با حافظه طولانی کوتاه مدت |

| | |
|----------|---|
| ۱۲۹..... | ۶-۳-۲ مدل های یادگیری عمیق مبتنی بر شبکه های با واحد های بازگشتی دروازه ای (<i>GRU</i>) |
| ۱۳۴..... | ۶-۳-۳ مدل های یادگیری عمیق چند کاناله |
| ۱۳۹..... | ۶-۳-۴ مدل های یادگیری عمیق مبتنی بر شبکه های کانولوشنی |
| ۱۴۱..... | ۶-۳-۴ مدل های یادگیری عمیق مبتنی بر شبکه های دوطرفه برمبنای مبدل ها |
| ۱۴۴..... | نتیجه گیری |
| ۱۴۵..... | منابع |

بخش اول : مقدمه

در بخش دوم ابتدا در مورد اهمیت تشخیص اخبار جعلی و کاربرد های آن توضیح داده شده است.

در بخش سوم در مورد روش های تشخیص اخبار جعلی بحث شده است و روش های حل مبتنی بر یادگیری ماشین و یادگیری عمیق مقایسه شده اند.

در بخش چهارم به انواع مدل های یادگیری عمیق و مقایسه انواع شبکه های عمیق و همچنین پارامتر ها و ابرپارامتر های مدل و نقش آن ها در افزایش دقت مدل پرداخته شده است .

در بخش پنجم به انواع روش های بدست آوردن بردار بازنمایی کلمات و پیش پردازش داده ها بحث شده است .

در بخش ششم خروجی مدل های عمیق بر روی مجموعه داده بررسی شده است و عملکرد شبکه ها بر مبنای معیار های ارزیابی عملکرد شبکه مقایسه شده است .

بخش دوم : بررسی اهمیت تشخیص اخبار جعلی و ویژگی های آن

۱-۲ اهمیت تشخیص اخبار جعلی

امروزه انتشار اخبار در شبکه های اجتماعی مزایای فراوانی مانند سرعت انتشار بالا ، دسترسی راحت و هزینه ناچیز دارد و در عین حال قادر است به راحتی اخبار و اطلاعات جعلی با هدف های متفاوت منتشر کند و انتشار این اخبار و رسیدن آن ها به دست خوانندگان می تواند اثرات منفی برای فرد و جامعه داشته باشد و باعث کاهش اعتماد به رسانه های خبری و شبکه های اجتماعی شود. به همین جهت تشخیص اخبار جعلی می تواند نقش مهمی در جامعه ایفا کند و یک زمینه تحقیقاتی منحصر بفرد را در این زمینه فراهم آورد.

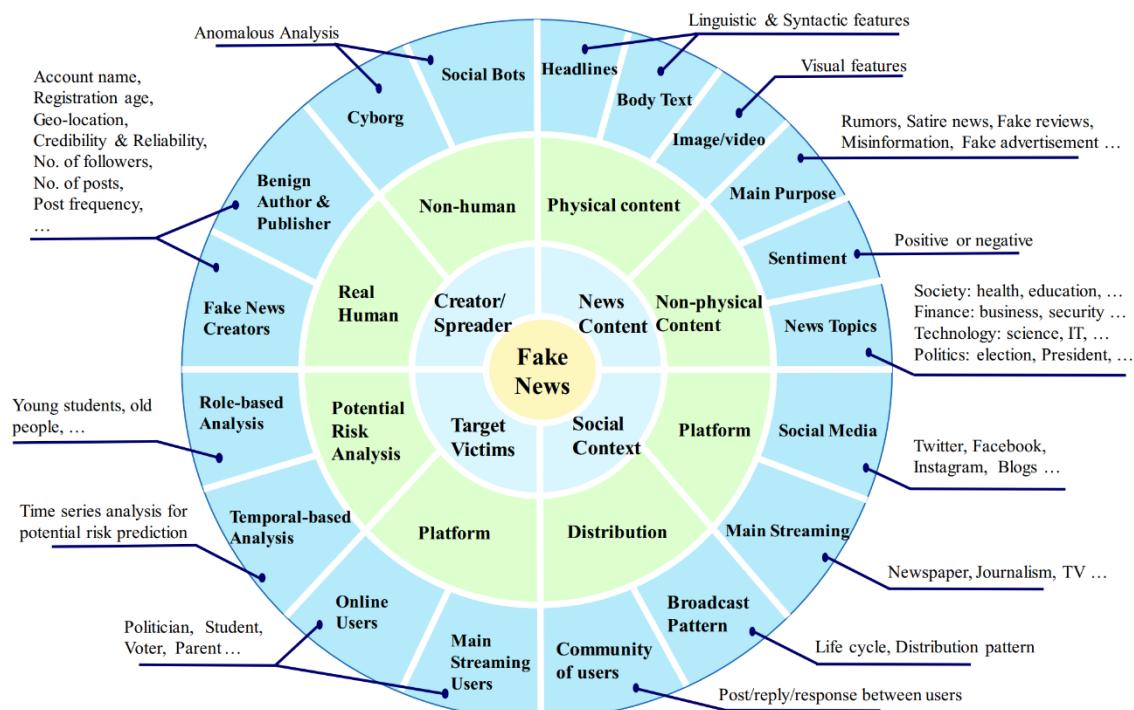
افزایش استفاده از شبکه های مجازی زمینه را برای انتشار اخبار و دیدگاه های جعلی در سطوح مختلف تجاری ، سیاسی و اجتماعی را فراهم آورده است و به راحتی موجب سردرگمی خوانندگان و بهره برداری از این سردرگمی در سطوح فردی و اجتماعی برای منافع تجاری و سیاسی خواهد شد .

به عنوان نمونه می توان به انتخابات ریاست جمهوری سال ۲۰۱۶ ایالات متحده اشاره کرد که چگونه اخبار جعلی یکی از دلایل افزایش چند قطبی سیاسی در کمپین های انتخاباتی شد و رای دهنده های دهنده های بسیار تحت تاثیر اخبار ساختگی و بیانیه های سیاسی جعلی قرار می گرفتند [۱].

با توجه به افزایش روز افرون اطلاعات در شکل های متفاوت اخبار ، نظرات و تبلیغات تشخیص اخبار جعلی را به صورت در لحظه کاری سخت و پیچیده بدل کرده است و همچنین جعل اخبار برای فریب خوانندگان نیز باعث می شود ویژگی های زبانی متون به تنها یی برای تشخیص این گونه اخبار کافی نباشد و نیاز به کشف الگو هایی عمیق تر و همچنین استفاده از ویژگی های دیگر به عنوان اطلاعات کمکی مانند اختصاص اعتبار به نویسنده اخبار و الگو و نحوه انتشار اخبار کمک شایانی به پیش بینی و تشخیص اخبار جعلی خواهد کرد [۱].

۲-۲ تعریف اخبار جعلی

امروزه اخبار جعلی در موضوعات و شکل ها و پلتفرم های متفاوتی منتشر می شود و تعریف آن به شکل کلی آسان نیست ، اما در [۲] تعریفی برای اخبار جعلی ارائه شده است " اخبار جعلی مقاله ای است که با مقاصد خاص و به صورت قابل تایید اشتباه است و می تواند باعث گمراحتی خواننده شود . "



شکل ۱-۲ : ویژگی های اخبار جعلی [۱]

۳-۲ ویژگی های اخبار جعلی

تنوع در اخبار جعلی : متون متنوع و متفاوتی را می توان نزدیک به تعریف اخبار جعلی دانست مانند : شایعات ، نظرات جعلی ، تبلیغات جعلی ، بیانیه های جعلی که هر کدام میتواند به نوعی افراد و جامعه را تحت تاثیر قرار دهد [۱] .

سرعت انتشار اخبار : اکثر اخبار جعلی منتشر شده در شبکه های اجتماعی بر رویداد ها و وقایع اخیر تمرکز می کنند و به همین علت معمولاً عمر کوتاهی دارند و شناسایی منبع و الگو انتشار این گونه اخبار را سخت خواهد کرد [۱] .

تعداد بالا اخبار : وب سایت ها و صفحات متنوعی در اینترنت وجود دارند که با هدف های متفاوت اقدام به انتشار اخبار جعلی در اشکال متفاوت می کنند و هر فردی به تنها ی می تواند اقدام به نوشتن و انتشار این دسته از اخبار کند [۱] .

۴-۲ منتشرکنندگان اخبار جعلی و کاربران هدف

ربات ها : متدائل ترین منبع غیر انسانی برای انتشار اخبار جعلی ، هرزنامه ها و اطلاعات نادرست ربات ها هستند که در واقع الگوریتم های کامپیوتری هستند که برای انتشار این گونه اخبار و تولید آن ها سعی میکنند مانند انسان ها رفتار کنند [۳], [۱] .

انسان ها : تشخیص اخبار جعلی و اطلاعات نادرست که توسط انسان ها و به عمد نوشته و منتشر می شود بسیار سخت تر خواهد بود زیرا تنها از طریق محتوا و تحلیل زبانی متون می توان آن ها را تشخیص داد [۱] .

۵-۲ انواع محتواهای خبری و اثر پژواک

در حالت کلی اخبار را می توان در دو دسته کلی در نظر گرفت ، اخباری که دارای پیکربندی هستند و شامل عنوان و بدنی اصلی خبر هستند مانند اخبار منتشر شده در خبرگزاری ها و دسته دوم اخباری که پیکربندی ندارند و به صورت ایده و احساسات بیان می شوند مانند نظراتی که در فروشگاه های اینترنتی ثبت می شوند [۱] ، [۴] .

اخبار جعلی در شبکه های اجتماعی :

شبکه های اجتماعی به علت هزینه پایین و دسترسی آسان شیوه استفاده از اخبار و اطلاعات را تغییر دادند و کاربران می توانند تجربه ها و تعاملات خود را با دوستان و دنبال کننده های خود به اشتراک بگذارند و اثرات این گونه پیام ها با انتشار در گروهی از افراد با خط فکری مشابه طبق اثر اکو چمبر تقویت خواهد شد [۵] .

اثر محفظه پژواک بر انتشار اخبار جعلی :

این اثر را می توان با ذکر یک مثال توضیح داد : به عنوان نمونه کاربران در شبکه های اجتماعی مانند توییتر معمولاً افرادی با خط فکری مشابه خود را دنبال می کنند و بنابراین اخباری را دریافت می کنند که مورد پسند خودشان قرار می گیرد و بنابراین افراد در شبکه های اجتماعی به قطب هایی با خط فکری وایده های مشابه تقسیم می شوند که این نتیجه اثر محفظه پژواک است [۵] .

اثر محفظه پژواک فرایند باور و استفاده اخبار جعلی را بوسیله دو مولفه تسهیل می کند :

۱ - اعتبار اجتماعی : به این معنا که افراد احتمال می دهند که یک منبع خبری معتبر است اگر سایر افراد تشخیص دهند که منبع معتبر است حتی اگر اطلاعات کافی برای اثبات اعتبار منبع خبری وجود نداشته باشد [۵] .

۲ - فرکانس اکتشاف : به این معنا که خوانندگان ممکن است به طور طبیعی اطلاعاتی را ترجیح دهند که به صورت مکرر شنیده اند حتی اگر آن ها اخبار جعلی باشند و مطالعات نشان می دهند افزایش میزان قرار گرفتن در معرض یک ایده به تنها یک نظر مثبت در فرد کافی است [۵] .

در نتیجه ، اثر محفظه پژواک جوامع همگن و تقسیم بندی شده ای را با اطلاعاتی محدود ایجاد می کند و تحقیقات نشان می دهند که این جوامع همگن به محرك اصلی انتشار اخبار و اطلاعات تبدیل می شوند و همواره قطبش میان جوامع را تقویت می کنند [۵] .

بخش سوم : روش های تشخیص اخبار جعلی

۱-۳ رویکردهای شناسایی اخبار جعلی

چند رویکرد کلی برای تشخیص اخبار جعلی وجود دارد که می تواند بر اساس کاربر ، محتوای خبر و یا شبکه ای شامل افراد باشد .

شناسایی بر اساس کاربر :

در این رویکرد هدف کاوش ویژگی ها و رفتار های کسانی است که به اخبار توجه می کنند و آن ها را می خوانند و با استخراج ویژگی ها و مدل سازی رفتار کاربران از طریق پروفایل آن ها سعی می شود اخبار جعلی برچسب گذاری شود .

ویژگی های پروفایل هر کاربر به دو دسته ویژگی های صریح و ضمنی تقسیم می شود که از ویژگی های صریح می توان به اطلاعات ثبت نامی کاربر و فعالیت های کاربر و همچنین دنبال کننده ها و دنبال شوندگان آن کاربر اشاره کرد و از ویژگی های ضمنی پروفایل می توان به سن ، شخصیت و تعصبات سیاسی اشاره کرد [۶] .

شناسایی بر اساس نظرات کاربران :

کاربرانی که اخبار را می خوانند نظرات و احساسات خود را در نظرات خود نشان می دهند که می تواند علاوه بر محتوای خبر به عنوان اطلاعات کمکی برای تشخیص اخبار جعلی راهگشا باشد [۶] .

شناسایی بر اساس شبکه ای از کاربران :

کاربران بر حسب علایق خود موضوعات مختلفی را دنبال می کنند و در شبکه هایی با موضوعات مورد علاقه خود عضو می شوند و نظرات خود را با اعضای شبکه به اشتراک می گذارند و به نظرات یکدیگر واکنش نشان می دهند که این تعاملات و استخراج الگو از روی این تعاملات می تواند منجر به شناسایی اخبار جعلی شود [۵] .

شناسایی اخبار جعلی بر اساس محتوا :

هر خبر می تواند اطلاعاتی مانند منبع خبر ، عنوان خبر ، متن و یا بدنه اصلی خبر و تصاویر مربوط به آن خبر را دارا باشد که هر کدام به سهم خود می توانند در تشخیص اخبار جعلی راهگشا باشند .

شناسایی اخبار جعلی بر اساس محتوا می تواند مبتنی بر پردازش زبان و یا تصویر به کار رفته در خبر باشد .

در شناسایی اخبار جعلی مبتنی بر پردازش زبان معمولاً کلمات و یا جملاتی ماهرانه برای فریب ذهن خواننده استفاده می شود که از اطلاعاتی کمکی در سطح کلمه و جمله می توان استفاده کرد ، در سطح کلمه مانند تعداد کل کلمات خبر و تعداد حروف موجود در کلمات خبر ، تعداد تکرار کلمات با تعداد حروف بیشتر و تعداد کلمات منحصر بفرد و همچنین ویژگی هایی در سطح جمله مانند اصطلاحات ، علائم نگارشی و نقل قول های به کار رفته در متن خبر [۵] .



شکل ۱-۳ : خبری جعلی و پیکربندی آن [۱]

۲-۳ مدل های شناسایی اخبار جعلی

انواع مدل های شناسایی اخبار جعلی و استخراج ویژگی ها می تواند مبنی بر دانش و یا سبک نوشتار باشد.

در مدل های شناسایی اخبار جعلی مبنی بر دانش هدف چک کردن حقیقت از طریق منابع دیگر است و رویکرد هایی برای چک کردن درستی اخبار مانند تشخیص خبره ، تشخیص بر اساس خردجمعی و تشخیص مبنی بر محاسبات موجود است [۵] .

در رویکرد تشخیص خبره چک کردن واقعیت بر عهده انسان است و با مطالعه استناد و مدارک موجود رای نهایی صادر می شود که فرآیندی است زمانبندی و در مقیاس های کوچک انجام می شود [۵] .

در رویکرد تشخیص مبنی بر خردجمعی هر یک از خوانندگان خبر ، رای خود را در رابطه با جعلی بودن هر یک از اخبار می دهند و در نهایت با رای جمعی از خوانندگان یک خبر جعلی تشخیص داده می شود[۵] .

در رویکرد های مبنی بر محاسبات هدف ایجاد یک سیستم برای استخراج ویژگی ها دسته بندی اخبار بدون دخالت انسان است که علاوه بر دسته بندی اخبار در زمان کوتاه قابلیت اعمال بر مقیاس های بزرگ از خبر ها را دارد.[۵]

۳-۳ مقایسه روش های مبتنی بر یادگیری ماشین و یادگیری عمیق

مدل های یادگیری ماشین به دلیل وابستگی به فضای ویژگی ها دقت های بسیار پایینی را برای داده های از جنس اخبار دارند زیرا ویژگی های پیچیده فراوانی در این نوع متون وجود دارد و مدل های یادگیری ماشین وابستگی فراوانی به مهندسی ویژگی های ورودی به مدل دارند اما در مدل های یادگیری عمیق وظیفه استخراج ویژگی ها به صورت کاملا خودکار بر عهده مدل قرار می گیرد و در حالت کلی مدل های یادگیری عمیق در داده های از جنس اخبار دقت های بسیار بالایی را نتیجه می دهند .

معایب رویکرد های سنتی یادگیری ماشین :

- ۱ - دشواری در مدل سازی داده های غیرخطی
- ۲ - وابستگی شدید به ویژگی های انتخاب شده
- ۳ - دشواری در انتخاب ویژگی

مزایای رویکرد های سنتی یادگیری ماشین :

- ۱ - تئوری اثبات آسانتر به نسبت رویکرد های یادگیری عمیق
- ۲ - دقت قابل قبول در حجم داده های کم

معایب رویکرد های یادگیری عمیق :

- ۱ - عدم دقت قابل قبول بر روی حجم داده های کم
- ۲ - تئوری اثبات سخت تر به نسبت رویکرد های یادگیری ماشین
- ۳ - وابستگی به سخت افزار های پردازشی قوی

مزایای رویکرد های یادگیری عمیق :

- ۱ - توانایی مدل سازی داده های غیرخطی
- ۲ - انتخاب خودکار ویژگی ها
- ۳ - عملکرد بهتر شبکه با افزایش حجم داده ها

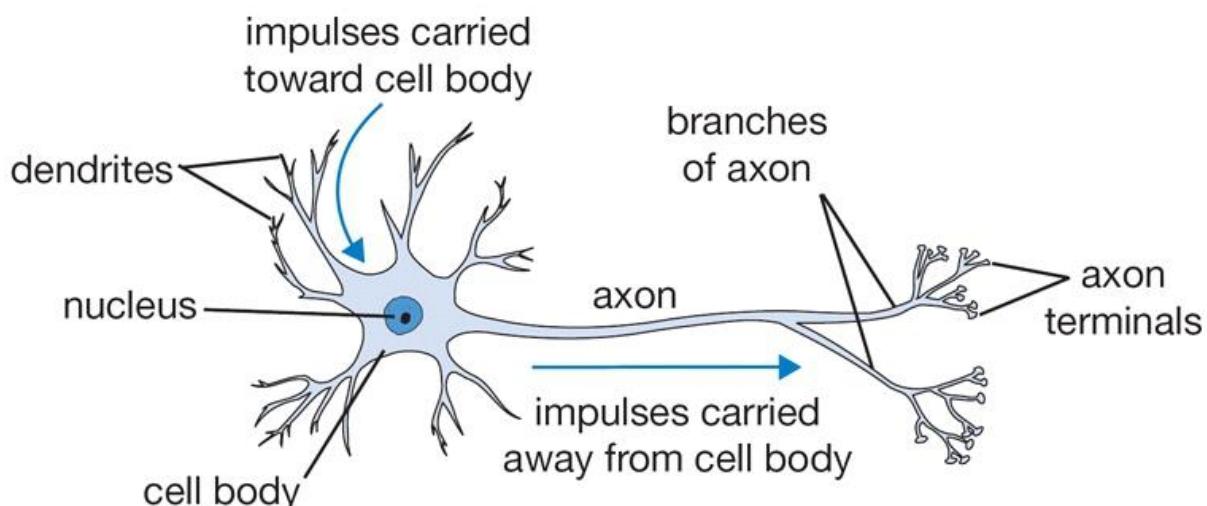
بخش چهارم : بررسی انواع مدل های یادگیری عمیق و پارامتر های آن ها

۱-۴ معرفی شبکه های عصبی

شبکه های عصبی و شبکه های عصبی عمیق امروزه در بسیاری از کاربردها مانند تشخیص صوت ، تشخیص دست خط و طبقه بندی متن و تصاویر مورد استفاده قرار می گیرند و با دقت های بالا به ورودی های مسئله پاسخ می دهند .

یک شبکه عصبی اساساً توالی عملیاتی است که به یک ماتریس از داده های ورودی اعمال می شود .

واحد محاسباتی پایه مغز یک نورون است و حدوداً ۸۶ میلیارد نورون می توان در سیستم عصبی انسان یافت که این نورون ها بوسیله حدوداً 10^{14} تا 10^{15} سیناپس به یکدیگر متصل شده اند [۷] .

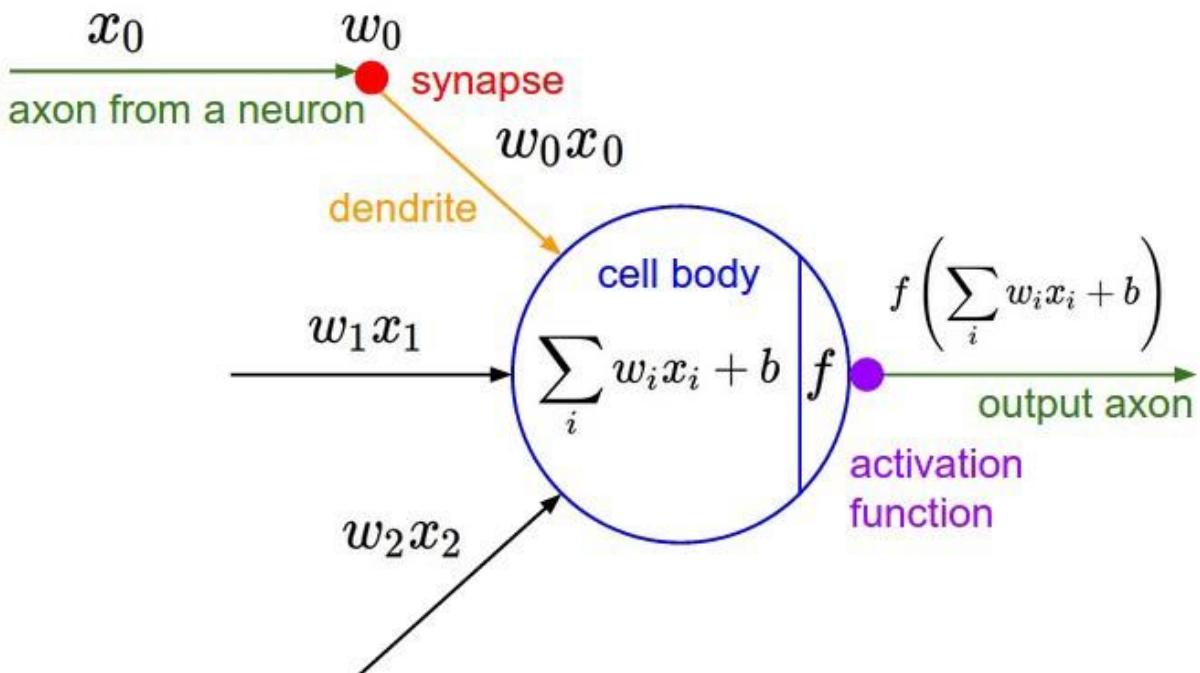


شکل ۱-۴ : نمونه یک نورون عصبی [۷]

واحد پایه محاسباتی یک شبکه عصبی نورون ها هستند که معمولاً به آنها گره یا واحد نیز می گویند .

هر نورون ورودی خود را از سایر نورون ها و یا یک منبع خارجی دریافت می کند و خروجی مربوط به خود را محاسبه می کند . هر ورودی تحت تاثیر وزن های ارتباط دهنده بین نورون ها قرار می گیرد و ورودی سایر نورون ها را می سازد و هر نورون تابع مربوط به خود را در جمع وزن دار ورودی های خود اعمال می کند و خروجی را تعیین می کند [۸] و [۹] .

ایده اصلی اختصاص دادن وزن های شبکه به سیناپس هاست که قابل کنترل و یادگیری هستند و دندانهای دندریت ها وظیفه حمل سیگنال و رساندن آن به سلول ها و در نهایت نورون ها را دارند و اگر مجموع سیگنال های دریافت شده از یک مقدار مشخص بیشتر باشد نورون مربوطه فعال می شود و سیگنالی را در طول اکسون ارسال می کند [۱۰] و [۸] .



شکل ۲-۴ : واحد شبکه های عصبی مصنوعی [۱۹]

آموزش یک نورون یا شبکه ای از نورون های متصل به هم به معنای یادگیری الگوی حاکم بر داده ها است که معادل آن در مدل محاسباتی وزن های موجود در شبکه است که بوسیله ی این وزن ها جریان اطلاعاتی و الگو موجود در داده ها استخراج و یادگیری می شود [۱۱].

معرفی برخی اصطلاحات موجود در شبکه های عصبی :

لایه : به مجموعه ای از نورون های شبکه یک لایه می گویند .

نورون های ورودی : در نورون های ورودی یا لایه ورودی شبکه های عصبی محاسباتی انجام نمی شود و تنها داده های ورودی را به لایه ی بعدی منتقل می کنند .

لایه خروجی : اخرین لایه از نورون های شبکه که خروجی آن ها با توجه به تابع فعالسازی نورون ها همان فرمت خروجی مطلوب شبکه خواهد بود .

اتصالات : هر شبکه شامل مجموعه ای از اتصالات در مدل محاسباتی دارای وزن مختص خود هستند و این اتصالات خروجی نورون های هر لایه از شبکه را به عنوان ورودی لایه بعدی از نورون ها فراهم می کنند [۱۲] .

لایه مخفی : در این لایه محاسبات شبکه انجام می شود به این صورت که هر نورون بعد از دریافت و محاسبه مجموع وزن دار سیگنال های ورودی به خود و اعمال آن به تابع فعالسازی مربوط به خود خروجی را می سازد و آن ها را به لایه بعد خود که می تواند لایه مخفی دیگر یا لایه خروجی باشد ارسال می کند .

تابع فعالسازی : تابع فعالسازی هر نورون با اعمال به ورودی های نورون خروجی آن نورون را تعریف می کند.

که در انواع مختلف توابع خطی و غیر خطی تعریف می شود .

قانون یادگیری شبکه : الگوریتمی است که پارامتر های شبکه شامل وزن ها و سایر پارامتر ها که در ادامه معرفی می شوند را اطلاع می کند .

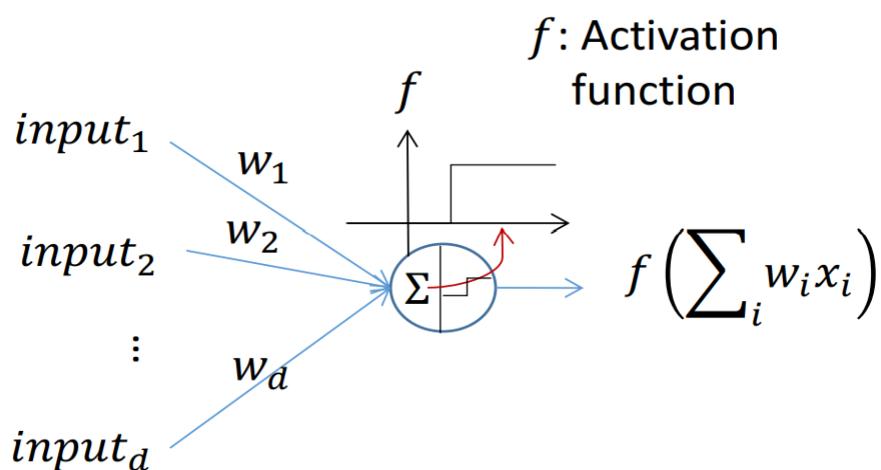
شبکه های عصبی پیشخور (feedforward) : در این شبکه ها اطلاعات تنها در جهت رو به جلو از نورون های ورودی و سپس نورون های لایه های مخفی در صورت وجود و در نهایت به نورون های خروجی می رسد و هیچ حلقه و یا بازخوردی از سمت خروجی شبکه به لایه های عقب تر وجود ندارد [۱۲] .

۲-۴ انواع شبکه های عصبی پرسپترون :

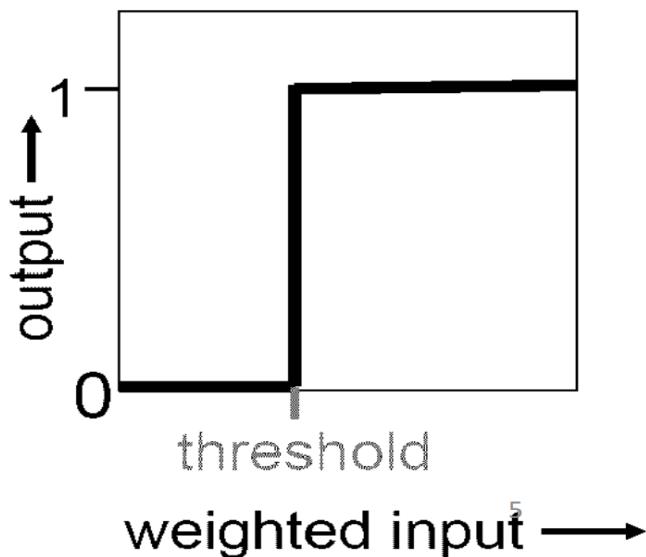
پرسپترون تک لایه :

در [۹] الگوریتم محاسبه خروجی یک نورون پرسپترون تک لایه ذکر شده است :

ابتدا جمع وزن دار ورودی ها به صورت پیشخور محاسبه می شود سپس اگر مجموع وزن دار از یک آستانه مشخص شده بیشتر باشد نورون فعالیت را آغاز می کند و خروجی را ارسال می کند و هر نورون مقداری از ارزش یک گزاره را از طریق اتصالات دریافت می کند و با ترکیب این ارزش ها ، درستی گزاره ای دیگر را بدست می آورد .

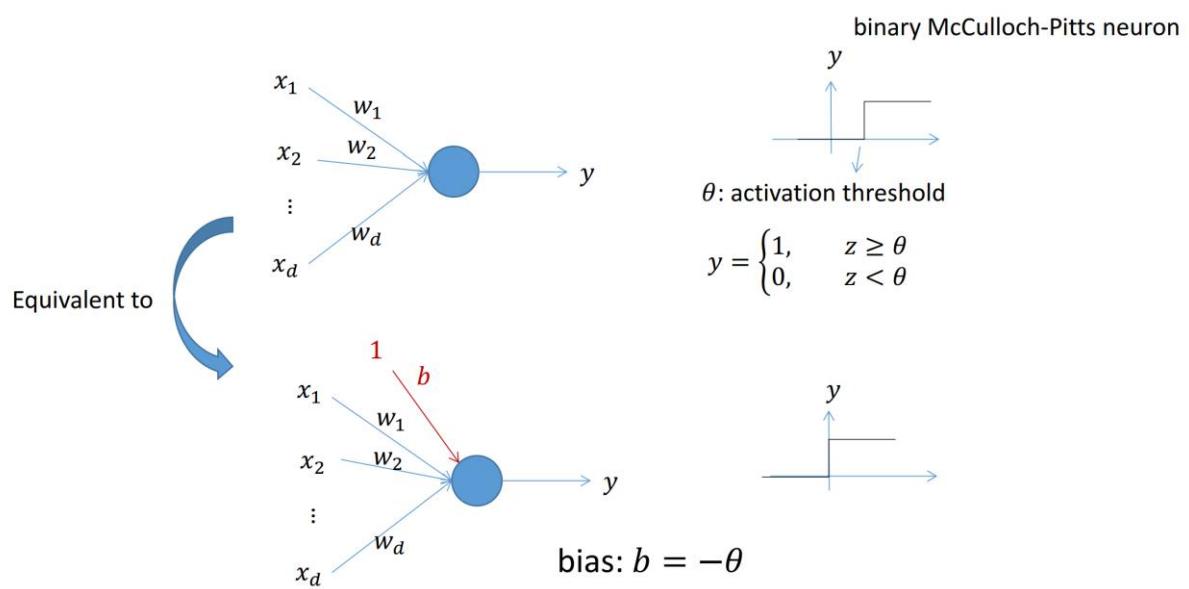


شکل ۳-۴ : نمونه محاسبات یک نورون عصبی [۱۹]



شکل ۴-۴ : تابع فعالسازی پله در یک نورون [۱۹]

تابع فعالسازی استفاده شده در این مدل دارای آستانه باینری می باشد به این صورت که اگر مقدار مجموع وزن دار ورودی به نورون از آستانه بیشتر باشد خروجی نورون یک و اگر کمتر از آستانه باشد خروجی نورون صفر خواهد بود و برای تغییر مقدار آستانه ورودی بایاس به نورون افزوده شد به این صورت که همواره مقدار ورودی این اتصال یک است و با وزن این اتصال که همان مقدار بایاس خواهد بود آستانه تابع فعالسازی باینری کنترل می شود .



شکل ۴-۵ : تابع فعالسازی پله واحد و کنترل آستانه آتش تابع فعالسازی [۱۹]

الگوریتم یادگیری پرسپترون :

همانطور که در [۱۱] آمده است الگوریتم یادگیری پرسپترون یک الگوریتم یادگیری با نظارت و تکرار شونده است که به صورت پیشخور سعی در دسته بندی داده های موجود دارد به این صورت که ابتدا وزن ها و بایاس به صورت تصادفی مقدار دهی می شوند و سپس در هر مرحله تکرار الگوریتم مقادیر وزن ها و بایاس را به گونه ای تغییر می دهد که نقاط بیشتری به درستی دسته بندی شوند .

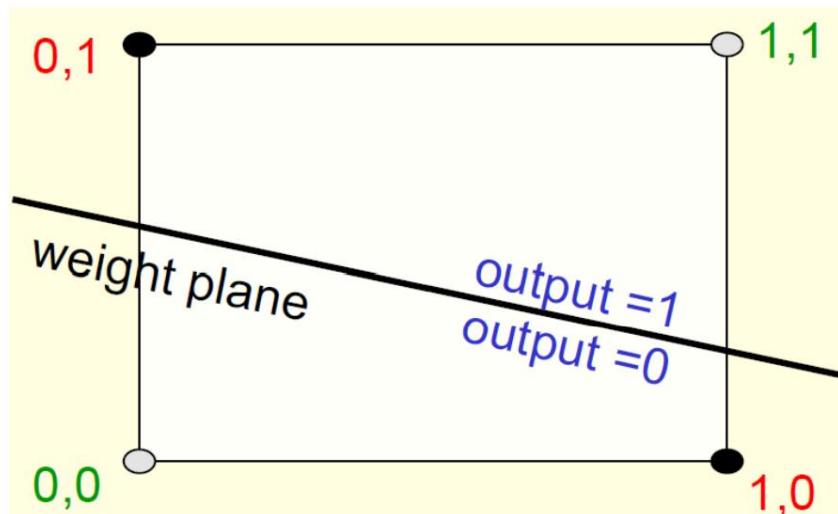
```

initialize:  $w^{(1)} = (0, \dots, 0)$ 
for  $t = 1, 2, \dots$ 
    if ( $\exists i \text{ s.t } y^{(i)} \times (w^{(t)} \cdot x^{(i)}) \leq 0$ )
         $w^{(t+1)} = w^{(t)} + y^{(i)} x^{(i)}$ 
    else
         $w^{(t+1)} = w^{(t)}$ 

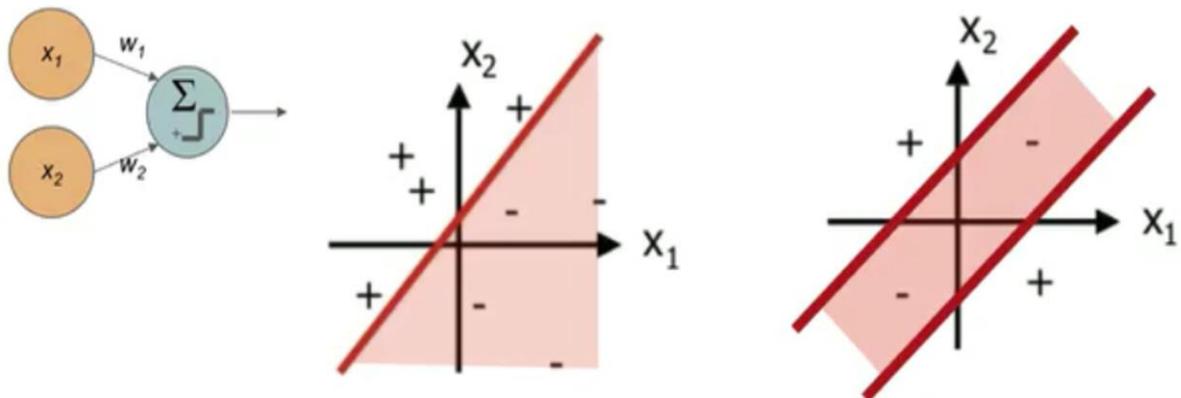
```

شکل ۶-۴ : الگوریتم یادگیری پرسپترون [۸]

در [۱۰] مطرح شده است مسئله ای که بعد ها موجب کنار گذاشتن این نوع شبکه ها شد مسئله خطی تفکیک پذیر بودن آن ها بود به گونه ای که برای داده هایی که به صورت خطی تفکیک پذیر نبودند قابل استفاده نبود و یکی از مثال هایی که مطرح شد مثال یادگیری عملگر XOR بود که این شبکه تک لایه در یادگیری آن ناتوان بود .



شکل ۷-۴ : ضعف پرسپترون تک لایه در یادگیری عملگر XOR [۱۹]

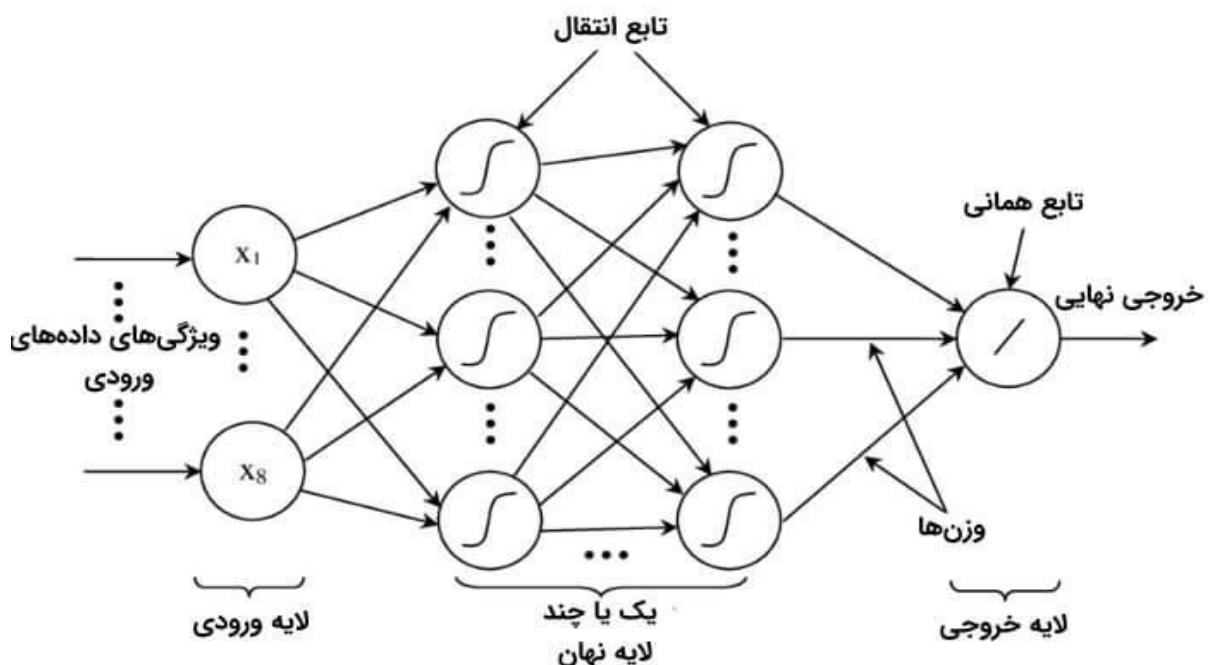


شکل ۸-۴ : ضعف پرسپترون در یادگیری داده های خطی تفکیک ناپذیر [۱۹]

شبکه های عصبی پرسپترون چند لایه :

در این نوع از شبکه های عصبی چند لایه نورون شامل لایه ورودی ، یک یا چند لایه مخفی و سپس لایه خروجی قرار دارد و از خروجی های هر لایه به عنوان ورودی لایه بعد استفاده می شود تا نهایتا اطلاعات به لایه خروجی و محاسبه خروجی شبکه منجر شود و همه می بین لایه ورودی و لایه خروجی قرار دارند لایه های مخفی نامیده می شوند .

این نوع از شبکه ها نیز شامل مجموعه ای از وزن ها و بایاس ها و با اینکه با استفاده از الگوریتم مناسب آموزش و یادگیری آن ها صورت گیرد .

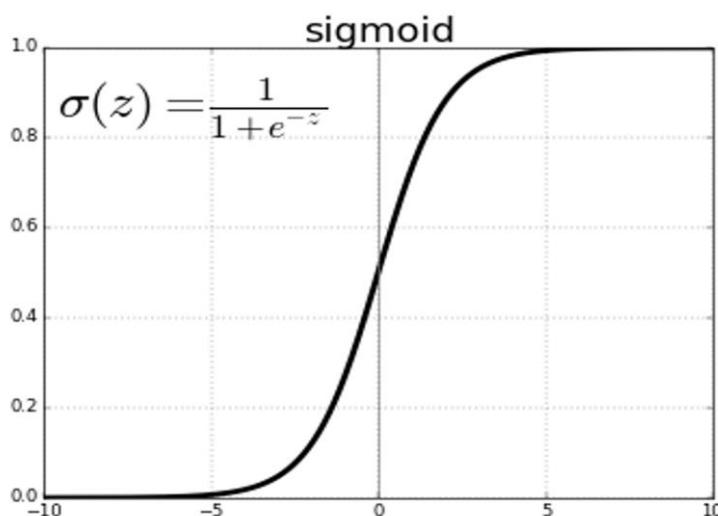


شکل ۹-۴ : شبکه عصبی پرسپترون چند لایه [۱۹]

از آن جایی که الگوریتم یادگیری پرسپترون از نوع الگوریتم های یادگیری ناظارت شده است به این معنا که هر ورودی از داده ها در مرحله یادگیری دارای برچسبی است که نشان می دهد متعلق به کدام دسته از داده ها قرار دارد بنابراین خروجی داده های آموزشی از قبل مشخص است و به این خروجی ها ، خروجی های مورد انتظار گفته می شود و از این خروجی ها برای سنجش عملکرد شبکه های عصبی استفاده می شود به این صورت که بر اساس اختلاف مقادیر خروجی پیش بینی شده توسط شبکه و خروجی مورد انتظار مقدار خطا شبکه عصبی محاسبه می شود [۱۱].

۳-۴ معرفی چند تابع فعالسازی پرکاربرد :

تابع فعالسازی sigmoid :

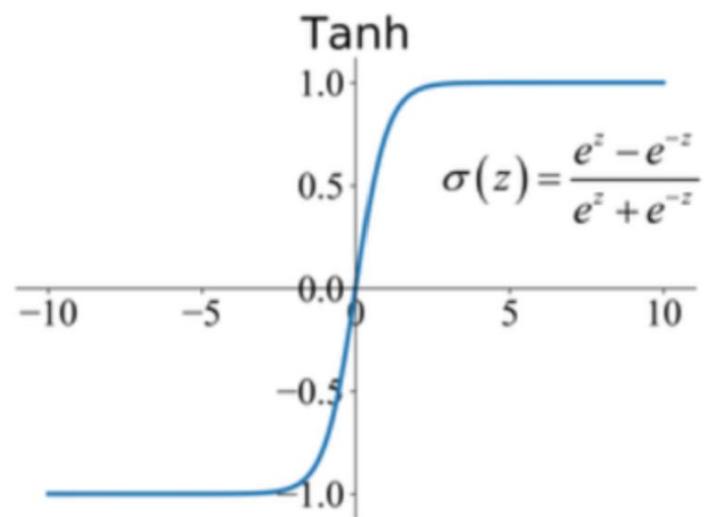


شکل ۱۰-۴ : تابع فعالسازی سیگموئید [۲۱]

تابع فعالسازی sigmoid که به عنوان تابع فعالسازی در لایه خروجی مسائل دسته بندی به ویژه مسائل دسته بندی باینری بیشترین کاربرد را دارد زیرا خروجی آن بین صفر و یک است [۱۴].

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad (4-3-1)$$

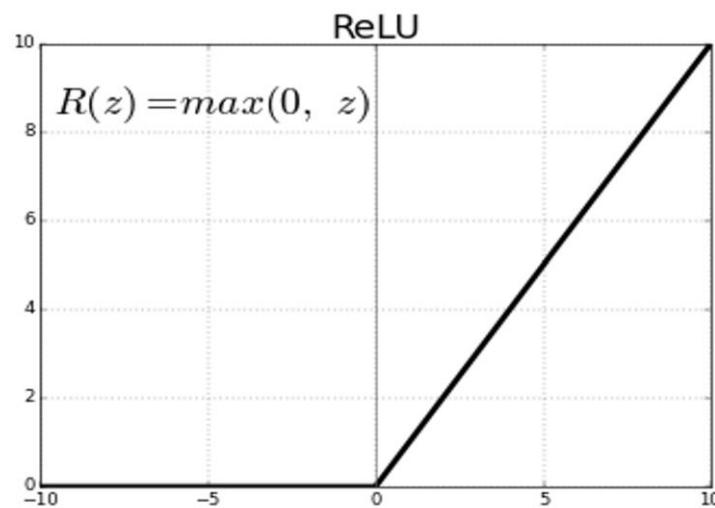
تابع فعالسازی تانژانت هایپربولیک :



شکل ۱۱-۴ : تابع فعالسازی تانژانت هایپربولیک [۳۱]

$$\sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \quad (4-3-2)$$

تابع فعالسازی ReLU :



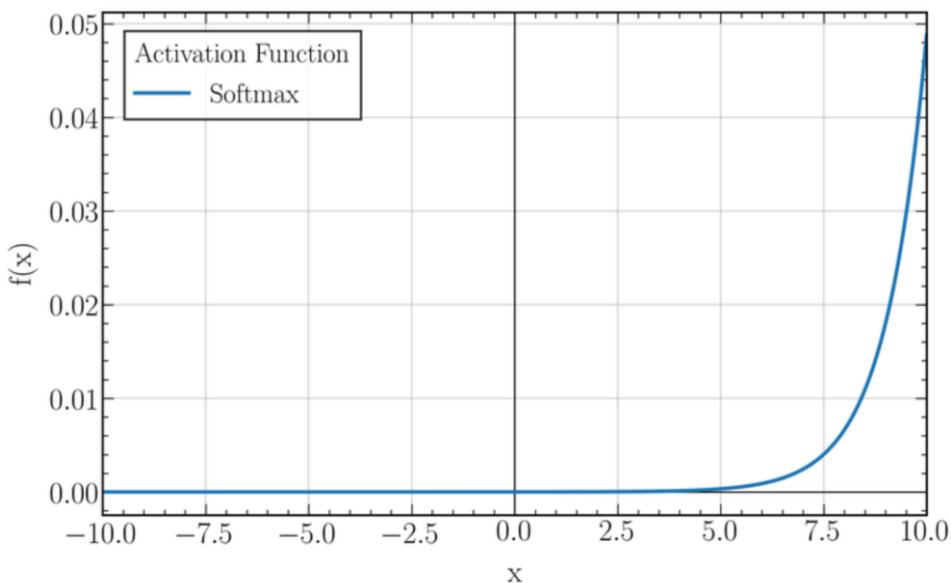
شکل ۱۲-۴ : تابع فعالسازی ReLU [۳۱]

^۱ Rectified Linear Unit

این تابع فعالسازی باعث می شود همه نورون ها به طور همزمان تحریک نشوند و برخی از نورون ها خاموش باشند به این معنا که خروجی آن ها صفر می شود .

از دیگر ویژگی های تابع فعالسازی Relu می توان به همگرا بی سریع تر و بهینه بودن از نظر محاسباتی آن اشاره کرد [۱۵] .

تابع فعالسازی softmax :



[۳۱] softmax : تابع فعالسازی

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \quad (4-3-3)$$

این تابع فعالسازی یک بردار از مقادیر را به یک بردار از احتمالات با مقادیری بین صفر و یک تبدیل می کند که همین ویژگی این تابع باعث می شود در مسائل دسته بندی های با بیش از ۲ دسته مورد استفاده قرار گیرد و از این منظر می توان به عنوان نسخه تعمیم یافته تابع فعالسازی sigmoid به آن نگاه کرد [۱۶] .

۴-۴ معرفی تابع خطا :

این تابع خطا که به لگاریتم خطا نیز معروف است عملکرد یک مدل دسته بندی را اندازه میگیرد و خروجی آن احتمالی بین صفر و یک است .

$$Loss = - \sum_{c=1}^M y_{oc} \log(p_{oc}) \quad (4-4-1)$$

M : تعداد دسته های موجود

y_{oc} : برچسب واقعی دسته c

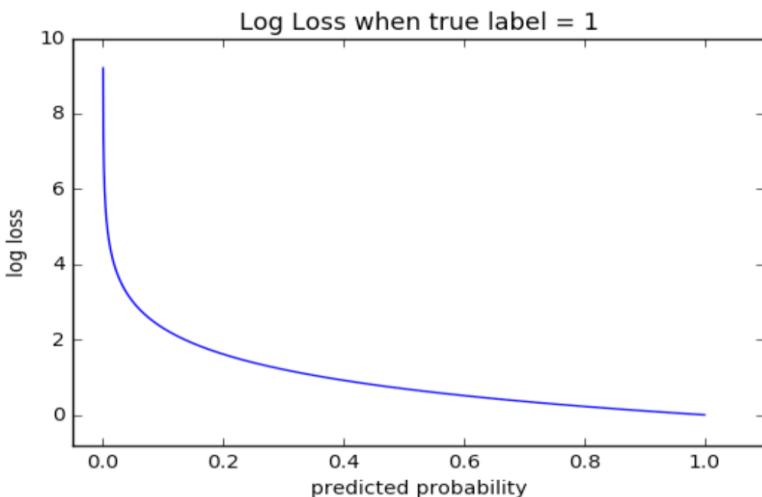
p_{oc} : مقدار احتمال خروجی دسته بند برای دسته c

اینتابع خطابازمانی که تعداد دسته های مدل برابر دو است به binary cross-entropy معروف است و فرمول بالا برای دو دسته به صورت زیر خواهد بود [۱۷].

for $M = 2$; (۴-۴-۲)

$$loss = -(y \log(p) + (1 - y) \log(1 - p))$$

مقدار اینتابع خطابازمانی در صورت واگرا بودن مقادیر خروجی مدل از خروجی واقعی، افزایش می یابد.



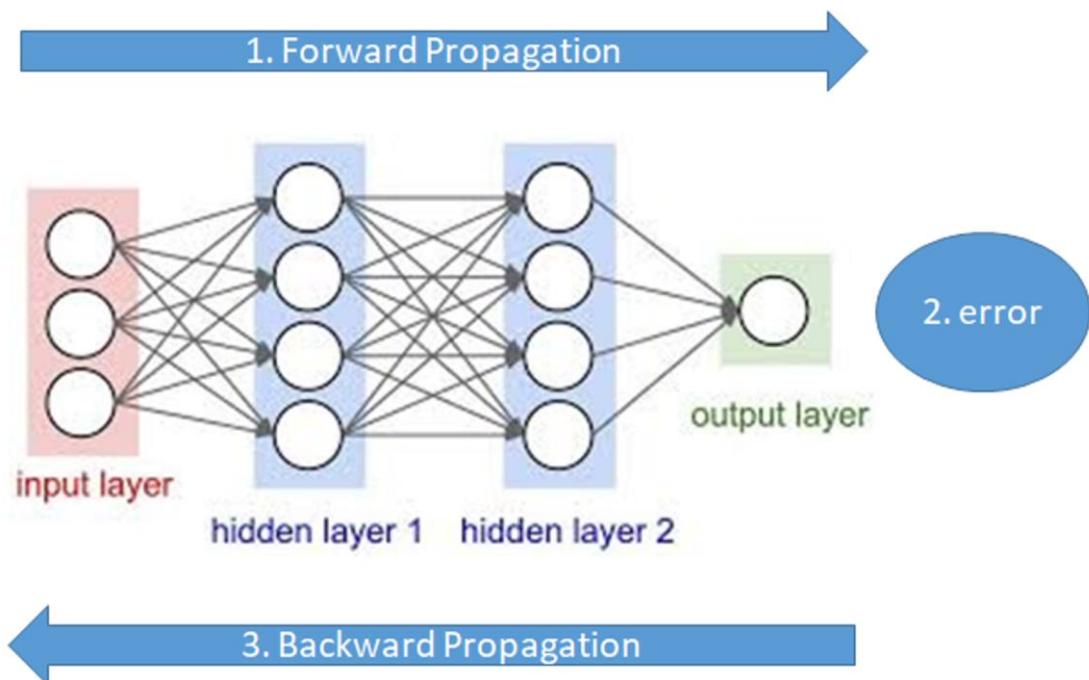
شکل ۱۴-۴ : لگاریتم تابع خطابازمانی که برچسب برابر یک باشد. [۱۷]

به عنوان مثال در صورتی که مقدار برچسب واقعی یک دسته برابر یک باشد و خروجی مدل مقدار یک دهم را برای آن دسته پیش بینی کند تابع خطابازمانی cross-entropy مقدار خطابالایی را به آن خروجی نسبت می دهد.

۴-۵ الگوریتم یادگیری و پس انتشار خطابا:

پس از محاسبه تابع خطابا که میتواند انواع مختلفی داشته باشد ، مقدار آن به صورت معکوس و رو به عقب از لایه خروجی به سمت لایه ورودی در شبکه منتشر می شود و با استفاده از مفهوم گرادیان مرتبه اول وزن هر کدام از اتصالات به اندازه ای که در خطابا خروجی شبکه نقش دارند تنبیه می شوند و هر وزن در جهت کاهش خطابا تغییر می کند و وزن های شبکه به روز رسانی می شوند به گونه ای که خطابا شبکه حداقل شود .

حال به معرفی الگوریتم پس انتشار خطابا می پردازیم که در واقع انتشار خطابا در جهت عکس جهت پیشخور پیش می رود که از آن برای بدست آوردن خروجی نهایی شبکه استفاده می شود .



شکل ۱۵-۴ : یک شبکه عصبی پرسپترون چندلایه و جریان های داده [۱۹]

خطای کل شبکه به ازای تمام داده های آموزشی به صورت زیر محاسبه می شود:

$$E = \sum_{n=1}^N loss(o^{(n)}.y^{(n)}) \quad (4-5-1)$$

سپس از مشتق خطای نسبت به وزن های شبکه محاسبه می شود :

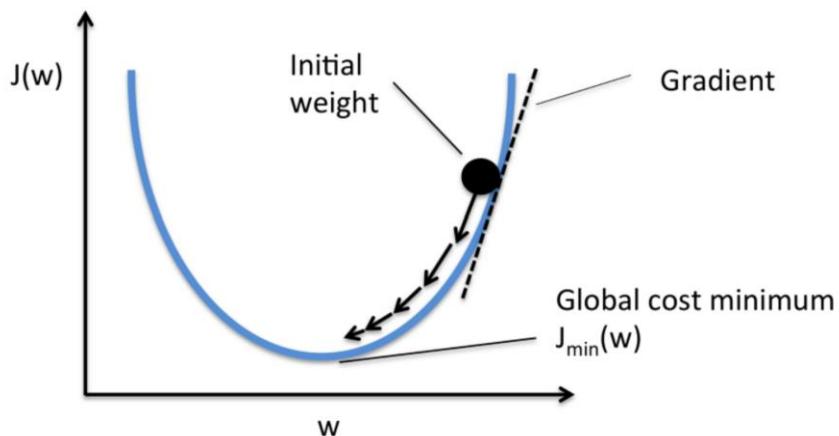
$$\frac{dE}{dw_{i,j}^{[k]}} = \sum_{n=1}^N \frac{loss(o^{(n)}.y^{(n)})}{dw_{i,j}^{[k]}} \quad (4-5-2)$$

وزن مربوط به نورون j از لایه k ام و نورون i از لایه $k-1$ ام $w_{i,j}^{[k]}$

و سپس با استفاده از الگوریتم گرادیان کاهشی وزن ها در خلاف جهت گرادیان به روز رسانی می شوند .

$$w_{i,j}^{[k]} = w_{i,j}^{[k]} - \eta \frac{dE}{dw_{i,j}^{[k]}} \quad (4-5-3)$$

η : نرخ یادگیری که به صورت یک ابرپارامتر در شبکه در نظر گرفته می شود و با توجه به سایر پارامتر های شبکه قابل بهینه سازی است .

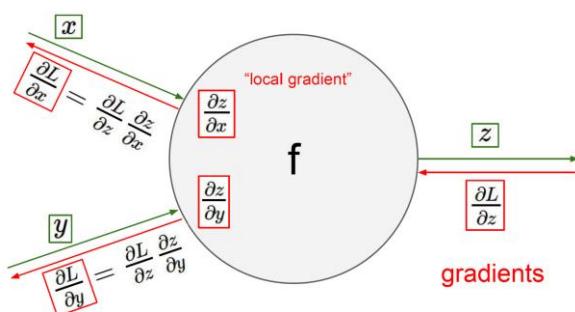


شکل ۱۶-۴ : تاثیر تنظیم مناسب نرخ یادگیری در یافتن نقاط بهینه [۱۹]

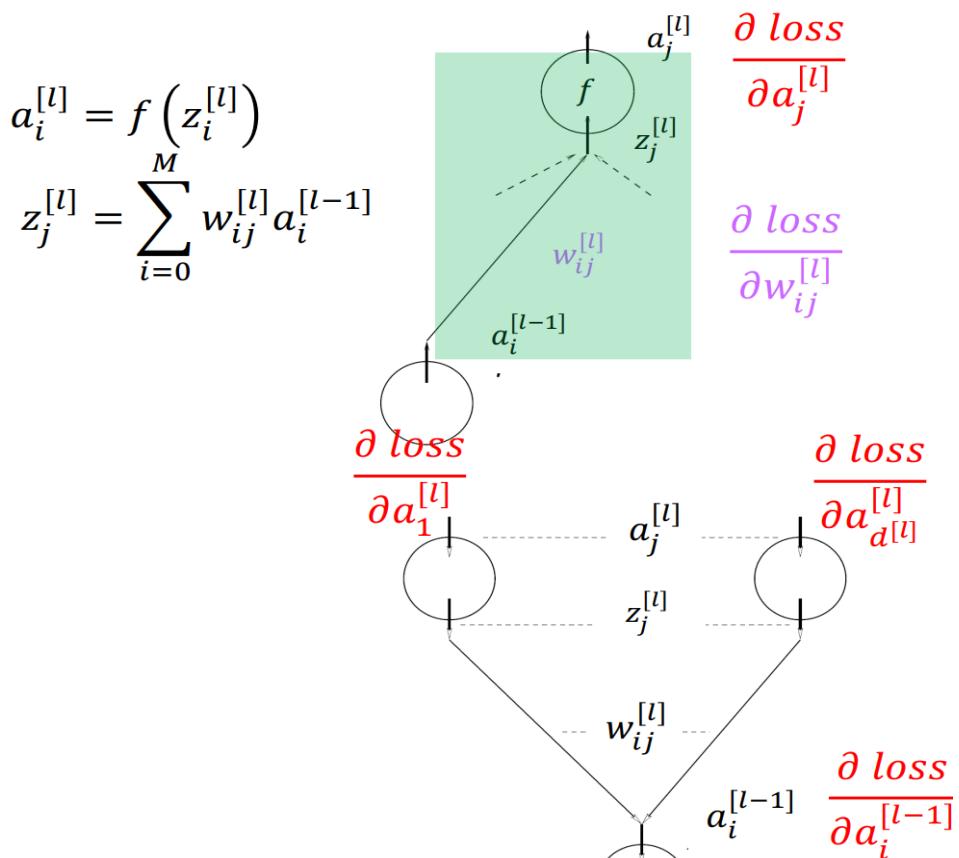
این الگوریتم تا زمان رسیدن به حداقل مقدار خطأ در شبکه ادامه می یابد تا در نهایت وزن های بهینه شبکه حاصل شوند .

در محاسبه عبارت مشتق خطأ نسبت به وزن های لایه های قبل از قاعده مشتق زنجیری استفاده می شود و مشتق خطأ نسبت به وزن های سایر لایه ها به صورت عقبگرد منتشر و محاسبه می شود [۱۹] .

$$\frac{dE}{dw_{i,j}^{[k]}} = \sum_{n=1}^N \frac{\text{loss}(o^{(n)}.y^{(n)})}{dw_{i,j}^{[k]}} \quad (4-5-4)$$



شکل ۱۷-۴ : پس انتشار خطأ در سطح یک نورون [۱۹]



شکل ۱۸-۴ : محاسبات پس انتشار خطای لایه های شبکه عصبی [۱۹]

$$\frac{\partial \text{loss}}{\partial w_{i,j}^{[l]}} = \frac{\partial \text{loss}}{\partial a_j^{[l]}} \frac{\partial a_j^{[l]}}{\partial w_{i,j}^{[l]}} \quad (4-5-5)$$

$$\frac{\partial a_j^{[l]}}{\partial w_{i,j}^{[l]}} = \frac{\partial a_j^{[l]}}{\partial z_j^{[l]}} \times \frac{\partial z_j^{[l]}}{\partial w_{i,j}^{[l]}} = f'(z_j^{[l]}) a_i^{[l-1]} \quad (4-5-6)$$

$$\begin{aligned} \frac{\partial \text{loss}}{\partial a_i^{[l-1]}} &= \sum_{j=1}^{d^{[l]}} \frac{\partial \text{loss}}{\partial a_j^{[l]}} \frac{\partial a_j^{[l]}}{\partial z_j^{[l]}} \frac{\partial z_j^{[l]}}{\partial a_i^{[l-1]}} \\ &= \sum_{j=1}^{d^{[l]}} \frac{\partial \text{loss}}{\partial a_j^{[l]}} \times f'(z_j^{[l]}) \times w_{i,j}^{[l]} \end{aligned} \quad (4-5-7)$$

تعریف می شود :

$$\delta_j^{[l]} = \frac{\partial \text{loss}}{\partial a_j^{[l]}} \quad (4-5-8)$$

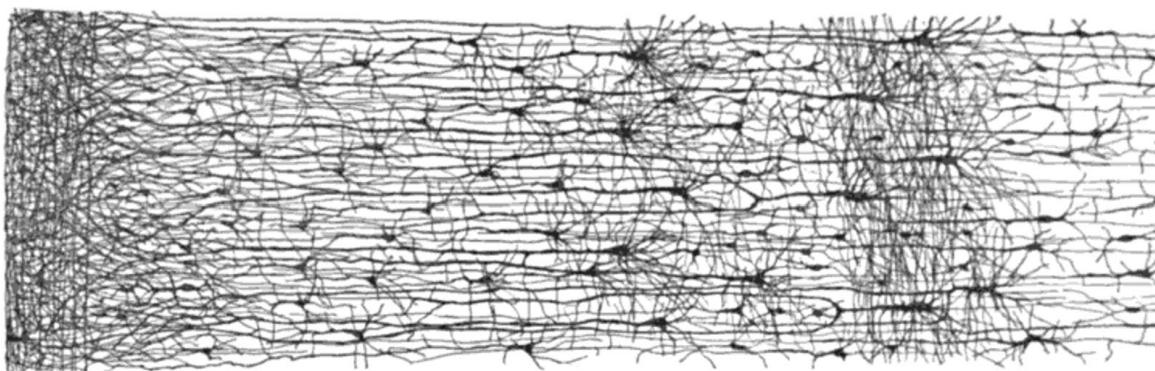
$$\frac{\partial \text{loss}}{\partial w_{i,j}^{[l]}} = \frac{\partial \text{loss}}{\partial a_j^{[l]}} \frac{\partial a_j^{[l]}}{\partial w_{i,j}^{[l]}} = \delta_j^{[l]} \times f'(z_j^{[l]}) \times a_i^{[l-1]} \quad (4-5-9)$$

$$\delta_i^{[l-1]} = \sum_{j=1}^{d^{[l]}} \delta_j^{[l]} \times f'(z_j^{[l]}) \times w_{i,j}^{[l]} \quad (4-5-10)$$

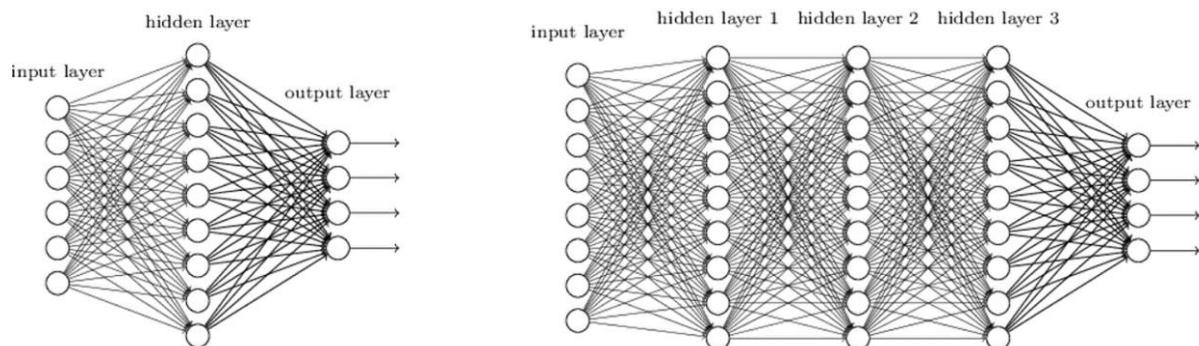
بر اساس رابطه (بالا) مشتق خطا نسبت به خروجی های هر لایه را می توان از مشتق خطا نسبت به لایه های بالاتر آن به دست آورد و انتشار خطا در لایه های ابتدایی شبکه را به دست آورد [۱۹].

فرآیند آموزش شبکه های عصبی خود شامل دو فرآیند پیشخور و پس انتشار خطا است که سبب بهینه شدن پارامتر های مدل می شود [۱۶].

به این ترتیب زمینه برای یادگیری و حرکت به سمت شبکه هایی با لایه های بیشتر فراهم شد و با در کنار هم قرار دادن شبکه های چند لایه شبکه های عصبی به سمت عمیق شدن پیش رفتند همانند مغز انسان که شامل تعداد زیادی از نورون ها و اتصالات است.

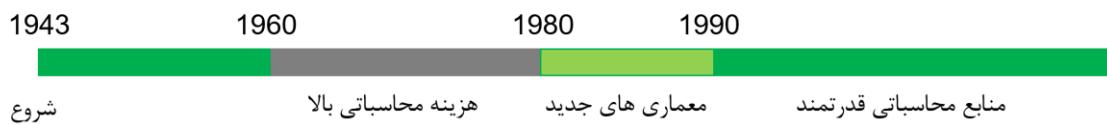


شکل ۱۹-۴ : نمونه واقعی یک شبکه عصبی عمیق [۷]



شکل ۲۰-۴ : افزایش تعداد لایه های مخفی در یک شبکه عصبی عمیق [۱۹]

هم زمان با ارائه الگوریتم ها و معماری های جدید چالش هایی مانند نیاز به حجم بالا داده ، منابع محاسباتی قدرتمند نیز وجود داشت که با افزایش قدرت منابع محاسباتی و افزایش حجم اطلاعات و داده های موجود و در دسترس به تدریج این مسائل قابل بررسی و حل شدند .



شکل ۲۱-۴ : چالش های شبکه های عصبی عمیق

۴-۶ روش های بهینه سازی وزن ها :

انواع الگوریتم مبتنی گرادیان کاهشی :

گرادیان کاهشی تصادفی :

در هر مرحله از اجرای این الگوریتم یک داده آموزش به عنوان ورودی به صورت تصادفی انتخاب می شود و الگوریتم با استفاده از همان یک داده انتخاب شده فرآیند به روزرسانی وزن ها را انجام می دهد .

ابتدا ترتیب داده ها عوض می شود تا عملیات انتخاب داده در الگوریتم تصادفی باشد سپس الگوریتم تکراری زیر برای به روزرسانی وزن های مدل برای تعداد N داده انجام می شود تا زمانی که مقادیر خطا مدل همگرا شود .

for all $n = 1:N$:

$$w = w - \eta \nabla_w loss(o^{(n)}.y^{(n)}) \quad (4-6-1)$$

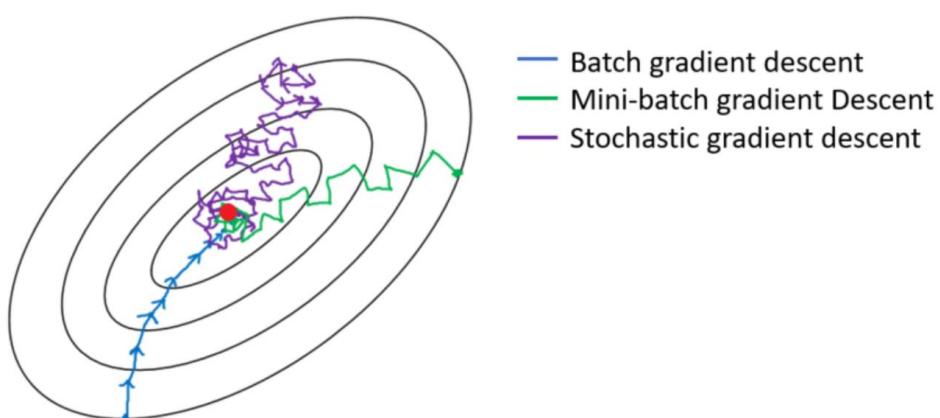
با توجه به انتخاب یک داده در هر مرحله از تکرار الگوریتم نیاز داریم خطا مربوط به هر داده محاسبه و ذخیره شود که این خود باعث افزایش زمان اجرا این الگوریتم می شود .

گرادیان کاهشی دسته ای :

در این الگوریتم برخلاف گرادیان کاهشی تصادفی تمام داده های آموزش به عنوان ورودی به الگوریتم گرادیان کاهشی داده می شود که باعث افزایش زمان یادگیری و به روزرسانی وزن های مدل می شود اما دقت همگرایی بالایی نسبت به الگوریتم تصادفی دارد [۱۳] و [۱۶].

گرادیان کاهشی با دسته های کوچک :

در این الگوریتم تعدادی از نمونه های ورودی به صورت دسته ای و تصادفی انتخاب می شوند و به سپس به الگوریتم گرادیان کاهشی داده می شوند و عملیات به روزرسانی وزن ها انجام می شود . این الگوریتم علاوه بر دقت مناسب از سرعت خوبی نیز نسبت به گرادیان کاهشی تصادفی برخوردار است [۱۶] .



شکل ۲۲-۴ : مقایسه الگوریتم های مبتنی بر گرادیان کاهشی [۱۹]

الگوریتم Adam :

الگوریتم Adam که در [۲۰] بحث شده است در گروه الگوریتم های تطبیق پذیر قرار می گیرد به این صورت که سعی می کند در طول الگوریتم نرخ یادگیری را تا جای ممکن اصلاح کند و سریعتر به نقاط بهینه دست یابد.

در این الگوریتم ابتدا یک ϵ کلی به عنوان مقدار گام اولیه در نظر گرفته می شود و همچنین دو مقدار ρ_1 و ρ_2 که مقادیری بین صفر و یک می توانند داشته باشند در صورتی که در [۲۰] به ترتیب مقادیر ۰,۹ و ۰,۹۹۹ پیشنهاد شده است و یک ثابت کوچک δ برای عدم واگرایی کسر مربوط به محاسبه $\Delta\theta$ در نظر گرفته می شود و همچنین مقادیر اولیه s و r صفر در نظر گرفته می شود.

در نهایت این الگوریتم تا زمانی که شرایط توقف برقرار نیست ادامه پیدا می کند و هر بار دسته ای از داده های آموزش $\{x^{(m)}, y^{(m)}\}$ را به عنوان ورودی و خروجی های مربوط به آن ها (i) دریافت می کند.

$$g \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$$

$$t \leftarrow t + 1$$

$$s \leftarrow \rho_1 s + (1 - \rho_1) g$$

$$r \leftarrow \rho_2 r + (1 - \rho_2) g \odot g \quad (4-6-2)$$

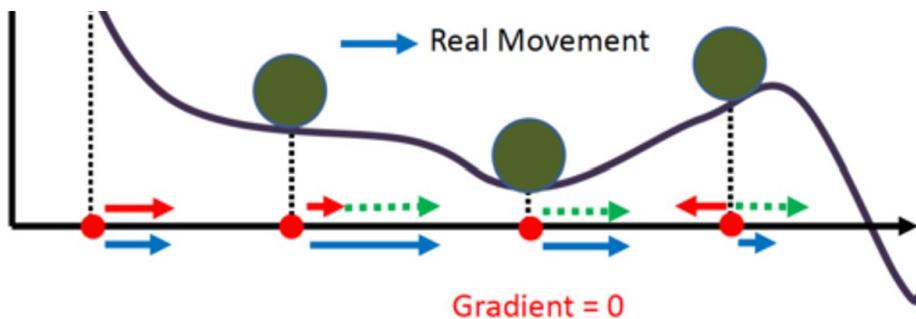
$$\hat{r} \leftarrow \frac{r}{1 - \rho_2^t} \quad . \quad \hat{s} \leftarrow \frac{s}{1 - \rho_1^t}$$

$$\Delta\theta = -\epsilon \frac{\hat{s}}{\sqrt{\hat{r}} + \delta}$$

$$\theta \leftarrow \theta + \Delta\theta$$

الگوریتم در مرحله اول گرادیان مرتبه اول را با توجه به داده های ورودی محاسبه می کند و در مرحله بعد به شمارنده الگوریتم یک واحد می افزاید.

در مرحله دوم الگوریتم با توجه به ضریب تنظیم ρ_1 هر بار ضریبی از گرادیان مرتبه اول فعلی را با گرادیان محاسبه شده در مراحل قبل جمع می کند و آن را در متغیر s قرار می دهد که این مرحله به الگوریتم کمک می کند با محاسبه گرادیان فعلی و تنظیم آن گام های خود را در طول فرایند آموزش کنترل کند تا علاوه بر سرعت با کیفیت بیشتری به نقاط بهینه نزدیک شود.



شکل ۲۳-۴ : افزایش سرعت الگوریتم Adam در رسیدن به نقاط بهینه [۲۰]

در مرحله سوم با در نظر گرفتن توان دوم گرادیان و تنظیم اثر مقدار فعلی این گرادیان در مقایسه با گرادیان محاسبه شده در مراحل قبل و محاسبه این مجموع و قرار دادن حاصل آن در متغیر r سعی شده است از توان دوم گرادیان برای کنترل نرخ یادگیری در طول فرایند آموزش استفاده شود.

با توجه به اینکه مقادیر s و r محاسبه شده در مراحل قبل تخمین گرهایی اریب برای گرادیان مرتبه اول و دوم هستند در مرحله چهارم این تخمین گرهایی ناریب تبدیل می‌شوند و در مرحله به روزرسانی وزن ها با در نظر گرفتن توان دوم گرادیان در مخرج کسر نرخ یادگیری کنترل می‌شود.

۷-۴ انواع داده ورودی به مدل

انواع داده برای آموزش و ارزیابی مدل [۲۱] :

داده های آموزش : جفت داده هایی که برای آموزش مدل و تنظیم پارامتر ها استفاده می‌شود.

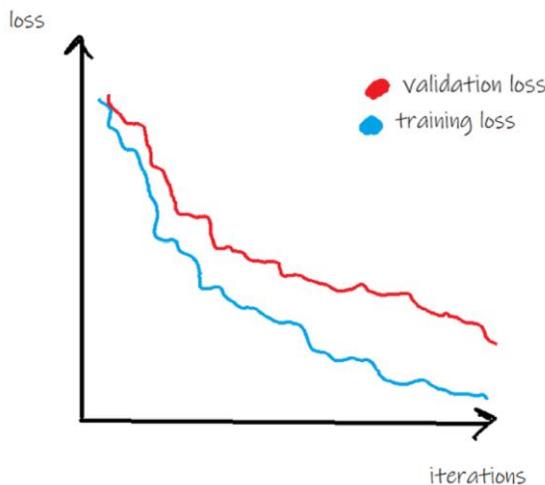
داده های اعتبارسنجی : جفت داده های شامل ورودی و خروجی که برای تنظیم ابرپارامتر های مدل مانند تعداد لایه های مخفی شبکه عصبی، نرخ یادگیری و ... استفاده می‌شود.

داده های سنجش : جفت داده های شامل ورودی و خروجی که برای ارزیابی عملکرد مدل استفاده می‌شود.

۸-۴ تعمیم پذیری در شبکه های عصبی عمیق

در [۲۲] بحث شده است که تعمیم پذیری شبکه عصبی عمیق عبارت است از توانایی شبکه و الگوریتم یادگیری در پیش بینی خروجی مطلوب برای داده های جدید که برای سنجش عملکرد و دقیقت شبکه مورد استفاده قرار می‌گیرند.

زمانی که خطای شبکه در هنگام فرآیند آموزش در حال کاهش باشد لزوماً به معنای آن نیست که مدل به خوبی داده های آموزش را آموخته است و می‌توان با استفاده از داده های اعتبارسنجی شبکه را مورد آزمایش قرار داد تا خطای شبکه را در برابر داده هایی که تا به حال مشاهده نکرده است ارزیابی کرد.



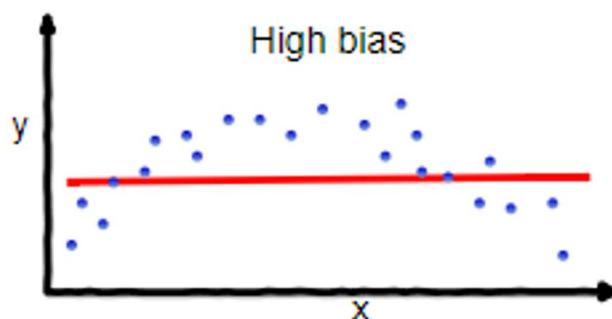
شکل ۲۴-۴ : افزایش تعمیم پذیری مدل با کاهش خطای اعتبارسنجی و آموزش [۲۵]

در طول هر تکرار کل داده های آموزش به مدل داده می شود و خطای حاصل از خروجی مدل به عنوان خطای کل مدل در نظر گرفته می شود .

۹-۴ تجزیه بایاس^۱ و واریانس^۲

: بایاس (Bias)

بایاس به صورت میانگین توان دوم اختلاف بین پیش بینی های مدل و مقادیر واقعی خروجی تعریف می شود و نشان می دهد که مدل به چه میزان داده های آموزش فرا گرفته است و افزایش بایاس به معنای عدم تعمیم پذیری شبکه به علت سادگی بیش از اندازه مدل و یا استفاده از داده هایی نامتناسب با مسئله میباشد به گونه ای که مدل در یادگیری داده ها آموزشی ناتوان است و در این حالت مدل دچار کم برآش شده است [۲۳] و [۲۴] .



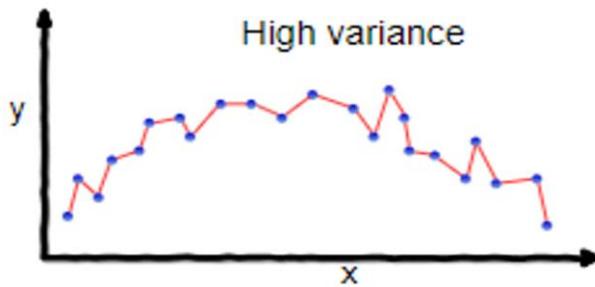
شکل ۲۵-۴ : عدم یادگیری در مدل های دچار کم برآش [۲۵]

¹Bias

²Variance

واریانس :

واریانس به معنای حساسیت مقادیر پیش بینی شده مدل نسبت به تغییرات کوچک در ورودی است و به این علت که مدل سعی در صرفا حفظ داده های آموزش و نه یادگیری آن دارد در برابر داده های اعتبارسنجی و ارزیابی عملکرد ضعیفی خواهد داشت و در این حالت مدل دچار بیش برازش شده است [۲۴] ، [۲۵].



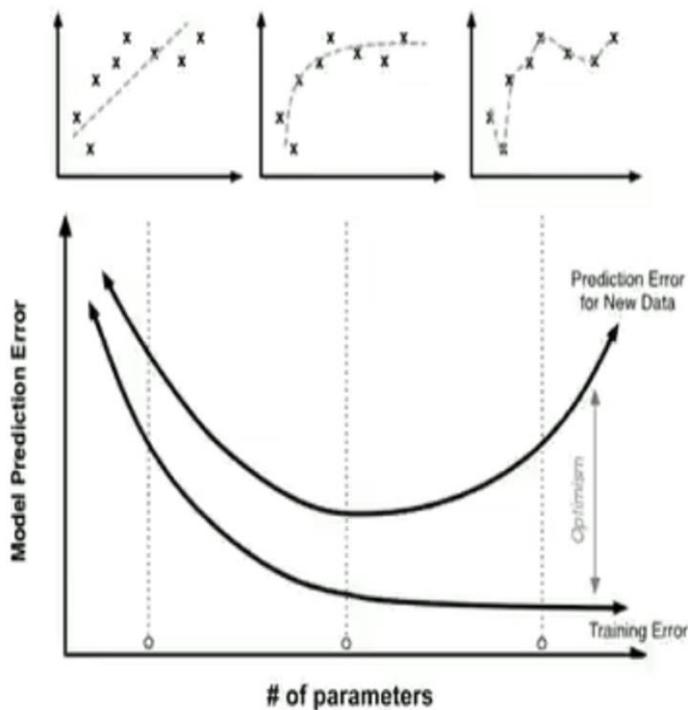
شکل ۲۶-۴ : عدم یادگیری در مدل های دچار بیش برازش [۲۵]

۱۰-۴ چالش های آموزش شبکه های عصبی عمیق

۱-۱۰-۴ بیش برازش مدل :

بیش برازش زمانی رخ می دهد که مدل همچنان در حال یادگیری داده های آموزشی است اما تعیین پذیری خوبی ندارد و مقادیر خطای بسیار بالایی را برای داده های اعتبارسنجی نتیجه می دهد و مقادیر این خطا در هر مرحله تکرار افزایش نیز می یابد .

هر چه تعداد پارامتر های مدل افزایش می یابد ، مدل سعی در یادگیری مسئله پیچیده تری دارد و در نتیجه آن پایداری سیستم در برابر داده های جدید کاهش پیدا خواهد کرد و مدل سعی در حفظ تمام داده های آموزشی با تمام جزئیات دارد که همین امر موجب بیش برازش مدل و کاهش دقت مدل خواهد شد [۲۶].



شکل ۲۷-۴ : تاثیر میزان برازش مدل در خطای خروجی [۱۹]

۴-۱۰-۴ روش های مقابله با بیش برازش :

روش تنظیم^۱ :

روش تنظیم وزن ها که شامل دو روش L1 و L2 می شود یکی از روش های جلوگیری از بیش برازش مدل است که در شبکه های عصبی عمیق با استفاده از جریمه وزن هایی که در شبکه مقادیر بزرگ تری به نسبت سایر وزن ها پیدا کرده اند .

وزن های مربوط به نورون هایی که بیشتر از سایر نورون ها تحت تاثیر داده هایی خاص قرار گرفته اند را کاهش می دهند [۱۶].

روش تنظیم L1 :

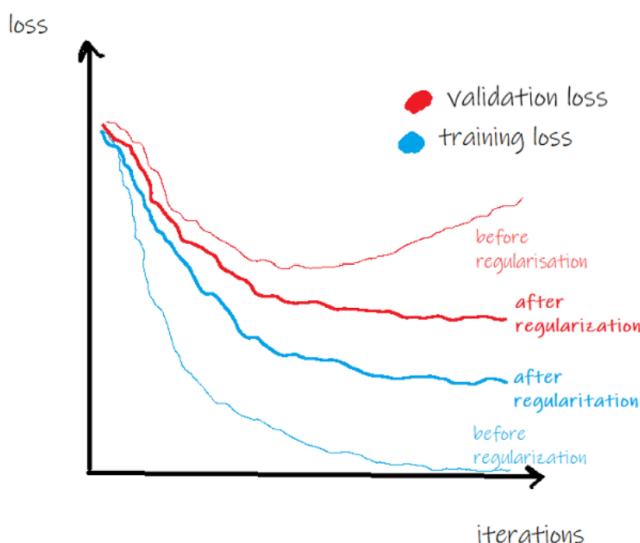
در این روش با اضافه کردن پارامتری دیگر به مقدار تابع خطا مدل سعی می شود از وابستگی مدل به داده هایی خاص و ویژگی های خاص جلوگیری شود .

استفاده از روش L1 کمک می کند پارامتر های مدل پراکندگی کمتری داشته باشند و همین امر باعث انتخاب ویژگی های مهم تر با حذف برخی از ویژگی ها می شوند .

^۱ Regularization

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^N |w_i| \quad (4-10-1)$$

ضریب λ که به عنوان یک ابرپارامتر شناخته می شود و قابل تنظیم و بهینه سازی است مقدار اهمیت خطا مدل به جمله دوم و یا همان وزن های شبکه را مشخص می کند و با افزایش آن می توان جریمه بیشتری را برای وزن ها در نظر گرفت و معمولاً مقادیر کوچک تر از یک برای آن در نظر گرفته می شود و در صورتی که مقدار آن مناسب با مدل و داده های ورودی نباشد ممکن است منجر به کاهش تعمیم پذیری مدل شود و مدل دچار کم برآش شود [۲۷] و [۱۶].

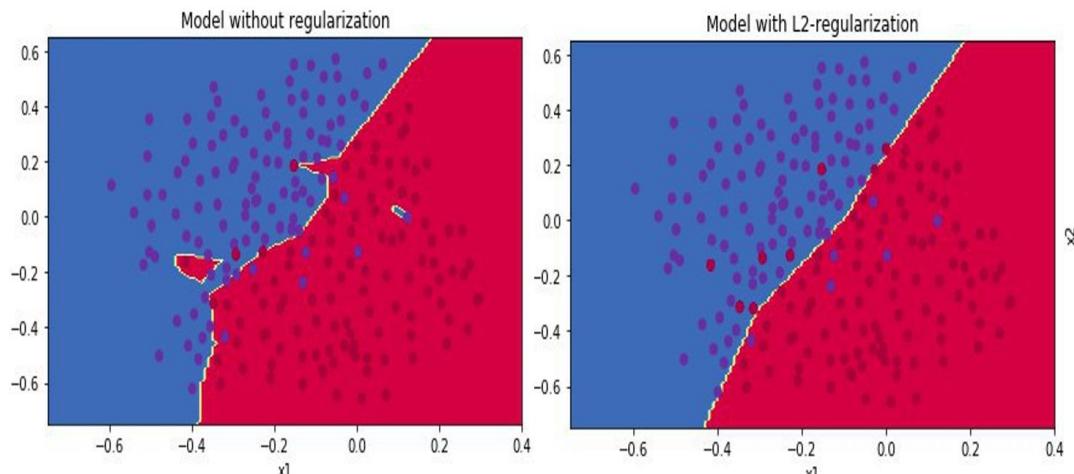


شکل ۴-۲۸ : تاثیر روش تنظیم در نزول خطا اعتبارسنجی [۲۵]

روش تنظیم L2

این روش نیز همانند روش L1 سعی در کاهش وابستگی مدل به داده ها و ویژگی های خاص دارد و از مهم ترین مزیت های آن می توان به استفاده از آن در مدل های پیچیده تر نسبت به روش L1 اشاره کرد با توجه به استفاده از توان دوم وزن ها در جمله دوم، علاوه بر جریمه پارامتر های مدل اما به اندازه روش L1 پراکندگی پارامتر ها را کاهش نمی دهد که این خود باعث می شود همه ویژگی های مدل در خروجی اثرگذار باشند و از بیش برآش مدل های با پیچیدگی بالاتر جلوگیری کند [۲۷] و [۱۶].

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^N w_i^2 \quad (4-10-2)$$

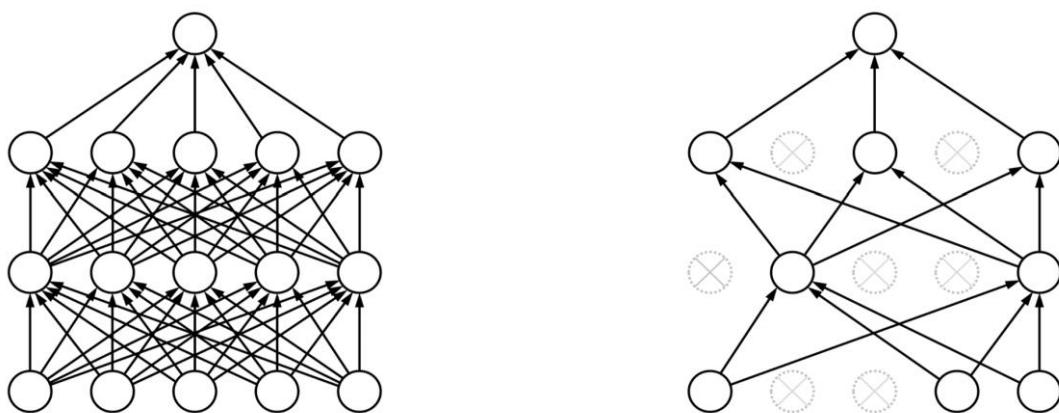


شکل ۲۹-۴ : تاثیر روش تنظیم L2 در دسته بندی داده های ورودی مدل [۱۳]

: (Dropout) روش حذف تصادفی

در [۲۸] بحث شده است که با توجه به اینکه در مدل های یادگیری عمیق تعداد پارامتر ها بسیار زیاد است ، مدل نیاز دارد به همان خوبی که مجموعه آموزشی را فرا گرفته است پیش بینی کند .

در این روش به صورت تصادفی P درصد از نورون های هر لایه مخفی انتخاب و حذف می شود .



شکل ۳۰-۴ : عملکرد روش حذف تصادفی [۲۸]

شبکه های عصبی و بویژه شبکه های عصبی عمیق همانطور که لایه به لایه مقادیر خروجی را محاسبه می کنند در واقع مجموعه ای جدید از ویژگی ها را در نظر می گیرند در نتیجه در آخرین لایه های مخفی یک سری ویژگی های سطح بالا یا پیچیده را استخراج می کنند . در این تکنیک با اجبار شبکه به داشتن بازنمایی دارای افزونگی سعی می شود شبکه اطلاعات را از روی سایر ویژگی ها استخراج کند و همین امر باعث افزایش قابلیت تعمیم پذیری مدل می

شود و در واقع در این تکنیک به جای آموزش یک مدل ، چند مدل را آموزش می دهیم و برای بدست آوردن نتایج از خروجی تمامی مدل های بدست آمده میانگین گرفته می شود .

با توجه به اینکه در زمان پیش بینی ، همه نورون ها فعال هستند باید حذف شدن نورون ها در زمان آموزش شبکه به گونه ای جبران شود .

با فرض نرخ p حذف تصادفی نورون های هر لایه در فرآیند آموزش برای جبران حذف شدن نورون ها در می توان مقدار فعالیت نورون ها در زمان پیش بینی در ضرب $\frac{1}{1-p}$ نمود .

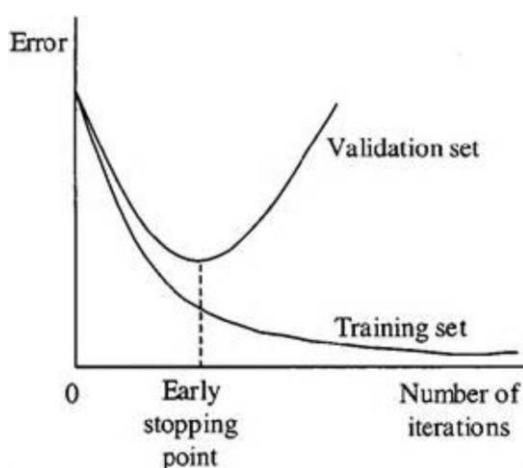
روش حذف تصادفی مونت کارلو (Monte Carlo dropout)

در این روش علاوه بر اعمال روش dropout در آموزش مدل ، در هنگام ارزیابی مدل نیز از این روش استفاده می شود با این تفاوت که با توجه به اعمال dropout بار ها شبکه مورد ارزیابی قرار می گیرد و در نهایت میانگین خروجی تعداد زیادی شبکه به عنوان خروجی این روش در نظر گرفته می شود و در نتیجه پیش بینی های انجام شده در مرحله ارزیابی دیگر قطعی نیستند و بسته به اینکه کدام نورون ها در شبکه به صورت تصادفی انتخاب شوند خروجی شبکه در مرحله ارزیابی متفاوت خواهد بود .

بنابراین با استفاده از این روش می توان به عدم قطعیت مدل پی برد و ضعف های مدل را شناسایی کرد و در نهایت هدف اصلی این روش که تولید پیش بینی های تصادفی و تفسیر آن ها به عنوان نمونه هایی از یک توزیع احتمالاتی است [۲۹] .

روش توقف زود هنگام (Early stopping)

یک مسئله که موجب بیش برآش مدل می شود انتخاب تعداد تکرار های بالا برای آموزش شبکه است به گونه ای که با انتخاب تعداد بالا تکرار و آموزش شبکه با کل داده های آموزش در هر تکرار موجب کاهش تعمیم پذیری مدل و افزایش خطأ در داده های اعتبارسنجی می شود و انتخاب تعداد تکرار مناسب برای توقف فرآیند آموزش با مشاهده و تفسیر میزان خطأ مدل در هنگام آموزش و اعتبار سنجی میسر می شود [۱۶] .

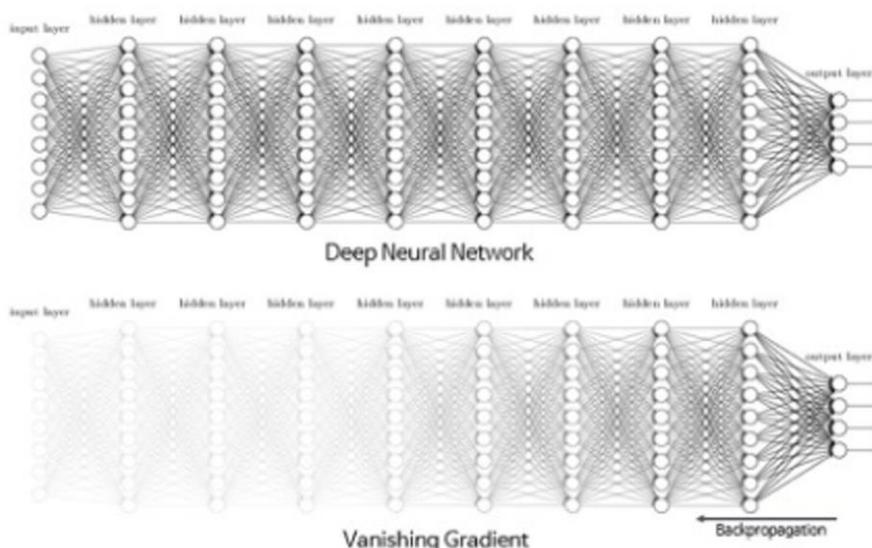


شکل ۳۱-۴ : تاثیر روش توقف زودهنگام در خطأ اعتبارسنجی [۲۷]

۴-۱۰-۳- انفجار و محوشوندگی گرادیان :

محوشوندگی گرادیان :

در هنگام اجرا الگوریتم پس انتشار خطا به صورت عقب گرد در یک شبکه عصبی عمیق نیازمند استفاده از مشتق جزئی و قاعده مشتق زنجیری برای به روز رسانی وزن های شبکه است و بر مبنای رابطه δ گرادیان هر لایه در گرادیان لایه های بعد خود ضرب می شود و با توجه به مقادیر اندک گرادیان در طول الگوریتم پس انتشار خطا گرادیان های محاسبه شده همواره کوچک تر می شوند به گونه ای که وزن های لایه های ابتدایی با مقادیر بسیار اندکی به روز رسانی می شوند و تقریباً تغییری در وزن های این لایه های ابتدایی در طول فرآیند یادگیری شبکه رخ نمی دهد و در نتیجه الگوریتم گرادیان کاهشی مورد استفاده به مقادیر بهینه مسئله همگرا نمی شود [۳۰] و [۳۱].



شکل ۳۲-۴ : تاثیر محوشوندگی گرادیان بر وزن های لایه های ابتدایی شبکه [۳۱]

انفجار گرادیان :

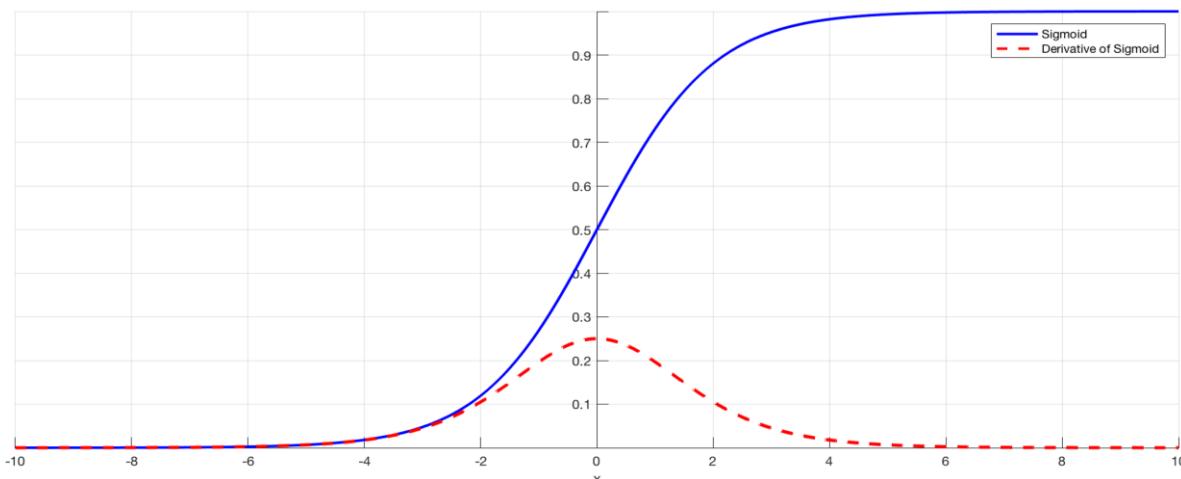
مقابل مسئله محو شدگی گرادیان مسئله انفجار گرادیان قرار دارد و زمانی رخ می دهد که مقادیر مشتق حاصل از الگوریتم پس انتشار خطا بیش از اندازه بزرگ باشند که با توجه به قاعده مشتق زنجیری منجر به واگرایی و یا نوسان حول نقاط بهینه الگوریتم گرادیان کاهشی برای به روز رسانی وزن های شبکه خواهد شد [۳۰].

۴-۱۰-۴ روش های مقابله با انفجار و محوشوندگی گرادیان :

مقدار دهی وزن های اولیه شبکه :

در [۳۲] نشان داده است که ترکیب تابع فعالسازی سیگموئید و مقدار دهی وزن های اولیه شبکه به صورت یک توزیع نرمال با میانگین صفر و انحراف معیار یک باعث می شود با پیشروی در شبکه واریانس مقادیر خروجی هر لایه افزایش یابد و با توجه به تابع فعالسازی سیگموئید که بازه بزرگی از ورودی ها را به یک بازه کوچک تر بین صفر و یک می نگارد مقادیر خروجی هر لایه وارد ناحیه اشباع این تابع فعالسازی شود .

بنابراین با توجه به مقادیر کوچک مشتق در نزدیکی ناحیه اشباع و مشتق صفر در خود ناحیه اشباع خود یک دلیل مهم برای محوشوندگی گرادیان در مرحله پس انتشار خطای خواهد بود .



شکل ۳۳-۴ : نمودار تابع فعالسازی سیگموئید و مشتق آن [۳۱]

در [۳۲] پیشنهاد داده شده است با کنترل واریانس ورودی و خروجی هر لایه سعی شود از محوشوندگی گرادیان تا حد امکان جلوگیری شود و پیشنهاد داده شده است با در نظر گرفتن fan_{in} به عنوان تعداد نورون های ورودی لایه مورد نظر و fan_{out} به عنوان تعداد نورون های لایه بعد و تعریف fan_{avg} با مقدار $\frac{fan_{in}+fan_{out}}{2}$ ، از یک توزیع نورمال با میانگین صفر و واریانس $\sigma^2 = \frac{1}{fan_{avg}}$ یا توزیع یکنواخت بین $-r$ و $+r$ استفاده شود که مقدار r به صورت $\sqrt{\frac{3}{fan_{avg}}}$ تعریف می شود .

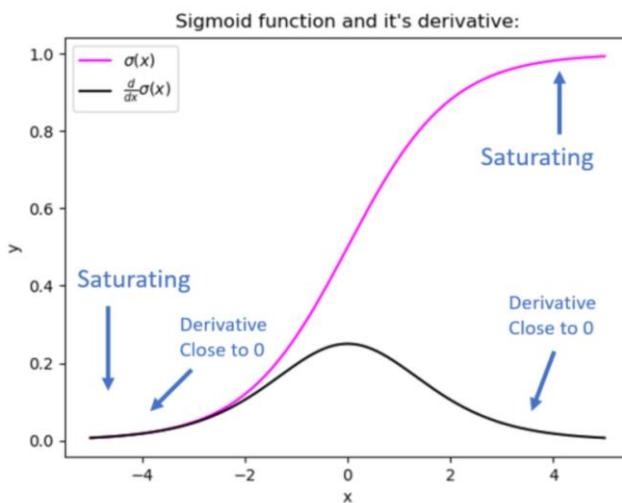
این شیوه مقدار دهی اولیه وزن ها باعث افزایش سرعت آموزش شبکه و همگرایی الگوریتم گرادیان کاهشی و افزایش تعمیم پذیری شبکه شده است .

در [۳۳] پیشنهاد داده شده است برای توابع فعالسازی از جنس ReLU می توان برای مقدار دهی اولیه وزن های شبکه از یک توزیع نرمال با میانگین صفر و واریانس $\frac{2}{fan_{in}}$ استفاده کرد .

توابع فعالسازی غیر قابل اشباع :

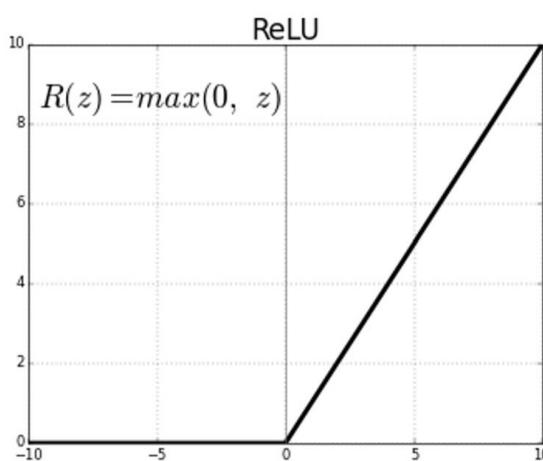
در بخش قبلی در مورد تابع فعالسازی سیگموئید و اشباع آن به معنای مشتق صفر در نواحی مثبت و منفی محور ورودی بحث شد و مشاهده شد که ناحیه وسیعی از مقادیر ورودی را به ناحیه محدودی می نگارد .

توابعی با بخش های قابل اشباع مانند سیگموئید و تانژانت هایپربولیک نقش قابل توجهی در محوشوندگی گرادیان در شبکه های عصبی عمیق دارند .



شکل ۳۴-۴ : نمودار تابع فعالسازی سیگموئید و مشتق آن به همراه نواحی اشباع [۳۱]

بنابراین در [۱۵] تابع فعالسازی Relu به عنوان یک توابع فعالسازی که ناحیه مثبت غیر قابل اشباع دارد معرفی شد .

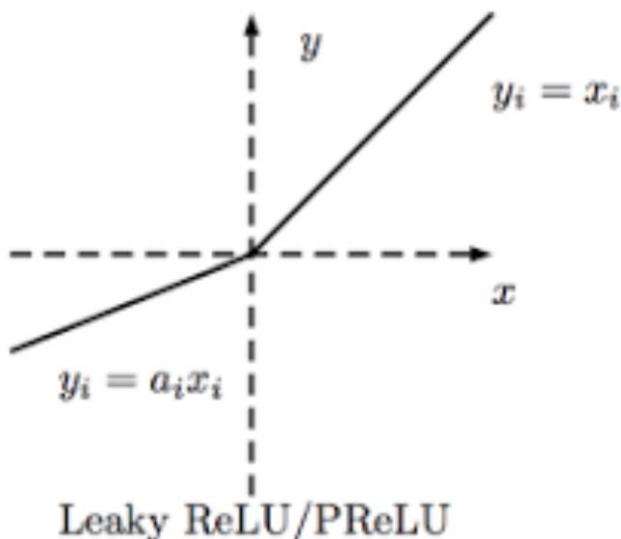


شکل ۳۵-۴ : نمودار تابع فعالسازی سیگموئید و مشتق آن [۳۱]

از مزایای تابع فعالسازی Relu می توان به افزایش سرعت یادگیری شبکه و مقابله با محوشوندگی گرادیان اشاره کرد اما یک ایراد این تابع فعالسازی ایجاد نورون هایی با خروجی صفر یا نورون های مرده است که در اثر صفر بودن این تابع در ناحیه ورودی های منفی بوجود می آید و منجر به خروجی صفر و بی اثر شدن برخی نورون ها در فرآیند یادگیری شبکه می شود .

در [۳۴] جایگزین دیگری به نام LReLU^۱ معرفی شده است که علاوه بر ناحیه مثبت ناچیه منفی آن هم دارای مشتقی ثابت است که علاوه بر جلوگیری از محوشوندگی گرادیان از ایجاد نورون های مرده نیز جلوگیری خواهد کرد اما یک ابر پارامتر که همان شبیب تابع در ناحیه منفی است به مدل اضافه خواهد کرد .

در [۳۴] نوع دیگری از تابع فعالسازی به نام PReLU^۲ معرفی شده است که در واقع همان نسخه LReLU است با این تفاوت که ابر پارامتر شبیب تابع در قسمت منفی به عنوان یک پارامتر در هنگام آموزش شبکه یادگرفته می شود .



شکل ۴-۳۶ : نمودار تابع PReLU و LReLU [۳۱]

نرمال سازی دسته ای ^۳ :

استفاده از مقدار دهی اولیه وزن ها با روش He و توابع فعالسازی از جنس Relu در ابتدا میتواند موجب کاهش محسوس انفجار و محوشدنگی گرادیان شود اما تضمینی برای عدم رخداد آن وجود نداشته است .

در [۳۵] روشی به نام نرمال سازی دسته ای معرفی شده است که برای مقابله با محوشدنگی و انفجار گرادیان و پایداری شبکه استفاده می شود .

در [۳۵] روشی معرفی شده است که چگونه قبل یا بعد از تابع فعالسازی در هر لایه خود مدل یاد بگیرد که چگونه داده ها را نرمال سازی کند ، در واقع در این روش با یادگیری دو پارامتر مقیاس و انتقال و نرمال سازی هر دسته داده با میانگین صفر داده های ورودی به هر لایه نرمال شده است .

¹ Leaky Rectified Linear Unit

² Parametric Rectified Linear Unit

³ Batch Normalization

در الگوریتم نرمال سازی دسته ای m تعداد داده های هر دسته وارد شده به الگوریتم و γ و β پارامتر های آن هاستند .

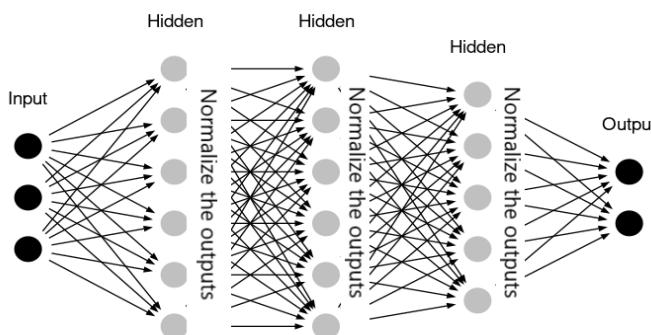
$$\mu_B \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad (4-10-3)$$

$$\sigma_B^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2 \quad (4-10-4)$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad (4-10-5)$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv BN_{\gamma, \beta}(x_i) \quad (4-10-6)$$

در نهایت برای استفاده از میانگین و واریانس دسته داده های ورودی به نرمال ساز با اعمال یک میانگین گیری تجمعی در همه دسته ها میانگین و واریانس نهایی برای استفاده از آن در مرحله ارزیابی مدل محاسبه شده است .



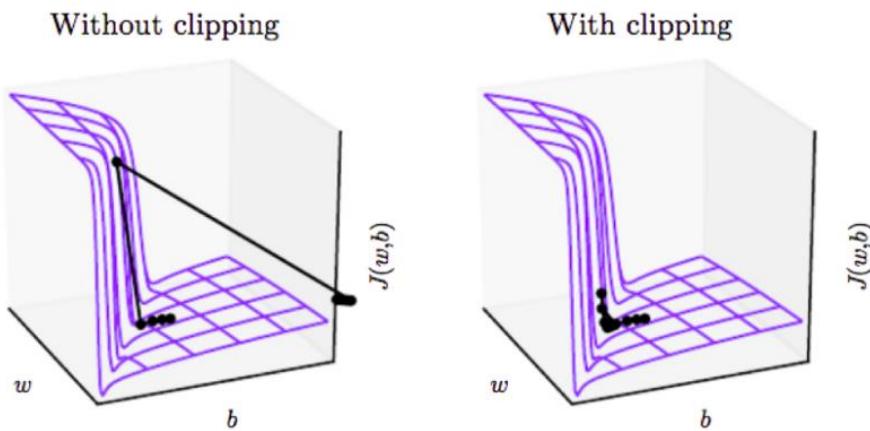
شکل ۳۷-۴ : اعمال لایه نرم سازی دسته ای بعد از هر لایه مخفی

برش گرادیان^۱:

به عنوان یک روش برای جلوگیری از انفجار گرادیان در شبکه ها می توان از برش گرادیان استفاده کرد به این صورت که بهینه ساز شبکه می تواند در هر مرحله وابسته به اینکه چه مقداری به عنوان مرز مقادیر گرادیان به عنوان ابرپارامتر به آن بهینه ساز داده می شود برش گرادیان را با تغییر گرادیان موجود به مرز های تعیین شده

¹ Gradient Clipping

انجام دهد به عنوان مثال اگر مرز تعیین شده برابر یک باشد بزرگترین مقدار گرادیان را به یک تغییر می دهد و سایر مقادیر را با توجه به مقیاس آن ها بین صفر و یک قرار می دهد .



شکل ۳۸-۴ : برش مقادیر گرادیان [۳۶]

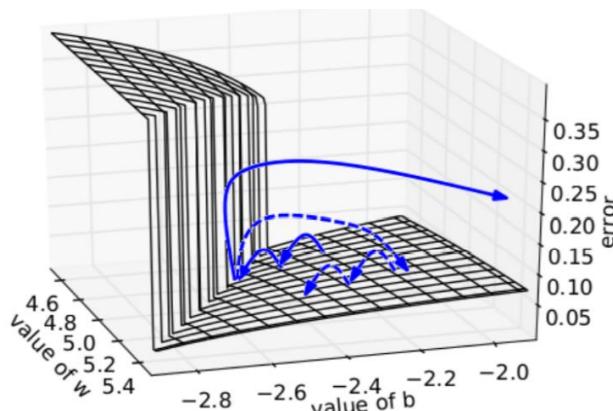
علاوه بر روش فوق می توان از نرم گرادیان نیز برای برش گرادیان استفاده نمود که علاوه بر جلوگیری از انفجار مقادیر گرادیان ، جهت گرادیان را برخلاف روش قبل حفظ می کند و باعث بهبود عملکرد مدل در بهینه سازی می شود و الگوریتم آن به صورت زیر می باشد [۳۶].

$$\hat{g} \leftarrow \frac{\partial \varepsilon}{\partial \theta} \quad (4-10-7)$$

if $\|\hat{g}\| \geq threshold$ *then :*

$$\hat{g} \leftarrow \frac{threshold}{\|\hat{g}\|} \hat{g} \quad (4-10-8)$$

end if



شکل ۳۹-۴ : برش گرادیان به روش نرم گرادیان [۳۶]

۴-۱۰-۵- داده های محدود

شبکه های عصبی عمیق به داده های بسیار زیادی برای یادگیری دقیق یک مسئله نیاز دارند و در برخی از مسائل تعداد بالا داده که بتوان با آن شبکه عمیق را آموزش داد وجود ندارد و یا در صورت وجود ممکن است برچسب گذاری آن داده ها فرایندی سخت و زمان بر باشد .

روش مقابله با داده های محدود :

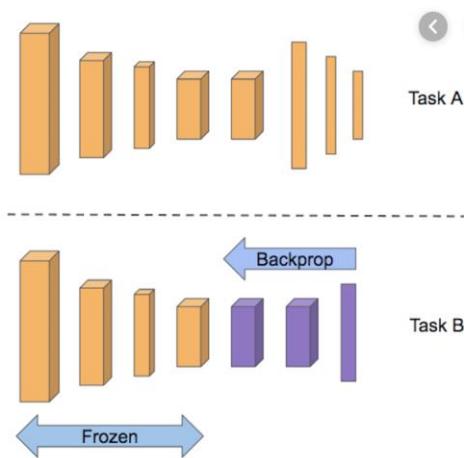
یادگیری انتقالی^۱:

در [۳۷] بحث شده است که یادگیری انتقالی یک رویکرد برای انتقال دانش از یک مدل به مدلی دیگر است و استفاده از یادگیری انتقالی این امکان را می دهد که با استفاده از مدلی که یک مسئله را با دقتی بالا پیش بینی می کند بتوان مسئله های با دامنه یکسان یا مشابه را حل کرد .

ساده ترین روش برای حل یک مسئله با استفاده از یادگیری انتقالی استفاده از یک مدل آموزش دیده است ، مدل هایی که معمولا با میلیون ها پارامتر آموزش دیده اند و زمان زیادی برای آموزش این نوع مدل ها صرف شده است و دقت بالایی دارند .

در روشهای دیگر از یک مدل آموزش دیده صرفا برای استخراج ویژگی استفاده شده است و در لایه آخر آن از یک دسته بند دیگر مانند ماشین بردار پشتیبان استفاده شده است تا با بوسیله آن خروجی لازم مدل را برای مسئله ای دیگر بدست آورد .

و در روشهای دیگر می توان یک قدم فراتر رفت و چند لایه آخر مدل آموزش دیده را دوباره تنظیم و آموزش داد ، با توجه به اینکه مدل در لایه های ابتدایی خود ویژگی های کلی داده ها را بدست آورده است با آموزش و تنظیم وزن های لایه های اخر شبکه می توان برای داده های مسئله ای خاص تر خروجی را پیش بینی نمود .



شکل ۴۰-۴ : یادگیری انتقالی و تنظیم لایه های انتهایی شبکه [۳۷]

¹ Transfer Learning

۴-۱۰-۶ چالش زمان در آموزش شبکه های عصبی عمیق

با افزایش پارامتر های شبکه های عصبی عمیق و پیچیده شدن آن ها از نظر تعداد نورون و لایه ها و انواع اتصالات میان لایه ها ، زمان آموزش شبکه ها یکی از پارامتر هایی است که همواره مورد توجه بوده است و نیاز به نوعی برقراری مصالحه میان زمان آموزش شبکه های عمیق و دقت خروجی آن ها وجود دارد .

روش های کاهش زمان آموزش شبکه های عمیق :

انتخاب بهینه ساز مناسب برای شبکه :

انتخاب بهینه ساز مناسب با توجه به مدل می تواند علاوه بر سرعت بخشیدن به فرآیند یادگیری موجب افزایش دقت مدل نیز بشود .

برنامه ریزی نرخ یادگیری الگوریتم بهینه سازی :

همانطور که مطرح شد یکی از مسائلی که در شبکه های عصبی عمیق مطرح است تنظیم ابرپارامتر های مسئله مانند تعداد لایه های مخفی ، نوع تابع فعالسازی ، نوع بهینه ساز ، نرخ یادگیری و .. است.

تنظیم مناسب ابرپارامتر هایی مانند نرخ یادگیری منجر به بهبود عملکرد شبکه از نظر سرعت و دقت خواهد شد .

نرخ یادگیری با کنترل گام های بهینه ساز مسئله منجر به رسیدن تابع خطا به حداقل مقدار خود می شود به گونه ای که مقادیر زیاد نرخ یادگیری منجر به واگرایی و نوسان حول نقطه بهینه و مقادیر اندک نرخ یادگیری منجر به کاهش سرعت الگوریتم و در موقعي عدم حصول نقطه بهینه برای مسئله بهینه سازی خواهد شد .

برنامه ریزی نرخ یادگیری با الگوریتمی از پیش تعیین شده منجر به تنظیم نرخ یادگیری در هنگام اجرای تکرار های مورد نیاز فرآیند آموزش می شود .

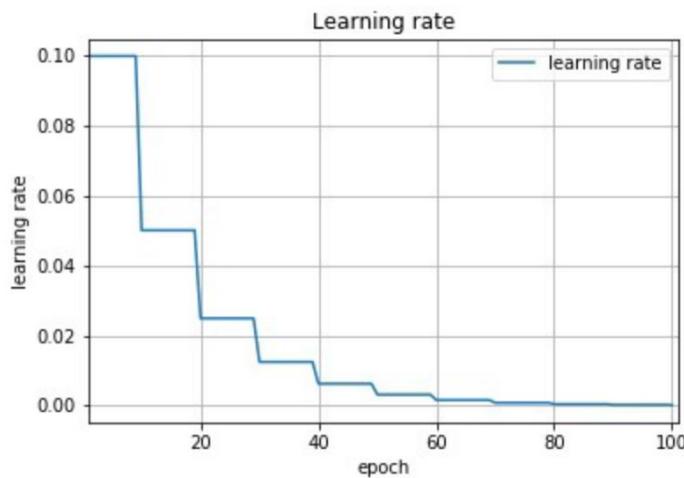
در [۳۸] انواع الگوریتم های برنامه ریزی نرخ یادگیری بحث شده است که شامل انواع برمبنا گام ، برمبنا زمان و برنامه ریزی نمایی است .

در برنامه ریزی نرخ یادگیری بر مبنا زمان از فرمول زیر برای برنامه ریزی استفاده می شود که در آن η_n نرخ یادگیری ، d پارامتر کاهش و n گام تکرار است .

$$\eta_{n+1} = \frac{\eta_n}{1 + dn} \quad (4-10-9)$$

در برنامه ریزی بر مبنا گام ، نرخ یادگیری بر اساس گام های از پیش تعریف شده تغییر خواهد کرد که در آن η_n نرخ یادگیری در تکرار n ام ، η_0 نرخ یادگیری اولیه و d مقدار تغییر نرخ یادگیری در هر گام است.

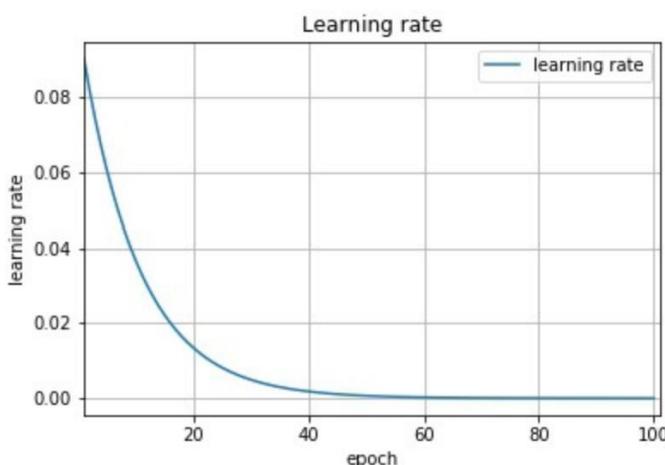
$$\eta_n = \eta_0 d^{\left\lfloor \frac{1+n}{r} \right\rfloor} \quad (4-10-10)$$



شکل ۴۱-۴ : نمودار تغییر نرخ یادگیری در برنامه ریزی بر مبنای گام [۳۸]

در برنامه ریزی نمایی نرخ یادگیری ، نرخ یادگیری بر مبنای یکتابع نمایی تغییر خواهد کرد که d پارامتر کاهش در نرخ یادگیری است .

$$\eta_n = \eta_0 e^{-dn} \quad (4-10-11)$$



شکل ۴۲-۴ : نمودار تغییر نرخ یادگیری در برنامه ریزی نمایی [۳۸]

با وجود برنامه ریزی های متفاوت برای نرخ یادگیری ، می توان از بهینه ساز هایی با در نظر گرفتن نرخ یادگیری وفقی مانند بهینه ساز Adam نیز استفاده کرد .

۴-۱۱-۴ معرفی شبکه های عصبی بازگشتی :

توسعه شبکه های چند لایه پرسپترون نقطه عطف مهمی در شبکه های عصبی مصنوعی بود که منجر به کنار هم قرار دادن این لایه ها و حل مسائل پیچیده تر شد .

شبکه های عصبی چند لایه پرسپترون به صورت محلی به این معنا که با نگاه به ورودی در یک زمان یا مکان خاص عمل می کنند به این صورت که یک بردار ورودی دریافت می کنند و محاسبات به سمت انتهای شبکه و رو به جلو انجام می شود و سپس پس انتشار خطا خروجی محاسبه می شود و زمانی که تابع خطا حداقل مقدار خود را دارد متوقف می شود و خروجی را تولید می کند .

اما در بسیاری از مسائل دنیای واقعی بعد زمان یا مکان نیز دخیل است و هر کدام از ورودی های مسئله ممکن است در زمان ها یا مکان های متفاوت به صورت یک توالی از ورودی ها به مدل وارد شوند .

به عنوان مثال می توان مسئله تحلیل احساسات نظرات کاربران را ذکر کرد ، ورودی این مسئله برای تحلیل جملات کاربران به صورت یک توالی از کلمات در مکان های متفاوت است . لزوماً توالی های ورودی توالی هایی در بعد زمان نیستند و ممکن است در بعد مکان دارای توالی باشند ، به عنوان نمونه برای توالی های ورودی در بعد زمان می توان به بررسی شاخص بورس برای هر سه‌ماه در روز های متفاوت اشاره کرد .

بنابراین نیاز به مدل هایی داریم که به ورودی های مسئله به جای یک ورودی واحد همانند یک توالی از ورودی ها نگاه کنند و علاوه بر ورودی در همان لحظه ، ورودی لحظات قبل را نیز در نظر بگیرند و حتی قادر به ارائه خروجی مدل به صورت یک توالی باشند مانند زمانی که نیاز به حل مسائلی مانند ترجمه باشد .

شبکه های عصبی بازگشتی با دریافت ورودی برای محاسبه هر خروجی علاوه بر ورودی و سایر پارامتر های مدل در همان لحظه به محاسبات لحظات گذشته نیز وابسته هستند [۳۹] .

اگر مدل را در فضا حالت در نظر بگیریم و حالت مدل در هر لحظه را وابسته به لحظه قبل و پارامتر های مدل بدانیم می توان به شکل زیر آن را در نظر گرفت [۱۸] :

$$s^{(t)} = f(s^{(t-1)}; \theta) \quad (4-11-1)$$

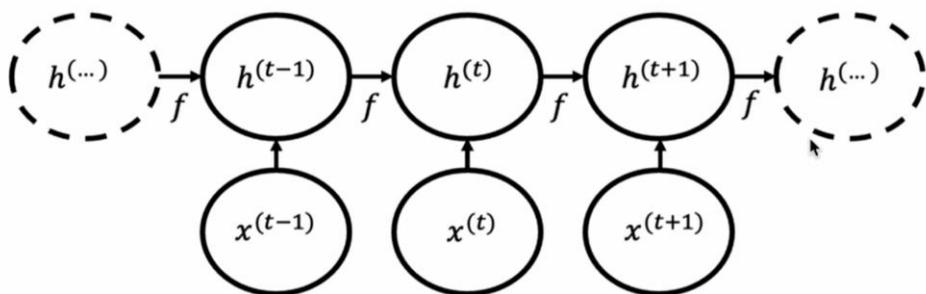


شکل ۴-۴ : مدل فضاحالت در شبکه های عصبی بازگشتی [۱۸]

^۱ Recurrent Neural Network(RNN)

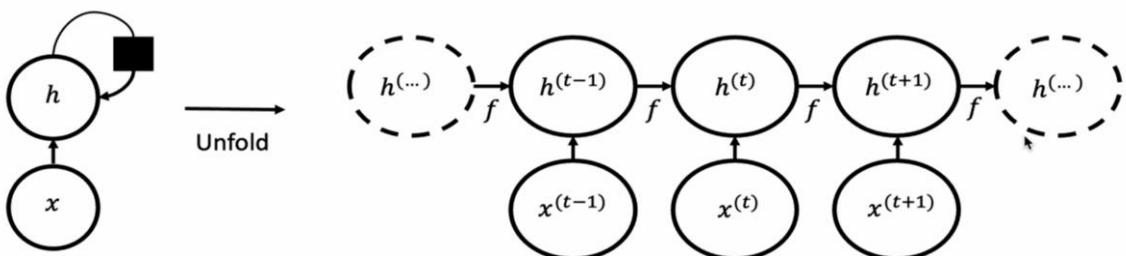
همچنین با اعمال توالی از ورودی ها به مدل و در نظر گرفتن هر حالت به عنوان لایه ای مخفی می توان آن را به صورت شکل زیر در نظر گرفت :

$$h^{(t)} = f(h^{(t-1)}, x^{(t)}; \theta) \quad (4-11-2)$$



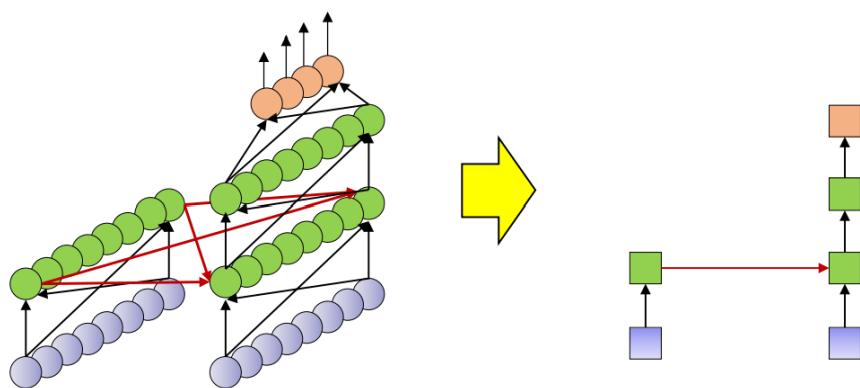
شکل ۴۴-۴ : مدل فضاحالت در شبکه های عصبی بازگشتی [۱۸]

برای خلاصه سازی این نوع مدل از گراف های تا شده استفاده می شود .



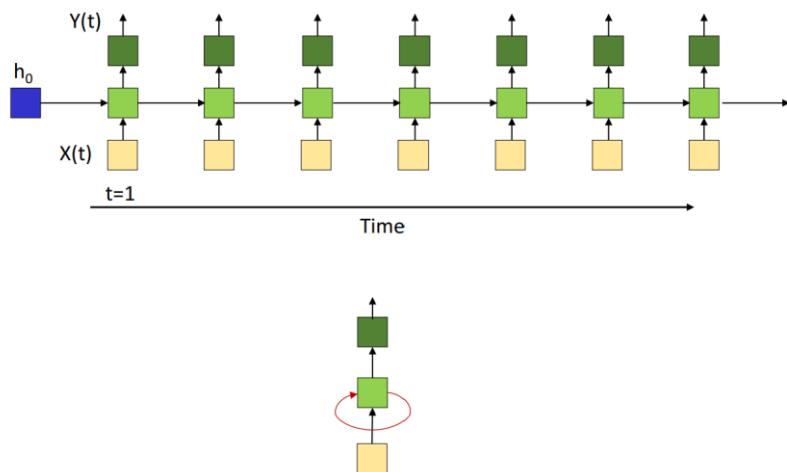
شکل ۴۵-۴ : مدل تا شده فضاحالت در شبکه های عصبی بازگشتی [۱۸]

یک راه مناسب برای به تصویر کشیدن و خلاصه سازی اتصالات این نوع شبکه ها استفاده از گراف های محاسباتی است و می توان هر کدام از لایه ها که مجموعه ای از نورون ها است را به صورت مربع هایی در شبکه نشان داد .



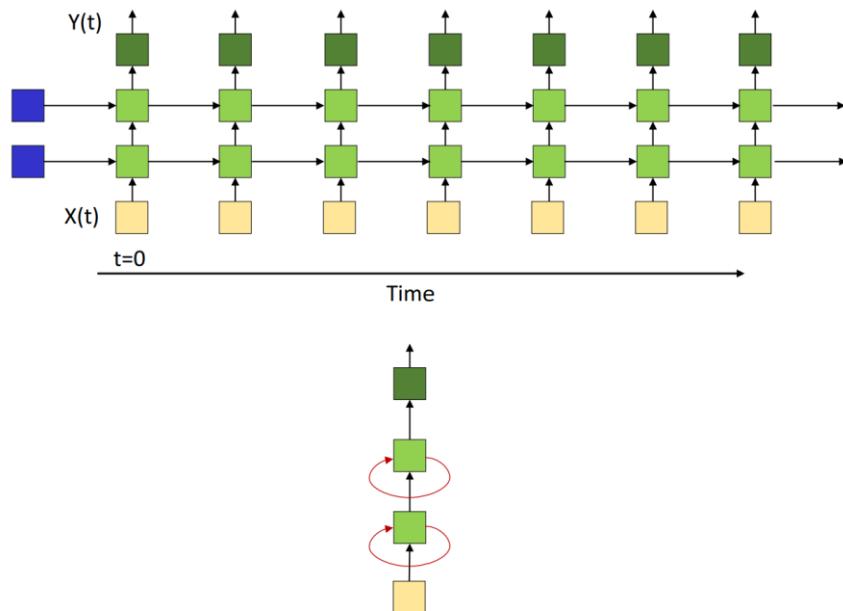
شکل ۴۶-۴ : گراف مدل شده شبکه های عصبی بازگشتی [۱۹]

شبکه های عصبی بازگشتی با یک لایه مخفی که نگه دارنده حالات مدل در بعد زمان یا مکان هستند را به صورت زیر نمایش می دهند که در آن بردار X شامل ورودی ها در لحظات متفاوت و بردار Y شامل خروجی های مدل در لحظات متفاوت و h_0 حالت اولیه مدل است.



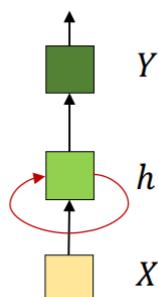
شکل ۴۷-۴ : گراف مدل شده شبکه های عصبی بازگشتی [۱۹]

همچنین شبکه های عصبی بازگشتی با چند لایه مخفی به صورت زیر نمایش داده می شوند ، به این صورت که مجموعه نورون های لایه مخفی دوم علاوه بر ورودی از مجموعه نورون های لایه مخفی قبل تا زمان خاص t در یک توالی تاثیر می پذیرند .



شکل ۴-۴ : گراف مدل شده شبکه عصبی بازگشتی چند لایه [۱۹]

در این نوع شبکه ها که هر لایه مجموعه ای از نورون ها است محاسبات پیشخور و رو به جلو در شبکه به صورت زیر انجام می شود [۱۹] :

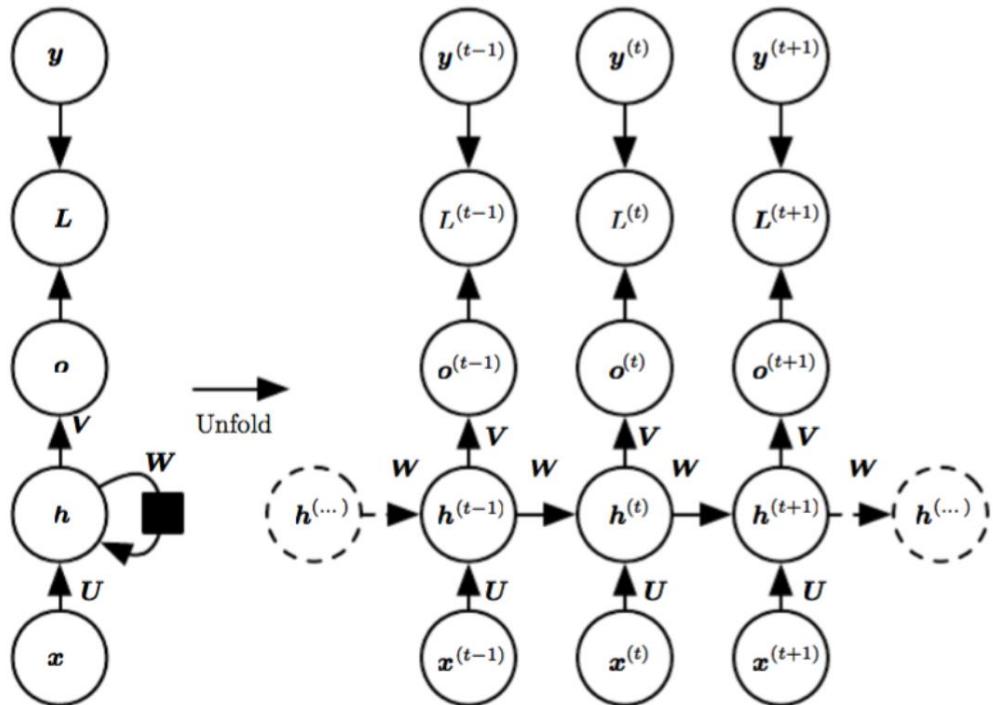


شکل ۴-۵ : گراف مدل شده شبکه عصبی بازگشتی یک لایه [۱۹]

$$h_i(t) = f_1 \left(\sum_j w_{ji}^{xh} X_j(t) + \sum_j w_{ji}^{hh} h_i(t-1) + b_i^h \right) \quad (4-11-3)$$

$$Y(t) = f_2 \left(\sum_j w_{jk}^{hy} h_j(t) + b_k^y \right) . k = 1 \dots M \quad (4-11-4)$$

همانطور که مشاهده می شود برای محاسبه خروجی هر لایه مخفی از دو ماتریس وزن W_{ji}^{hh} و W_{ji}^{xh} استفاده می شود و همینطور برای محاسبه خروجی شبکه نیاز به ماتریس W_{jk}^{hy} و مقادیر بایاس در شبکه داریم که تمام این وزن ها توسط الگوریتم یادگیری آموخته می شوند .



شکل ۵۰-۴ : گراف مدل شده شبکه های عصبی بازگشتی [۱۸]

به عنوان نمونه در شبکه بالا خروجی به صورت زیر بدست می آید [۱۹]:

$$a^{(t)} = Wh^{(t-1)} + Ux^{(t)} + b \quad (۴-۱۱-۵)$$

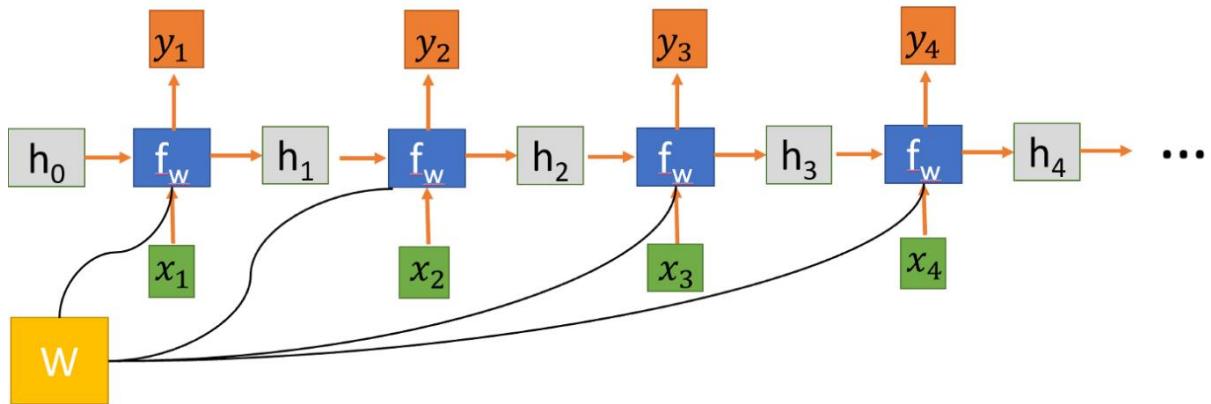
$$h^{(t)} = \tanh(a^{(t)}) \quad (۴-۱۱-۶)$$

$$o^{(t)} = c + Vh^{(t)} \quad (۴-۱۱-۷)$$

$$\hat{y}^{(t)} = \text{softmax}(o^{(t)}) \quad (۴-۱۱-۸)$$

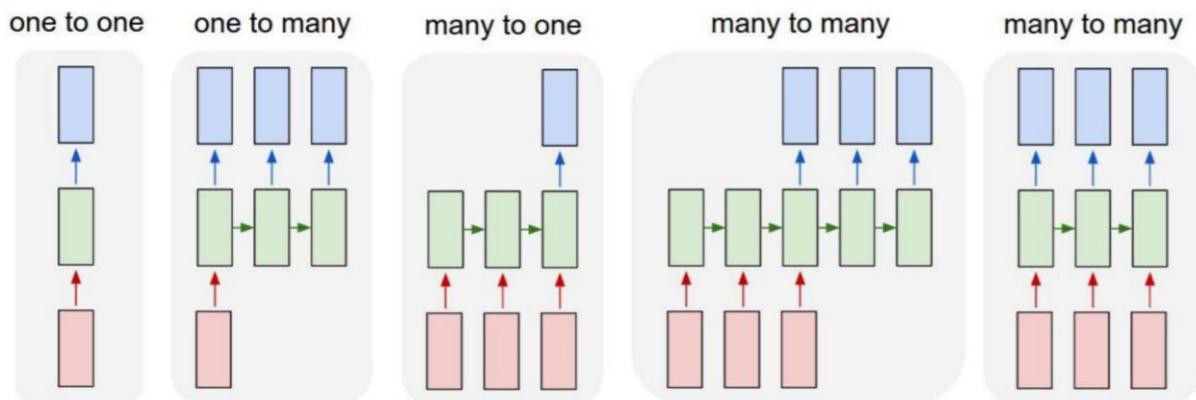
و سپس در الگوریتم یادگیری تلاش می شود پارامتر های مدار که شامل ماتریس های W ، U ، V و b می شود به گونه انتخاب شود که خطای کل شبکه حداقل شود .

همانطور که مشاهده می شود ماتریس های W ، U ، V در شبکه مشترک هستند و این اشتراک پارامتر های شبکه و عدم نیاز به یادگیری پارامتر های جدید در طول شبکه باعث افزایش تعمیم پذیری شبکه و افزایش سرعت آموخته شبکه به علت پارامتر های کمتر می شود .



شکل ۵۱-۴ : گراف مدل شده شبکه های عصبی بازگشتی به همراه وزن های اشتراکی هر واحد [۳۹]

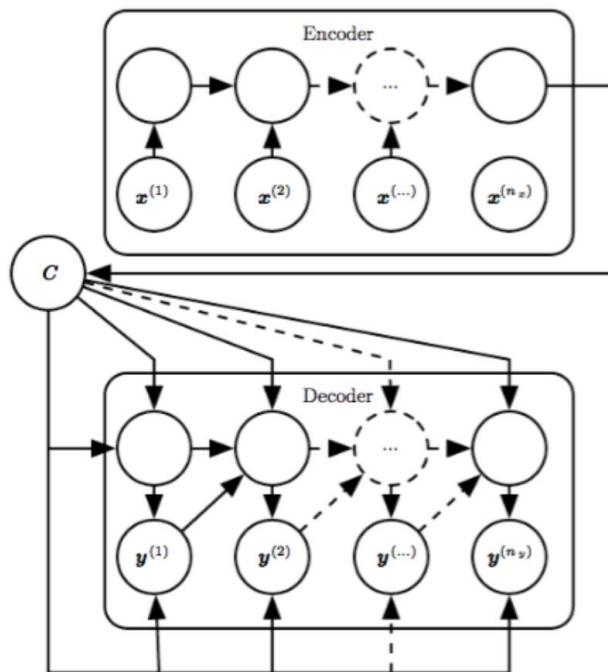
۱-۱۱-۴ انواع الگوهای پردازش توالی :



شکل ۵۲-۴ : گراف انواع الگوهای پردازش توالی [۴۰]

از مهم ترین آن ها می توان به الگوهای یک به چند مانند عنوان گذاری تصاویر و الگوهای چند به یک مانند تشخیص اخبار جعلی و تشخیص احساسات در جملات و الگوهای چند به چند همراه تاخیر مانند ترجمه ماشینی که توالی ورودی به طور کامل به مدل داده می شود و سپس خروجی تولید می شود و الگوهای چند بدون تاخیر مانند دسته بندی ویدئو در سطح فریم که به ازای هر ورودی ، خروجی ها بدون تاخیر تولید می شوند [۴۰].

۴-۱۱-۲ معماری کدگذار و گدگشا در شبکه های عصبی بازگشتی :

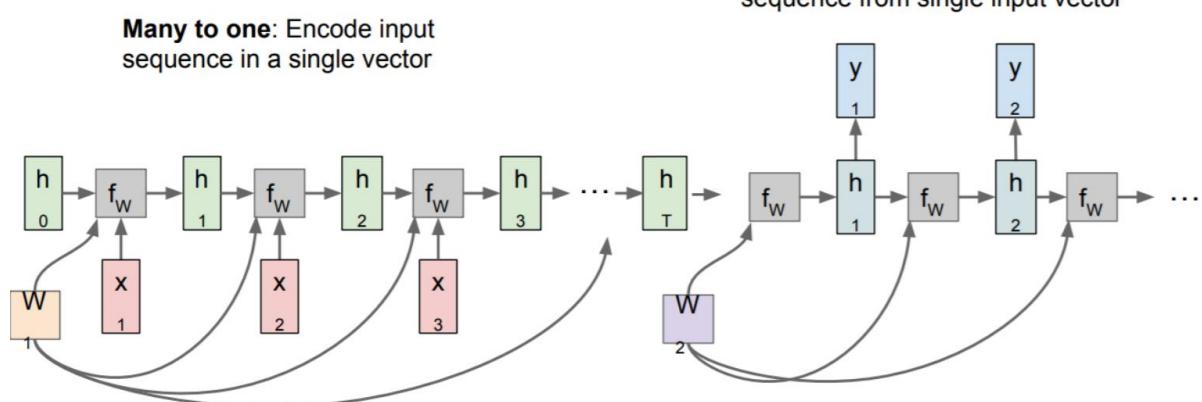


شکل ۴-۴ : گراف معماری کدگذار و گدگشا [۱۸]

در قسمت کدگذار، توالی به عنوان ورودی دریافت می شود و پس از دریافت توسط لایه مخفی کد می شود و کد یا محتوا C را به وجود می آورد و سپس در قسمت کدگشایی با دریافت محتوا C شبکه با استفاده از لایه مخفی خروجی را تولید می کند [۱۸].

الگوهای توالی چند به چند با تاخیر را می توان ترکیبی از یک کدگذار و یک کدگشا دانست .

One to many: Produce output sequence from single input vector



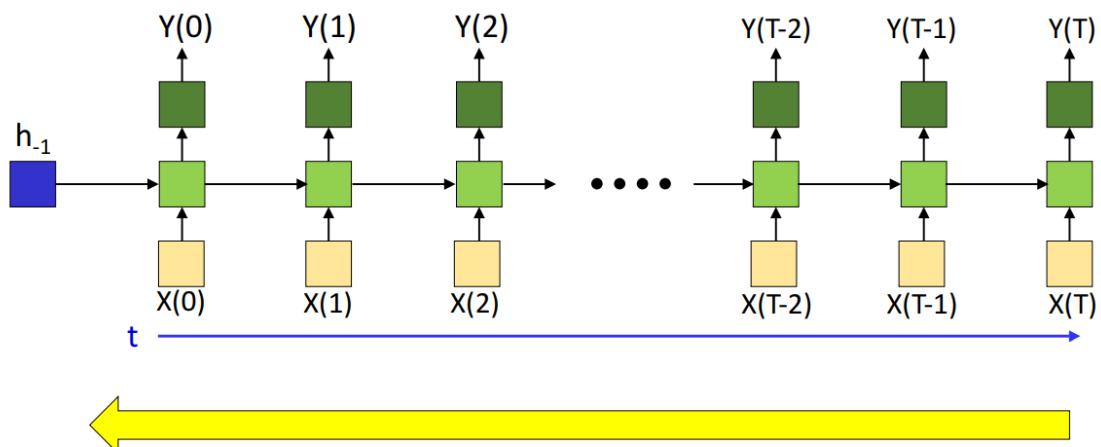
شکل ۴-۴ : گراف مدل شده کدگذار و گدگشا در شبکه های عصبی بازگشتی [۱۹]

¹ Encoder

² Decoder

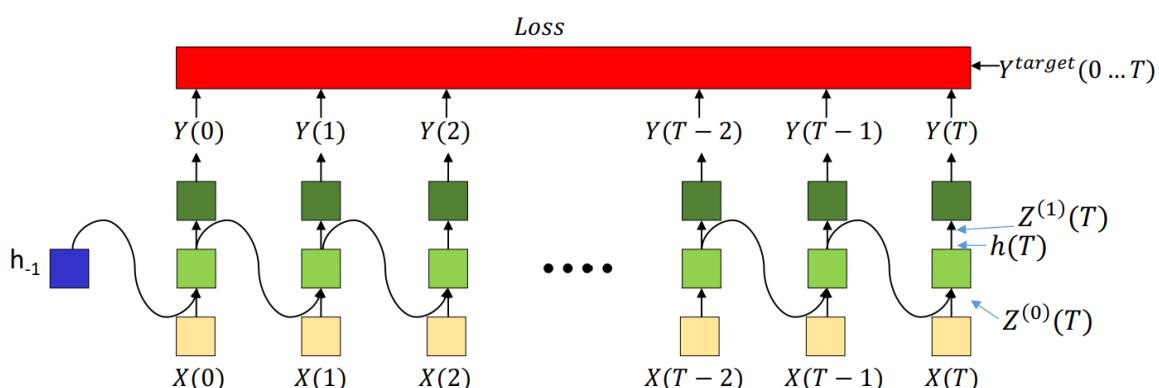
۱۱-۳ نحوه آموزش شبکه های بازگشتی :

همانند الگوریتم پس انتشار خطا که در شبکه های عصبی چند لایه پرسپترون مورد استفاده قرار گرفت در شبکه های بازگشتی نیز با انتشار خطا رو به عقب می توان خطای کل شبکه را محاسبه کرد با این تفاوت که الگوریتم پس انتشار خطا را در حوزه زمان اجرا می شود و در حوزه زمان نیز رو به عقب حرکت می شود [۱۹].



شکل ۵۵-۴ : مسیر پس انتشار خطا در حوزه زمان در شبکه عصبی بازگشتی [۱۹]

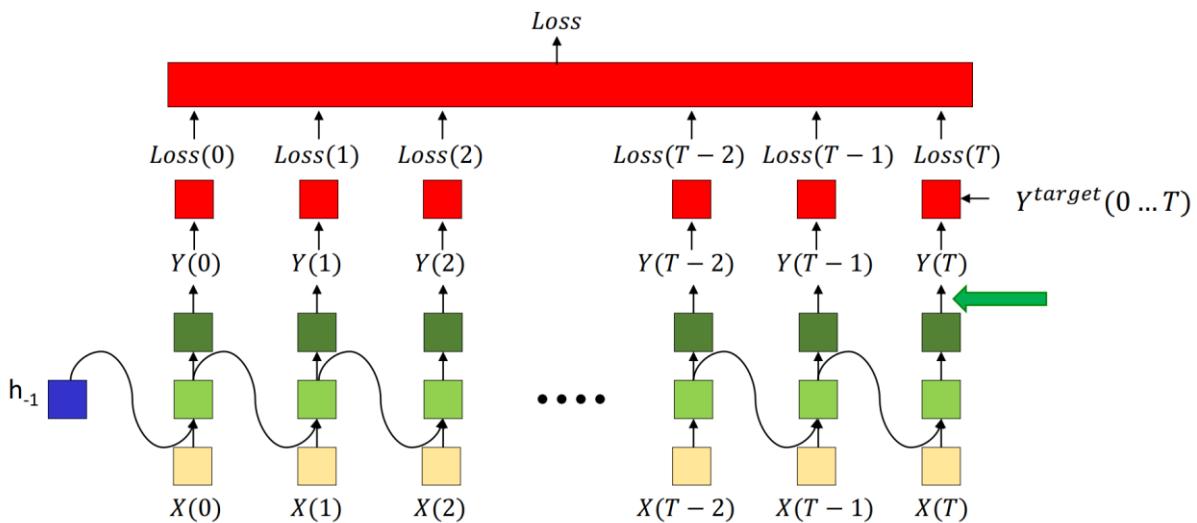
به عنوان نمونه برای یک شبکه عصبی بازگشتی با یک لایه مخفی می توان الگوریتم پس انتشار خطا را با محاسبات زیر انجام داد .



شکل ۵۶-۴ : گراف محاسباتی شبکه های عصبی بازگشتی [۱۹]

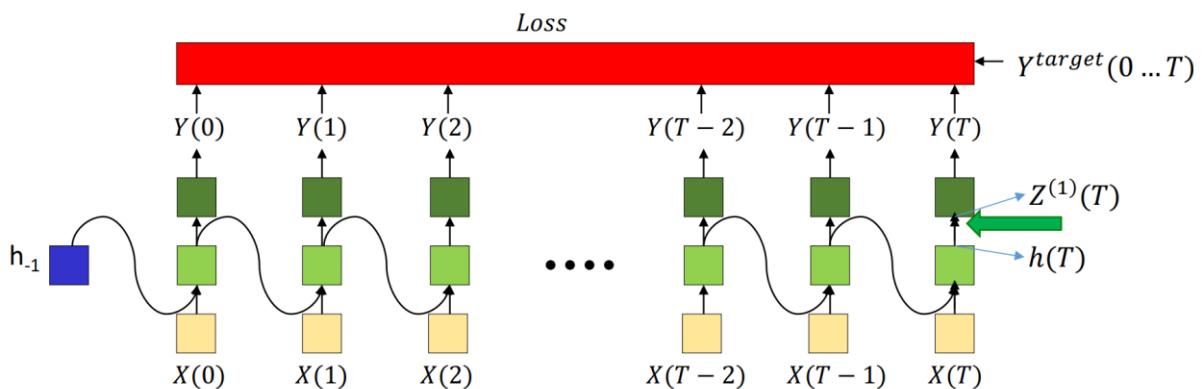
$h(t)$ خروجی در لحظه t و $Z^{(1)}(t)$ مقدار نورون های لایه خروجی قبل از اعمال تابع فعالسازی در لحظه t و $Y(t)$ خروجی لایه مخفی در لحظه t و $Z^{(0)}(t)$ مقدار نورون های لایه مخفی قبل از اعمال تابع فعالسازی در لحظه t است . [۱۹]

محاسبات الگوریتم پس انتشار خطا در حوزه زمان :



شکل ۵۷-۴ : گراف محاسباتی پس انتشار خطا در شبکه های عصبی بازگشتی [۱۹]

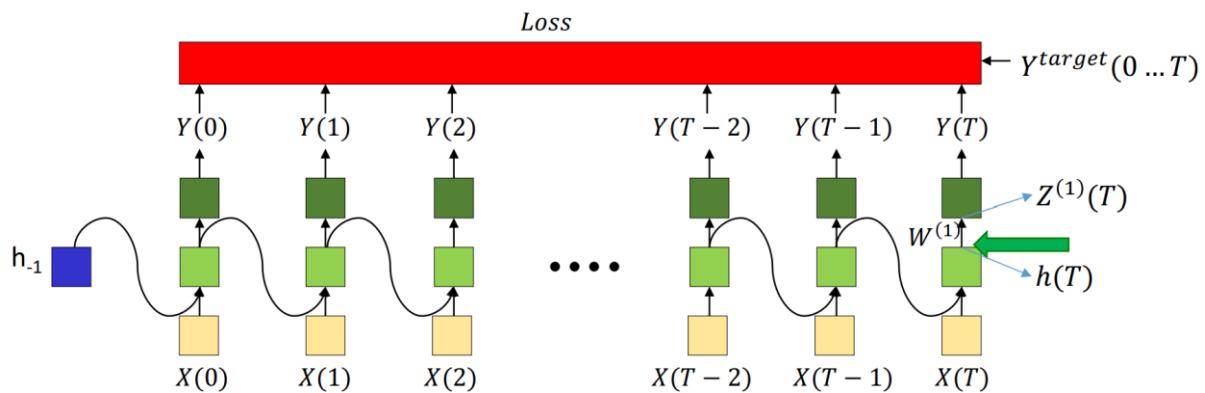
ابتدا برای همه زمان ها (t) و نورون های خروجی (i) محاسبه می شود و مشتق خطا تا ابتدای شبکه به صورت عقبگرد پیش می رود [۱۹].



شکل ۵۸-۴ : گراف محاسباتی پس انتشار خطا در شبکه های عصبی بازگشتی [۱۹]

$$\nabla_{Z^{(1)}(T)} Loss = \nabla_{Z^{(1)}(T)} Y(T) \nabla_{Y(T)} Loss \quad (4-11-9)$$

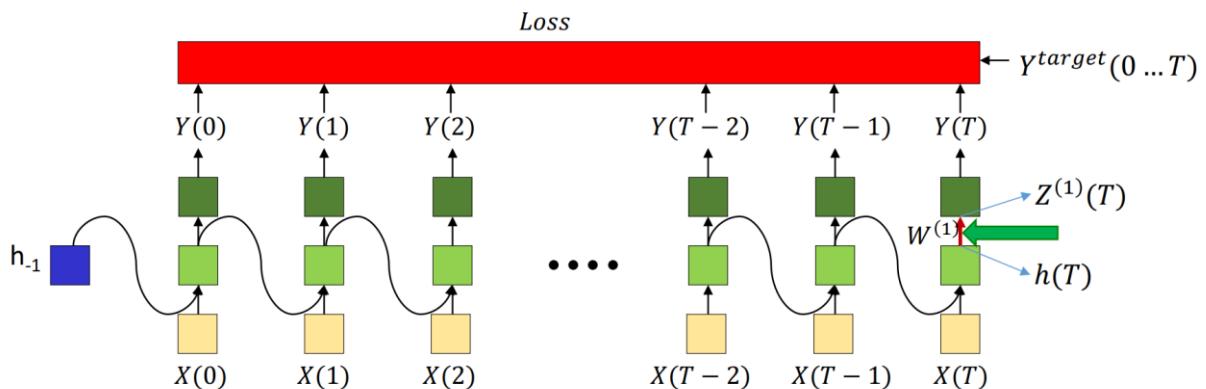
$$\frac{dLoss}{dZ_i^{(1)}(T)} = \sum_j \frac{dLoss}{dY_j(T)} \frac{dY_j(T)}{dZ_i^{(1)}(T)} \quad (4-11-10)$$



شکل ۵۹-۴ : گراف محاسباتی پس انتشار خطا در شبکه های عصبی بازگشتی [۱۹]

$$\nabla_{h(T)} Loss = W^{(1)} \nabla_{Z^{(1)}(T)} Loss \quad (\text{۴-۱۱-۱۱})$$

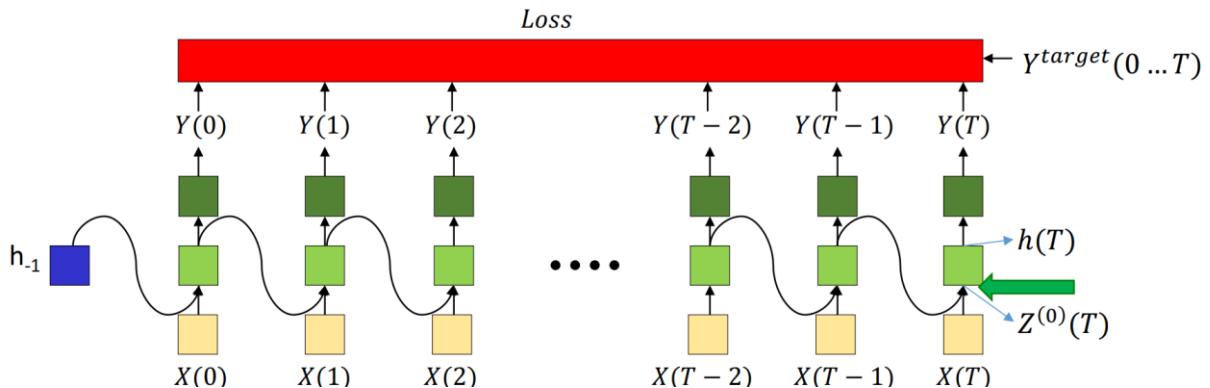
$$\begin{aligned} \frac{dLoss}{dh_i(T)} &= \sum_j \frac{dLoss}{dZ_j^{(1)}(T)} \frac{dZ_j^{(1)}(T)}{dh_i(T)} \\ &= \sum_j w_{ij}^{(1)} \frac{dLoss}{dZ_j^{(1)}(T)} \end{aligned} \quad (\text{۴-۱۱-۱۲})$$



شکل ۶۰-۴ : گراف محاسباتی پس انتشار خطا در شبکه های عصبی بازگشتی [۱۹]

$$\nabla_{W^{(1)}} Loss = h(T) \nabla_{Z^{(1)}(T)} Loss \quad (\text{۴-۱۱-۱۳})$$

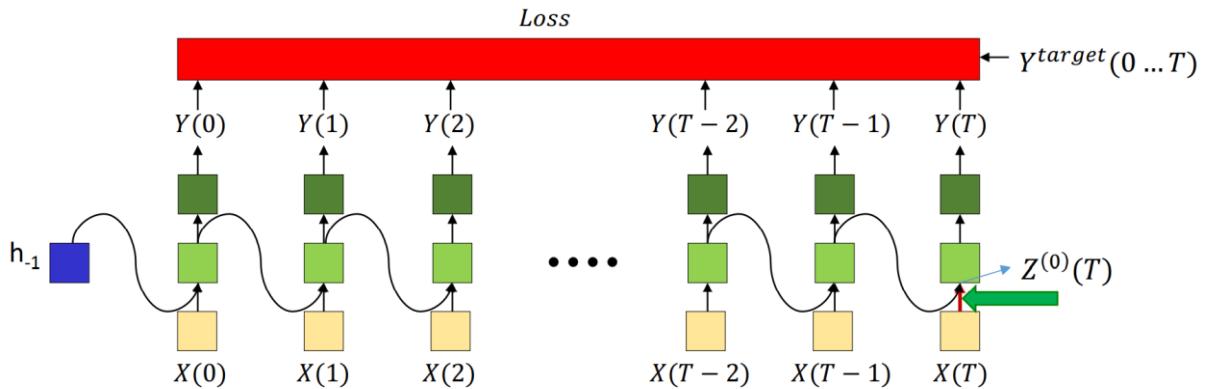
$$\frac{dLoss}{dw_{ij}^{(1)}} = \frac{dLoss}{dZ_j^{(1)}(T)} h_i(T) \quad (\text{۴-۱۱-۱۴})$$



شکل ۶۱-۴ : گراف محاسباتی پس انتشار خطا در شبکه های عصبی بازگشتی [۱۹]

$$\nabla_{Z^{(0)}(T)} Loss = \nabla_{Z^{(0)}(T)} h(T) \nabla_{h(T)} Loss \quad (\text{F-11-15})$$

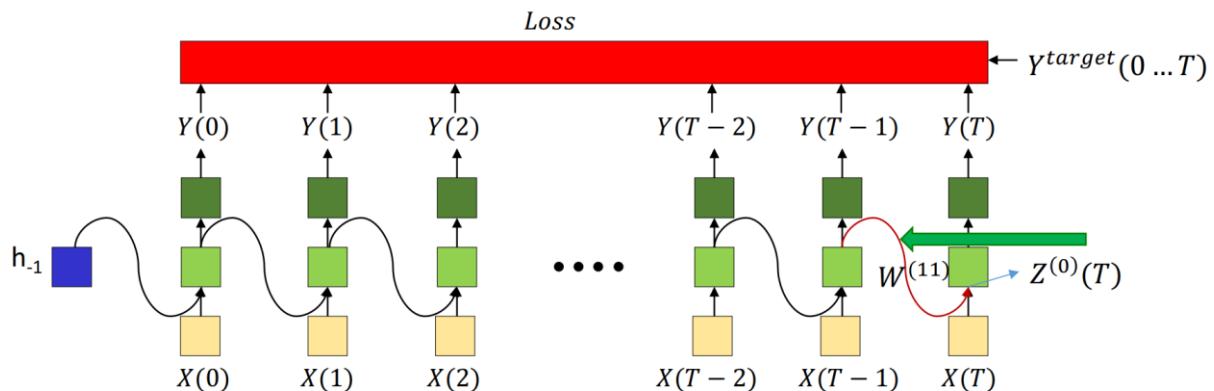
$$\frac{dLoss}{dZ_i^{(0)}(T)} = \frac{dLoss}{dh_i(T)} \frac{dh_i(T)}{dZ_i^{(0)}(T)} \quad (\text{F-11-16})$$



شکل ۶۲-۴ : گراف محاسباتی پس انتشار خطا در شبکه های عصبی بازگشتی [۱۹]

$$\nabla_{W^{(0)}} Loss = X(T) \nabla_{Z^{(0)}(T)} Loss \quad (\text{F-11-17})$$

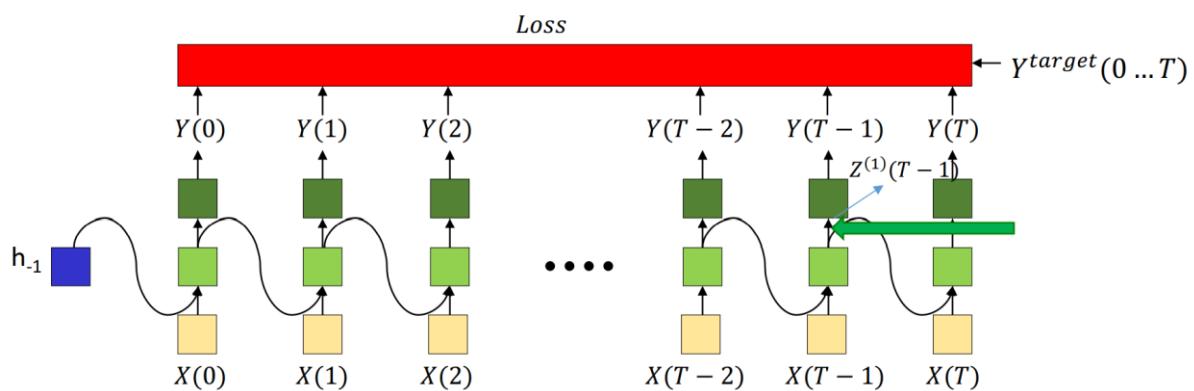
$$\frac{dLoss}{dw_{ij}^{(0)}} = \frac{dLoss}{dZ_j^{(0)}(T)} X_i(T) \quad (\text{F-11-18})$$



شکل ۶۳-۴ : گراف محاسباتی پس انتشار خطا در شبکه های عصبی بازگشتی [۱۹]

$$\nabla_{W^{(11)}} Loss = h(T-1) \nabla_{Z_j^{(0)}(T)} Loss \quad (۴-۱۱-۱۹)$$

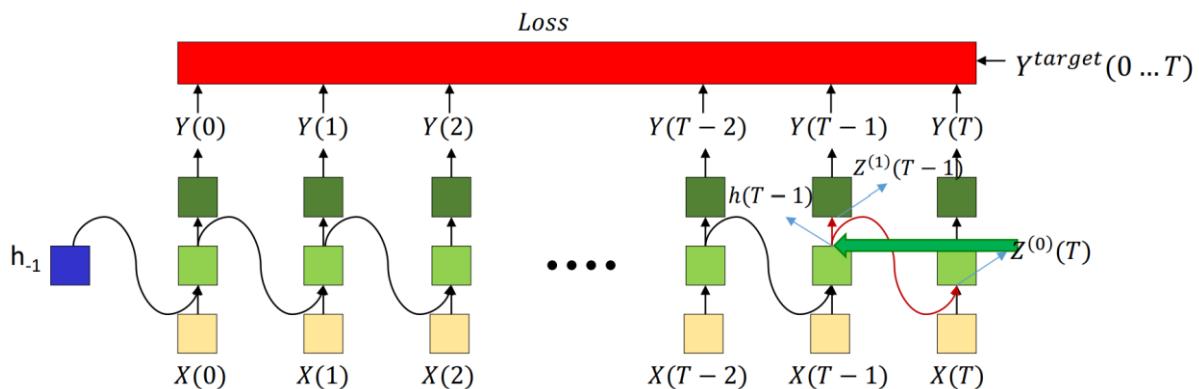
$$\frac{dLoss}{dW_{ij}^{(11)}} = \frac{dLoss}{dZ_j^{(0)}(T)} h_i(T-1) \quad (۴-۱۱-۲۰)$$



شکل ۶۴-۴ : گراف محاسباتی پس انتشار خطا در شبکه های عصبی بازگشتی [۱۹]

$$\nabla_{Z^{(1)}(T-1)} Loss = \nabla_{Z^{(1)}(T)} Y(T-1) \nabla_{Y(T-1)} Loss \quad (۴-۱۱-۲۱)$$

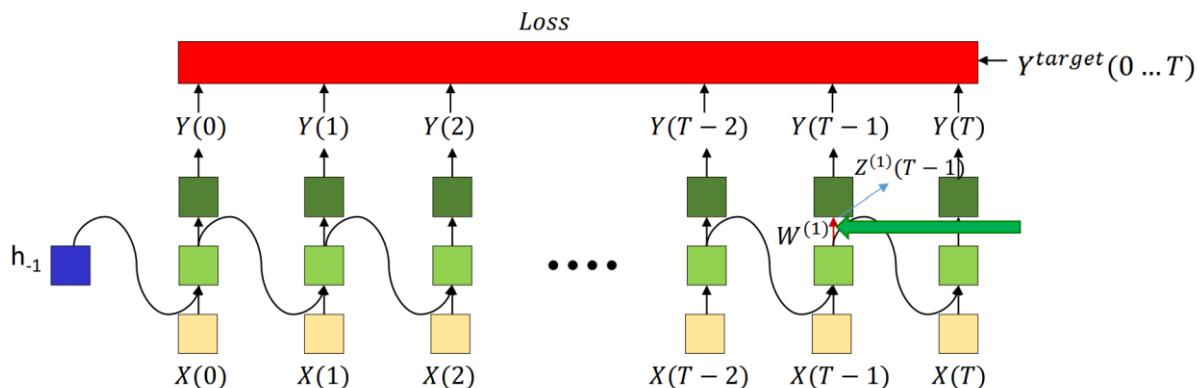
$$\frac{dLoss}{dZ_i^{(1)}(T-1)} = \sum_j \frac{dLoss}{dY_j(T-1)} \frac{dY_j(T-1)}{dZ_i^{(1)}(T-1)} \quad (۴-۱۱-۲۲)$$



شکل ۶۵-۴ : گراف محاسباتی پس انتشار خطا در شبکه های عصبی بازگشتی [۱۹]

$$\begin{aligned} \nabla_{h(T-1)} Loss &= W^{(1)} \nabla_{Z^{(1)}(T-1)} Loss \\ &+ W^{(11)} \nabla_{Z^{(0)}(T)} Loss \end{aligned} \quad (\text{۴-۱۱-۲۳})$$

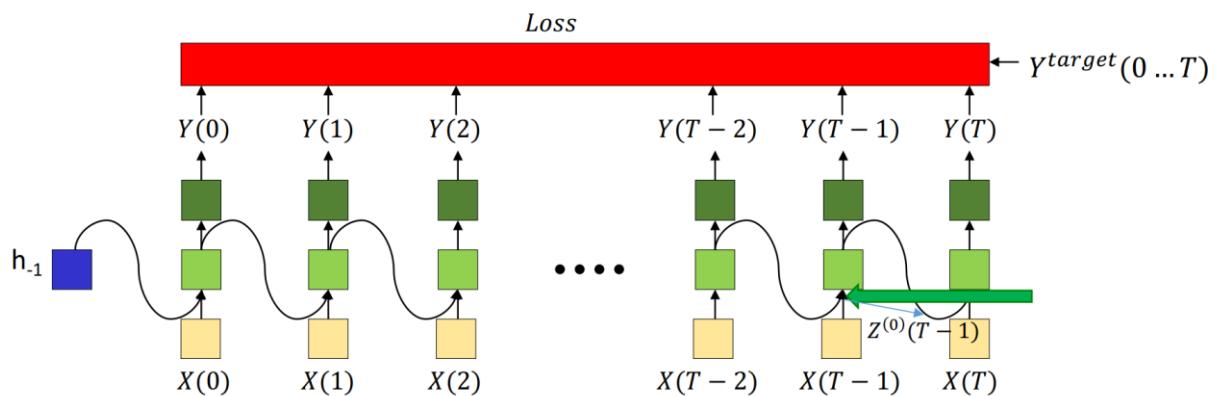
$$\begin{aligned} \frac{dLoss}{dh_i(T-1)} &= \sum_j w_{ij}^{(1)} \frac{dLoss}{dZ_j^{(1)}(T-1)} \\ &+ \sum_j w_{ij}^{(11)} \frac{dLoss}{dZ_j^{(0)}(T)} \end{aligned} \quad (\text{۴-۱۱-۲۴})$$



شکل ۶۶-۴ : گراف محاسباتی پس انتشار خطا در شبکه های عصبی بازگشتی [۱۹]

$$\nabla_{W^{(1)}} Loss + h(T-1) \nabla_{Z^{(1)}(T-1)} Loss \quad (\text{۴-۱۱-۲۵})$$

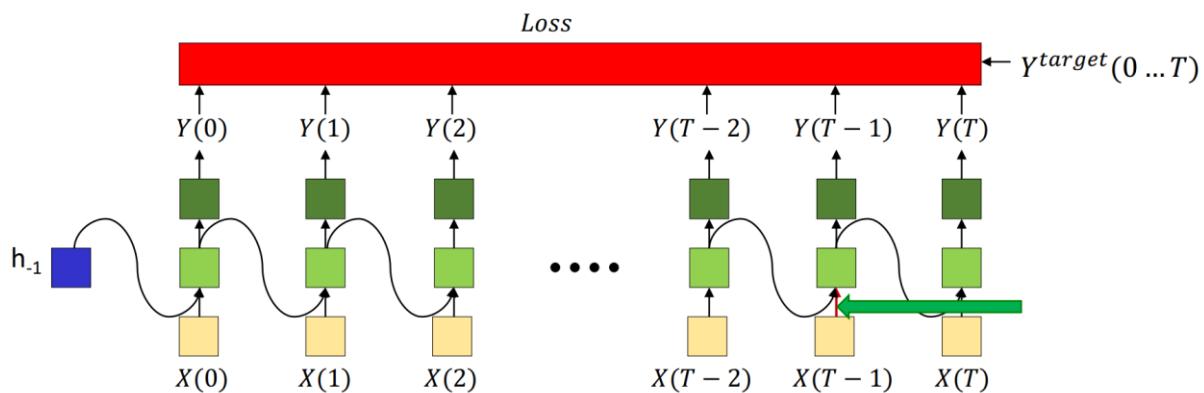
$$\frac{dLoss}{dw_{ij}^{(1)}} += \frac{dLoss}{dZ_j^{(1)}(T-1)} h_i(T-1) \quad (\text{۴-۱۱-۲۶})$$



[۱۹] شکل ۶۷-۴ : گراف محاسباتی پس انتشار خطا در شبکه های عصبی بازگشتی

$$\nabla_{Z^{(0)}(T-1)} Loss = \nabla_{Z^{(0)}(T-1)} h(T-1) \nabla_{h(T-1)} Loss \quad (\text{۴-۱۱-۲۷})$$

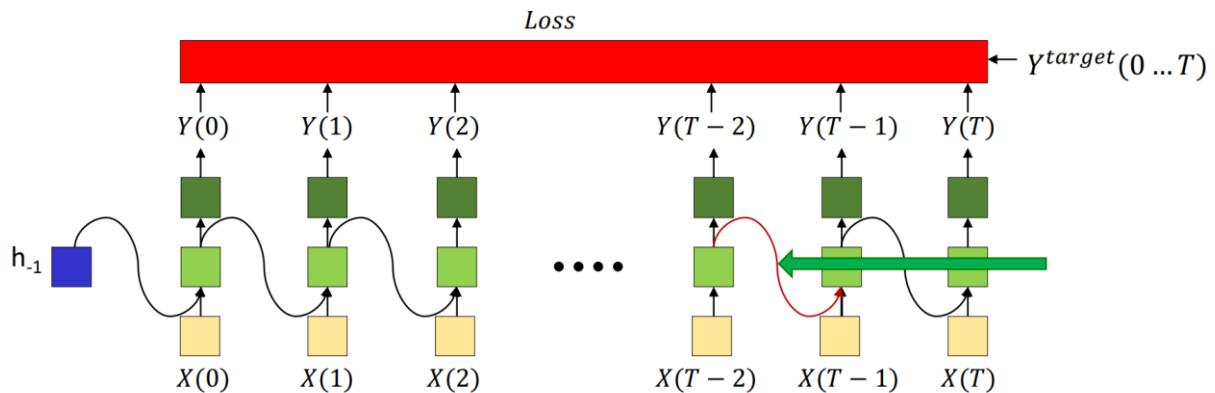
$$\frac{dLoss}{dZ_i^{(0)}(T-1)} = \frac{dLoss}{dh_i(T-1)} \frac{dh_i(T-1)}{dZ_i^{(0)}(T-1)} \quad (\text{۴-۱۱-۲۸})$$



[۱۹] شکل ۶۸-۴ : گراف محاسباتی پس انتشار خطا در شبکه های عصبی بازگشتی

$$\nabla_{W^{(0)}} Loss + X(T-1) \nabla_{Z^{(0)}(T-1)} Loss \quad (\text{۴-۱۱-۲۹})$$

$$\frac{dLoss}{dw_{ij}^{(0)}} + \frac{dLoss}{dZ_j^{(0)}(T-1)} X_i(T-1) \quad (\text{۴-۱۱-۳۰})$$

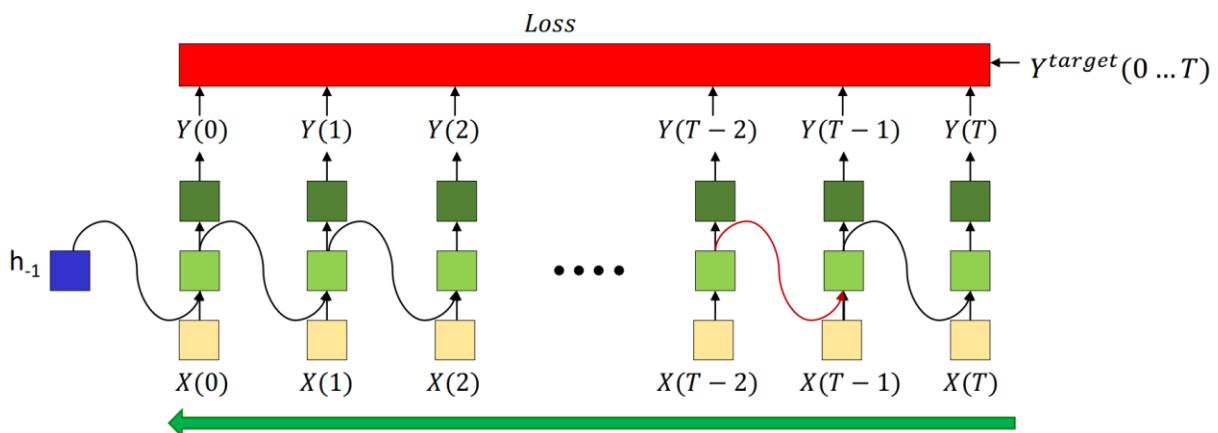


شکل ۶۹-۴ : گراف محاسباتی پس انتشار خطا در شبکه های عصبی بازگشتی [۱۹]

$$\nabla_{W^{(11)}} Loss += h(T-2) \nabla_{Z_j^{(0)}(T-1)} Loss \quad (4-11-31)$$

$$\frac{dLoss}{dW_{ij}^{(11)}} += \frac{dLoss}{dZ_j^{(0)}(T-1)} h_i(T-2) \quad (4-11-32)$$

پس انتشار خطا در حوزه زمان به صورت عقبگرد تا لحظه ابتدایی ادامه می یابد و براساس گرادیان خطا محاسبه شده وزن های جدید بروزرسانی می شوند .

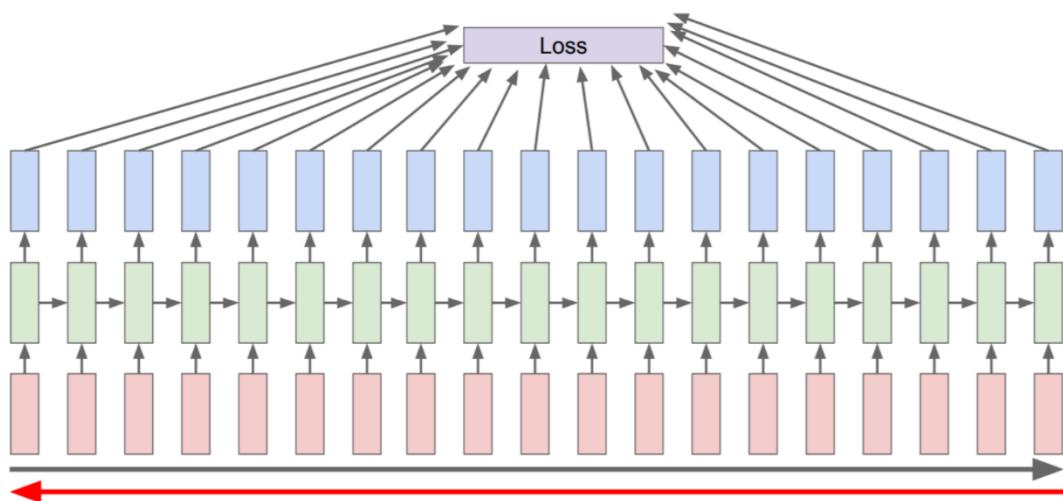


شکل ۷۰-۴ : گراف محاسباتی پس انتشار خطا در شبکه های عصبی بازگشتی [۱۹]

$$\nabla_{h_{-1}} Loss = W^{(11)} \nabla_{Z_j^{(1)}(0)} Loss \quad (4-11-33)$$

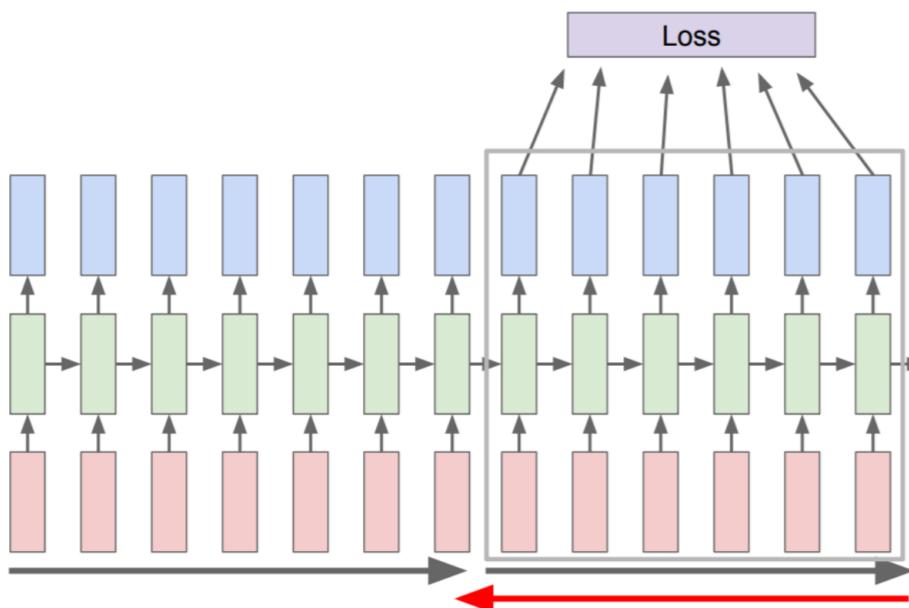
$$\frac{dLoss}{dh_{-1}} = \sum_j w_{ij}^{(11)} \frac{dLoss}{dZ_j^{(1)}(0)} \quad (4-11-34)$$

روش کوتاه شده پس انتشار خطا در حوزه زمان^۱:



شکل ۷۱-۴ : گراف محاسباتی پس انتشار خطا در شبکه های عصبی بازگشتی [۱۹]

در این روش به جهت کاهش بار محاسباتی و افزایش سرعت آموزش ، شبکه به قطعات مساوی بر حسب طول توالی تقسیم می شود و بعد از حرکت رو به جلو در شبکه عصبی بازگشتی و محاسبه مقادیر لایه مخفی ، خروجی ها و مقادیر خطا برای هر خروجی ، الگوریتم پس انتشار خطا در حوزه زمان برای قطعات شبکه به صورت جداگانه اجرا می شود و با توجه به این که در توالی های ورودی با طول های زیاد به روزرسانی پارامتر های شبکه کند و از لحاظ محاسباتی دشوار خواهد بود در نتیجه این روش عملیات پس انتشار خطا را در طول های مشخصی انجام میدهد و با ارائه تقریبی از گرادیان های موجود در شبکه سرعت یادگیری و حجم محاسبات را کاهش می دهد[۴۱] .

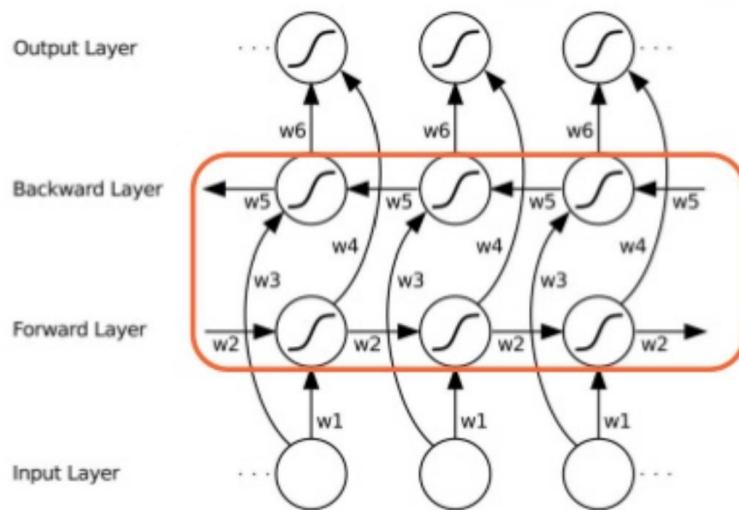


شکل ۷۲-۴ : گراف محاسباتی پس انتشار خطا به روش کوتاه شده [۱۹]

^۱ Truncated backpropagation through time

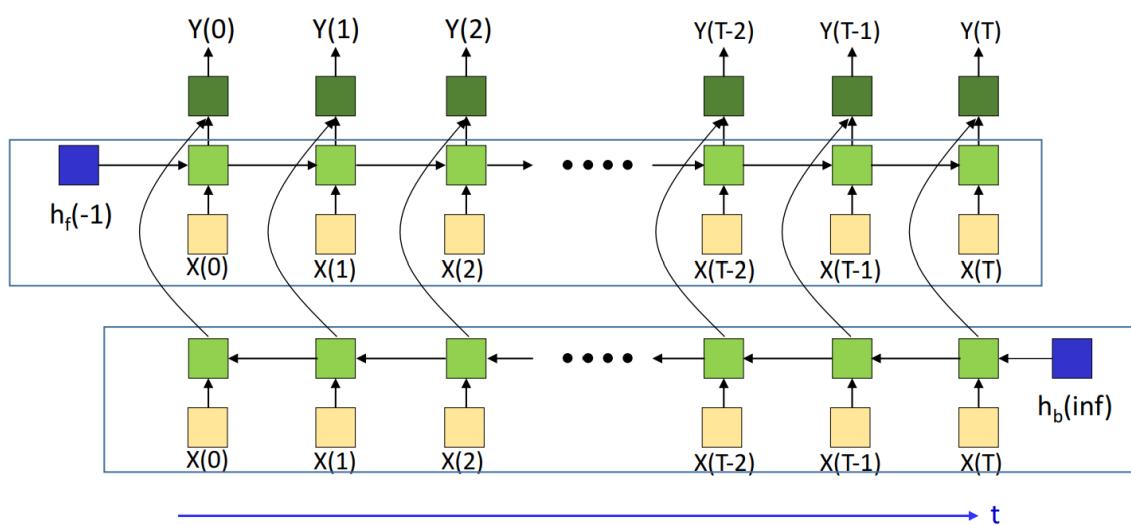
۴-۱۱-۴ شبکه های عصبی بازگشتی دو طرفه^۱:

در بسیاری از مسائل شبکه های عصبی بازگشتی توالی ورودی را به طور کامل دریافت می کنند و سپس خروجی را پیش بینی می کنند بنابراین کل توالی ورودی در دسترس است و می توان از آن برای آموزش شبکه استفاده کرد.



شکل ۴-۴: گراف محاسباتی شبکه های عصبی بازگشتی دو طرفه [۱۹]

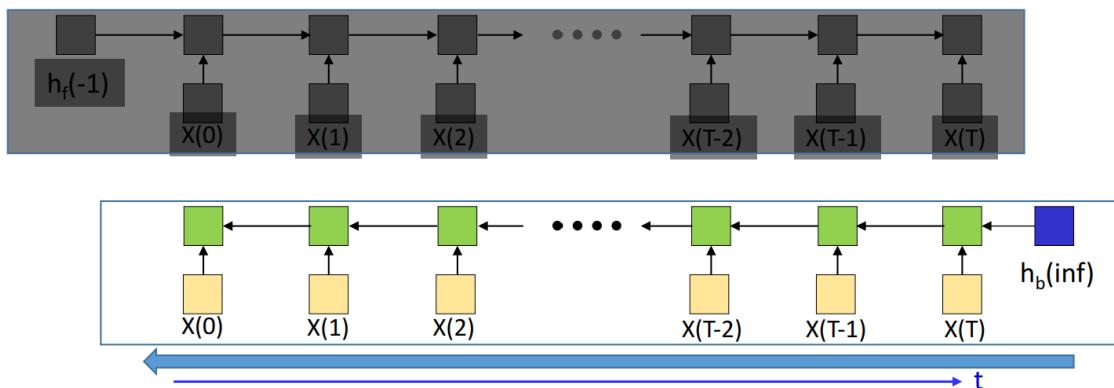
شبکه های عصبی بازگشتی دو طرفه از ترکیب دو شبکه عصبی بازگشتی که یکی از آن ها در به صورت رو به جلو و پیشخور در زمان حرکت می کند و دیگری با شروع از انتهای توالی ورودی شبکه و به صورت عقبگرد در زمان حرکت می کند ، بنابراین دو شبکه در دو جهت متفاوت توالی های ورودی را آموخته اند و علاوه بر گذشته توالی ورودی ، آینده آن نیز در نظر گرفته می شود و با ترکیب خروجی های این دو شبکه عصبی بازگشتی و پیش بینی خروجی کل شبکه از کل اطلاعات توالی های ورودی استفاده می شود [۴۲] .



شکل ۴-۴: گراف محاسباتی شبکه های عصبی بازگشتی دو طرفه [۱۹]

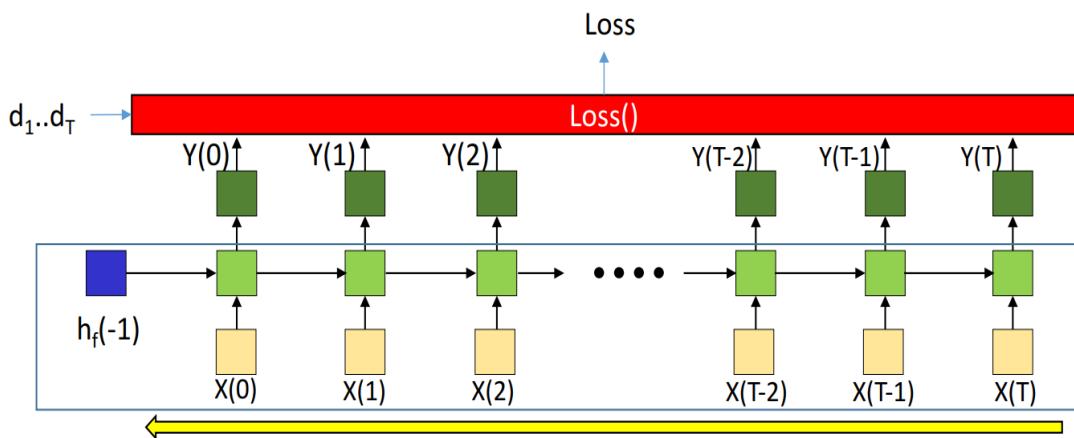
¹ Bidirectional recurrent neural networks

در مرحله آموزش نیز هر کدام از شبکه های عصبی بازگشتی یک طرفه با پارامتر های مستقل و به صورت جداگانه آموزش داده می شوند و در نهایت با ترکیب خروجی های این دو شبکه آموزش دیده ، خروجی کل شبکه بددست می آید ، همچنین پارامتر های حالت های اولیه دو شبکه بازگشتی یک طرفه نیز شامل $(-1) h_f$ و $h_f(inf)$ قابل تنظیم و آموزش هستند [۱۹].

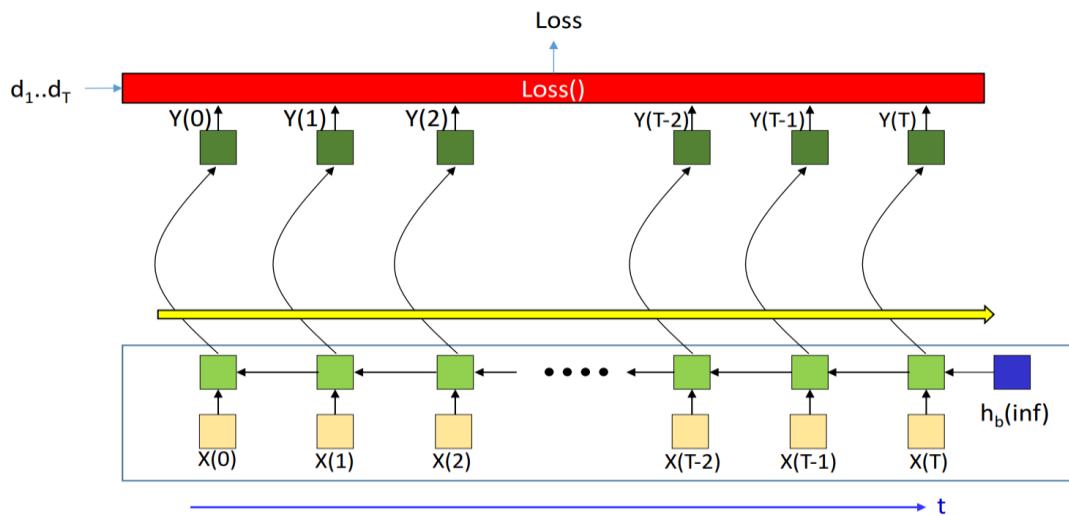


شکل ۷۵-۴ : گراف محاسباتی لایه معکوس در شبکه های عصبی بازگشتی دو طرفه [۱۹]

با توجه به اینکه جهت رو به جلو در دو شبکه عصبی بازگشتی یک طرفه متفاوت و عکس یکدیگر تعریف می شود . جهت الگوریتم پس انتشار خطأ در حوزه زمان نیز برای این دو شبکه یک طرفه متفاوت و عکس یکدیگر خواهد بود .



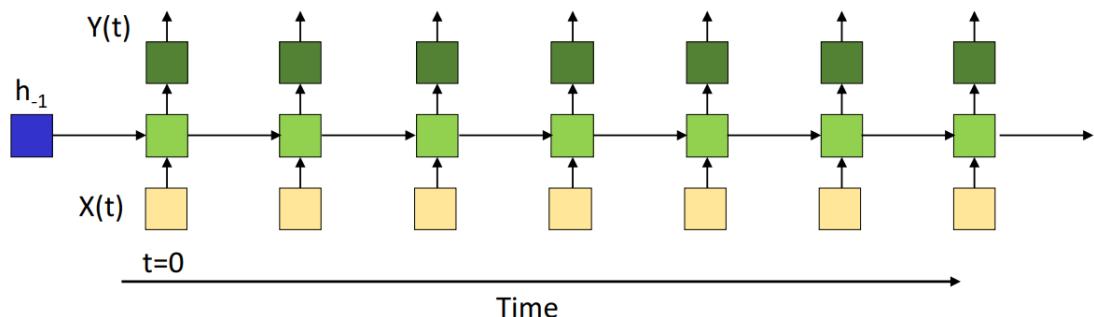
شکل ۷۶-۴ : گراف محاسباتی مسیر پس انتشار خطأ در شبکه های عصبی بازگشتی دو طرفه [۱۹]



شکل ۷۷-۴ : گراف محاسباتی مسیر پس انتشار خطا لایه معکوس در شبکه های عصبی بازگشتی دو طرفه [۱۹]

۱۱-۵ پایداری شبکه های عصبی بازگشتی :

اگر پایداری را از نوع^۱ BIBO به معنای خروجی محدود در ازای ورودی محدود تعریف شود میتوان با فرض خطی بودن سیستم و در نظر گرفتنتابع همانی به عنوان تابع فعالسازی موجود در شبکه راحت تر این نوع پایداری را در شبکه های عصبی بازگشتی بررسی کرد [۱۹].



شکل ۷۸-۴ : گراف محاسباتی شبکه های عصبی بازگشتی [۱۹]

$$z_k = W_h h_{k-1} + W_x x_k \quad h_k = z_k \quad (۱۱-۳۵)$$

$$h_k = W_h h_{k-1} + W_x x_k \quad (۱۱-۳۶)$$

$$h_{k-1} = W_h h_{k-2} + W_x x_{k-1} \quad (۱۱-۳۷)$$

^۱ Bounded Input Bonded Output

$$h_k = W_h^2 h_{k-2} + W_h W_x x_{k-1} + W_x x_k \quad (4-11-38)$$

$$h_k = W_h^{k+1} h_{-1} + W_h^k W_x x_0 + W_h^{k-1} W_x x_1 + \dots \quad (4-11-39)$$

همانطور که ملاحظه می شود با پیشروی در لایه مخفی وزن های شبکه بسته به مقدار آن ها می توانند مقادیر بسیار بزرگ و یا بسیار کوچک را بگیرند.

و با توجه به خطی بودن سیستم و استفاده از قضیه جمع آثار می توان با اعمال یک ورودی پاسخ سیستم را بدست آورد.

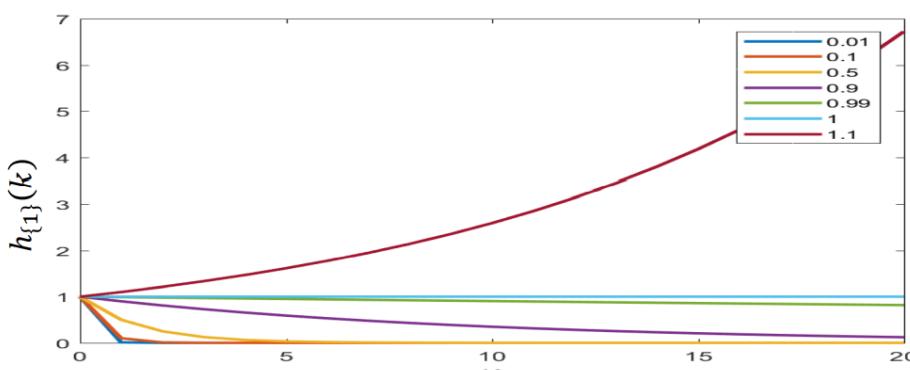
$$\begin{aligned} h_k &= H_k(h_{-1}) + H_k(x_0) + H_k(x_1) + H_k(x_2) + \dots \\ &= h_{-1}(1_{-1}) + x_0 H_k(1_0) + x_1 H_k(1_1) + x_2 H_k(1_2) \\ &\quad + \dots \end{aligned} \quad (4-11-40)$$

که در آن $H_k(1_t)$ پاسخ لایه مخفی در زمان k است زمانی که ورودی به صورت یک بردار که تنها درایه t ام آن مقدار یک را دارد و سایر درایه های آن صفر است.

بنابراین با قرار دادن مقادیر عددی به عنوان وزن و بدست آوردن پاسخ سیستم به تک ورودی در لحظه یک خواهیم داشت [۱۹] :

$$h(t) = w_h h(t-1) + w_x x(t) \quad (4-11-41)$$

$$h_{\{1\}}(t) = w_h^t w_x x(1) \quad (4-11-42)$$



شکل ۷۹-۴ : نمودار پاسخ سیستم به ازای وزن های متفاوت [۱۹]

همانطور که ملاحظه می شود به ازای w_h های بزرگ تر از یک پاسخ مقادیر بزرگ و به ازای w_h های کوچکتر از یک پاسخ به سرعت مقادیر صفر را به خود می گیرد و می توان حساسیت بالای شبکه به مقادیر وزن ها در حرکت پیشخور و رو به جلو را مشاهده کرد که به راحتی اطلاعات در شبکه بوسیله ی پدیده محوشوندگی و یا انفجار از بین می رود .

در حالت کلی و با قرار دادن یک بردار به جای مقادیر عددی برای وزن های شبکه می توان نوشت [۱۹] :

$$h(t) = W_h h(t-1) + W_x x(t) \quad (۴-۱۱-۴۳)$$

$$h_{\{1\}}(t) = W_h^t W_x x(1) \quad (۴-۱۱-۴۴)$$

و همچنین با تجزیه ماتریس وزن های لایه مخفی به مقادیر و بردار های ویژه می توان نوشت :

$$W_h = U \Lambda U^{-1} \quad (۴-۱۱-۴۵)$$

$$W_h u_i = \lambda_i u_i \quad (۴-۱۱-۴۶)$$

با توجه به اینکه می توان هر بردار v را بر حسب بردار های ویژه W_h نوشت داریم [۱۹] :

$$v = a_1 u_1 + a_2 u_2 + a_3 u_3 + \dots + a_n u_n \quad (۴-۱۱-۴۷)$$

$$W_h v = a_1 \lambda_1 u_1 + a_2 \lambda_2 u_2 + \dots + a_n \lambda_n u_n \quad (۴-۱۱-۴۸)$$

$$W_h^t v = a_1 \lambda_1^t u_1 + a_2 \lambda_2^t u_2 + \dots + a_n \lambda_n^t u_n \quad (۴-۱۱-۴۹)$$

$$\lim_{t \rightarrow \infty} W_h^t v = a_m \lambda_m^t u_m \quad where m = \operatorname{argmax}_j \lambda_j \quad (۴-۱۱-۵۰)$$

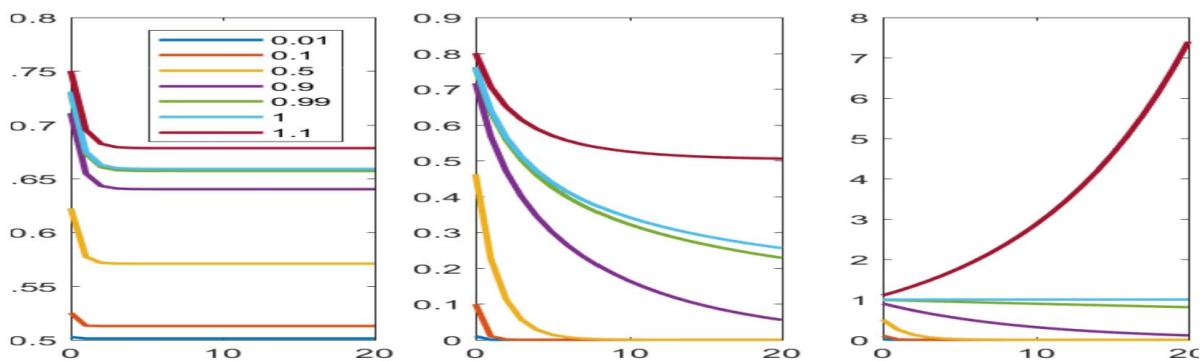
در نتیجه با پیشروی در شبکه مقادیر ویژه بزرگ تر باقی می مانند و بزرگترین مقدار ویژه ماتریس W_h رفتار سیستم را تعیین می کند به طوری که اگر بزرگترین مقدار ویژه بزرگتر از یک باشد سیستم به سمت انفجار مقادیر پارامتر ها و اگر بزرگترین مقدار ویژه کوچک از یک باشد سیستم به سمت محوشوندگی مقادیر پارامتر ها می رود ، بنابراین رفتار شبکه در بلندمدت کاملا به مقادیر ویژه ماتریس وزن های لایه مخفی وابسته است .

با بررسی دو حالت ذکر شده می توان نتیجه گرفت این نوع شبکه ها در مرحله آموزش و پس انتشار خطاب نیز تنها تحت تاثیر گرادیان های نزدیک تر قرار بگیرند و گرادیان لایه های دور تر را به نوعی فراموش کنند.

در صورتی که سیستم را با اعمال توابع فعالسازی غیرخطی در نظر بگیریم می توان سیستم را به صورت غیرخطی در نظر گرفت و رفتار سیستم را در صورت اعمال انواع توابع فعالسازی بررسی نمود [۱۹].

$$h(t) = f(w_h h(t-1) + w_x x(t)) \quad (۴-۱۱-۵۱)$$

سه تابع فعالسازی سیگموئید و تانژانت هایپربولیک و ReLU را به عنوان تابع فعالسازی f و مقدار عددی ۱ را به عنوان نقطه شروع در نظر می گیریم و نتایج را بررسی می کنیم.



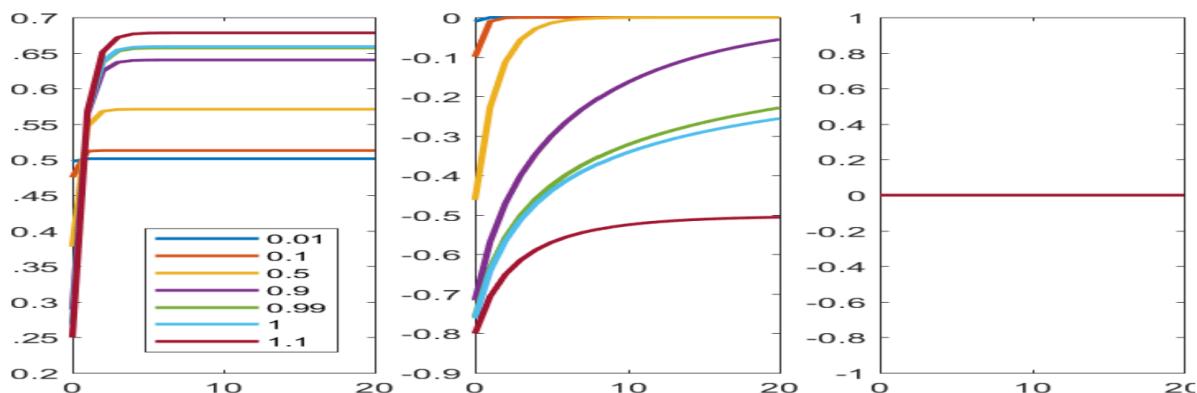
شکل ۴-۱۱-۵۱: نمودار پاسخ سیستم به ازای وزن های متفاوت و توابع فعالسازی به ترتیب ReLU و تانژانت هایپربولیک و سیگموئید [۱۹]

نمودار سمت چپ نتیجه اعمال تابع فعالسازی سیگموئید است و همانطور که ملاحظه می شود پاسخ سیستم مستقل از ورودی سریعاً همگرا شده است و کاملاً وابسته به وزن های شبکه است.

نمودار سمت راست نتیجه اعمال تابع فعالسازی ReLU است و همانطور که ملاحظه می شود پاسخ سیستم به ازای وزن های بزرگ تر از یک کاملاً واگرا است و به ازای وزن های کوچک تر از یک سریعاً صفر می شود و کاملاً حساس به مقادیر وزن ها است.

نمودار وسط نتیجه اعمال تابع فعالسازی تانژانت هایپربولیک است و همانطور که ملاحظه می شود این تابع نسبت به توابع دیگر دیرتر همگرا می شود و به ورودی اعمالی شده حساس تر است.

همچنین با درنظر گرفتن نقطه ۱- به عنوان نقطه شروع و اعمال توابع فعالسازی نتایج زیر حاصل می شود.



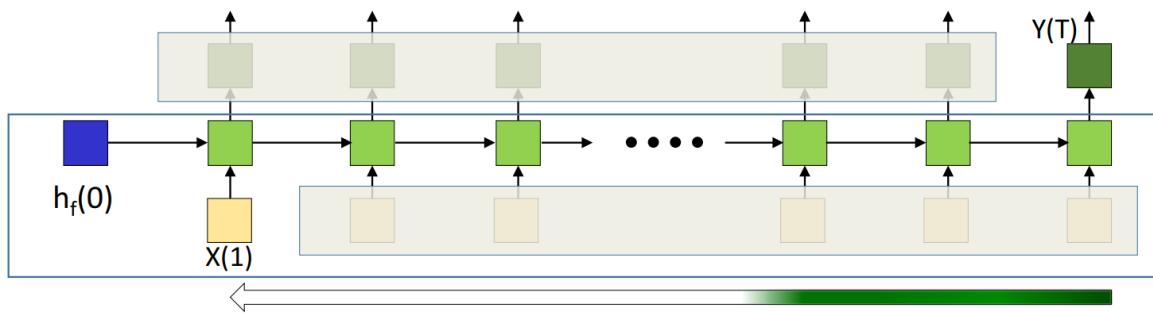
شکل ۸۱-۴: نمودار پاسخ سیستم به ازای وزن های متفاوت و توابع فعالسازی به ترتیب Relu و تانژانت هایپربولیک و سیگموئید [۱۹]
همانطور که ملاحظه می شود نمودار سمت چپ مربوط به تابع فعالسازی سیگموئید با تغییر ورودی حساسیتی نشان نداد و کاملاً وابسته به وزن ها است و در عمل در شبکه یادگیری صورت نمی گیرد.

با توجه به منفی بودن نقطه شروع همانطور که ملاحظه می شود خروجی تابع Relu برابر صفر خواهد بود.

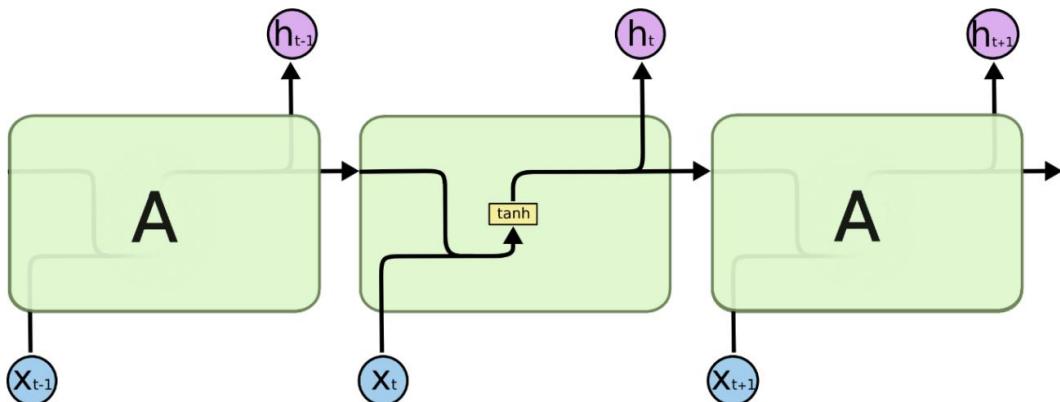
همانطور که ملاحظه می شود با توجه به حساسیت بیشتر به ورودی و سرعت همگرایی تابع تانژانت هایپربولیک بهترین گزینه برای شبکه های عصبی بازگشتی با در نظر گرفتن حافظه در این نوع شبکه ها به نسبت دو تابع فعالسازی دیگر است [۱۹].

در نتیجه در صورتی که تعداد ورودی های شبکه های عصبی بازگشتی بالا بود شبکه به راحتی اطلاعات را از دست می دهد و به نوعی ورودی های اولیه را فراموش میکند و از حافظه خوبی برخوردار نیست.

به عنوان نمونه این نوع شبکه ها در مرحله آموزش با توالی هایی از کلمات با تعداد بالا که نیاز به حافظه برای به یاد سپردن فاعل و فعل جملات دارند به خوبی عمل نمی کنند و ممکن است فاعل جملات را فراموش کنند.

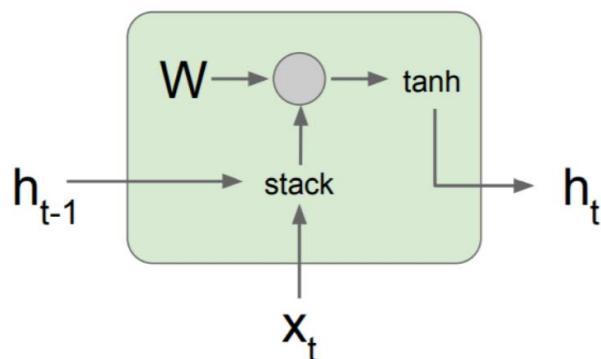


شکل ۸۲-۴: کاهش حافظه شبکه های عصبی بازگشتی به ازای توالی های طولانی [۱۹]



شکل ۸۳-۴ : بلوک های سازنده شبکه های عصبی بازگشتی [۴۳]

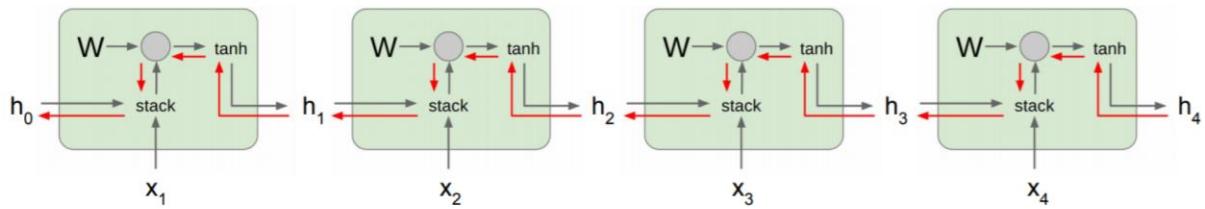
بنابراین می توانیم بلوک های سازنده یک شبکه عصبی بازگشتی را به صورت زیر در نظر بگیریم :



شکل ۸۴-۴ : بلوک سازنده شبکه های عصبی بازگشتی [۴۳]

با در کنار هم قرار دادن ماتریس های وزن W_{hh} و W_{hx} ماتریس W تشکیل می شود و مراحل به صورت ضرب ماتریسی در هر بلوک جلو خواهد رفت [۳۶] و [۱۹].

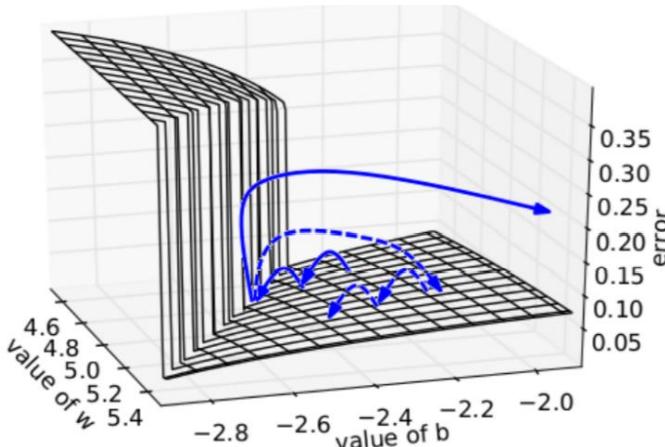
$$\begin{aligned}
 h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\
 &= \tanh((W_{hh} \quad W_{xh}) \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}) \\
 &= \tanh(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix})
 \end{aligned} \tag{۴-۱۱-۵۲}$$



شکل ۸۵-۴ : جریان پس انتشار خطا در بلوک های شبکه های عصبی بازگشتی [۴۳]

همانطور که در بخش قبل بحث شد ضرب ماتریس های وزن مشترک در هر بلوک شبکه های عصبی بازگشتی سبب حساسیت شبکه به مقادیر ویژه این ماتریس وزن می شود و موجب انفجار و یا محوشوندگی اطلاعات در طول شبکه می شود و به همین علت شبکه های عصبی بازگشتی از حافظه خوبی برخوردار نیستند.

یکی از راه هایی که میتواند از انفجار گرادیان در این شبکه ها در فرآیند آموزش جلوگیری کند تکنیک برش گرادیان با استفاده از نرم گرادیان است که باعث می شود بر خلاف برش گرادیان صرفا بر اساس مقدار آن جهت گرادیان نیز حفظ شود.

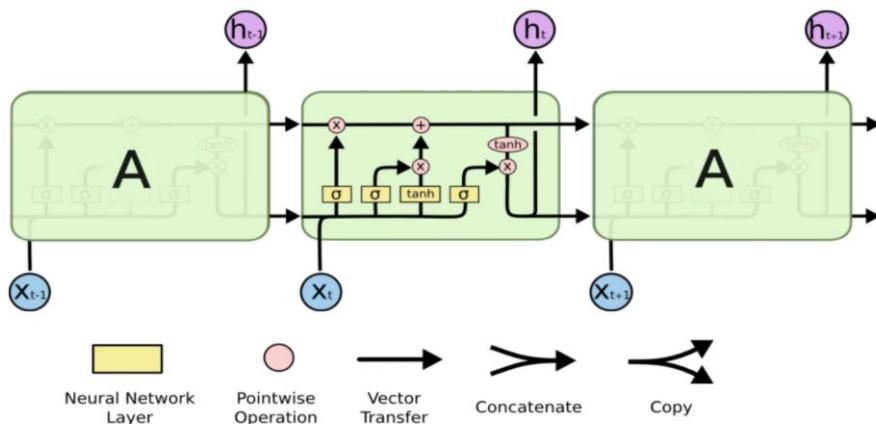


شکل ۸۶-۴ : برش گرادیان به روشن نرم گرادیان [۳۶]

۱۲-۴ معرفی شبکه های عصبی با حافظه طولانی کوتاه مدت^۱ (LSTM) :

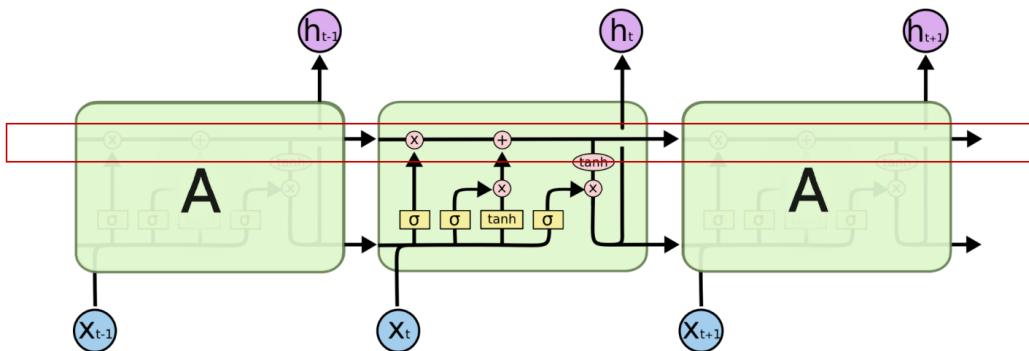
ایده اصلی این نوع از شبکه ها در جهت بهبود شبکه های عصبی بازگشتی ، حفظ اطلاعات در بلندمدت با استفاده از بلوک های پایه پیچیده تر بوده است به گونه ای که در هر بلوک با توجه به ورودی آن مشخص می شود چه مقدار از اطلاعات بلوک قبل حفظ و چه مقدار از آن فراموش شود و در این حالت پارامتر های شبکه مشترک نیست که این خود باعث می شود هر بلوک با توجه به ورودی های خود تصمیم بر حفظ و یا فراموشی اطلاعات بلوک های قبل بگیرد [۱۸] و [۴۳].

^۱ Long short-term memory



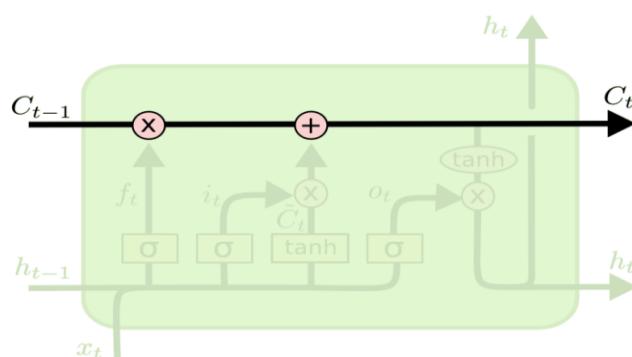
شکل ۸۷-۴ : بلوک های سازنده شبکه های با حافظه طولانی کوتاه مدت [۴۳]

همانطور که ملاحظه می شود هر بلوک شامل توابع و اتصالاتی است که هر کدام از آن ها وظیفه خاصی را بر عهده دارند.



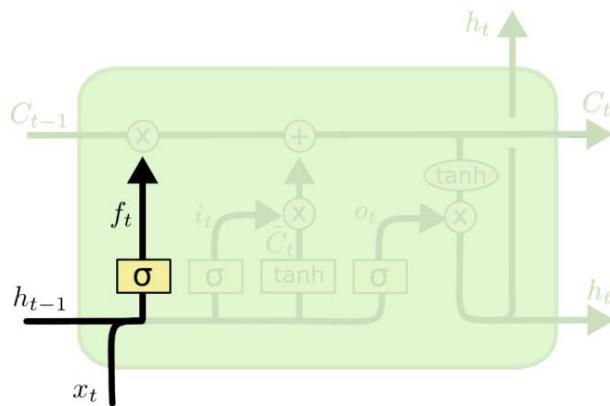
شکل ۸۸-۴ : بلوک های سازنده شبکه های با حافظه طولانی کوتاه مدت [۴۳]

یکی از عناصر مهم این نوع شبکه ها سلول حافظه شبکه می باشد و توسط دو دروازه که تحت تاثیر ورودی در همان لحظه و حالت بلوک قبل تصمیم می گیرد که کدام اطلاعات در حافظه شبکه بمانند و کدام اطلاعات فراموش شوند و چه اطلاعات جدیدی به عنوان حافظه سیستم در نظر گرفته شود [۴۳] .



شکل ۸۹-۴ : بلوک های سازنده شبکه های با حافظه طولانی کوتاه مدت و سلول حافظه [۴۳]

دروازه فراموشی :



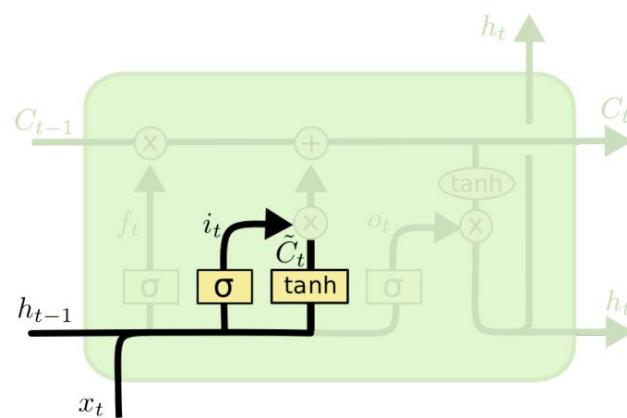
شکل ۹۰-۴ : بلوک های سازنده شبکه های با حافظه طولانی کوتاه مدت و دروازه فراموشی [۴۳]

$$f_t = \sigma(W_f \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} + b_f) \quad (4-12-1)$$

این دروازه با اعمال وزن های مربوط به خود به ورودی بلوک و حالت قبلی شبکه و اعمال تابع سیگموئید که مقادیر بین صفر و یک را خروجی می دهد [۴۳] .

این دروازه تعیین می کند که چه میزان از حافظه شبکه به بلوک بعدی انتقال داده شود بنابراین شبکه سعی می کند با یافتن وزن های مناسب مربوط به دروازه های فراموشی (W_f) و مقادیر بایاس (b_f) آموزش ببیند که چه اطلاعاتی از حافظه باید فراموش شوند و چه اطلاعاتی از حافظه انتقال داده شوند .

دروازه ورودی :



شکل ۹۱-۴ : بلوک های سازنده شبکه های با حافظه طولانی کوتاه مدت و دروازه ورودی [۴۳]

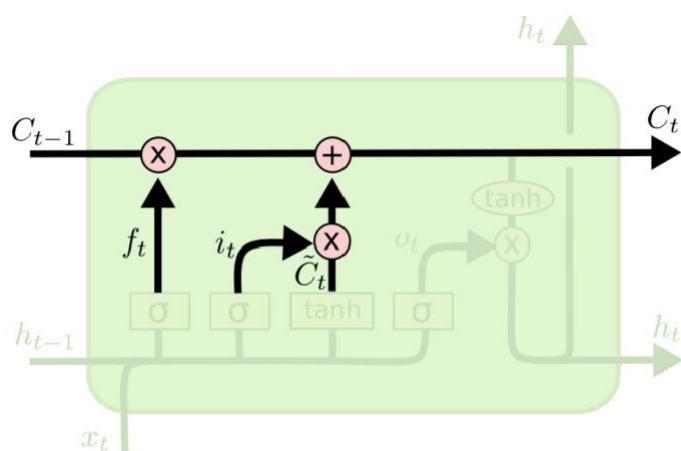
$$\tilde{C}_t = \tanh(W_C \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} + b_C) \quad (4-12-2)$$

$$i_t = \sigma(W_i \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} + b_i) \quad (4-12-3)$$

در محاسبه \tilde{C}_t با در نظر گرفتن ورودی و حالت لحظه قبل به عنوان ورودی یک لایه پرسپترون و اعمال خروجی لایه پرسپترون به تابع تانژانت هایپربولیک که مشابه یک بلوک از شبکه های عصبی بازگشتی است سعی می شود اطلاعات مفید موجود در ورودی و حالت لحظه قبل برای به روزرسانی حافظه استخراج شود.

و در شاخه دروازه i_t تصمیم گرفته می شود که از بین اطلاعات استخراج شده از ورودی کدام اطلاعات به حافظه سپرده شود و برای به روزرسانی به سلول حافظه ارسال شود [۴۳].

دروازه های به روزرسانی سلول حافظه :



شکل ۹۲-۴ : دروازه بروزرسانی سلول حافظه [۴۳]

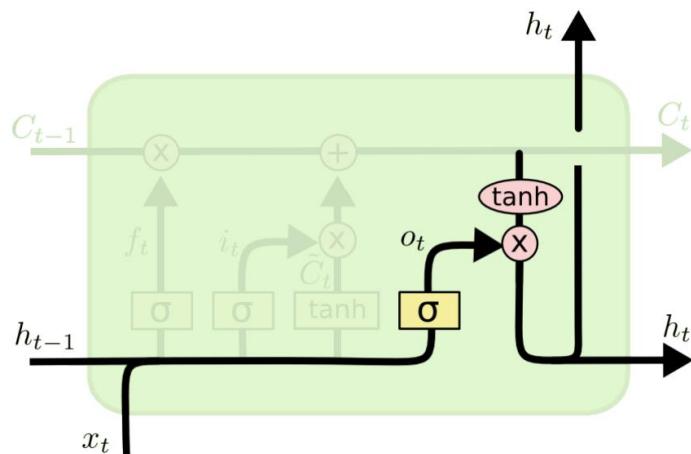
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \quad (4-12-4)$$

در این بخش از بلوک شبکه های LSTM به بروزرسانی سلول حافظه پرداخته می شود .

خروجی دروازه فراموشی همانطور که بحث شد مشخص می کند که از بین اطلاعات موجود در حافظه چه اطلاعاتی نیاز به فراموشی دارند و چه اطلاعاتی همچنان باید حفظ شود و با ضرب نقطه ای بردار مربوط به خروجی این دروازه با حافظه قبلی C_{t-1} این امر محقق می شود [۴۳] .

با ضرب نقطه ای ماتریس های C_t و i_t مشخص می شود که چه اطلاعاتی با توجه به ورودی و حالت قبلی برای بروزرسانی حافظه انتخاب می شوند و در نهایت با جمع ماتریس ها مقادیر حافظه بروزرسانی می شوند.

دروازه خروجی :



شکل ۹۳-۴ : دروازه خروجی شبکه های با حافظه طولانی کوتاه مدت [۴۳]

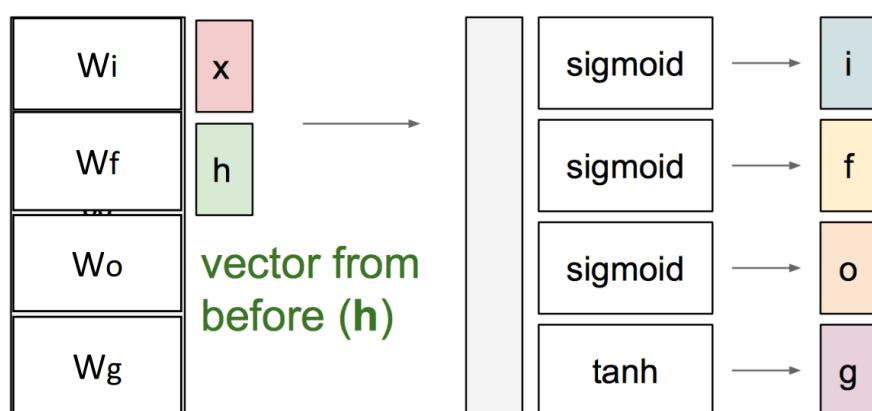
$$o_t = \sigma(W_o \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} + b_o) \quad (4-12-5)$$

$$h_t = o_t \odot \tanh(C_t) \quad (4-12-6)$$

در این بخش از بلوک شبکه ابتدا مقادیر موجود در حافظه با اعمال به تابع تائزانت هایپربولیک بین -1 و $+1$ فشرده می شود و با محاسبه دروازه خروجی o_t و آموزش وزن های مربوط به آن شبکه تصمیم می گیرد که چه اطلاعاتی از حافظه فعلی را به عنوان مقادیر حالت به حالت بعدی منتقل کند [۴۳].

در مجموع محاسبات ماتریسی مربوط به یک بلوک شبکه LSTM به صورت زیر خواهد بود .

نام دیگر دروازه \tilde{C}_t ، دروازه g می باشد .

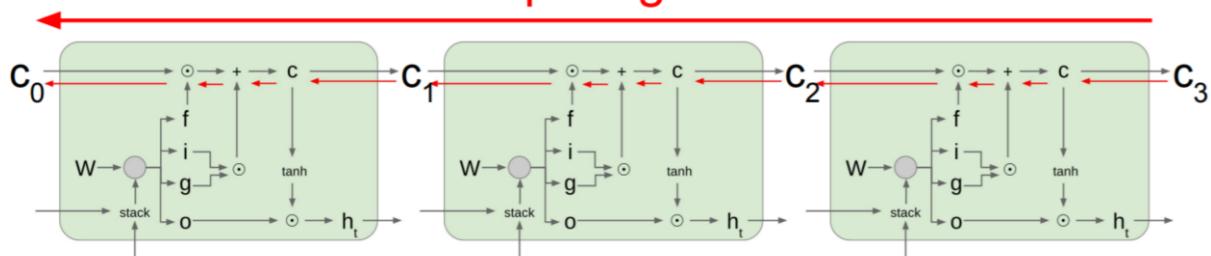


شکل ۹۴-۴ : محاسبات ماتریسی شبکه های با حافظه طولانی کوتاه مدت [۴۳]

$$g = \tilde{C}_t \quad (4-12-7)$$

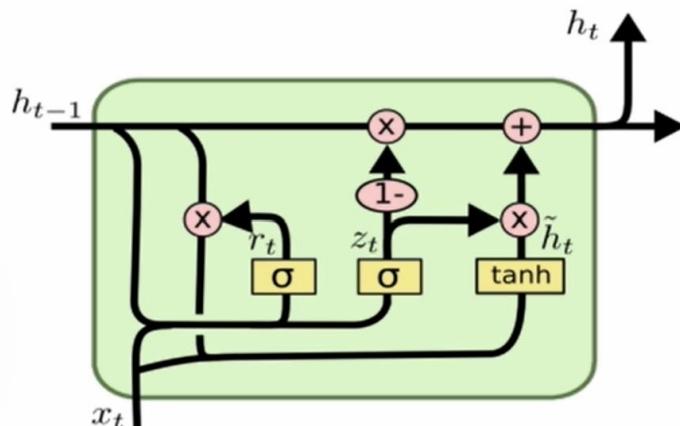
همانطور که مشاهده می شود بر خلاف بلوک های سازنده شبکه های عصبی بازگشتی که همواره یک ماتریس وزن اشتراکی در ورودی و حالت قبلی ضرب ماتریسی می شود در سلول حافظه شبکه های LSTM تنها دو عملگر جمع ماتریسی و ضرب نقطه ای که به صورت ضرب درایه به درایه عمل می کند در نظر گرفته شده است و با توجه به اینکه دیگر پارامتر های شبکه به صورت اشتراکی استفاده نمی شوند و هر بلوک وزن های مخصوص به خود را فرا گرفته است مشکل محوشدنگی و انفجار گرادیان در طول شبکه به طور چشمگیری کاهش می یابد [۴۳] و [۴۴].

Uninterrupted gradient flow!



شکل ۹۵-۴ : بلوک های سازنده شبکه های با حافظه طولانی کوتاه مدت و جریان گرادیان در سلول حافظه [۴۳]

۱۳-۴ معرفی شبکه های عصبی با واحد های بازگشتی دروازه ای^۱ (GRU) :



شکل ۹۶-۴ : بلوک های سازنده شبکه های با واحد های بازگشتی دروازه ای [۴۳]

$$z_t = \sigma(W_z \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} + b_z) \quad (4-13-1)$$

$$r_t = \sigma(W_r \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} + b_r) \quad (4-13-2)$$

¹ Gated Recurrent Unit

$$\tilde{h}_t = \tanh(W_m \begin{bmatrix} r_t & h_{t-1} \\ x_t \end{bmatrix} + b_m) \quad (4-13-3)$$

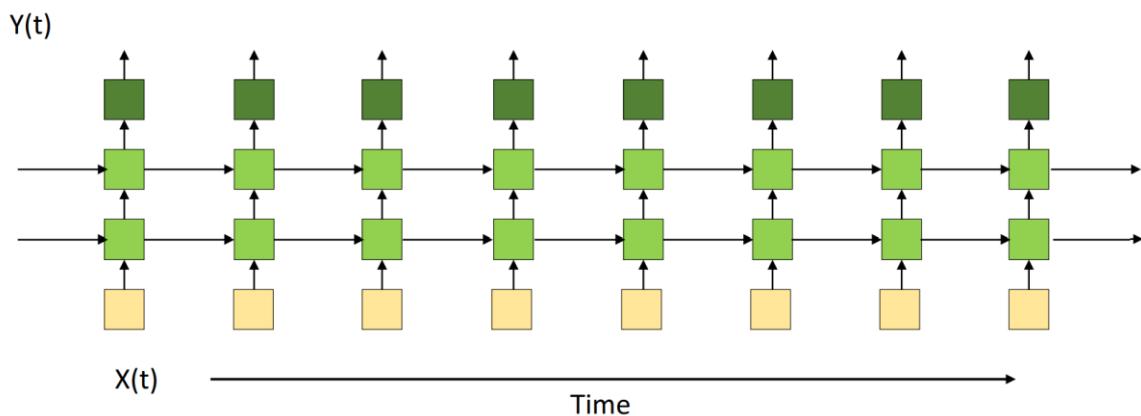
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \quad (4-13-4)$$

این نوع از واحد ها به منظور ساده سازی واحد های تشکیل دهنده شبکه های LSTM معرفی شدند و با کاهش متغیر ها و دروازه های هر واحد و حذف سلول حافظه سعی شده است که با کنترل حالت هر واحد از پیچیدگی واحد های LSTM کاسته شود [۴۵].

با محاسبه دروازه مربوط به متغیر r_t مشخص می شود که چه اطلاعاتی از حالت قبل برای بروزرسانی حالت جدید در نظر گرفته شود و چه اطلاعاتی فراموش شود ، به عنوان مثال اگر توالی های ورودی شبکه هایی که واحد سازنده آن ها به صورت دروازه های بازگشتی است دارای وابستگی های بلند مدت نباشند معمولاً متغیر r_t مقادیر نزدیک به صفر پیدا می کند و حالت های قبلی سیستم را در نظر نمی گیرد [۴۶].

محاسبه دروازه مربوط به متغیر \tilde{h}_t اطلاعات لازم را با توجه به ورودی و خروجی اعمال دروازه متغیر r_t به حالت قبلی برای بروزرسانی حالت جدید استخراج می کند و دروازه متغیر z_t مشخص می کند که چه اطلاعاتی از حالت های قبلی سیستم و چه اطلاعاتی جدیدی از طریق متغیر \tilde{h}_t وارد حالت جدید شود و در نهایت h_t به عنوان حالت جدید سیستم محاسبه می شود [۴۶] و [۱۹].

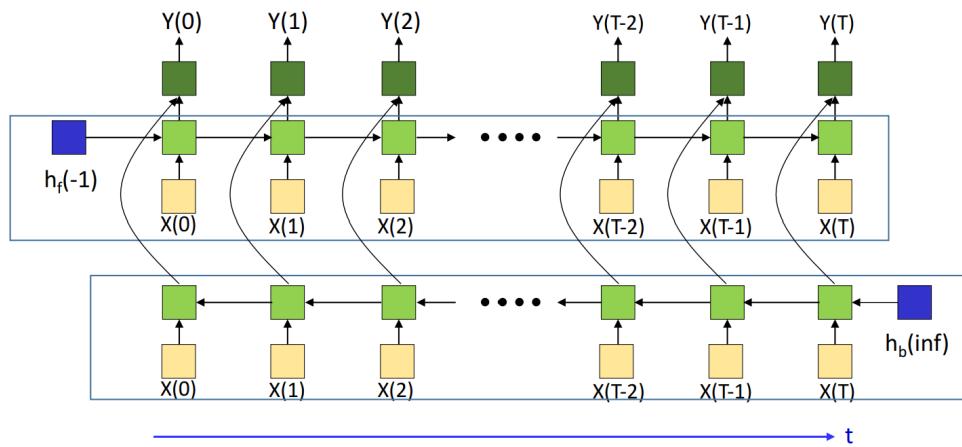
همچنین می توان از کنار هم قرار دادن واحد های متوالی LSTM و یا GRU به صورت پشته شبکه هایی با بیش از یک لایه را برای انواع مسائل استفاده کرد و با افزایش تعداد لایه ها الگوهای پیچیده تری را از توالی ورودی شبکه استخراج کرد .



شکل ۴-۹۷: گراف محاسباتی شبکه های عصبی چند لایه [۱۹]

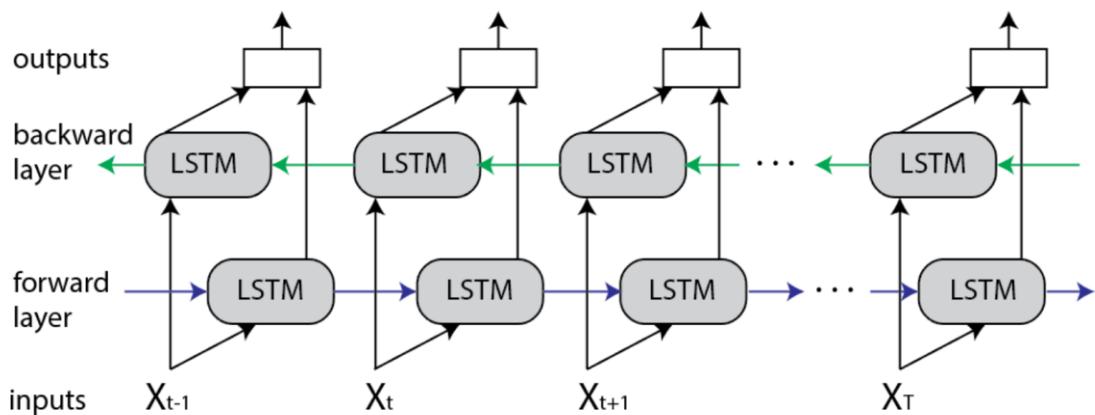
۱۴-۴ شبکه های عصبی دو طرفه :

همانطور که در شبکه های عصبی بازگشتی بحث شد می توان از دو لایه از شبکه های عصبی بازگشتی در دو جهت متفاوت یکی از اول توالی ورودی به سمت آخر آن و دیگری از آخر توالی ورودی به اول آن استفاده کرد و با آموزش این دو شبکه و استفاده از خروجی هر دو این شبکه های یک طرفه خروجی شبکه دو طرفه را پیش بینی کرد .



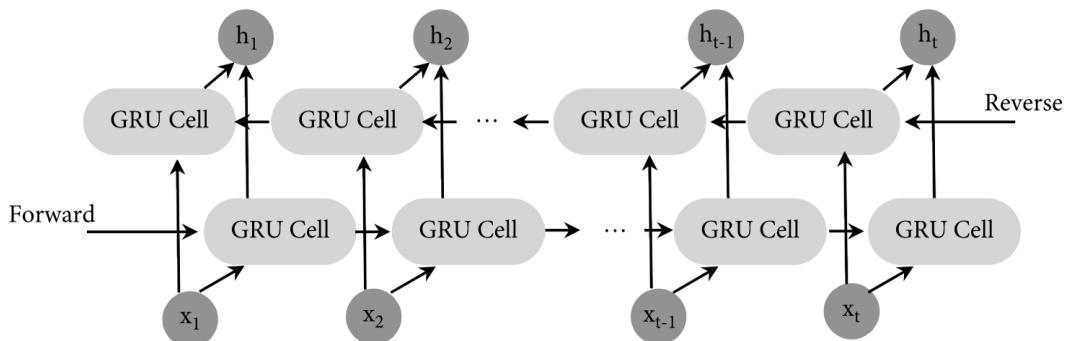
شکل ۹۸-۴ : گراف شبکه های عصبی بازگشتی دو طرفه [۱۹]

حال اگر به جای واحد های سازنده شبکه های عصبی بازگشتی از واحد های LSTM استفاده شود یک شبکه LSTM دو طرفه خواهیم داشت و این نوع از شبکه ها کاربرد فراوانی در پردازش زبان طبیعی دارند چون علاوه بر گذشته یک توالی ورودی ، آینده آن را نیز در نظر می گیرند اما سرعت فرایند آموزش آن زمان بیشتری را نیاز دارد [۴۷].



شکل ۹۹-۴ : گراف شبکه های عصبی با حافظه طولانی کوتاه مدت دو طرفه [۴۷]

می توان با قرار دادن واحد های سازنده شبکه های GRU به جای LSTM به یک شبکه دو طرفه GRU دست یافت که با توجه به ساده تر بودن واحد های GRU به نسبت LSTM و پارامتر های کمتر آن می توان انتظار افزایش سرعت فرایند آموزش شبکه را داشت .

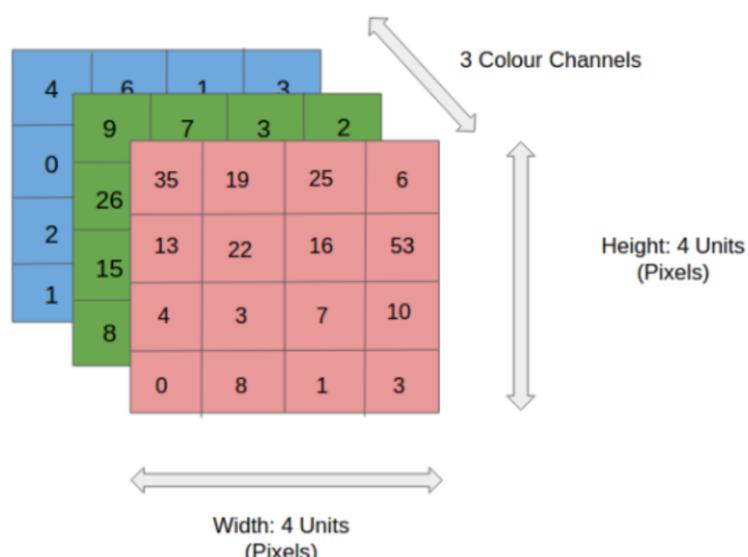


شکل ۱۰۰-۴ : گراف شبکه های با واحد های بازگشتی دروازه ای دو طرفه [۴۷]

۱۵-۴ شبکه های عصبی کانولوشنی^۱:

شبکه های عصبی کانولوشنی دارای الگوریتم هایی هستند که با دریافت یک ماتریس از ورودی ها که می تواند یک تصویر که نشان دهنده مقدار هر پیکسل و یا ماتریسی متعدد از کنار هم قرار دادن بردار کلمات باشد و یادگیری ویژگی های ساده و پیچیده آن که بستگی به عمق شبکه دارد سعی می کند ویژگی های هر ماتریس را یاد بگیرد و بتواند بین ورودی های جدید تمایز قائل شود .

در شبکه های عصبی کانولوشنی وابسته به عمق شبکه همواره ماتریس ورودی در طول شبکه به فرم های ساده تری برای پردازش تبدیل می شوند و سعی می شود در این فرآیند ویژگی های مهمی که در پیش بینی بهتر شبکه موثر هستند باقی بمانند .



شکل ۱۰۱-۴ : ماتریس سه کanal رنگی تصاویر [۴۸]

¹ Convolutional Neural Networks(CNN)

در این نوع شبکه در هر لایه می تواند فیلتر های متفاوتی وجود داشته باشد که با اعمال آن ها به ماتریس های لایه قبل عملیات کانولوشن صورت می گیرد . به عنوان نمونه با اعمال فیلتر زیر و حرکت روی آن روی ماتریس ورودی حاصل کانولوشن آن فیلتر با ماتریس ورودی به دست می آید [۴۸] .

$$\text{فیلتر} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

| | | | | |
|--------------------|--------------------|--------------------|---|---|
| 1 _{x1} | 1 _{x0} | 1 _{x1} | 0 | 0 |
| 0 _{x0} | 1 _{x1} | 1 _{x0} | 1 | 0 |
| 0 _{x1} | 0 _{x0} | 1 _{x1} | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Image

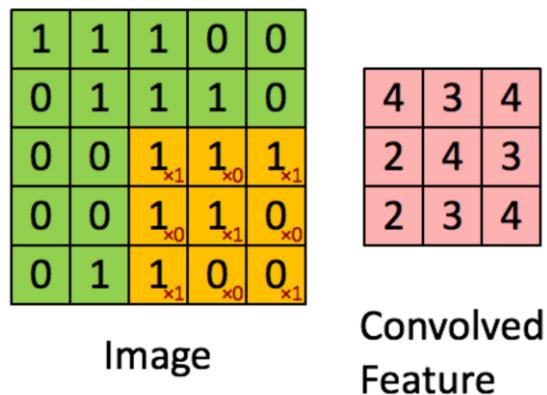
Convolved Feature

| | | | | |
|---|--------------------|--------------------|--------------------|---|
| 1 | 1 _{x1} | 1 _{x0} | 0 _{x1} | 0 |
| 0 | 1 _{x0} | 1 _{x1} | 1 _{x0} | 0 |
| 0 | 0 _{x1} | 1 _{x0} | 1 _{x1} | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Image

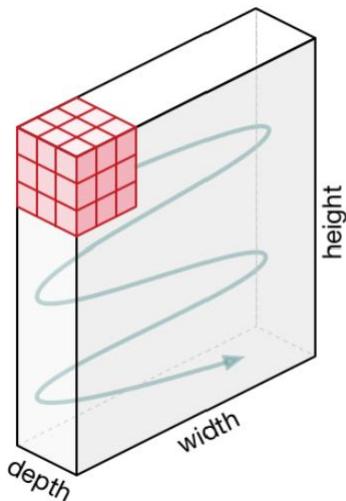
Convolved Feature

شكل ۱۰۲-۴ : عملیات کانولوشن و اعمال فیلتر ها [۴۸]



شکل ۱۰۳-۴ : عملیات کانولوشن و اعمال فیلتر ها [۴۸]

همانطور که مشاهده می شود با اعمال فیلتر مورد نظر به ماتریس ورودی به صورت ضرب درآیده آن ماتریس و سپس جمع مقادیر حاصل و حرکت فیلتر با گام یک در طول و عرض ماتریس ورودی ، ماتریس حاصل کانولوشن فیلتر با ورودی حاصل شده است و وابسته به ورودی و معماری شبکه فیلتر ها می توانند ابعاد متفاوتی داشته باشند و همواره عمق فیلتر ها با عمق ماتریس ورودی یکسان است و انواع ویژگی ها را از ماتریس ورودی استخراج می کنند .



شکل ۱۰۴-۴ : عملیات کانولوشن و اعمال فیلتر ها [۴۸]

گام حرکت فیلتر(stride) در طول و عرض ماتریس ورودی نیز یک پارامتر قابل تنظیم است و می توان با گام های بیشتر نیز در طول و یا عرض حرکت کرد که باعث تغییر ابعاد ماتریس خروجی عملیات کانولوشن خواهد شد و معمولاً زمانی که نیاز داریم پارامتر های شبکه را کاهش دهیم از گام های بیشتر از یک استفاده می شود [۴۸].

همانطور که مشاهده می شود اندازه ماتریس حاصل شده از عملیات کانولوشن کوچک تر ماتریس ورودی است و در بسیاری از مسائل نیاز داریم نتیجه عملیات کانولوشن از لحاظ ابعاد ماتریس با ورودی یکسان باشد و همه اطلاعات موجود در ماتریس ورودی در نظر گرفته شود و این امر از طریق کاشتن درایه هایی جدید(padding) در اطراف ماتریس ورودی حاصل می شود [۴۸].

| Image | Filter | Convolved Image |
|-------------------|------------|-----------------|
| 1 5 4 3 2 | 0 1 0 | 4 3 0 |
| 2 2 3 4 5 | 0 -1 0 | 2 3 4 |
| 2 1 2 1 1 | 0 1 0 | 2 1 3 |
| 1 1 2 1 4 | | |
| 5 2 1 3 2 | | |

| Padding | Stride |
|---------------------------|-------------------|
| 0 0 0 0 0 0 0 | 1 5 4 3 2 |
| 0 1 5 4 3 2 0 | 2 2 3 4 5 |
| 0 2 2 3 4 5 0 | 2 1 2 1 1 |
| 0 2 1 2 1 1 0 | 1 1 2 1 4 |
| 0 1 1 2 1 4 0 | 5 2 1 3 2 |
| 0 5 2 1 3 2 0 | |
| 0 0 0 0 0 0 0 | |

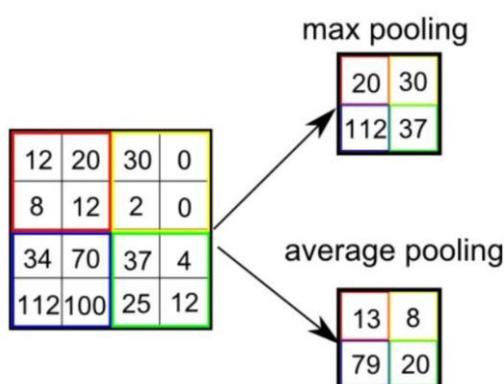
شکل ۱۰۵-۴ : عملیات کانولوشن و اعمال فیلتر ها و کاشت درایه صفر [۴۸]

در شبکه های کانولوشنی فیلتر های اعمالی معادل نورون ها در شبکه های عصبی پرسپترون هستند و با یادگیری وزن های مربوط به هر فیلتر در طول فرآیند آموزش ویژگی های ماتریس های ورودی استخراج می شوند.

لایه ادغام (pooling) :

این لایه با توجه به اندازه آن همانند فیلتر ها با حرکت روی ماتریس ورودی و نوع لایه ادغام عملگر های متفاوتی را روی ناحیه های مختلف ورودی اعمال می کند.

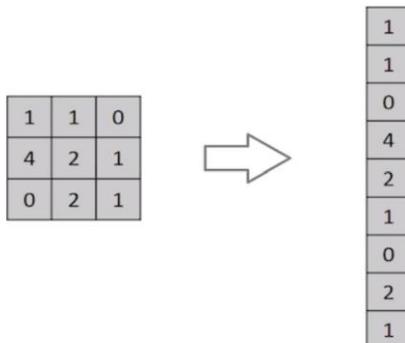
به عنوان نمونه دو نوع لایه ادغام ماکزیمم و میانگین بر روی ماتریس زیر اعمال شده است.



شکل ۱۰۶-۴ : عملیات ادغام [۴۸]

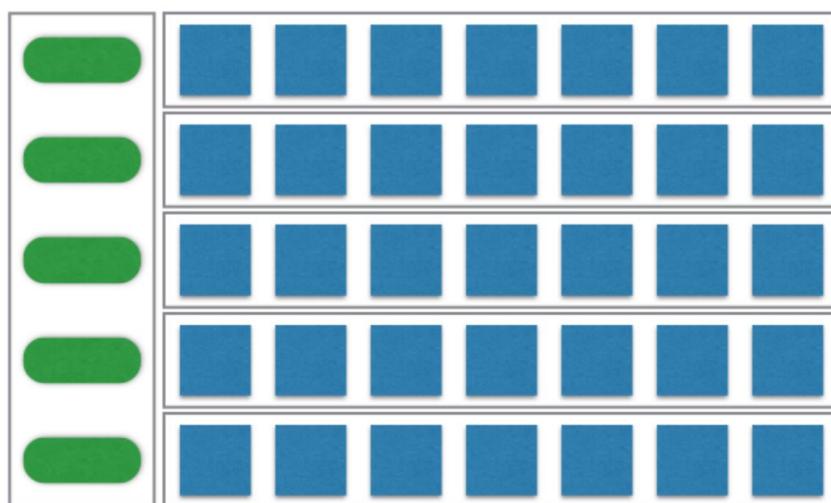
همانطور که مشاهده می شود اعمال این لایه بر روی ورودی باعث کاهش شبکه و کاهش خطر بیش برآش مدل و همچنین کاهش پیچیدگی محاسباتی مدل و افزایش سرعت فرآیند آموزش می شود .

یکی دیگر از لایه های پر استفاده در شبکه های کانولوشنی لایه مسطح سازی است که یک ماتریس را به عنوان ورودی می گیرد و با در کنار هم قرار دادن درایه های آن ماتریس ، برداری از درایه های آن را به عنوان خروجی می دهد .



شکل ۱۰۷-۴ : عملیات مسطح سازی [۴۸]

همانطور که مشاهده می شود ورودی شبکه های کانولوشنی یک ماتریس است که می تواند از مقادیر پیکسل های تصویر و یا از کنار هم قرار دادن بردار های مربوط به یک توالی در حوزه زمان یا مکان تشکیل شده باشد .

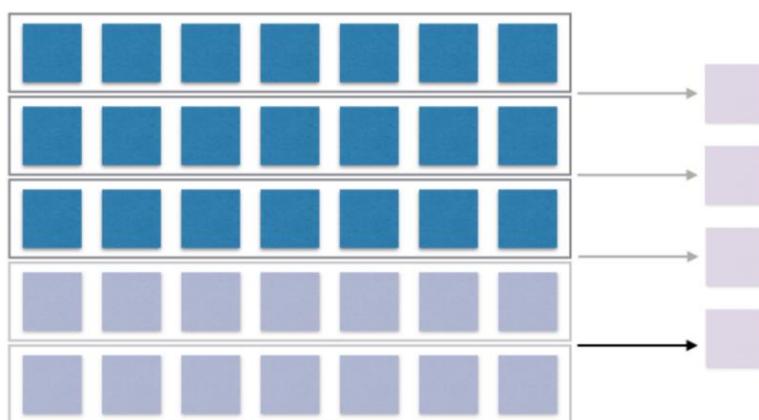


شکل ۱۰۸-۴ : تشکیل ماتریس توسط بردار های بازنمایی هر کلمه [۴۹]

به عنوان مثال در پردازش زبان طبیعی هر کلمه در متن می تواند توسط یک بردار از مقادیر نمایش داده شود که با روی هم قرار دادن این بردار ها به ماتریس کلمات موجود در متن خواهیم رسید و همچنین با تعریف فیلتر با ابعاد مورد نیاز مسئله و اعمال عملیات کانولوشن ویژگی های موجود در متن استخراج می شود [۴۹] .

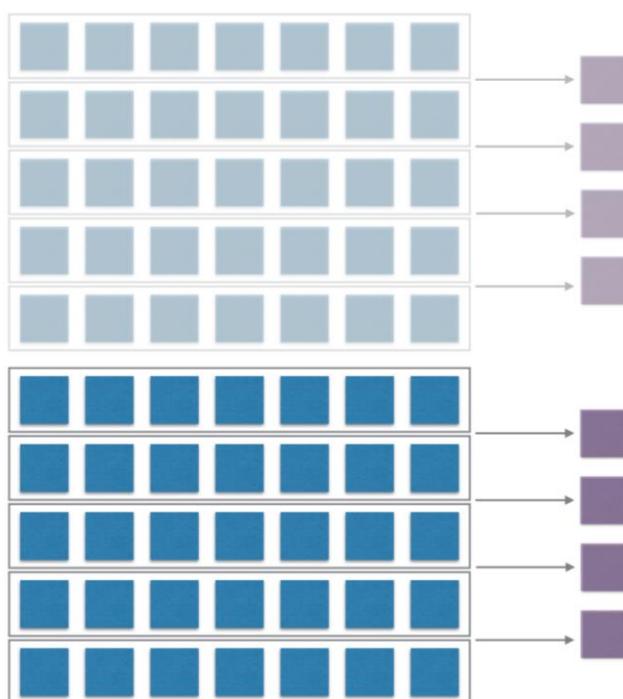


شکل ۱۰۹-۴ : فیلتر اعمالی بر ماتریس کلمات [۴۹]



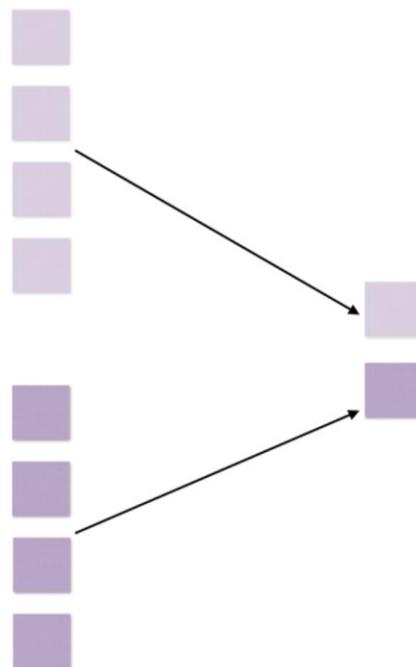
شکل ۱۱۰-۴ : عملیات کانولوشن [۴۹]

با اعمال دو فیلتر با به ماتریس بازنمایی کلمات ورودی ویژگی های موجود با توجه به ابعاد فیلتر در همسایگی کلمات مورد نظر استخراج می شود .

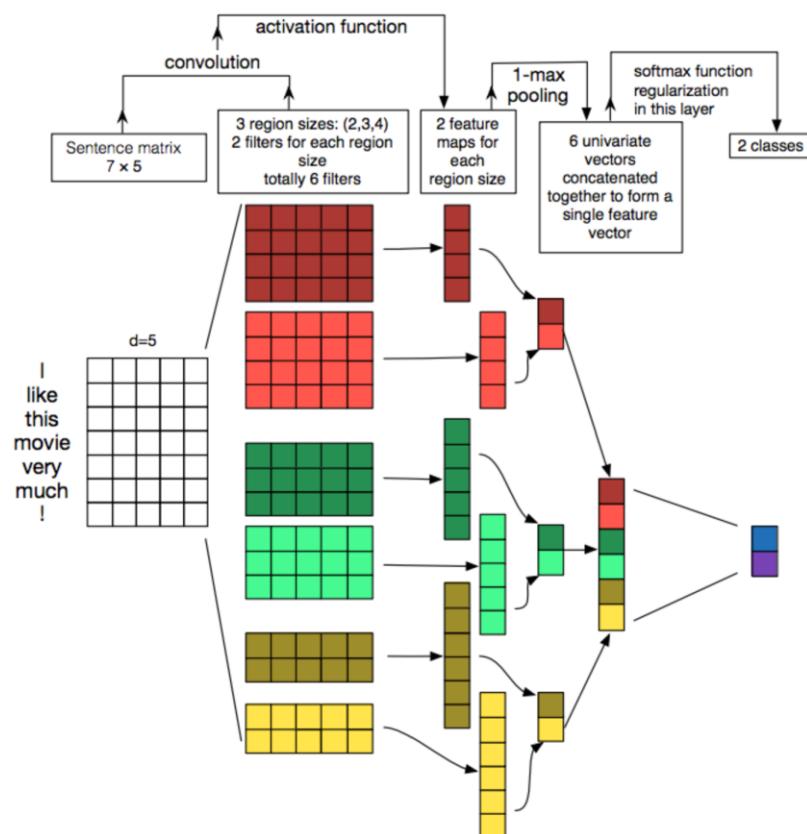


شکل ۱۱۱-۴ : دو فیلتر اعمالی بر ماتریس کلمات [۴۹]

در نهایت همانند تصاویر می توان از لایه های ادغام نیز استفاده نمود و دو بردار ویژگی را با یکدیگر ادغام نمود و پیچیدگی محاسباتی شبکه را کاهش داد.



شکل ۱۱۲-۴ : عملیات ادغام [۴۹]

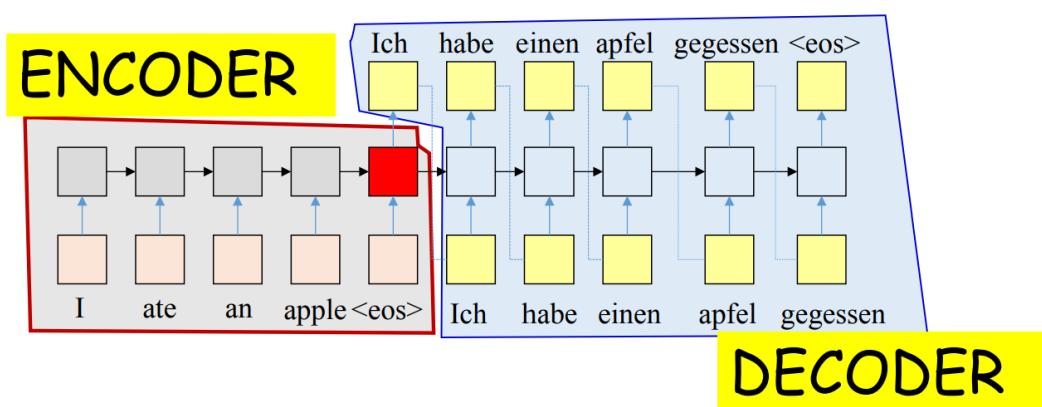


شکل ۱۱۳-۴ : نمای کلی یک شبکه کانالوشن با ورودی ماتریس کلمات [۴۹]

۱۶-۴ مدل های بر پایه توجه :

همانطور که در بخش های قبل در مورد انواع مدل های مبتنی بر شبکه های عصبی بازگشتی بحث شد این مدل ها می توانند انواع متفاوتی با توجه به توالی ورودی و خروجی و مسئله داشته باشند .

به عنوان نمونه در مدل های از نوع چند به چند که علاوه بر ورودی ، خروجی نیز دارای توالی مشخصی است شبکه به دو بخش کدگذار و کدگشا تقسیم می شود که در بخش کدگذار تمام دنباله ورودی در یک واحد لایه مخفی کد می شوند که این فرآیند باعث سریز اطلاعات و از بین رفتی بخشی از اطلاعات توالی ورودی می شود به ویژه اگر دنباله ورودی طولانی باشد و علاوه بر آن با افزایش طول توالی ورودی ، پارامتر های شبکه نیز افزایش می یابند و ممکن است شبکه دچار بیش برازش شود [۱۹]-[۵۱].

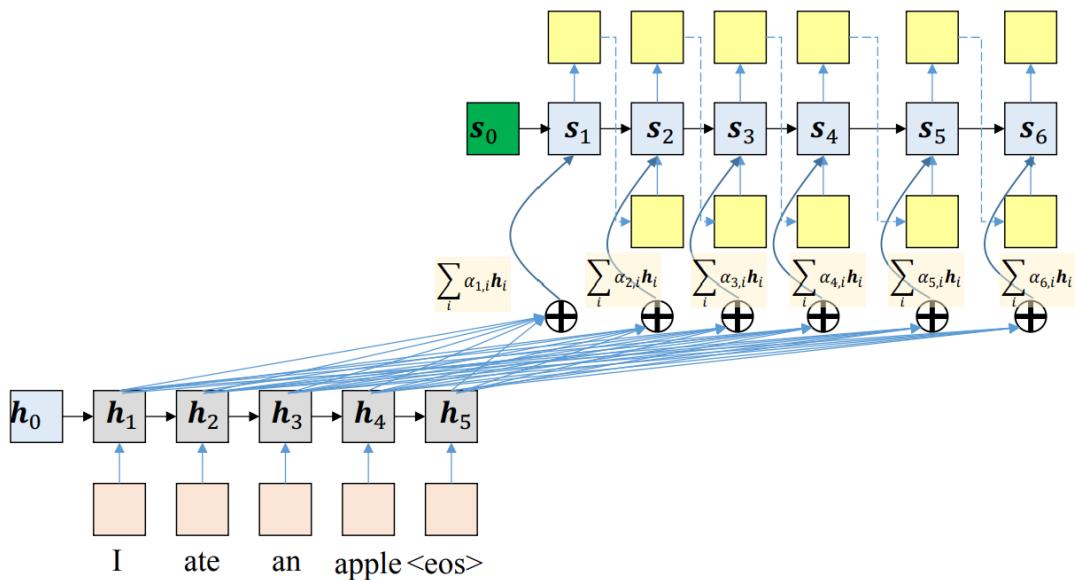


شکل ۱۱۴-۴ : گراف معماری کدگذار و کدگشا [۱۹]

در [۵۰] بحث شده است که یک رویکرد برای جلوگیری از سریز اطلاعات و از دست رفتن اطلاعات مربوط به توالی ورودی توجه به بخش های خاصی از دنباله ورودی برای بدست آوردن هر خروجی شبکه است بنابراین نیازی به در نظر گرفتن تمام دنباله طولانی ورودی نیست و صرفا برای بدست آوردن هر خروجی به بخش های خاصی از دنباله ورودی توجه می شود .

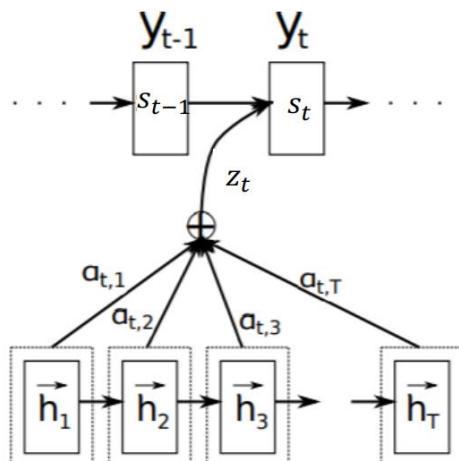


شکل ۱۱۵-۴ : بدست آوردن توزیع بر روی کلمات ورودی [۱۹]



شکل ۱۱۶-۴ : گراف شبکه های مبتنی بر توجه [۱۹]

بنابراین در مدل های مبتنی بر توجه با افزودن لایه توجه بین کدگذار و کدگشا در هر لحظه متفاوت از لحظه قبل به دنباله ورودی نگاه می شود و با افزایش طول دنباله ورودی ، پارامتر های شبکه به نسبت مدل هایی که مبتنی بر توجه نیستند کمتر افزایش می یابد [۵۰] .



شکل ۱۱۷-۴ : گراف شبکه های مبتنی بر توجه [۱۹]

$$e_{ti} = g(s_{t-1} \cdot h_i) \quad (4-16-1)$$

$$\alpha_{t,i} = \frac{\exp(e_{ti})}{\sum_{k=1}^T \exp(e_{tk})} \quad (4-16-2)$$

$$z_t = \sum_{i=1}^T \alpha_{t,i} h_i \quad (4-16-3)$$

h_i : حالت i ام لایه مخفی کدگذار

s_{t-1} : حالت لحظه $t-1$ کدگذار

$\alpha_{t,i}$: وزن مربوط به لحظه t دنباله خروجی و حالت i ام لایه مخفی

e_{ti} : امتیاز مربوط به حالت i ام لایه مخفی و لحظه t دنباله خروجی

z_t : بردار متن مربوط به لحظه t دنباله خروجی

برای محاسبه e_{ti} ابتدا تابع g به روش های متفاوتی تعریف می شود [۵۰] :

$$g(s_{t-1}, h_i) = h_i^T s_{t-1} \quad (4-16-4)$$

$$g(s_{t-1}, h_i) = h_i^T W_g s_{t-1} \quad (4-16-5)$$

$$g(s_{t-1}, h_i) = MLP([h_i, s_{t-1}]) \quad (4-16-6)$$

در رابطه (4-16-4) با محاسبه ضرب داخلی دو بردار s_{t-1} را با تک تک h_i ها مقایسه می کنیم و از این مقایسه میزان توجه به هر h_i محاسبه می شود [۵۲] و [۵۰].

در رابطه (4-16-5) در صورتی که دو بردار h_i^T و s_{t-1} دارای ابعاد یکسان نباشند از ماتریس W_g برای یکسان سازی ابعاد استفاده می شود و دو ماتریس s_{t-1} و h_i^T را به فضا مشترک می برد.

در رابطه (4-16-6) از یک شبکه عصبی چند لایه برای مقایسه s_{t-1} و h_i استفاده می شود و با یادگیری وزن های مربوط به شبکه و یافتن شباهت های s_{t-1} و h_i ، میزان توجه به هر حالت لایه مخفی بدست می آید [۵۰].

بعد از محاسبه مقادیر e_{ti} که نشان دهنده امتیاز هر یک از گره های لایه مخفی با توجه به مقدار امتیاز هر گره وزن مربوط به آن ($\alpha_{t,i}$) محاسبه می شود و سپس با یافتن وزن هر گره مخفی، بردار متن (z_t) با استفاده از جمع وزن دار هر حالت های مخفی بدست می آید .

انواع مدل های توجه [۵۱] :

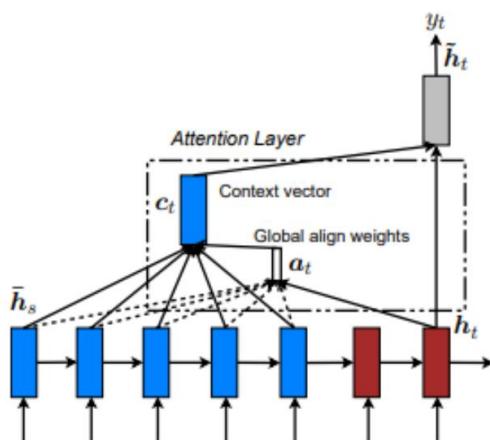
۱- مدل توجه کلی و محلی

۲- مدل توجه سخت و نرم

۳- مدل های خود-توجه

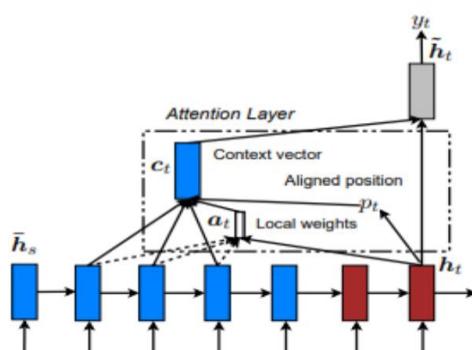
مدل های توجه کلی و محلی :

در مدل های توجه کلی تمام توالی ورودی و گره های لایه مخفی کدگذار و تمام حالات کدگشا برای محاسبه خروجی مدل استفاده می شود و برای همه توالی ورودی و گره های لایه مخفی وزن در نظر گرفته می شود[۵۱].



شکل ۱۱۸-۴ : گراف شبکه های مبتنی بر توجه کلی [۵۱]

در مدل های توجه محلی برخلاف مدل های توجه کلی تنها به مکان خاصی از کدگذار توجه و وزن داده می شود و با تشکیل بردار متن برای مکان های مورد توجه این بردار به عنوان ورودی به کدگشا وارد می شود و خروجی شبکه محاسبه می شود[۵۱].



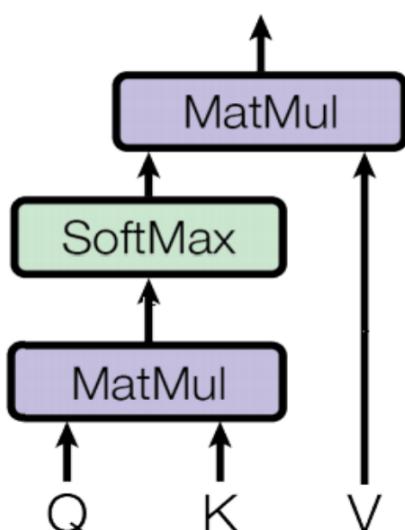
شکل ۱۱۹-۴ : گراف شبکه های مبتنی بر توجه محلی [۵۱]

مدل های توجه نرم و سخت :

در مدل های توجه نرم بردار متن به واسطه یک جمع وزن دار از حالت های مخفی کدگذار تشکیل می شود، ولی در مدل های توجه سخت به جای محاسبه جمع وزن دار همه حالت های مخفی از امتیاز توجه برای انتخاب تنها یک حالت مخفی استفاده می شود [۵۳] .

۴-۱۶ مدل های خود-توجه^۱ :

در مدل های خود-توجه ارتباط موقعیت های مختلف یک توالی ورودی را بررسی می کند به عبارت ساده تر فرایند توجه به خود به ورودی ها اجازه می دهد تا با یکدیگر تعامل داشته باشند و دریابند که به کدام موقعیت ها باید بیشتر توجه شود و خروجی با استفاده از این تعاملات و امتیاز موقعیت های مورد توجه بدست می آید .



شکل ۱۲۰-۴ : بلوک پایه لایه خود توجه [۵۲]

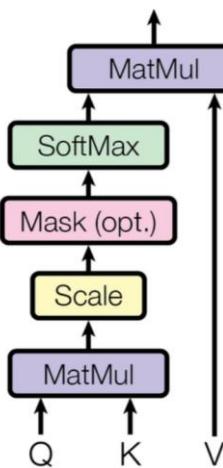
همانطور که مشاهده می شود بلوک پایه لایه خود-توجه دارای سه ورودی پرسش ، کلید و مقدار است که می توان پرسش ها را معادل حالت های بلوک کدگشا در مدل توجه نرم در نظر گرفت و کلید ها و مقدار آن ها معادل حالت های مخفی بلوک کدگذار است که هر کدام از این مقادیر از روی بردار کلمه ورودی بدست آورده می شود [۵۴] .

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T)V \quad (4-16-7)$$

با ضرب داخلی مقادیر ماتریس پرسش در ماتریس کلید همانند مدل های توجه نرم توزیع توجه بر روی توالی ورودی بدست می آید و با ضرب در ماتریس مقادیر ، میزان توجه بدست می آید .

¹ Self-Attention

Scaled Dot-Product Attention

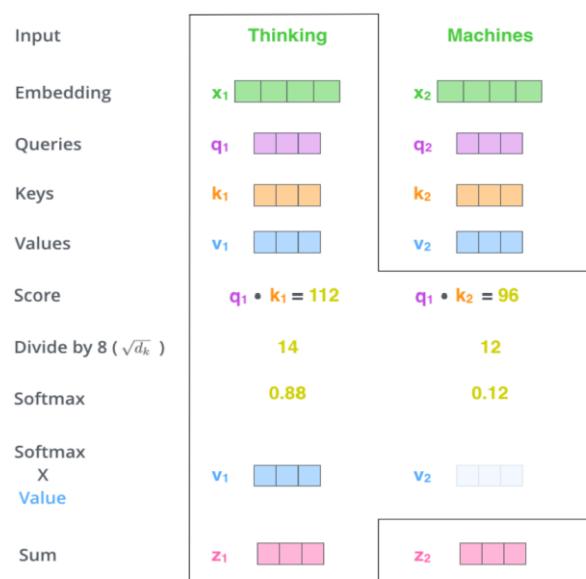


شکل ۱۲۱-۴ : بلوک پایه لایه خود توجه [۵۲]

در نهایت با افزودن بلوک مقیاس، نرمالسازی بر روی خروجی بلوک ضرب انجام می شود و از افزایش بیش از حد مقادیر ضرب داخلی ماتریس پرسش و کلید ، در نتیجه افزایش بیش از حد امتیاز بخش های خاصی از توالی ورودی جلوگیری می شود [۵۴].

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V \quad (4-16-8)$$

: اندازه بردار کلید d_k



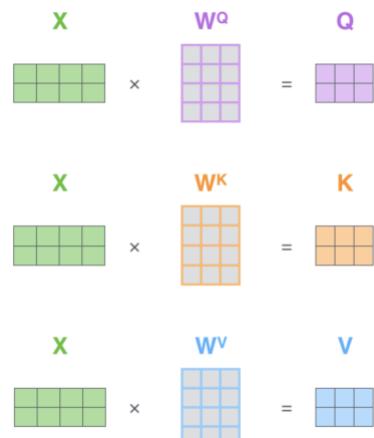
شکل ۱۲۲-۴ : نمونه محاسبات انجام شده در لایه خود توجه [۵۵]

در مدل های خود توجه با دریافت بردار کلمات ورودی ، بردار های پرسش ، کلید و مقدار با توجه به وزن های مربوط به خود که در طول فرایند آموزش یادگرفته می شود محاسبه می شوند [۵۵].

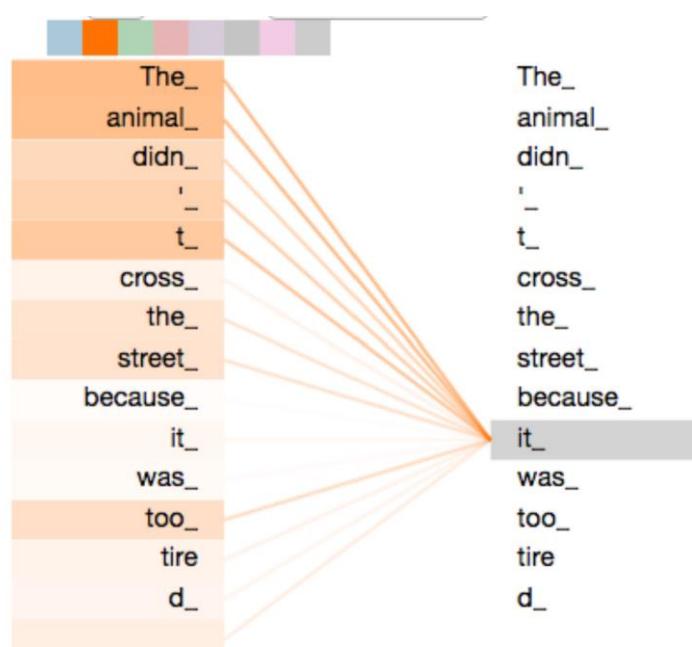
$$Q_i = W_Q X_i \quad (\text{Eq. 1})$$

$$K_1 \ldots K_n = W_K X_1 \ldots W_K X_n \quad (\text{F-1 F-1 } \cdot)$$

$$V_1 \dots V_n = W_V X_1 \dots W_V X_n \quad (\text{Fact 11})$$



^۴شکا ۱۲۳-۴: محاسبه ماتریس‌های کلید، مقدار و پیش، در لایه خودتوجه [۵۵]

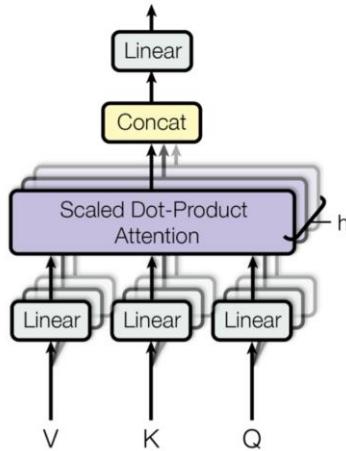


شکا، ۱۲۴-۴ : مدلسازی توجه به احیای حمله [۵۵]

همانطور که مشاهده می شود با محاسبه ضرب داخلی بردار پرسش کلمه مورد نظر با سایر کلمات موجود در جمله و محاسبه تابع توجه می توان ارتباط ضمیر با مرجع ضمیر و انواع روابط موجود در یک توالی را بدست آورد .

۴-۱۶-۴ مدل های توجه چندسر^۱ :

Multi-Head Attention



شکل ۴-۱۲۵ : مدل های توجه چندسر [۵۲]

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V \quad (4-16-12)$$

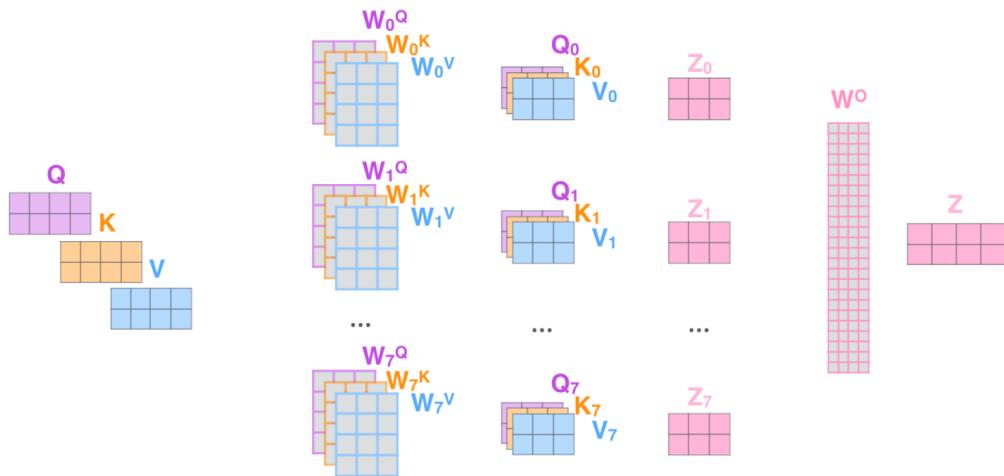
$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \quad (4-16-13)$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \quad (4-16-14)$$

مدل توجه چندسر که از کنار هم قرار دادن تعدادی لایه خودتوجه تشکیل شده است با استفاده از بلوک های خطی ماتریس وزن های W_i^Q و W_i^K و W_i^V که به ترتیب مربوط به سر اام ورودی های پرسش ، کلید و مقدار است در طول فرآیند آموزش یادگرفته می شوند [۵۴] .

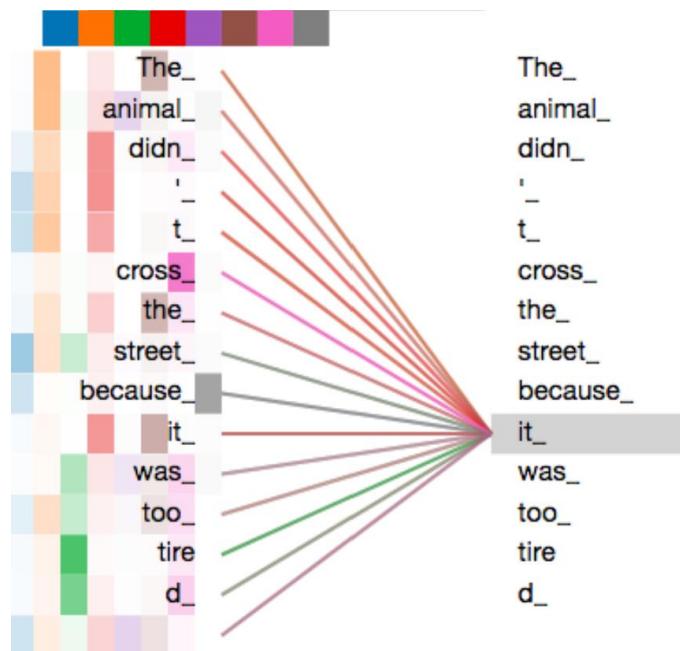
در (۴-۱۶-۱۳) برای هر کدام از سر های توجه به صورت موازی مقدار توجه و موقعیت مورد توجه محاسبه می شود و در (۴-۱۶-۱۴) با کنار هم قرار دادن ماتریس خروجی لایه های توجه مربوط به هر سر ، ماتریس خروجی لایه توجه چند سر بدست می آید و با اعمال به لایه خطی ماتریس وزن های W^O در خروجی مدل چند سر ضرب می شود که باعث می شود خروجی با ابعاد ماتریس ورودی که حاصل کنار هم قرار دادن بردار کلمات ورودی است برابر شود . [۵۴]

¹ Multi-Head Attention

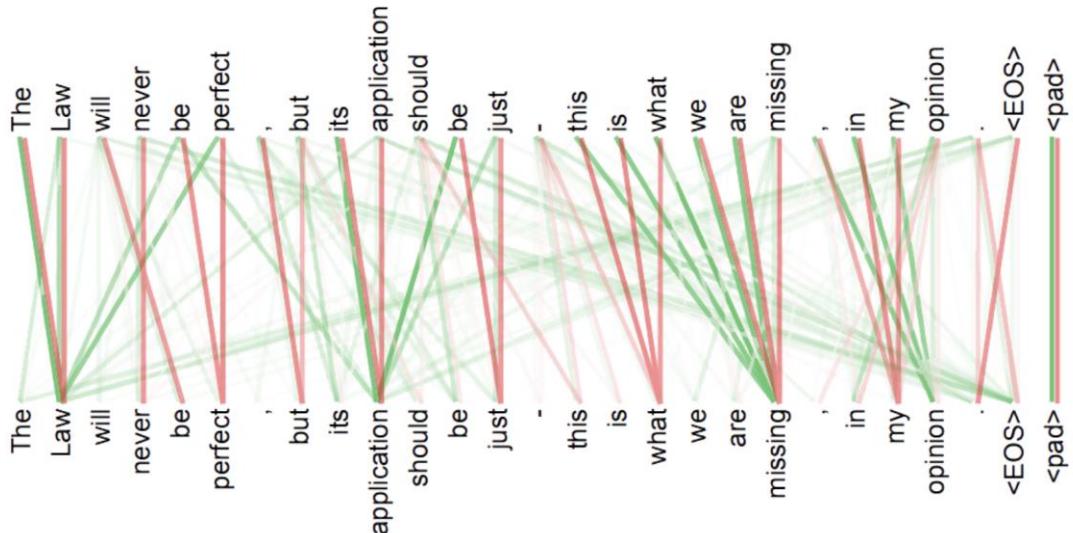


شکل ۱۲۶-۴ : محاسبه ماتریس خروجی در لایه توجه [۵۵]

بنابراین با ارائه چند مدل توجه در توجه چندسر ، توالی ورودی به زیر فضا های متفاوت می رود و نمایش ها و رویکرد های متفاوت برای توجه به توالی ورودی ارائه می شود [۵۵].



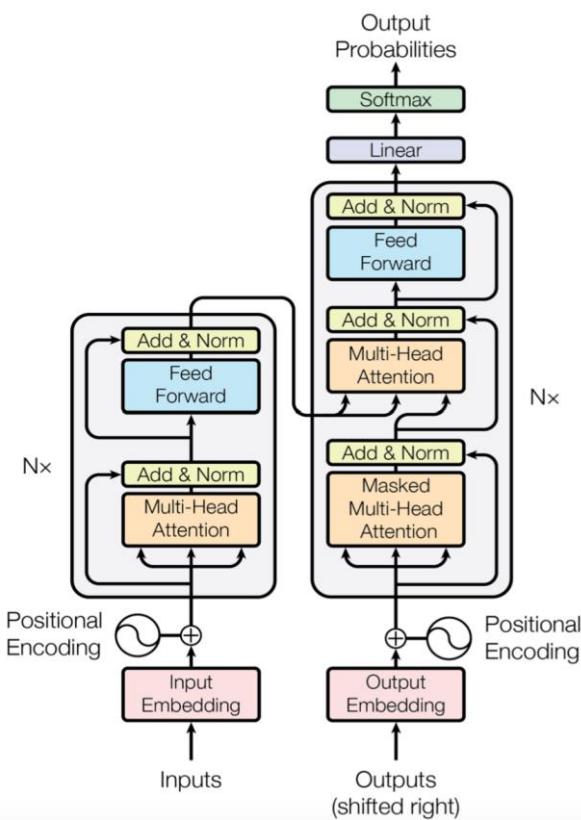
شکل ۱۲۷-۴ : مدل های توجه چندسر [۱۹]



شکل ۱۲۸-۴ : توجه به اجزا مختلف جمله در لایه توجه [۱۹]

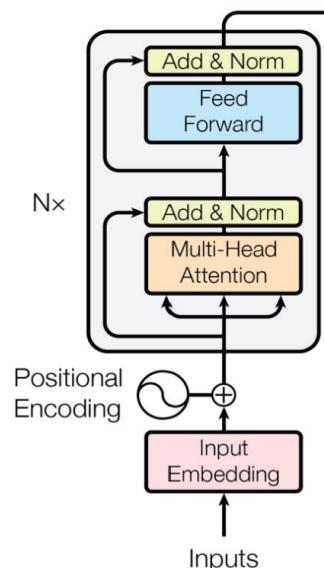
همانطور که مشاهده می شود اگر توالی ورودی متن باشد با استفاده از مدل های توجه چندسر می توان انواع روابط میان اجزا جمله را استخراج کرد همانند ارتباط بین ضمایر و مرجع آن ها ، موصوف و صفت ، فاعل و فعل جملات .

۱۶-۴-۳ شبکه های مبدل^۱:



شکل ۴-۱۲۹: شبکه مبدل [۵۲]

شبکه های مبدل شامل دو بلوک کلی کدگذار و کدگشا می باشد ابتدا به بررسی بلوک کدگذار می پردازیم .



شکل ۴-۱۳۰: بلوک کدگذار شبکه مبدل [۵۲]

¹ Transformer Networks

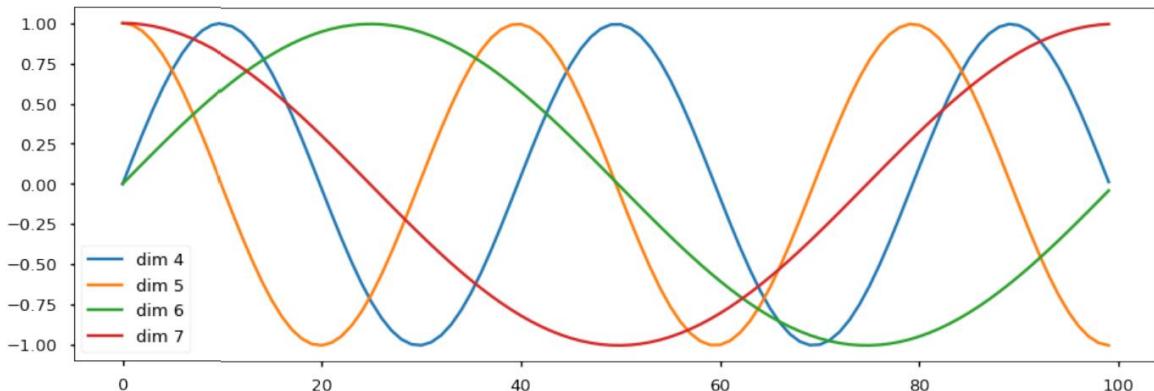
همانطور که مشاهده می شود ابتدا بردارهای بازنمایی مربوط به توالی ورودی که می تواند بردارهای حاصل از بازنمایی کلمات یک جمله باشد محاسبه می شوند و با بردار های حاصل از بلوک کدگذار موقعیت، جمع می شوند .

بلوک کدگذار موقعیت مکان هر یک از داده های ورودی در یک توالی را کدگذاری می کند که روش پیشنهاد شده در [۵۴] به صورت مجموعه ای از توابع سینوس و کسینوس با فرکانس متفاوت (متعماد) است و مقدار هر موقعیت با استفاده از روابط زیر بدست می آیند .

$$PE_{(pos.2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \quad (4-16-15)$$

$$PE_{(pos.2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \quad (4-16-16)$$

: اندازه بردار بازنمایی مربوط به داده مشخص در یک توالی ورودی

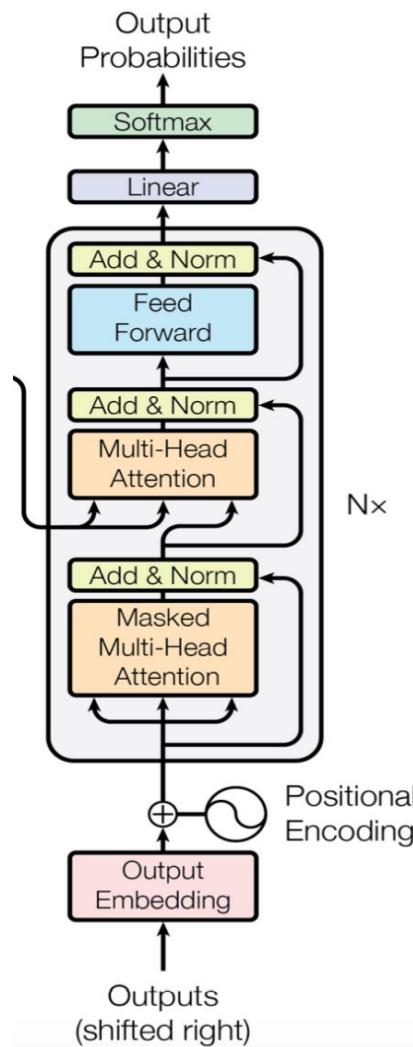


شکل ۱۳۱-۴ : مجموعه ای توابع سینوس و کسینوس خروجی کدگذار موقعیت [۱۹]

بعد از محاسبه مجموع بردارهای بازنمایی ورودی و موقعیت آن ها نتیجه وارد بلوک توجه چندسر می شود که شامل ۸ لایه خودتوجه است و ماتریس های پرسش ، کلید و مقدار محاسبه می شوند و خروجی بلوک توجه چندسر که شامل ماتریس حاصل از بردار های توجه مربوط به هر کدام از داده های توالی ورودی است برای نرمالسازی مقادیر و عملیات جمع با بردار های بازنمایی ورودی بلوک توجه چند-سر به بلوک نرمالسازی و جمع وارد می شوند .

یکی از دلایل استفاده از بلوک نرمالسازی و جمع با ورودی های بلوک توجه چندسر و استفاده از اتصالات پرشی علاوه بر کمک به آموزش سریع تر شبکه از مشکل محوشوندگی گرادیان است که به علت عدم اعمال توابع فعالسازی غیرخطی در این نوع اتصالات ، از محو شدگی گرادیان جلوگیری می شود و علاوه بر آن همواره نسخه اصلی ماتریس بازنمایی ورودی در دسترس شبکه خواهد بود و آن ها در طول فرآیند آموزش فراموش نخواهد کرد .

در نهایت خروجی بلوک نرمالسازی و جمع وارد یک شبکه عصبی چند لایه پرسپترون می شود و به عنوان یک درجه آزادی بیشتر به شبکه ، وزن های این شبکه عصبی چند لایه به گونه ای یادگرفته می شوند تا به عنوان تبدیلی غیرخطی برای فهم محتوا و ارتباط بین کلمات با نقش های متفاوت در جمله به بهترین شکل اطلاعات را به کدگشا انتقال دهند .

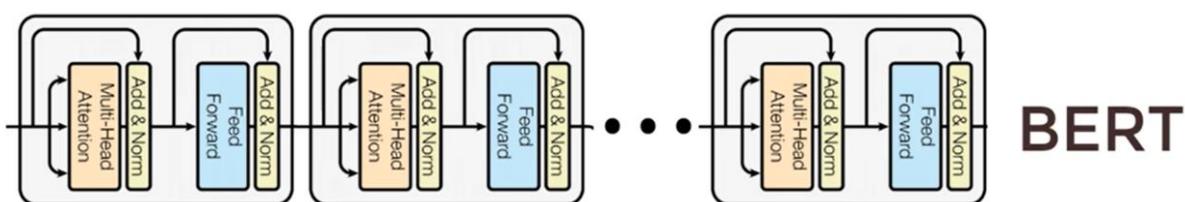


شکل ۱۳۲-۴ : بلوک کدگشا شبکه مبدل [۵۲]

در بلوک کدگشا ابتدا بردار بازنمایی خروجی توالی ورودی به کدگذار و همچنین بازنمایی مکانی آن ها بدست می آید و این دو ماتریس به صورت عنصر به عنصر با یکدیگر جمع می شوند و همانند بلوک کدگذار به عنوان پرسش ، کلید و مقدار وارد بلوک توجه چندسر می شوند با این تفاوت که در این بلوک بر خلاف بلوک به کار رفته در کدگذار تمام خروجی به بلوک توجه چندسر نشان داده نمی شود و تنها خروجی هایی که ورودی مناسب با آن ها در مرحله کدگذاری به شبکه وارد شده اند می توانند وارد لایه توجه بشوند به این علت که بر خلاف معماری های قبلی در شبکه های دارای لایه خودتوجه اگر تمام خروجی به صورت یک جا به شبکه داده شود لایه خودتوجه به تمام دنباله خروجی امتیاز می دهد و شبکه از خروجی های دارای برچسب نیز برای پیش بینی خروجی نهایی مدل استفاده می کند [۵۵] .

در نهایت در بلوک بعدی همانند بلوک کدگذار عملیات نرمالسازی و جمع با اتصالات پرشی انجام می شود و خروجی این بلوک به عنوان ماتریس مقدار و خروجی های بلوک کدگذار به عنوان ماتریس پرسش و کلید وارد بلوک توجه چندسر می شوند و با استفاده از اطلاعات استخراج شده از ورودی شبکه در ماتریس های کلید و پرسش شبکه آموزش می بینند که چگونه ارزش های جدید را که در ماتریس مقادیر از طریق توالی خروجی بدست آمده اند محقق کند و در نهایت با اعمال یک شبکه عصبی چند لایه به خروجی تمام ارتباطات غیرخطی موجود در نظر گرفته می شود و خروجی پیش بینی می شود [۱۹].

۴-۱۶-۴ شبکه های کدگذار دوطرفه برمبنا مبدل ها^۱ : (BERT)



شکل ۱۳۳-۴ : شبکه کدگذار دوطرفه مبتنی بر مبدل ها [۵۷]

این نوع شبکه ها از کنار هم قرار دادن چندین بلوک کدگذار مبدل ها تشکیل می شوند و با توجه آن چه در بخش بلوک کدگذار مبدل ها بحث شد این بلوک ها با استفاده از لایه های خودتوجه به تعامل بین عناصر توالی ورودی توجه می کنند و تمام توالی ورودی را به صورت یک جا به عنوان ورودی دریافت می کنند و در طول فرآیند آموزش وزن های شبکه بلوک کدگذار به گونه ای بهینه می شوند که ارتباطات بین بخش های مختلف توالی ورودی را کشف و محتوا آن را استخراج کنند.

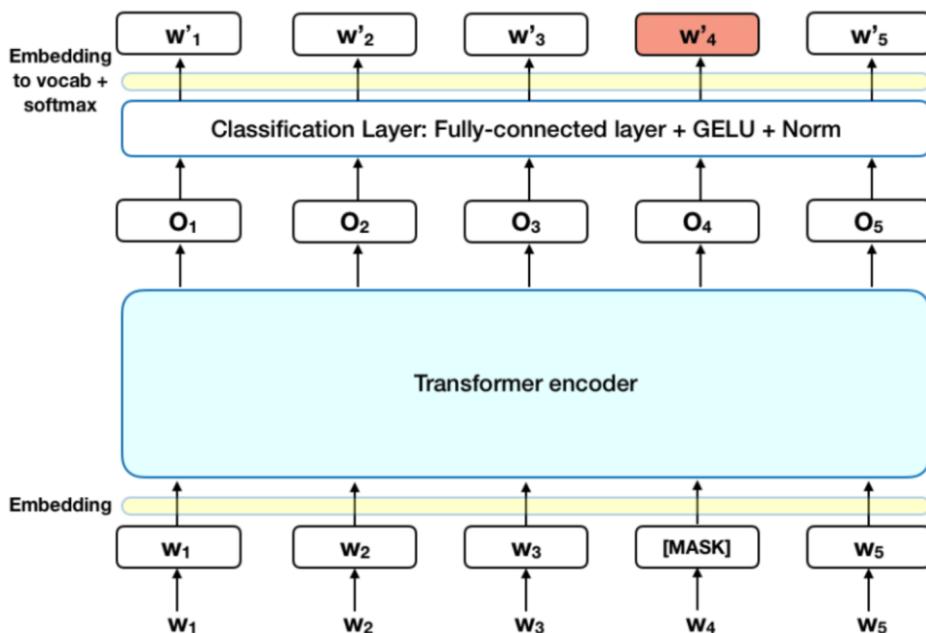
با در کنار هم قرار دادن بلوک های کدگذار مبدل ها به صورت پشتی و عمیق شدن شبکه های کدگذار دو طرفه ویژگی های پیچیده تر از توالی ورودی شبکه استخراج می شود و در کاربرد پردازش زبان های طبیعی با اعمالی توالی هایی از کلمات می توان از این نوع شبکه ها برای دستیابی به یک مدل زبانی استفاده کرد [۵۶].

یکی از دلایل نامگذاری این نوع از شبکه ها به عنوان شبکه های کدگذار دو طرفه این است که توالی ورودی به صورت کامل به لایه خودتوجه بلوک های کدگذار وارد می شود و لایه خودتوجه تشخیص می دهد که به کدام قسمت توالی ورودی توجه کند بنابراین مانند شبکه های عصبی بازگشتی لزوماً توالی ورودی از چپ به راست و از راست به چپ به شبکه داده نمی شود بلکه دو طرفه بودن یکی از ویژگی های ذاتی لایه های خودتوجه است و برای بدست آوردن روابط و محتوا توالی ورودی همزمان به تمام قسمت های آن توجه می کنند.

¹ Bidirectional Encoder Representation from Transformers

به طور کلی از دو روش به صورت هم زمان برای آموزش این نوع شبکه ها استفاده می شود که اگر توالی های ورودی را در حوزه پردازش زبان طبیعی توالی از کلمات در نظر بگیریم شامل دو روش یادگیری بدون ناظر به این معنا که مجموعه داده های ورودی به مدل برای آموزش برچسب گذاری نشده اند و خروجی مدل برای داده های آموزشی از قبل تعیین نشده است^[۵۷].

دو روش مدل زبانی پنهان شده^۱ و پیش بینی جمله بعدی^۲، دو روش استفاده شده در آموزش این نوع شبکه ها است^{[۵۶] و [۵۷]}.



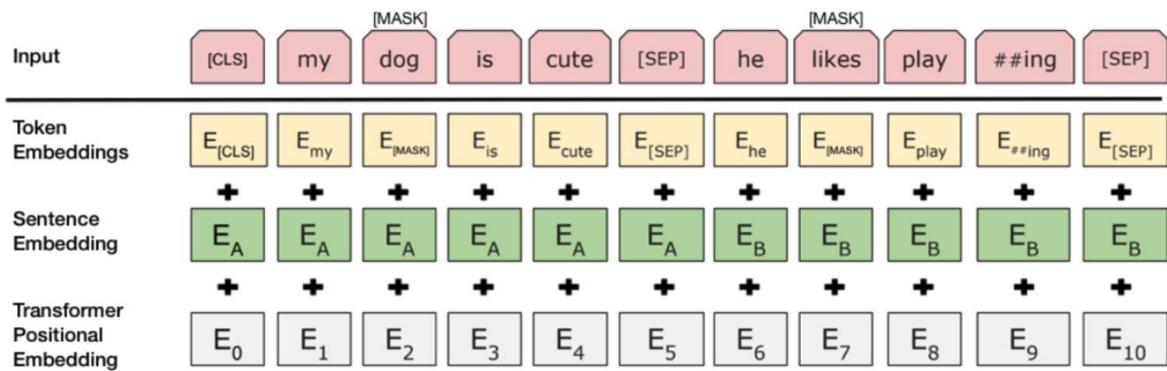
شکل ۱۳۵-۴ : روش مدل زبانی پنهان شده [۵۷]

در روش مدل زبانی پنهان شده توالی ورودی که می تواند توالی از کلمات باشد به شبکه وارد می شود با این تفاوت که برخی از کلمات توالی به صورت تصادفی پنهان شده اند و شبکه باید کلمات پنهان شده را پیش بینی کند و به شبکه کمک می کند با یافتن محتوا در سطح کلمات ارتباطات معنا دار بین کلمات برقرار کند^{[۵۶] و [۵۷]}.

در روش پیش بینی جمله بعد که به صورت هم زمان با روش مدل زبانی پنهان شده برای آموزش این نوع شبکه ها استفاده می شود دو جمله به عنوان ورودی به شبکه وارد می شود و شبکه باید سعی کند ارتباطی معنا دار میان آن دو جمله برقرار کند و تشخیص دهد که جمله دوم در راستا جمله اول است یا خیر ، بنابراین در این مرحله شبکه سعی می کند ارتباطات در سطح جملات را تشخیص بدهد و ترکیب دو روش ذکر شده کمک می کند شبکه بهترین مدل زبانی را از طریق توالی های ورودی استخراج کند^[۵۷].

¹ Masked language model

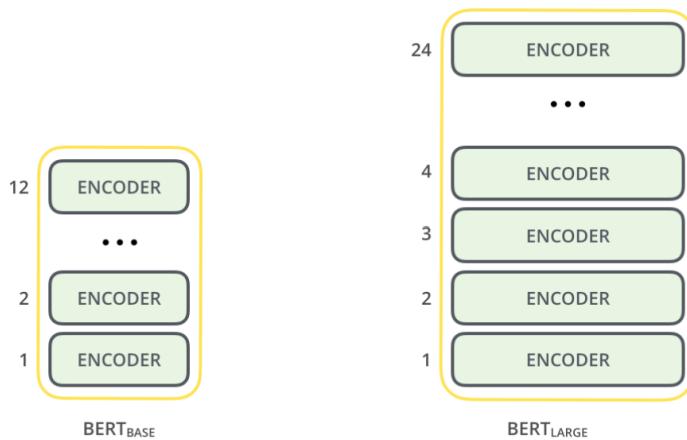
² Next sentence prediction



شکل ۱۳۶-۴ : بازنمایی کلمات ورودی [۵۲]

نشانه [CLS] برای نشان دادن ابتداء اولین جمله ورودی و نشانه [SEP] برای نشان دادن پایان جمله استفاده می شود و بردار بازنمایی ورودی شبکه حاصل جمع سه بردار بازنمایی کلمات ، موقعیت و شماره جمله است [۵۶] و [۵۷].

در [۵۶] دو نوع شبکه کدگذار دو طرفه مبتنی بر مبدل ها معرفی شده است که نوع اول آن شامل ۱۲ لایه کدگذار مبدل و ۱۱۰ میلیون پارامتر و نوع دوم آن شامل ۲۴ لایه و ۳۴۰ میلیون پارامتر می باشد که بوسیله مفهوم یادگیری انتقالی می توان از نسخه های آموزش دیده این شبکه ها برای حل مسائل مختلف استفاده کرد .



شکل ۱۳۷-۴ : دو نوع شبکه آموزش دیده کدگذار دو طرفه [۵۶]

بخش پنجم : پیش پردازش داده ها و بازنمایی کلمات

۱-۵ پیش پردازش داده ها :

پیش پردازش داده ها یکی از مهم ترین گام ها برای استفاده از مدل های یادگیری عمیق است و نقش مهمی را در عملکرد مدل ایفا می کند.

در مدل های پردازش زبان طبیعی داده های ورودی معمولاً متن هستند و برای پیش پردازش این نوع از داده های ورودی و آماده سازی آن ها به گونه ای که بتوان آن ها را به عنوان ورودی به مدل های یادگیری عمیق داد مراحلی طی می شود.

یکسان سازی تمام حروف به حروف کوچک :

با توجه به اینکه در یک متن ممکن است کلماتی با حروف بزرگ به کار رفته باشند و در صورت وجود کلمات یکسان اما متفاوت از لحاظ نوشتار مدل های یادگیری عمیق آن ها را به عنوان دو کلمه متفاوت در نظر می گیرند برای بهبود عملکرد مدل تمام حروف به کار رفته در مدل به حروف کوچک می شود.^[۶۰]

نشانه سازی^۱:

در مرحله نشانه سازی که می تواند در سطح جمله و یا کلمه باشد ، متن ورودی به بخش های کوچک تر تقسیم می شود و اگر نشانه سازی در سطح کلمه باشد تمام کلمات متن به صورت جداگانه در یک لیست از کلمات قرار می گیرند.^[۶۱]

حذف علائم نگارشی :

در این مرحله علائم نگارشی به کار رفته در لیست حاصل از نشانه سازی حذف می شود و حاصل آن لیستی تنها از کلمات و یا جملات به کار رفته در متن خواهد بود.^[۶۰]

حذف کلمات بدون بار معنایی :

در هر زبانی کلماتی وجود دارند که بار ها در یک متن به کار می روند اما بار معنایی به جملات اضافه نمی کنند و معمولاً کلماتی هستند که مربوط به رعایت قواعد نحوی زبان مورد نظر هستند که با حذف آن ها می توان لغات موجود در لیست نشانه ها را کاهش داد.^[۶۰]

ریشه یابی کلمات با تقطیع^۲:

در این مرحله هر یک از کلمات ریشه یابی می شوند و ریشه آن ها به جای کلمات در لیست نشانه ها قرار می گیرد و موجب یکسان سازی کلماتی می شود که ریشه یکسانی دارند . به عنوان نمونه کلمه "دیدار" بعد از اعمال ریشه یابی با تقطیع به کلمه "دید" تبدیل می شود.^[۶۱]

¹ Tokenization

² Stemming

معنایابی کلمات^۱

در مرحله ریشه یابی کلمات با تقطیع ممکن است برخی کلمات به گونه ای قطع شده باشند که کلمات حاصل کلماتی بدون معنا در زبان مورد نظر باشند ، درنتیجه در مرحله معنایابی سعی می شود کلمات موجود در لیست نشانه ها به گونه ای تغییر داده شوند که بار معنای خود را بدست آورند [۶۰] .

۲-۵ بازنمایی کلمات^۲

یک رویکرد برای نمایش کلمات و اسناد متنی بازنمایی هر یک از کلمات در یک بردار است .

روش کدگذاری یک-روشن^۳ :

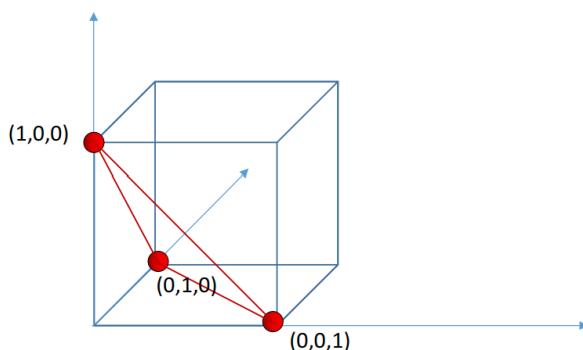
در این روش کدگذاری که برای بازنمایی کلمات نیز می توان از آن استفاده کرد برای هر کلمه یک یک بردار به طول تعداد کل کلمات متن در نظر گرفته می شود و در هر بردار تنها یک درایه منحصر بفرد مقدار یک را می گیرد و سایر درایه های آن بردار صفر می شوند به این ترتیب بازنمایی هر کلمه با کلمات دیگر متفاوت خواهد بود .

$$[0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0]$$

شکل ۱-۵ : نمونه کدگذاری یک-روشن [۱۹]

اما در این روش کدگذاری تمام بردار ها بر هم عمودند و ضرب داخلی آن ها صفر می شود و فاصله کلمات با هم مساوی در نظر گرفته می شود و در نتیجه شباهت و معنا کلمات در نظر گرفته نمی شود [۱۹] .

با توجه به چگالی کم فضای بردار ها در این روش کدگذاری و رشد نمایی تعداد نقاط، شباهت بین کلماتی که معنای مشابه دارند در نظر گرفته نمی شود [۱۹] .



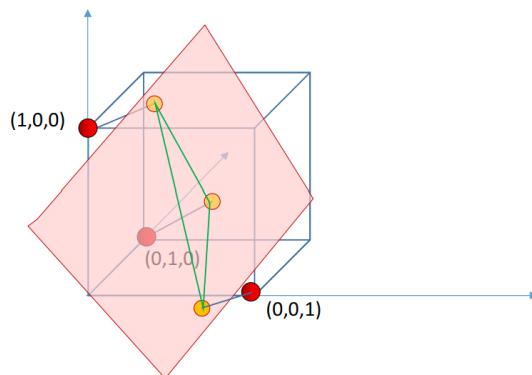
شکل ۲-۵ : نمونه فضای برداری کلمات [۱۹]

¹ Lemmatization

² Word Embedding

³ One-hot encoding

در سایر روش هایی که در ادامه بحث می شود سعی شده است با کاهش ابعاد فضا و در نتیجه کاهش چگالی فضای بردار ها و نزدیک شدن بردار ها به یکدیگر شباهت بین کلمات در بازنمایی درنظر گرفته شود.



شکل ۳-۵ : نمونه فضای برداری کلمات [۱۹]

روش کیسه کلمات^۱:

در این روش که یکی از ساده ترین روش های بازنمایی از کلمات داخل متون است با تشکیل یک ماتریس که سطر های آن جملات و ستون های آن کلمات هستند هر درایه نشان دهنده رخداد کلمه نسبت داده شده به ستون آن درایه در جمله همان سطر است و به این طریق هر درایه نشان دهنده تعداد تکرار کلمات داخل هر جمله است [۶۲].

| | 1 This | 2 movie | 3 is | 4 very | 5 scary | 6 and | 7 long | 8 not | 9 slow | 10 spooky | 11 good |
|-------------|-----------|------------|---------|-----------|------------|----------|-----------|----------|-----------|--------------|------------|
| Review 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| Review 2 | 1 | 1 | 2 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| Review 3 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |

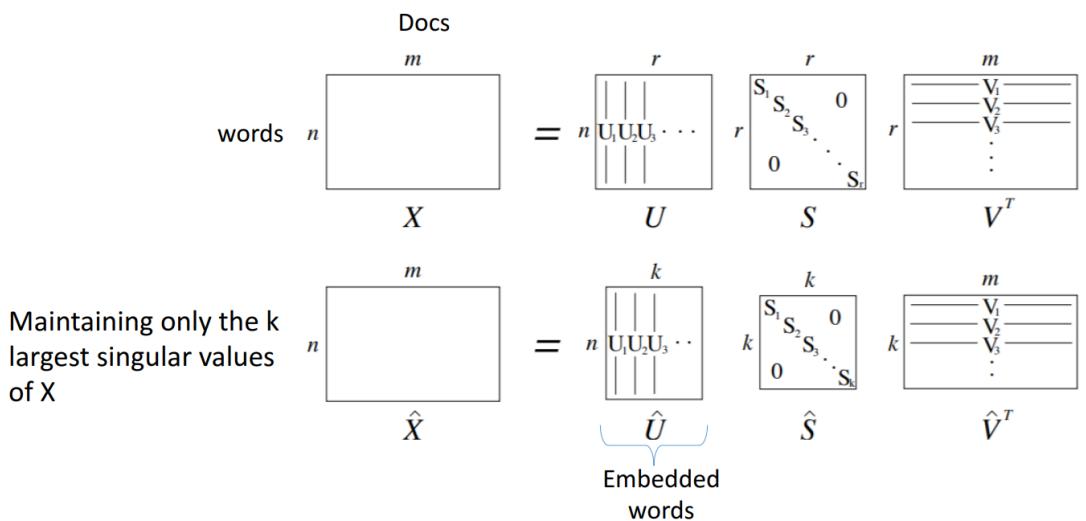
شکل ۴-۵ : نمونه روش کیسه کلمات [۶۲]

در این روش ماتریس تشکیل شده می تواند شامل تعداد زیادی صفر باشد که باعث می شود بدون ارائه اطلاعات صرفا بار محاسباتی مسئله افزایش پیدا کند و همچنین اطلاعاتی از شباهت کلمات و قواعد نحوی ارائه نمی دهد [۶۲].

روش استفاده از ماتریس هم رخدادی تمام کلمات متن :

در این روش با تشکیل ماتریس هم رخدادی که نشان دهنده تعداد تکرار هر کلمه داخل متون است و تجزیه ماتریس به مقادیر ویژه و حذف مقادیر ویژه کوچک ابعاد فضا کاهش می یابد و هر بردار ویژه یک بازنمایی برای کلمات موجود در متون ارائه می دهد [۱۹].

¹ Bag of words



شکل ۵-۵ : بدست آوردن ماتریس بازنمایی از روی ماتریس هم رخدادی [۱۹]

این روش به علت بار محاسباتی بالا برای در صورت بالا بودن تعداد متون و کلمات و هزینه بالای محاسباتی در صورت افزودن متن و یا کلمه جدید امروزه کمتر مورد استفاده قرار می گیرد.

روش کلمه به بردار^۱:

ایده این روش استفاده از خود کلمات بکار رفته در متن برای یادگیری بردار کلمات است و معنی یک کلمه و همچنین بردار آن را از روی کلمات مجاور آن تعیین می کند [۶۴].

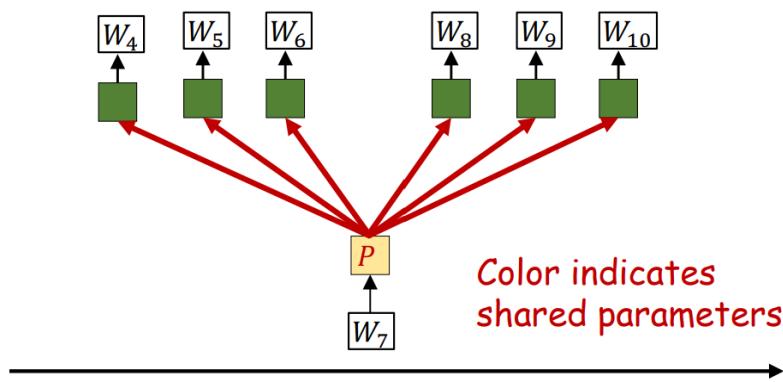
government debt problems turning into banking crises as has happened in
saying that Europe needs unified banking regulation to replace the hodgepodge

These words will represent *banking*

شکل ۶-۵ : نمونه متن و کلمه مرکزی [۱۹]

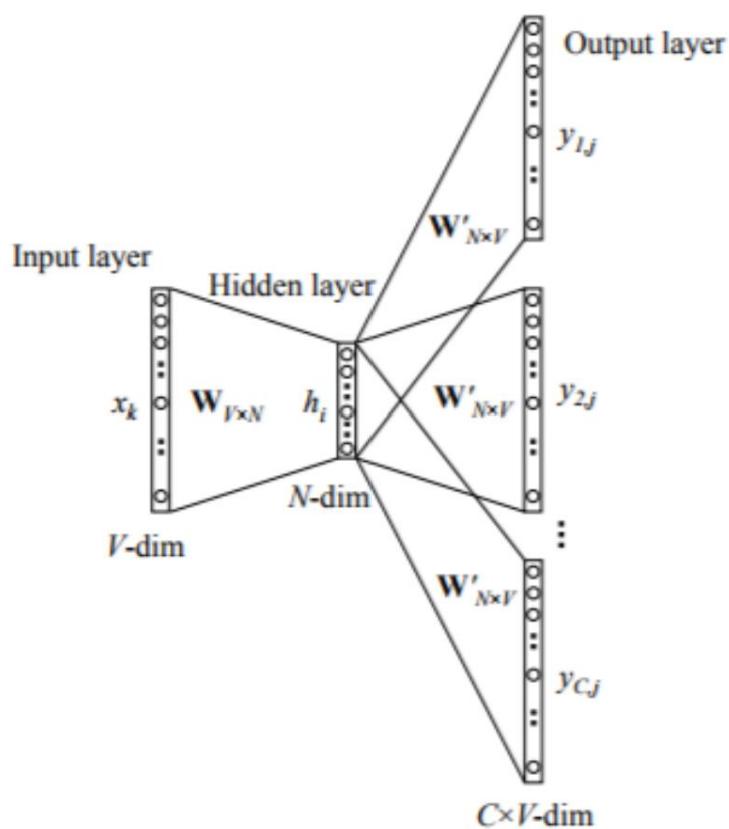
در این روش که با حجم زیاد داده ورودی می تواند بصورت خودکار بردار ها را یاد بگیرد دیگر نگرانی بابت تغییر محتوا متون وجود ندارد و با تغییر محتوا مدل سعی می کند بردار های جدیدی را برای بازنمایی بهتر کلمات تولید کند و کلمه را به گونه ای کدگذاری کند که به بهترین شکل قابل بازنمایی باشد و مدل بتواند به خوبی کلمات اطراف آن و محتوا را پیش بینی کند.

^۱ Word2vec



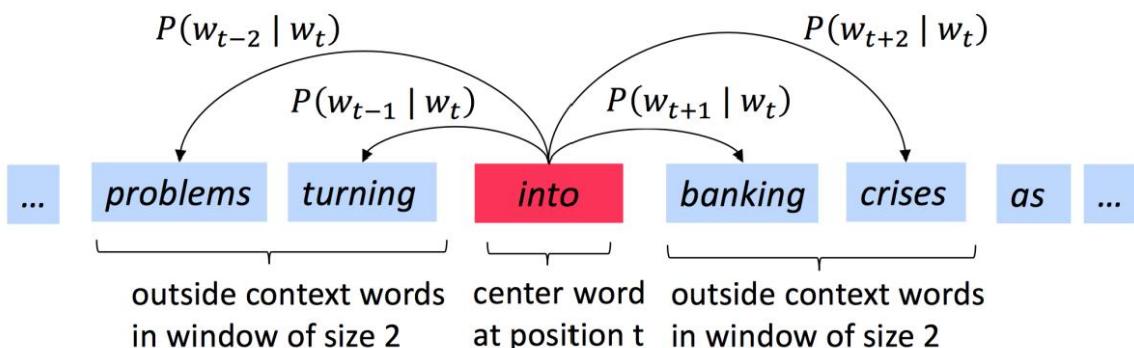
شکل ۷-۵ : مدل روش کلمه به بردار [۱۹]

با اعمال بردار کلمه مورد نظر به صورت یک-روشن به ورودی شبکه ، شبکه سعی می کند با یافتن بهترین وزن ها بهترین نمایش را با استفاده از یک تبدیل خطی و بدون اعمال توابع فعالسازی برای کلمه مورد نظر در فضایی با ابعاد کمتر ارائه دهد و در لایه خروجی احتمال کلمات مجاور یا همان محتوا را بدست می آورد [۶۵].

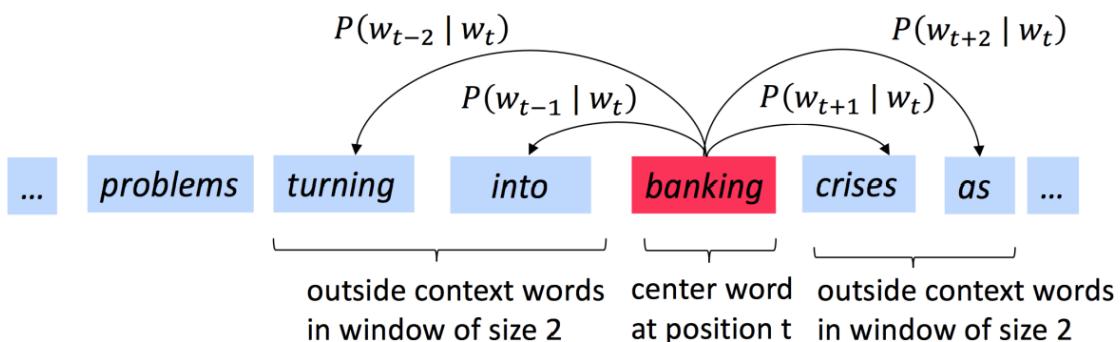


شکل ۸-۵ : مدل شبکه عصبی چند لایه روش کلمه به بردار [۱۹]

در روش کلمه به بردار پنجره ای به ابعاد مشخص در نظر گرفته می شود و کلمه ای که به دنبال بازنمایی آن هستیم به عنوان مرکز پنجره در نظر گرفته می شود و از شباهت بین بردار کلمات مجاور و کلمه مرکزی استفاده می شود تا مقدار احتمال کلمات مجاور به شرط کلمه مرکزی محاسبه شود و در هر مرحله این احتمال افزایش یابد [۶۴].



شکل ۹-۵ : نمونه روش کلمه به بردار [۱۹]



شکل ۱۰-۵ : نمونه روش کلمه به بردار [۱۹]

بنابراین تابع هدف به صورت زیر خواهد بود :

$$\prod_{t=1}^T \prod_{-m \leq j \leq m} p(w_{t+j} | w_t) \quad (5-2-1)$$

T: اندازه مجموعه داده آموزش

m: اندازه پنجره متن

w_j : بردار زامین کلمه

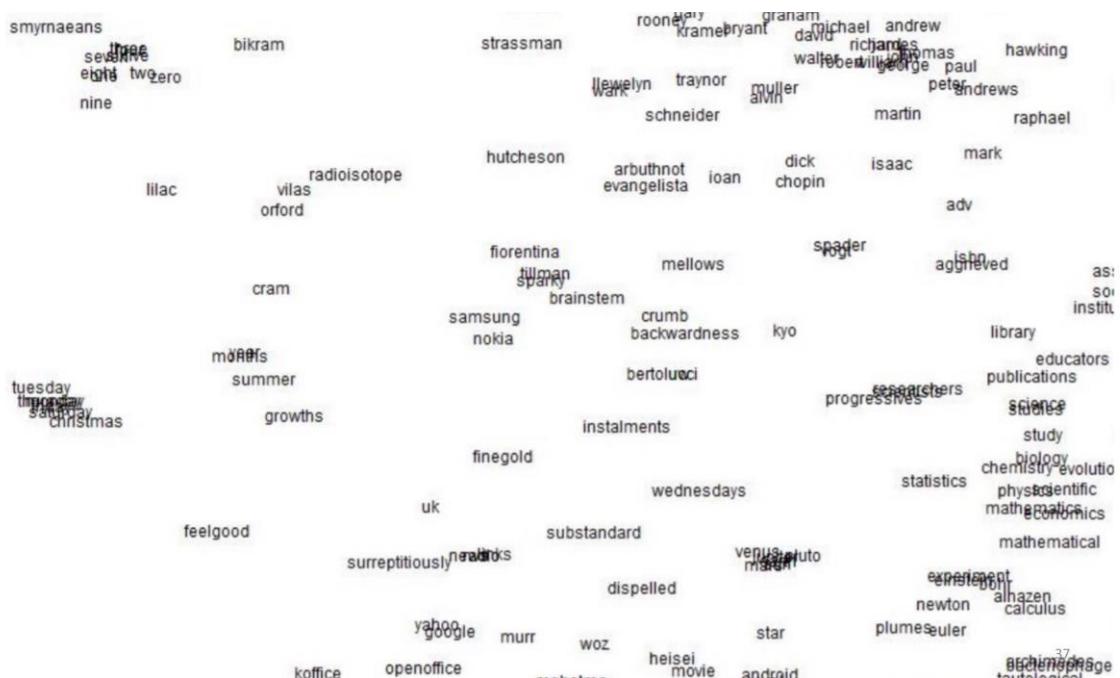
مقدار احتمال $p(w_0|w_I)$ با اعمال تابع softmax به صورت زیر با ضرب سطر های ماتریس v_w سطری از این ماتریس است زمانی که w کلمه مرکز است در ستون های ماتریس u_w ستونی از این ماتریس است زمانی که w کلمه ای در مجاورت کلمه مرکزی است بدست می آید و با ضرب داخلی بازنمایی کلمه مرکزی در کلمات موجود در همسایگی شباهت آن ها بدست می آید [۶۵].

$$p(w_O | w_I) = \frac{e^{u_O^T v_I}}{\sum_i e^{u_k^T v_I}}$$

با استفاده از اعمال تابع لگاریتم به تابع هدف و تبدیل ضرب به جمع میتوان مسئله را به صورت کمینه سازی تابع هزینه تعریف کرد.

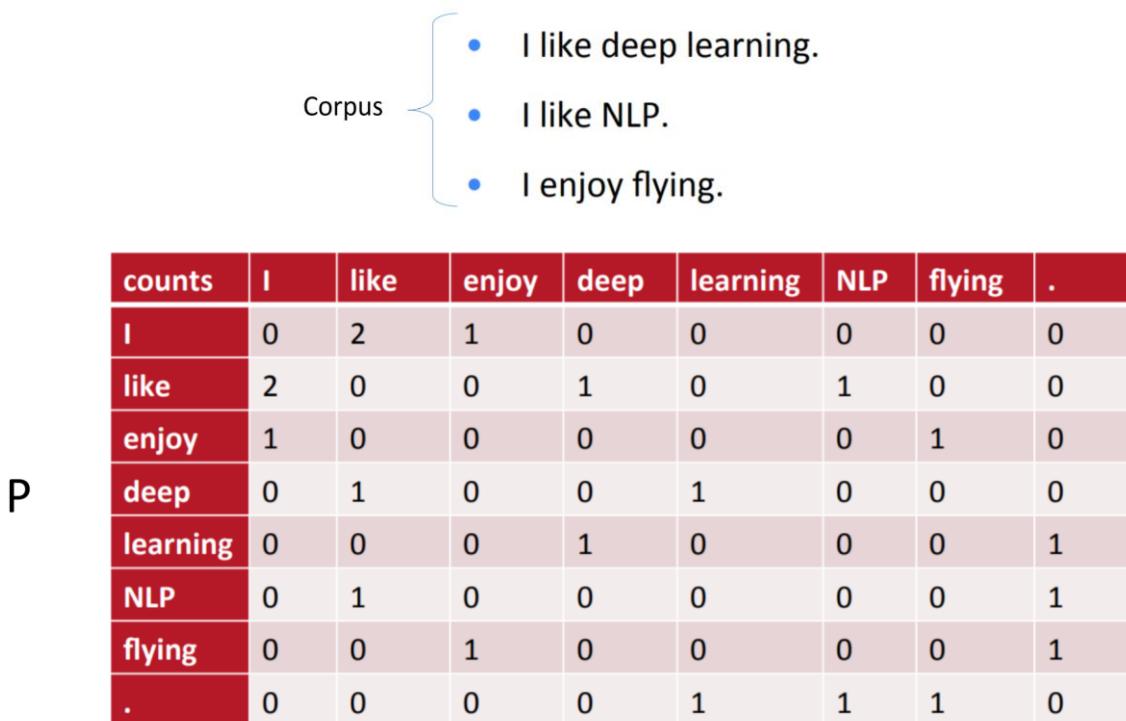
$$J(\theta) = - \sum_{t=1}^T \sum_{-m \leq j \leq m} \log p(w_{t+j} | w_t) \quad (\Delta-2-3)$$

در این روش با توجه به محدودیت اندازه پنجره در نظر گرفته شده اطلاعات به صورت محلی برای هر پنجره استخراج شده است و علاوه بر حجم محاسبات بالا با حرکت پنجره در متن هایی با تعداد کلمات زیاد، مدل بازنمایی مناسبی برای کلمات جدید و هم معنا ارائه نمی دهد [۶۵].



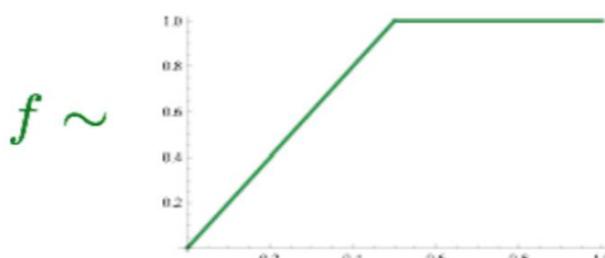
شکل ۱۱-۵: فضای پردارها در روش کلمه به پردار [۱۹]

در روش بردار های سراسری برای بازنمایی کلمات با تغییر تابع هزینه و استفاده از ماتریس هم رخدادی کلمات علاوه بر در نظر گرفتن پنجره های محلی می توان علاوه بر اطلاعات محلی هر پنجره از اطلاعات کل متن استفاده کرد و برای متن های با تعداد کلمات بالا مناسب است [۶۶].



شکل ۱۲-۵ : نمونه ماتریس هم رخدادی [۱۹]

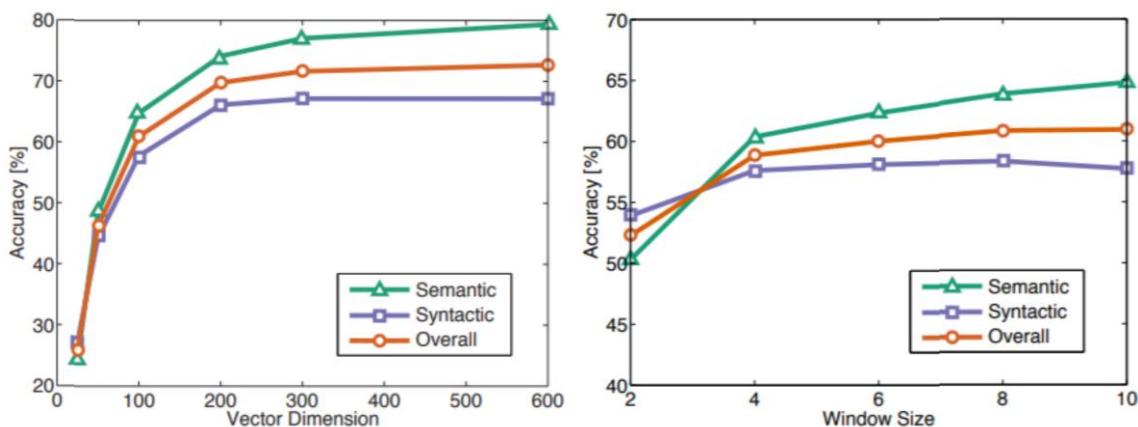
تابع هزینه بر اساس ماتریس هم رخدادی و بردار های وزن کلمه مرکزی و کلمات مجاور به صورت جمع وزن دار از تابع f تعریف می شود و تابع f به گونه ای در نظر گرفته می شود که تاثیر رخداد بیش از حد کلمات را در نظر نگیرد [۶۶].



شکل ۱۳-۵ : تابع پیشنهادی در روش Glove [۱۹]

$$J(\theta) = \frac{1}{2} \sum_{i,j=1}^W f(P_{ij})(u_i^T v_j - \log P_{ij})^2 \quad (۵-۲-۴)$$

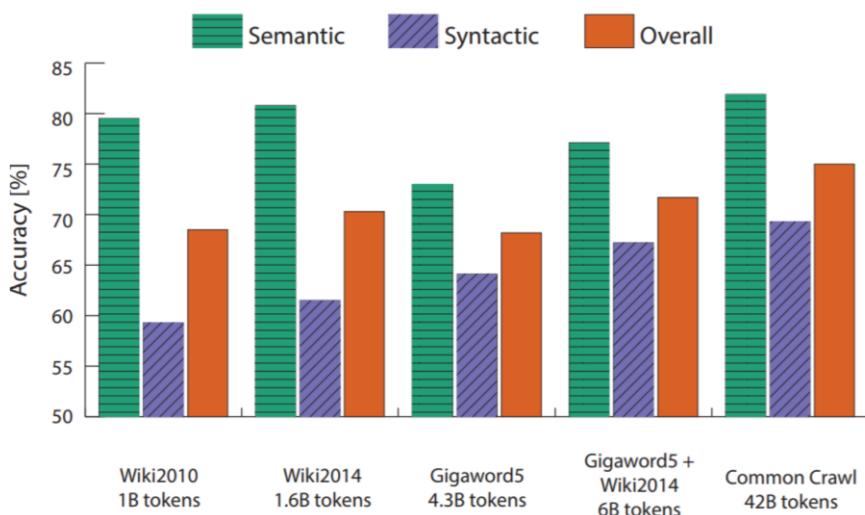
۵-۳ انتخاب ابرپارامتر های مناسب برای بازنمایی کلمات :



شکل ۱۴-۵ : بررسی تاثیر ابرپارامتر ها در بازنمایی کلمات [۶۶]

همانطور که مشاهده می شود با افزایش ابعاد بردار بازنمایی کلمات به بیش از حدودا ۳۰۰ بعد ، با توجه به اینکه تمام قواعد زبانی در تعداد ابعاد کمتری قابل یادگیری است ، دقت مدل در یادگیری قواعد نحوی تقریبا ثابت می شود ولی با توجه به اینکه با افزایش ابعاد مدل از نظر معنایی غنی تر می شود و معانی جدیدی را استخراج می کند در نتیجه دقت مدل با افزایش تعداد ابعاد همچنان افزایش دارد[۶۶] .

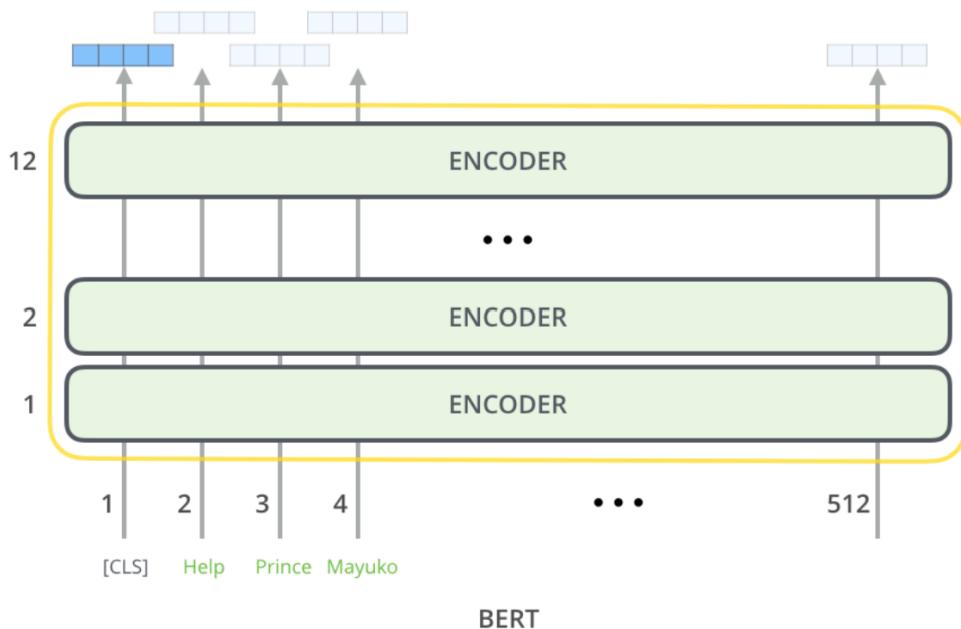
اندازه پنجره کلمات مرکزی نیز یکی از ابرپارامتر های قابل تنظیم است که همانطور که مشاهده می شود با افزایش اندازه پنجره به بیش از ۸ مدل همچنان از نظر معنایی عنی تر می شود و ارتباطات جدیدی را استخراج می کند اما از نظر یادگیری قواعد نحوی با افزایش بیش از حد پنجره همسایگی علاوه بر افزایش پیچیدگی محاسباتی مدل ، ممکن است مدل نتواند قواعد نحوی را به خوبی استخراج کند و انتخاب مناسب آن در حدود ۸ با حدود ۸ با توجه به نوع مسئله و داده های ورودی مدل می توان به دقت های بالاتری در پیش بینی دست یافت[۶۶] .



شکل ۱۵-۵ : مقایسه اثر افزایش داده های مدل در دقت یادگیری معنا و قواعد نحوی [۶۶]

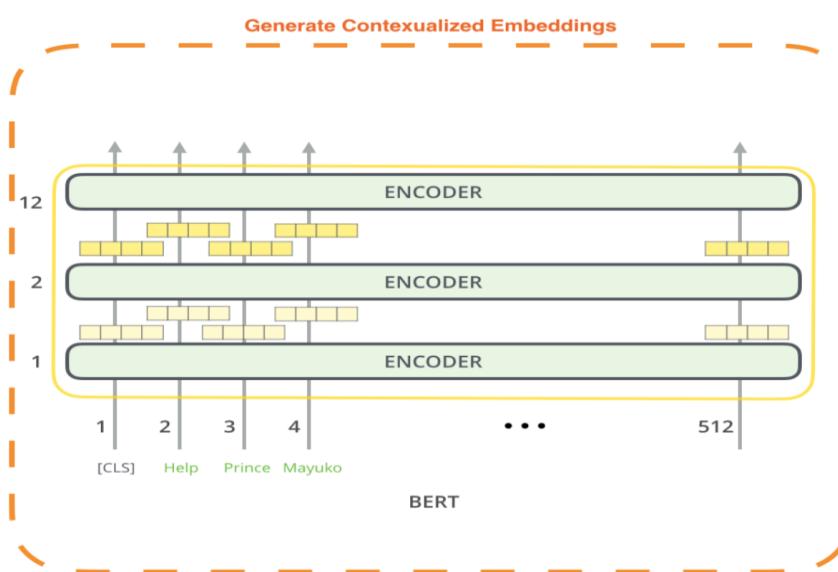
همانطور که در شکل (۱۵-۵) مشاهده می شود افزایش تعداد داده های ورودی و انتخاب منبع با توجه به نوع مسئله می تواند در مجموع باعث افزایش دقت مدل شود [۶۶].

۴-۵ شبکه های آموزش دیده دو طرفه مبتنی بر مبدل ها :

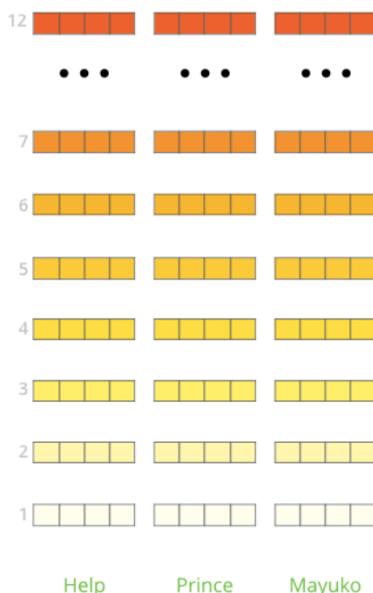


شکل ۱۶-۵ : بدست آوردن بازنمایی کلمات با شبکه های آموزش دیده مبتنی بر مبدل ها [۶۷]

همانطور که در بخش معرفی این نوع شبکه ها بحث شد در شبکه های دو طرفه مبتنی بر مبدل ها که با روی هم قرار دادن چند کدگذار شبکه های خودتوجه بوجود می آیند با استفاده از روش کدگذاری مبتنی بر خودتوجه در هر لایه کدگذار، ورودی به بهترین شکل ممکن کد می شود و با عمیق شدن در شبکه و توجه به محتوا کلمات ورودی و استخراج مفاهیم پیچیده تر، بهترین کدگذاری ممکن برای داده های ورودی بدست می آید.



شکل ۱۷-۵ : بدست آوردن بازنمایی کلمات با شبکه های آموزش دیده مبتنی بر مبدل ها [۶۷]



شکل ۱۸-۵ : بازنمایی کلمات با شبکه های آموزش دیده مبتنی بر مبدل ها [۶۷]

در استخراج ویژگی توسط کدگذار یک بازنمایی جدید از کلمات ورودی توالی ارائه می دهد اما در روش های قبلی بازنمایی کلمات صرفا یک بردار برای هر کلمه ارائه می شود که یکی از نقاط ضعف روش های قبلی بازنمایی کلمات محسوب می شود[۶۷].

برای مثال کلمه خودرو در دو جمله زیر معنی های متفاوتی پیدا می کند[۶۷]:
سرخس گیاهی خودرو است .

صنعت خودرو با مشکل مواجه شده است .

اما در شبکه های دو طرفه مبتنی بر مبدل ها قسمت بزرگی از متن همزمان به شبکه به عنوان ورودی داده می شود و برای تولید بازنمایی هر کلمه به کلمات قبل و بعد نیز توجه می شود.

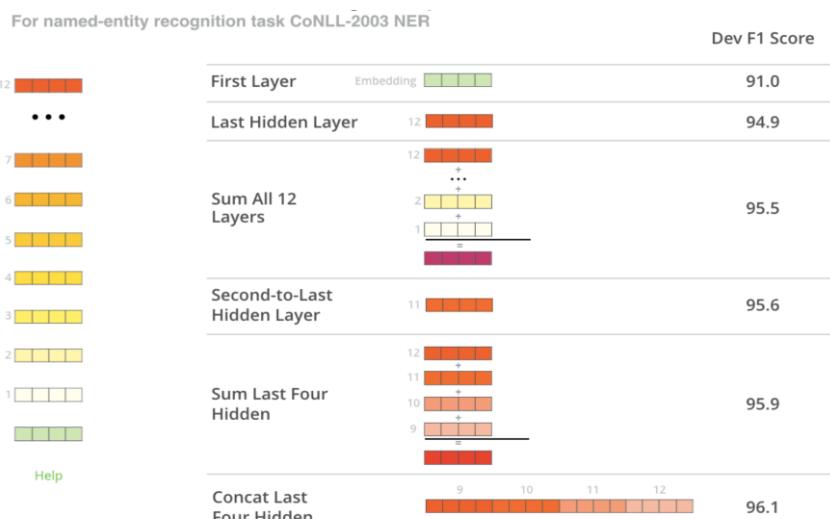
با استفاده از مفهوم یادگیری انتقالی و آموزش این نوع از شبکه ها بر روی داده های ورودی حجیم می توان شبکه را بهترین شکل ممکن برای کدگذاری و استخراج ویژگی آموزش داد و از این شبکه که با بارمحاسباتی بالا ، زمان زیادی برای آموزش آن صرف شده است استفاده کرد[۶۷].

شبکه های دو طرفه مبتنی بر مبدل ها در دو نوع پایه و بزرگ به صورت آموزش دیده با ۱۶ گیگابایت داده ورودی که از منابعی مانند ویکیپدیا و انواع کتاب ها بدست آمده که شامل حدودا ۳ میلیارد کلمه موجود است و شبکه های پایه شامل ۱۱۰ میلیون پارامتر و شبکه های از نوع بزرگ آن شامل ۳۴۰ میلیون پارامتر است[۶۷].

نوع جدیدی از این شبکه نیز با بهینه سازی پارامتر ها توسط افزایش داده های ورودی به ۱۶۰ گیگابایت و استفاده از توالی هایی بلند تر و افزایش داده های دسته ای آموزش نیز وجود دارد که موجب افزایش دقت این نوع شبکه ها تا حدود ۲۰ درصد شده است[۶۸].

به منظور مقایسه روش های موجود بازنمایی کلمات می توان از انواع مسائل متفاوت مانند مسائل دسته بندی استفاده کرد و میزان دقت بازنمایی به کار برده شده را ارزیابی کرد.

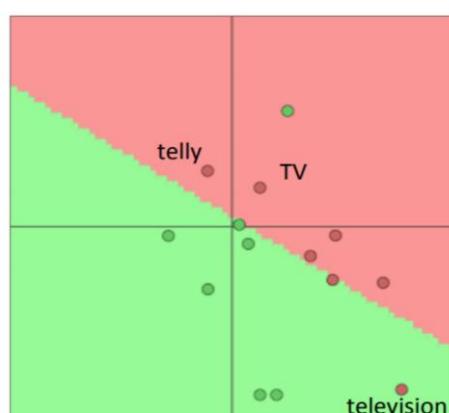
نامگذاری و شناسایی موجودیت ها یکی از انواع مسائل دسته بندی مرسوم است که به برای ارزیابی دقت بازنمایی کلمات نیز مورد استفاده قرار می گیرد.



شکل ۱۹-۵ : بدست آوردن دقت بازنمایی کلمات با شبکه های آموزش دیده مبتنی بر مدل ها [۶۷]

همانطور که مشاهده می شود با در کنار هم قرار دادن ۴ بازنمایی آخر از کلمه مورد نظر می توان یک بازنمایی با دقت بالا برای کلمات یافت که از کنار هم قرار دادن بازنمایی ۴ لایه آخر کدگزار بدست آمده اند و شامل بهترین ویژگی های ممکن برای بازنمایی کلمات هستند[۶۷].

آموزش بازنمایی کلمات بر روی داده های ورودی مسئله ممکن است مزیتی مانند دقت بالا در داده های آموزش را داشته باشد اما در مواقعی که داده های ورودی مسئله به اندازه کافی غنی نباشد ممکن است موجب کاهش تعمیم پذیری مدل و کاهش دقت مدل بر روی داده های سنجش شبکه بشود بنابراین در بیشتر موارد استفاده از شبکه های آموزش دیده با داده های ورودی زیاد و غنی و استفاده از یک مدل تحت نظرات در لایه های آخر شبکه به عنوان تنظیم کننده پارامتر های مدل برای حل مسئله ای خاص تر به کار می رود[۱۹].



شکل ۲۰-۵ : اثر پادگیری دسته بند بدون استفاده از مجموعه داده های غنی [۱۹]

بخش ششم: پیاده سازی و نتایج

۱-۶ مجموعه داده :

مجموعه داده استفاده شده در پیاده سازی شامل بیش از ۴۰۰۰ خبر واقعی و جعلی می باشد که از انواع وبسایت های خبری معتبر جمع آوری و برچسب گذاری شده است [۶۹].

کد تمام قسمت های پیاده سازی در [۷۰] قرار دارد .

| News | Size (Number of articles) | Subjects | |
|-----------|------------------------------|------------------------|----------------------|
| | | Type | Articles size |
| Real-News | 21417 | <i>World-News</i> | 10145 |
| | | <i>Politics-News</i> | 11272 |
| Fake-News | 23481 | <i>Type</i> | <i>Articles size</i> |
| | | <i>Government-News</i> | 1570 |
| | | <i>Middle-east</i> | 778 |
| | | <i>US News</i> | 783 |
| | | <i>left-news</i> | 4459 |
| | | <i>politics</i> | 6841 |
| | | <i>News</i> | 9050 |

شکل ۱-۶ : مشخصات مجموعه داده مورد استفاده [۶۹]

۲-۶ پیش پردازش داده ها :

ابتدا مجموعه داده را فراخوانی می کنیم و برچسب صفر به اخبار جعلی و برچسب یک را به اخبار واقعی نسبت می دهیم !

| | title | text | subject | date | label | year | month |
|---|---|---|--------------|------------|-------|------|---------|
| 0 | As U.S. budget fight looms, Republicans flip t... | WASHINGTON (Reuters) - The head of a conservat... | politicsNews | 2017-12-31 | 1 | 2017 | 2017-12 |
| 1 | U.S. military to accept transgender recruits o... | WASHINGTON (Reuters) - Transgender people will... | politicsNews | 2017-12-29 | 1 | 2017 | 2017-12 |
| 2 | Senior U.S. Republican senator: 'Let Mr. Muell... | WASHINGTON (Reuters) - The special counsel inv... | politicsNews | 2017-12-31 | 1 | 2017 | 2017-12 |
| 3 | FBI Russia probe helped by Australian diplomat... | WASHINGTON (Reuters) - Trump campaign adviser ... | politicsNews | 2017-12-30 | 1 | 2017 | 2017-12 |
| 4 | Trump wants Postal Service to charge 'much mor... | SEATTLE/WASHINGTON (Reuters) - President Donal... | politicsNews | 2017-12-29 | 1 | 2017 | 2017-12 |

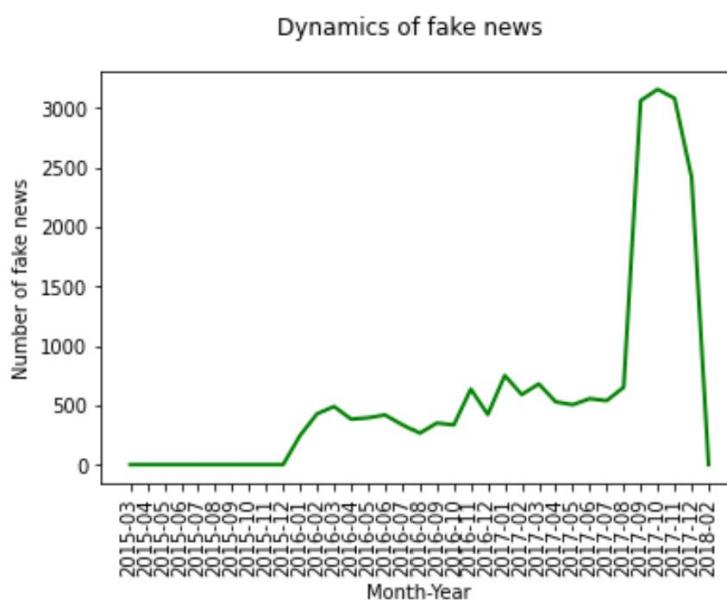
شکل ۲-۶ : نمونه پنج خبر واقعی در مجموعه داده مورد استفاده [۷۰]

^۱ <https://colab.research.google.com/drive/1NwnG6H2RGnKpTfIG9cvnBxp3sOvSFdly?usp=sharing>

| | title | text | subject | date | label |
|---|--|---|---------|-------------------|-------|
| 0 | Donald Trump Sends Out Embarrassing New Year'... | Donald Trump just couldn't wish all Americans ... | News | December 31, 2017 | 0 |
| 1 | Drunk Bragging Trump Staffer Started Russian ... | House Intelligence Committee Chairman Devin Nu... | News | December 31, 2017 | 0 |
| 2 | Sheriff David Clarke Becomes An Internet Joke... | On Friday, it was revealed that former Milwau... | News | December 30, 2017 | 0 |
| 3 | Trump Is So Obsessed He Even Has Obama's Name... | On Christmas day, Donald Trump announced that ... | News | December 29, 2017 | 0 |
| 4 | Pope Francis Just Called Out Donald Trump Dur... | Pope Francis used his annual Christmas Day mes... | News | December 25, 2017 | 0 |

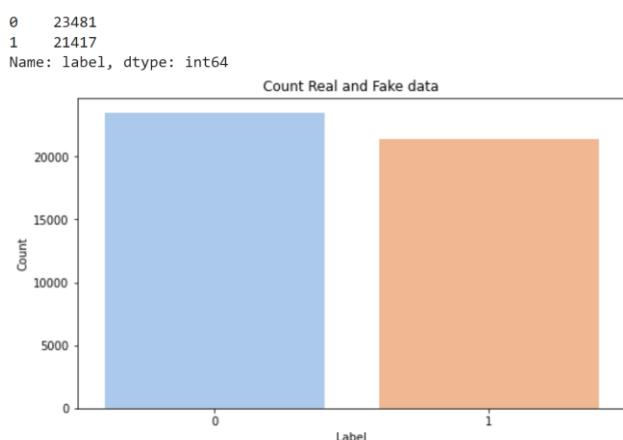
شکل ۳-۶: نمونه پنج خبر جعلی در مجموعه داده مورد استفاده [۷۰]

سپس به بررسی زمان انتشار اخبار جعلی می پردازیم و با رسم نمودار تعداد اخبار جعلی بر حسب زمان انتشار ، روند رو به رشد تعداد اخبار جعلی قابل مشاهده است .

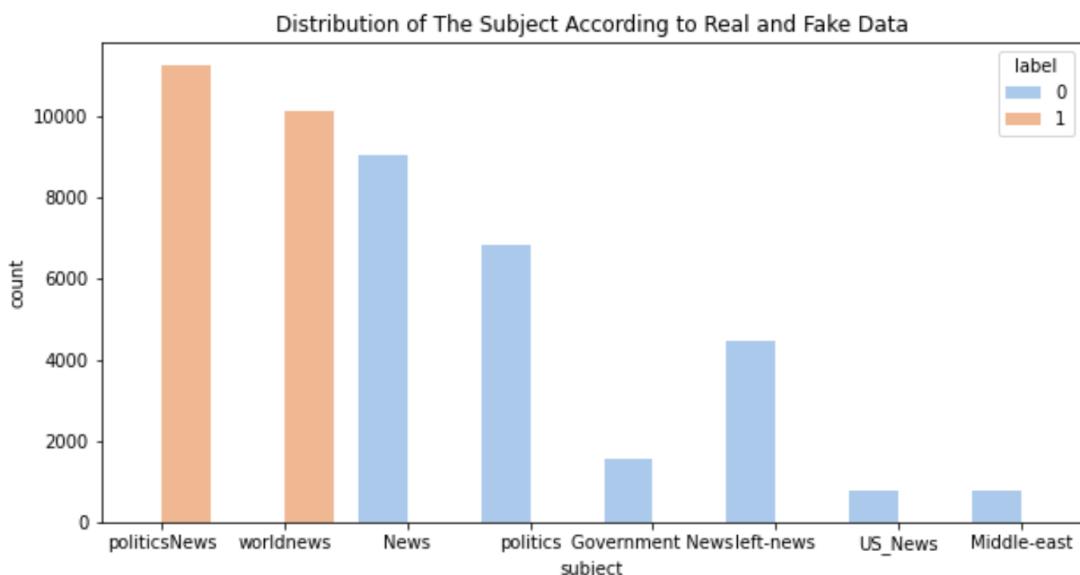


شکل ۴-۶ : زمان انتشار اخبار جعلی در مجموعه داده مورد استفاده [۷۰]

در مرحله بعد تعداد کل اخبار جعلی و اخبار واقعی موجود در مجموعه داده و موضوعات آن ها را بررسی می کنیم .



شکل ۵-۶ : مقایسه تعداد اخبار واقعی و جعلی در مجموعه داده مورد استفاده [۷۰]



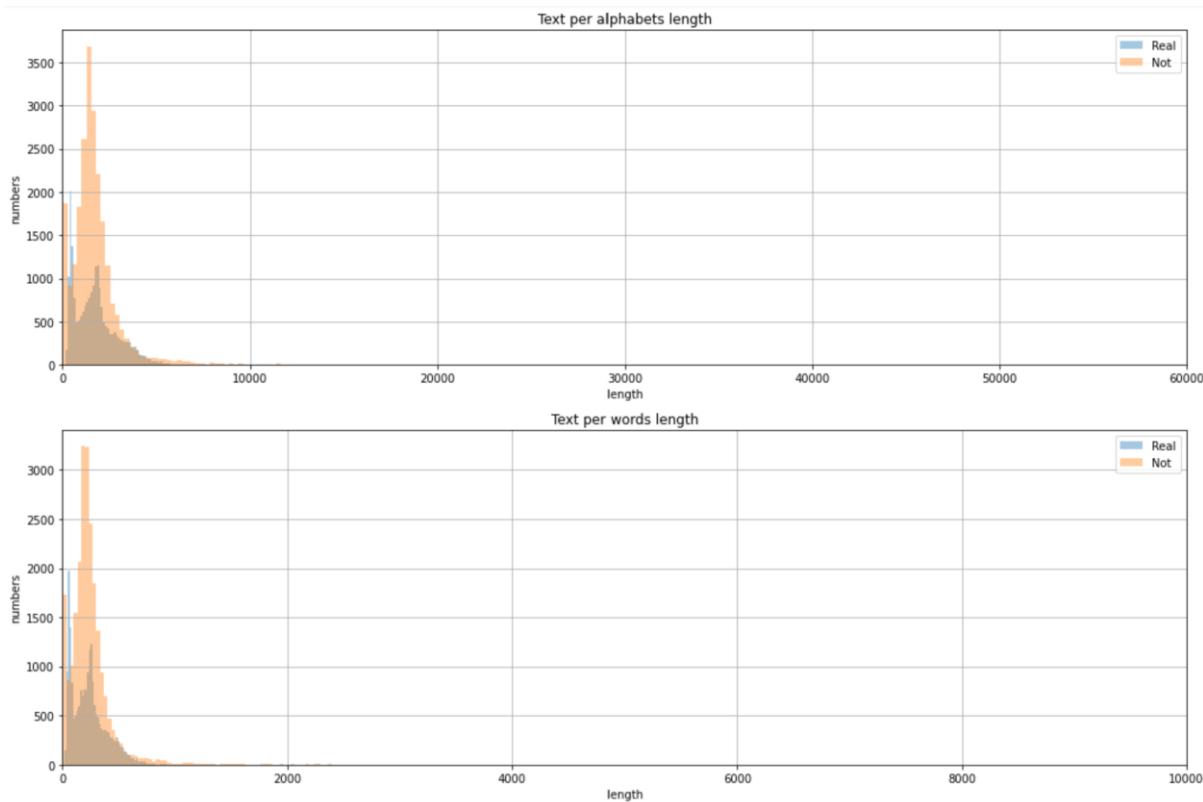
شکل ۶-۶ : مقایسه موضوع اخبار واقعی و جعلی در مجموعه داده مورد استفاده [۷۰]

در مرحله بعد به پاکسازی و پیش پردازش متون مربوط به هر خبر می پردازیم و با روش هایی که در بخش پیش پردازش داده در مورد آن ها بحث شد داده ها را برای ورود به مدل های یادگیری عمیق آماده می کنیم.

| | text | label | length_alphabets | length_words |
|---|---|-------|------------------|--------------|
| 0 | u budget fight loom republican flip fiscal scr... | 1 | 3305 | 448 |
| 1 | u military accept transgender recruit monday p... | 1 | 3023 | 384 |
| 2 | senior u republican senator let mr mueller job... | 1 | 1981 | 270 |
| 3 | fbi russia probe helped australian diplomat ti... | 1 | 1816 | 238 |
| 4 | trump want postal service charge much amazon s... | 1 | 3624 | 502 |

شکل ۷-۶ : مقایسه تعداد کلمات در مجموعه داده مورد استفاده [۷۰]

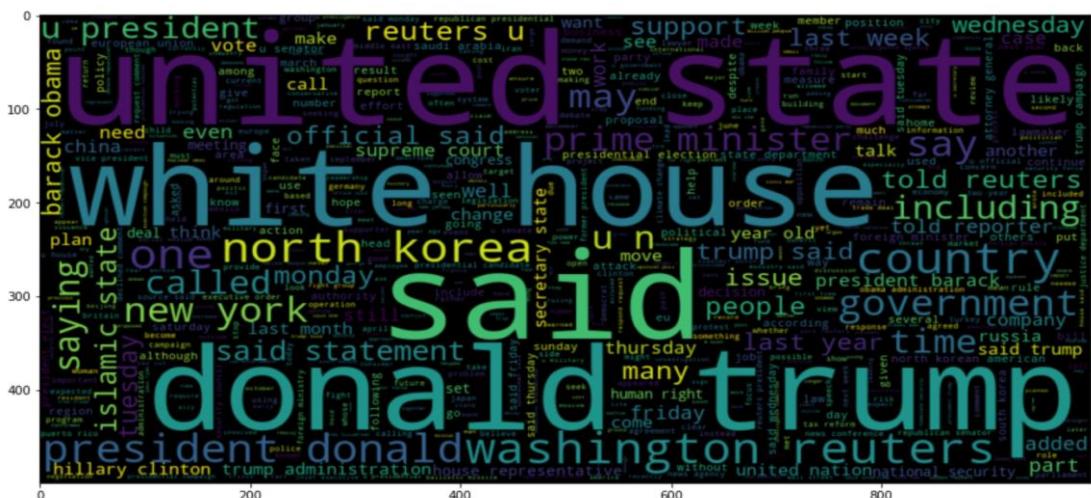
با بدست آوردن تعداد کلمات و حروف الفبا به کار رفته در هر خبر نمودار زیر رارسم می کنیم :



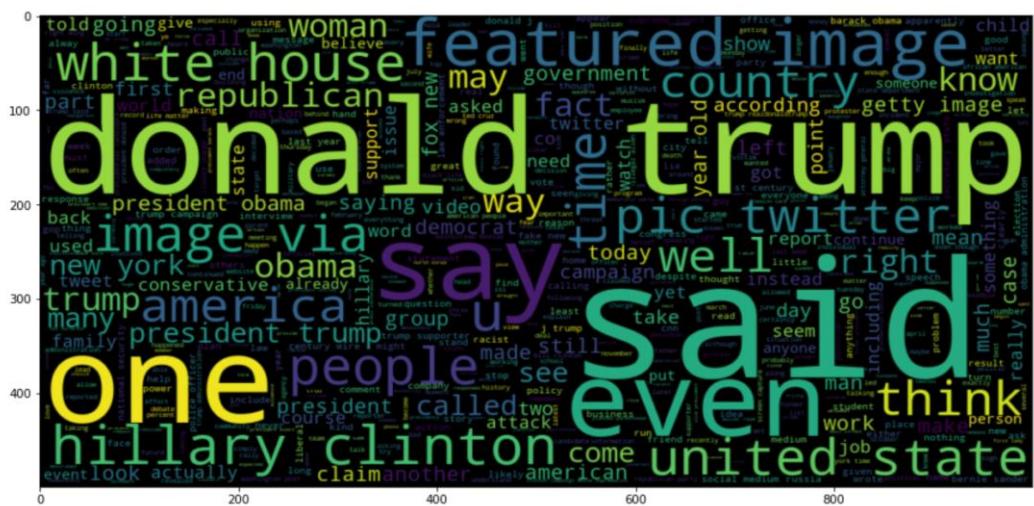
شکل ۸-۶: مقایسه تعداد کلمات در مجموعه داده مورد استفاده [۷۰]

همانطور که مشاهده می شود در مجموعه داده بررسی شده معمولاً در اخبار جعلی تعداد کلمات به کار رفته بیشتر از اخبار واقعی است.

همچنین با ابزار های موجود در کتابخانه تنسورفلو می توان نمای کلی از کلماتی که در اخبار جعلی و واقعی بیشتر به کار رفته اند را نشان داد.

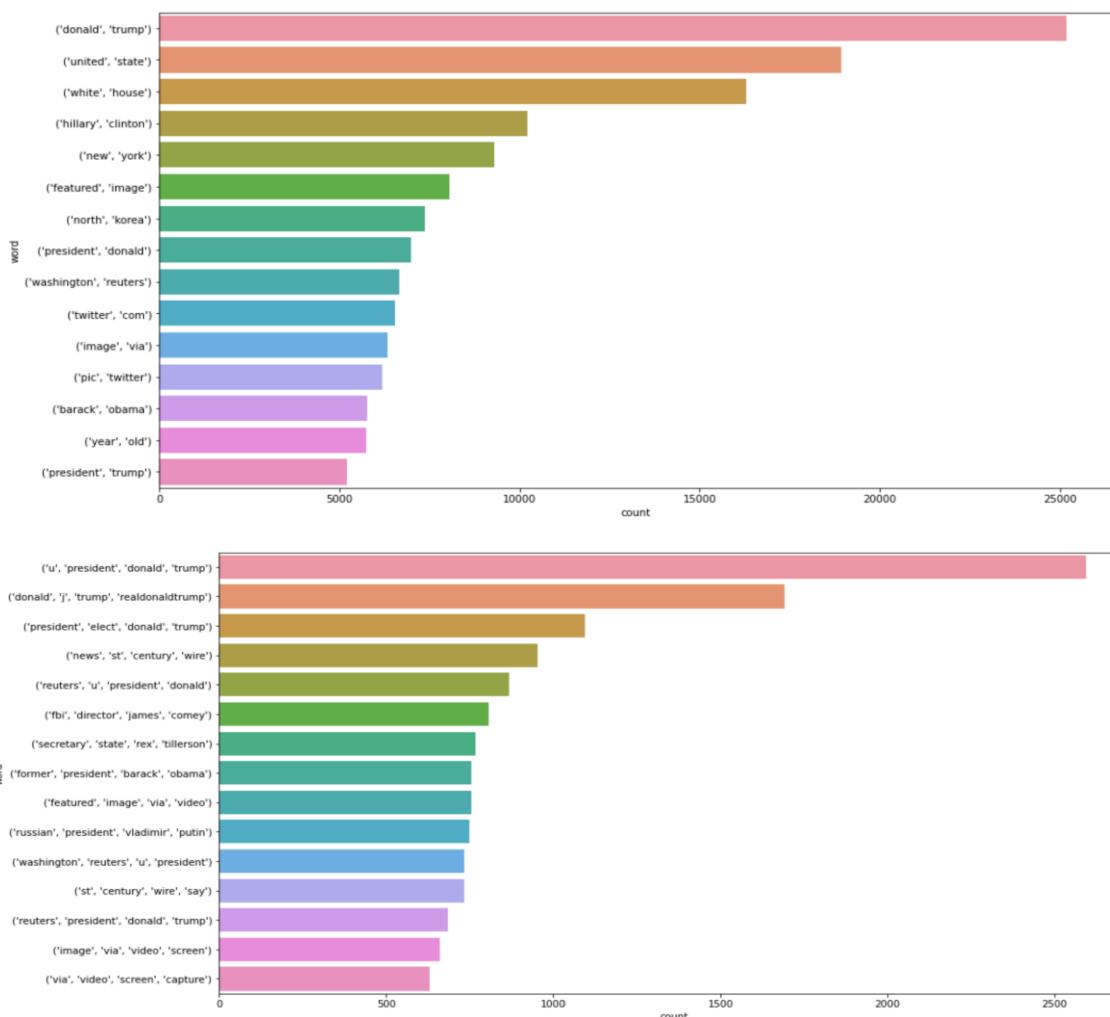


شکل ۶-۹: ابر کلمات بیشتر به کار رفته در اخبار واقعی [۷۰]



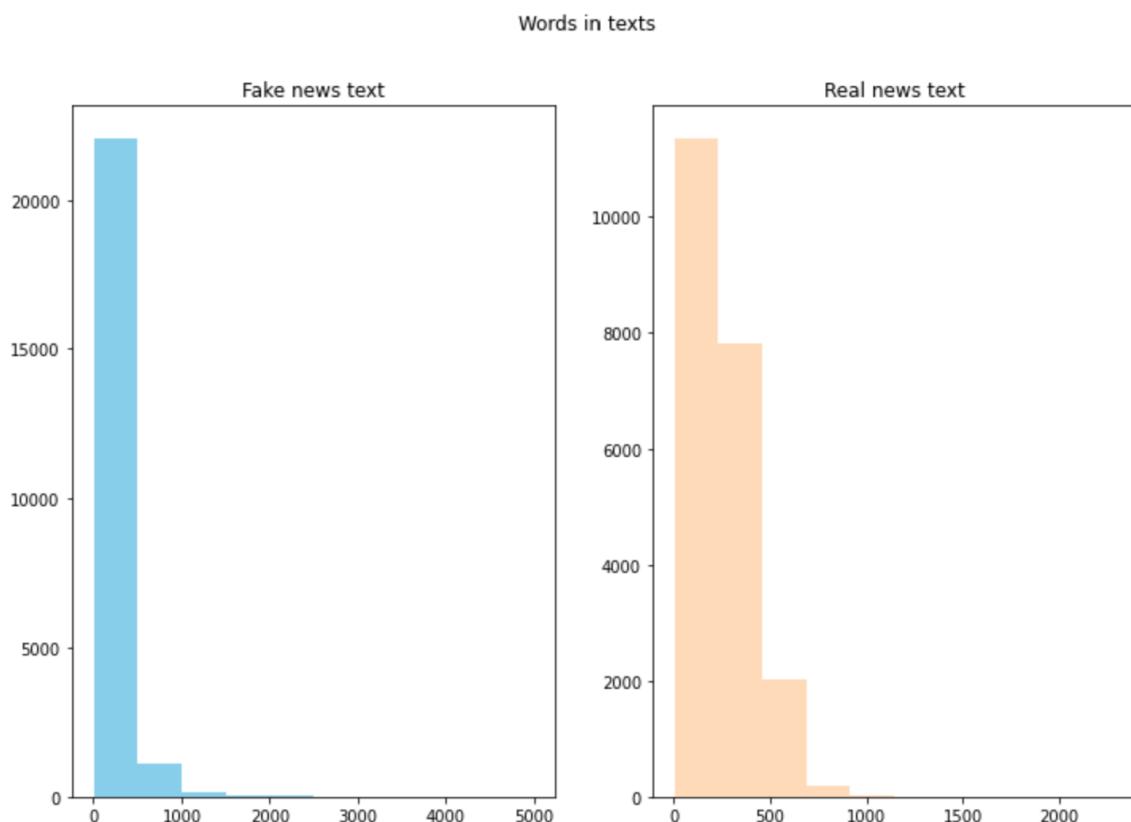
شکل ۶-۱۰: ابر کلمات بیشتر به کار رفته در اخبار جعلی [۷۰]

همچنین می‌توان ترکیب کلماتی که در مجموعه داده بیشتر مورد استفاده قرار گرفته است را نیز رسم و بررسی نمود.



شکل ۱۱-۶: ترکیب کلمات پر استفاده در اخبار [۷۰]

همانطور که در نمودار زیر مشخص است بیشتر اخبار جعلی یا واقعی معمولاً در حدود ۳۰۰ کلمه بیان می‌شوند بنابراین برای تشخیص اخبار جعلی و تشكیل بردار ورودی مدل تنها از ۳۰۰ کلمه استفاده می‌کنیم.



شکل ۱۲-۶ : تعداد کلمات به کار رفته در اخبار واقعی و جعلی [۷۰]

سپس مجموعه داده موجود را به دو دسته داده های آموزش و سنجش تقسیم می‌کنیم:

```
X_train, X_test, y_train, y_test = train_test_split(data['text'], data['label'], random_state=0)
```

شکل ۱۳-۶: تقسیم مجموعه داده به داده های آموزش و سنجش [۷۰]

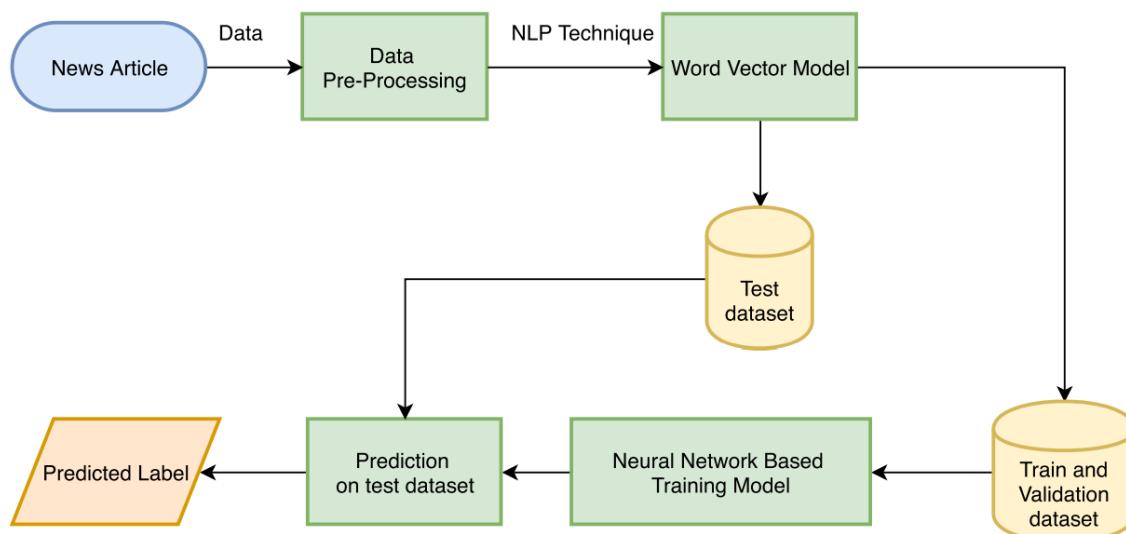
در مرحله بعد تنها ۳۰۰ کلمه از داده های انتخاب شده برای آموزش و سنجش را انتخاب می‌کنیم و در مرحله بعد که همان مرحله نشانه سازی است با اعمال تبدیلی خاص به هر کدام از کلمات یک عدد نسبت داده می‌شود.

```
tokenizer = text.Tokenizer(num_words=max_features)
tokenizer.fit_on_texts(X_train)
tokenized_train = tokenizer.texts_to_sequences(X_train)
X_train = sequence.pad_sequences(tokenized_train, maxlen=maxlen)
tokenized_test = tokenizer.texts_to_sequences(X_test)
X_test = sequence.pad_sequences(tokenized_test, maxlen=maxlen)
```

شکل ۱۴-۶: نشانه سازی برای داده های آموزش و سنجش [۷۰]

۳-۶ پیاده سازی مدل های یادگیری عمیق :

در این مرحله داده هایی که در مراحل قبل پیش پردازش شده اند و به دو دسته داده های آموزش و سنجش تقسیم شده اند را به مدل های یادگیری عمیق پیاده سازی شده اعمال می کنیم و دقت هر کدام از مدل ها را بررسی می کنیم .



شکل ۱۵-۶ : مراحل پیاده سازی مدل یادگیری عمیق

۳-۶ مدل های یادگیری عمیق مبتنی بر شبکه های باحافظه طولانی کوتاه مدت :

در این مدل داده های نشانه سازی شده را ابتدا به لایه بازنمایی اعمال می کنیم تا برای هر کدام از کلمات بازنمایی مناسب ارائه شود و سپس بازنمایی های ارائه شده به عنوان ورودی به شبکه با حافظه طولانی کوتاه مدت اعمال می کنیم و سپس دو لایه شبکه عصبی پرسپترون با اعمال حذف های تصادفی در نظر گرفته شده است و با توجه به اینکه خروجی شبکه باید مقادیر بین صفر و یک برای تعیین برچسب خروجی داشته باشد ،تابع فعالسازی خروجی به صورت sigmoid در نظر گرفته می شود .

```

model = Sequential()

model.add(Embedding(max_features, output_dim=embed_size, input_length=maxlen, trainable=False))
model.add(LSTM(50,dropout = 0.2,recurrent_dropout = 0.2))
model.add(Dense(20,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(5,activation='relu'))
model.add(Dense(1,activation='sigmoid'))

#Compile model with binary_crossentropy loss, Adam optimizer, and accuracy metrics
model.compile(optimizer=tf.keras.optimizers.Adam(lr = 0.01),loss="binary_crossentropy", metrics=['accuracy'])
model.summary()
  
```

شکل ۱۶-۶ : پیاده سازی مدل یادگیری عمیق با شبکه های باحافظه طولانی کوتاه مدت [۷۰]

در این مدل اجازه یادگیری بازنمایی کلمات به لایه بازنمایی داده نشده است و صرفا بر اساس کلمات نشانه سازی شده سعی می کند یک بردار مناسب بدون عقبگرد و یادگیری بازنمایی های موجود ارائه دهد.

Model: "sequential_1"

| Layer (type) | Output Shape | Param # |
|-------------------------|------------------|---------|
| <hr/> | | |
| embedding_1 (Embedding) | (None, 300, 100) | 1000000 |
| lstm_1 (LSTM) | (None, 50) | 30200 |
| dense_3 (Dense) | (None, 20) | 1020 |
| dropout_1 (Dropout) | (None, 20) | 0 |
| dense_4 (Dense) | (None, 5) | 105 |
| dense_5 (Dense) | (None, 1) | 6 |
| <hr/> | | |
| Total params: | 1,031,331 | |
| Trainable params: | 31,331 | |
| Non-trainable params: | 1,000,000 | |

شکل ۱۷-۶ : پیاده سازی مدل یادگیری عمیق با شبکه های باحافظه طولانی کوتاه مدت [۷۰]

بعد از آماده سازی مدل فرآیند یادگیری مدل آغاز می شود و در هر تکرار مجموعه ای از داده های آموزش به شبکه اعمال می شوند :

```
history = model.fit(X_train, y_train, validation_split=0.3, epochs=10, batch_size=batch_size, shuffle=True, verbose = 1)

Epoch 1/10
93/93 [=====] - 129s 1s/step - loss: 0.4317 - accuracy: 0.8007 - val_loss: 0.3365 - val_accuracy: 0.8420
Epoch 2/10
93/93 [=====] - 126s 1s/step - loss: 0.2909 - accuracy: 0.8773 - val_loss: 0.2935 - val_accuracy: 0.8699
Epoch 3/10
93/93 [=====] - 125s 1s/step - loss: 0.2509 - accuracy: 0.8992 - val_loss: 0.2176 - val_accuracy: 0.9216
Epoch 4/10
93/93 [=====] - 125s 1s/step - loss: 0.2320 - accuracy: 0.9152 - val_loss: 0.1868 - val_accuracy: 0.9348
Epoch 5/10
93/93 [=====] - 125s 1s/step - loss: 0.2019 - accuracy: 0.9230 - val_loss: 0.1657 - val_accuracy: 0.9413
Epoch 6/10
93/93 [=====] - 125s 1s/step - loss: 0.1860 - accuracy: 0.9267 - val_loss: 0.2706 - val_accuracy: 0.8903
Epoch 7/10
93/93 [=====] - 124s 1s/step - loss: 0.2655 - accuracy: 0.8930 - val_loss: 0.3692 - val_accuracy: 0.8216
Epoch 8/10
93/93 [=====] - 124s 1s/step - loss: 0.3513 - accuracy: 0.8544 - val_loss: 0.2711 - val_accuracy: 0.8887
Epoch 9/10
93/93 [=====] - 124s 1s/step - loss: 0.3192 - accuracy: 0.8728 - val_loss: 0.4548 - val_accuracy: 0.8505
Epoch 10/10
93/93 [=====] - 128s 1s/step - loss: 0.7052 - accuracy: 0.5317 - val_loss: 0.6702 - val_accuracy: 0.5682
```

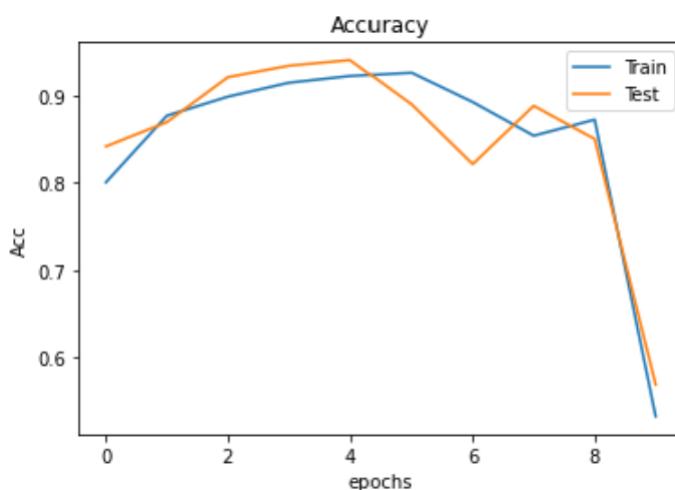
شکل ۱۸-۶ : پیاده سازی مدل یادگیری عمیق با شبکه های باحافظه طولانی کوتاه مدت [۷۰]

و در نهایت با اعمال مجموعه داده سنجش به مدل آموزش دیده ، عملکرد شبکه را بررسی می کنیم :

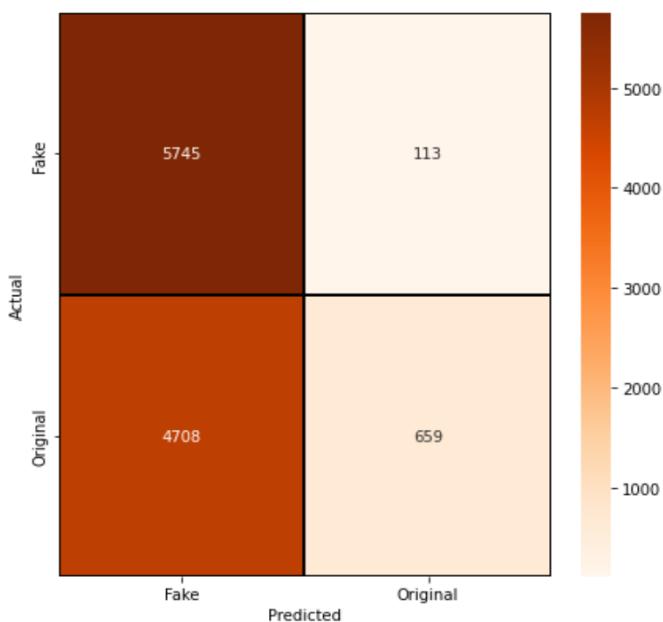
```
1053/1053 [=====] - 60s 57ms/step - loss: 0.6706 - accuracy: 0.5690
Accuracy of the model on Training Data is - 56.903159618377686 %
351/351 [=====] - 20s 56ms/step - loss: 0.6687 - accuracy: 0.5705
Accuracy of the model on Testing Data is - 57.05122351646423 %
```

شکل ۱۹-۶ : پیاده سازی مدل یادگیری عمیق با شبکه های باحافظه طولانی کوتاه مدت [۷۰]

همانطور که مشاهده می شود در این مدل دقیقت بدست آمده برای مجموعه داده سنجش در حدود ۵۷ درصد است که دقیقت نشان می دهد مدل دقیقت کافی را در مواجهه با ورودی های جدید را ندارد و از دلایل آن نیز می توان به لایه بازنمایی اشاره کرد که با توجه به اینکه اجازه یادگیری بازنمایی های موجود به آن داده نشده بود نمی تواند بازنمایی مناسبی برای کلمات موجود ارائه دهد و در نهایت به کاهش دقیقت مدل منجر می شود .



شکل ۲۰-۶ : نمودار دقیقت مدل بر روی داده های آموزش و سنجش در هر تکرار [۷۰]



شکل ۲۱-۶ : عملکرد شبکه در پیش بینی اخبار [۷۰]

در مدل پیاده سازی شده زیر به لایه بازنمایی اجازه یادگیری بردار های بازنمایی داده شده است و در نتیجه با ارائه بازنمایی هایی بهتر و اعمال ورودی نشانه سازی شده به یک شبکه عصبی پرسپترون چند لایه سعی می کند بهترین بازنمایی را برای هر کلمه در لایه بازنمایی بدست آورد و به سایر لایه های مدل برای افزایش دقت مدل ارائه دهد .

```
model_trainable_embedding = Sequential()

model_trainable_embedding.add(Embedding(max_features, output_dim=embed_size, input_length=maxlen, trainable=True))
model_trainable_embedding.add(LSTM(50,dropout = 0.2,recurrent_dropout = 0.2))
model_trainable_embedding.add(Dense(20,activation='relu'))
model_trainable_embedding.add(Dropout(0.2))
model_trainable_embedding.add(Dense(5,activation='relu'))
model_trainable_embedding.add(Dense(1,activation='sigmoid'))

#Compile model with binary_crossentropy loss, Adam optimizer, and accuracy metrics
model_trainable_embedding.compile(optimizer=tf.keras.optimizers.Adam(lr = 0.01),loss="binary_crossentropy", metrics=['accuracy'])
model_trainable_embedding.summary()
```

شکل ۲۲-۶ : پیاده سازی مدل یادگیری عمیق با شبکه های باحافظه طولانی کوتاه مدت [۷۰]

Model: "sequential_4"

| Layer (type) | Output Shape | Param # |
|-----------------------------|------------------|---------|
| <hr/> | | |
| embedding_4 (Embedding) | (None, 300, 100) | 1000000 |
| lstm_4 (LSTM) | (None, 50) | 30200 |
| dense_12 (Dense) | (None, 20) | 1020 |
| dropout_4 (Dropout) | (None, 20) | 0 |
| dense_13 (Dense) | (None, 5) | 105 |
| dropout_5 (Dropout) | (None, 5) | 0 |
| dense_14 (Dense) | (None, 1) | 6 |
| <hr/> | | |
| Total params: 1,031,331 | | |
| Trainable params: 1,031,331 | | |
| Non-trainable params: 0 | | |

شکل ۲۲-۶ : پیاده سازی مدل یادگیری عمیق با شبکه های باحافظه طولانی کوتاه مدت [۷۰]

همانطور که در شکل بالا مشخص است تمام پارامتر های مدل قبل یادگیری هستند .

```

Epoch 1/10
93/93 [=====] - 152s 2s/step - loss: 0.2537 - accuracy: 0.9152 - val_loss: 0.0899 - val_accuracy: 0.9725
Epoch 2/10
93/93 [=====] - 143s 2s/step - loss: 0.0622 - accuracy: 0.9807 - val_loss: 0.0848 - val_accuracy: 0.9745
Epoch 3/10
93/93 [=====] - 142s 2s/step - loss: 0.0441 - accuracy: 0.9871 - val_loss: 0.0851 - val_accuracy: 0.9715
Epoch 4/10
93/93 [=====] - 141s 2s/step - loss: 0.0290 - accuracy: 0.9919 - val_loss: 0.0547 - val_accuracy: 0.9858
Epoch 5/10
93/93 [=====] - 141s 2s/step - loss: 0.0092 - accuracy: 0.9976 - val_loss: 0.0557 - val_accuracy: 0.9857
Epoch 6/10
93/93 [=====] - 141s 2s/step - loss: 0.0111 - accuracy: 0.9968 - val_loss: 0.0536 - val_accuracy: 0.9871
Epoch 7/10
93/93 [=====] - 141s 2s/step - loss: 0.0028 - accuracy: 0.9992 - val_loss: 0.0566 - val_accuracy: 0.9901
Epoch 8/10
93/93 [=====] - 147s 2s/step - loss: 8.3793e-04 - accuracy: 0.9998 - val_loss: 0.0620 - val_accuracy: 0.9897
Epoch 9/10
93/93 [=====] - 143s 2s/step - loss: 0.0014 - accuracy: 0.9997 - val_loss: 0.0630 - val_accuracy: 0.9894
Epoch 10/10
93/93 [=====] - 142s 2s/step - loss: 5.0127e-04 - accuracy: 0.9999 - val_loss: 0.0805 - val_accuracy: 0.9847

```

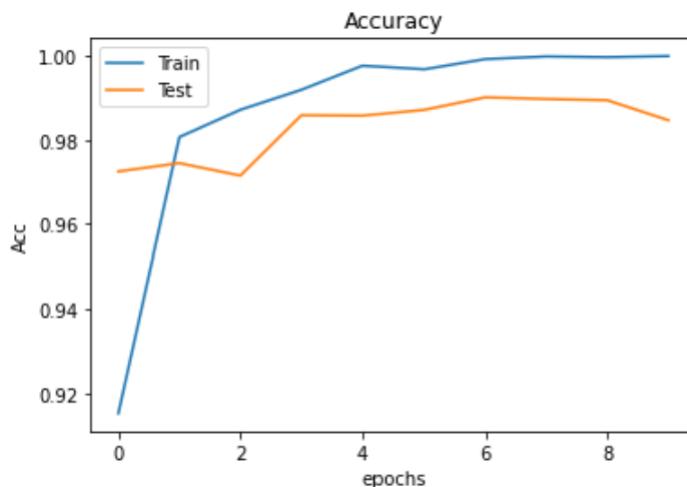
شکل ۲۴-۶: پیاده سازی مدل یادگیری عمیق با شبکه های باحافظه طولانی کوتاه مدت [۷۰]

```

1053/1053 [=====] - 64s 60ms/step - loss: 0.0251 - accuracy: 0.9951
Accuracy of the model on Training Data is - 99.5070219039917 %
351/351 [=====] - 20s 57ms/step - loss: 0.0791 - accuracy: 0.9856
Accuracy of the model on Testing Data is - 98.55679273605347 %

```

شکل ۲۵-۶: پیاده سازی مدل یادگیری عمیق با شبکه های باحافظه طولانی کوتاه مدت [۷۰]



شکل ۲۶-۶: نمودار دقت مدل بر روی داده های آموزش و سنجش در هر تکرار [۷۰]

با بررسی روند یادگیری شبکه و دقت بدست آمده در هر مرحله مشاهده می شود که با افزایش تکرار دقت مدل بر روی مجموعه داده سنجش به نسبت مجموعه داده آموزش روند نزولی پیدا می کند و مدل به سمت بیش برازش خواهد رفت و یکی از دلایل مهم آن می توان به تعداد بالای پارامتر ها در لایه بازنمایی اشاره کرد که برای یافتن بهترین بازنمایی تعداد پارامتر های مدل را به شدت افزایش می دهد بنابراین برای جلوگیری از بیش برازش مدل علاوه بر افزودن لایه های حذف تصادفی (dropout) می توان از لایه های بازنمایی از قبل آموزش دیده استفاده کرد که علاوه بر دقت بالا در بازنمایی موجب بالا رفتن تعداد پارامتر های مدل نخواهد شد.

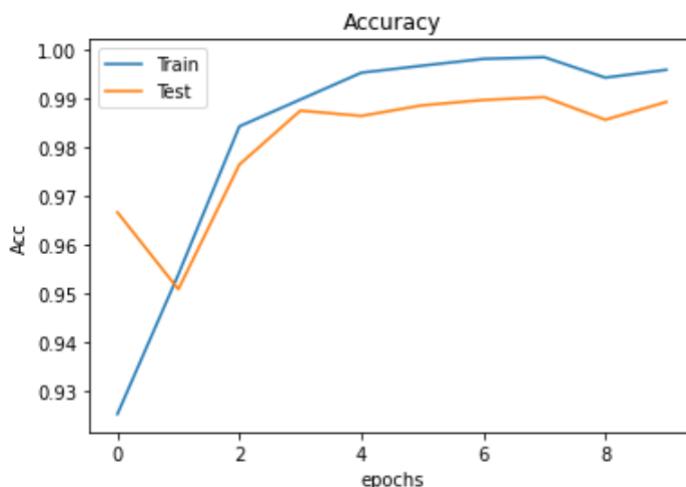
در این مرحله سعی می شود با افزودن حذف تصادفی به لایه های مدل تا حدی از بیش برآش مدل جلوگیری شود و همانطور که مشاهده می شود مدل با دقت بیشتری به داده های جدید واکنش نشان می دهد.

```
Epoch 1/10
93/93 [=====] - 152s 2s/step - loss: 0.2583 - accuracy: 0.9252 - val_loss: 0.1187 - val_accuracy: 0.9667
Epoch 2/10
93/93 [=====] - 141s 2s/step - loss: 0.1677 - accuracy: 0.9540 - val_loss: 0.1278 - val_accuracy: 0.9509
Epoch 3/10
93/93 [=====] - 141s 2s/step - loss: 0.0877 - accuracy: 0.9844 - val_loss: 0.1456 - val_accuracy: 0.9765
Epoch 4/10
93/93 [=====] - 141s 2s/step - loss: 0.0579 - accuracy: 0.9899 - val_loss: 0.0830 - val_accuracy: 0.9876
Epoch 5/10
93/93 [=====] - 145s 2s/step - loss: 0.0348 - accuracy: 0.9954 - val_loss: 0.0664 - val_accuracy: 0.9865
Epoch 6/10
93/93 [=====] - 142s 2s/step - loss: 0.0262 - accuracy: 0.9969 - val_loss: 0.0880 - val_accuracy: 0.9887
Epoch 7/10
93/93 [=====] - 141s 2s/step - loss: 0.0161 - accuracy: 0.9983 - val_loss: 0.0858 - val_accuracy: 0.9898
Epoch 8/10
93/93 [=====] - 141s 2s/step - loss: 0.0137 - accuracy: 0.9986 - val_loss: 0.0918 - val_accuracy: 0.9904
Epoch 9/10
93/93 [=====] - 141s 2s/step - loss: 0.0298 - accuracy: 0.9944 - val_loss: 0.0643 - val_accuracy: 0.9857
Epoch 10/10
93/93 [=====] - 141s 2s/step - loss: 0.0200 - accuracy: 0.9960 - val_loss: 0.0835 - val_accuracy: 0.9894
```

شکل ۲۷-۶ : پیاده سازی مدل یادگیری عمیق با شبکه های باحافظه طولانی کوتاه مدت [۷۰]

```
1053/1053 [=====] - 60s 57ms/step - loss: 0.0268 - accuracy: 0.9964
Accuracy of the model on Training Data is - 99.6406614780426 %
351/351 [=====] - 20s 56ms/step - loss: 0.0876 - accuracy: 0.9879
Accuracy of the model on Testing Data is - 98.78841638565063 %
```

شکل ۲۸-۶ : پیاده سازی مدل یادگیری عمیق با شبکه های باحافظه طولانی کوتاه مدت [۷۰]



شکل ۲۹-۶ : نمودار دقت مدل بر روی داده های آموزش و سنجش در هر تکرار [۷۰]

در مدل ارائه شده بعدی با استفاده از لایه بازنمایی از قبل آموزش دیده بر مبنای روش بردار های بازنمایی کلی (Glove) سعی می شود علاوه بر افزایش دقت مدل ، تعداد پارامتر های قابل یادگیری مدل کاهش یابد تا از بیش برآزش مدل تا حد امکان جلوگیری شود و آموزش مدل با سرعت بیشتری انجام شود .

```
GLOVE_EMBEDDING = "/content/drive/MyDrive/glove.twitter.27B.100d.txt"
```

شکل ۳۰-۶ : فراخوانی بردار کلمات آموزش دیده [۷۰]

Model: "sequential_1"

| Layer (type) | Output Shape | Param # |
|---------------------------------|------------------|---------|
| <hr/> | | |
| embedding (Embedding) | (None, 300, 100) | 1000000 |
| lstm (LSTM) | (None, 50) | 30200 |
| dense (Dense) | (None, 20) | 1020 |
| dense_1 (Dense) | (None, 5) | 105 |
| dense_2 (Dense) | (None, 1) | 6 |
| <hr/> | | |
| Total params: 1,031,331 | | |
| Trainable params: 31,331 | | |
| Non-trainable params: 1,000,000 | | |

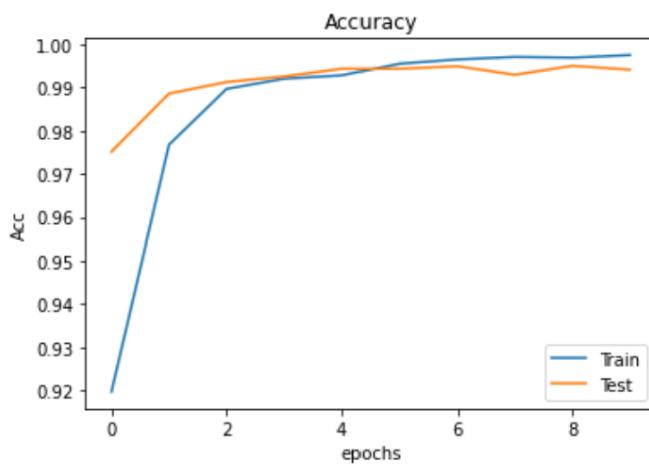
شکل ۳۱-۶ : پیاده سازی مدل یادگیری عمیق با شبکه های باحافظه طولانی کوتاه مدت [۷۰]

```
Epoch 1/10
93/93 [=====] - 129s 1s/step - loss: 0.2344 - accuracy: 0.9197 - val_loss: 0.0789 - val_accuracy: 0.9752
Epoch 2/10
93/93 [=====] - 125s 1s/step - loss: 0.0713 - accuracy: 0.9768 - val_loss: 0.0405 - val_accuracy: 0.9886
Epoch 3/10
93/93 [=====] - 125s 1s/step - loss: 0.0322 - accuracy: 0.9897 - val_loss: 0.0267 - val_accuracy: 0.9913
Epoch 4/10
93/93 [=====] - 123s 1s/step - loss: 0.0229 - accuracy: 0.9921 - val_loss: 0.0226 - val_accuracy: 0.9926
Epoch 5/10
93/93 [=====] - 127s 1s/step - loss: 0.0201 - accuracy: 0.9928 - val_loss: 0.0193 - val_accuracy: 0.9944
Epoch 6/10
93/93 [=====] - 124s 1s/step - loss: 0.0152 - accuracy: 0.9955 - val_loss: 0.0210 - val_accuracy: 0.9944
Epoch 7/10
93/93 [=====] - 124s 1s/step - loss: 0.0114 - accuracy: 0.9965 - val_loss: 0.0175 - val_accuracy: 0.9950
Epoch 8/10
93/93 [=====] - 124s 1s/step - loss: 0.0095 - accuracy: 0.9971 - val_loss: 0.0282 - val_accuracy: 0.9930
Epoch 9/10
93/93 [=====] - 124s 1s/step - loss: 0.0094 - accuracy: 0.9969 - val_loss: 0.0210 - val_accuracy: 0.9951
Epoch 10/10
93/93 [=====] - 123s 1s/step - loss: 0.0075 - accuracy: 0.9975 - val_loss: 0.0228 - val_accuracy: 0.9942

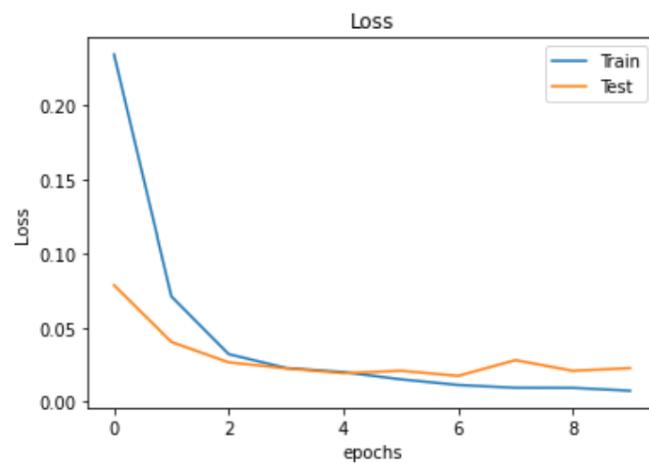
1053/1053 [=====] - 63s 60ms/step - loss: 0.0102 - accuracy: 0.9971
Accuracy of the model on Training Data is - 99.71490502357483 %
351/351 [=====] - 22s 61ms/step - loss: 0.0210 - accuracy: 0.9946
Accuracy of the model on Testing Data is - 99.45657253265381 %
```

شکل ۳۲-۶ : پیاده سازی مدل یادگیری عمیق با شبکه های باحافظه طولانی کوتاه مدت [۷۰]

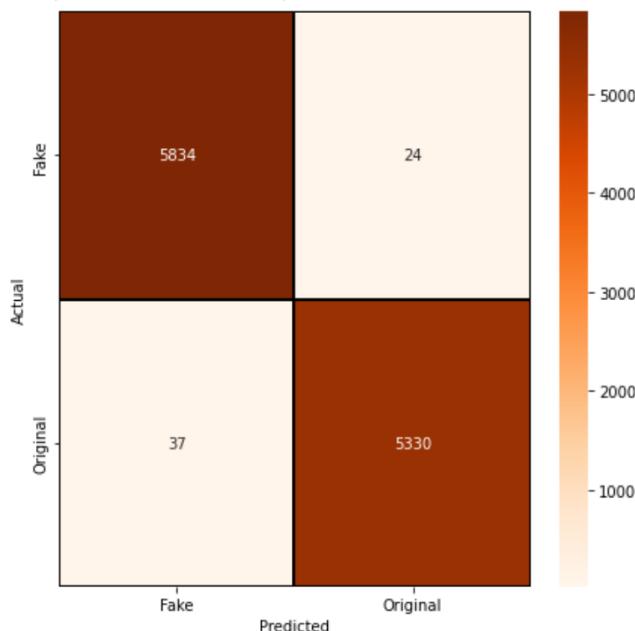
همانطور که مشاهده می شود دقت مدل افزایش می یابد و به ۹۹,۴۵ درصد می رسد .



شکل ۳۳-۶ : نمودار دقیقت مدل بر روی داده های آموزش و سنجش در هر تکرار [۷۰]



شکل ۳۴-۶ : نمودار خطا مدل بر روی داده های آموزش و سنجش در هر تکرار [۷۰]



شکل ۳۵-۶ : عملکرد شبکه در پیش بینی اخبار [۷۰]

در ادامه مدل های مبتنی بر شبکه های با حافظه طولانی کوتاه مدت می توان به شبکه های دوطرفه با حافظه طولانی کوتاه مدت اشاره کرد.

Model: "sequential_5"

| Layer (type) | Output Shape | Param # |
|---------------------------------|------------------|---------|
| <hr/> | | |
| embedding_4 (Embedding) | (None, 300, 100) | 1000000 |
| bidirectional_1 (Bidirectional) | (None, 100) | 60400 |
| dense_8 (Dense) | (None, 20) | 2020 |
| dropout_1 (Dropout) | (None, 20) | 0 |
| dense_9 (Dense) | (None, 5) | 105 |
| dense_10 (Dense) | (None, 1) | 6 |
| <hr/> | | |
| Total params: | 1,062,531 | |
| Trainable params: | 62,531 | |
| Non-trainable params: | 1,000,000 | |

شکل ۳۶-۶ : پیاده سازی مدل یادگیری عمیق با شبکه های با حافظه طولانی کوتاه مدت دوطرفه [۷۰]

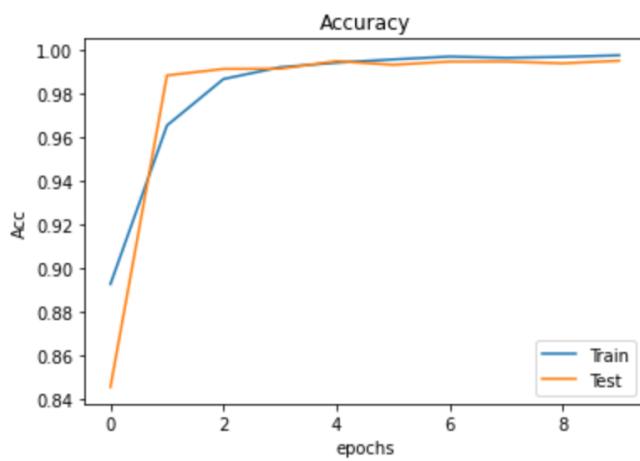
در این مدل نیز همانند مدل قبل از لایه بازنمایی از قبل آموزش دیده به روش بردار بازنمایی کلی (Glove) استفاده شده است.

```
Epoch 1/10
93/93 [=====] - 260s 3s/step - loss: 0.2624 - accuracy: 0.8929 - val_loss: 0.3919 - val_accuracy: 0.8458
Epoch 2/10
93/93 [=====] - 254s 3s/step - loss: 0.0998 - accuracy: 0.9654 - val_loss: 0.0380 - val_accuracy: 0.9883
Epoch 3/10
93/93 [=====] - 251s 3s/step - loss: 0.0385 - accuracy: 0.9867 - val_loss: 0.0258 - val_accuracy: 0.9913
Epoch 4/10
93/93 [=====] - 253s 3s/step - loss: 0.0240 - accuracy: 0.9921 - val_loss: 0.0273 - val_accuracy: 0.9914
Epoch 5/10
93/93 [=====] - 253s 3s/step - loss: 0.0177 - accuracy: 0.9941 - val_loss: 0.0212 - val_accuracy: 0.9949
Epoch 6/10
93/93 [=====] - 253s 3s/step - loss: 0.0140 - accuracy: 0.9956 - val_loss: 0.0230 - val_accuracy: 0.9932
Epoch 7/10
93/93 [=====] - 250s 3s/step - loss: 0.0100 - accuracy: 0.9969 - val_loss: 0.0250 - val_accuracy: 0.9947
Epoch 8/10
93/93 [=====] - 253s 3s/step - loss: 0.0117 - accuracy: 0.9964 - val_loss: 0.0218 - val_accuracy: 0.9947
Epoch 9/10
93/93 [=====] - 253s 3s/step - loss: 0.0110 - accuracy: 0.9968 - val_loss: 0.0249 - val_accuracy: 0.9939
Epoch 10/10
93/93 [=====] - 253s 3s/step - loss: 0.0070 - accuracy: 0.9975 - val_loss: 0.0208 - val_accuracy: 0.9951

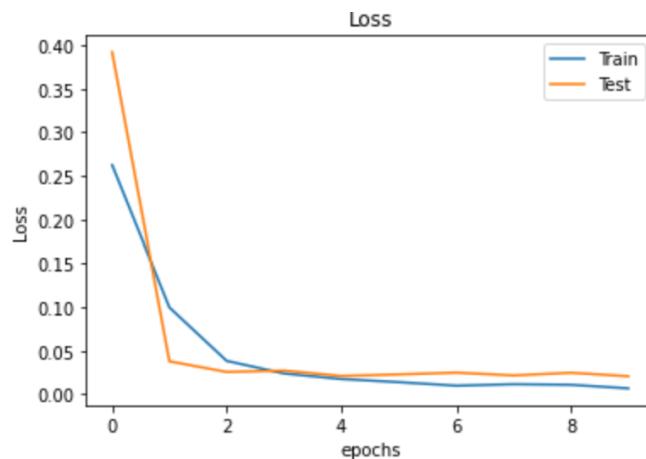
1053/1053 [=====] - 110s 104ms/step - loss: 0.0080 - accuracy: 0.9981
Accuracy of the model on Training Data is - 99.80696439743042 %
351/351 [=====] - 36s 102ms/step - loss: 0.0176 - accuracy: 0.9949
Accuracy of the model on Testing Data is - 99.49220418930054 %
```

شکل ۳۷-۶ : پیاده سازی مدل یادگیری عمیق با شبکه های با حافظه طولانی کوتاه مدت دو طرفه [۷۰]

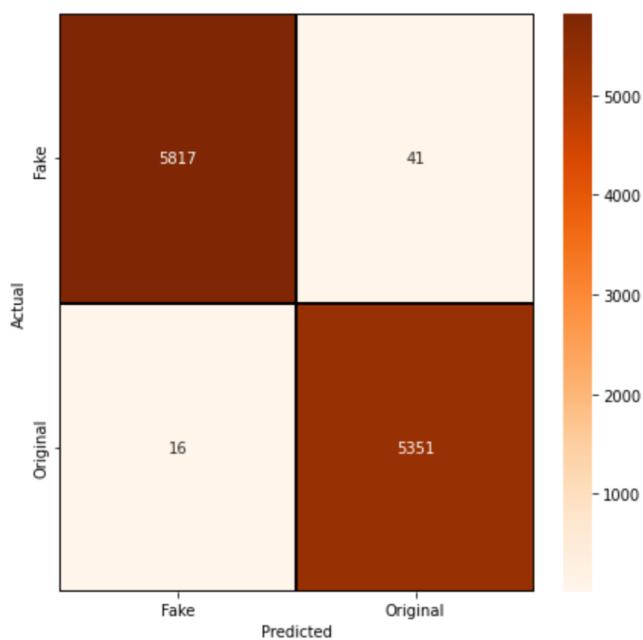
که در نهایت بهترین دقت حاصل شده از این مدل برای مجموعه داده سنجش ۹۹,۴۹ درصد است.



شکل ۳۸-۶ : نمودار دقت مدل بر روی داده های آموزش و سنجش در هر تکرار [۷۰]



شکل ۳۹-۶ : نمودار خطا مدل بر روی داده های آموزش و سنجش در هر تکرار [۷۰]



شکل ۴۰-۶ : عملکرد شبکه در پیش بینی اخبار [۷۰]

۶-۳-۶ مدل های یادگیری عمیق مبتنی بر شبکه های با واحد های بازگشتی دروازه ای (GRU) :

همانطور که در بخش معرفی مدل های شبکه های عصبی عمیق بحث شد شبکه های با واحد های بازگشتی دروازه ای به نسبت شبکه های با حافظه طولانی کوتاه مدت پیچیدگی محاسباتی کمتری دارند که باعث می شود تعداد پارامتر های آن ها نیز کاهش یابد و سرعت یادگیری مدل افزایش می یابد و از بیش برآش مدل نیز جلوگیری می کند.

Model: "sequential_6"

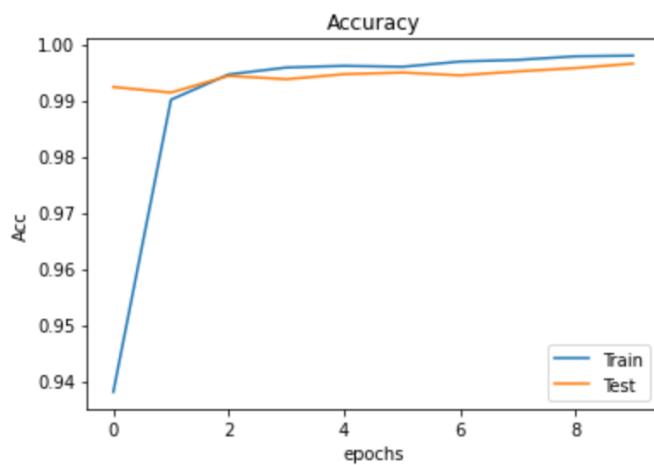
| Layer (type) | Output Shape | Param # |
|---------------------------------|------------------|---------|
| <hr/> | | |
| embedding_5 (Embedding) | (None, 300, 100) | 1000000 |
| gru (GRU) | (None, 50) | 22800 |
| dense_11 (Dense) | (None, 20) | 1020 |
| dropout_2 (Dropout) | (None, 20) | 0 |
| dense_12 (Dense) | (None, 5) | 105 |
| dense_13 (Dense) | (None, 1) | 6 |
| <hr/> | | |
| Total params: 1,023,931 | | |
| Trainable params: 23,931 | | |
| Non-trainable params: 1,000,000 | | |

شکل ۴۱-۶ : پیاده سازی مدل های یادگیری عمیق مبتنی بر شبکه های با واحد های بازگشتی دروازه ای [۷۰]

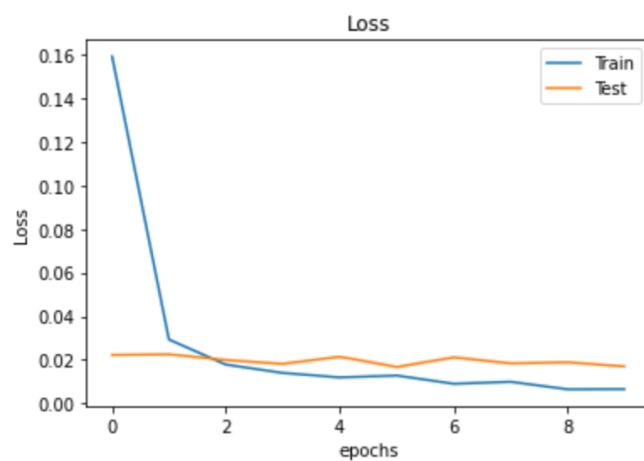
در این مدل نیز همانند مدل قبل از لایه بازنمایی از قبل آموزش دیده به روش بردار بازنمایی کلی (Glove) استفاده شده است.

```
1053/1053 [=====] - 63s 60ms/step - loss: 0.0063 - accuracy: 0.9987
Accuracy of the model on Training Data is - 99.8663604259491 %
351/351 [=====] - 26s 73ms/step - loss: 0.0174 - accuracy: 0.9965
Accuracy of the model on Testing Data is - 99.65255856513977 %
```

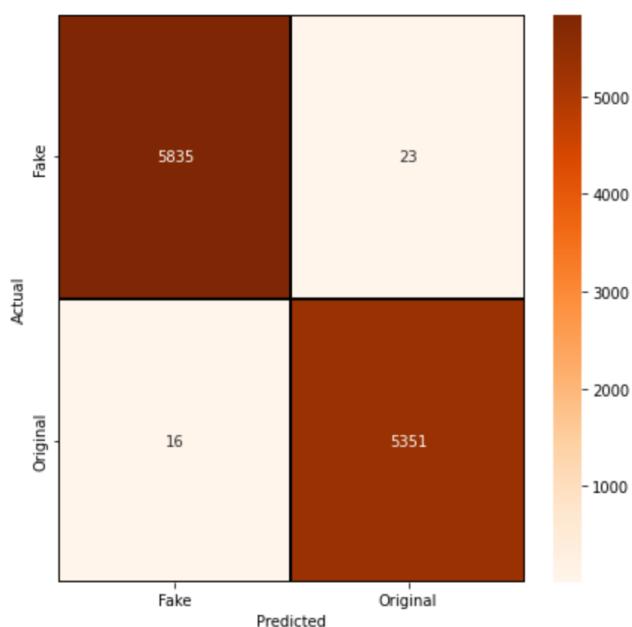
شکل ۴۲-۶ : اندازه گیری دقت بر روی داده های سنجش در شبکه های با واحد های بازگشتی دروازه ای [۷۰]



شکل ۴۳-۶ : نمودار دقت مدل بر روی داده های آموزش و سنجش در هر تکرار [۷۰]



شکل ۴۴-۶ : نمودار خطا مدل بر روی داده های آموزش و سنجش در هر تکرار [۷۰]



شکل ۴۵-۶ : عملکرد شبکه در پیش بینی اخبار [۷۰]

همانطور که مشاهده می شود با اینکه پارامتر های مدل به نسبت سایر مدل ها کمتر است اما با دقت بسیار بالا خروجی را پیش بینی می کند.

از شبکه های با واحد های بازگشتی دروازه ای دو طرفه نیز می توان استفاده کرد که علاوه بر حافظه به علت پیچیدگی محاسباتی کمتر و تعداد پارامتر کمتر برتری دارند.

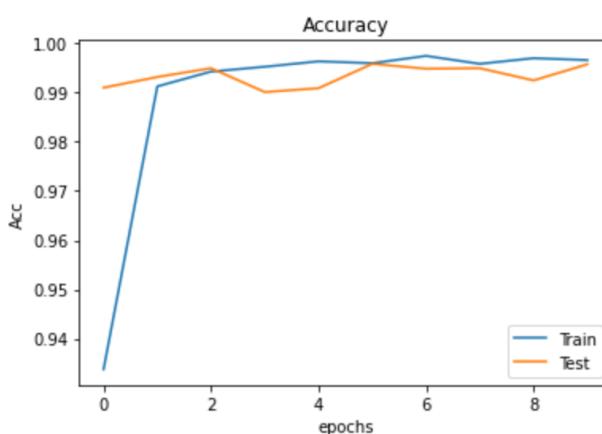
Model: "sequential_8"

| Layer (type) | Output Shape | Param # |
|---------------------------------|------------------|---------|
| <hr/> | | |
| embedding_7 (Embedding) | (None, 300, 100) | 1000000 |
| bidirectional_4 (Bidirectional) | (None, 100) | 45600 |
| dense_17 (Dense) | (None, 20) | 2020 |
| dropout_4 (Dropout) | (None, 20) | 0 |
| dense_18 (Dense) | (None, 5) | 105 |
| dense_19 (Dense) | (None, 1) | 6 |
| <hr/> | | |
| Total params: 1,047,731 | | |
| Trainable params: 47,731 | | |
| Non-trainable params: 1,000,000 | | |

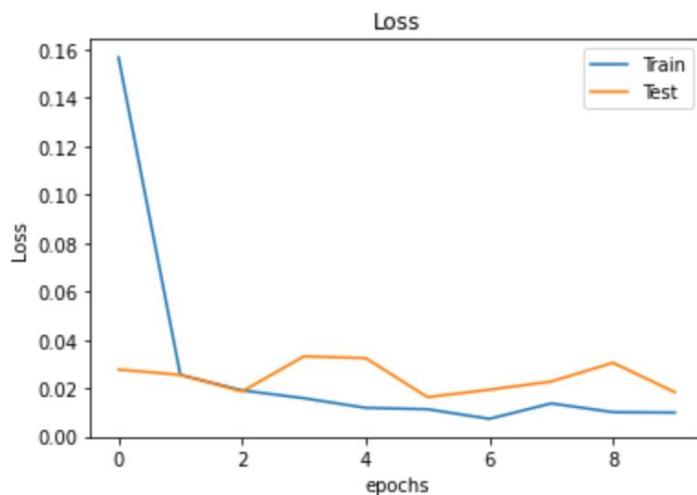
شکل ۴۶-۶ : پیاده سازی مدل های یادگیری عمیق مبتنی بر شبکه های با واحد های بازگشتی دروازه ای [۷۰]

```
1053/1053 [=====] - 125s 118ms/step - loss: 0.0064 - accuracy: 0.9985
Accuracy of the model on Training Data is - 99.84557628631592 %
351/351 [=====] - 35s 99ms/step - loss: 0.0173 - accuracy: 0.9950
Accuracy of the model on Testing Data is - 99.50111508369446 %
```

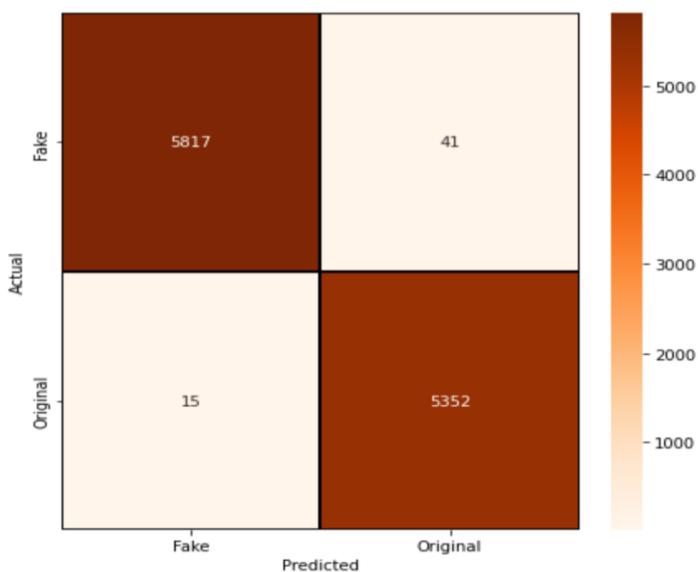
شکل ۴۷-۶ : اندازه گیری دقت بر روی داده های سنجش در شبکه های با واحد های بازگشتی دروازه ای [۷۰]



شکل ۴۸-۶ : نمودار دقت مدل بر روی داده های آموزش و سنجش در هر تکرار [۷۰]



شکل ۴۹-۶ : نمودار خطای مدل بر روی داده های آموزش و سنجش در هر تکرار [۷۰]



شکل ۵۰-۶ : عملکرد شبکه در پیش بینی اخبار [۷۰]

همانطور که در بخش بررسی انواع شبکه های عصبی عمیق بحث شد می تواند از شبکه های چند لایه مبتنی بر حافظه های طولانی کوتاه مدت نیز استفاده کرد و ویژگی های پیچیده تری را استخراج کرد اما با توجه به تعداد پارامتر بالای این نوع از واحد های شبکه های عصبی عمیق میتوان از یک شبکه چند لایه مبتنی بر واحد های بازگشتی دروازه ای استفاده کرد که تا حد امکان از بیش برازش مدل نیز جلوگیری کرد.

```
Model: "sequential_9"
```

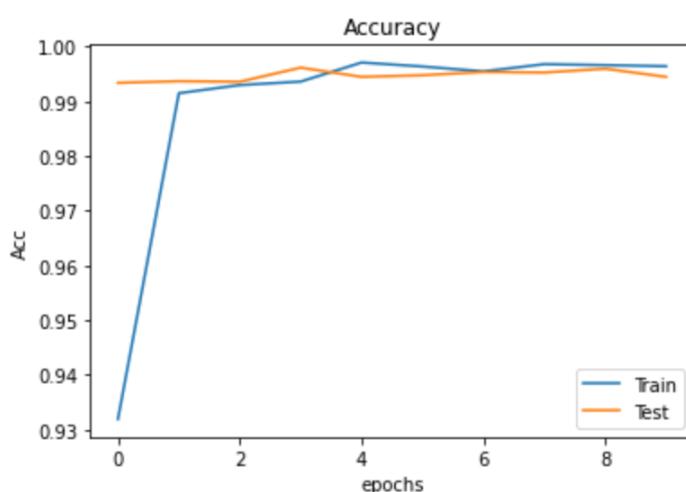
| Layer (type) | Output Shape | Param # |
|---------------------------------|------------------|---------|
| <hr/> | | |
| embedding_8 (Embedding) | (None, 300, 100) | 1000000 |
| gru_4 (GRU) | (None, 300, 128) | 88320 |
| gru_5 (GRU) | (None, 64) | 37248 |
| dense_20 (Dense) | (None, 20) | 1300 |
| dropout_5 (Dropout) | (None, 20) | 0 |
| dense_21 (Dense) | (None, 5) | 105 |
| dense_22 (Dense) | (None, 1) | 6 |
| <hr/> | | |
| Total params: 1,126,979 | | |
| Trainable params: 126,979 | | |
| Non-trainable params: 1,000,000 | | |

شکل ۵۱-۶ : پیاده سازی مدل های یادگیری عمیق مبتنی بر شبکه های با واحد های بازگشتی در واژه ای چندلایه [۷۰]

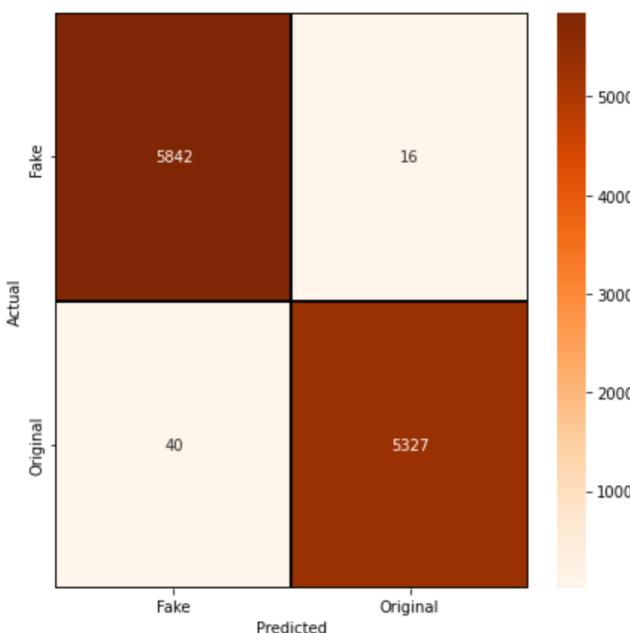
در این مدل نیز همانند مدل قبل از لایه بازنمایی از قبل آموزش دیده به روش بردار بازنمایی کلی (Glove) استفاده شده است .

```
1053/1053 [=====] - 140s 133ms/step - loss: 0.0112 - accuracy: 0.9967
Accuracy of the model on Training Data is - 99.673330783844 %
351/351 [=====] - 45s 129ms/step - loss: 0.0169 - accuracy: 0.9950
Accuracy of the model on Testing Data is - 99.50111508369446 %
```

شکل ۵۲-۶ : اندازه گیری دقت بر روی داده های سنجش در شبکه های با واحد های بازگشتی در واژه ای [۷۰]



شکل ۵۳-۶ : نمودار دقت مدل بر روی داده های آموزش و سنجش در هر تکرار [۷۰]



شکل ۵۴-۶ : عملکرد شبکه در پیش بینی اخبار [۷۰]

۳-۳-۶ مدل های یادگیری عمیق چند کاناله :

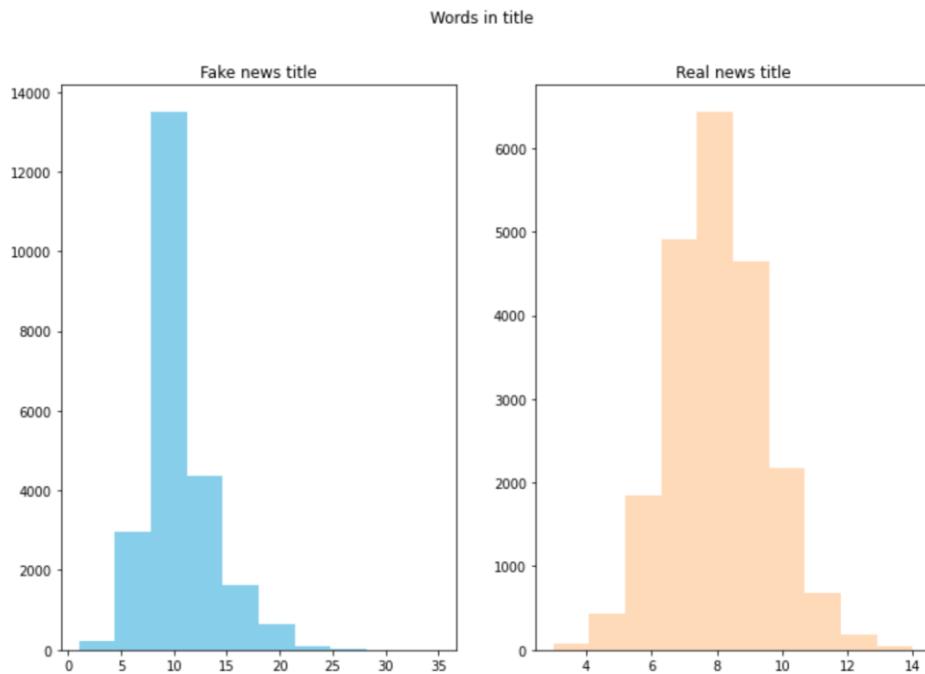
علاوه بر مدل های پیاده شده در مراحل قبل که تمام آن ها تک کanal بودند به این معنا که تنها یک جریان داده ورودی داشتند که به یک جریان داده خروجی منجر می شد اما بر خلاف مدل های قبل می توان از مدل هایی با چند کanal جریان داده ورودی استفاده کرد که نهایتاً به یک خروجی منجر شود به این صورت که می توان عنوان اخبار را یک جریان ورودی مستقل و متن خبر را نیز به عنوان یک جریان ورودی مستقل دیگر در نظر گرفت و از دو کanal ورودی برای پیش بینی خروجی استفاده کرد .

از مزیت های این روش می توان به جلوگیری از پیچیدگی بیش از حد شبکه و غافل نشدن شبکه از یادگیری الگوهای ساده اشاره کرد ، به عنوان مثال در عنوان اخبار می تواند کلمات چکیده اما مهمی وجود داشته باشد که با همان چند کلمه بتوان به راحتی چند خبر را دسته بندی نمود .

| | title | text | label |
|-------|---|---|-------|
| 0 | u budget fight loom republican flip fiscal script | washington reuters head conservative republica... | 1 |
| 1 | u military accept transgender recruit monday p... | washington reuters transgender people allowed ... | 1 |
| 2 | senior u republican senator let mr mueller job | washington reuters special counsel investigati... | 1 |
| 3 | fbi russia probe helped australian diplomat ti... | washington reuters trump campaign adviser geor... | 1 |
| 4 | trump want postal service charge much amazon s... | seattle washington reuters president donald tr... | 1 |
| ... | ... | ... | ... |
| 44893 | mcpain john mccain furious iran treated u sail... | st century wire say wire reported earlier week... | 0 |
| 44894 | justice yahoo settle e mail privacy class acti... | st century wire say familiar theme whenever di... | 0 |
| 44895 | sunnistan u allied safe zone plan take territo... | patrick henningsen st century wireremember oba... | 0 |
| 44896 | blow million al jazeera america finally call q... | st century wire say al jazeera america go hist... | 0 |
| 44897 | u navy sailor held iranian military sign neoco... | st century wire say wire predicted new year lo... | 0 |

44898 rows × 3 columns

شکل ۵۵-۶ : مجموعه داده مورد استفاده به همراه ستون عنوان [۷۰]



شکل ۵۶-۶ : تعداد کلمات استفاده شده در عنوان خبرهای واقعی و جعلی [۷۰]

برای کanal عنوان خبر تعداد ۲۰ کلمه و برای کanal متن خبر تعداد ۳۰۰ کلمه به عنوان کلمات موثر درنظر گرفته می شوند .

```

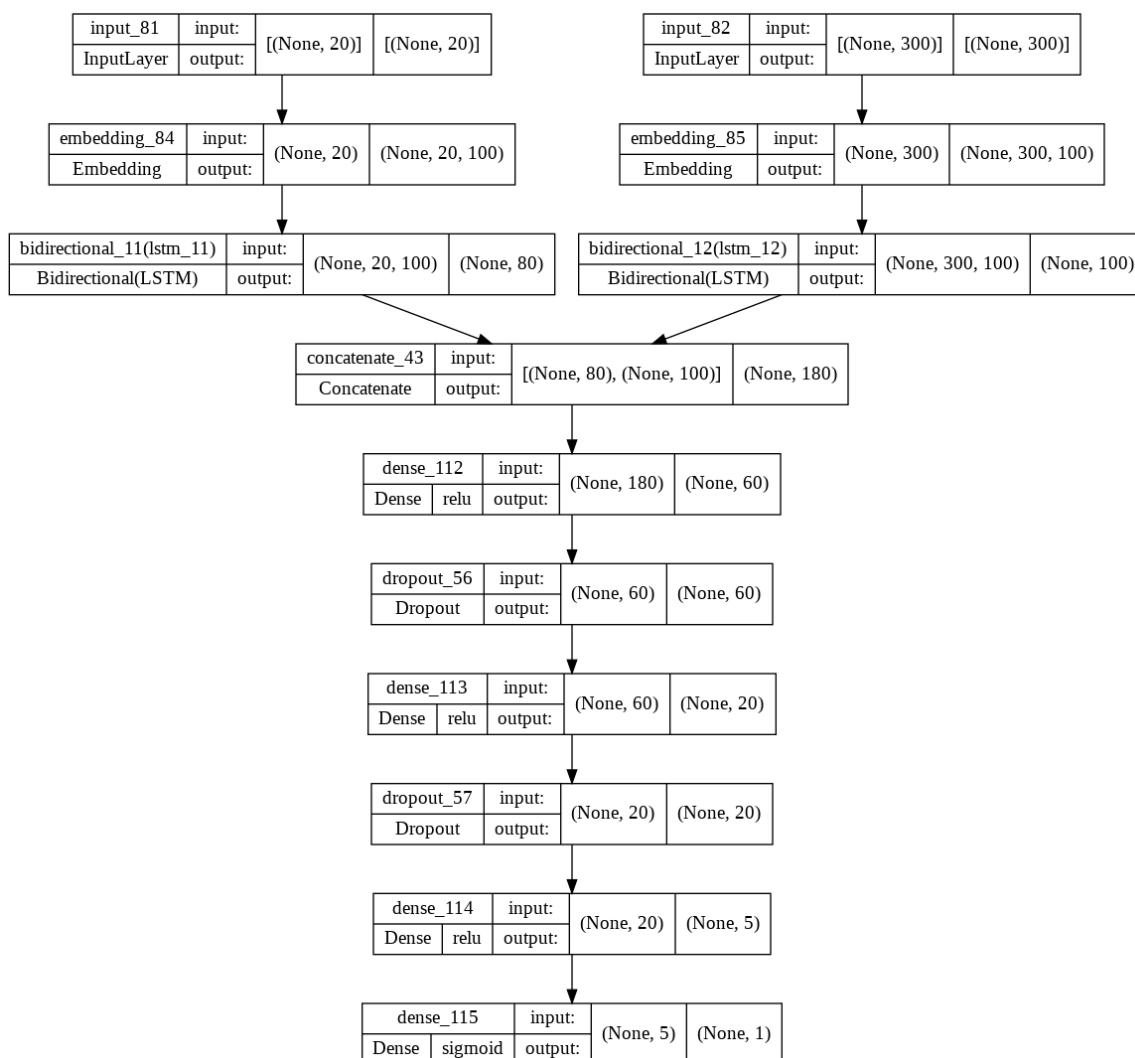
input_title = Input(shape=(maxlen_title,))
input_text = Input(shape=(maxlen,))
embedding_title = Embedding(max_features, output_dim=embed_size,weights = [embedding_matrix], input_length=maxlen_title, trainable=False)(input_title)
Bilstm_title = keras.layers.Bidirectional(LSTM(48,dropout = 0.2,recurrent_dropout = 0.2))(embedding_title)

embedding_text = Embedding(max_features, output_dim=embed_size,weights = [embedding_matrix], input_length=maxlen, trainable=False)(input_text)
Bilstm_text = keras.layers.Bidirectional(LSTM(50,dropout = 0.2,recurrent_dropout = 0.2))(embedding_text)

concatenate_layer = concatenate([Bilstm_title,Bilstm_text])
Dense1 = keras.layers.Dense(60,activation='relu')(concatenate_layer)
Dropout1 = keras.layers.Dropout(0.3)(Dense1)
Dense2 = keras.layers.Dense(20,activation='relu')(Dropout1)
Dropout2 = keras.layers.Dropout(0.2)(Dense2)
Dense3 = keras.layers.Dense(5,activation='relu')(Dropout2)
Output = keras.layers.Dense(1)(Dense3)
model_multichannel = keras.Model(inputs = [input_title,input_text],outputs=[Output])

```

شکل ۵۷-۶: پیاده سازی مدل یادگیری عمیق دو کاناله [۷۰]



شکل ۵۸-۶: پیاده سازی مدل یادگیری عمیق دو کاناله [۷۰]

در این مدل در هر کanal به صورت جداگانه ابتدا بردار های بازنمایی کلمات عنوان و متن اصلی اخبار را به یک شبکه با حافظه طولانی کوتاه مدت دو طرفه اعمال می کنیم و خروجی آن ها در لایه بعد به یک دیگر الحقق می شوند و در نهایت به یک شبکه عصبی پرسپترون چند لایه با حذف های تصادفی اعمال می شوند.

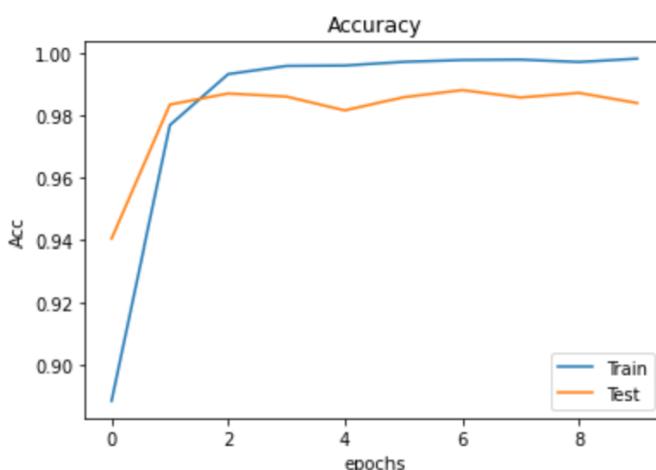
Model: "model_3"

| Layer (type) | Output Shape | Param # | Connected to |
|----------------------------------|------------------|---------|---|
| <hr/> | | | |
| input_13 (InputLayer) | [(None, 20)] | 0 | [] |
| input_14 (InputLayer) | [(None, 300)] | 0 | [] |
| embedding_11 (Embedding) | (None, 20, 100) | 1000000 | ['input_13[0][0]'] |
| embedding_12 (Embedding) | (None, 300, 100) | 1000000 | ['input_14[0][0]'] |
| bidirectional_9 (Bidirectional) | (None, 80) | 45120 | ['embedding_11[0][0]'] |
| bidirectional_10 (Bidirectional) | (None, 100) | 60400 | ['embedding_12[0][0]'] |
| concatenate_4 (Concatenate) | (None, 180) | 0 | ['bidirectional_9[0][0]', 'bidirectional_10[0][0]'] |
| dense_12 (Dense) | (None, 60) | 10860 | ['concatenate_4[0][0]'] |
| dropout_6 (Dropout) | (None, 60) | 0 | ['dense_12[0][0]'] |
| dense_13 (Dense) | (None, 20) | 1220 | ['dropout_6[0][0]'] |
| dropout_7 (Dropout) | (None, 20) | 0 | ['dense_13[0][0]'] |
| dense_14 (Dense) | (None, 5) | 105 | ['dense_13[0][0]'] |
| dense_15 (Dense) | (None, 1) | 6 | ['dense_14[0][0]'] |
| <hr/> | | | |
| Total params: 2,117,711 | | | |
| Trainable params: 117,711 | | | |
| Non-trainable params: 2,000,000 | | | |

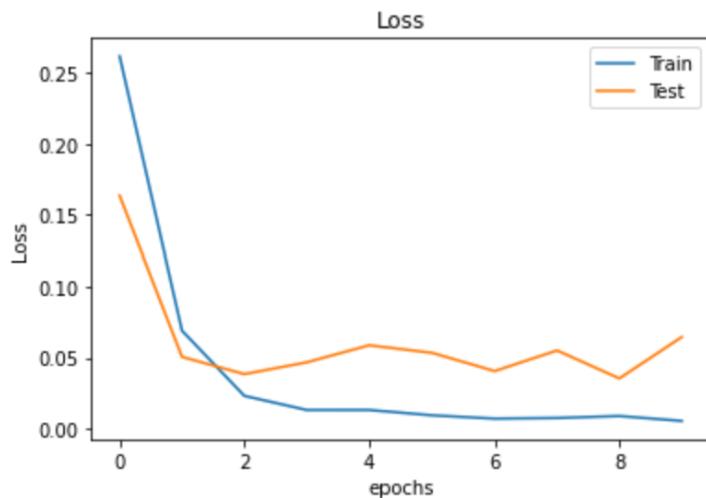
شکل ۵۹-۶ : پیاده سازی مدل یادگیری عمیق دو کاناله [۷۰]

351/351 [=====] - 40s 113ms/step - loss: 0.0178 - accuracy: 0.9958
Accuracy of the model on Testing Data is - 99.58128929138184 %

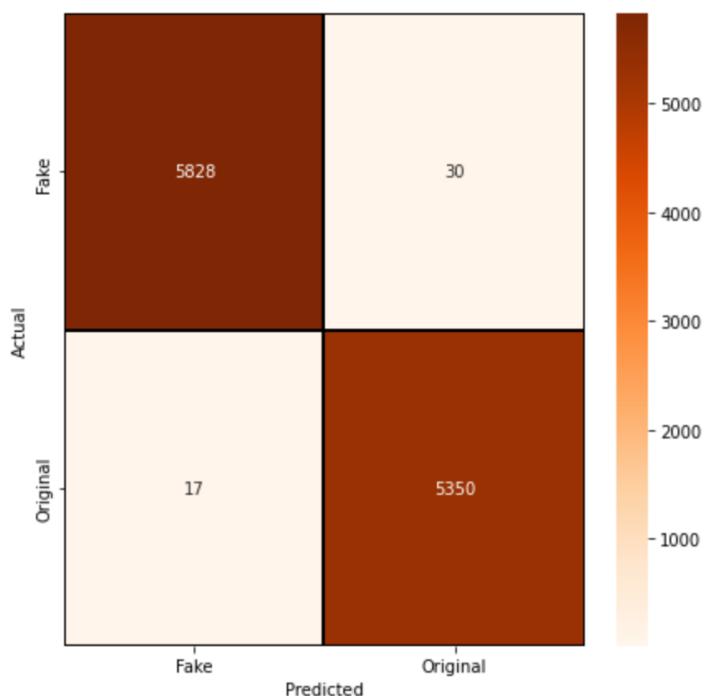
شکل ۶۰-۶ : اندازه گیری دقیق بر روی داده های سنجش در مدل چند کاناله [۷۰]



شکل ۶۱-۶ : نمودار دقیق مدل بر روی داده های آموزش و سنجش در هر تکرار [۷۰]



شکل ۶۲-۶ : نمودار خطا مدل بر روی داده های آموزش و سنجش در هر تکرار [۷۰]



شکل ۶۳-۶ : عملکرد شبکه در پیش بینی اخبار [۷۰]

۴-۳-۶ مدل های یادگیری عمیق مبتنی بر شبکه های کانولوشنی :

همانطور که در بخش بررسی شبکه های عصبی عمیق در مورد شبکه های عصبی کانالوشنی بحث شد می توان از این نوع شبکه با اعمال آن ها بر روی ماتریس ورودی برای استخراج ویژگی ها بر اساس فیلتر های مناسب استفاده کرد.

Model: "sequential_2"

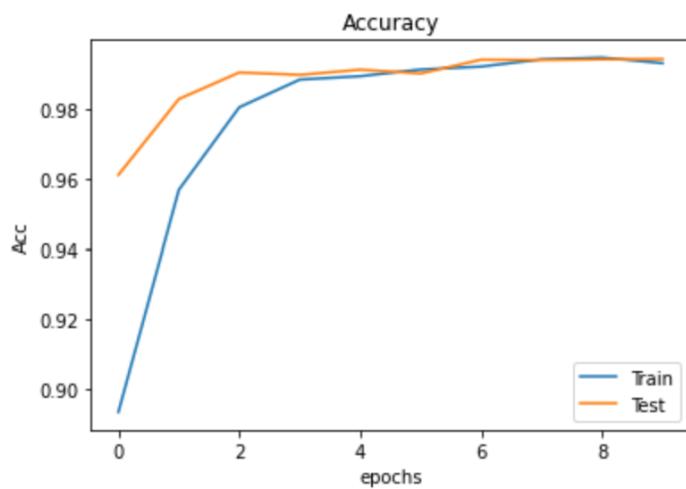
| Layer (type) | Output Shape | Param # |
|----------------------------------|------------------|---------|
| <hr/> | | |
| embedding_88 (Embedding) | (None, 300, 100) | 1000000 |
| dropout_60 (Dropout) | (None, 300, 100) | 0 |
| conv1d_135 (Conv1D) | (None, 296, 64) | 32064 |
| max_pooling1d_131 (MaxPooling1D) | (None, 74, 64) | 0 |
| bidirectional_15 (Bidirectional) | (None, 128) | 66048 |
| dense_118 (Dense) | (None, 1) | 129 |
| <hr/> | | |
| Total params: 1,098,241 | | |
| Trainable params: 98,241 | | |
| Non-trainable params: 1,000,000 | | |

شکل ۶۴-۶: پیاده سازی مدل یادگیری عمیق مبتنی بر شبکه های کانولوشنی [۷۰]

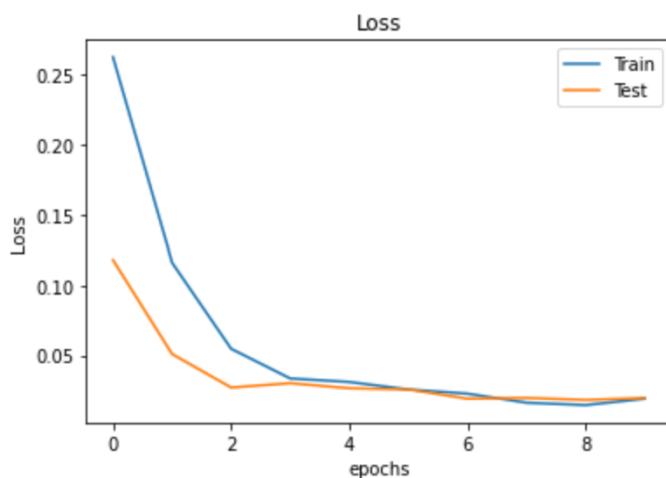
در این مدل بردار های بازنمایی را ابتدا به لایه کانولوشنی با ۶۴ فیلتر که اندازه هر کدام از آنها برابر ۵ است اعمال می کنیم و در لایه بعد با اعمال یک لایه ادغام که اندازه آن برابر ۴ است و بزرگترین مقدار را انتخاب می کند، پارامتر های مدل را کاهش می دهیم و ویژگی های استخراج شده را به عنوان ورودی به یک شبکه با حافظه طولانی کوتاه مدت دوطرفه اعمال می کنیم.

```
1053/1053 [=====] - 57s 54ms/step - loss: 0.0096 - accuracy: 0.9973
Accuracy of the model on Training Data is - 99.72975254058838 %
351/351 [=====] - 13s 36ms/step - loss: 0.0193 - accuracy: 0.9945
Accuracy of the model on Testing Data is - 99.44766163825989 %
```

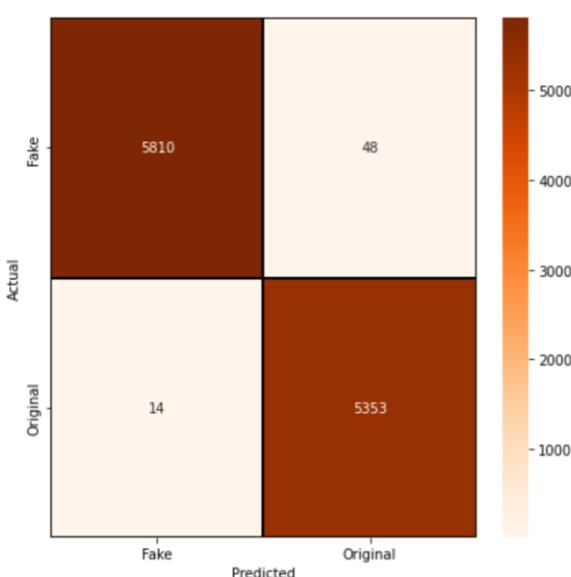
شکل ۶۵-۶: اندازه گیری دقت بر روی داده های سنجش در مدل مبتنی بر شبکه های کانولوشنی [۷۰]



شکل ۶۶-۶ : نمودار دقت مدل بر روی داده های آموزش و سنجش در هر تکرار [۷۰]



شکل ۶۷-۶ : نمودار خطا مدل بر روی داده های آموزش و سنجش در هر تکرار [۷۰]



شکل ۶۸-۶ : عملکرد شبکه در پیش بینی اخبار [۷۰]

۴-۳-۶ مدل های یادگیری عمیق مبتنی بر شبکه های دو طرفه برمبنای مدل ها^۱:

همانطور که در بخش پنجم به معرفی این نوع از مدل های یادگیری عمیق پرداختیم می توان با استفاده از مفهوم یادگیری انتقالی این نوع شبکه ها را بر روی مجموعه های غنی و عظیم از داده ها آموزش داد و از ضرایب و وزن های آموزش دیده به منظور حل مسائلی که در یک دامنه قرار می گیرند استفاده کرد ، نوع پایه آموزش دیده شبکه های دو طرفه مبتنی بر مدل ها شامل ۱۱۰ میلیون پارامتر ، ۱۲ لایه توجه و ۷۶۸ لایه مخفی می باشد که به صورت از قبل آموزش دیده بر روی ۱۱۰۳۸ کتاب و میلیون ها عنوان در ویکیپدیا در کتابخانه تنسورفلو موجود است.

```
bert_preprocess = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3")
bert_encoder = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4")
```

شکل ۶-۶: فراخوانی شبکه آموزش دیده [۷۱]

پس از فراخوانی شبکه از قبل آموزش دیده و نشانه سازی داده های آموزش برای این نوع از شبکه ها تنها با اعمال یک لایه حذف تصادفی و تک نورون لایه خروجی می توان از مفاهیم استخراج شده در کدگذار های این نوع شبکه ها استفاده کرد و با آموزش لایه خروجی و تنظیم مناسب وزن ها از این مفاهیم برای تشخیص اخبار جعلی استفاده نمود .

```
preprocessor = hub.load(
    "https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3")

# Step 1: tokenize batches of text inputs.
text_inputs = [tf.keras.layers.Input(shape=(), dtype=tf.string)]
tokenize = hub.KerasLayer(preprocessor.tokenize)
tokenized_inputs = [tokenize(segment) for segment in text_inputs]

# Step 2 (optional): modify tokenized inputs.
pass

# Step 3: pack input sequences for the Transformer encoder.
seq_length = 300
bert_pack_inputs = hub.KerasLayer(
    preprocessor.bert_pack_inputs,
    arguments=dict(seq_length=seq_length)) # Optional argument.
encoder_inputs = bert_pack_inputs(tokenized_inputs)

outputs = bert_encoder(encoder_inputs)
```

شکل ۶-۷: تنظیم اندازه بردار نشانه سازی و نشانه سازی کلمات ورودی [۷۱]

^۱ <https://colab.research.google.com/drive/1NwnG6H2RGnKpTFIG9cvnBxp3sOvSFdly?usp=sharing>

```

l = tf.keras.layers.Dropout(0.1)(outputs['pooled_output'])

l = tf.keras.layers.Dense(1, activation='sigmoid', name="output")(l)

```

شکل ۷۱-۶: در نظر گرفتن حذف تصادفی و تک نورون لایه خروجی برای انتخاب ویژگی ها [۷۱]

Model: "model"

| Layer (type) | Output Shape | Param # | Connected to |
|-----------------------------------|---|-----------|---|
| <hr/> | | | |
| input_1 (InputLayer) | [None,] | 0 | [] |
| keras_layer_2 (KerasLayer) | (None, None, None) | 0 | ['input_1[0][0]'] |
| keras_layer_3 (KerasLayer) | {'input_type_ids': (None, 300), 'input_word_ids': (None, 300), 'input_mask': (None, 300)} | 0 | ['keras_layer_2[0][0]'] |
| keras_layer_1 (KerasLayer) | {'sequence_output': 109482241, (None, 300, 768), 'default': (None, 768), 'pooled_output': (None, 768), 'encoder_outputs': [(None, 300, 768), (None, 300, 768)]} | 109482241 | ['keras_layer_3[0][0]', 'keras_layer_3[0][1]', 'keras_layer_3[0][2]'] |
| dropout (Dropout) | (None, 768) | 0 | ['keras_layer_1[0][13]'] |
| output (Dense) | (None, 1) | 769 | ['dropout[0][0]'] |
| <hr/> | | | |
| Total params: 109,483,010 | | | |
| Trainable params: 769 | | | |
| Non-trainable params: 109,482,241 | | | |

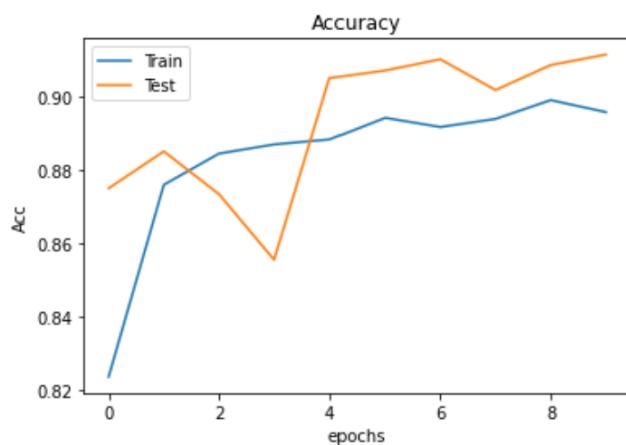
شکل ۷۲-۶: پیاده سازی و تنظیم مدل یادگیری عمیق از قبل آموزش دیده مبتنی بر مبدل ها [۷۱]

```

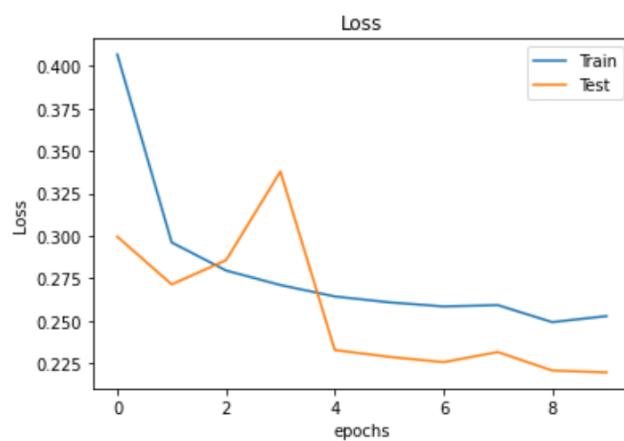
1053/1053 [=====] - 838s 796ms/step - loss: 0.2186 - accuracy: 0.9125
Accuracy of the model on Training Data is - 91.25115275382996 %
351/351 [=====] - 280s 799ms/step - loss: 0.2226 - accuracy: 0.9060
Accuracy of the model on Testing Data is - 90.60133695602417 %

```

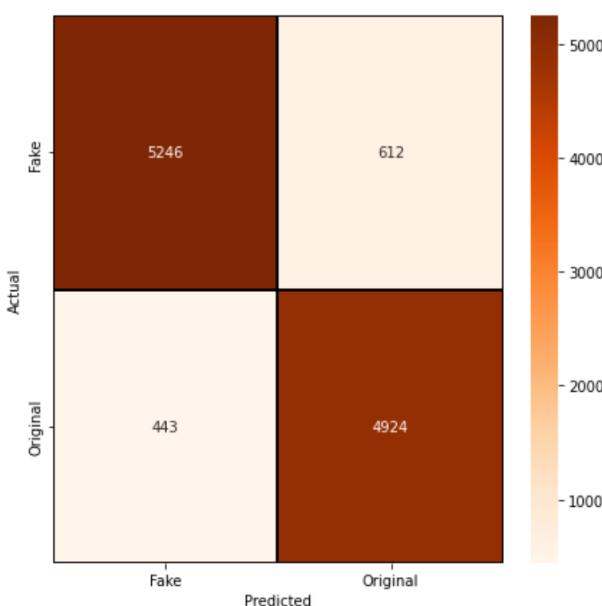
شکل ۷۳-۶: اندازه گیری دقت بر روی داده های سنجش در مدل مبتنی بر مبدل ها [۷۱]



شکل ۷۴-۶ : نمودار دقت مدل بر روی داده های آموزش و سنجش در هر تکرار [۷۱]



شکل ۷۵-۶ : نمودار خطا مدل بر روی داده های آموزش و سنجش در هر تکرار [۷۱]



شکل ۷۶-۶ : عملکرد شبکه در پیش بینی اخبار [۷۱]

نتیجه گیری

ابتدا در مورد اهمیت تشخیص اخبار جعلی و ویژگی های این دسته از اخبار بحث شد سپس انواع رویکرد ها و مدل های شناسایی اخبار جعلی معرفی شد و رویکرد های مبتنی بر یادگیری ماشین و یادگیری عمیق مقایسه شد و انواع مدل ها و شبکه های یادگیری عمیق بررسی و پارامتر ها و ابرپارامتر های آن ها در طول فرآیند آموزش شبکه معرفی شدند.

در این پژوهه سعی شده است با ارائه بهترین بازنمایی از داده های ورودی مسئله ، مدل هایی با حداقل تعداد پارامتر و حداقل پیچیدگی محاسباتی و زمانی پیاده سازی شوند که علاوه بر کاهش هزینه محاسباتی و افزایش سرعت پیاده سازی از تعمیم پذیری بالایی برخوردار باشند و در مواجه با نمونه های جدید دقت های بالایی را ارائه دهند.

| Model | Accuracy(%) |
|------------------------------------|---------------|
| LSTM | 98.55% |
| LSTM + Dropout | 98.78% |
| LSTM + Glove Embedding | 99.45% |
| BiLSTM + Glove Embedding | 99.49% |
| GRU + Glove Embedding | 99.65% |
| BiGRU + Glove Embedding | 99.50% |
| Stacked GRU + Glove Embedding | 99.50% |
| 2 Channel BiLSTM + Glove Embedding | 99.58% |
| Conv1D + BiLSTM + Glove Embedding | 99.44% |
| Pre-trained BERT + Fine Tuning | 90.60% |

جدول ۶-۱ : نتایج

منابع

- [1] Xichen Zhang, Ali A. Ghorbani , "An overview of online fake news: Characterization, detection, and discussion",Information Processing & Management, 2020
- [2] "Fake News", https://en.wikipedia.org/wiki/Fake_news
- [3] Ferrara, E., Varol, O., Davis, C., Menczer, F., & Flammini, A. (2016). The rise of social bots. Communications of the ACM, 59(7), 96–104.
- [4] Kumar, S., West, R., & Leskovec, J. (2016). Disinformation on the web: Impact, characteristics, and detection of wikipedia hoaxes. Proceedings of the 25th international conference on world wide web.
- [5] Kai Shu,Amy Sliva,Suhang Wang,Jiliang Tang and Huan Liu "Fake news detection on social media: A data mining perspective",ACM SIGKDD Explorations Newsletter,
<https://doi.org/10.1145/3137597.3137600>
- [6] Kai Shu and Huan Liu , "Detecting Fake News on Social Media",DOI
10.2200/S00926ED1V01Y201906DMK018
- [7] "Neuron" , <https://en.wikipedia.org/wiki/Neuron>
- [8] "Artificial Neural Networks" , https://en.wikipedia.org/wiki/Artificial_neural_network
- [9] McCulloch, W.S., Pitts, W. A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics 5, 115–133 (1943)
- [10] "Perceptrons" Marvin Minsky and Seymour Papert
- [11] David Rumelhart, Geoffrey Hinton, Ronald Williams,"Learning Internal Representations by Error Propagatio
- [12] Hinton, "NN for MachineLearning",coursera,2015
- [13] Fei Fei Li lectures,cs231n,Stanford 2017
- [14] "Activation Functions" , https://en.wikipedia.org/wiki/Activation_function
- [15] Nair, Vinod; Hinton, Geoffrey E. (2010), "Rectified Linear Units Improve Restricted Boltzmann Machines", 27th International Conference on International Conference on Machine Learning
- [16] Bishop, Christopher M. (2006). Pattern Recognition and Machine Learning. Springer
- [17] Kiprono Elijah Koech , "Cross-Entropy Loss Function" , <https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e>
- [18] Ian Goodfellow and Yoshua Bengio and Aaron Courville , "Deep Learning" ,MIT Press book
- [19] M.Soleymani,"DeepLearning course",Sharif University of Technology,2020
- [20] Diederik P. Kingma, Jimmy Ba , " Adam: A Method for Stochastic Optimization " ,the 3rd International Conference for Learning Representations

-
- [21] Brian Ripley, Pattern Recognition and Neural Networks, 1996
- [22] Dive into Deep Learning , https://d2l.ai/chapter_multilayer-perceptrons/generalization-deep.html
- [23] "Bias-variance tradeoff" , https://en.wikipedia.org/wiki/Bias-variance_tradeoff
- [24] Jason Brownlee , "Master Machine Learning Algorithms",Introduction to the Bias-Variance Trade-Off in Machine Learning, <https://machinelearningmastery.com/gentle-introduction-to-the-bias-variance-trade-off-in-machine-learning/>
- [25] Seema Singh , Understanding the Bias-Variance Tradeoff ,
<https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229>
- [26] "overfitting" , <https://en.wikipedia.org/wiki/Overfitting>
- [27] Abhinav Sagar, VIT Vellore on December 6, 2019 in Neural Networks ,
<https://www.kdnuggets.com/2019/12/5-techniques-prevent-overfitting-neural-networks.html>
- [28] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov; "Dropout: A Simple Way to Prevent Neural Networks from Overfitting"
- [29] Yarin Gal, Zoubin Ghahramani , "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning",Published in ICML 2016
- [30] Kurtis Pykes , "The Vanishing/Exploding Gradient Problem in Deep Neural Networks" ,
<https://towardsdatascience.com/the-vanishing-exploding-gradient-problem-in-deep-neural-networks-191358470c11>
- [31] Yash Bohra , "The Challenge of Vanishing/Exploding Gradients in Deep Neural Networks" ,
<https://www.analyticsvidhya.com/blog/2021/06/the-challenge-of-vanishing-exploding-gradients-in-deep-neural-networks/>
- [32] Xavier Glorot , Yoshua Bengio, "Understanding the difficulty of training deep feedforward neural networks"
- [33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification"
- [34] Bing Xu, Naiyan Wang, Tianqi Chen, Mu Li , "Empirical Evaluation of Rectified Activations in Convolutional Network"
- [35] Sergey Ioffe, Christian Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift"
- [36] Razvan Pascanu, Tomas Mikolov, Yoshua Bengio , "On the difficulty of training Recurrent Neural Networks"
- [37] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu , "A Survey on Deep Transfer Learning"
- [38] "Learning rate , Learning rate schedule" , https://en.wikipedia.org/wiki/Learning_rate
- [39] Renu Khandelwal , "Recurrent Neural Network-RNN" ,
<https://medium.datadriveninvestor.com/recurrent-neural-network-rnn-52dd4f01b7e8>

- [40] "The Unreasonable Effectiveness of Recurrent Neural Networks" ,May 21 2015
<https://karpathy.github.io/2015/05/21/rnn-effectiveness/>
- [41] Corentin Tallec, Yann Ollivier , "Unbiasing Truncated Backpropagation Through Time",Neural and Evolutionary Computing
- [42] Mike Schuster and Kuldip K. Paliwal , "Bidirectional Recurrent Neural Networks" IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 45
- [43] "Understanding LSTM Networks" , <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- [44] S Hochreiter, J Schmidhuber , "Long short-term memory" , Neural computation, 1997
- [45] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation",Neural and Evolutionary Computing
- [46] "Gated recurrent unit" , https://en.wikipedia.org/wiki/Gated_recurrent_unit
- [47] "Differences Between Bidirectional and Unidirectional LSTM" by Enes Zvornicanin ,
<https://www.baeldung.com/cs/bidirectional-vs-unidirectional-lstm>
- [48] "A Comprehensive Guide to Convolutional Neural Networks" by Sumit Saha ,
<https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
- [49] "Understanding Convolutions in Text" ,
<https://debjyotidatta.github.io/nlp/deep/learning/word-embeddings/2016/11/27/Understanding-Convolutions-In-Text/>
- [50] Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio , "Neural Machine Translation by Jointly Learning to Align and Translate"
- [51] Abhishek Singh,Sep 7 2019, <https://towardsdatascience.com/attention-networks-c735befb5e9f>
- [52] Lilian Weng, Attention? Attention! , <https://lilianweng.github.io/posts/2018-06-24-attention/>
- [53] MAHENDRAN VENKATACHALAM ,JULY 6 2019 , "Different types of Attention in Neural Networks" , <https://gotensor.com/2019/07/06/different-types-of-attention-in-neural-networks/#:~:text=Hard%20vs%20Soft%20attention&text=in%20their%20paper%2C%20soft%20attention,select%20a%20single%20hidden%20state.>
- [54] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin , "Attention Is All You Need", NIPS 2017
- [55] Jay Alammar , "The Illustrated Transformer" , <https://jalammar.github.io/illustrated-transformer/>
- [56] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"
- [57] Rani Horev,"BERT Explained: State of the art language model for NLP",
<https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270>

-
- [58] Raimi Karim,"Illustrated: Self-Attention", <https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a>
- [59] MAHENDRAN VENKATACHALAM,"From Attention to Self Attention to Transformers", <https://gotensor.com/2019/07/10/from-attention-to-self-attention-to-transformers>
- [60] Tithi Sreemany,"Essential Text Pre-processing Techniques for NLP" , <https://www.analyticsvidhya.com/blog/2021/09/essential-text-pre-processing-techniques-for-nlp/>
- [61] Adithya Challa,"Preprocessing steps in Natural Language Processing (NLP)" , <https://www.educative.io/answers/preprocessing-steps-in-natural-language-processing-nlp>
- [62] Purva Huilgol , "Quick Introduction to Bag-of-Words (BoW) and TF-IDF for Creating Features from Text" , <https://www.analyticsvidhya.com/blog/2020/02/quick-introduction-bag-of-words-bow-tf-idf/>
- [63] "Word Embeddings in NLP", <https://www.geeksforgeeks.org/word-embeddings-in-nlp/>
- [64] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, "Efficient Estimation of Word Representations in Vector Space"
- [65] Jay Alammar,"The Illustrated Word2vec" , <https://jalammar.github.io/illustrated-word2vec/>
- [66] Jeffrey Pennington,Richard Socher,Christopher D. Manning,"GloVe: Global Vectors for Word Representation", <https://nlp.stanford.edu/projects/glove>
- [67] Jay Alammar , "The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)", <http://jalammar.github.io/illustrated-bert>
- [68] Shri Varsheni R , "Why and how to use BERT for NLP Text Classification?", <https://www.analyticsvidhya.com/blog/2021/06/why-and-how-to-use-bert-for-nlp-text-classification/>
- [69] ISOT Fake News Dataset , <https://www.uvic.ca/ecs/ece/isot/datasets/fake-news/index.php>
- [70] Project codes1 , <https://colab.research.google.com/drive/10UHVVDNnc1YhVNlxhQRRm1clsKsxSYooI?usp=sharing>
- [71] Project Codes2 , <https://colab.research.google.com/drive/1NwnG6H2RGnKpTFlG9cvnBxp3sOvSFdly?usp=sharing>