



دانشگاه تربیت مدرس

دانشکده علوم ریاضی

پایان نامه دوره کارشناسی ارشد علوم کامپیوتر

روش های عمیق مبتنی بر مبدل های بینایی در
تحلیل داده های تصویری

توسط

سید محمد بادزه ره

استاد راهنما

آقای دکتر منصور رزقی

تابستان ۱۴۰۴

تعدیم به

مدرسگوار و مادر محربانم و برادر عزیزم
آن ها که از خواسته هایشان گذشتند، سختی هارا بجهان خریدند و خود را سپرپلای مشکلات و
ناملایات کر دند تا من به جایگاهی که اکنون در آن ایستاده ام برسم.

قدرتانی

از استادگر اتفاق، جناب آقای دکتر رزقی که باراهمایی های دلوزانه و ارزشمند خود، همواره در مسیر تحقیق این پایان نامه یار و راهنمای من بودند، نهایت سپاس و قدردانی را دارم.

از خانواده عزیزم که با محبت بی پایان، صبوری و حیات های بی دین شان، همواره پشتیبان من در طی این مسیر سخت و پرچالش بودند، صمیمانه سپاسگزارم.

سید محمد باذر حره
تابستان ۱۴۰۴

چکیده

مبدل‌های بینایی در سال‌های اخیر به یکی از ساختارهای اصلی در مدل‌های یادگیری عمیق برای پردازش تصاویر تبدیل شده‌اند. با این حال، ساختارهای کلاسیک این مبدل‌ها معمولاً شامل لایه‌های کاملاً متصل هستند که نیازمند تبدیل تانسورهای چندبعدی ورودی به بردارهای مسطح می‌باشند. این فرآیند منجر به افزایش چشمگیر پارامترها و از بین رفتن ساختار فضایی و روابط میان‌بعدی داده می‌شود. در این پژوهش، چارچوبی تانسوری برای طراحی مدل مبدل تصویری پنجه‌های متوجه پیشنهاد شده است که با بهره‌گیری از لایه‌های فشرده‌سازی تانسوری^۱ و رگرسیون تانسوری^۲، نگاشت‌های چندخطی میان ابعاد داده را مدل‌سازی می‌کند. این روش با حفظ ساختار چندبعدی تصویر، موجب کاهش قابل توجه تعداد پارامترها، حفظ اطلاعات ساختاری، و افزایش تفسیرپذیری مدل می‌شود. نتایج حاصل از پیاده‌سازی و ارزیابی مدل در مجموعه داده‌های استاندارد نشان می‌دهد که استفاده از ساختارهای تانسوری نه تنها پیچیدگی محاسباتی را کاهش می‌دهد، بلکه دقیق‌ترین طبقه‌بندی را نیز بهبود می‌بخشد.

swin transformer^۱

TCL^۲

TRL^۳

فهرست مطالب

و

فهرست جداول

ز

فهرست تصاویر

۱

پیشگفتار

۲

۱ مقدمه

۳

۱.۰۰.۱ آغاز هوش مصنوعی و هدف اصلی

۳

۲.۰۰.۱ دوره طلایی و پیشرفت‌های اولیه

۴

۳.۰۰.۱ انتظارات بیش از حد و ظهور عصر تاریک

۴

۴.۰۰.۱ عوامل اصلی عصر تاریک هوش مصنوعی

۵

۵.۰۰.۱ پایان عصر تاریک و بازگشت هوش مصنوعی

۶

۱.۱ انواع مدل یادگیری ماشین و شبکه‌های عصبی

۶

۱.۱.۱ یادگیری ماشین: مروری کلی

۶

۲.۱.۱ تقسیم‌بندی‌های اصلی در یادگیری ماشین

۷

۳.۱.۱ یادگیری ناظارت شده

۸

۴.۱.۱ یادگیری بدون ناظارت

۹

۵.۱.۱ یادگیری تقویتی

آ

۱۰	۶.۱.۱	معرفی چند مدل از الگوریتم یادگیری کلاسیک
۱۱	۷.۱.۱	نزدیک‌ترین همسایه
۱۲	۸.۱.۱	ماشین بردار پشتیبان
۱۳	۹.۱.۱	بیز ساده
۱۴	۱۰.۱.۱	شبکه‌های عصبی بازگشتی و شبکه‌های حافظه بلندمدت کوتاه‌مدت
۱۴	۱۱.۱.۱	شبکه‌های عصبی بازگشتی
۱۵	۱۲.۱.۱	ساختار و عملکرد شبکه‌های عصبی بازگشتی
۱۶	۱۳.۱.۱	شبکه‌های حافظه بلندمدت_کوتاه‌مدت
۱۶	۱۴.۱.۱	ناپدید شدن گرادیان در شبکه‌های بازگشتی
۱۷	۱۵.۱.۱	ظهور شبکه‌های حافظه بلندمدت_کوتاه‌مدت
	۱۶.۱.۱	راه حل شبکه‌های حافظه بلندمدت_کوتاه‌مدت برای پایداری جریان گرادیان‌ها
۱۸	۲.۱	ساختار شبکه‌های حافظه بلند_مدت کوتاه_مدت
۱۸	۱.۲.۱	وضعیت سلولی
۱۹	۲.۲.۱	دروازه‌ها
۲۰	۳.۲.۱	به روزرسانی وضعیت سلولی
۲۱	۴.۲.۱	مشکلات کلی شبکه‌های بازگشتی و ظهور مبدل‌ها
۲۱	۵.۲.۱	مشکل وابستگی ترتیبی در شبکه‌های بازگشتی
۲۲	۶.۲.۱	پیچیدگی محاسباتی و حافظه در شبکه‌های بلند مدت کوتاه مدت
۲۳	۷.۲.۱	مشکل پردازش وابستگی‌های غیرمتوالی
۲۳	۸.۲.۱	گرادیان‌های ناپایدار و مشکلات بهینه‌سازی

فهرست مطالب

۲۵	مشکلات ترجمه ماشینی و مبدل ها	۱.۲
۲۶	ظهور ترانسفورمرها	۲.۲
۲۷	معماری مبدل ها	۳.۲
۲۷	جاسازی	۱.۳.۲
۲۸	جاسازی موقعیتی	۲.۳.۲
۳۱	توجه	۳.۳.۲
۳۳	توجه چند سر	۴.۳.۲
۳۵	اتصال باقی مانده	۵.۳.۲
۳۶	نرمال سازی لایه ها	۴.۲
۳۸	اتصال باقی مانده	۱.۴.۲
۳۹	رمزگشا	۵.۲
۴۰	توجه چند سری ماسک شده	۶.۲
۴۱	مثال عددی توجه ماسک شده	۷.۲
۴۲	مبدل های بینایی	۸.۲
۴۲	جاسازی پچ ها در مبدل های بینایی	۱.۸.۲
۴۳	شکل پچ ها:	۲.۸.۲
۴۳	تعداد پچ ها:	۳.۸.۲
۴۴	بردارکردن هر پچ	۴.۸.۲
۴۵	اعمال لایه خطی	۹.۲
۴۶	توکن کلاس بندی	۱.۹.۲
۴۷	رمزگذار در مبدل های بینایی	۲.۹.۲
۴۸	مبدل پنجره ای متحرک	۱۰.۲
۵۰	قطعه بندی پچ در مبدل پنجره متحرک	۱.۱۰.۲

فهرست مطالب

۵۱	۲.۱۰.۲	توجه چند سر پنجره ای
۵۲	۳.۱۰.۲	توجه
۵۳	۴.۱۰.۲	پنجره متحرک جا به جا شده
۵۶	۵.۱۰.۲	پرسپتروون چند لایه
۵۷	۶.۱۰.۲	ترکیب پچ ها
۶۰		۳	روش های پیشنهادی
۶۳	۱.۰۰.۳	لایه فشرده سازی تانسوری
۶۵	۲.۰۰.۳	لایه رگرسیون تانسوری (TRL)
۶۶	۳.۰۰.۳	چرا در مبدل های بینایی از تانسور استفاده می کنیم؟
۶۷	۴.۰۰.۳	روش تانسوری مبدل پنجره متحرک:
۶۷	۵.۰۰.۳	پیاده سازی مرحله‌ی تعییه پچ با استفاده از فشرده سازی تانسوری
۷۰	۶.۰۰.۳	ماژول توجه سلف چند سری مبتنی بر پنجره به صورت تانسوری
۷۴	۷.۰۰.۳	توجه علامت دار
۷۷	۸.۰۰.۳	توجه مبتنی بر پنجره های جابه جا شده به صورت تانسوری
۷۸	۹.۰۰.۳	ادغام پچ ها به صورت تانسوری
۸۰	۱۰.۰۰.۳	مرحله‌ی طبقه بندی نهایی با استفاده از رگرسیون تانسوری
۸۳		۴	نتایج و تحلیل ها
۸۳	۱.۴	ارزیابی بر روی مجموعه داده CIFAR-10
۸۳	۱.۱.۴	خلاصه نتایج کمی
۸۴	۲.۱.۴	تحلیل نتایج
۸۵	۳.۱.۴	نمایش روند آموزش
۸۵	۴.۱.۴	جمع بندی

فهرست مطالب

۸۶	نتایج بر روی دیتاست MNIST	۲.۴
۸۶	خلاصه نتایج کمی	۱.۲.۴
۸۶	نمایش روند آموزش	۲.۲.۴
۸۷	تحلیل و بحث	۳.۲.۴
۸۹	نتایج بر روی دیتاست Tiny ImageNet	۳.۴
۸۹	خلاصه نتایج کمی	۱.۳.۴
۸۹	نمایش روند آموزش	۲.۳.۴
۸۹	تحلیل و بحث	۳.۳.۴
۹۳		کتابنامه
۹۹	آ جزئیات مدل‌ها و جدول پارامترها	

فهرست جداول

۲۱	۱.۲.۱ مقایسه ویژگی‌های RNN و LSTM
۸۴	۴.۱.۱ مقایسه عملکرد مدل اصلی و مدل پیشنهادی بر روی مجموعه داده CIFAR-10 بر حسب دقتهای Top-1 و Top-5.
۸۷	۴.۲.۲ مقایسه‌ی عملکرد مدل اصلی و مدل تانسوری بر روی MNIST (فقط Top-1 و Top-5).
۸۹	۴.۳.۳ مقایسه‌ی عملکرد مدل‌ها بر روی Tiny ImageNet

فهرست تصاویر

۱	کلاس بندی	۱.۱.۱
۸	رگرسیون	۱.۱.۲
۹	خوشه بندی	۱.۱.۳
۱۰	یادگیری تقویتی	۱.۱.۴
۱۱	الگوریتم نزدیکترین همسایه	۱.۱.۵
۱۲	الگوریتم ماشین بردار پشتیبان	۱.۱.۶
۱۵	شبکه‌های عصبی بازگشتی	۱.۱.۷
۱۹	مدل حافظه بلند مدت کوتاه مدت	۱.۲.۸
۲۸	معماری ترانسفورمراه	۲.۳.۱
۲۹	جاسازی کلمه ای	۲.۳.۲
۳۰	اضافه کردن جا سازی موقعیتی به جاسازی کلمه ای	۲.۳.۳
۳۲	نحوه به دست آوردن پرسشن، کلید و مقدار	۲.۳.۴
۳۳	توجه	۲.۳.۵
۳۴	توجه چند سر	۲.۳.۶
۴۰	رمزگشا در مبدل ها	۲.۵.۷
۴۳	بخش بندی تصاویر	۲.۸.۸

فهرست تصاویر

۴۵	۲.۹.۹ مبدل های بینایی
۴۸	۲.۹.۱۰ گوکن توجه در مبدل های بینایی
۴۹	۲.۹.۱۱ امبلد پنجره متحرک
۵۵	۲.۹.۱۲ چلا به جایی چرخه ای
۵۹	۲.۹.۱۳ ادغام پچ ها
۶۱	۳.۰.۱ شبکه عصبی پرسپترون چند لایه
۶۴	۳.۰.۲ لایه فشرده ساز تانسوری
۶۹	۳.۰.۳ تعییه پچ ها
۷۹	۳.۰.۴ ادغام پچ ها
۸۰	۴.۱.۱ روند تغییرات دقت Top-1 مدل اصلی Swin-Tiny بر روی مجموعه داده CIFAR
۸۵	۴.۱.۲ روند تغییرات دقت Top-1 مدل پیشنهادی Tensorized Swin بر روی مجموعه داده CIFAR-10
۸۷	۴.۲.۳ روند دقت Top-1 مدل اصلی Swin-Tiny بر روی MNIST
۸۸	۴.۲.۴ روند دقت Top-1 مدل تانسوری پیشنهادی بر روی MNIST
۹۰	۴.۳.۵ روند دقت Top-1 مدل اصلی Tiny Swin بر روی Tiny ImageNet
۹۱	۴.۳.۶ روند دقت Top-1 مدل تانسوری با بهینه ساز Adam بر روی Tiny ImageNet
۹۲	۴.۳.۷ روند دقت Top-1 مدل تانسوری با بهینه ساز AdamW بر روی Tiny ImageNet

پیش‌گفتار

پیش‌گفتار

مبدل‌های بینایی^۴ در سال‌های اخیر توجه زیادی در حوزه بینایی ماشین و یادگیری عمیق به خود جلب کرده‌اند. این مدل‌ها با وجود عملکرد قوی، دارای تعداد زیادی پارامتر هستند و معمولاً ساختار فضایی داده را از بین می‌برند. استفاده از روش‌های تانسوری را حلی برای این مسئله ارائه می‌دهد. در این پژوهش، چارچوبی مبتنی بر ساختارهای تانسوری برای بهینه‌سازی مدل‌های بینایی طراحی شده است. هدف از این کار، کاهش تعداد پارامترها، حفظ ساختار داده و بهبود تفسیرپذیری مدل است. از لایه‌های فشرده‌سازی تانسوری (TCL) و رگرسیون تانسوری (TRL) برای مدل‌سازی نگاشت‌های چندبعدی استفاده شده است.

این پایان‌نامه شامل چهار فصل است:

- فصل اول: بیان مقدمه، مسئله تحقیق، اهداف پژوهش و اهمیت موضوع. - فصل دوم: مرور پیشینه پژوهش، معرفی ساختار مدل‌های بینایی و مدل‌های پنجره متحرک. - فصل سوم: تشریح روش پیشنهادی، طراحی مدل تانسوری و پیاده‌سازی معماری مبدل پنجره متحرک به صورت تانسوری. - فصل چهارم: ارزیابی مدل، تحلیل نتایج حاصل از آزمایش‌ها و مقایسه عملکرد مدل پیشنهادی با مدل‌های پایه.

در پایان، جمع‌بندی، نتیجه‌گیری و پیشنهادهایی برای پژوهش‌های آینده ارائه شده است.

فصل ۱

مقدمه

در سال‌های اخیر، توسعه سریع فناوری‌های مبتنی بر داده و نیاز روزافزون به تحلیل هوشمند اطلاعات، موجب رشد چشمگیر الگوریتم‌های یادگیری ماشین و یادگیری عمیق شده است. در این میان، مدل‌های مبتنی بر معماری‌های شبکه‌های عصبی، به ویژه در حوزه‌هایی چون پردازش زبان طبیعی، بینایی ماشین، و تحلیل سری‌های زمانی، جایگاه ویژه‌ای یافته‌اند. یکی از تحولات بنیادی در این مسیر، ظهور معماری مبدل‌ها^۱ بوده است که با بهره‌گیری از مکانیزم توجه، رویکردی نوین و موثر برای مدل‌سازی وابستگی‌های طولانی در داده‌های ترتیبی ارائه می‌دهد. در ک جایگاه و کارکرد این معماری، مستلزم شناخت دقیق‌تر از مفاهیم بنیادین در هوش مصنوعی، یادگیری ماشین، شبکه‌های عصبی و ساختارهای بازگشتی است. از این‌رو، فصل حاضر به منظور ارائه بستر نظری لازم، به بررسی سیر تاریخی هوش مصنوعی، معرفی روش‌های یادگیری ماشین، مروری بر الگوریتم‌های کلاسیک، و تحلیل ساختار شبکه‌های بازگشتی از جمله شبکه‌های حافظه کوتاه مدت طولانی^۲ اختصاص دارد.

Transformer^۱
LSTM^۲

۱.۰۰.۱ آغاز هوش مصنوعی و هدف اصلی

هوش مصنوعی به عنوان شاخه‌ای از علوم کامپیوتر، در دهه ۱۹۵۰ با هدف ساخت سیستم‌ها و ماشین‌هایی که توانایی تقلید از هوش انسانی را دارند، آغاز شد. نخستین بار، مکارتی^۳ در سال ۱۹۵۶ این اصطلاح را به کار گرفت [۳۷] و هوش مصنوعی به عنوان علمی که در آن به مطالعه الگوریتم‌هایی برای تقلید رفتار انسانی می‌پردازد، شناخته شد. اهداف اولیه هوش مصنوعی شامل توانایی درک زبان، یادگیری، حل مسئله و تولید موجودات هوشمند بود. در این دوران پژوهش‌های تحقیقاتی زیادی به امید دستیابی به هوش مصنوعی عمومی^۴ شروع به کار کردند [۴۵، ۸].

۲.۰۰.۱ دوره طلایی و پیشرفت‌های اولیه

در دهه ۵۰ و ۶۰ میلادی، هوش مصنوعی^۵ به عنوان یکی از پرچمداران پژوهش‌های نوین شناخته می‌شد. الگوریتم‌های اولیه با تکیه بر روش‌های منطقی و ریاضیاتی برای حل مسئله و بازی‌های ساده توسعه یافتند؛ مانند انواع الگوریتم‌های جستجوی درختی^۶ که در این دوره به وجود آمدند و زمینه‌ساز اولین دستاوردهای هوش مصنوعی در بازی‌های تخته‌ای^۷ همچون شطرنج^۸ شدند [۴۴]. در این دوران، پیشرفت‌های بیشتری در پردازش زبان طبیعی^۹ و سیستم‌های خبره^{۱۰} نیز صورت گرفت که این امید را در دانشمندان و محققان تقویت کرد که دستیابی به هوش مصنوعی عمومی^{۱۱} بهزودی ممکن خواهد بود [۱۴].

John McCarthy^۳

AGI, Artificial General Intelligence^۴

Artificial Intelligence (AI)^۵

Tree Search Algorithms^۶

Board Games^۷

Chess^۸

Natural Language Processing (NLP)^۹

Expert Systems^{۱۰}

Artificial General Intelligence (AGI)^{۱۱}

۳.۰.۱ انتظارات بیش از حد و ظهر عصر تاریک

با وجود پیشرفت‌های هوش مصنوعی، محدودیت‌های تکنولوژی (مثل عدم وجود پردازنده‌های گرافیکی^{۱۲} پرقدرت در آن زمان) و همچنین کمبود داده‌های کافی برای آموزش مدل‌های پیچیده‌تر، باعث شد که بسیاری از پروژه‌های تحقیقاتی نتوانند به نتایج پیش‌بینی شده قابل قبول دست یابند. در نتیجه، هوش مصنوعی در دهه ۷۰ به مرحله‌ای از رکود وارد شد که به آن عصر تاریک هوش مصنوعی یا زمستان هوش مصنوعی^{۱۳} می‌گویند [۸، ۳۳]. در این دوران، بسیاری از پروژه‌ها تعطیل و سرمایه‌گذاری‌ها قطع شدند و دولت‌ها و سازمان‌های سرمایه‌گذار به دلیل عدم دستیابی به نتایج مطلوب از ادامه سرمایه‌گذاری منصرف شدند.

۴.۰.۱ عوامل اصلی عصر تاریک هوش مصنوعی

محدودیت‌های سخت‌افزاری: در آن زمان، سیستم‌های اولیه هوش مصنوعی به محاسبات سنگینی نیاز داشتند که با توان پردازشی محدود آن دوره همخوانی نداشت [۴۵].

کمبود داده‌ها: در آن زمان، دسترسی به داده‌های کافی برای آموزش مدل‌های پیچیده ممکن نبود و الگوریتم‌های موجود به داده‌های بیشتری نیاز داشتند تا بتوانند به درستی آموزش بینند و عملکرد مطلوبی داشته باشند [۸].

روش‌های محدود یادگیری: الگوریتم‌های اولیه به شدت به برنامه‌ریزی انسانی وابسته بودند و در بسیاری از موارد، مدل‌ها قادر به تعمیم به مسائل جدید نبودند و نمی‌توانستند تعمیم‌پذیری خیلی بالایی داشته باشند [۵۰].

GPU^{۱۲}

AI Winter^{۱۳}

۵.۰.۱ پایان عصر تاریک و بازگشت هوش مصنوعی

پس از چندین سال رکود و عدم سرمایه‌گذاری در حوزه هوش مصنوعی، سرانجام در دهه ۱۹۸۰ و ۱۹۹۰ عصر تاریک هوش مصنوعی با تحولات تکنولوژی و از همه مهم‌تر ظهر سیستم‌های خبره به پایان رسید [۱۴]. سیستم‌های خبره به عنوان یکی از اولین تلاش‌های موفق برای کاربردهای صنعتی در هوش مصنوعی به وجود آمدند. بر خلاف الگوریتم‌های اولیه، این سیستم‌ها از پایگاه بزرگ قواعد و قوانین ^{۱۴} استفاده می‌کردند. در سیستم‌های خبره، به جای تلاش برای شبیه‌سازی کلی هوش مصنوعی، بر حل مسائل تخصصی برای صنایع و سازمان‌ها تمرکز می‌شد. برای مثال، سیستم‌های خبره در پزشکی برای تشخیص بیماری‌ها و پیشنهاد درمان، در صنعت برای مدیریت و پیش‌بینی خرابی ماشین‌آلات، و در امور مالی برای تحلیل و ارزیابی ریسک کاربرد داشتند [۳۸]. هرچند این سیستم‌ها نمی‌توانستند در ک عمیق و هوشمندی عمومی را ایجاد کنند، اما برای رفع نیازهای پیچیده مناسب بودند. همزمان با موفقیت این سیستم‌ها، بهبودهای زیادی در ساخت افزارها و کاهش هزینه‌های پردازش به وجود آمد. در دهه‌های ۱۹۸۰ و ۱۹۹۰، کامپیوترها به تدریج قوی‌تر و مقرن به صرفه‌تر شدند و امکان پردازش داده‌های بیشتر و اجرای الگوریتم‌های پیچیده‌تر فراهم شد. این افزایش توان محاسباتی، نیاز به پردازش داده‌های بزرگ و پیچیده را برآورده کرد و در نتیجه دسترسی به داده‌ها و انجام محاسبات سنگین برای توسعه الگوریتم‌های جدید تسهیل شد. از سوی دیگر، پیشرفت‌های انجام‌شده در ذخیره‌سازی داده و رشد اینترنت باعث دسترسی گسترده‌تر به داده‌ها و منابع اطلاعاتی گردید [۴۵].

به این ترتیب، مجموعه‌ای از عوامل، شامل ظهور سیستم‌های خبره، افزایش قدرت پردازش و دسترسی به داده‌های بیشتر، منجر به بازگشت هوش مصنوعی شد. این دوره نه تنها پایان عصر تاریک هوش مصنوعی بود، بلکه راه را برای الگوریتم‌های یادگیری ماشین و توسعه شبکه‌های عصبی هموار کرد [۵۰].

۱.۱ انواع مدل یادگیری ماشین و شبکه‌های عصبی

یادگیری ماشین و شبکه‌های عصبی به عنوان دو زیرشاخه مهم از هوش مصنوعی، در سال‌های اخیر به طور گستردگی مورد توجه پژوهشگران و صنعت قرار گرفته‌اند. این مدل‌ها با هدف یادگیری الگوها و روابط موجود در داده‌ها توسعه یافته‌اند و امروزه در حوزه‌های مختلفی از جمله بینایی ماشین، پردازش زبان طبیعی، پیش‌بینی سری‌های زمانی و داده‌کاوی مورد استفاده قرار می‌گیرند. [۴۰، ۴۲]

۱.۱.۱ یادگیری ماشین: مروری کلی

یادگیری ماشین^{۱۵} شاخه‌ای از هوش مصنوعی است که به مدل‌های محاسباتی این امکان را می‌دهد الگوها را از داده‌ها به شکل خودکار یاد بگیرند و بتوانند تصمیم‌گیری کنند [۱۶، ۴۰]. در واقع، هدف یادگیری ماشین این است که مدل‌ها بتوانند از داده‌ها الگوها و روابط پنهان را استخراج کنند و به نتایج و تصمیم‌های قابل اعتماد دست یابند.

۲.۱.۱ تقسیم‌بندی‌های اصلی در یادگیری ماشین

به طور کلی، یادگیری ماشین به سه دسته اصلی تقسیم می‌شود:

- یادگیری با ناظارت^{۱۶}

- یادگیری بدون ناظارت^{۱۷}

- یادگیری تقویتی^{۱۸}

این طبقه‌بندی در بسیاری از کتاب‌ها و مراجع مهم یادگیری ماشین مطرح شده است [۴۲، ۴].

Machine Learning^{۱۵}

Supervised Learning^{۱۶}

Unsupervised Learning^{۱۷}

Reinforcement Learning^{۱۸}

۳.۱.۱ یادگیری نظارت شده

یادگیری نظارت شده یکی از رایج‌ترین روش‌ها در یادگیری ماشین شناخته می‌شود که در آن از مجموعه داده‌های برچسب‌گذاری شده برای آموزش مدل استفاده می‌کنیم [۲۴]. هدف این الگوریتم تشخیص الگوهای داده‌های ورودی است تا بتواند روی داده‌های جدید پیش‌بینی یا طبقه‌بندی انجام دهد. این نوع شامل دو دسته الگوریتم رگرسیون^{۱۹} و کلاس‌بندی^{۲۰} می‌شود.

کلاس‌بندی

در میان روش‌های یادگیری با نظارت، کلاس‌بندی^{۲۱} یکی از پرکاربردترین مسائل محسوب می‌شود. هدف در این مسئله، تشخیص هر نمونه ورودی به یکی از برچسب‌های گسسته از پیش تعریف شده است. مدل با استفاده از داده‌های آموزشی که شامل ویژگی‌ها و برچسب صحیح هستند، الگوهای حاکم بر داده را می‌آموزد و تلاش می‌کند بر اساس آن، داده‌های جدید را به دسته مناسب اختصاص دهد. این روش در کاربردهای متنوعی نظری تشخیص هرزنامه^{۲۲}، شناسایی بیماری‌ها، طبقه‌بندی تصاویر و تحلیل احساسات به‌طور گسترده مورد استفاده قرار می‌گیرد [۴۲، ۴].

رگرسیون

در مقابل، رگرسیون^{۲۳} به مسئله پیش‌بینی مقادیر پیوسته می‌پردازد. در این نوع از یادگیری نظارت شده، مدل می‌کوشد رابطه‌ای میان متغیرهای ورودی و خروجی‌های عددی برقرار کند تا بر اساس آن، بتواند خروجی نمونه‌های جدید را تخمین بزند. تفاوت اصلی رگرسیون با کلاس‌بندی در ماهیت خروجی است؛ به‌طوری‌که در رگرسیون، خروجی به جای دسته‌بندی، یک مقدار عددی خواهد بود. از جمله کاربردهای رایج این نوع مدل‌ها می‌توان به پیش‌بینی قیمت مسکن، تخمین میزان فروش، و پیش‌بینی

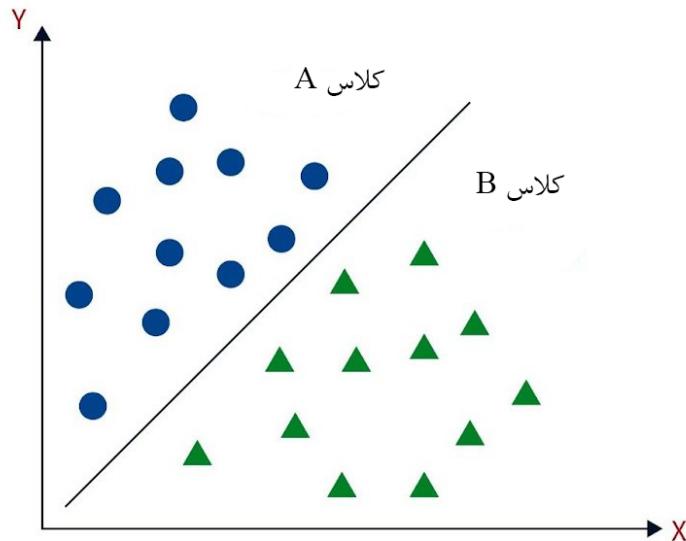
Regression^{۱۹}

Classification^{۲۰}

Classification^{۲۱}

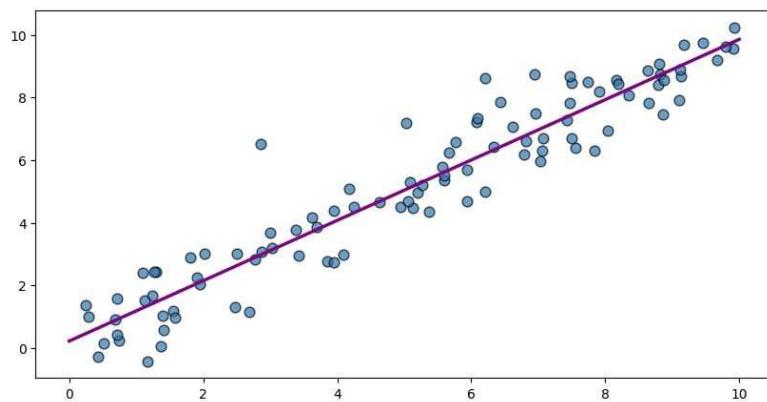
Spam Detection^{۲۲}

Regression^{۲۳}



شکل ۱.۱.۱: کلاس بندی

وضعیت آب و هوا اشاره کرد [۴۱].



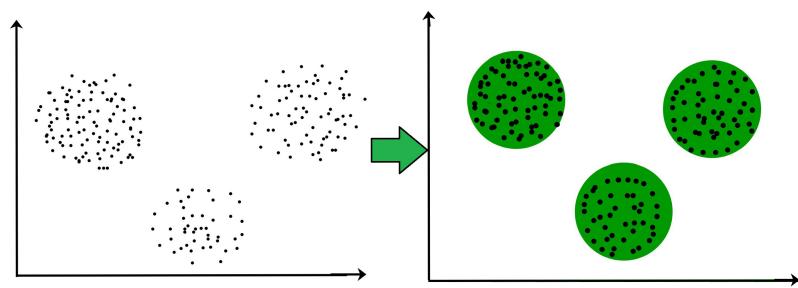
شکل ۱.۱.۲: رگرسیون

۴.۱.۱ یادگیری بدون نظارت

یادگیری بدون نظارت ^{۲۴} نوعی از روش‌های یادگیری ماشین است که در آن مدل بدون استفاده از برچسب‌های خروجی، سعی در کشف الگوهای ساختارها، روابط پنهان میان داده‌ها دارد [۴]. در

این رویکرد، داده‌های ورودی تنها شامل ویژگی‌ها هستند و هدف، گروه‌بندی یا استخراج ساختار درونی آن‌ها بدون دانش قبلی از دسته‌بندی صحیح است. برخلاف یادگیری با نظارت که مدل از پاسخ صحیح برای آموزش استفاده می‌کند، در یادگیری بدون نظارت، مدل باید به‌نهایی ساختارهای معنادار را در داده‌ها کشف کند.

از کاربردهای رایج یادگیری بدون نظارت می‌توان به خوشبندی^{۲۵}، کاهش ابعاد^{۲۶}، آشکارسازی ناهنجاری‌ها^{۲۷}، و استخراج ویژگی‌ها اشاره کرد.



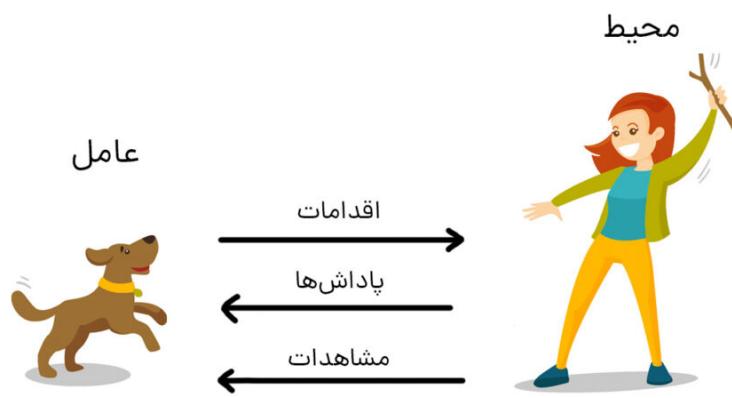
شکل ۱.۱.۳: خوشبندی

۵.۱.۱ یادگیری تقویتی

یادگیری تقویتی،^{۲۸} نوعی یادگیری بر پایه پاداش و تنبیه است که در آن مدل با محیط تعامل می‌کند و بر اساس پاداش یا تنبیه یاد می‌گیرد [۵۲]. برخلاف یادگیری نظارت شده و بدون نظارت، یادگیری تقویتی به مدل این امکان را می‌دهد تا از طریق آزمون و خطا بهترین راهکارها را برای انجام یک عمل یاد بگیرد. در این روش، مدل به جای برچسب، از یک تابع پاداش استفاده می‌کند که مشخص می‌کند چه اقداماتی باعث نتیجه بهینه می‌شود. از کاربردهای یادگیری تقویتی می‌توان به بازی‌ها^{۲۹}،

Clustering ^{۲۵}
Dimensionality Reduction ^{۲۶}
Anomaly Detection ^{۲۷}
reinforcement learning ^{۲۸}
Games ^{۲۹}

کنترل رباتیک^{۳۰} و سیستم‌های توصیه‌گر^{۳۱} اشاره کرد.



شکل ۱.۱.۴: یادگیری تقویتی

۶.۱.۱ معرفی چند مدل از الگوریتم یادگیری کلاسیک

الگوریتم‌های یادگیری کلاسیک، پایه و اساس بسیاری از پیشرفتهای اولیه در یادگیری ماشین را شکل داده‌اند. این الگوریتم‌ها با وجود سادگی نسبی، در بسیاری از کاربردها همچنان عملکرد قابل قبولی از خود نشان می‌دهند و در بسیاری از سامانه‌های عملیاتی مورد استفاده قرار می‌گیرند.

الگوریتم‌هایی مانند نزدیک‌ترین همسایه^{۳۲}، ماشین بردار پشتیبان^{۳۳}، بیز ساده^{۳۴} و درخت تصمیم^{۳۵} از جمله مشهورترین روش‌های کلاسیک یادگیری هستند که هر یک بر اساس اصول ریاضی و آماری متفاوتی طراحی شده‌اند. این مدل‌ها معمولاً برای مسائل طبقه‌بندی یا رگرسیون به کار می‌روند و از مزایایی چون پیاده‌سازی آسان، قابلیت تفسیر بالا و نیاز کمتر به تنظیمات پیچیده برخوردارند.

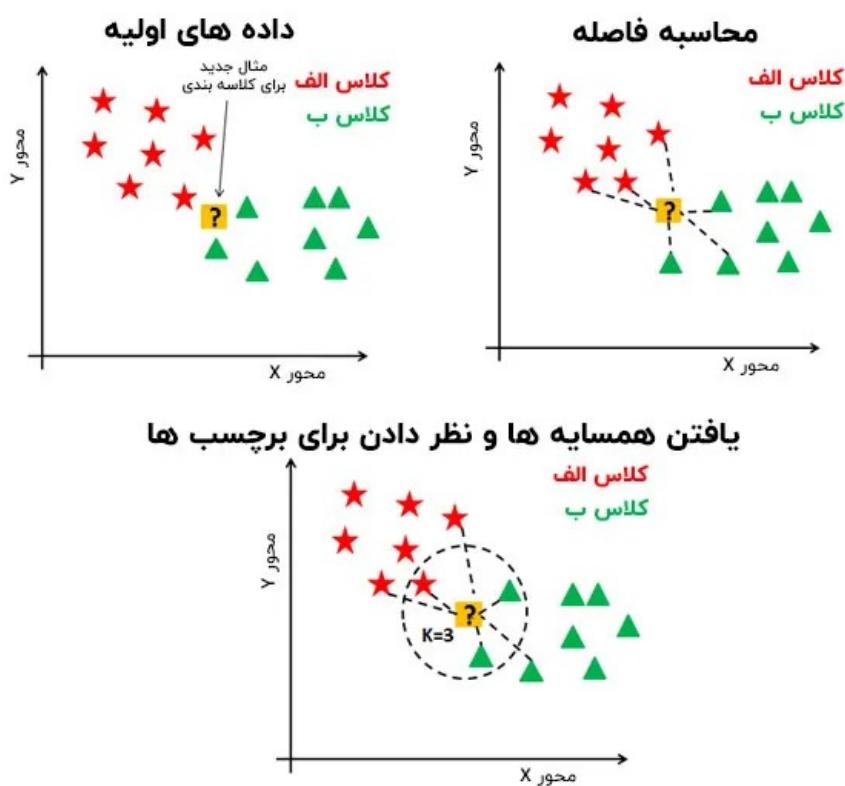
در این بخش، به معرفی اجمالی چند مورد از این الگوریتم‌ها پرداخته می‌شود تا زمینه درک

Robotic Control ^{۳۰}
Recommender Systems ^{۳۱}
k-nearest neighbors algorithm ^{۳۲}
Support vector machine ^{۳۳}
Naive Bayes ^{۳۴}
Decision Tree ^{۳۵}

عمیق‌تر روش‌های نوین‌تر یادگیری، مانند شبکه‌های عصبی و مدل‌های عمیق، فراهم گردد.

۷.۱.۱ نزدیک‌ترین همسایه

الگوریتم نزدیک‌ترین همسایه^{۳۶} یکی از روش‌های ساده و در عین حال کارآمد در یادگیری نظارت شده است که هم در دسته‌بندی و هم در رگرسیون کاربرد دارد [۴۰، ۱۲، ۷]. این الگوریتم برای پیش‌بینی دسته‌بندی یک نمونه جدید، به k نزدیک‌ترین داده‌ها در فضای ویژگی نگاه می‌کند و بر اساس اکثریت نزدیکی همسایه‌ها، آن را به یک دسته اختصاص می‌دهد.



شکل ۱.۱.۵: الگوریتم نزدیک‌ترین همسایه

از مهم‌ترین مزایای این الگوریتم می‌توان به سادگی و قابل فهم بودن اشاره کرد؛ چرا که تنها با اندازه‌گیری فاصله بین نقاط داده عمل می‌کند و بدون نیاز به آموزش یک مدل پیچیده، قابل استفاده

^{۳۶} k-Nearest Neighbors

است [۷]. همچنین، عملکرد خوب در داده‌های با تعداد ویژگی کم از دیگر نقاط قوت آن است، زیرا در مسائلی که تعداد ویژگی‌ها محدود است، این الگوریتم اغلب نتایج مطلوبی ارائه می‌دهد [۲۴]. در مقابل، الگوریتم نزدیک‌ترین همسایه دارای برخی معایب نیز هست. یکی از مهم‌ترین آن‌ها حساسیت به داده‌های پرت است؛ به طوری‌که نقاط پرت می‌توانند تأثیر قابل توجهی بر نتایج داشته باشند [۱۲]. علاوه بر این، کندی در داده‌های بزرگ از دیگر مشکلات این روش است، زیرا برای هر نقطه جدید نیاز به محاسبه فاصله با تمام داده‌ها دارد که در مجموعه داده‌های بزرگ منجر به بار محاسباتی بالا می‌شود [۴۰]. در نهایت، این الگوریتم در داده‌های با ابعاد بالا کارایی مناسبی ندارد و با افزایش تعداد ویژگی‌ها عملکرد آن به طور محسوسی افت می‌کند [۴۲].

۸.۱.۱ ماشین بردار پشتیبان

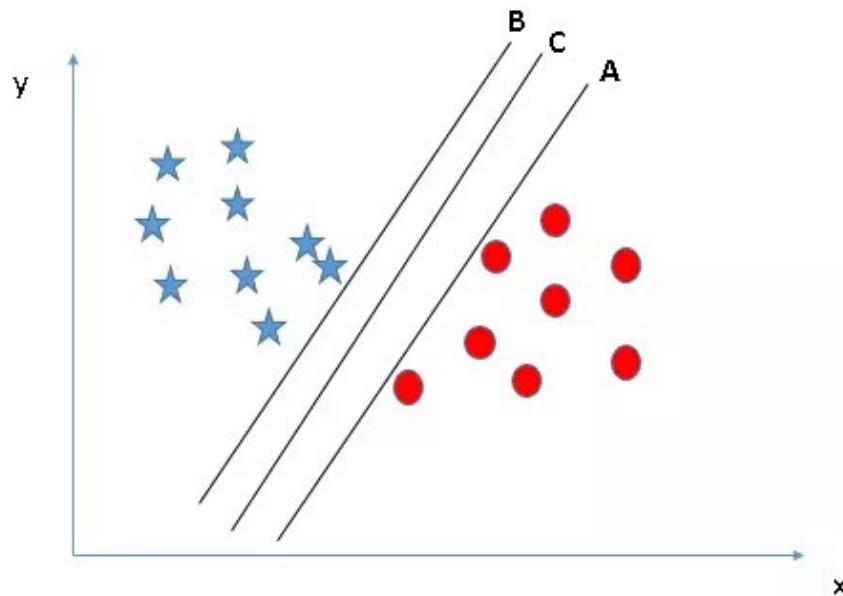
الگوریتم ماشین بردار پشتیبان^{۳۷} با یافتن یک ابرصفحه بهینه، داده‌ها را به کلاس‌های مختلف تقسیم می‌کند [۶، ۵۳]. این الگوریتم یک ابرصفحه به دست می‌آورد که هدف آن حداقل کردن فاصله میان داده‌های دو کلاس است و به این ترتیب می‌تواند طبقه‌بندی دقیقی داشته باشد. از مهم‌ترین مزایای ماشین بردار پشتیبان می‌توان به توانایی مقابله با داده‌های پیچیده و ابعاد بالا اشاره کرد؛ این الگوریتم قادر است به خوبی داده‌های چندبعدی و پیچیده را مدیریت کند و در چنین شرایطی همچنان عملکرد دقیقی داشته باشد [۵۳]. علاوه بر این، مقاومت در برابر بیش‌برازش^{۳۸} یکی دیگر از نقاط قوت آن است. با استفاده از توابع هسته^{۳۹}، داده‌های غیرخطی به فضای ویژگی‌های بالاتر نگاشت می‌شوند و این امکان فراهم می‌شود که مرز جداسازی بهتری میان کلاس‌ها ایجاد گردد [۶].

با وجود مزایای ذکر شده، ماشین بردار پشتیبان معایبی نیز دارد. نخستین مسئله، پیچیدگی محاسباتی آن است؛ آموختش این الگوریتم به دلیل نیاز به حل مسائل بهینه‌سازی، در حجم‌های بالای

Support Vector Machine, SVM^{۳۷}

Overfitting^{۳۸}

Kernels^{۳۹}



شکل ۱.۱.۶: الگوریتم ماشین بردار پشتیبان

داده بسیار زمان برخواهد بود [۴۲]. همچنین، کارایی پایین در داده‌های پرت از دیگر نقاط ضعف آن است، چرا که وجود تعداد زیادی داده پرت می‌تواند به شکل قابل توجهی دقت مدل را کاهش دهد [۴۳].

۹.۱.۱ بیز ساده

بیز ساده^{۴۰} مبتنی بر قضیه بیز^{۴۱} است و فرض می‌کند ویژگی‌ها به صورت شرطی مستقل از هم هستند [۴۰، ۱۰]. این مدل برای اولین بار در حوزه پردازش متن به کار رفت و هنوز هم در بسیاری از کاربردها مانند طبقه‌بندی ایمیل و تحلیل احساسات مورد استفاده قرار می‌گیرد [۳۶]. در بیز ساده بر اساس احتمالات محاسبه می‌شود که یک نمونه جدید به کدام دسته تعلق دارد. این الگوریتم بر اساس قضیه بیز، احتمال تعلق یک نمونه به هر دسته را به ازای هر ویژگی محاسبه کرده و در نهایت بالاترین احتمال را به عنوان جواب نهایی در نظر می‌گیرد [۲].

Bayes Naive^{۴۰}
Bayes' theorem^{۴۱}

از مهمترین مزایای بیز ساده می‌توان به سرعت بالا اشاره کرد؛ چرا که به دلیل محاسبات ساده و فرض استقلال ویژگی‌ها، این الگوریتم بسیار سریع و کم حجم بوده و برای مسائل با داده‌های زیاد یا در شرایط نیاز به پاسخ بلاذرنگ کارایی بالایی دارد [۳۶]. علاوه بر این، کارایی در داده‌های کوچک از دیگر نقاط قوت آن است، زیرا حتی با وجود حجم محدود داده‌ها، مدل همچنان قادر است عملکرد قابل قبولی ارائه دهد [۴۲].

با وجود مزایا، بیز ساده محدودیت‌هایی هم دارد. یکی از مهمترین آن‌ها فرض استقلال ویژگی‌هاست؛ فرضی که در بسیاری از مسائل واقعی برقرار نیست و می‌تواند باعث کاهش دقت مدل شود [۱۰]. علاوه بر این، این الگوریتم نسبت به کیفیت داده‌ها بسیار حساس است و حساسیت به داده‌های نادرست یا پرت می‌تواند موجب کاهش چشمگیر دقت آن شود [۴].

۱۰.۱.۱ شبکه‌های عصبی بازگشتی و شبکه‌های حافظه بلندمدت کوتاه‌مدت

شبکه‌های عصبی بازگشتی ^{۴۲} و مدل‌هایی با حافظه بلندمدت – کوتاه‌مدت ^{۴۳} با هدف پردازش داده‌های ترتیبی و وابسته به زمان توسعه یافتند [۲۱، ۴۹]. این مدل‌ها به ویژه در تحلیل زبان طبیعی، پردازش صوت و پیش‌بینی سری‌های زمانی بسیار موفق عمل کرده‌اند؛ زیرا قادر به حفظ اطلاعات گذشته هستند و از این اطلاعات برای پیش‌بینی در لحظهٔ حال و آینده استفاده می‌کنند [۱۵].

۱۱.۱.۱ شبکه‌های عصبی بازگشتی

مدل‌های اولیه شبکه‌های عصبی، مانند شبکه‌های چندلایه ^{۴۴}، قادر به پردازش داده‌های مستقل و ثابت بودند و نمی‌توانستند وابستگی‌های زمانی را یاد بگیرند ^{۴۵}. در بسیاری از مباحث دنیای واقعی مانند تحلیل متن، صدا و داده‌ها به توالی خاصی وابسته هستند. به همین دلیل، شبکه‌های عصبی بازگشتی معرفی شدند تا بتوانند از اطلاعات پیشین در پردازش داده‌های بعدی استفاده کنند ^{۴۶}.

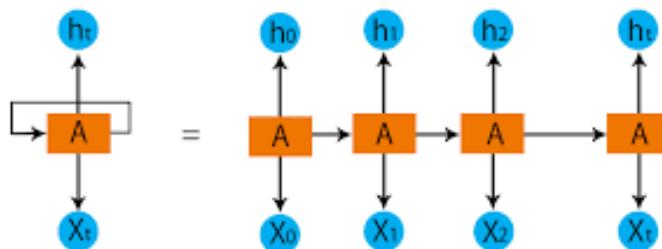
RNN^{۴۲}

LSTM^{۴۳}

MLP^{۴۴}

۱۲.۱.۱ ساختار و عملکرد شبکه‌های عصبی بازگشتی

شبکه‌های عصبی بازگشتی^{۴۵} دارای حلقه بازگشتی هستند که به مدل این امکان را می‌دهد اطلاعات را در توالی نگه دارد و در هر گام زمانی، ورودی فعلی x_t و وضعیت قبلی h_{t-1} را به عنوان ورودی دریافت کند [۱۶].



شکل ۱.۱.۷: شبکه‌های عصبی بازگشتی

$$h_t = \sigma(W \cdot x_t + U \cdot h_{t-1} + b) \quad (1.1.1)$$

در این رابطه، h_t نشان‌دهنده وضعیت مخفی یا حالت در گام زمانی t است. ماتریس W وزن‌هایی را مشخص می‌کند که به ورودی x_t اعمال می‌شوند، در حالی که ماتریس U وزن‌های مربوط به وضعیت قبلی h_{t-1} را در نظر می‌گیرد. مقدار b بایاس مدل است و در نهایت، تابع σ نقش تابع فعال‌سازی را ایفا می‌کند که معمولاً از نوع تانژانت هیپربولیک یا سیگموید انتخاب می‌شود. به کمک این ساختار، مدل می‌تواند اطلاعات گذشته را در خود ذخیره کرده و در پردازش‌های بعدی از آن‌ها بهره ببرد.

از مهم‌ترین مزایای شبکه‌های عصبی بازگشتی می‌توان به حفظ وابستگی زمانی اشاره کرد. این مدل‌ها قادر به پردازش توالی‌های طولانی هستند و می‌توانند اطلاعات پیشین را در طول توالی به خاطر

بسپارند و در تصمیم‌گیری‌های بعدی به کار گیرند [۱۳]. همچنین، کاربردهای گسترده در داده‌های ترتیبی از دیگر نقاط قوت آن‌هاست؛ چرا که در حوزه‌هایی مانند تحلیل زبان طبیعی، پیش‌بینی سری‌های زمانی و پردازش صوت، نتایج بسیار موفقی داشته‌اند [۱۵].

با وجود این مزایا، شبکه‌های عصبی بازگشتی معایبی هم دارند. یکی از اصلی‌ترین مشکلات آن‌ها ناپدید شدن و انفجار گرادیان^{۴۶} است. در فرایند آموزش با روش پسانشان^{۴۷}، اگر توالی داده‌ها طولانی باشد، گرادیان‌ها ممکن است بسیار کوچک یا بسیار بزرگ شوند که این امر منجر به ناپایداری در آموزش و کاهش دقت می‌شود [۲۰]. علاوه بر این، این شبکه‌ها محدودیت در پردازش توالی‌های بسیار بلند دارند؛ به این معنا که در حفظ وابستگی‌های طولانی مدت دچار مشکل شده و عملکرد ضعیفی نشان می‌دهند [۱۶، ۲۱].

۱۳.۱.۱ شبکه‌های حافظه بلندمدت – کوتاه‌مدت

شبکه‌های حافظه بلندمدت – کوتاه‌مدت به عنوان یک راه حل برای یکی از بزرگ‌ترین مشکلات شبکه‌های عصبی بازگشتی معرفی شدند [۲۱]. یکی از برجسته‌ترین مشکلات موجود در شبکه‌های عصبی بازگشتی، معضل ناپدید شدن گرادیان بود که مانع یادگیری وابستگی‌های بلندمدت می‌شد [۱۶، ۲۰]. برای درک عمیق‌تر این مسئله، ابتدا به توضیح مشکل ناپدید شدن گرادیان و سپس راهکار شبکه‌های حافظه بلندمدت – کوتاه‌مدت می‌پردازیم.

۱۴.۱.۱ ناپدید شدن گرادیان در شبکه‌های بازگشتی

شبکه‌های عصبی بازگشتی برای پردازش داده‌های ترتیبی از حلقه‌های بازگشتی بهره می‌برند. در فرایند آموزش شبکه‌های عصبی بازگشتی، از الگوریتم پسانشان خطأ از طریق زمان^{۴۸} استفاده می‌شود که گرادیان‌ها را جهت به روزرسانی وزن‌ها محاسبه می‌کند. [۴۹]

Vanishing and Exploding Gradient^{۴۶}

Backpropagation^{۴۷}

Backpropagation Through Time^{۴۸}

با این حال، شبکه‌های عصبی بازگشتی در یادگیری وابستگی‌های بلندمدت معمولاً^{۴۸} ناکام می‌مانند. علت اصلی این امر شامل موارد زیر است:

- ضرایب‌های بازگشتی کوچک‌تر از ۱: در فرایند محاسبه گرادیان‌ها، اگر مقدار مشتقات یا ضرایب در هر مرحله کوچک‌تر از ۱ باشد، ضرب مکرر این ضرایب در طول توالی منجر به کوچک‌شدن گرادیان‌ها به سمت صفر می‌شود؛ پدیده‌ای که به ناپدید شدن گرادیان^{۴۹} معروف است [۲۰].

فرمول کلی گرادیان در زمان t به صورت زیر است:

$$\frac{\partial L}{\partial W} = \prod_{k=1}^t \frac{\partial h_k}{\partial h_{k-1}} \cdot \frac{\partial h_t}{\partial L} \quad (1.1.2)$$

در این فرمول، $\frac{\partial h_k}{\partial h_{k-1}}$ ممکن است مقداری کوچک‌تر از ۱ باشد، و ضرب مکرر آن در طول توالی باعث کاهش شدید مقدار گرادیان می‌گردد.

- تأثیر مستقیم بر وزن‌ها: زمانی که گرادیان‌ها به صفر نزدیک می‌شوند، وزن‌های مدل عملاً به روزرسانی نمی‌شوند و این امر مانع از یادگیری وابستگی‌های طولانی مدت در داده‌ها می‌شود [۱۶].

۱۵.۱.۱ ظهور شبکه‌های حافظه بلندمدت – کوتاه‌مدت

در سال ۱۹۹۷، شبکه‌های حافظه بلندمدت – کوتاه‌مدت معرفی شد. [۲۱]. انگیزه اصلی توسعه این شبکه حل مشکل ناپدید شدن گرادیان در شبکه‌های عصبی بازگشتی بود. این مشکل در مسائل یادگیری داده‌های ترتیبی طولانی مانع می‌شد شبکه‌های عصبی بازگشتی وابستگی‌های بلندمدت را به درستی فراگیرد.

۴۹ vanishing gradient

۱۶.۱.۱ راه حل شبکه های حافظه بلندمدت - کوتاه مدت برای پایداری جریان گرادیان ها

با معرفی شبکه حافظه بلند مدت، کوتاه مدت^{۵۰} در شبکه های بازگشتی، جریان گرادیان ها را در طول توالی پایدار نگه می دارد. این کار از طریق اضافه کردن وضعیت سلولی^{۵۱} و دروازه ها^{۵۲} به ساختار شبکه های عصبی بازگشتی انجام می شود. [۱۵]. این اجزا به این شبکه این امکان را می دهند:

۱. اطلاعات غیر ضروری را فراموش کند،

۲. اطلاعات مهم جدید را اضافه کند،

۳. اطلاعات مهم قبلی را حفظ کند.

۲۰.۱ ساختار شبکه های حافظه بلند- مدت کوتاه- مدت

شبکه های حافظه بلند- مدت شامل اجزای جدیدی است که به آن امکان مدیریت بهتر اطلاعات را می دهد:

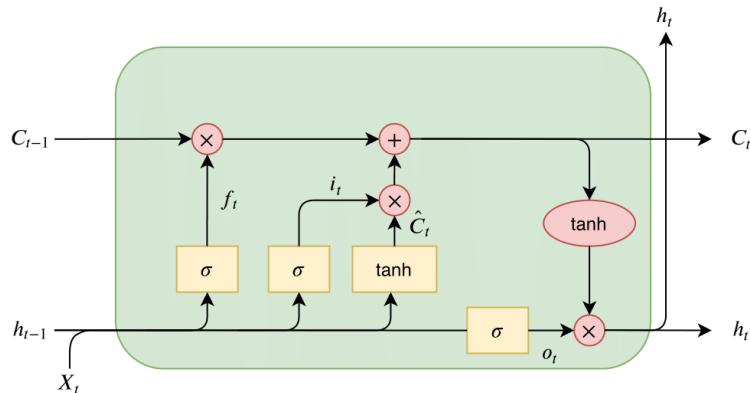
۱۰.۱ وضعیت سلولی

مسیر اصلی ذخیره اطلاعات در شبکه های حافظه بلند- مدت کوتاه مدت است که می تواند اطلاعات مهم را در طول توالی حفظ کند. برخلاف شبکه های عصبی بازگشتی که عمدهاً بر خروجی های بازگشتی h_t متکی است، شبکه حافظه بلند مدت، کوتاه مدت یک مسیر جداگانه برای عبور اطلاعات از وضعیت سلولی دارد که به حفظ گرادیان ها کمک شایانی می کند [۲۱].

long short-term memory^{۵۰}

Cell State^{۵۱}

Gates^{۵۲}



شکل ۱.۲.۸ : مدل حافظه بلند مدت کوتاه مدت

۲.۲.۱ دروازه‌ها

دوازه‌ها نقش فیلترهای اطلاعاتی را دارند که جریان اطلاعات را در طول فرایند یادگیری کنترل می‌کنند.

دوازه فراموشی^{۵۳} تعیین می‌کند چه اطلاعاتی از وضعیت سلولی باید حذف شود [۱۵].

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (1.2.3)$$

در این معادله، f_t بیانگر میزان فراموشی برای هر مؤلفه از وضعیت سلولی در گام زمانی جاری است. این مقدار با استفاده از تابع فعال‌ساز سیگموید σ محاسبه می‌شود که خروجی آن بین صفر تا یک قرار دارد. هرچه مقدار f_t به صفر نزدیک‌تر باشد، آن مؤلفه از وضعیت سلولی بیشتر فراموش می‌شود؛ و بالعکس، مقدار نزدیک به یک نشان‌دهنده حفظ کامل آن اطلاعات در حافظه سلولی است.

دوازه ورودی^{۵۴}: تعیین می‌کند چه اطلاعات جدیدی باید به وضعیت سلولی اضافه شود:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (1.2.4)$$

Forget Gate^{۵۳}
Input Gate^{۵۴}

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (1.2.5)$$

که در آن i_t میزان اطلاعات جدید و \tilde{C}_t مقدار جدید قابل اضافه شدن به وضعیت سلولی را نشان می دهد.

دروازه خروجی ^{۵۵}:

تعیین می کند چه اطلاعاتی از وضعیت سلولی به خروجی منتقل شود:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o),$$

$$h_t = o_t \cdot \tanh(C_t).$$

۳.۲.۱ به روزرسانی وضعیت سلولی

وضعیت سلولی C_t با استفاده از اطلاعات جدید و قدیمی به روزرسانی می شود:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \quad (1.2.6)$$

این ساختار باعث می شود اطلاعات قدیمی مهم حفظ و داده های غیر ضروری حذف شوند.

علت پایداری گرادیان در شبکه های حافظه بلند-مدت کوتاه مدت

- حذف ضربه های مکرر: برخلاف شبکه های بازگشتی که به ضربه های مکرر وزن ها و گرادیان ها وابسته است، شبکه های حافظه بلند-مدت کوتاه مدت با مسیر جداگانه وضعیت سلولی، از کاهش نمایی گرادیان جلوگیری می کند ^[۲۰].

- استفاده از توابع سیگموید و تانژانت هیپربولیک: توابع سیگموید در دروازه ها و تانژانت هیپربولیک در وضعیت سلولی مقادیر را محدود می کنند و مانع از انفجار گرادیان می شوند ^[۱۵].

. [۱۶]

- مدیریت اطلاعات توسط دروازه‌ها: دروازه‌های فراموشی و ورودی به مدل اجازه می‌دهند تنها اطلاعات مهم حفظ شود و داده‌های غیرضروری حذف شوند. این موضوع از پیچیدگی محاسباتی غیرضروری جلوگیری می‌کند [۲۱].

LSTM	RNN	ویژگی
برطرف شده	وجود دارد	مشکل ناپدید شدن گرادیان
بسیار خوب	محدود به وابستگی کوتاه‌مدت	توانایی حفظ وابستگی‌های طولانی‌مدت
دارای دروازه‌های فراموشی، ورودی و خروجی	ندارد	ساختار دروازه‌ها
پایدار	ضعیف	پایداری گرادیان

جدول ۱.۲۰.۱ : مقایسه ویژگی‌های LSTM و RNN

۴.۲.۱ محدودیت‌های شبکه‌های بازگشته

شبکه‌های بازگشته داده‌ها را به صورت ترتیبی پردازش می‌کنند؛ به این معنی که برای پردازش داده‌های گام زمانی t ، باید تمامی داده‌های قبلی ($1 - t$) پردازش شده باشند [۲۱، ۴۹]. این ویژگی موجب می‌شود که پردازش داده‌ها به صورت موازی امکان‌پذیر نباشد و همین امر زمان محاسباتی را افزایش دهد. در داده‌های بلند، مانند متن‌های طولانی یا سری‌های زمانی بزرگ، این مشکل نمود بیشتری پیدا می‌کند. علاوه بر این، پردازش خطی داده‌ها موجب کندی در آموزش و استنتاج مدل‌ها می‌شود، به‌ویژه زمانی که حجم داده‌ها بسیار زیاد باشد.

با وجود پیشرفت‌هایی که در شبکه‌های حافظه بلندمدت – کوتاه‌مدت نسبت به شبکه‌های بازگشته معمولی حاصل شده است، این مدل‌ها همچنان در یادگیری وابستگی‌های بسیار طولانی محدودیت دارند [۲۰]. به طور مثال، در جملات طولانی ممکن است ارتباط میان کلمات ابتدایی و انتهايی اهمیت بالایی داشته باشد، اما ظرفیت این شبکه‌ها در حفظ چنین اطلاعاتی محدود است و با افزایش طول توالی، دقت مدل کاهش می‌یابد. همچنین داده‌های ابتدایی توالی به مرور اهمیت خود را از دست می‌دهند، زیرا گرادیان‌ها به تدریج ضعیفتر می‌شوند.

از سوی دیگر، ساختار پیچیده شبکه‌های حافظه بلندمدت – کوتاه‌مدت، که شامل چندین ماتریس

ضرب برای دروازه‌های ورودی، خروجی و فراموشی و بروزرسانی وضعیت سلول است، باعث افزایش نیاز به حافظه و محاسبات می‌شود [۱۶]. در مقیاس‌های بزرگ، این موضوع به افزایش هزینه محاسباتی و کندی اجرای مدل منجر می‌گردد.

شبکه‌های بازگشتی همچنین در پردازش وابستگی‌های غیرمتوالی با محدودیت‌هایی روبرو هستند. برای مثال، در ترجمه زبان یا تحلیل متون طولانی، ممکن است کلمه‌ای در ابتدای جمله با کلمه‌ای در انتهای آن ارتباط معنایی داشته باشد [۲]. این نوع وابستگی‌های غیر محلی برای شبکه‌های بازگشتی به سختی قابل یادگیری است.

مشکل دیگر به پایداری گرادیان‌ها بازمی‌گردد. با وجود بهبودهایی که شبکه‌های حافظه بلندمدت – کوتاه‌مدت در این زمینه نسبت به شبکه‌های بازگشتی معمولی داشته‌اند، همچنان در توالی‌های بسیار بلند گرادیان‌ها ممکن است دچار کاهش یا انفجار شوند که این مسئله پایداری آموزش و دقت مدل را تحت تأثیر قرار می‌دهد [۲۰]. افزون بر این، در مسائل پیچیده یافتن مینیمم مناسب تابع هزینه دشوار است و بهینه‌سازی مدل با چالش روبرو می‌شود.

در نهایت، شبکه‌های بازگشتی برای مسائل پیچیده‌تر به ظرفیت و سرعت بالاتری نیاز دارند، اما محدودیت در حافظه و پردازش مانع از دستیابی به این هدف می‌شود. همچنین این مدل‌ها در داده‌های چندوجهی^{۵۶}، مانند داده‌هایی که ترکیبی از متن، صوت و تصویر هستند، عملکرد مناسبی ندارند؛ زیرا توانایی لازم برای پردازش موازی این نوع اطلاعات را در اختیار ندارند.

در مجموع، وابستگی ترتیبی در شبکه‌های بازگشتی مانع اساسی برای استفاده از این مدل‌ها در مسائل پیچیده و بزرگ بود که در نهایت به ظهور مبدل‌ها منتهی شد [۵۴]. مبدل‌ها با طراحی مبتنی بر موازی‌سازی و مکانیزم توجه^{۵۷}، این محدودیت را برطرف کرده و راه حلی کارآمدتر برای پردازش داده‌های ترتیبی ارائه دادند.

فصل ۲

پیشینه پژوهش

با توجه به مشکلات مطرح شده شبکه های بازگشتی، معماری جدیدی به نام مبدل ها^۱ مطرح شد ظهور مبدل ها یکی از تحولات اساسی در حوزه پردازش زبان طبیعی^۲ و یادگیری ماشین به شمار می رود.^[۵۴، ۲] این مدل ها باعث تغییرات عمده ای در نحوه ساخت و آموزش مدل های زبانی و همچنین در بسیاری از کاربردهای دیگر یادگیری ماشین شده اند و توانستند بسیاری از مشکلات مدل های قبلی را حل کنند.^[۴۸، ۹]

۱.۲ مشکلات ترجمه ماشینی و مبدل ها

در ابتدا، ترجمه ماشینی^۳ یک چالش اساسی در زمینه پردازش زبان طبیعی بود. مدل های اولیه ای مانند مدل های مبتنی بر قواعد^۴ برای ترجمه استفاده می شدند که در آن ها، ترجمه ها به صورت دستی با استفاده از قواعد زبانی مشخص تنظیم می شدند.^[۴۲، ۲۲] این روش ها هر چند دقیق بودند، اما

Transformer^۱

NLP^۲

machine translation^۳

Rule-based Models^۴

محدودیت‌های زیادی داشتند و نمی‌توانستند ویژگی‌های پیچیده‌تر زبان را مدل‌سازی کنند. سپس مدل‌های آماری^۵ معرفی شدند [۵، ۲۶]. این مدل‌ها از داده‌های ترجمه‌شده برای آموزش مدل‌های آماری استفاده می‌کردند که احتمال ترجمه صحیح را براساس شواهد آماری محاسبه می‌کردند. مدل‌هایی مانند مدل‌های ترجمه آماری مبتنی بر جمله^۶ [۲۷] از این نوع بودند که قادر به ترجمه جملات بهتر از مدل‌های مبتنی بر قواعد بودند، اما هنوز هم در ترجمه‌های پیچیده با مشکلاتی روبرو بودند.

بعد از این مدل‌ها، مدل‌های بازگشتی^۷ به وجود آمدند که مشکلات آن‌ها در فصل گذشته بیان شد [۱۳، ۵۱]. در نهایت، این مشکلات باعث به وجود آمدن ترانسفورمرها شد [۲].

۲.۲ ظهر ترانسفورمرها

در سال ۲۰۱۷، مقاله‌ای توسط گوگل^۸ منتشر شد که مفهوم جدیدی به نام مبدل‌ها^۹ را معرفی کرد [۵۴]. این مقاله به موضوع ترجمه ماشینی پرداخت و نشان داد که با استفاده از مکانیزم توجه می‌توان بسیاری از مشکلات مدل‌های قبلی را حل کرد [۳۵].

مدل‌های ترانسفورمر برخلاف مدل‌های قبلی که از پردازش سریالی استفاده می‌کردند، از پردازش موازی بهره می‌برند. این ویژگی به ترانسفورمرها اجازه می‌دهد که به‌طور همزمان به تمام بخش‌های ورودی توجه کنند. این قابلیت باعث شد که مبدل‌ها در پردازش تصویر و متن بسیار سریع‌تر و دقیق‌تر از مدل‌های قبلی عمل کنند [۵۴].

Statistical Models ^۵
Phrase-based Statistical Models ^۶
Recurrent Models ^۷
google ^۸
Transformers ^۹

۳.۲ معماری مبدل‌ها

در تصویر ۲.۳.۱، معماری مبدل نمایش داده شده است و بخش‌ها و اجزای مختلف آن مشخص شده است. معماری ترانسفورمر از دو بخش اصلی تشکیل شده است:

- رمزگذار^{۱۰}: وظیفه رمزگذار این است که داده ورودی را دریافت کند و ویژگی‌های آن را استخراج کند.
- رمزگشای^{۱۱}: وظیفه رمزگشای این است که ویژگی‌های استخراج شده را به زبان مقصد تبدیل کند.

۱.۳.۲ جاسازی

در زبان طبیعی، کلمات به شکل رشته‌های متنی هستند مانند کتاب، ماشین و ... کامپیوترها نمی‌توانند به‌طور مستقیم این کلمات را به شکل رشته‌های متنی پردازش کنند. به همین دلیل، در یادگیری ماشین این کلمات را به شکل یک بردار نمایش می‌دهیم. این بردار بیانگر آن کلمه در مدل است تا ماشین بتواند آن کلمه را پردازش کند.

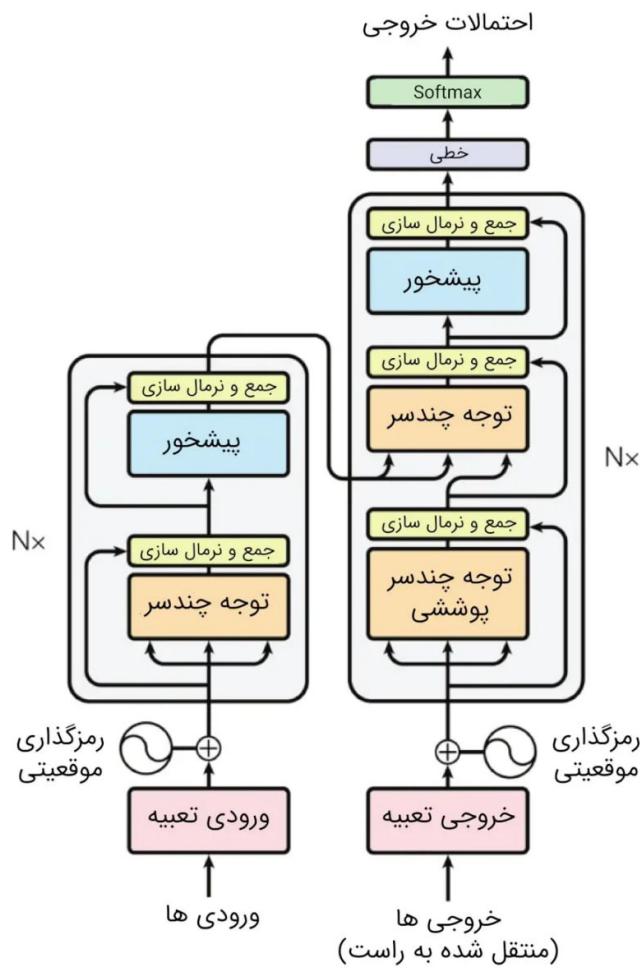
این بردارها ویژگی‌های کلمه را در فضای عددی نمایش می‌دهند. روش‌های مختلفی برای تبدیل متن به بردار وجود دارند. از جمله این روش‌ها می‌توان به روش‌های Word2Vec [۳۹] و GloVe [۴۷] اشاره کرد.

همان‌طور که در شکل ۲.۳.۲ نشان داده شده است، هر کلمه که به صورت توکن است، ابتدا در دیکشنری تعریف شده پیدا می‌شود و پس از پیدا شدن در دیکشنری، با استفاده از روش‌های تعبیه کردن^{۱۲}، هر کلمه به برداری از اعداد تبدیل می‌شود. این جاسازی‌ها شباهت‌های معنایی بین کلمات

Encoder^{۱۰}

Decoder^{۱۱}

Embedding^{۱۲}

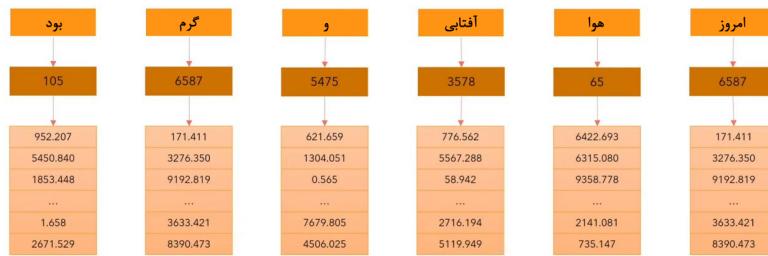


شکل ۲.۳.۱: معماری ترانسفورمرها

را مدل‌سازی می‌کنند و کلماتی که از نظر معنایی شبیه به هم هستند، بردار آن‌ها نیز به یکدیگر نزدیک‌تر است. به این ترتیب، کلمات برای مدل‌ها و شبکه‌های عصبی قابل فهم می‌شوند [۴۷، ۳۹].

۲.۳.۲ جاسازی موقعیتی

هر کلمه را به برداری از اعداد که برای مدل قابل فهم باشد، تبدیل کرده‌ایم. اما مدل‌های ترانسفورمر نمی‌توانند جایگاه هر کلمه را تشخیص دهند. در مدل‌های ترانسفورمر، برخلاف مدل‌های بازگشتی، به دلیل اینکه کلمات به صورت موازی وارد می‌شوند، نیاز داریم تا جایگاه هر کلمه را بدانیم. به طور



شکل ۲.۳.۲ : جاسازی کلمه ای

مثال، در جمله «من تو را دوست دارم» باید به طور دقیق بدانیم که «من» کلمه اول جمله است، «تو» کلمه دوم جمله است و

حال باید به مدل توالی این کلمات را بفهمانیم. بنابراین، نیاز داریم به مدل یک سری اطلاعات اضافی بدهیم به طوری که مدل توالی کلمات را یاد بگیرد. روش‌های مختلفی برای اضافه کردن جاسازی موقعیتی^{۱۳} به مدل وجود دارد. در ترانسفورمرها از روش جاسازی موقعیت سینوسی^{۱۴} استفاده می‌شود [۵۴].

این روش قابل یادگیری نیست و صرفاً از یک سری فرمول‌های ساده برای جاسازی موقعیتی استفاده می‌کند. برای موقعیت مشخص^{۱۵} در توالی و بعد در فضای برداری، تعییه موقعیتی به صورت زیر تعریف می‌شود.

و برای مقادیر زوج:

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \quad (2.3.1)$$

و برای مقادیر فرد:

Positional Embedding^{۱۳}
Sinusoidal Positional Embedding^{۱۴}
pos^{۱۵}

$$PE(pos, 2i + 1) = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \quad (2.3.2)$$

- pos : موقعیت کلمه در توالی است (مثلاً از ۰ تا $N - 1$ برای یک توالی N -تایی).
- i : شاخص بعد در بردار موقعیتی (از ۰ تا $d - 1$ برای بعد فضای برداری d).
- d : ابعاد فضای برداری مدل که نشان می‌دهد هر کلمه در چند بعد نمایش داده می‌شود.
- 10000: یک مقدار ثابت برای تنظیم مقیاس توابع تناوبی و ایجاد فرکانس‌های مختلف در ابعاد گوناگون.

همان‌طور که در شکل [شکل ۲.۳.۳](#) مشاهده می‌کنید، بعد از جاسازی کلمات، به آن جا سازی موقعیتی اضافه می‌شود. در این روش از توابع سینوس و کسینوس استفاده می‌شود. این توابع موقعیت‌ها را در فضای برداری به گونه‌ای نگاشت می‌کنند که مدل بتواند از ترتیب کلمات در توالی آگاه باشد [۵۴]. این ویژگی به مدل کمک می‌کند تا توالی زمانی را درک کرده و الگوهای زمانی را شبیه‌سازی کند. از مزایای این روش می‌توان به عدم نیاز به آموزش و توزیع متوازن جایگاه کلمات اشاره کرد.

بود	گرم	و	آفتابی	هوا	امروز
952.207	171.411	621.659	776.562	6422.693	171.411
5450.840	3274.350	1304.051	5567.288	6315.000	3274.350
1053.448	9192.819	0.565	58.942	9358.778	9192.819
1.458	3613.421	7679.805	2716.194	2141.081	3613.421
26713.329	6390.473	4556.025	5119.949	725.147	6390.473
+	+	+	+	+	+
...	1644.048	1201.458
...	8080.133	7902.890
...	2620.399	912.970
...	3821.102
...	9386.405	1601.217
-	3120.159	7018.820

شکل ۲.۳.۳: اضافه کردن جا سازی موقعیتی به جاسازی کلمه ای

۳.۳.۲ توجه

در روش شبکه های بازگشتی، توالی ورودی (مثلاً یک جمله) معمولاً به صورت گام به گام پردازش می شد [۲۱، ۱۳]. اما در ترانسفورمر می خواهیم مدلی داشته باشیم که به هر موقعیت (مثلاً یک کلمه) در توالی نگاه کند و به همه موقعیت های دیگر نیز به صورت موازی دسترسی داشته باشد. به این مفهوم توجه^{۱۶} می گوییم.

به زبان ساده، وقتی توکن (کلمه) i به توکن های دیگر نگاه می کند، می خواهد بداند کدام توکن ها برای تفسیر معنای خودش مهم ترند.

به طور مثال در جمله‌ی «یک گربه روی زمین نشسته است» می خواهد بداند کلمه‌ی «گربه» به واژه‌ی «نشستن» بیشتر توجه کند یا به «زمین». در اینجا فعل «نشستن» ارتباط نزدیک‌تری به «گربه» دارد و از نظر معنایی مرتبط‌تر است.

سه تا از اجزای اصلی یک توجه شامل موارد زیر است.

$$Q = \text{Query}, \quad K = \text{Key}, \quad V = \text{Value} \quad (\text{مقدار / ارزش}) \quad (\text{پرسش})$$

در ضرب شباهت های توجه^{۱۷} [۵۴]، ابتدا شباهت یا ارتباط بین پرسش^{۱۸} و کلید^{۱۹} را با محاسبه ضرب داخلی^{۲۰} به دست می آوریم، سپس آن را نرمال می کنیم (با تقسیم بر d_k) و از تابع سافت مکس^{۲۱} استفاده می کنیم تا ضرایب توجه^{۲۲} را به دست آوریم. در نهایت با همین ضرایب، ترکیبی خطی از بردارهای مقدار^{۲۳} را می گیریم.

فرمول به شکل زیر است:

attention ^{۱۶}
Scaled Dot-Product Attention ^{۱۷}
Query ^{۱۸}
Key ^{۱۹}
Dot Product ^{۲۰}
softmax ^{۲۱}
Attention Weights ^{۲۲}
value ^{۲۳}

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V \quad (2.3.3)$$

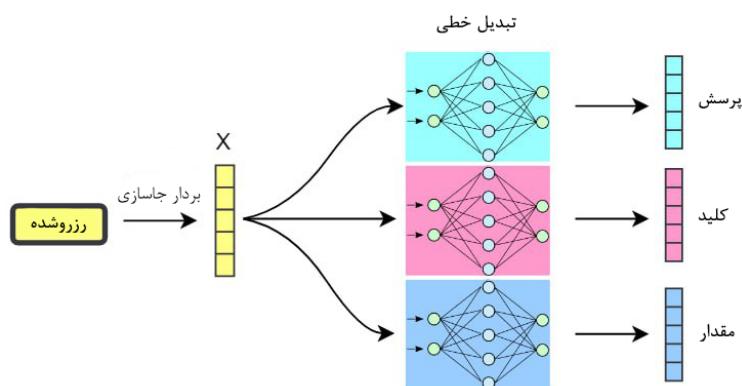
که در آن:

$Q \in \mathbb{R}^{n \times d_k}$ ماتریس پرسش برای

$K \in \mathbb{R}^{n \times d_k}$ ماتریس کلید برای

$V \in \mathbb{R}^{n \times d_v}$ ماتریس مقدار

ماتریس‌های وزنی قابل آموزش هستند که طی فرآیند یادگیری بهروزرسانی می‌شوند.



شکل ۲.۳.۴: نحوه به دست آوردن پرسش، کلید و مقدار

در واقع پرسش، کلید و مقدار با استفاده از mlp تولید می‌شود.

تقسیم بر d_k باعث می‌شود مقدار ضرب داخلی در ابعاد بالا خیلی بزرگ نشود و شبکه‌ها گرادیان پایدار بمانند.

$$\alpha = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) \quad (2.3.4)$$

یک ماتریس با ابعاد $n \times n$ است که سطر i -ام آن ضرایب توجه برای توکن i را نشان می‌دهد. تفسیر ضرایب توجه: هر سطر از α نشان می‌دهد که توکن فعلی به چه توکن‌هایی در جمله، با چه شدتی توجه می‌کند.

	بود	گرم	و	آفتابی	هوا	امروز
امروز	0.268	0.119	0.134	0.148	0.179	0.152
هوا	0.124	0.278	0.201	0.128	0.154	0.115
آفتابی	0.147	0.132	0.262	0.097	0.218	0.145
و	0.210	0.128	0.206	0.212	0.119	0.125
گرم	0.146	0.158	0.152	0.143	0.227	0.174
بود	0.195	0.114	0.203	0.103	0.157	0.229

شکل ۲.۳.۵: توجه

۴.۳.۲ توجه چند سر

در ایده چند سری ^{۲۴} بهجای آنکه فقط یکبار Q, K, V بسازیم و عملیات توجه را انجام دهیم، چندین مجموعه متفاوت Q_i, K_i, V_i می‌سازیم (هر کدام یک «سر» ^{۲۵} نام دارد) و به صورت موازی محاسبات توجه را انجام می‌دهیم. سپس خروجی همه این سرها را کنار هم قرار داده ^{۲۶} و در نهایت با یک ماتریس وزن دیگر ضرب می‌کنیم تا به بعد اصلی بازگردیم.

فرمول مربوط به این ایده به شکل زیر است:

multi head attention^{۲۴}

head^{۲۵}

concatenate^{۲۶}

$$\text{head}_i = \text{Attention}(Q_i, K_i, V_i) \quad (2.3.5)$$

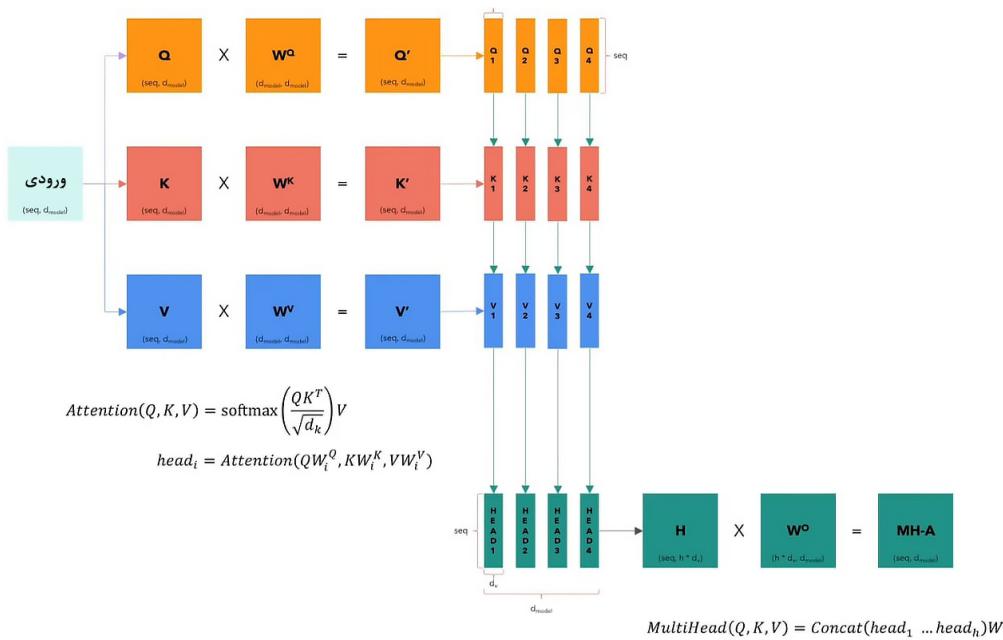
$$\text{MultiHead}(Q, K, V) = [\text{head}_1 \oplus \dots \oplus \text{head}_h]W_O \quad (2.3.6)$$

که در آن \oplus نشان‌دهنده عمل الحاق \bowtie است.

ماتریس وزن W_O به شکل زیر است:

$$W_O \in \mathbb{R}^{(h \cdot d_v) \times d_{\text{model}}}$$

که W_O ماتریسی است که خروجی الحاق شده را به بعد d_{model} برمی‌گرداند.



شکل ۲.۳.۶: توجه چند سر

concatenate^{VV}

چرا چندین سر؟

مشاهده چند منظر متفاوت: هر سر می‌تواند الگوهای گوناگونی از وابستگی‌ها را بیاموزد (مثلاً یک سر می‌تواند یاد بگیرد کلمه فعلی با کلمات همسایه نزدیک خود بیشتر مرتبط شود، یک سر دیگر روی ارتباط با کلماتی در فاصله دورتری متمرکز باشد، سر دیگر روی مطابقت جنس و تعداد در دستور زبان و ...).

افزایش ظرفیت مدل: با داشتن چند سر، مدل می‌تواند قدرت بیان بیشتری داشته باشد. ابعاد کمتر در هر سر: در عمل، اگر d_{model} مثلاً ۵۱۲ باشد، و تعداد سرها $h = 8$ ، آنگاه هر سر ابعادی در حدود $d_k = 64$ خواهد داشت؛ و این محاسبات ضرب داخلی را نیز مقیاس پذیر و قابل موازی‌سازی می‌کند.

۵.۳.۲ اتصال باقی مانده

در معماری‌های عمیق، هنگامی که تعداد لایه‌ها زیاد می‌شود، اغلب دچار ناپایداری گرادیان می‌شوند و این مشکل باعث دشواری در آموزش مدل می‌گردد [۲۱، ۳].

در مبدل‌ها [۵۴]، به جای این که خروجی توجه را به صورت مستقیم به لایه بعدی بدھیم، ورودی آن را نیز حفظ کرده و به خروجی اضافه می‌کنیم. ایده اصلی این روش از اتصالات باقی‌مانده^{۲۸} در شبکه‌های عمیق الهام گرفته شده است [۱۸].

اگر x ورودی به زیرماژول و $\text{SubLayer}(x)$ خروجی آن زیرماژول باشد، در انتهای کار عبارت

زیر را محاسبه می‌کنیم:

$$x + \text{SubLayer}(x) \quad (2.3.7)$$

این جمع به صورت عنصر به عنصر^{۲۹} انجام می‌شود.

Residual Connection^{۲۸}
Element-wise Addition^{۲۹}

اتصال باقیمانده در مبدل‌ها چندین مزیت دارد که عبارت اند از:

کمک به جریان یافتن گرادیان

وقتی ورودی مستقیماً به خروجی اضافه می‌شود، مسیری مستقیم برای عبور شیب (گرادیان) به عقب ایجاد می‌گردد. در صورت نبود این اتصال، اگر شبکه عمیق شود، گرادیان‌ها ممکن است در لایه‌های پایین محو شوند و عملاً ناپذید شدن گرادیان^{۳۰} رخ دهد [۲۱، ۲۲].

حفظ اطلاعات اصلی (هویت ورودی)

حتی اگر زیرماژول تغییری در اطلاعات ورودی ایجاد کند، با وجود اتصال باقیمانده^{۳۱}، ورودی اصلی همواره در خروجی نهایی حضور دارد. این ویرگی باعث می‌شود در صورت ناکافی بودن یادگیری زیرماژول یا در مراحل اولیه آموزش، دست‌کم بخشی از سیگنال (اطلاعات) خام به لایه‌های بالاتر برسد [۵۴، ۱۸].

کاهش ریسک تخریب ویرگی‌ها

در شبکه‌های عمیق، یکی از مشکلات این است که هر لایه ممکن است بخشی از اطلاعات مفید را تخریب کند. اتصال باقی مانده تضمین می‌کند که اگر لایه‌ای به هر دلیل نتوانست الگوی بهینه را یاد بگیرد، اطلاعات قبلی حداقل به صورت دست‌نخورده تا حدی منتقل می‌شود.

۴.۲ نرمال‌سازی لایه‌ها

در یادگیری عمیق، نرمال‌سازی^{۳۲} داده‌های یک لایه یا فعال‌سازی‌ها، اغلب به سرعت بخشیدن به همگرایی و پایدار کردن آموزش کمک شایانی می‌کند. شاید معروف‌ترین نوع نرمال‌سازی، نرمال

gradient vanishing^{۳۰}
Residual Connection^{۳۱}
Normalization^{۳۲}

سازی بچ^{۳۳} باشد که پیشتر در کارهای بینایی کانولوشنی^{۳۴} بسیار مورد استفاده قرار گرفت [۲۳]. نرمال سازی لایه ها^{۳۵} روشی جایگزین است که در ترانسفورمر استفاده می شود [۱، ۵۴]. علت اصلی این انتخاب، ماهیت توالی محور^{۳۶} بودن داده ها در پردازش زبان طبیعی و عدم تمایل به وابستگی به آمار مینی بچ^{۳۷} است.

نرمال سازی بچ

در نرمال سازی بچ ها، برای نرمال سازی، میانگین و واریانس روی تمام نمونه های موجود در مینی بچ (و نیز در طول ابعاد ویژگی) محاسبه می شود [۲۳]. این موضوع در پردازش زبان طبیعی کمی در دسرساز است؛ چون ترتیب توکن ها، طول جمله ها و حتی اندازه مینی بچ ممکن است نامنظم باشد. همچنین به خاطر تنوع طول توالی ها، پیاده سازی نرمال سازی بچ ها می تواند پیچیده شود.

نرمال سازی لایه ها

در نرمال سازی لایه ها، برای هر توکن به صورت جداگانه (در طول بُعد ویژگی)، میانگین^{۳۸} و واریانس^{۳۹} گرفته می شود [۱]. فرض کنید در یک لایه، بردار $h_i \in \mathbb{R}^{d_{\text{model}}}$ مربوط به توکن i باشد؛ یعنی ابعاد ویژگی آن d_{model} است. ما میانگین μ_i و واریانس σ_i^2 را از اجزای این بردار محاسبه می کنیم:

$$\mu_i = \frac{1}{d_{\text{model}}} \sum_{k=1}^{d_{\text{model}}} h_{i,k}, \quad \sigma_i^2 = \frac{1}{d_{\text{model}}} \sum_{k=1}^{d_{\text{model}}} (h_{i,k} - \mu_i)^2 \quad (2.4.8)$$

سپس نرمال سازی برای هر مؤلفه k در بردار توکن i به شکل زیر انجام می شود:

Batch Normalization^{۳۳}

CNN^{۳۴}

Layer Normalization^{۳۵}

sequence^{۳۶}

mini-batch^{۳۷}

mean^{۳۸}

variance^{۳۹}

$$\hat{h}_{i,k} = \frac{h_{i,k} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} \quad (2.4.9)$$

در نهایت، برای اینکه مدل بتواند مقیاس و بایاس جدیدی یاد بگیرد، شبیه بچ نرم ، دو پارامتر γ و β نیز در طول بعد ویژگی اعمال می‌شوند:

$$\text{LayerNorm}(h_i) = \gamma \odot \hat{h}_i + \beta \quad (2.4.10)$$

که در آن $\gamma, \beta \in \mathbb{R}^{d_{\text{model}}}$ هستند و \odot ضرب عنصر به عنصر است [۱].

مزایای نرمال سازی لایه در مبدل ها

- بی نیازی از وابستگی به ابعاد مینی بچ: با نرمال سازی لایه، می‌توان حتی با اندازه مینی بچ برابر ۱ نیز به خوبی آموزش دید، چراکه آمارها وابسته به ابعاد ویژگی اند و نه مینی بچ [۱].
- پایدارسازی توزیع فعال سازی ها: زمانی که مدل در حال یادگیری است، توزیع های داخلی لایه های میانی ممکن است تغییر کند. ^{۴۰} نرمال سازی لایه با نرمال سازی این توزیع، آموزش را پایدارتر و سریع تر می کند [۱، ۲۳].

- سازگاری با داده های توالی محور: هر توکن را جداگانه نرمال می کند و نگرانی ای بابت ترتیب طول جمله ها، یا قرار گرفتن چند جمله کوتاه / بلند در یک مینی بچ نداریم [۵۴].

در معماری مبدل ها، پس از خروجی هر زیر ماثول، مراحل به شکل زیر است:

۱.۴.۲ اتصال باقی مانده

ابتدا ورودی همان زیر ماثول (مثلاً بردار x) را با خروجی زیر ماثول (SubLayer(x)) جمع می کنیم. حاصل این جمع را می توان چنین نوشت:

Internal Covariate Shift^{۴۰}

$$z = x + \text{SubLayer}(x) \quad (2.4.11)$$

این z حالت ترکیبی از اطلاعات اصلی ورودی و اطلاعات یادگرفته شده توسط SubLayer است.

نرمال سازی لایه سپس این بردار z را وارد لایه LayerNorm می کنیم.

$$y = \text{LayerNorm}(z)$$

خروجی نهایی را می توان به لایه بعدی پاس داد یا به مرحله بعدی در همین لایه.

به عبارتی اگر بخواهیم در یک فرمول واحد بیان کنیم:

$$\text{Norm \& Add} = \text{LayerNorm}(x + \text{SubLayer}(x))$$

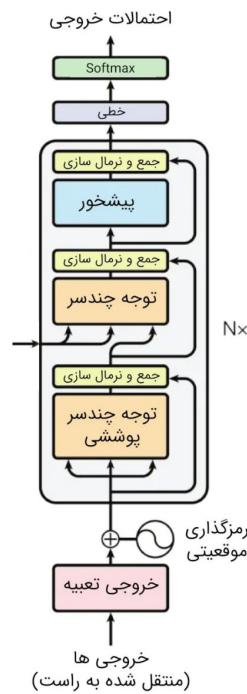
۵.۲ رمزگشا

رمزگشا در معماری ترانسفورمرها وظیفه تولید خروجی نهایی را بر عهده دارد. این خروجی معمولاً می تواند توالي هدف ^{۴۱} باشد، مانند ترجمه یک جمله یا پیش بینی توکن های بعدی در یک توالي [۵۴]. در این بخش، رمزگشا دو ورودی اصلی دارد: ۱. توالي هدف که معمولاً به صورت خودکار تولید می شود (مثلاً در ترجمه ماشینی یا تولید متن)، ۲. نمایش اطلاعات کدشده که توسط انکودر تولید شده است و شامل ویژگی های استخراج شده از توالي ورودی می باشد.

رمزگشا از این ورودی ها استفاده می کند تا به صورت گام به گام، خروجی نهایی خود را تولید کند. [۵۱، ۲]

همان طور که در شکل ۲.۵.۷ مشاهده می کنید، رمزگشا دو ورودی دارد.

Target Sequence^{۴۱}



شکل ۲.۰.۷: رمزگشا در مبدل
ها

تمامی بخش‌های رمزگشا مانند رمزگدار هستند اما در دیکودر توجه چند سر ماسک شده^{۴۲} وجود دارد [۵۴].

۶.۲ توجه چند سری ماسک شده

در مبدل‌ها، مکانیزم توجه چند سری^{۴۳} در بخش دیکودر به صورت ماسک شده^{۴۴} پیاده‌سازی می‌شود تا مدل نتواند توکن‌های آینده را ببیند و به صورت خودبازگشتی^{۴۵} توکن بعدی را پیش‌بینی کند [۵۴]. در واقع ایده اصلی استفاده از ماسک جلوگیری از مشاهده آینده است.

در معماری‌های خودبازگشتی، مدل در گام i از دیکودر تنها باید به توکن‌های قبلی $\{y_1, \dots, y_{i-1}\}$

Masked Multi-Head Attention^{۴۲}

Multi-Head Attention^{۴۳}

Masked^{۴۴}

Autoregressive^{۴۵}

دسترسی داشته باشد؛ اما نه به توکن‌های $\{y_{i+1}, y_{i+2}, \dots\}$. اگر مدل بتواند توکن‌های آینده را «نگاه» کند، پیش‌بینی توکن بعدی آسان و غیرواقعی می‌شود (مشکل نشت اطلاعات) [۵۱، ۲]. به همین دلیل در توجه چند سری ماسک شده در دیکودر، از یک ماتریس ماسک M استفاده می‌کنیم که اجازه نمی‌دهد هر توکن به توکن‌های آینده‌اش توجه کند.

۷.۲ مثال عددی توجه ماسک شده

فرض کنید دنبالهٔ ۴ توکنی داریم:

$$[y_1, y_2, y_3, y_4]$$

خروجی ضرب داخلی (قبل از softmax) یک ماتریس 4×4 خواهد بود:

$$S = \begin{bmatrix} s_{1,1} & s_{1,2} & s_{1,3} & s_{1,4} \\ s_{2,1} & s_{2,2} & s_{2,3} & s_{2,4} \\ s_{3,1} & s_{3,2} & s_{3,3} & s_{3,4} \\ s_{4,1} & s_{4,2} & s_{4,3} & s_{4,4} \end{bmatrix}$$

- سطر ۱ (توکن اول): تنها می‌تواند خودش (ستون ۱) را ببیند، اما ستون‌های ۲ تا ۴ ماسک می‌شوند.
- سطر ۲ (توکن دوم): می‌تواند به ستون‌های ۱ و ۲ نگاه کند، اما ستون‌های ۳ و ۴ ماسک می‌شوند.
- سطر ۳: می‌تواند ستون‌های ۱، ۲ و ۳ را ببیند، اما ستون ۴ ماسک می‌شود.
- سطر ۴: می‌تواند به ستون‌های ۱، ۲، ۳ و ۴ دسترسی داشته باشد (چهارمین توکن می‌تواند توکن‌های قبلی را ببیند. همچنین این توکن خودش نیز معمولاً^۲ در دسترس است. بسته به پیاده‌سازی، ممکن است توکن فعلی از خودش نیز استفاده کند یا نه. در معماری استاندارد، سطر^۳ معمولاً^۳ به ستون^۴ هم دسترسی دارد).

در عمل، ماتریس ماسک M به شکل زیر خواهد بود (با نشانه‌گذاری پایین مثلثی):

$$M = \begin{bmatrix} 0 & -\infty & -\infty & -\infty \\ 0 & 0 & -\infty & -\infty \\ 0 & 0 & 0 & -\infty \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

به این ترتیب، پس از جمع شدن با S و اجرای softmax در هر سطر، ضرایب توجه ستون‌های ماسک‌شده به صفر می‌کنند [۵۴].

۸.۲ مبدل‌های بینایی

ایده ترانسفورمرها در حوزه بینایی ^{۴۶} از تعمیم ترانسفورمر متن به تصاویر به وجود آمده است [۱۱].

ما در این بخش از مبدل‌های بینایی برای وظیفه کلاس‌بندی استفاده می‌کنیم.

در روش‌های متداول برای پردازش تصویر، از کانولشن ^{۴۷} های متوالی استفاده می‌کردند؛ اما در ترانسفورمرها تصاویر به پچ‌های مختلف شکسته می‌شوند [۱۱]. هر پچ شکسته شده از تصویر می‌تواند با سایر پچ‌ها به صورت موازی وارد مکانیزم توجه شود و شباهت یا ارتباطشان با یکدیگر سنجیده شود. در بخش‌های بعد، به طور مفصل روند انجام این کار را توضیح خواهیم داد.

۱.۸.۲ جاسازی پچ‌ها در مبدل‌های بینایی

در ترانسفورمرهای مبتنی بر متن، هر کلمه به توکن تبدیل می‌شود و سپس هر کلمه به برداری تبدیل می‌گردد. این بردارها پس از افزودن جاسازی موقعیتی وارد مکانیزم توجه می‌شوند [۵۴].

حال همین ایده در تصویر پیاده‌سازی شده است. همان‌طور که در شکل ۲.۸.۸ مشاهده می‌کنید، در مبدل‌های بینایی، به جای استفاده از عملیات کانولشن‌های متوالی که در شبکه‌های پیچشی ^{۴۸}

^{۴۶} vision transformer

^{۴۷} convolution

^{۴۸} convolutional neural network

مرسوم است [۳۰، ۳۲، ۱۸]، تصویر را به بلاک‌های غیرهمپوشان ($P \times P$) تقسیم می‌کنیم. این کار علاوه بر ساده‌سازی موازی‌سازی، به مدل اجازه می‌دهد از سازوکار توجه برای ارتباط بین این بلاک‌ها استفاده کند [۱۱].



شکل ۲.۸.۸: بخش‌بندی تصاویر

۲.۸.۲ شکل پچ‌ها:

فرض کنید ابعاد تصویر ورودی ($H \times W \times C$) باشد. به عنوان مثال، اگر اندازه تصویر $3 \times 224 \times 224$ باشد، طول و عرض تصویر به ترتیب 224 و تصویر دارای سه کanal رنگی است:

$$H = 224, \quad W = 224, \quad C = 3$$

حال اگر اندازه هر پچ ($P \times P$) باشد (برای نمونه 16×16)، تصویر به صورت یک جدول مشبک از پچ‌های کوچک تقسیم می‌شود. به هر پچ می‌توان مانند یک «کاشی» از تصویر نگاه کرد: - پچ اول: مختصات (در ارتفاع ۱۵ تا ۰) و (در عرض ۱۵ تا ۰)، - پچ دوم: مختصات (در ارتفاع ۱۵ تا ۰) و (در عرض ۳۱ تا ۱۶)، - و به همین ترتیب تا کل تصویر پوشش داده شود.

۳.۸.۲ تعداد پچ‌ها:

اگر پچ‌ها بدون همپوشانی باشند، ابعاد پچ باید بر ابعاد تصویر بخش‌پذیر باشد.

$$\text{تعداد پچ‌های افقی: } \frac{W}{P} \quad \text{تعداد پچ‌های عمودی: } \frac{H}{P}$$

در مجموع:

$$\left(\frac{H}{P}\right) \times \left(\frac{W}{P}\right) = \frac{H}{P} \times \frac{W}{P}. \quad (2.8.12)$$

برای مثال اگر:

$$H = 224, \quad W = 224, \quad P = 16 :$$

$$\frac{224}{16} = 14 \quad \Rightarrow \quad 14 \times 14 = 196 \quad (\text{تعداد پچها}).$$

در اکثر نسخه‌های مبدل‌های بینایی، پچ‌ها بدون همپوشانی^{۴۹} هستند. اندازه پچ‌های کوچک باعث می‌شود تعداد پچ‌ها زیاد شود و در نتیجه هزینه توجه بالا رود. از طرفی، پچ‌های بزرگ هزینه توجه را کاهش می‌دهند؛ اما ممکن است جزئیات محلی^{۵۰} را از دست بدهیم [۱۱].

۴.۸.۲ بردارکردن هر پچ

هر پچ دارای ابعاد $(P \times P \times C)$ است. برای مثال اگر $P = 16$ و $C = 3$ ، آنگاه پچ ابعاد $16 \times 16 \times 3$ خواهد داشت. برای این‌که بتوانیم پچ‌ها را مانند «токن»‌های پردازش زبان طبیعی به مبدل‌ها بدهیم، باید آن‌ها را به یک بردار یک بعدی تبدیل کنیم. در صورت قرار دادن پیکسل‌های پچ به صورت ردیفی^{۵۱}، طول این بردار خواهد بود:

$$P \times P \times C = P^2 \times C. \quad (2.8.13)$$

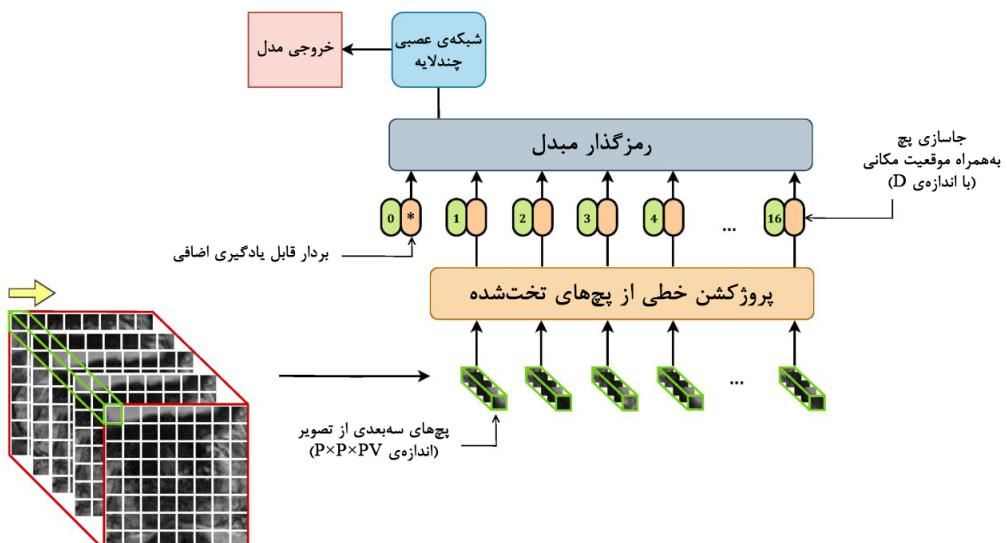
در مثال $(3 \times 16 \times 16)$ ، طول بردار می‌شود 768.

Non-overlapping^{۴۹}
Local Details^{۵۰}
Row-major^{۵۱}

۹.۲ اعمال لایه خطی

بعد از کنار هم چیدن پچ ها^{۵۲}، معمولاً یک لایه خطی^{۵۳} روی این بردار اعمال می شود تا آن را به بعد d_{model} (مثالاً ۷۶۸ یا ۱۰۲۴) ببرد. در حقیقت، این لایه یک تبدیل ویژگی^{۵۴} انجام می دهد تا همه پچ ها یک نمایندگی (تعییه شده) با ابعاد یکنواخت d_{model} پیدا کنند:

$$(P^2 \times C) \rightarrow d_{\text{model}}$$



شکل ۲.۹.۹: مبدل های بینایی

این مرحله شبیه ساخت توکن در پردازش زبان طبیعی است؛ با این تفاوت که در پردازش زبان طبیعی، توکن «کلمه» یا «زیرکلمه» است و از قبل دارای بردار تعییه شده جاساز شده بوده است [۵۴]. در مبدل های بینایی [۱۱]، ما ابتدا باید تصاویر را پچ کنیم و سپس بردارهای جاساز را از این پچ ها به دست آوریم.

ترانسفورمر نیاز دارد ورودی اش توالی توکن ها باشد. در پردازش زبان طبیعی توالی کلمات داریم،

Flatten^{۵۲}
Fully-Connected Layer^{۵۳}
Feature Transformation^{۵۴}

در مبدل های بینایی توالی «پچ»ها:

$$\{x_{\text{patch}_1}, x_{\text{patch}_2}, \dots, x_{\text{patch}_N}\}.$$

هر پچ اکنون یک بردار d_{model} -بعدی است. پس یک مجموعه با طول N (تعداد پچ‌ها) و عرض d_{model} خواهیم داشت. اگر عدد پچ‌ها N باشد (مثالاً ۱۹۶)، ترانسفورمر می‌تواند با مکانیزم توجه خود سر، وابستگی میان پچ‌ها را یاد بگیرد: کدام بخش از تصویر برای کدام بخش دیگر مهم‌تر است [۱۱، ۵۴].

معمولًاً پچ‌ها را به صورت ردیفی شماره‌گذاری می‌کنند (ابتدا پچ‌های ردیف بالایی از چپ به راست، سپس ردیف بعدی و ...)، تا مدل در صورت نیاز بتواند از موقعیت‌ها، اطلاعات مکانی تقریبی داشته باشد. در عمل، چون قصد داریم (در مراحل بعد) به هر پچ یک جاسازی موقعیتی هم اضافه کنیم، مکان دقیق هر پچ در بعد دوم (ویژگی) کد می‌شود.

در مبدل بینایی [۱۱] دیگر به کانولوشن وابسته نیستیم. در عوض، از جاسازی استفاده می‌شود. تقسیم کردن تصویر به بلاک‌های $(P \times P)$ ، کنار هم چیدن و تبدیل آن به جاساز همگی عملیات ریاضی ساده‌ای هستند که به راحتی روی TPU/GPU قابل موازی‌سازی‌اند.

۱.۹.۲ توکن کلاس بندی

توکن کلاس بندی ^{۵۵} یک بردار ویژه است که به ابتدای دنباله ورودی اضافه می‌شود و نقش آن، خلاصه کردن اطلاعات کل ورودی (چه متن، چه تصویر) است [۹، ۱۱].

در مبدل بینایی، این توکن در ابتدای پچ‌های تصویری قرار می‌گیرد. این توکن یک بردار با ابعاد d_{model} است (همان ابعاد سایر توکن‌ها) و پارامتری یادگرفتنی محسوب می‌شود؛ یعنی مدل طی آموزش، مقادیر آن را برای ذخیره و تجمع اطلاعات بهینه می‌کند.

در وظایف دسته‌بندی کلاس بندی، هدف این است که یک پیش‌بینی کلی برای کل ورودی (مثالاً یک جمله یا یک تصویر) ارائه دهیم؛ توکن کلاس بندی دقیقاً همین وظیفه را بر عهده دارد [۹]. این

توکن از طریق مکانیزم توجه چند سر در مبدل‌ها با تمامی توکن‌های دیگر (پچ‌های تصویر) ارتباط می‌گیرد و اطلاعات مهم آن‌ها را در لایه‌های مختلف مبدل‌ها را به صورت تجمعی یاد می‌گیرد. به عبارتی، توکن کلاس بندی نقش نماینده کل تصویر یا متن را بر عهده دارد.

توکن کلاس بندی از طریق ضرب داخلی در مکانیزم توجه، می‌تواند به تمام پچ‌ها نگاه کند و با ضرایب توجه (α) مشخص کند که از هر پچ چه مقدار اطلاعات بگیرد. بدین ترتیب، به‌طور ضمنی یاد می‌گیرد روی ویژگی‌هایی که برای دسته‌بندی مهم هستند (نظریه‌الگوها، اشکال و بخش‌های کلیدی تصویر) متمرکز شود.

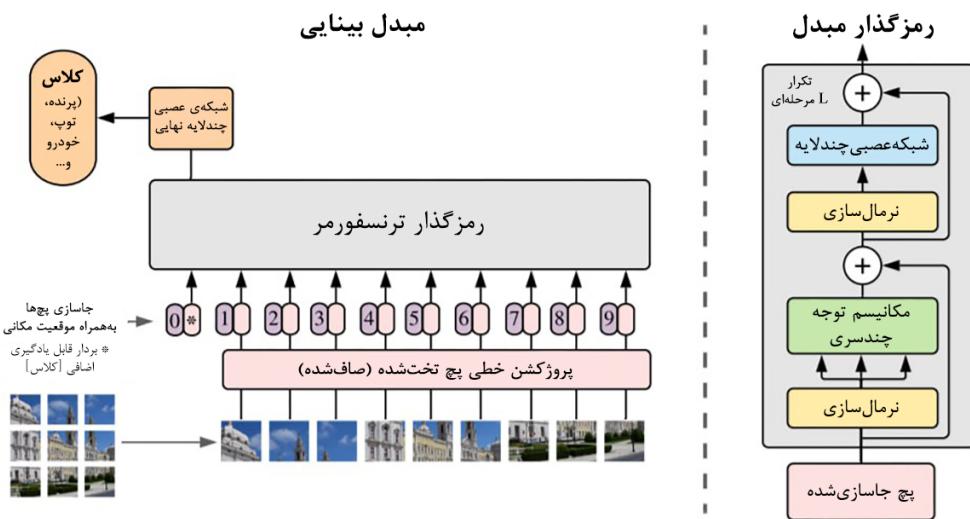
در طول لایه‌های ترانسفورمر، توکن کلاس بندی نقش محوری در خلاصه‌سازی بازنمایی کل تصویر ایفا می‌کند. این توکن به صورت پارامتر قابل یادگیری تعریف شده و در طول فرآیند آموزش به روزرسانی می‌شود [۱۱، ۹].

۲.۹.۲ رمزگذار در مبدل‌های بینایی

رمزگذار در ترانسفورمرها همانند مبدل اصلی است [۵۴]، با این تفاوت که در مبدل‌های بینایی [۱۱] دیگر به رمزگشنا نمی‌رویم. پس از عبور از بلاک‌های ترانسفورمر، در ساده‌ترین حالت یک لایه خطی^{۵۶} یا یک لایه MLP^{۵۷} بر روی بردار نهایی اعمال می‌شود و این لایه‌ها به تعداد کلاس‌ها خروجی می‌دهند. سپس خروجی هر لایه با گذر از تابع سافت‌مکس^{۵۸} به احتمال هر کلاس تبدیل می‌شود و در نهایت مدل کلاس با بیشترین احتمال را به عنوان خروجی پیش‌بینی می‌کند.

در مبدل‌ها، هر لایه رمزگشا و رمزگذار با پردازش عمیق‌تر روی توالی ورودی، می‌تواند نمایش بهتری از ویژگی‌ها به دست بیاورد [۵۴]. تکرار چندین باره رمزگشنا یا رمزگذار موجب می‌شود مدل بتواند ساختارهای پیچیده‌ای را یاد بگیرد و کیفیت و دقت آن در شناسایی توالی‌های طولانی و معانی پنهان افزایش یابد [۱۱، ۵۴]. در نتیجه، مدل با تعداد لایه‌های بیشتر اغلب عملکرد بهتری از خود

Fully Connected^{۵۶}
Multi-Layer Perceptron^{۵۷}
softmax^{۵۸}



شکل ۲.۹.۱۰: توکن توجه در مبدل های بینایی

نشان می دهد.

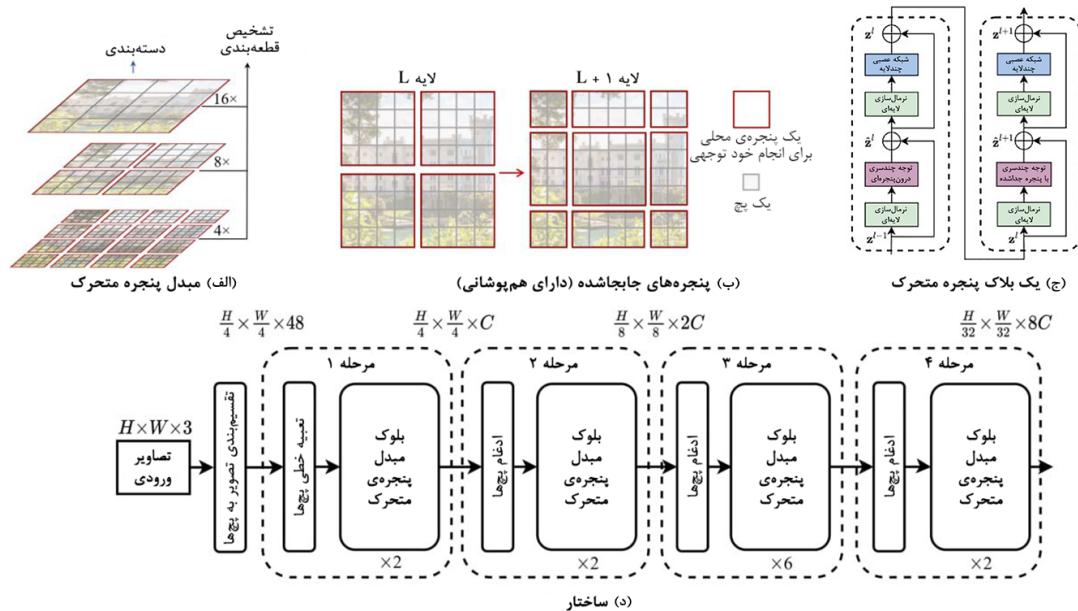
۱۰.۲ مبدل پنجره‌ای متحرک

ایده مبدل پنجره‌ای متحرک^{۵۹} از ترکیب چند مفهوم کلیدی در مدل‌های ترانسفورمر و شبکه‌های پیچشی شکل گرفت [۳۴، ۱۸، ۵۴].

یکی از بزرگترین مشکلات در ترانسفورمرهای اولیه، نیاز به محاسبات بسیار زیاد در زمانی بود که تصویر ورودی ابعاد بسیار بزرگی داشت [۱۱]. در ترانسفورمر معمولی هر پچ به تمامی پچ‌های دیگر توجه می‌کرد و در مواقعی که تعداد پچ‌ها زیاد می‌شد، هزینه محاسباتی و حافظه بهشت افزایش پیدا می‌کرد.

در شبکه‌های پیچشی، معماری معمولاً به صورت سلسله‌مراتبی پیش می‌رود [۱۸]؛ یعنی ابتدا ویژگی‌های محلی استخراج می‌شود، سپس با عمیق‌تر شدن لایه‌ها، این ویژگی‌ها در سطوح بالاتر با یکدیگر ترکیب می‌شوند. در مبدل پنجره‌ای متحرک [۳۴]، با دانش بر این موضوع توانسته‌اند هم

هزینه‌های محاسباتی را کاهش دهنده هم دقت مدل را افزایش دهنده.



شکل ۲.۱۰.۱۱: مدل پنجره متحرک

در مدل پنجره‌ای متحرک، به جای آنکه مدل به تمام پچ‌ها در یک سطح ویژگی نگاه کند، تصویر را به «پنجره‌های محلی»^{۶۰} تقسیم می‌کند و توجه را محدود به همان ناحیه می‌سازد [۳۴]. سپس با تکنیک جابه‌جایی^{۶۱} این پنجره‌ها در لایه‌های بعدی، توان مدل برای ترکیب اطلاعات از نواحی مختلف تصویر (و در نهایت دیدن کل تصویر) افزایش پیدا می‌کند. این رویکرد، ایده کلیدی‌ای بود که باعث شد مدل هم محاسبات سبک‌تری داشته باشد و هم بتواند ارتباط‌های جهانی^{۶۲} را در طول لایه‌ها به دست آورد.

یکی دیگر از ایده‌های مهم در در مدل پنجره‌ای متحرک، کوچک کردن تدریجی نقشه ویژگی در طول معماری است؛ مشابه کاری که در ResNet یا سایر CNN‌ها انجام می‌شود [۱۸]. این امر ضمن کاهش هزینه محاسباتی، باعث می‌شود مدل بتواند با سطوح مختلفی از ویژگی‌ها کار کند و

Local Windows^{۶۰}

Shift^{۶۱}

Global^{۶۲}

در نهایت خروجی نهایی با کیفیت‌تری ارائه دهد.

۱۰.۱۰.۲ قطعه‌بندی پچ در مبدل پنجره متحرک

فرض کنیم تصویر ورودی I دارای ابعاد $(H \times W \times 3)$ باشد. گام نخست، تقسیم تصویر به پچ‌های کوچک $(P \times P)$ است [۱۱]. اگر P اندازه پچ^{۶۳} باشد، آنگاه تعداد پچ‌ها در بعد افقی و عمودی، به ترتیب $\frac{W}{P}$ و $\frac{H}{P}$ خواهد بود. هر پچ را می‌توان به صورت یک بردار درآورد:

$$X_{\text{patch}} \in \mathbb{R}^{(P^2 \cdot 3)}.$$

سپس کل تصویر به $\frac{H}{P} \times \frac{W}{P}$ پچ تبدیل خواهد شد و در نتیجه، ماتریس X از کنار هم قرار گرفتن این پچ‌ها به صورت زیر به دست می‌آید:

$$X \in \mathbb{R}^{\left(\frac{H}{P} \cdot \frac{W}{P}\right) \times (P^2 \cdot 3)}.$$

برخلاف مبدل بینایی اصلی این کارد مبدل پنجره متحرک با استفاده از کانولوشن^{۶۴} انجام می‌شود. در واقع سایز کرنل در کانولوشن همان فضای برداری هر پچ هست که ما فرض می‌کنیم سایز کرنل برابر C است. پس در نتیجه

$$Z = X \cdot W_{\text{embed}} + b_{\text{embed}}, \quad Z \in \mathbb{R}^{\left(\frac{H}{P} \cdot \frac{W}{P}\right) \times C}. \quad (۲.۱۰.۱۴)$$

در عمل، این عملیات معادل یک تبدیل خطی ساده است:

$$W_{\text{embed}} \in \mathbb{R}^{(P^2 \cdot 3) \times C}, \quad b_{\text{embed}} \in \mathbb{R}^C.$$

patch size^{۶۳}
convolution^{۶۴}

پس از این مرحله، ما در هر موقعیت (w) (از شبکه پچها) یک بردار $z_{h,w} \in \mathbb{R}^C$ داریم. این ماتریس Z ورودی اولین مرحله (Stage) از مبدل های پنجره متحرک خواهد بود [۳۴]. هر بلوک مبدل پنجره متحرک از چند بخش اصلی تشکیل شده است [۳۴]:

- پنجره‌بندی تصویر^{۶۵} یا پنجره‌بندی جابه‌جاشده^{۶۶}

- اعمال توجه چمد سر پنجره‌ای^{۶۷}

- لایه Layer Norm^{۶۸} و Skip Connection

- مسیر پرسپیکترون چندلایه^{۷۰}:

۲.۱۰.۲ توجه چند سر پنجره‌ای

تعریف پنجره‌های محلی

در مبدل های پنجره متحرک، به جای آنکه تمام پیکسل های یک نقشه ویژگی بزرگ را یکجا در محاسبه توجه درگیر کنیم، نقشه ویژگی را به قطعه های کوچکی به اندازه $(M \times M)$ تقسیم می‌کنیم. این قطعه های کوچک را «پنجره های محلی» می‌نامیم.

اگر اندازه نقشه ویژگی در یک لایه $(H' \times W')$ باشد، با تقسیم آن به پنجره های $(M \times M)$ ، در راستای طول تقریباً $\frac{H'}{M}$ پنجره خواهیم داشت و در راستای عرض هم $\frac{W'}{M}$ پنجره. (برای راحتی، فرض می‌کنیم H' و W' دقیقاً مضربی از M باشند تا تقسیم بدون باقیمانده انجام شود.) هر کدام از این پنجره های $(M \times M)$ دارای M^2 پیکسل (یا موقعیت مکانی) است، و در هر پیکسل هم یک بردار ویژگی با بعد C قرار دارد.

Window Partition^{۶۵}

Shifted Window Partition^{۶۶}

Window Multi-Head Self Attention^{۶۷}

Skip Connection^{۶۸}

Layer Norm^{۶۹}

MLP^{۷۰}

به بیان ساده‌تر:

- نقشهٔ ویرگی مثل یک صفحهٔ بزرگ است.
- آن را مانند شترنج به مربع‌های کوچکی ($M \times M$) بخش می‌کنیم.
- در هر مربع (پنجه)، فقط به همان مربع نگاه می‌کنیم و محاسبات توجه را انجام می‌دهیم.
- این کار باعث می‌شود تعداد پیکسل‌هایی که درگیر محاسبهٔ توجه هستند، به مرتب کمتر شود و هزینهٔ محاسباتی کاهش یابد.

۳.۱۰.۲ توجه

برای هر بلوک، ابتدا بردارهای پرسش، کلید، مقدار ساخته می‌شوند. اگر $z_i \in \mathbb{R}^C$ بردار ورودی مربوط به موقعیت i باشد، آنگاه:

$$q_i = z_i W_Q, \quad k_i = z_i W_K, \quad v_i = z_i W_V, \quad (2.10.15)$$

که

$$W_Q, W_K, W_V \in \mathbb{R}^{C \times d}.$$

پارامتر d معمولاً به صورت $\frac{C}{h}$ در نظر گرفته می‌شود که در آن h تعداد سربندی سرها است. در توجه چند سر، خروجی نهایی با ترکیب h سر توجه محاسبه می‌شود.

در یک سر توجه، توجه به صورت زیر تعریف می‌شود:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V, \quad (2.10.16)$$

که در آن:

Q, K, V به ترتیب ماتریس‌هایی هستند که از کنار هم قرار دادن q_i, k_i, v_i (برای تمام پیکسل‌های آن پنجره) ساخته می‌شوند.

• \sqrt{d} : عامل مقیاس‌گذاری برای جلوگیری از بزرگ شدن بیش از حد ضرب داخلی است.

در مبدل‌های پنجره متحرک، این محاسبات به صورت پنجره‌ای انجام می‌شوند؛ یعنی برای هر پنجره، تنها پیکسل‌های داخل همان پنجره در ماتریس‌های Q, K و V لحاظ می‌شوند. به این ترتیب، زمان محاسبه و مصرف حافظه به شدت کاهش می‌یابد (در مقایسه با مبدل‌های بینایی که همه‌چیز را با هم مقایسه می‌کند).

تعداد سربندی h معمولاً طوری انتخاب می‌شود که $C = h \times d$. خروجی هر سرپس از محاسبه توجه به صورت زیر با هم ادغام می‌شوند:

$$\text{MultiHead}(Q, K, V) = [\text{head}_1, \text{head}_2, \dots, \text{head}_h] W_O, \quad (2.10.17)$$

که

$$\text{head}_j = \text{Attention}(Q_j, K_j, V_j), \quad W_O \in \mathbb{R}^{C \times C}$$

ماتریس ترکیب نهایی است.

۴.۱۰.۲ پنجره متحرک جا به جا شده

در مبدل‌های پنجره متحرک، ایده «پنجره‌های جابه‌جاشده»^{۷۱} به این منظور ارائه شده است تا مدل ارتباط پیکسل‌های واقع در پنجره‌های مجاور را هم یاد بگیرد [۳۴]. اگر فقط از پنجره‌های ثابت (بدون جابه‌جایی) استفاده کنیم، هر بلوک از تصویر تنها با پیکسل‌های همان پنجره در ارتباط خواهد بود و ممکن است اطلاعات نواحی مرزی با نواحی مجاور به خوبی تبادل نشود.

روش مبدل های پنجره متحرک برای رفع این محدودیت از یک تکنیک ساده اما مؤثر استفاده می کند [۳۴]:

- در یک لایه، محاسبات توجه در پنجره های محلی ثابت انجام می شود.
- در لایه بعدی، پنجره ها به اندازه های مشخص جابه جا می شوند (به صورت شیفت افقی و عمودی) تا نواحی مرزی نیز در محاسبات گنجانده شوند.
- این فرآیند باعث می شود که پیکسل ها در پنجره های مختلف (و در مرزهای مختلف) در محاسبات دخیل شوند و تبادل اطلاعات بهتری میان نواحی تصویر رخ دهد.

توجه چند سری پنجره ای

در توجه چند سری پنجره ای^{۱۷}، نقشه ویژگی به پنجره های ($M \times M$) تقسیم می شود [۳۴]. هیچ جابه جایی در این تقسیم بندی وجود ندارد؛ یعنی اگر نقشه ویژگی را یک مستطیل بزرگ در نظر بگیریم، آن را شبیه کاشی کاری یا شطرنج بندی به بلوک های مربعی ($M \times M$) برش می زنیم. در این حالت، پیکسل های هر پنجره فقط با هم دیگر (دروں همان پنجره) ارتباط برقرار می کنند.

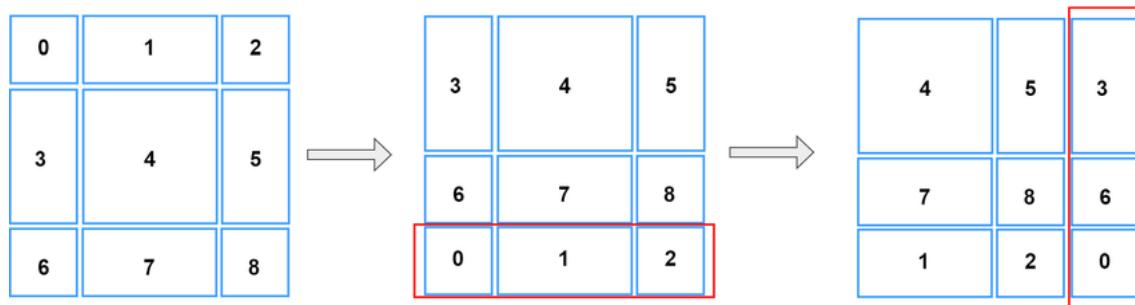
توجه چند سری پنجره ای جا به جا شده

مطابق شکل ۲.۱۰.۱۲، بعد از اینکه بلوک اول (توجه چند سری پنجره ای) کارش تمام شد، در بلوک دوم، قبل از تقسیم بندی به پنجره های ($M \times M$)، نقشه ویژگی را جابه جا می کنیم [۳۴]. در مقاله اصلی، این مقدار جابه جایی معمولاً نیم اندازه پنجره $\frac{M}{2}$ در راستای افقی و عمودی است. به این ترتیب:

- پیکسل هایی که پیش از این در دو پنجره جداگانه قرار داشتند، ممکن است حالا به دلیل جابه جایی وارد یک پنجره مشترک شوند.

- مدل حالا می‌تواند بین این پیکسل‌های «مرزی» نیز توجه برقرار کند و اطلاعات را بهتر مبادله کند.

با این جابه‌جایی، بخشی از پیکسل‌ها در نقشه ویژگی از یک طرف «خارج» می‌شوند. برای اینکه این پیکسل‌ها را از دست ندهیم، از ترفندی به نام جابجایی چرخه‌ای^{۷۳} استفاده می‌شود. در جا به جایی چرخه‌ای، پیکسل‌هایی که از سمت راست بیرون می‌روند دوباره از سمت چپ وارد می‌شوند و بالعکس؛ درست شبیه وقتی که یک تصویر را به صورت حلقه‌ای اسکرول می‌کنیم^{۷۴}. مثالی از جا به جایی چرخه‌ای در شکل ۲.۱۰.۱۲ آمده است.



شکل ۲.۱۰.۱۲: جا به جایی چرخه‌ای

در بلوک اول (بدون جابه‌جایی)، پنجره‌ها ثابت‌اند و پیکسل‌های مرزی در هر پنجره ممکن است فرصت کافی برای تبادل اطلاعات با پیکسل‌های مرزی پنجره‌کناری را نداشته باشند. در بلوک دوم (جابه‌جاشده)، مرزهای پنجره‌ها تغییر می‌کند و برخی پیکسل‌هایی که قبلاً در پنجره‌های جدا بودند، اکنون در یک پنجره مشترک‌اند؛ در نتیجه مدل می‌تواند رابطه و همبستگی بین آنها را هم یاد بگیرد.

این جابه‌جایی و قرارگیری مجدد پیکسل‌ها کنار هم در نهایت کمک می‌کند تا مدل بتواند اطلاعات کل تصویر را با هزینه محاسباتی کمتر (نسبت به توجه سراسری کامل) در اختیار داشته باشد [۳۴].

Cyclic Shift^{۷۳}

Wrap around^{۷۴}

اگر بخواهیم با مثال توضیح دهیم، فرض کنید در یک تابلوی شطرنجی، خانه‌های کناری هم‌دیگر را «نمی‌بینند» چون در دو بلوک مختلف هستند. اما اگر کمی تابلوی شطرنجی را به سمت بالا-چپ یا پایین-راست جابه‌جا کنیم، حالا بخشی از آن خانه‌ها وارد یک بلوک واحد می‌شوند و اطلاعاتشان با هم ترکیب می‌شود. سپس به‌طور دوره‌ای (*Cyclic*)، گوشه‌های اضافی را به آن سمت دیگر تابلوی شطرنجی می‌آوریم تا هیچ چیز از دست نرود.

به این شکل، سری اول و دوم بلوک‌های مبدل‌های پنجره متحرک تکمیل‌کنندهٔ یکدیگر می‌شوند

: [۳۴]

- بلوک اول: محاسبهٔ توجه در چهارچوب پنجره‌های ثابت.
- بلوک دوم: محاسبهٔ توجه در پنجره‌های جابه‌جا شده که منجر به تعامل بیشتر بین مرزهای مختلف می‌شود.

۵.۱۰.۲ پرسپتروون چند لایه

پس از انجام توجه چند سری پنجره‌ای جا به جا شده خروجی به یک مسیر MLP می‌رود [۳۴]. ساختار این MLP به صورت زیر است:

$$X' = \text{GELU}(XW_1 + b_1) W_2 + b_2, \quad (2.10.18)$$

که در آن

$$W_1 \in \mathbb{R}^{C \times (rC)}, \quad W_2 \in \mathbb{R}^{(rC) \times C}$$

هستند و r معمولاً ضریب افزایش بعد را نشان می‌دهد (مثلاً ۴).

تابع فعال‌ساز GELU (یا ReLU و سایر توابع) نیز در این جا قابل استفاده است [۱۹].

۶.۱۰.۲ ترکیب پچ‌ها

در مدل مبدل‌های پنجره متحرک، ساختار سلسله‌مراتبی به این معناست که ما در چند مرحله (Stage) مختلف، نقشه‌ویژگی را کوچک‌تر می‌کنیم و در عین حال، عمق (تعداد کانال‌های ویژگی) را افزایش می‌دهیم. هدف اصلی از این کار عبارت است از:

- استخراج ویژگی‌های سطح بالاتر: وقتی نقشه‌ویژگی کوچک‌تر می‌شود، هر واحد از نقشه‌ویژگی بیانگر بخش گسترده‌تری از تصویر اصلی است؛ پس مدل به تدریج جزئیات محلی را با درک کلی‌تری از تصویر جایگزین می‌کند [۱۸].
- کاهش هزینه محاسبات: در مراحل بعدی، چون ابعاد فضایی کمتر می‌شود، مدل راحت‌تر می‌تواند با ویژگی‌های جدید کار کند (چون مثلاً به جای $(H \times W)$ پیکسل، تعداد کمتری پیکسل داریم) [۳۴].

پس از چندین بلوک پردازشی، نقشه‌ویژگی، ابعادی به شکل $(\frac{H}{P}, \frac{W}{P})$ با تعداد کanal C دارد. این یعنی پس از برش‌دادن تصویر به پچ‌ها و گذر از چند لایه، اکنون یک نقشه‌ویژگی داریم که کوچک‌تر از تصویر اصلی است، اما هنوز ممکن است خیلی بزرگ باشد.

در مرحله بعد (Stage بعدی)، می‌خواهیم این نقشه را نصف کنیم (یعنی طول و عرض را دو برابر کوچک کنیم) و در عوض عمق کanal را دو برابر کنیم (تا ظرفیت مدل در استخراج ویژگی‌های پیچیده‌تر بیشتر شود). برای انجام این کار از فرایندی به نام ترکیب پچ‌ها استفاده می‌کنیم. [۳۴]:

۱. انتخاب بلوک‌های (2×2)

ابتدا نقشه‌ویژگی را در بعد مکانی به بلوک‌های (2×2) تقسیم می‌کنیم. اگر $Z_{i,j}$ ویژگی مکان (i, j) باشد، یک بلوک (2×2) شامل چهار پیکسل است:

$$Z_{2i,2j}, \quad Z_{2i,2j+1}, \quad Z_{2i+1,2j}, \quad Z_{2i+1,2j+1}.$$

۲. ادغام ویژگی‌های چهار پیکسل

برای هر بلوک (2×2) ، این چهار پیکسل را در بعد کanal به هم می‌چسبانیم. اگر هر پیکسل یک بردار از بعد C باشد، اکنون بعد حاصل از کنار هم گذاشتن این چهار پیکسل می‌شود $4C$. نام این بردار ادغام شده را Z' می‌گذاریم.

۳. لایه خطی برای تغییر بعد

وقتی چهار بردار C -بعدی را کنار هم می‌گذاریم، یک بردار $4C$ -بعدی شکل می‌گیرد. حال با یک لایه خطی، بعد $4C$ را به بعد جدیدی تبدیل می‌کنیم. معمولاً این بعد جدید برابر $2C$ در نظر گرفته می‌شود؛ یعنی دو برابر بزرگ‌تر از قبل اما نه چهار برابر:

$$Z' \mapsto Z'' = Z' W_{\text{merge}} + b_{\text{merge}}, \quad (2.10.19)$$

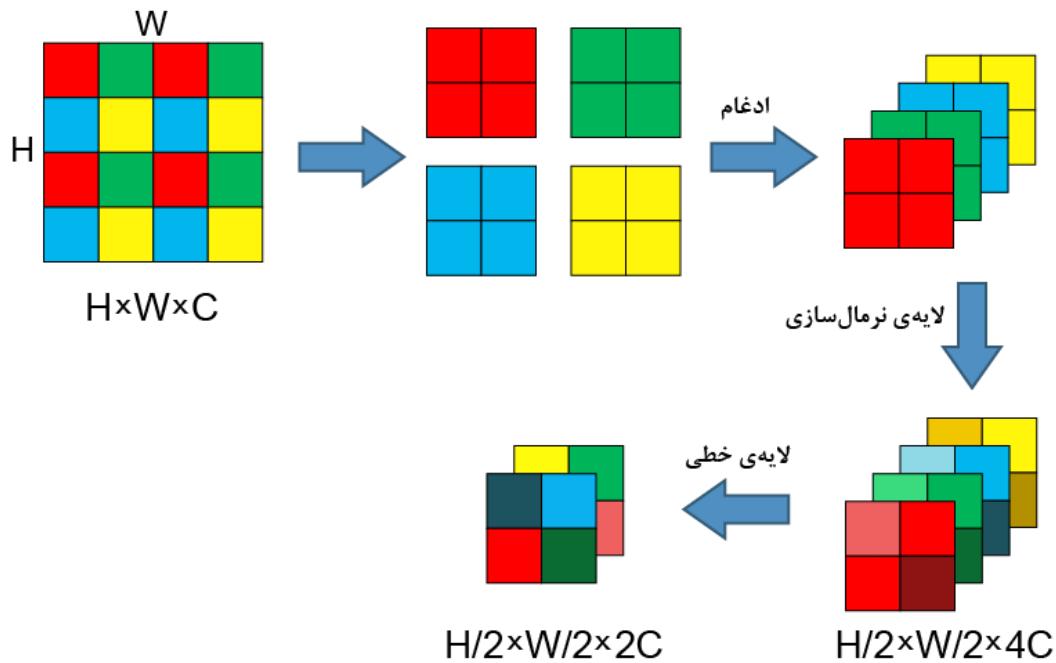
که بعد ویژگی را از $4C$ به $2C$ کاهش می‌دهد.

۴. کاهش ابعاد مکانی

در عین حال، وقتی هر چهار پیکسل (2×2) را ادغام می‌کنیم، نقشهٔ ویژگی ما ابعاد فضایی $(\frac{H}{2P} \times \frac{W}{2P})$ خواهد داشت (چون هر بلوک (2×2) تبدیل به یک بردار می‌شود). به عبارت دیگر، تعداد نقاط مکانی نصف می‌شود (هم در طول و هم در عرض)، اما کanal از C به $2C$ افزایش می‌یابد.

در شبکه‌های کانولوشنی، مرتباً از لایه‌های ادغام^{۷۵} یا کانولوشن با گام^{۷۶} برای کوچک‌کردن ابعاد استفاده می‌شود تا اطلاعات سطح بالاتر (مثل ساختار کلی اشیا) راحت‌تر استخراج شود [۱۸]. در مبدل پنجگره متحرک هم همین ایده سلسله‌مراتب را به دنیای مبدل‌ها آورده است [۳۴]. همچنین اگر ابعاد فضایی را کم نکنیم، هزینهٔ توجه به شدت زیاد می‌شود (چون باید در هر لایه برای همه

Pooling^{۷۵}
Stride-Convolution^{۷۶}



شکل ۲.۱۰.۱۳: ادغام پچ ها

پیکسل‌ها توجه محاسبه گردد).

در معماری کلی کبدل‌های پنجره متحرک، پس از Stage 1 و عبور از بلوک‌های توجه چند سر پنجره‌ای و توجه چند سر پنجره‌ای جایه جا شده، عملیات ادغام پچ‌ها انجام می‌شود. سپس در Stage 2، ویژگی‌های کوچکتری داریم، اما تعداد کانال‌ها افزایش یافته است [۳۴]. مشابه معماری‌های کانولوشنی، با افزایش عمق^{vii}، ابعاد فضایی کاهش و تعداد کانال‌ها افزایش پیدا می‌کند.

در انتهای Stage آخر، خروجی به یک لایه FC داده می‌شود تا تعداد کلاس‌ها را پیش‌بینی کند. پس از گذراز Softmax، احتمال هر کلاس به دست می‌آید و مدل در نهایت کلاس نهایی را بر می‌گزیند.

فصل ۳

روش‌های پیشنهادی

استفاده از روش‌های تانسوری در شبکه‌های عصبی چندلایه

در سال‌های اخیر، استفاده از روش‌های تانسوری به عنوان روشی نوین در بهینه‌سازی معماری‌های شبکه‌های عصبی، به ویژه در مدل‌هایی که تعداد پارامترهای آن‌ها بسیار زیاد است، مانند شبکه‌های عصبی چندلایه^۱، توجه بسیاری را به خود جلب کرده است [۴۶]. تانسورها تعمیمی از ماتریس‌ها به ابعاد بالاتر هستند و به طور طبیعی برای نمایش داده‌های چندبعدی همچون تصاویر، ویدیوها یا سری‌های زمانی چندکاناله مناسب‌اند. بهره‌گیری از ساختار تانسوری در معماری شبکه، این امکان را فراهم می‌سازد که بدون نیاز به فشرده‌سازی اولیه (مانند مسطح سازی^۲ کردن ورودی)، اطلاعات ساختاری میان ابعاد مختلف حفظ شده و مدل بتواند از روابط درون‌تعاملی موجود میان این ابعاد بهره‌برداری نماید.

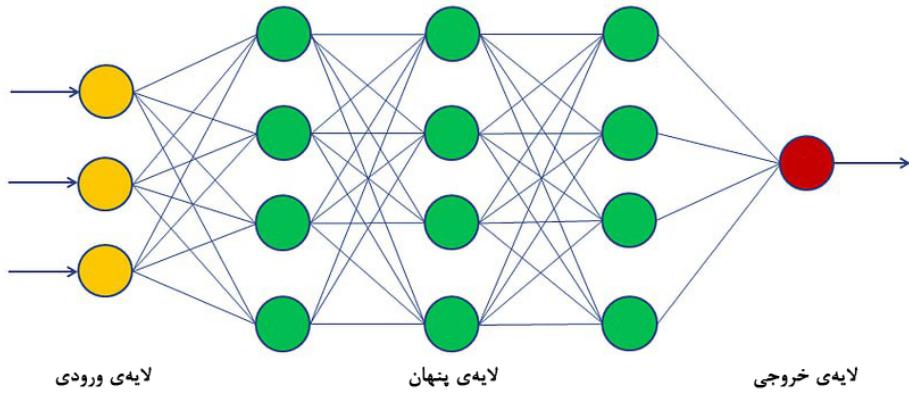
در معماری سنتی شبکه عصبی چند لایه، نگاشت از ورودی $x \in \mathbb{R}^n$ به خروجی $y \in \mathbb{R}^m$

به وسیله ضرب ماتریسی انجام می‌گیرد:

Mlp^۱
flatten^۲

$$\mathbf{y} = W\mathbf{x} + \mathbf{b}$$

که در آن $W \in \mathbb{R}^{m \times n}$ ماتریس وزن و \mathbf{b} بردار بایاس است.



شکل ۳.۰.۱: شبکه عصبی پرسپترون چند لایه

در مقابل، در روش‌های تansوری، وزن‌ها به صورت یک تانسور مرتبه بالاتر مدل‌سازی می‌شوند و نگاشت ورودی به خروجی با استفاده از ضرب‌های چندحالتی (ضرب تانسوری در ابعاد مختلف) صورت می‌پذیرد:

$$\mathbf{y} = \mathcal{W} \times_1 \mathbf{x}_1 \times_2 \mathbf{x}_2 \times_3 \cdots + \mathbf{b} \quad (3.0.1)$$

در این رابطه، \mathcal{W} یک تانسور وزن است و $\text{عملگر}_{n \times \dots \times n}$ نشان‌دهنده ضرب تانسوری در بعد n ام می‌باشد.

روش‌هایی نظیر TCL^۳ و TRL^۴ به عنوان نمونه‌هایی از این رویکرد، با بهره‌گیری از تکنیک‌های تجزیه تانسوری همچون تاکر^۵ یا تجزیه CP، نه تنها باعث کاهش چشمگیر در تعداد پارامترها می‌شوند، بلکه ساختار چندبعدی داده‌ها را نیز حفظ می‌نمایند [۲۸، ۲۹].

^۳layer contraction tensor

^۴layer regression Tensor

^۵Tucker^۶

مزایای استفاده از روش‌های تانسوری

استفاده از روش‌های تانسوری در شبکه‌های عصبی چندلایه مزایای متعددی به همراه دارد که برخی از مهم‌ترین آن‌ها عبارت‌اند از:

- کاهش چشمگیر تعداد پارامترها: با بهره‌گیری از فشرده‌سازی تانسوری، می‌توان ابعاد تانسور وزن‌ها را به‌گونه‌ای کاهش داد که بدون افت محسوس در عملکرد مدل، مصرف حافظه و پیچیدگی محاسباتی به‌طور قابل توجهی کاهش یابد [۲۹، ۳۱].
- حفظ ساختار داده‌های ورودی: برخلاف روش‌های سنتی که در آن‌ها داده‌ها پیش از ورود به لایه‌های چگال^۶ باید مسطح‌سازی^۷ شوند، استفاده از ساختار تانسوری این امکان را فراهم می‌کند که ساختار فضایی، زمانی یا کانالی داده‌ها حفظ شده و ارتباط میان ابعاد مختلف ورودی بهتر درک و پردازش شود [۲۹].

محدودیت‌ها و چالش‌ها

با وجود مزایای متعدد، بهره‌گیری از روش‌های تانسوری در معماری‌های شبکه‌های عصبی با چالش‌ها و محدودیت‌هایی نیز همراه است که در ادامه به برخی از مهم‌ترین آن‌ها اشاره می‌شود:

- پیچیدگی بالاتر در پیاده‌سازی: پیاده‌سازی لایه‌های مبتنی بر عملیات تانسوری معمولاً به ابزارها و کتابخانه‌های خاصی همچون Tensor Toolbox یا Tensorly نیاز دارد. این موضوع فرآیند طراحی و توسعه مدل را پیچیده‌تر از لایه‌های استاندارد مانند شبکه عصبی چند لایه می‌سازد [۲۸].

- بهینه‌سازی دشوارتر: فرآیند آموزش مدل‌های تانسوری می‌تواند نسبت به مدل‌های معمولی کنتر باشد. الگوریتم‌های مبتنی بر گرادیان ممکن است در فضای پارامتری تانسورها با

سطوح خطای غیرهموار یا چندوجهی مواجه شوند که روند همگرایی را دشوار می‌کند [۵۵].

- احتمال کاهش دقت در فشردهسازی شدید: در صورتی که میزان فشردهسازی تانسورها بیش از حد بالا باشد، مدل ممکن است توانایی لازم برای نمایش روابط غیرخطی و الگوهای پیچیده را از دست داده و در نتیجه، دقت نهایی پیش‌بینی کاهش یابد [۲۹].

۱۰.۳ لایه فشردهسازی تانسوری

در بسیاری از مدل‌های یادگیری عمیق—به‌ویژه شبکه‌های عصبی پیچشی^۸ فعال‌سازی‌های لایه‌های میانی به‌شکل تانسورهایی با مرتبه بالا ظاهر می‌شوند. به‌طور سنتی، برای اعمال لایه‌های کاملاً متصل^۹، این تانسورها ابتدا با عملیات مسطح‌سازی به بردار تبدیل و سپس به فضای خروجی نگاشت می‌شوند؛ روشی که ساختار چندخطی^{۱۰} داده را از بین برد و به‌تبع، تعداد پارامترها افسارگ‌سیخته افزایش می‌یابد [۲۹].

در پاسخ به این چالش، لایه‌ای با عنوان *لایه فشردهسازی تانسوری*^{*} یا به اختصار **TCL** معرفی شده است. این لایه بدون نیاز به مسطح‌سازی، هر مد (mode) از تانسور ورودی را مستقل فشردهسازی می‌کند و ساختار چندخطی داده را حفظ می‌نماید [۲۸، ۲۹].

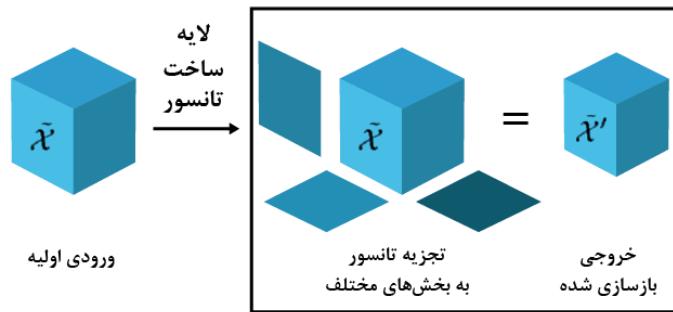
فرمول‌بندی ریاضی

فرض کنید تانسور فعال‌سازی ورودی:

$$\mathcal{X} \in \mathbb{R}^{S \times I_0 \times I_1 \times \dots \times I_N}$$

باشد (که S اندازه دسته‌بندی و I_k ‌ها ابعاد کanal/فضایی را نشان می‌دهند). هدف لایه **TCL**، نگاشت:

convolution^۸
Fully Connected^۹
multilinear structure^{۱۰}.



شکل ۲.۰.۲: لایه فشرده ساز تانسوری

$$\mathcal{X}' = \mathcal{X} \times_1 V^{(0)} \times_2 V^{(1)} \cdots \times_{N+1} V^{(N)} \quad (3.0.2)$$

است، که در آن:

$$V^{(k)} \in \mathbb{R}^{R_k \times I_k}, \quad \forall k = 0, \dots, N.$$

با این عملگرهای ضرب چندحالته (\times_n) تعداد پارامترها به شکلی چشمگیر کاهش می‌یابند [۲۸].

تحلیل تعداد پارامترها

در TCL، کل پارامترهای قابل آموزش برابر است با:

$$\text{TCL}_{\text{پارامترهای}} = \sum_{k=0}^N I_k \cdot R_k \quad (3.0.3)$$

در حالی که در لایه متصل سنتی (پس از مسطح‌سازی):

$$\text{FC}_{\text{پارامترهای}} = \left(\prod_{k=0}^N I_k \right) \cdot O \quad (3.0.4)$$

۳. روش‌های پیشنهادی

که O تعداد نورون‌های خروجی لایه است. نمایش ریاضی بالا بهوضوح نشان می‌دهد که TCL با جایگزینی ضرب با جمع ساده، بهصرفه‌سازی قابل توجهی در حافظه و محاسبات مدل منجر می‌شود [۲۹، ۲۸].

۲۰.۳ لایه رگرسیون تانسوری (TRL)

این لایه، خروجی مدل را مستقیماً از تانسور ورودی تولید می‌کند، بدون مسطح‌سازی؛ بهطوری‌که ساختار چندبعدی ورودی حفظ و نگاشت خروجی نیز بهصورت مرتبه پایین^{۱۱} مدل شده است [۲۹].

فرمول‌بندی ریاضی

فرض مجدد تانسور:

$$\mathcal{X} \in \mathbb{R}^{S \times I_0 \times I_1 \times \dots \times I_N}$$

و تانسور وزن:

$$\mathcal{W} \in \mathbb{R}^{I_0 \times I_1 \times \dots \times I_N \times O}.$$

نگاشت خروجی:

$$\mathbf{Y} = \langle \mathcal{X}, \mathcal{W} \rangle_N + \mathbf{b} \quad (3.0.5)$$

در اینجا عملگر $\langle \cdot, \cdot \rangle$ ضرب داخلی روی N بعد اول \mathcal{W} و N بعد اول \mathcal{X} را نشان می‌دهد [۲۹].

برای کاهش پارامترها، \mathcal{W} با کمک تجزیه توکر به شکل زیر نوشته می‌شود:

$$\mathcal{W} = \mathcal{G} \times_0 U^{(0)} \times_1 U^{(1)} \dots \times_N U^{(N)} \times_{N+1} U^{(N+1)}, \quad (3.0.6)$$

low-rank^{۱۱}

۳. روش‌های پیشنهادی

که \mathcal{G} هستهٔ کم مرتبه^{۱۲} و $U^{(k)}$ ماتریس‌های فشرده‌سازی هستند [۲۹].

فرمول نهایی:

$$\mathbf{Y} = \langle \mathcal{X} \times_0 (U^{(0)})^\top \cdots \times_N (U^{(N)})^\top, \mathcal{G} \times_{N+1} U^{(N+1)} \rangle_N + \mathbf{b} \quad (3.0.7)$$

که به صورت محاسباتی بهینه‌تر عمل می‌کند [۲۹].

تحلیل تعداد پارامترها

در لایه رگرسیون تانسوری با تجزیه تاکر^{۱۳}، تعداد پارامترها برابر است با:

$$\text{TRL} = \left(\prod_{k=0}^{N+1} R_k \right) + \left(\sum_{k=0}^N I_k R_k \right) + R_{N+1} \cdot O, \quad (3.0.8)$$

که معمولاً بسیار کمتر از تعداد پارامترهای لایه FC است، بهویژه هنگامی که $I_k \ll R_k$ انتخاب شوند [۲۹].

۳.۰.۳ چرا در مدل‌های بینایی از تانسور استفاده می‌کنیم؟

۱. کاهش تعداد پارامترها و حافظه مصرفی

یکی از چالش‌های اصلی در معماری‌های مبتنی بر مدل‌های بینایی، رشد نمایی تعداد پارامترها در لایه‌های کاملاً متصل، بهویژه در بخش‌های انتهایی شبکه است. با جایگزینی این لایه‌ها با ساختارهای تانسوری مانند TCL و TRL می‌توان از ساختار چندبعدی داده بهره برده و از طریق تجزیه‌های کم مرتبه، تعداد پارامترها را به صورت چشمگیری کاهش داد [۲۹، ۲۸، ۴۶]. این جایگزینی نه تنها موجب صرفه‌جویی در حافظه می‌شود، بلکه پیچیدگی محاسباتی مدل را نیز کاهش داده و امکان به کارگیری آن را در محیط‌های کم منبع (مانند دستگاه‌های لبه‌ای و موبایل) فراهم می‌سازد [۱۷].

core tensor^{۱۲}
Tucker^{۱۳}

۲. افزایش تفسیرپذیری با حفظ ساختار چندخطی داده

در حالی‌که عملیات مسطح‌سازی بر روی تانسورهای ورودی، ساختار ارتباطی میان ابعاد داده را از بین می‌برد، استفاده از لایه‌های تانسوری این ساختار چندخطی را حفظ می‌کند. این امر نه تنها مدل را برای کار با داده‌های فضایی یا زمانی چندکاناله مطلوب‌تر می‌سازد بلکه قابلیت تفسیرپذیری پیش‌بینی‌های مدل را نیز ارتقا می‌دهد. [۱۷]

۳. کاهش نیاز به داده‌های آموزشی بزرگ

مدل‌های بینایی مبتنی بر مبدل، به‌دلیل فقدان سوگیری‌های مکانی ذاتی شبکه‌های پیچشی، نیازمند حجم عظیمی از داده برای آموزش مؤثر هستند. با استفاده از لایه‌های تانسوری مانند TCL و TRL که نگاشتها را به صورت چندخطی و فشرده مدل‌سازی می‌کنند و ساختار درونی داده را حفظ می‌کنند، می‌توان از ظرفیت مدل به صورت بهینه‌تر بهره برد. این ساختار نه تنها کمک می‌کند وابستگی‌های میان‌بعدی بهتر درک شوند بلکه از بیش‌برازش در شرایط داده‌ی محدود جلوگیری می‌کند؛ به عنوان مثال، در روش FacT^{۱۴}، فقط ۰.۰۱٪ پارامترهای مبدل‌ها برای تعداد بسیار کمی از نمونه‌های آموزش داده می‌شود و با صرفه‌جویی قابل توجه در پارامترها، عملکردی مشابه یا بهتر از تنظیم کامل^{۱۵} ارائه می‌دهد [۲۵].

۴.۰.۳ روش تانسوری مبدل پنجره متحرك:

۵.۰.۳ پیاده‌سازی مرحله‌ی تعییه پچ با استفاده از فشرده‌سازی تانسوری

در معماری‌های کلاسیک مبتنی بر مبدل‌های بیانی^{۱۶} و همچنین در ساختار مبدل‌های پنجره‌ای^{۱۷}، مرحله‌ی اولیه‌ی پردازش تصویر شامل تقسیم تصویر به قطعات کوچک (پچ‌ها) و سپس نگاشت هر

Factor-Tuning^{۱۴}

full-tuning^{۱۵}

Vision transformer^{۱۶}

swin transformer^{۱۷}

۳. روش‌های پیشنهادی

پچ به یک بردار نهفته با ابعاد ثابت است. این نگاشت معمولاً^{۱۸} با استفاده از یک لایه‌ی خطی^{۱۸} یا پیچشی^{۱۹} با کرنل و گام برابر با اندازه‌ی پچ انجام می‌شود.

در روش تانسوری بهجای استفاده از نگاشت برداری ساده، از لایه‌ی فشرده‌سازی تانسوری^{۲۰} برای تبدیل هر پچ به یک تانسور چندبعدی در فضای ویژگی استفاده شده است. این نگاشت تانسوری نه تنها باعث حفظ ساختار چندخطی پچ‌ها می‌شود، بلکه با فشرده‌سازی موثر، منجر به کاهش پارامترها می‌شود.

در ابتدا فرض می‌کنیم ورودی مدل تصویری با اندازه‌ی (B, C, H, W) باشد که در آن:

- B اندازه‌ی دسته‌ی آموزشی^{۲۱} است.

- C تعداد کانال‌های تصویر (برای مثال ۳ در تصاویر رنگی).

- $H \times W$ ابعاد تصویر است.

این تصویر با استفاده از پچ‌هایی با اندازه‌ی $(P \times P)$ به $\frac{H}{P} \times \frac{W}{P}$ قطعه تقسیم می‌شود. سپس با استفاده از عملیات بازارایی^{۲۲}، ساختار ورودی به شکل زیر تبدیل می‌شود:

$$\mathcal{X} \in \mathbb{R}^{B \times P_1 \times P_2 \times C_1 \times C_2 \times C_3}$$

که در آن:

$P_2 = \frac{W}{P}$ و $P_1 = \frac{H}{P}$ • به ترتیب تعداد پچ‌ها در راستای ارتفاع و عرض تصویر هستند.

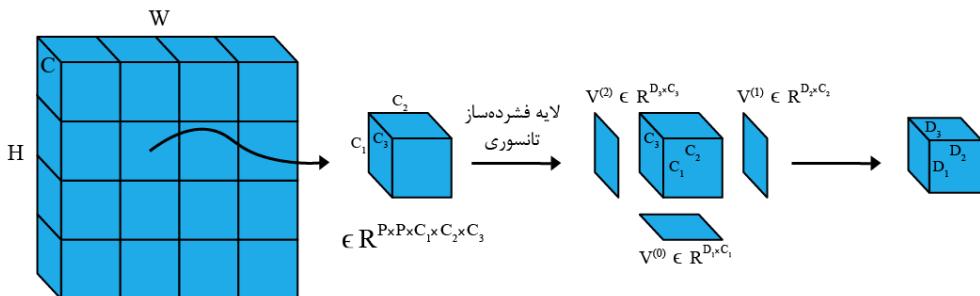
$C_2 = P$ و $C_1 = P$ • ابعاد مکانی هر پچ‌اند.

linear ^{۱۸}	
convolution ^{۱۹}	
tensor contraction layer ^{۲۰}	
Batch Size ^{۲۱}	
rearrangement ^{۲۲}	

۳. روش های پیشنهادی

$C_3 = C$ تعداد کانال های تصویر ورودی است.

در این بازنمایی، هر پچ در موقعیت (i, j) به صورت یک تانسور سه بعدی با ابعاد $C_1 \times C_2 \times C_3$ به صورت یک تانسور سه بعدی با ابعاد $C_1 \times C_2 \times C_3$ نمایش داده می شود. این ساختار چند بعدی امکان استفاده ای مستقیم از عملیات تانسوری بدون نیاز به تخت سازی را فراهم می سازد.



شکل ۳.۰.۳: تعبیه پچ ها

برای نگاشت هر پچ به فضای نهفته، به جای استفاده از mlp از یک لایه فشرده ساز تانسوری استفاده می شود. در واقع سه تا بعد اخرا با استفاده از لایه فشرده ساز تنسوری تعبیه می شود.

$$\mathcal{Z} \in \mathbb{R}^{B \times P_1 \times P_2 \times D_1 \times D_2 \times D_3}$$

در این ساختار، هر پچ به جای تبدیل به یک بردار تخت، به یک تانسور سه بعدی در فضای نهفته تبدیل می شود. این تانسور ساختار درونی پچ را حفظ کرد می کند.

فرمول بندی ریاضی عملیات فشرده سازی

فرض می کنیم $\mathcal{X}_{patch} \in \mathbb{R}^{C_1 \times C_2 \times C_3}$ نمایانگر یک پچ ورودی باشد. برای فشرده سازی این تانسور به فضای نهفته (D_1, D_2, D_3) ، از سه ماتریس فشرده سازی قابل آموختن استفاده می شود:

$$V^{(0)} \in \mathbb{R}^{D_1 \times C_1}, \quad V^{(1)} \in \mathbb{R}^{D_2 \times C_2}, \quad V^{(2)} \in \mathbb{R}^{D_3 \times C_3}$$

عملیات فشرده‌سازی با استفاده از ضرب‌های چندحاله^{۲۳} صورت می‌گیرد:

$$\mathcal{X}_{emb} = \mathcal{X}_{patch} \times_0 V^{(0)} \times_1 V^{(1)} \times_2 V^{(2)} \quad (3.0.9)$$

که در آن $\mathcal{X}_{emb} \in \mathbb{R}^{D_1 \times D_2 \times D_3}$ نگاشت نهفته‌ی تansوری برای آن پچ خواهد بود.

در این مدل ساختار چند بعدی به پچ‌ها داده می‌شود و همچنین با استفاده از تansور تعداد پارامتر کاهش پیدا می‌کنند و همچنین ظرفیت مدل نیز افزایش پیدا می‌کند.

در نتیجه، خروجی نهایی مرحله‌ی تعبیه پچ به صورت زیر تعریف می‌شود:

$$\mathcal{Z} \in \mathbb{R}^{B \times P_1 \times P_2 \times D_1 \times D_2 \times D_3}$$

که در آن هر پچ به صورت یک تansور کم بعد در فضای نهفته نمایش داده شده است.

۶.۰.۳ ماژول توجه سلف چندسری مبتنی بر پنجره به صورت تansوری

همانطور که در فصل قبل بیان شد در ساختار مبدل پنجره‌ای، جهت کاهش پیچیدگی محاسباتی، از روشی با عنوان^{۲۴} بهره گرفته می‌شود. در این روش، ویژگی‌های مکانی تصویر به پنجره‌های کوچک با اندازه‌ی ثابت (بدون همپوشانی) تقسیم شده و سپس عملیات خودتوجهی به صورت محلی و درون هر پنجره مستقل انجام می‌گیرد.

در روش تansوری، به جای تولید بردارهای تخت‌شده‌ی Q ، K و V ، این بردارها به صورت تansورهای چندبعدی با استفاده از لایه فشرده ساز تansوری تولید می‌شوند.

n-mode product^{۲۳}
Window-based Multi-Head Self-Attention^{۲۴}

ساختار ورودی و تقسیم به پنجره‌ها

خروجی مرحله تعبیه به صورت تانسوری به صورت زیر است

$$\mathcal{X} \in \mathbb{R}^{B \times H \times W \times D_1 \times D_2 \times D_3}$$

در گام اول، همانند مبدل‌های پنجره‌ای به پنجره‌های $w \times w$ تقسیم می‌شوند. و خروجی ما به صورت زیر خواهد بود.

$$\mathcal{X}_{win} \in \mathbb{R}^{B \times N_H \times N_W \times w \times w \times D_1 \times D_2 \times D_3}$$

که در آن $N_H = \frac{H}{w}$ و $N_W = \frac{W}{w}$ به ترتیب تعداد پنجره‌ها در راستای ارتفاع و عرض تصویر هستند.

۲. محاسبه‌ی K و V با استفاده از لایه فشرده ساز تانسوری

برای هر پنجره‌ی مکانی، سه لایه‌ی فشرده‌سازی تانسوری مستقل (TCL) برای استخراج تانسورهای K و V طراحی شده‌اند. فرض می‌شود تانسور ورودی هر پنجره دارای ابعاد:

$$\mathcal{X}_{patch} \in \mathbb{R}^{w \times w \times D_1 \times D_2 \times D_3}$$

باشد. با اعمال سه عملیات ضرب مد- n به ترتیب در ابعاد تانسوری، نگاشتهای K و V و Q به صورت زیر حاصل می‌شوند:

$$Q = \mathcal{X}_{patch} \times_2 V_Q^{(1)} \times_3 V_Q^{(2)} \times_4 V_Q^{(3)} \quad (3.0.10)$$

$$K = \mathcal{X}_{patch} \times_2 V_K^{(1)} \times_3 V_K^{(2)} \times_4 V_K^{(3)} \quad (3.0.11)$$

$$\mathcal{V} = \mathcal{X}_{patch} \times_2 V_V^{(1)} \times_3 V_V^{(2)} \times_4 V_V^{(3)} \quad (3.0.12)$$

که در آن هر $V^{(i)}$ ماتریس فشرده‌سازی با ابعاد $\mathbb{R}^{D_i \times D_i}$ است. بنابراین، خروجی هر لایه به

فضای تانسوری مشابه با ورودی بازمی‌گردد:

$$\mathcal{Q}, \mathcal{K}, \mathcal{V} \in \mathbb{R}^{w \times w \times D_1 \times D_2 \times D_3}$$

۳. تقسیم سرهای توجه چندگانه

برای پیاده‌سازی مکانیزم چندسر^{۲۵}، ابعاد تانسوری خروجی به بخش‌های کوچک‌تری تقسیم می‌شود.

فرض می‌شود:

$$D_1 = h_1 \cdot d_1, \quad D_2 = h_2 \cdot d_2, \quad D_3 = h_3 \cdot d_3$$

که در آن:

تعداد سرهای در هر بعد تانسوری h_1, h_2, h_3 ●

ابعاد نهفته‌ی هر سر در هر بعد هستند. d_1, d_2, d_3 ●

با استفاده از عملیات بازارایی، تانسورهای K و Q به شکل زیر سازمان‌دهی می‌شوند:

$$\mathcal{Q}_{head} \in \mathbb{R}^{B \times N_H \times N_W \times w \times w \times h_1 \times h_2 \times h_3 \times d_1 \times d_2 \times d_3}$$

و ساختار مشابهی برای K و V در نظر گرفته می‌شود.

multi-head^{۲۵}

۴. محاسبه ماتریس توجه تانسوری

عملیات ضرب داخلی تعمیم‌یافته بین تانسورهای Q و K برای محاسبه ماتریس توجه انجام می‌شود. برای هر سر (a, b, c) و برای موقعیت‌های (i, j) و (k, l) در پنجره، مقدار توجه به صورت زیر محاسبه می‌شود:

$$\text{Attention}_{ijkl}^{abc} = \sum_{x=1}^{d_1} \sum_{y=1}^{d_2} \sum_{z=1}^{d_3} Q_{ij}^{abcxyz} \cdot K_{kl}^{abcxyz} \quad (3.0.13)$$

که نتیجه نهایی یک ماتریس توجه با ابعاد زیر است:

$$\text{Attention} \in \mathbb{R}^{B \times N_H \times N_W \times h_1 \times h_2 \times h_3 \times w^2 \times w^2}$$

برای پایدارسازی محاسبات، نرمال‌سازی زیر اعمال می‌شود:

$$\text{scale} = \frac{1}{\sqrt{d_1 d_2 d_3}} \quad (3.0.14)$$

و از نسخه علامت‌دار تابع softmax برای اعمال توجه استفاده می‌گردد:

$$\text{Attention}_{\text{soft}} = \text{sign}(A) \cdot \text{softmax}(|A|) \quad (3.0.15)$$

۵. اعمال توجه و بازسازی ویژگی

پس از محاسبه ماتریس توجه، تانسور خروجی هر پنجره با استفاده از ترکیب توجه و تانسور V محاسبه می‌شود:

$$Y_{ij}^{abcxyz} = \sum_{k,l} \text{Attention}_{ijkl}^{abc} \cdot V_{kl}^{abcxyz} \quad (3.0.16)$$

و پس از انجام توجه برای هر پنجه دوباره باز آرایی می‌شوند و به حالت اولیه باز می‌گردند و ابعاد آن به صورت زیر می‌شود.

$$\mathcal{Y} \in \mathbb{R}^{B \times H \times W \times D_1 \times D_2 \times D_3}$$

در این مدل چون سرها در سه بعد تفسیم می‌شوند در نتیجه این مدل آزادی خیلی بیشتری دارد.

۷.۰.۳ توجه علامت‌دار

در مکانیزم‌های استاندارد خود توجهی که در مبدل‌ها مورد استفاده قرار می‌گیرند، ماتریس‌های Q ، V و K با یکدیگر تعامل دارند تا وزن‌های توجه استخراج شوند. در این فرآیند، بردارهای Q ^{۲۶} و K

با استفاده از ضرب داخلی محاسبه می‌شوند و نتیجه برای هر توکن، بیان‌گر میزان اهمیت نسبی سایر توکن‌ها در ارتباط با آن توکن است.^{۲۷} فرمول رایج محاسبه توجه به صورت زیر تعریف می‌شود:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \quad (۳.۰.۱۷)$$

در این ساختار، خروجی QK^\top می‌تواند شامل مقادیر مثبت، منفی یا صفر باشد؛ اما پس از اعمال تابع Softmax، تمامی مقادیر به صورت نرمال‌سازی شده و مثبت خواهند بود. در نتیجه، اطلاعات قطبیت (علامت) که در نتیجه‌ی ضرب داخلی بین بردارهای Q و K وجود داشت، از بین می‌رود. این ویژگی باعث محدود شدن ظرفیت مدل در شناسایی «رابطه منفی یا متضاد» میان توکن‌ها می‌شود. برای حفظ این اطلاعات قطبیت، د از رویکردی با عنوان توجه علامت دار استفاده شده است. ایده‌ی اصلی این روش آن است که پس از محاسبه امتیازهای توجه $A = QK^\top$ ، عملیات

۳. روش‌های پیشنهادی

نرمال‌سازی Softmax بر قدر مطلق این امتیازها انجام شده و سپس علامت اولیه آن‌ها به نتیجه بازگردانده می‌شود.

فرمول این روش به صورت زیر تعریف می‌شود:

$$\text{Attention}_{\text{sign}} = \text{sign}(A) \cdot \text{Softmax}(|A|) \quad (3.0.18)$$

که در آن:

$A = QK^T$ • ماتریس امتیاز‌های اولیه توجه است.

$\text{sign}(A)$ عملیاتی است که علامت هر مقدار را حفظ می‌کند (مقادیر $+1$ ، -1 یا 0).

$|A|$ قدر مطلق مقادیر است که شدت شباهت را بدون در نظر گرفتن جهت نشان می‌دهد.

برای درک بهتر تفاوت میان مکانیزم‌های توجه معمولی و توجه علامت دار یک مثال عددی ساده اما گویای زیر ارائه می‌شود.

فرض کنیم بردار پرس‌وجو (Q) و دو بردار کلید (K) به صورت زیر تعریف شده‌اند:

$$Q = [1, 2, -1], \quad K_1 = [1, 0, -1], \quad K_2 = [-1, 1, 1]$$

محاسبه امتیاز توجه ابتدا شباهت میان Q و هر کلید با استفاده از ضرب داخلی محاسبه می‌شود:

$$Q \cdot K_1 = (1)(1) + (2)(0) + (-1)(-1) = 1 + 0 + 1 = 2$$

$$Q \cdot K_2 = (1)(-1) + (2)(1) + (-1)(1) = -1 + 2 - 1 = 0$$

بنابراین بردار امتیاز توجه به صورت زیر حاصل می‌شود:

$$A = [2, 0]$$

۳. روش‌های پیشنهادی

الف) توجه معمولی: در توجه معمولی، از تابع softmax مستقیماً روی مقادیر A استفاده می‌شود:

$$\text{Softmax}(A) = \left[\frac{e^2}{e^2 + e^0}, \frac{e^0}{e^2 + e^0} \right] \approx [0.88, 0.12]$$

در اینجا، هر دو کلید مقدار وزن مثبت می‌گیرند، حتی اگر امتیاز شباهت دومی برابر با صفر بوده باشد. مدل نمی‌تواند تفاوت میان بی‌اثر بودن و تأثیر منفی یا متضاد را به خوبی درک کند.

ب) توجه علامت‌دار: در این رویکرد، ابتدا بردار A به دو بخش علامت و قدرمطلق تفکیک می‌شود:

$$\text{sign}(A) = [1, 0], \quad |A| = [2, 0]$$

سپس تابع softmax فقط روی قدرمطلق‌ها اعمال شده و خروجی زیر حاصل می‌شود:

$$\text{Softmax}(|A|) = \left[\frac{e^2}{e^2 + e^0}, \frac{e^0}{e^2 + e^0} \right] \approx [0.88, 0.12]$$

در نهایت، با ضرب عنصر به عنصر با علامت‌ها:

$$\text{Attention}_{\text{sign}} = \text{sign}(A) \cdot \text{Softmax}(|A|) = [0.88, 0.00]$$

در این حالت، کلید دوم به طور کامل کنار گذاشته شده است، چرا که امتیاز آن صفر بوده و نشانی از تأثیر مثبت یا منفی ندارد.

در توجه معمولی، تمامی مقادیر صفر یا منفی به وزن‌های مثبت نرمال‌سازی می‌شوند. در مقابل، توجه علامت‌دار توانایی تمایز بین ارتباط مثبت، منفی و خنثی را دارد. این قابلیت می‌تواند در مدل‌سازی دقیق‌تر تضادهای معنایی یا شباهت‌های منفی نقش مهمی ایفا کند.

۸.۰.۳ توجه مبتنی بر پنجره‌های جابه‌جاشده به صورت تانسوری

توجه پنجره‌ای دارای یک محدودیت مهم است: پچ‌های واقع در پنجره‌های متفاوت هیچ‌گونه تبادل اطلاعاتی ندارند. در حالی که ممکن است این پچ‌ها به هم نزدیک باشند به منظور رفع این محدودیت، مبدل‌های پنجره‌ای مکانیزم توجه مبتنی بر پنجره جا به جا شده معرفی شده است که در آن پنجره‌ها در برخی لایه‌ها به صورت جابه‌جاشده تعریف می‌شوند و از ماسک توجه برای جلوگیری از تداخل غیرمجاز استفاده می‌شود.

مراحل دقیق مکانیزم

۱. شیفت چرخشی تانسور ورودی: خروجی مرحله قبلی با یک شیفت چرخه‌ای^{۲۸} به اندازه $\left[\frac{w}{2}\right]$ پیکسل در راستای افقی و عمودی جابه‌جا می‌شود. این جابه‌جایی باعث می‌شود که پچ‌هایی که پیش‌تر در پنجره‌های جداگانه بودند، اکنون در یک پنجره جدید قرار گیرند. به این شیفت روی بعد اول و دوم تانسور که مکان پچ را مشخص می‌کرد صورت می‌گیرد.

۲. اعمال پنجره‌بندی جدید: تانسور جابه‌جاشده به پنجره‌های بدون همپوشانی جدید با اندازه $w \times w$ تقسیم می‌شود. پنجره‌های جدید از موقعیت‌های مختلف تصویر تشکیل شده‌اند.

۳. اعمال خودتوجه‌ی به صورت محلی با ماسک: از آنجا که پنجره‌ها پس از شیفت ممکن است شامل توکن‌هایی از مکان‌های نامرتبط باشند، نیاز است که توجه به صورت مفید انجام شود. یک ماسک دودویی روی ماتریس توجه اعمال می‌شود که فقط اجازه توجه به توکن‌هایی را می‌دهد که واقعاً در یک پنجره معتبر قرار دارند.

خودتوجه‌ی پنجره جابه‌جا شده باعث ارتباط میان پچ‌های نزدیکی می‌شود که در خودتوجه‌ی پنجره‌ای در یک پنجره قرار نداشته‌اند.

cyclic shift^{۲۸}

۹.۰.۳ ادغام پچ‌ها به صورت تانسوری

در مدل‌های سلسله‌مراتبی بینایی مانند پس از هر مرحله از استخراج ویژگی، لازم است ابعاد مکانی تصویر کاهش یابد تا ویژگی‌ها در سطوح بالاتر به صورت فشرده‌تر و معنایی‌تر نمایش داده شوند. این فرایند با عنوان ادغام پچ‌ها^{۲۹} شناخته می‌شود. در نسخه‌ی اصلی این مدل، ادغام پچ‌ها با استفاده از ترکیب \mathcal{P} پچ مجاور (به صورت یک پنجره‌ی 2×2) و سپس اعمال یک لایه‌ی خطی پس از مسطح‌سازی انجام می‌شود. این کار باعث نصف شدن ابعاد مکانی و افزایش ابعاد ویژگی می‌شود. در این پژوهش، رویکرد متفاوتی برای ادغام پچ‌ها ارائه شده است که مبتنی بر نگاشت تانسوری است. به جای انجام^{۳۰} بر روی پچ‌ها، از ساختار چندبعدی پچ‌ها استفاده شده و با کمک یک لایه‌ی فشرده‌سازی تانسوری، ادغام پچ‌ها بدون از بین رفتن ساختار چندخطی آن‌ها انجام می‌شود.

ساختار ورودی

مانند خروجی مرحله قبلی شبکه دارای ساختار تانسوری زیر باشد:

$$\mathcal{X} \in \mathbb{R}^{B \times H \times W \times R_1 \times R_2 \times C}$$

که در آن:

- B اندازه دسته آموزشی،

- ابعاد مکانی پچ‌ها،

- ابعاد فضای نهفته‌ی هر پچ به صورت تانسوری.

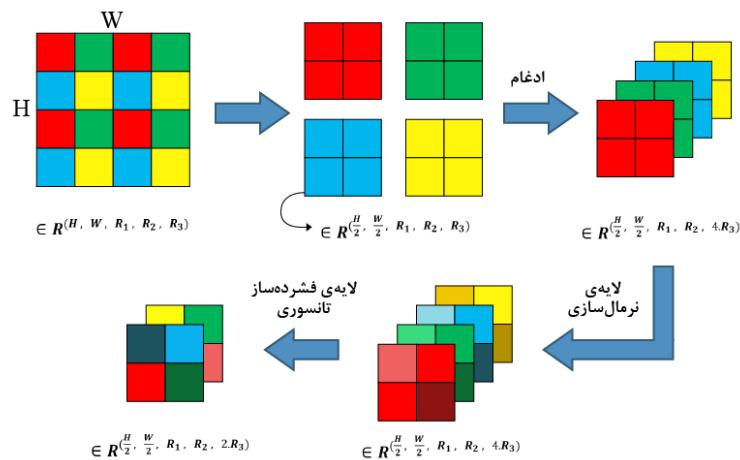
هدف این مرحله، کاهش ابعاد مکانی به $(\frac{H}{2}, \frac{W}{2})$ و در عین حال، افزایش غنای بازنمایی ویژگی‌ها است.

Patch Merging^{۲۹}
flatten^{۳۰}

ادغام پچ های مجاور

برای کاهش ابعاد مکانی، ابتدا هر چهار پچ مجاور در پنجره‌ای 2×2 انتخاب می‌شوند. این چهار پچ به صورت زیر دسته‌بندی می‌شوند:

- پچ بالا-چپ،
- پچ بالا-راست،
- پچ پایین-چپ،
- پچ پایین-راست.



شكل ۳.۰.۴: ادغام پچ ها

در مرحله بعد، این چهار پچ با یکدیگر ترکیب می‌شوند، به این صورت که هر کدام به صورت مستقل نگه داشته شده و در امتداد بعد کانالی (C) به یکدیگر متصل می‌شوند. در نتیجه، در هر موقعیت مکانی جدید، تانسوری با ابعاد $(R_1, R_2, 4C)$ خواهیم داشت. در سطح کلان، خروجی میان مرحله‌ای به صورت زیر خواهد بود:

$$\mathcal{X}_{\text{merged}} \in \mathbb{R}^{B \times \frac{H}{2} \times \frac{W}{2} \times R_1 \times R_2 \times 4C}$$

فسردهسازی ویژگی‌های ادغام‌شده

پس از ادغام کانالی، لازم است ابعاد ویژگی‌ها به صورت کنترل شده کاهش یابد تا بتوان آن را به شکل ورودی ماذول‌های بعدی تطبیق داد. برای این منظور، از یک لایه‌ی فشردهسازی تانسوری استفاده می‌شود که به صورت مستقیم بر روی ابعاد نهفته $(R_1, R_2, 4C)$ عمل می‌کند. برخلاف لایه‌های خطی کلاسیک، در اینجا عملیات فشردهسازی در چندین بعد به صورت همزمان و با حفظ ساختار چندخطی انجام می‌شود.

خروجی حاصل، تانسوری با ساختار:

$$\mathcal{Y} \in \mathbb{R}^{B \times \frac{H}{2} \times \frac{W}{2} \times R'_1 \times R'_2 \times C'}$$

که در آن (R'_1, R'_2, C') ابعاد جدید ویژگی‌ها هستند و با توجه به محدودیت طراحی، باید رابطه زیر میان ابعاد قدیم و جدید برقرار باشد:

$$2 \cdot R'_1 \cdot R'_2 \cdot C' = 4 \cdot R_1 \cdot R_2 \cdot C \quad (3.0.19)$$

این رابطه تضمین می‌کند که ادغام از نظر اطلاعاتی قابل پشتیبانی و از نظر مدل‌سازی متوازن باقی می‌ماند.

۱۰۰.۳ مرحله‌ی طبقه‌بندی نهایی با استفاده از رگرسیون تانسوری

پس از عبور تصویر ورودی از مراحل سلسله‌مراتبی مختلف شامل تعییه پچ، ماذول‌های خودتوجهی تانسوری، و لایه‌های ادغام و فشردهسازی، خروجی نهایی آخرین مرحله به صورت یک تانسور

۳. روش‌های پیشنهادی

چندبعدی در فضای نهفته حاصل می‌شود. این تانسور حاوی بازنمایی غنی از ویژگی‌های سطح بالا و ساختار درونی تصویر است که برای تصمیم‌گیری نهایی مورد استفاده قرار می‌گیرد.

میانگین‌گیری تانسوری در ابعاد مکانی

فرض میکنیم خروجی مرحله‌ی نهایی دارای ساختار تانسوری زیر باشد:

$$\mathcal{X} \in \mathbb{R}^{B \times H \times W \times D_1 \times D_2 \times D_3}$$

برای استخراج یک بازنمایی کلی از کل تصویر، میانگین‌گیری در دو بعد مکانی (H, W) انجام می‌شود. در واقع همانند مبدل پنجره‌ای متحرک از مکان‌های پچ‌ها میانگین‌گیری گرفته می‌شود. این عملیات باعث فشرده‌سازی اطلاعات مکانی و حفظ ساختار تانسوری در فضای ویژگی می‌شود.

$$\bar{\mathcal{X}} = \frac{1}{H \cdot W} \sum_{i=1}^H \sum_{j=1}^W \mathcal{X}_{[:, i, j, :, :, :]} \in \mathbb{R}^{B \times D_1 \times D_2 \times D_3} \quad (3.0.20)$$

این عملیات نوعی تجمعی جهانی در ابعاد فضایی محسوب می‌شود که مشابه با میانگین‌گیری سراسری^{۳۱} در شبکه‌های پیچشی عمل می‌کند، با این تفاوت که ساختار تانسوری ویژگی‌ها را حفظ می‌نماید.

طبقه‌بندی با استفاده از لایه رگرسیون تانسوری

برای نگاشت خروجی $\bar{\mathcal{X}}$ به برچسب نهایی، از یک لایه رگرسیون تانسوری (TRL) استفاده شده است. این لایه با بهره‌گیری از ضرب داخلی تعمیم‌یافته میان تانسور ورودی و تانسور وزن، نگاشتی خطی از فضای تانسوری به فضای خروجی اعمال می‌کند:

$$\mathbf{Y} = \langle \bar{\mathcal{X}}, \mathcal{W} \rangle_3 + \mathbf{b} \quad (3.0.21)$$

^{۳۱}global average pooling

که در آن

$\bar{\mathcal{X}} \in \mathbb{R}^{B \times D_1 \times D_2 \times D_3}$ ● بازنمایی فشرده‌ی ورودی،

$\mathcal{W} \in \mathbb{R}^{D_1 \times D_2 \times D_3 \times C}$ ● تansور وزن‌های رگرسیون،

$\langle \cdot, \cdot \rangle_3$ ضرب داخلی تعمیم‌یافته در سه بعد آخر $\bar{\mathcal{X}}$ و سه بعد اول \mathcal{W} است، ●

$\mathbf{b} \in \mathbb{R}^C$ ● بردار بایاس،

● C تعداد کلاس‌های خروجی است.

فصل ۲

نتایج و تحلیل‌ها

۱.۴ ارزیابی بر روی مجموعه‌داده CIFAR-10

به منظور ارزیابی عملکرد مدل پیشنهادی، دو پیکربندی مجزا مورد مقایسه قرار گرفتند:

۱. مدل مبنا: Tiny Swin Transformer به عنوان معماری اصلی بدون اعمال تغییرات.
۲. مدل پیشنهادی: Tensorized Swin Transformer که در آن لایه‌های Patch Embedding و Patch Merging به کمک تکنیک فشرده‌سازی تansوری بازطراحی شده‌اند.

۱.۱.۴ خلاصه نتایج کمی

جدول ۴.۱.۱ نتایج کمی این دو مدل را در شاخص‌های دقت Top-1 و Top-5 برای داده‌های آموزش و آزمون نشان می‌دهد. در این جدول ترتیب ستون‌ها به گونه‌ای تنظیم شده است که مقایسه میان عملکرد آموزشی و آزمونی به صورت همزمان و از راست به چپ قابل مشاهده باشد.

۴. نتایج و تحلیل‌ها

جدول ۴.۱.۱: مقایسه عملکرد مدل اصلی و مدل پیشنهادی بر روی مجموعه داده CIFAR-10 بر حسب دقتهای Top-1 و Top-5.

مدل	#پارامترها	داده آزمون			
		Top-1	Top-5	Top-1	Top-5
Tiny Swin	27,528,690	97.48%	99.97%	80.92%	96.45%
Tensorized Swin	1,368,626	80.30%	98.97%	81.80%	99.21%

۲.۱.۴ تحلیل نتایج

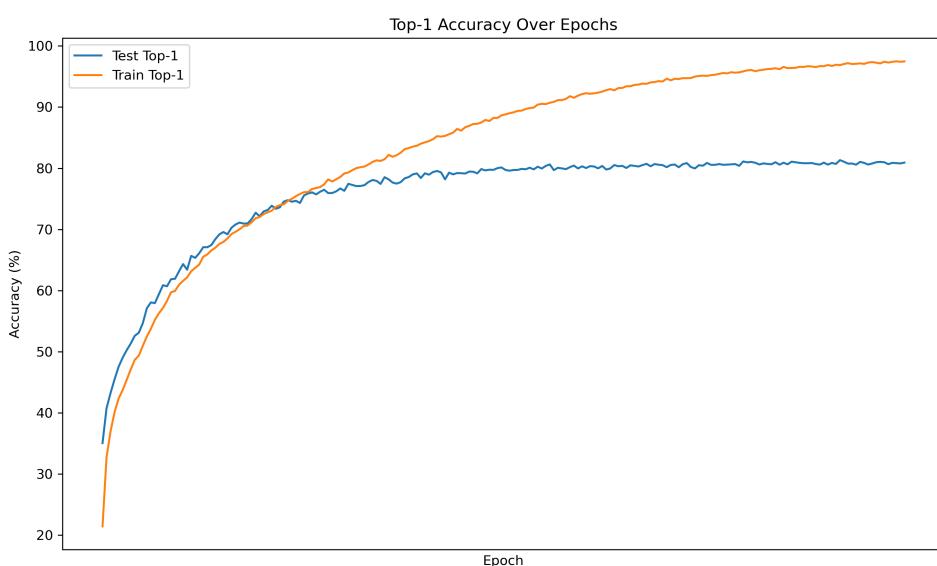
مطابق داده‌های جدول ۴.۱.۱، مدل پیشنهادی با وجود کاهش چشمگیر تعداد پارامترها (از حدود 27.5M به حدود 1.37M کاهش 95%)، توانسته است دقتهای آزمون را حفظ یا حتی اندکی بهبود بخشد. در شاخص Top-1، دقتهای مدل اصلی در داده آزمون برابر 80.92% بوده که در مدل تانسوری به 81.80% رسیده است. همچنین در شاخص Top-5، بهبود محسوس‌تری مشاهده می‌شود؛ به طوری که مقدار دقتهای آزمون از 96.45% به 99.21% افزایش یافته است.

از سوی دیگر، بررسی شکاف آموزش-آزمون نشان می‌دهد که مدل اصلی با اختلاف 16.56 واحد درصد بین دقتهای آموزشی (97.48%) و آزمونی (80.92%) دچار بیش‌برازش (Overfitting) بوده است. در حالی که مدل تانسوری نه تنها چنین شکافی ندارد، بلکه دقتهای آزمونی آن اندکی از دقتهای آموزشی بیشتر است (اختلاف ۱.۵- واحد درصد). این امر بیانگر وجود نوعی منظم‌سازی ذاتی (Low-Rank) ناشی از فشرده‌سازی تانسوری و اعمال قیود کم‌مرتبه (Implicit Regularization) است. Constraints

بهبود محسوس در Top-5 نیز حاکی از آن است که مدل پیشنهادی فضای ویژگی غنی‌تری ایجاد کرده که امکان پوشش کلاس‌های صحیح در میان پنج پیش‌بینی برتر را افزایش داده و در نتیجه اطمینان کلی تصمیم‌گیری مدل را ارتقاء داده است.

۳.۱.۴ نمایش روند آموزش

برای درک بهتر فرایند همگرایی، روند تغییرات دقت Top-1 در طول آموزش برای هر دو مدل در شکل‌های ۴.۱.۱ و ۴.۱.۲ نمایش داده شده است. این نمودارها نشان می‌دهند که مدل اصلی به سرعت به دقت بالایی در داده آموزشی می‌رسد ولی در داده آزمون افت می‌کند، در حالی که مدل تansوری با روندی یکنواخت‌تر و پایدارتر به دقت نهایی می‌رسد.

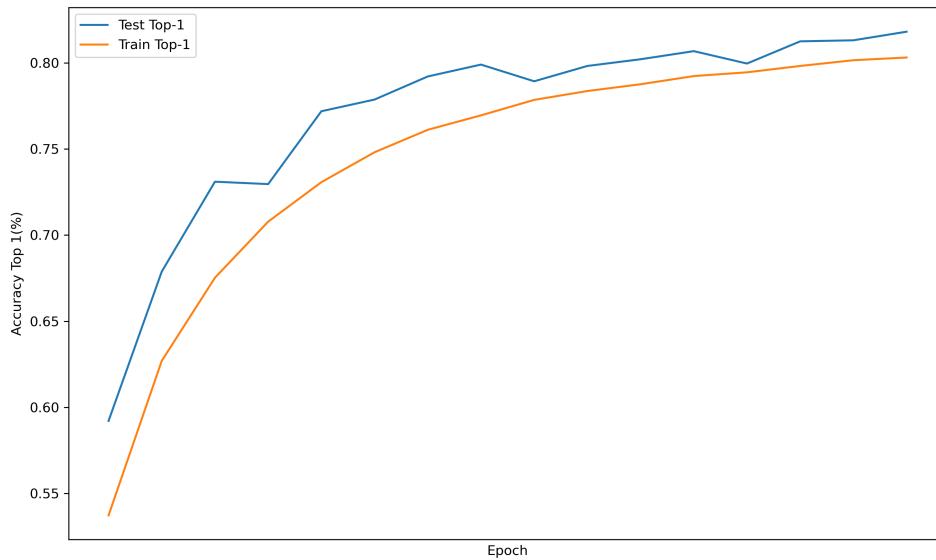


شکل ۴.۱.۱: روند تغییرات دقت Top-1 مدل اصلی Swin-Tiny بر روی مجموعه‌داده CIFAR-10.

۴.۱.۴ جمع‌بندی

به‌طور کلی، نتایج به‌دست‌آمده نشان می‌دهد که استفاده از ساختار تansوری در بخش‌های کلیدی مدل، علاوه بر کاهش چشم‌گیر پیچیدگی محاسباتی و نیاز حافظه، باعث بهبود تعمیم‌پذیری و کاهش بیش‌برازش نیز می‌شود. این امر اهمیت به کارگیری روش‌های فشرده‌سازی ساختاریافته را در طراحی مدل‌های کارا برای کاربردهای کم منبع تأیید می‌کند.

۴. نتایج و تحلیل‌ها



شکل ۴.۱.۲: روند تغییرات دقت Top-1 مدل پیشنهادی Tensorized Swin بر روی مجموعه داده CIFAR-10.

۲.۴ نتایج بر روی دیتاست MNIST

برای ارزیابی عملکرد مدل پیشنهادی بر روی داده‌های ساده‌تر و با ابعاد کوچک‌تر، آزمایش‌ها بر روی دیتاست MNIST نیز انجام شده است. در این ارزیابی دو پیکربندی زیر مقایسه شده‌اند:

۱. مدل اصلی (Tiny Swin Transformer)

۲. مدل پیشنهادی تانسوری (Tensorized Swin Transformer)

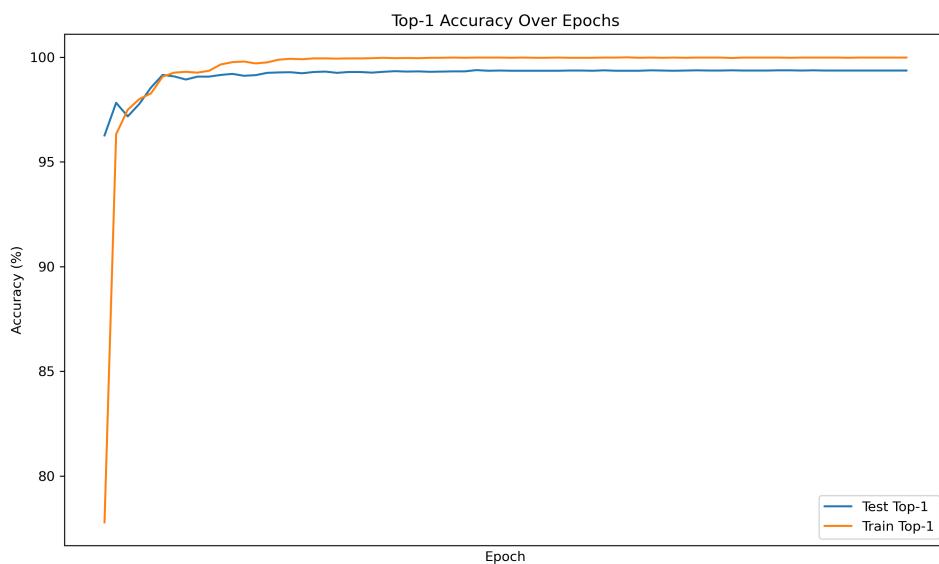
۱.۲.۴ خلاصه‌ی نتایج کمی

۲.۲.۴ نمایش روند آموزش

شکل‌های ۴.۲.۳ و ۴.۲.۴ روند تغییرات دقت Top-1 را برای مدل اصلی و مدل تانسوری بر روی مجموعه داده MNIST نشان می‌دهند. هر دو مدل به سرعت به دقت بسیار بالا رسیده‌اند، اما مدل تانسوری با وجود تعداد پارامترهای بسیار کمتر، دقت نهایی بالاتری در داده‌های آزمون کسب کرده است.

جدول ۴.۲.۲: مقایسه‌ی عملکرد مدل اصلی و مدل تانسوری بر روی MNIST (فقط Top-۱ و Top-۵).

مدل	#پارامترها	آموزش		تست	
		Top-۱	Top-۵	Top-۱	Top-۵
Tiny Swin	27,528,690	95.8%	99.9%	97.0%	99.9%
Tensorized Swin	1,368,626	97.3%	99.9%	98.9%	100%



شکل ۴.۲.۳: روند دقت Top-1 مدل اصلی Swin-Tiny بر روی MNIST

۴.۲.۴ تحلیل و بحث

نتایج جدول ۴.۲.۲ نشان‌دهنده‌ی نکات مهم زیر است:

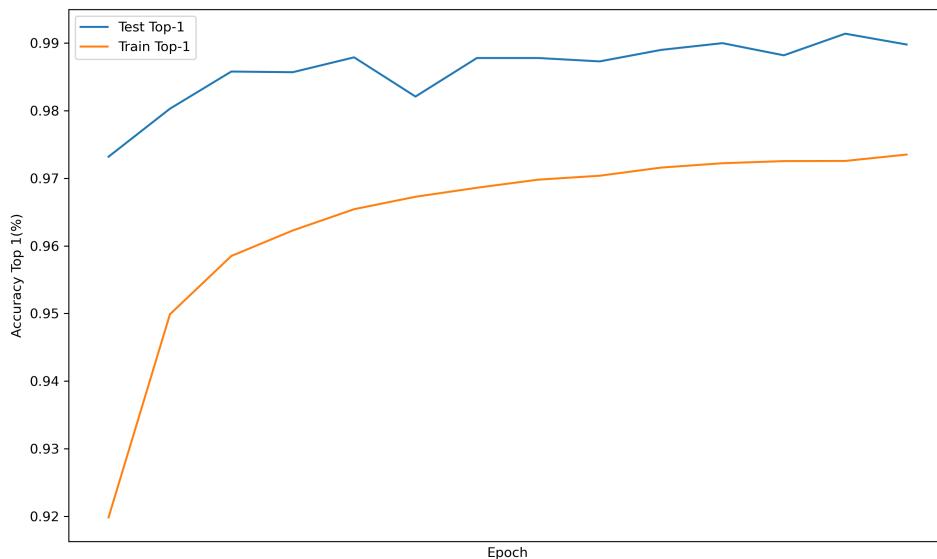
- کاهش چشمگیر پارامترها: مدل تانسوری فقط 1,368,626 پارامتر دارد در حالی‌که مدل اصلی

- ۲۰.۱ حدوداً برابر (نزدیک به 95%) کاهش در تعداد 27,528,690 پارامتر دارد؛ یعنی کاهش حدوداً ۲۰.۱ برابر (نزدیک به 95%) کاهش در تعداد پارامترها).

- بهبود دقت آزمون: دقت Top-1 آزمون از 97.0% در مدل اصلی به 98.9% در مدل تانسوری

- رسیده است (بهبود مطلق +1.9 واحد درصد). همچنین دقت Top-5 از 99.9% به 100%

۴. نتایج و تحلیل‌ها



شکل ۴.۲.۴: روند دقت Top-1 مدل تansوری پیشنهادی بر روی MNIST.

افزایش یافته است.

- کاهش شکاف آموزش-آزمون: در مدل اصلی شکاف بین آموزش و آزمون برای Top-1 برابر ۱.۲ واحد درصد است (۰.۹۷ در برابر ۰.۹۵). در مدل تansوری این شکاف به ۱.۶ واحد درصد رسیده است که نشان‌دهنده عملکرد پایدارتر و تعمیم‌پذیری بهتر است.

- پایداری در داده‌های ساده‌تر: با وجود سادگی مجموعه‌داده‌ی MNIST، مدل تansوری همچنان توانسته است دقت را نسبت به مدل اصلی بهبود دهد، که بیانگر قدرت فشرده‌سازی تansوری در حفظ یا حتی افزایش دقت در کنار کاهش منابع محاسباتی است.

جمع‌بندی. بر اساس نتایج به‌دست‌آمده، مدل تansوری نه تنها حجم مدل را به‌شدت کاهش داده است، بلکه توانسته در یک مجموعه‌داده‌ی ساده مانند MNIST، دقت آزمون را نسبت به مدل اصلی افزایش دهد. این نتایج مؤید این است که فشرده‌سازی تansوری می‌تواند حتی در شرایطی که داده‌ها ساده هستند نیز به عنوان یک رویکرد بهینه و کارا مورد استفاده قرار گیرد.

۳.۴ نتایج بر روی دیتابست Tiny ImageNet

برای بررسی عملکرد مدل در یک سناریوی چالش‌برانگیزتر، آزمایش‌هایی بر روی دیتابست Tiny ImageNet انجام شد. در این بخش، علاوه بر مدل اصلی Tiny Swin Transformer، مدل پیشنهادی تansوری نیز با دو بهینه‌ساز متفاوت (AdamW و Adam) ارزیابی شده است.

۱.۳.۴ خلاصه نتایج کمی

جدول ۴.۳.۳: مقایسه‌ی عملکرد مدل‌ها بر روی Tiny ImageNet

بهینه‌ساز	مدل	#پارامتر	آموزش		آزمون	
			Top-1	Top-5	Top-1	Top-5
—	Tiny Swin	27,528,690	42.26%	59.03%	64.14%	85.42%
Adam	Tensorized Swin	1,368,626	83.89%	98.50%	30.85%	54.17%
AdamW	Tensorized Swin	1,368,626	55.30%	80.90%	35.90%	62.07%

۲.۳.۴ نمایش روند آموزش

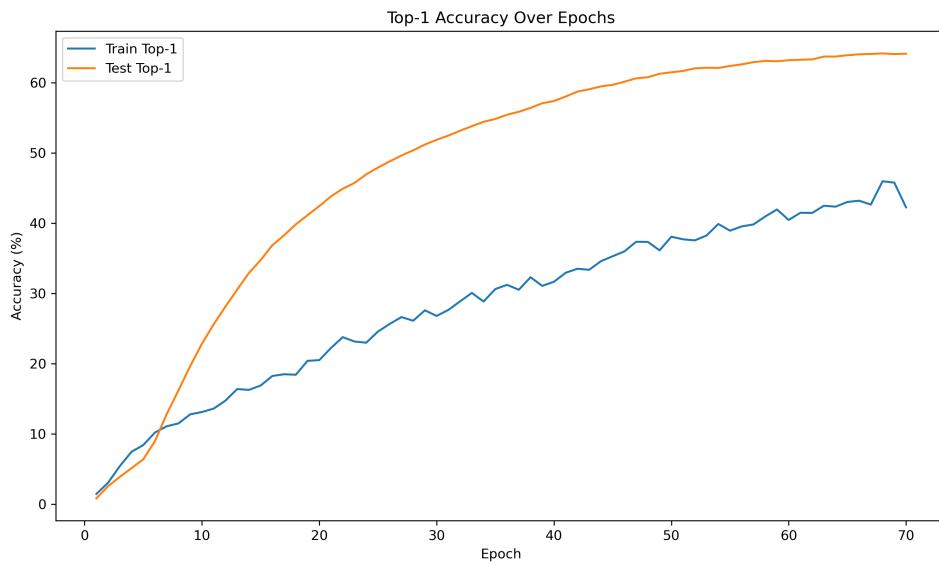
شکل ۴.۳.۵ روند تغییر دقت Top-1 Tiny Swin را برای مدل اصلی نشان می‌دهد. همچنین، شکل‌های ۴.۳.۶ و ۴.۳.۷ مربوط به مدل تansوری با بهینه‌سازهای AdamW و Adam هستند.

۳.۳.۴ تحلیل و بحث

بر اساس جدول ۴.۳.۳ و نمودارهای ارائه شده، نکات زیر قابل توجه است:

- بیش‌بازش در مدل اصلی: مدل Tiny Swin با وجود تعداد بالای پارامترها (27,528,690) توانسته دقت آزمون Top-1 معقولی (64.14%) بدست آورد، اما اختلاف زیاد بین دقت آموزش (42.26%) و آزمون نشان‌دهنده ناتوانی در یادگیری کامل ویژگی‌ها و احتمال وجود محدودیت در داده یا تنظیمات بهینه‌سازی است.

۴. نتایج و تحلیل‌ها

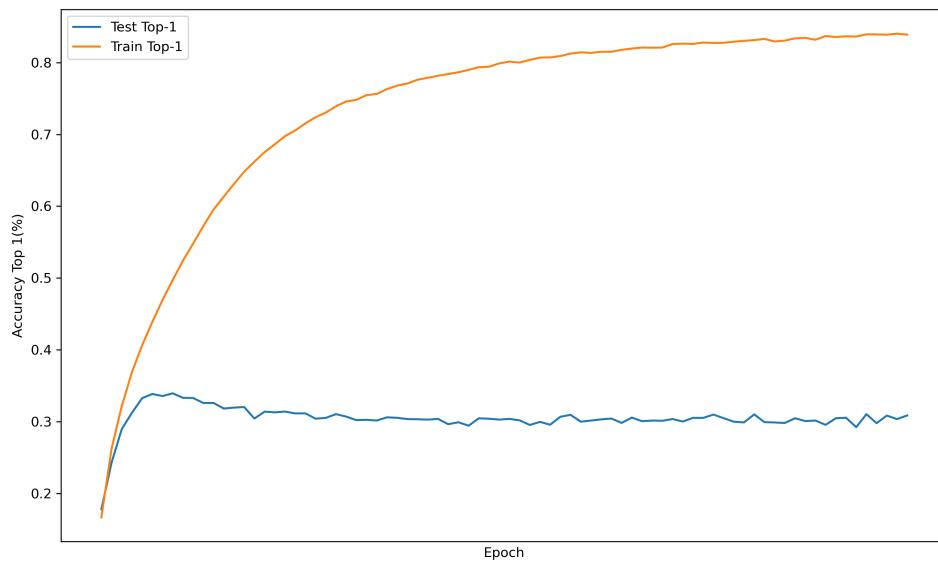


. شکل ۴.۳.۵: روند دقت Top-1 مدل اصلی Tiny Swin بر روی Tiny ImageNet.

- بیش‌برازش شدید در مدل تانسوری با Adam: مدل تانسوری با بهینه‌ساز Adam دقت آموزش بسیار بالایی (83.89%) Top-1 دارد، اما دقت آزمون آن به شدت افت کرده (30.85%) Top-1)، که نشانه‌ی آشکار overfitting است.
- بهبود تعادل با AdamW: تغییر بهینه‌ساز به AdamW باعث شده شکاف آموزش–آزمون کاهش یابد و عملکرد آزمون (35.90%) Top-1 نسبت به حالت Adam بهبود پیدا کند، هرچند همچنان فاصله‌ی زیادی تا مدل اصلی وجود دارد.
- تأثیر پیچیدگی داده: نتایج نشان می‌دهد که کاهش شدید پارامترها در دیتاست‌های پیچیده‌تر مانند Tiny ImageNet ممکن است منجر به کاهش توان مدل در تعمیم‌پذیری شود، مگر این که تکنیک‌های منظم‌سازی و بهینه‌سازی به درستی انتخاب شوند.

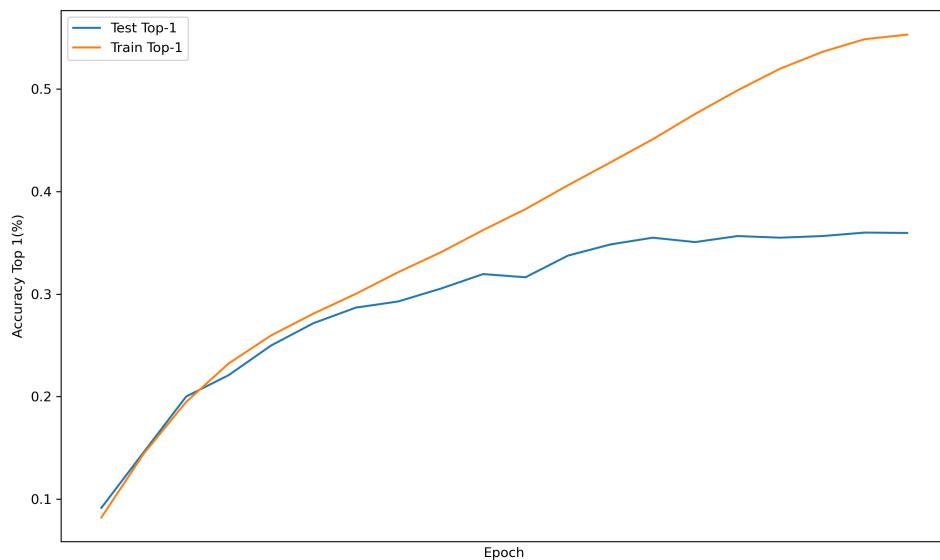
جمع‌بندی. برخلاف نتایج موفقیت‌آمیز بر روی CIFAR-10 و MNIST، در دیتاست Tiny ImageNet کاهش پارامترها و استفاده از فشرده‌سازی تانسوری، به دلیل پیچیدگی بالای داده و محدودیت ظرفیت مدل، منجر به افت عملکرد آزمون شده است. با این حال، استفاده از بهینه‌ساز مانند AdamW

۴. نتایج و تحلیل‌ها



شکل ۴.۳.۶: روند دقت Top-1 مدل تansوری با بهینه‌ساز Adam بر روی Tiny ImageNet.

توانسته تا حدی این شکاف را کاهش دهد.



شکل ۴.۳.۷: روند دقت Top-1 مدل تانسوری با بهینه‌ساز AdamW بر روی Tiny ImageNet

كتاب نامه

- [1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. [37](#), [38](#)
- [2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 2015 International Conference on Learning Representations (ICLR)*, San Diego, CA, 2015. [23](#), [25](#), [26](#), [39](#), [41](#)
- [3] Yoshua Bengio et al. *Learning long-term dependencies with gradient descent is difficult*. IEEE Transactions on Neural Networks, 1994. [35](#), [36](#)
- [4] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006. [6](#), [7](#), [8](#), [13](#), [14](#)
- [5] Peter F Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311, 1993. [26](#)
- [6] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. [12](#)
- [7] Thomas M. Cover and Peter E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967. [11](#), [12](#)
- [8] Daniel Crevier. *AI: The Tumultuous History of the Search for Artificial Intelligence*. Basic Books, New York, 1993. [3](#), [4](#)

- [9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. [25](#), [46](#), [47](#)
- [10] Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2–3):103–130, 1997. [13](#), [14](#)
- [11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, et al. An image is worth 16x16 words: Transformers for image recognition at scale, 2020. [42](#), [43](#), [44](#), [45](#), [46](#), [47](#), [48](#), [50](#)
- [12] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, 1973. [11](#), [12](#)
- [13] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990. [16](#), [26](#), [31](#)
- [14] Edward A. Feigenbaum and Pamela McCorduck. *The Fifth Generation: Artificial Intelligence and Japan’s Computer Challenge to the World*. Addison-Wesley, Reading, MA, 1983. [3](#), [5](#)
- [15] Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. In *Proceedings of the Ninth International Conference on Artificial Neural Networks (ICANN-99)*, pages 850–855, Edinburgh, UK, 1999. [14](#), [16](#), [18](#), [19](#), [20](#)
- [16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, 2016. [6](#), [15](#), [16](#), [17](#), [20](#), [21](#), [22](#)
- [17] Safa Hamreras, Sukhbinder Singh, and Román Orús. Tensorization as a powerful tool for compression and interpretability in neural networks: A position paper. *arXiv preprint arXiv:2505.20132*, 2025. [.](#), [66](#), [67](#)
- [18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. [35](#), [36](#), [43](#), [48](#), [49](#), [57](#), [58](#)

- [19] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. [56](#)
- [20] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(2):107–116, 1998. [16](#), [17](#), [20](#), [22](#)
- [21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. [14](#), [16](#), [17](#), [18](#), [21](#), [31](#), [35](#), [36](#)
- [22] John Hutchins. *Machine translation: past, present, future*. Ellis Horwood Chichester, 1986. [25](#)
- [23] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456, 2015. [37](#), [38](#)
- [24] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer, New York, 2013. [7](#), [12](#)
- [25] Shibo Jie and Zhi-Hong Deng. Fact: Factor-tuning for lightweight adaptation on vision transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023. 0.01% ViT . [67](#)
- [26] Philipp Koehn. *Statistical Machine Translation*. Cambridge University Press, 2010. [26](#)
- [27] Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proc. of NAACL/HLT*, pages 48–54, 2003. [26](#)
- [28] Jean Kossaifi, Aran Khanna, Zachary C. Lipton, Tommaso Furlanello, and Anima Anandkumar. Tensor contraction layers for parsimonious deep nets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017. [61](#), [62](#), [63](#), [64](#), [65](#), [66](#)

- [29] Jean Kossaifi, Zachary C. Lipton, Arinbjørn Kolbeinsson, Aran Khanna, Tommaso Furlanello, and Anima Anandkumar. Tensor regression networks. *Journal of Machine Learning Research*, 21(123):1–21, 2020. [61](#), [62](#), [63](#), [65](#), [66](#)
- [30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. [43](#)
- [31] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan V. Oseledets, and Victor S. Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. In *International Conference on Learning Representations (ICLR)*, 2015. [62](#)
- [32] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. [43](#)
- [33] James Lighthill. *Artificial Intelligence: A General Survey*. HM Stationery Office, London, 1973. Science Research Council Report. [4](#)
- [34] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, 2021. [48](#), [49](#), [51](#), [53](#), [54](#), [55](#), [56](#), [57](#), [58](#), [59](#)
- [35] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *Proc. of EMNLP*, pages 1412–1421, 2015. [26](#)
- [36] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, pages 41–48, Madison, WI, 1998. [13](#), [14](#)
- [37] John McCarthy, Marvin Minsky, Nathaniel Rochester, and Claude E. Shannon. A proposal for the dartmouth summer research project on artificial intelligence. *Dartmouth College AI Archive*, 1956. [3](#)

- [38] Pamela McCorduck. *Machines Who Think: A Personal Inquiry into the History and Prospects of Artificial Intelligence*. A. K. Peters, Ltd., Natick, MA, 2nd edition, 2004. 5
- [39] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013. 27, 28
- [40] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997. 6, 11, 12, 13
- [41] Douglas C. Montgomery, Elizabeth A. Peck, and Geoffrey G. Vining. *Introduction to Linear Regression Analysis*. John Wiley & Sons, Hoboken, NJ, 6th edition, 2021. 8
- [42] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, Cambridge, MA, 2012. 6, 7, 12, 13, 14
- [43] Makoto Nagao. A framework of a mechanical translation between japanese and english by analogy principle. In *Proc. of the international NATO symposium on artificial and human intelligence*, pages 173–180, 1984. 25
- [44] Allen Newell, J. Clifford Shaw, and Herbert A. Simon. Report on a general problem-solving program. In *Proceedings of the International Conference on Information Processing*, pages 256–264, 1959. 3
- [45] Nils J. Nilsson. *The Quest for Artificial Intelligence: A History of Ideas and Achievements*. Cambridge University Press, Cambridge, 2010. 3, 4, 5
- [46] Alexander Novikov, Dmitry Podoprikhin, Anton Osokin, and Dmitry P. Vetrov. Tensorizing neural networks. In *Advances in Neural Information Processing Systems*, volume 28, pages 442–450, 2015. 60, 66
- [47] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proc. of EMNLP*, pages 1532–1543, 2014. 27, 28
- [48] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. OpenAI report, 2018. 25

- [49] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. [14](#), [16](#), [21](#)
- [50] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, London, 3rd edition, 2016. [4](#), [5](#)
- [51] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014. [26](#), [39](#), [41](#)
- [52] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2nd edition, 2018. [9](#)
- [53] Vladimir Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998. [12](#)
- [54] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. [21](#), [23](#), [25](#), [26](#), [29](#), [30](#), [31](#), [35](#), [36](#), [37](#), [38](#), [39](#), [40](#), [42](#), [45](#), [46](#), [47](#), [48](#)
- [55] Yinchong Yang, Denis Krompass, and Volker Tresp. Tensor-train recurrent neural networks for video classification. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 3891–3900, 2017. [63](#)

آپیوست پیوست

جزئیات مدل‌ها و جدول پارامترها

Abstract

In recent years, vision transformers have emerged as one of the core architectures in deep learning models for image processing. However, conventional designs of these transformers often rely on fully connected layers that require flattening high-order input tensors into vectors. This process not only leads to a significant increase in the number of parameters but also results in the loss of spatial structures and inter-dimensional relationships within the data. In this study, we propose a tensor-based framework for the design of a window-based vision transformer that leverages Tensor Compression Layers (TCL) and Tensor Regression Layers (TRL) to effectively model multilinear mappings across data dimensions. By preserving the multidimensional structure of images, the proposed method achieves a substantial reduction in the number of parameters while maintaining structural information and improving the interpretability of the model. Experimental results on standard benchmark datasets demonstrate that incorporating tensorized structures not only reduces computational complexity but also enhances classification accuracy.

Key Words: Vision Transformer, Tensor Decomposition, Tensor Compression Layer (TCL), Tensor Regression Layer (TRL), Multilinear Mapping, Image Classification, Deep Learning.



Vision Transformer

A Thesis Presented for the Degree of
Master in Computer Science

Faculty of Mathematical Sciences

Tarbiat Modares University

by
Seyed Mohammad Badzohreh

Supervisor
Dr. Mansoor Rezghi

2024