به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین پنجم

نام عضو اول	فاطمه رشیدی شهری
شماره دانشجویی	۶۱۰۳۹۹۱۳۱
نام عضو دوم	آراد وزیرپناه
شماره دانشجویی	۶۱۰۳۹۹۱۸۲

فهرست

شکل‌ها

جدول‌ها

قوانین

قبل از پاسخ دادن به پرسش‌ها، موارد زیر را با دقت مطالعه نمایید:

- از پاسخ‌های خود یک گزارش در قالبی که در صفحه‌ی درس در سامانه‌ی Elearn با نام ***REPORTS_TEMPLATE.docx*** قرار داده شده تهیه نمایید.
- پیشنهاد می‌شود تمرین‌ها را در قالب گروه‌های دو نفره انجام دهید. (بیش از دو نفر مجاز نیست و تحویل تک نفره نیز نمره‌ی اضافی ندارد) توجه نمایید الزامی در یکسان ماندن اعضای گروه تا انتهای ترم وجود ندارد. (یعنی، می‌توانید تمرین اول را با شخص A و تمرین دوم را با شخص B و ... انجام دهید)
- **کیفیت گزارش شما در فرآیند تصحیح از اهمیت ویژه‌ای برخوردار است**؛ بنابراین، لطفا تمامی نکات و فرض‌هایی را که در پیاده‌سازی‌ها و محاسبات خود در نظر می‌گیرید در گزارش ذکر کنید.
- در گزارش خود مطابق با آنچه در قالب نمونه قرار داده شده، برای شکل‌ها زیرنویس و برای جدول‌ها بالانویس در نظر بگیرید.
- الزامی به ارائه توضیح جزئیات کد در گزارش نیست، اما باید نتایج بدست آمده از آن را گزارش و تحلیل کنید.
- **تحلیل نتایج الزامی می‌باشد، حتی اگر در صورت پرسش اشاره‌ای به آن نشده باشد.**
- **دستیاران آموزشی ملزم به اجرا کردن کدهای شما نیستند**؛ بنابراین، هرگونه نتیجه و یا تحلیلی که در صورت پرسش از شما خواسته شده را به طور واضح و کامل در گزارش بیاورید. در صورت عدم رعایت این مورد، بدیهی است که از نمره تمرین کسر می‌شود.
- **کدها حتما باید در قالب نوت‌بوک با پسوند .ipynb تهیه شوند، در پایان کار، تمامی کد اجرا شود و خروجی هر سلول حتما در این فایل ارسالی شما ذخیره شده باشد.** بنابراین برای مثال اگر خروجی سلولی یک نمودار است که در گزارش آورده‌اید، این نمودار باید هم در گزارش هم در نوت‌بوک کدها وجود داشته باشد.
- **در صورت مشاهده‌ی تقلب امتیاز تمامی افراد شرکت‌کننده در آن، 100- لحاظ می‌شود.**
- تنها زبان برنامه نویسی مجاز **Python** است.
- **استفاده از کدهای آماده برای تمرین‌ها به هیچ وجه مجاز نیست. در صورتی که دو گروه از یک منبع مشترک استفاده کنند و کدهای مشابه تحویل دهند، تقلب محسوب می‌شود.**

- نحوه محاسبه تاخیر به این شکل است: پس از پایان رسیدن مهلت ارسال گزارش، حداکثر تا یک هفته امکان ارسال با تاخیر وجود دارد، پس از این یک هفته نمره آن تکلیف برای شما صفر خواهد شد.
  - سه روز اول: بدون جریمه
  - روز چهارم: ۵ درصد
  - روز پنجم: ۱۰ درصد
  - روز ششم: ۱۵ درصد
  - روز هفتم: ۲۰ درصد
- حداکثر نمره‌ای که برای هر سوال می‌توان اخد کرد ۱۰۰ بوده و اگر مجموع بارم یک **سوال** بیشتر از ۱۰۰ باشد، در صورت اخد نمره بیشتر از ۱۰۰، اعمال نخواهد شد.
  - برای مثال: اگر نمره اخذ شده از سوال ۱ برابر ۱۰۵ و نمره سوال ۲ برابر ۹۵ باشد، نمره نهایی تمرین ۹۷.۵ خواهد بود و نه ۱۰۰.
- لطفا گزارش، کدها و سایر ضمایم را به در یک پوشه با نام زیر قرار داده و آن را فشرده سازید، سپس در سامانه‌ی Elearn بارگذاری نمایید:

HW[Number] _[Lastname]_[StudentNumber]_[Lastname]_[StudentNumber].zip

(مثال: HW1_Ahmadi_810199101_Bagheri_810199102.zip)

- برای گروه‌های دو نفره، بارگذاری تمرین از جانب یکی از اعضا کافی است ولی پیشنهاد می‌شود هر دو نفر بارگذاری نمایند.

پرسش ۱. تشخیص اخبار جعلی مبتنی بر مدل‌های ترنسفورمر

۱-۱. آشنایی با BERT و CT-BERT

۱-۱-۱. پرسش اول

تکنیک یادگیری انتقالی یا Transfer Learning یکی از پرکاربردترین روش‌های Machine Learning است که به ویژه در مدل‌های زبان طبیعی مانند BERT استفاده می‌شود. Transfer Learning به طور کلی به معنای استفاده از دانش (وزن‌های یک مدل در واقع) مدل‌های از پیش آموزش دیده بر روی مجموعه داده بسیار بزرگ‌تر، جامع‌تر و عمومی است که حالا از آن برای انجام وظایف کوچک‌تر و خاص‌تر (با داده‌های محدود) است.

برای استفاده از این روش، دو مرحله اصلی وجود دارد:

1. Pre-training:

مدل بر روی یک مجموعه داده بزرگ و عمومی (مانند کتاب‌ها، مقالات، و وب‌سایت‌ها) آموزش داده می‌شود تا ویژگی‌ها و الگوهای زبان را به‌طور عمومی یاد بگیرد. در این مرحله، مدل تلاش می‌کند تا نمایه‌های معنایی و نحوی کلمات و جملات را یاد بگیرد.

در مورد BERT، این مرحله شامل دو وظیفه اصلی است: مدل ماسک شده (Masked Language Model) و پیش‌بینی جمله بعدی (Next Sentence Prediction).

2. Fine-tuning یا Feature-based:

مدل پیش‌آموزش دیده شده بر روی مجموعه داده‌های خاص‌تر و کوچک‌تر با برچسب‌های (label) خاص تنظیم می‌شود. این مرحله معمولاً نیاز به زمان و داده‌های کمتری دارد چرا که مدل قبلاً ویژگی‌های عمومی زبان را یاد گرفته است. و یا اینکه از این مدل به عنوان Feature extractor استفاده می‌شود و سپس مدل ساده‌تری که بعد از آن می‌آید را آموزش می‌دهیم.

در این مرحله، مدل به وظایف خاصی مانند طبقه‌بندی متون، تشخیص موجودیت‌های نام‌دار (NER)، ترجمه ماشینی، یا تحلیل احساسات تنظیم می‌شود.

حال مزایا این روش را بررسی می‌کنیم:

- صرفه جویی در منابع محاسباتی و داده: با استفاده از مدل‌های پیش‌آموزش دیده، نیازی به آموزش مدل از ابتدا بر روی مجموعه داده‌های بزرگ نیست و می‌توان با استفاده از داده‌های محدودتر و زمان کمتر به نتایج قابل قبول دست یافت.
- بهبود عملکرد مدل: مدل‌هایی که از یادگیری انتقالی استفاده می‌کنند، به‌طور کلی عملکرد بهتری نسبت به مدل‌هایی دارند که از ابتدا آموزش داده می‌شوند، زیرا ویژگی‌های عمومی زبان را بهتر یاد گرفته‌اند.

استفاده از این روش زمانی مفید است که:

- داده‌های برچسب‌گذاری شده کمیاب باشد: اگر برای وظیفه خاصی داده‌های برچسب‌گذاری شده کافی وجود نداشته باشد، استفاده از مدل‌های پیش‌آموزش دیده به‌طور چشمگیری کمک می‌کند.
- زمان و منابع محاسباتی محدود باشد: آموزش یک مدل از ابتدا نیاز به منابع زیادی دارد، در حالی که استفاده از مدل‌های پیش‌آموزش دیده زمان و منابع کمتری نیاز دارد.
- وظایف متنوع و پیچیده‌ای در زبان طبیعی مد نظر باشد: مدل‌های پیش‌آموزش دیده مانند BERT توانایی یادگیری ویژگی‌های عمومی زبان را دارند و می‌توانند برای وظایف مختلف زبان طبیعی با تنظیم دقیق مورد استفاده قرار گیرند. (در مسئله ما نیز به این دلیل از یادگیری انتقالی استفاده شده)

۱-۱-۲. پرسش دوم

- رویکرد Feature-based:

در این روش مدل از پیش آموزش دیده به عنوان استخراج کننده ویژگی (Feature Extractor) استفاده می‌شود. در واقع مدل از پیش آموزش دیده مانند BERT، بر روی داده‌های ورودی اعمال می‌شود تا بردار ویژگی یا در واقع hidden representationها از داده‌های استخراج شوند. سپس این ویژگی‌های بدست آمده را به یک مدل ساده‌تر داده می‌شود تا برای وظیفه خاص مدل آموزش ببیند و از آن استفاده کند. این روش معمولا زمانی استفاده می‌شود که یا داده‌های موجود محدود باشند و یا منابع محاسباتی کافی برای استفاده از روش Feature-based موجود نباشد.

- رویکرد Fine-tuning:

در این رویکرد، کل مدل از پیش آموزش دیده با وظیفه خاصی که در نظر داریم، تنظیم می‌شود. در واقع کل مدل (یا ممکن است تصمیم بگیریم تنها بخشی از آن را دوباره آموزش دهیم) بر روی مجموعه داده مورد نظر آموزش می‌بیند تا پارامترهای مدل برای این وظیفه بهینه شوند. این روش زمانی استفاده می‌شود که داده‌های با label کافی موجود داشته باشیم و همچنین منابع محاسباتی کافی برای انجام این کار داشته باشیم.

مقایسه این دو روش:

- انعطاف پذیری:

Feature-based: انعطاف‌پذیری کمتری دارد چون مدل اصلی فقط به عنوان استخراج‌کننده ویژگی‌ها استفاده می‌شود و به‌طور کامل برای وظیفه جدید تنظیم نمی‌شود.

Fine-tuning: انعطاف‌پذیری بیشتری دارد چون مدل به‌طور کامل برای وظیفه جدید تنظیم می‌شود و می‌تواند بهبودهای بیشتری در عملکرد داشته باشد.

- پیچیدگی محاسباتی:

Feature-based: ساده‌تر و سریع‌تر است چون فقط استخراج ویژگی‌ها انجام می‌شود و مدل اصلی تغییری نمی‌کند.

Fine-tuning: پیچیده‌تر و زمان‌برتر است چون نیاز به تنظیم دقیق کل مدل دارد.

- نیاز به داده:

Feature-based: می‌تواند با داده‌های کمتر هم کار کند چون نیاز به آموزش مدل از ابتدا یا تنظیم دقیق کل مدل ندارد.

Fine-tuning: معمولاً به داده‌های برچسب‌گذاری شده بیشتری نیاز دارد تا مدل بتواند به خوبی تنظیم شود.

- کارایی:

Feature-based: ممکن است در برخی وظایف کارایی خوبی داشته باشد، اما معمولاً به اندازه Fine-tuningعملکرد خوبی ندارد.

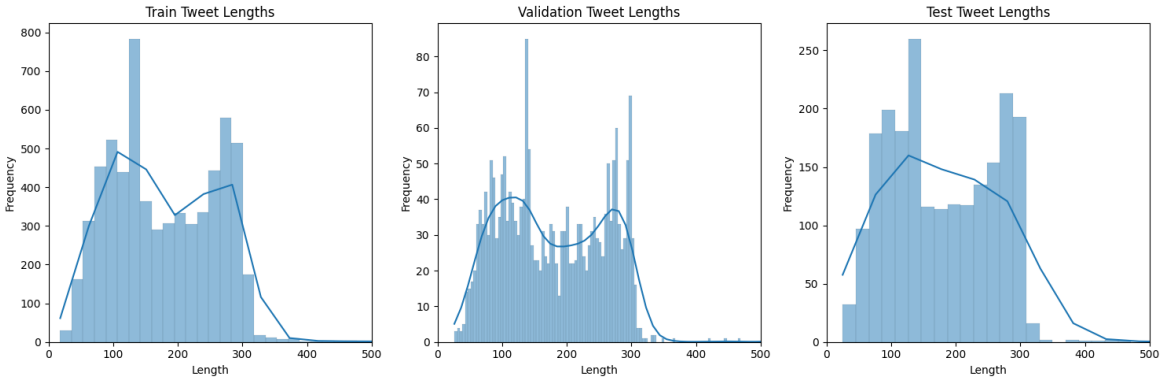
Fine-tuning: معمولاً عملکرد بهتری دارد چون مدل به‌طور کامل برای وظیفه خاص تنظیم شده است.

۱-۲. دادگان

این مجموعه دادگان شامل توئیت‌های راجع به بیماری کرونا است که labelهای آن بیان می‌کند که این خبر در غالب توئیت، یک خبر درست و صادق بوده (real) و یا اینکه یک خبر دروغ و جعلی است (fake). هدف ما نیز ساخت و آموزش مدلی است که بتواند تصمیم بگیرد که آیا یک توئیت در مورد کرونا، جزو اخبار درست بوده و یا اینکه یک خبر جعلی است.

در این مجموعه دادگان، ۶۴۲۰ توئیت (حدود ۶۰ درصد کل) مربوط به دادگان آموش است. همچنین برای هر کدام از دادگان تست و اعتبارسنجی، حدود ۲۰ درصد دادگان کل، یعنی چیزی حدود ۲۱۴۰ توئیت اختصاص داده شده است.

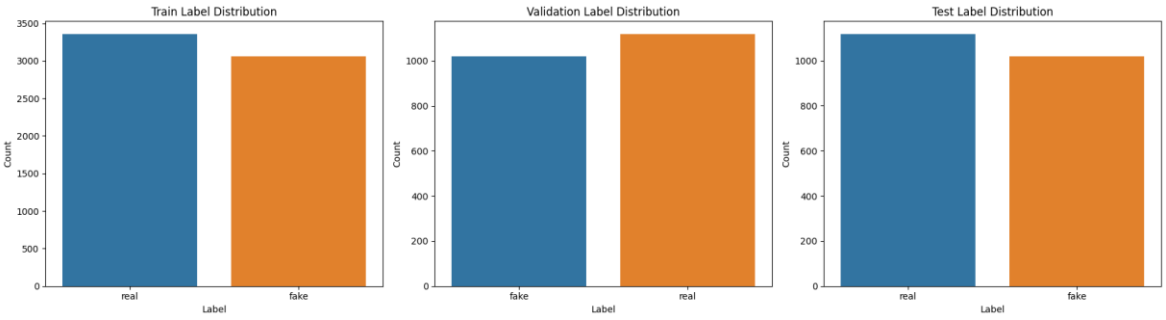
در نمودار زیر می‌توانیم توزیع طول توئیت‌ها در مجموعه‌های آموزش، تست و اعتبارسنجی را ببینیم.



شکل 1. توزیع طول توئیت‌ها

همانطور که مشاهده می‌کنیم، توزیع طول توئیت‌ها در مجموعه دادگان مختلف، تقریبا شبیه به هم بوده و یکسان است. بعد از انجام پیش پردازش (preprocessing)، به گونه‌ای عمل می‌کنیم که طول توئیت‌ها (در واقع وکتور و آرایه متناظر با آن‌ها) طول برابر با ۱۲۸ را داشته باشد. به صورتی که توئیت‌های بلندتر از ۱۲۸ کلمه، تنها ۱۲۸ کلمه اول و برای توئیت‌های کوتاه‌تر از آن، padding انجام می‌شود تا آن‌ها نیز طول ۱۲۸ داشته باشند. این کار به این دلیل انجام می‌شود تا بتوانیم مدل را آموزش دهیم و از آن استفاده کنیم. اگر طول‌های متفاوت باشند، نمی‌توان به این سادگی مدل را آموزش داد و از آن استفاده کرد.

در نمودار زیر توزیع برچسب‌های مختلف در هر مجموعه را مشاهده می‌کنیم.



شکل 2. توزیع برچسب‌ها

همانطور که مشاهده می‌کنیم توزیع labelهای real و fake در همه مجموعه‌های تقریبا یکسان و ثابت بوده. بنابراین نیازی به balance کردن این مجموعه داده نداریم.

۱-۲-۱. پیش پردازش دادگان

برای پیش پردازش دادگان، از روش‌های زیر استفاده می‌کنیم:

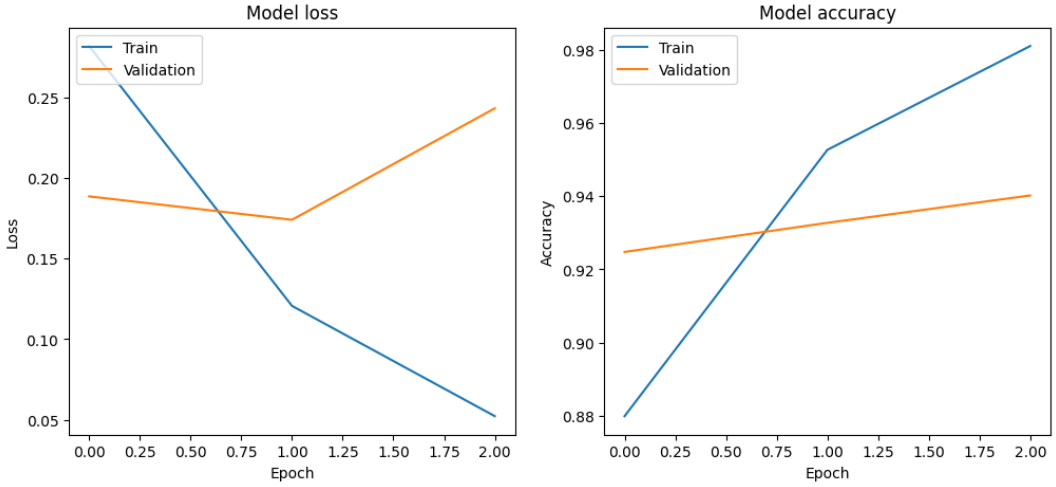
1. ابتدا HTML و URL های موجود در متون توئیت‌ها را حذف می‌کنیم. این کار به این دلیل است که این لینک‌ها هیچ فایده‌ای در تشخیص صحیح و یا غلط بودن خبر ندارد و حتی ممکن است باعث آموزش اشتباه مدل شود.
2. سپس emojiهای موجود در متن را با معنای متناظر آن‌ها در زبان انگلیسی جا به جا می‌کنیم. این کار به این دلیل انجام می‌شود که emojiها بیانگر احساسات هستند و ممکن است ما را در تصمیم بهتر نسبت به درست یا غلط بودن خبر، کمک کنند.
3. سپس نقطه‌گذاری‌ها را حذف می‌کنیم.
4. حال stop wordها را حذف می‌کنیم. چرا که این کلمات هیچ بار معنایی به جمله اضافه نمی‌کنند و ممکن است باعث اشتباه در آموزش مدل شوند.
5. حال ریشه کلمات را جایگزین آن‌ها می‌کنیم، در این صورت تصمیم گیری راجع به یک جمله راحت‌تر خواهد شد.
6. در نهایت از توکنایزر مدل‌ها (BERT و CT-BERT با توجه به مدل استفاده شده در تسک) توئیت‌های از توکن شده input_ids، attention_mask و token_type_ids را استخراج می‌کنیم تا در آموزش مدل از آن‌ها به جای خود توکن‌ها استفاده کنیم. در این مرحله طول آرایه‌ها نیز ۱۲۸ در نظر گرفته می‌شود.

۱-۳. پیاده سازی مدل با رویکرد Fine-tuning

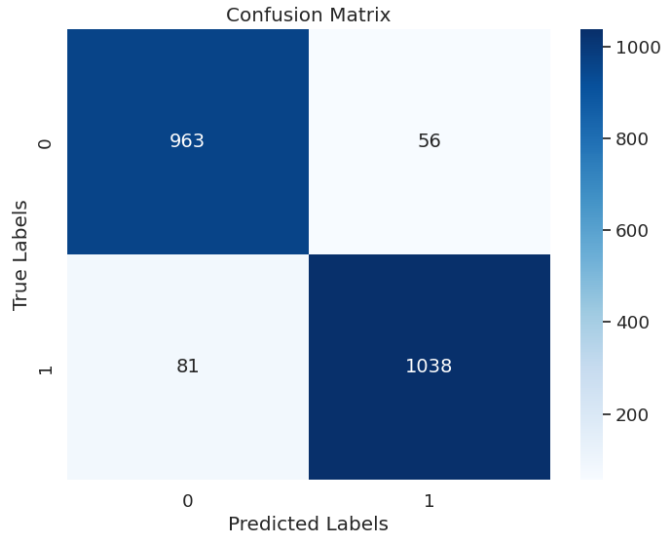
در این بخش از رویکرد Fine-tuning استفاده می‌کنیم. یعنی اینکه کل مدل را آموزش می‌دهیم. لازم به ذکر است که همه مدل‌ها با پارامترهای گفته شده در مقاله اجرا شده‌اند.

۱-۳-۱. مدل اول

در این مدل از BERT استفاده کرده و سپس خروجی لایه آخر آن را به یک لایه Dense می‌دهیم که یک نورون خروجی دارد. همچنین به دلیل وجود یک نورون در لایه انتهایی (۰ و ۱ بودن داده‌های target)، از تابع فعال‌ساز sigmoid استفاده می‌کنیم. نتایج بدست آمده در زمان آموزش مدل را می‌توانیم در زیر ببینیم. همچنین نتیجه دقت و F1-score آن روی داده‌های تست به ترتیب برابر 93.6 و 93.8 است.



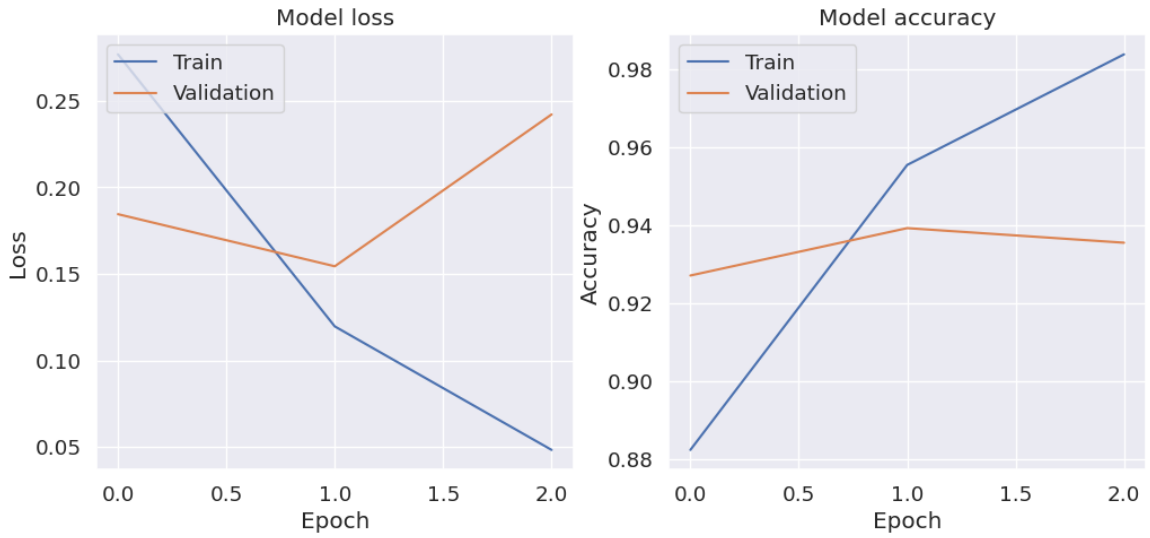
شکل 3. روند آموزش مدل اول با رویکرد **fine-tuning**



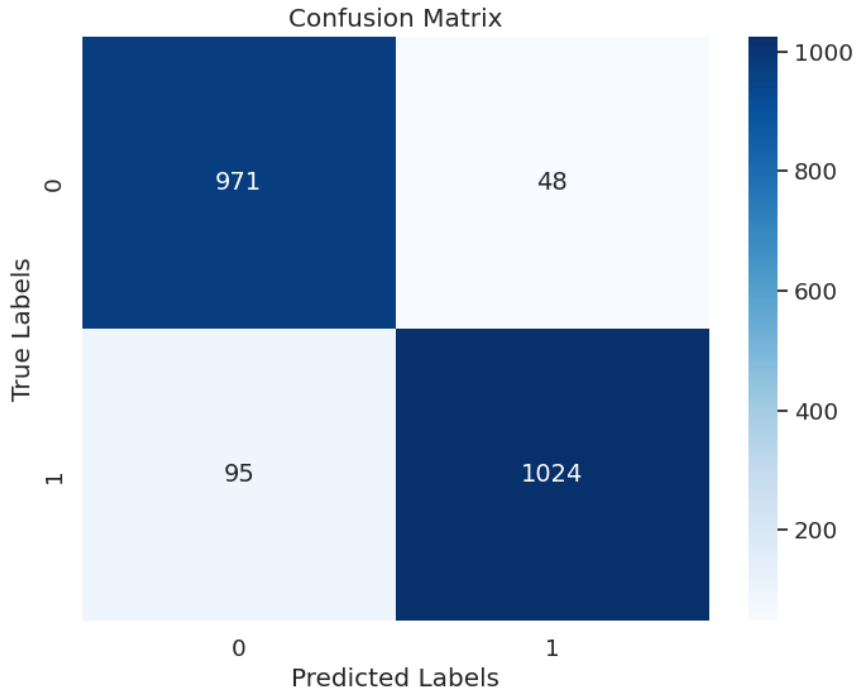
شکل 4. ماتریس آشفتگی داده‌های تست مدل سوم با رویکرد **fine-tuning**

۱-۳-۲. مدل دوم

در این مدل دوباره از BERT استفاده کرده ولی بعد از آن از یک BiGRU استفاده می‌کنیم که ورودی آن آخرین لایه ترنسفورمری و hidden state مدل BERT است. و سپس از یک لایه Dense مانند قبل استفاده می‌کنیم تا بتوانیم ورودی‌ها را classify کنیم. نتایج بدست آمده در زمان آموزش مدل را می‌توانیم در زیر ببینیم. همچنین نتیجه دقت و F1-score آن روی داده‌های تست به ترتیب برابر 93.3 و 93.4 است.



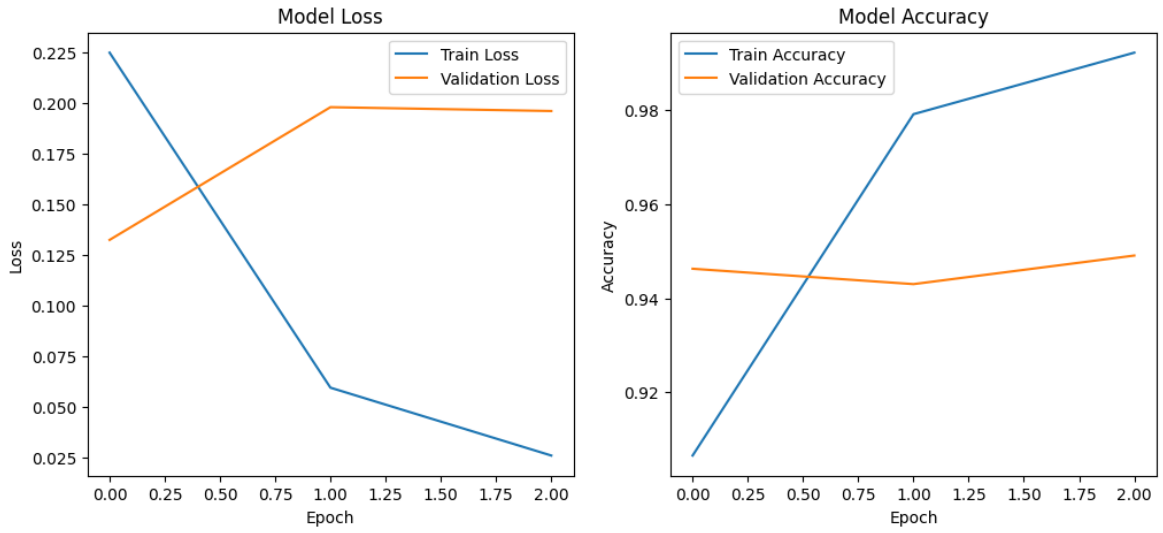
شکل 5. روند آموزش مدل دوم با رویکرد **fine-tuning**



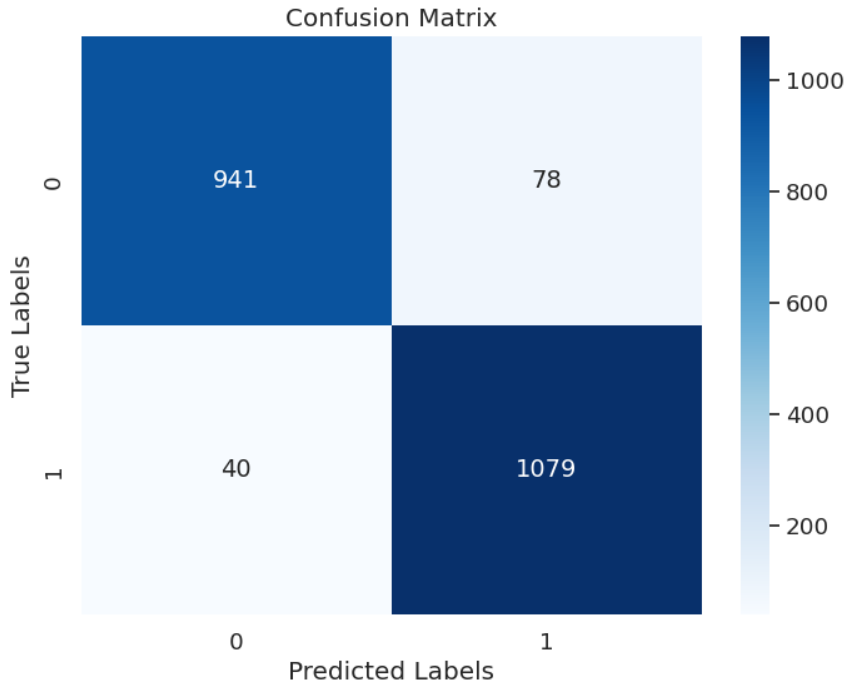
شکل 6. ماتریس آشفتگی داده‌های تست مدل سوم با رویکرد **fine-tuning**

۱-۳-۳. مدل سوم

در این مدل مانند مدل دوم عمل می‌کنیم، صرفا به جای استفاده از BERT، از CT-BERT استفاده می‌کنیم. این مدل به طور اختصاصی روی داده‌های جامع‌تری مربوط به Covid آموزش دیده و انتظار داریم که با استفاده از این مدل، عملکرد بهتری را شاهد باشیم. نتایج بدست آمده در زمان آموزش مدل را می‌توانیم در زیر ببینیم. همچنین نتیجه دقت و F1-score آن روی داده‌های تست به ترتیب برابر 94.5 و 94.8 است.



شکل 7. روند آموزش مدل سوم با رویکرد **fine-tuning**



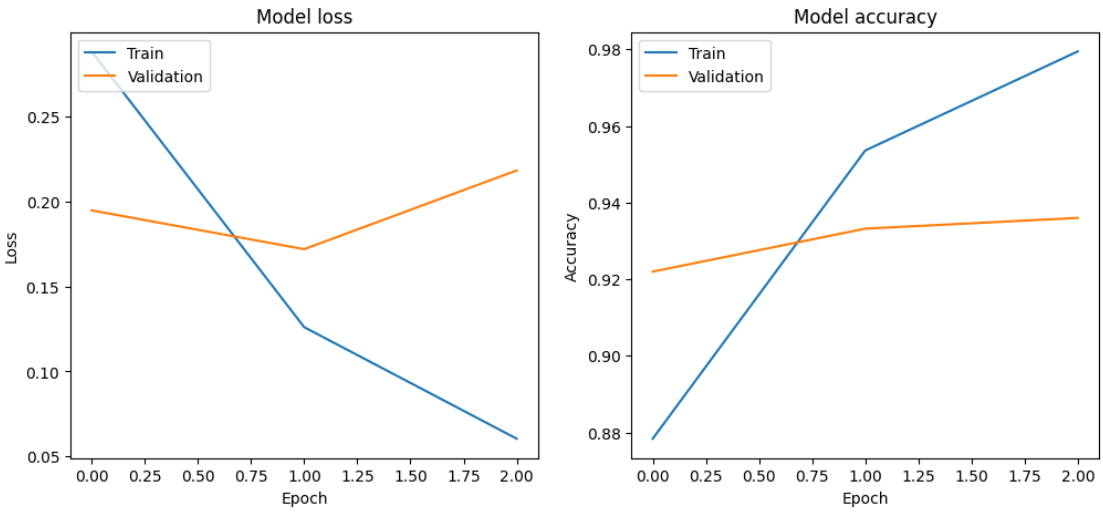
شکل 8. ماتریس آشفتگی داده‌های تست مدل سوم با رویکرد **fine-tuning**

۱-۴. پیاده سازی مدل با رویکرد Feature-based

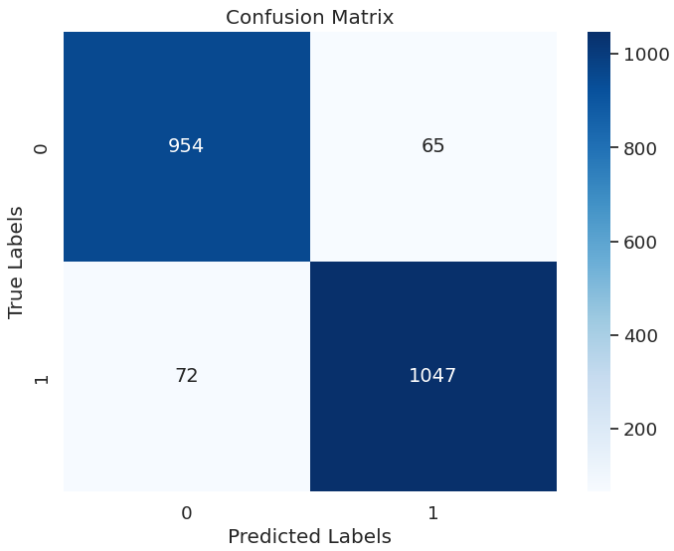
در این رویکرد، ما کل مدل را آموزش نمی‌دهیم و BERT یا CT-BERT مورد استفاده را Freeze می‌کنیم تا آموزش ندیده و وزن‌های آن تغییری نکنند. لازم به ذکر است که همه مدل‌ها با پارامترهای گفته شده در مقاله اجرا شده‌اند.

۱-۴-۱. مدل اول

این مدل مانند مدل اول در بخش fine-tuning تعریف شده، تنها بخش BERT آن را Nontrainable کردیم. نتایج بدست آمده در زمان آموزش مدل را می‌توانیم در زیر ببینیم. همچنین نتیجه دقت و F1-score آن روی داده‌های تست به ترتیب برابر 93.5 و 93.8 است.



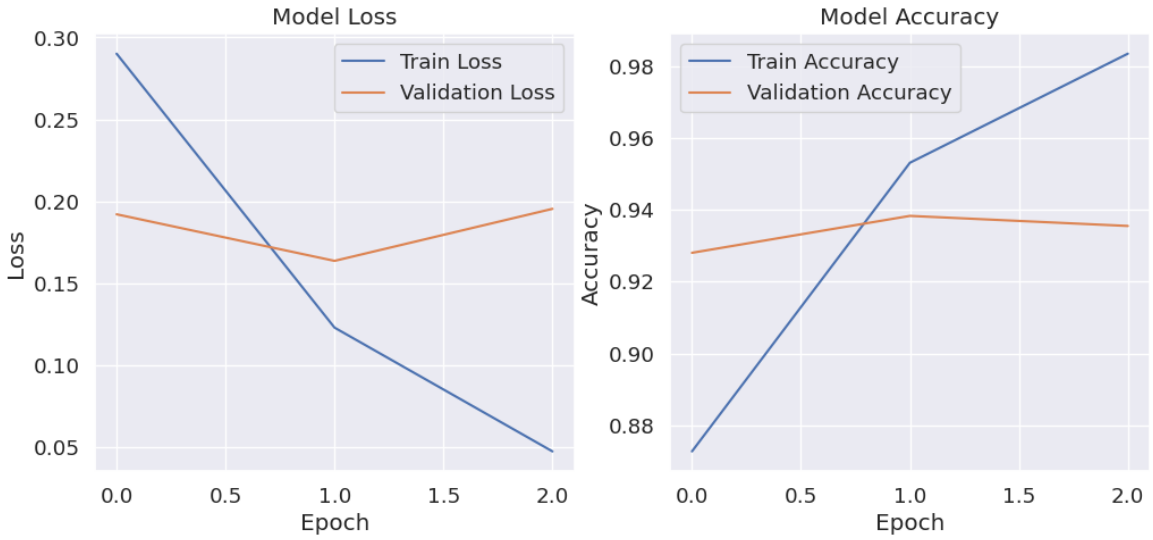
شکل 9. روند آموزش مدل اول با رویکرد **feature-based**



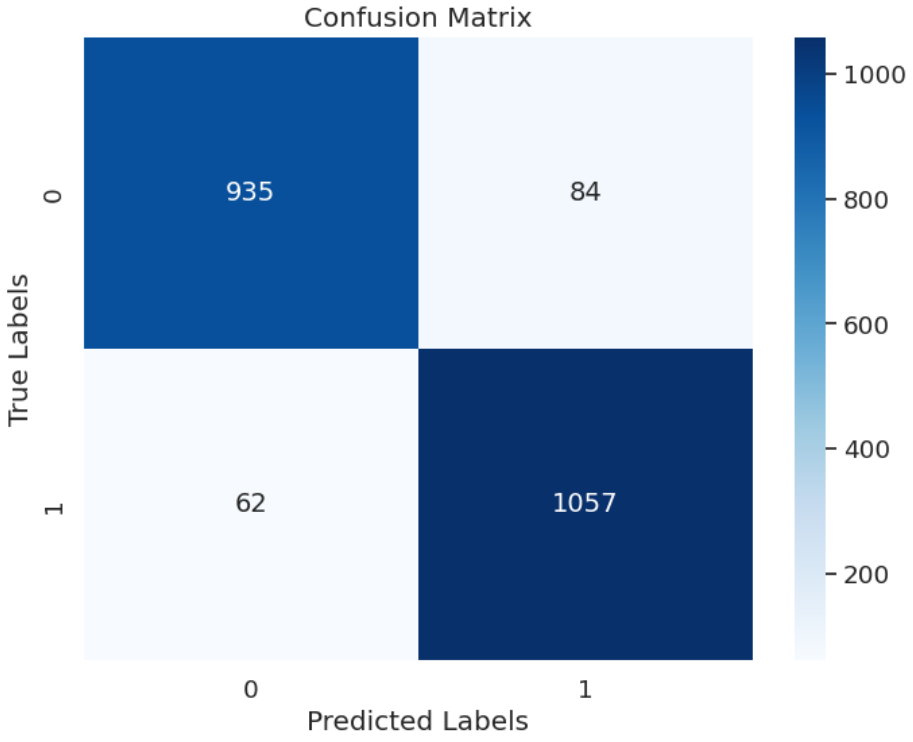
شکل 10. ماتریس آشفتگی داده‌های تست مدل اول با رویکرد **fine-tuning**

۱-۴-۲. مدل دوم

این مدل مانند مدل دوم در بخش fine-tuning تعریف شده، تنها بخش BERT آن را Nontrainable کردیم. نتایج بدست آمده در زمان آموزش مدل را می‌توانیم در زیر ببینیم. همچنین نتیجه دقت و F1-score آن روی داده‌های تست به ترتیب برابر 93.1 و 93.5 است.



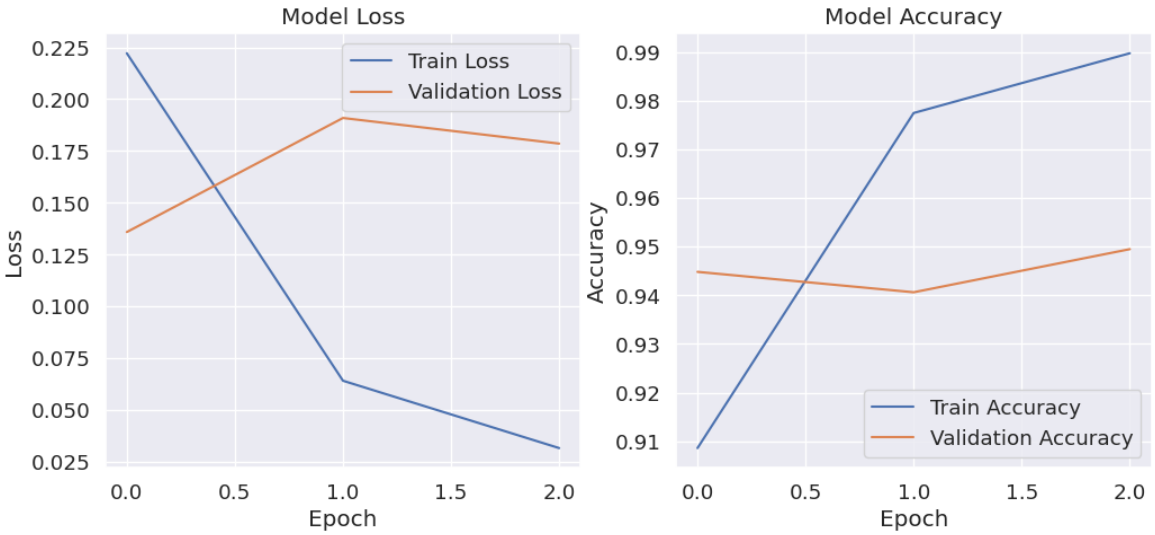
شکل 11. روند آموزش مدل دوم با رویکرد **feature-based**



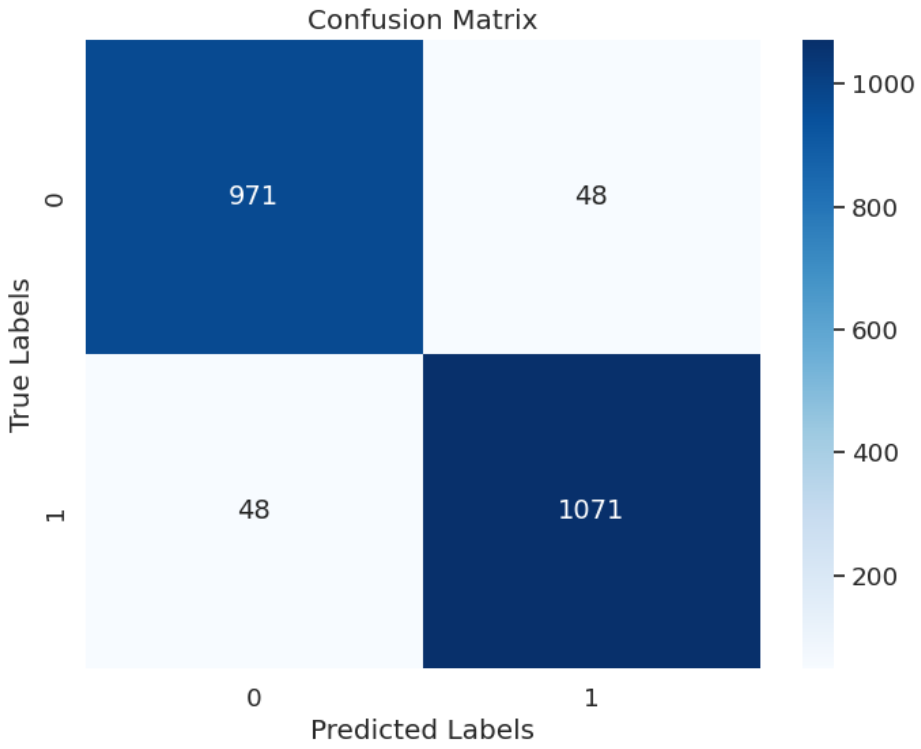
شکل 12. ماتریس آشفتگی داده‌های تست مدل دوم با رویکرد **fine-tuning**

۱-۴-۳. مدل سوم

این مدل مانند مدل سوم در بخش fine-tuning تعریف شده، تنها بخش CT-BERT آن را Nontrainable کردیم. نتایج بدست آمده در زمان آموزش مدل را می‌توانیم در زیر ببینیم. همچنین نتیجه دقت و F1-score آن روی داده‌های تست به ترتیب برابر 95.5 و 95.7 است.



شکل 13. روند آموزش مدل سوم با رویکرد **feature-based**



شکل 14. ماتریس آشفتگی داده‌های تست مدل سوم با رویکرد **fine-tuning**

۱-۵. تحلیل نتایج

می‌توانیم خلاصه نتایج بدست آمده از آموزش مدل‌های مختلف با رویکردهای مختلف را در جدول‌های ۱ و ۲ مشاهده کنیم. لازم به ذکر است که همه مدل‌ها با پارامترهای گفته شده در مقاله اجرا شده‌اند.

	BERT + Dense	BERT + BiGRU	CT-BERT + BiGRU
Accuracy	93.6	93.3	94.5
F1-score	93.8	93.4	94.5

جدول 1. نتایج مدل‌های رویکرد **Fine-tuning**

	BERT + Dense	BERT + BiGRU	CT-BERT + BiGRU
Accuracy	93.6	93.2	95.5
F1-score	93.8	93.5	95.7

جدول 2. نتایج مدل‌های رویکرد **Feature-based**

۱-۵-۱. پرسش اول

با توجه به نتایج بدست آمده، به طور مشخص CT-BERT همواره از مدل متناظر با BERT با رویکردهای مختلف نتایج بهتری داشته. حتی در رویکرد Feature-based تفاوت بیشتر نیز هست. دلیل این اتفاق مخصوصا در روش Feature-based، به طور مشخص پیش آموزش مدل CT-BERT بر روی داده‌های مربوط به کووید است، که به همین علت، CT-BERT به عنوان یک Feature extractor که روی دیتاست جامع‌تری آموزش دیده، عملکرد خیلی بهتری دارد. بنابراین این مدل هم نقش Feature extractor را بهتر بازی می‌کند و هم برای آموزش، وزن‌های اولیه بهتر تنظیم شده و آموزش آن راحت‌تر خواهد بود.

۱-۵-۲. پرسش دوم

در مدل اول بعد از BERT، تنها یک لایه Dense برای تسک classification استفاده می‌شود. اما در مدل دوم، ابتدا از یک BiGRU استفاده می‌کنیم. مشاهده می‌کنیم که مدل دوم همواره عملکرد بدتری داشته و این مدل ساده اول بوده که علمکرد بهتری نشان داده. احتمالا این اتفاق دلیل بر پیچیده‌تر بودن مدلی که از BiGRU استفاده کرده دارد. مدل پیچیده‌تر نیازمند داده بیشتر و آموزش بیشتر است. بنابراین با آموزش یکسان دو مدل، طبیعتا مدل پیچیده‌تر عملکرد بهتری نخواهد داشت. احتمالا اگر این مدل را بیشتر آموزش دهیم. عملکرد بهتری نشان دهد.

۱-۵-۳. پرسش سوم

برای پاسخ به این سوال، باید مدل‌های BERT و CT-BERT را جداگانه تحلیل کنیم.

- BERT: در مدل‌های که از BERT به عنوان backbone با وزن‌های از پیش‌آموزش دیده استفاده کردیم، نتایج تقریبا یکسان بوده و تفاوت آن‌چنانی در آن‌های دیده نمی‌شود. این اتفاق دلیل بر این دارد که این شبکه بر روی متون بسیار جامع‌تری آموزش دیده و خب احتمالا برای مدلی که ما تعریف کردیم و با تعداد کمی از ایپاک برای آموزش استفاده کردیم، وزن‌های این قسمت از مدل (BERT) با در نظر گرفتن وزن‌های آموزش دیده روی قسمت‌های بعدی مدل، تفاوت آنچنانی در نتیجه ایجاد نمی‌کند. اما اگر تعداد ایپاک‌ها برای آموزش را بیشتر کنیم، احتمالا روش fine-tuning عملکرد بهتری خواهد داشت، چرا که برای تسک ما آموزش خواهد دید و خاص این مسئله با این داده‌ها می‌شود.
- CT-BERT: در مدل‌هایی که از این مدل به عنوان backbone استفاده کردند، نتایج در کل بهتر بود که دلیل آن مشخص است و قبلا به آن اشاره شده. اما در آموزش مدلی که از رویکرد Feature-based استفاده کردیم، نتایج خیلی بهتر بود چرا که CT-BERT به صورت جامع‌تری به مسائل مربوط به کویید نگاه می‌کند، در حالی در رویکرد fine-tuning، خیلی خاص‌تر و بسته به مسئله می‌شود. اما مانند مدل BERT، احتمالا اگر فرصت بیش‌تری برای آموزش به مدل‌هایی که از رویکرد fine-tuning استفاده می‌کنند، داده شود، روی دادگان خاص نتیجه بهتری داشته باشند (احتمال overfit شدن مدل نیز هست). اما خب با آموزش کم، CT-BERT به عنوان feature-extractor عملکرد بهتری خواهد داشت.

۱-۵-۴. پرسش چهارم

- مدل اول:
  - توئیتی که خبر درستی است، اما به اشتباه، خبر جعلی پیش‌بینی شده:

“If you have had a flu shot in the last 3-5 years you will probably test positive” for COVID-19.

احتمالا به خاطر کلماتی مانند probably یا استفاده از گزاره شرطی، اشتباه پیش‌بینی شده که خبر جعلی است.

- توئیتی که خبر جعلی است، اما به اشتباه خبر درست پیش‌بینی شده:

A photo shows a 19-year-old vaccine for canine coronavirus that could be used to prevent the new coronavirus causing COVID-19.

جمله گفته شده خیلی به حالت رسمی نزدیک است و گویی که با دلیل راجع به چیزی صحبت می‌کند، احتمالا دلیل اشتباه مدل در این است.

- مدل دوم:
  - توئیتی که خبر درستی است، اما به اشتباه، خبر جعلی پیش‌بینی شده:

#CoronaVirusUpdates: #COVID19 testing status update: @ICMRDELHI stated that 60565728 samples tested up to September 16 2020 1136613 sample tested on September 16 2020 #StaySafe #IndiaWillWin https://t.co/oUcwFRtz6T

احتمالا به خاطر پیش پردازش یک سری از اطلاعات حذف شده و باقی‌مانده آن‌ها خبر درستی را بیان نمی‌کرده.

  - توئیتی که خبر جعلی است، اما به اشتباه خبر درست پیش‌بینی شده:

President Uhuru Kenyatta of Kenya ordered credit reference bureaus to delist Kenyans who had defaulted on loans to protect Kenyans from the economic effects of COVID-19.

به نظرم با وجود اطلاعاتی که مدل دارد (نه اطلاعات آنلاین و درک کامل از اتفاقات) منطقی است که این توئیت یک خبر درست پیش‌بینی شود.

- مدل سوم:
  - توئیتی که خبر درستی است، اما به اشتباه، خبر جعلی پیش‌بینی شده:

Israel has no deaths from COVID-19; tea made of lemon and bicarbonate can cure coronavirus.

احتمالا به خاطر درمان خاصی که برای کویید پیشنهاد داده، جعلی پیش‌بینی شده.

  - توئیتی که خبر جعلی است، اما به اشتباه خبر درست پیش‌بینی شده:

There are four patients with COVID-19 in Middlemore Hospital. Two are stable and each of these is in isolation on a ward. Two are in ICU and are in critical condition. These are the same four patients previously reported and are all part of the community cluster.

خبر خیلی دقیق و گزارش طور بیان شده و به همین علت جزو اخبار درست دسته‌بندی شده.

پرسش ۲ – بکارگیری مدل‌های ترنسفورمری در طبقه بندی تصاویر

۲-۱. آشنایی با ترنسفومرهای تصویر

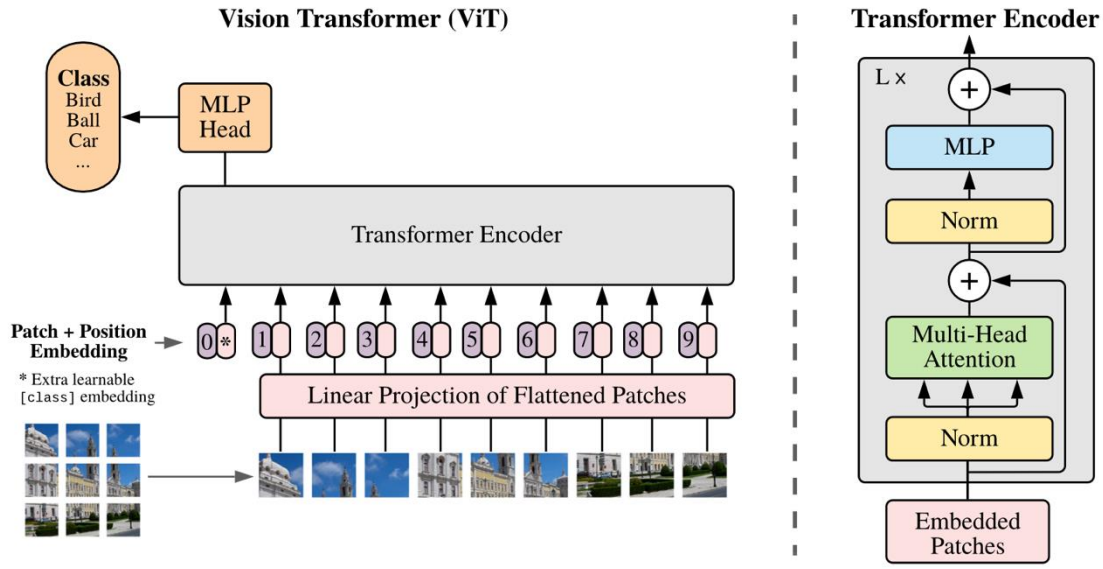
الف) ساختار و نحوه‌ی کارکرد ViT:

همانطور که می‌دانیم، ترنسفورمرهای اولین بار در سال ۲۰۱۷ و بر پایه‌ی attention برای داده‌هایی که به صورت sequential هستند، معرفی شدند. در ترنسفورمرهای تصویر، از بخش encoderموجود در مدل اصلی ترنسفورمرها استفاده می‌شود طوری که در آن هر تصویر به یک بردار تبدیل می‌شود و به عنوان ورودی به encoder داده می‌شود. ساختار ViTها شامل موارد زیر است:

- Patch embedding: ابتدا تمام تصاویر به قطعات کوچکتر تقسیم می‌شوند (مثلا ۱۶ × ۱۶) و طی مراحلی، هر عکس به یک بردار تبدیل می‌شود طوری که اگر فرض کنیم ابعاد هر تصویر به صورت $[P, P, C]$ باشد، آنگاه بردار متناظر با آن دارای بعد $[P \times P \times C, 1]$ خواهد بود.
- Position embedding: از آنجا که برخلاف متون، در تصاویر spatial information نیز مطرح است، پس نیاز است embedding متناظر با هر patch، اطلاعاتی در مورد مکان آن patch در عکس اصلی را شامل شود که این اطلاعات در حین یادگیری، پوسط مدل بدست می‌آیند.
- Transformation encoder: این بخش کاملا مشابه encoder در ترنسفورمرهای متناظر با متون است. بنابراین، شامل تعدادی multi-head self-attention و شبکه‌های feed-forward است.
- Classification head: خروجی آخرین لایه‌ی encoder شامل تعدادی بردار است که به تعداد $P \times P + 1$ هستند. همانطور که در معماری شبکه‌ی متناظر با ViT گفته شده، قبل از ورودی embeddingها به بخش encoder یک CLS token با بعد هم اندازه‌ی patch embeddingها در نظر گرفته می‌شود و در لایه‌ی خروجی، بردار متناظر با این توکن، مشخص کننده‌ی کلاس تصویر ورودی خواهد بود.

در مورد نحوه‌ی کارکرد این مدل‌ها نیز می‌توان گفت که ابتدا تصاویر را پیش‌پردازش کرده و به قطعات کوچکتر تقسیم می‌کنیم. هر تصویر را به یک بردار تبدیل کرده و برای هر تصویر نیز یک positional embedding نیز به دست می‌آوریم و با آنها جمع می‌کنیم. سپس، این embeddingها به encoder وارد میشوند و لایه‌های self-attention, feed-forward و normalization روی آنها اعمال میشود. نهایتا با اعمال یک فعال‌ساز روی embedding متناظر با CLS token، کلاس عکس ورودی را پیش‌بینی می‌کنیم.

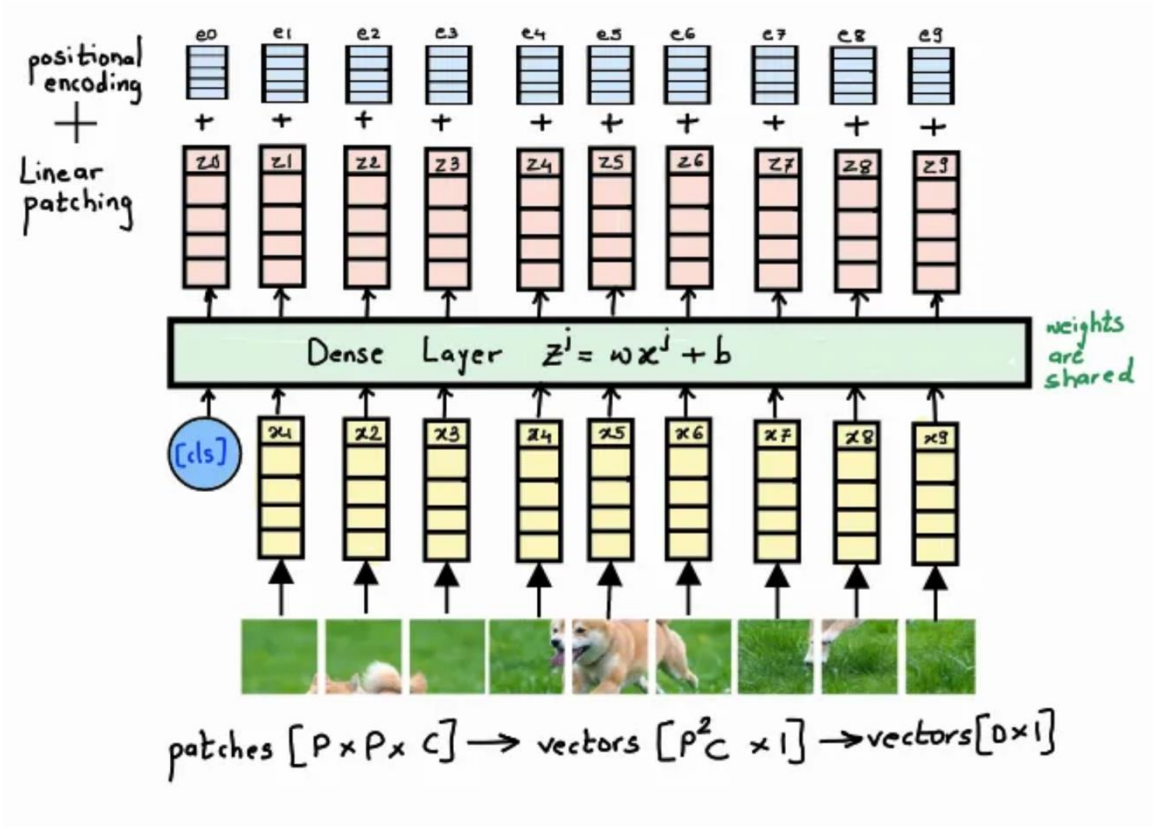
ب) بخش‌های مختلف معماری ViT و نحوه‌ی ایجاد ورودی:



شکل 15. ساختار **ViT**

در شکل فوق بخش‌های مختلف این معماری مشخص شده‌اند. همانطور که در بخش قبلی توضیح دادیم، این معماری به طور کلی به ۴ بخش تقسیم می‌شود. بخش اول که در آن تصاویر پیش‌پردازش شده و عمل patching روی آنها انجام می‌شودو پس از آن، به یک embedding متناظر با هر patch دست پیدا می‌کنیم. در قسمت دوم، position embedding متناظر با هر patch را بدست آورده و با embedding متناظر با آن patch جمع می‌کنیم. (در این بخش یک embedding مضاعف نیز در نظر گرفته می‌ؤود که در طی فرایند training، مدل آن را یاد می‌گیرد.) سپس، در بخش سوم، این بردارهای حاصل را به encoder پاس می‌دهیم. self-attention در encoder امتیازهای attention متناظر با هر patch را محاسبه می‌کند و یک بازنمایی contextualized از هر قطعه از تصویر را ایجاد می‌کند. شبکه‌ی feed-forward در encoder خروجی attention را دریافت کرده و آن را تغییر می‌دهد. همچنین، لایه‌های normalization و residual connection، یادگیری پایدار و عملکرد بهتر مدل را تضمین می‌کنند. در بخش چهارم نیز که طبقه بندی تصویر ورودی انجام می‌شود، بازنمایی متناظر با CLS token به یک شبکه‌ی fully connected وارد می‌شود و طبقه بندی نهایی صورت می‌گیرد.

برای آماده سازی تصاویر به ViT، تمام تصاویر patch میشوند. این عمل طوری صورت می‌گیرد که تصاویر به patchهایی با سایز $[P, P]$ تقسیم بندی می‌شوند. سپس، هر یک از این قطعات تصویر، به یک بردار تبدیل می‌شود و متناظر با هر قطعه، یک *embedding* می‌سازیم. در واقع، اگر تعداد کانال‌های یک *patch* برابر C باشد، آن قطعه از سایز $[P, P, C]$ خواهد بود و برای تبدیل آن به یک *embedding*، به یک بردار با سایز $[P \times P \times C, 1]$ تبدیل می‌شود. سپس، این بردارها به همراه *CLS token* به طور موازی به یک شبکه‌ی *feed-forward* با تابع فعال‌ساز خطی داده می‌شوند که یک *patch projection* خطی از بعد $[D, 1]$ بسازند. پس از آن، متناظر با هر بردار ورودی به لایه‌ی *dense*، یک *positional embedding* نیز در نظر می‌گیریم و با جمع خروجی لایه‌ی *dence* و *positional embedding*ها، میتوانیم بردارهای حاصل را به *encoder* ورودی دهیم. شکل زیر شمای کلی ورودی برای این شبکه‌ها را نشان می‌دهد:



شکل 16. نحوه‌ی ساخت ورودی **ViT**

ج) ایرادات ViT و بهبود معماری آن:

1. یکی از اصلی ترین مشکلاتی که این مدل دارد، این است که به داده‌ی برچسب‌گذاری شده‌ی بسیار زیادی نیاز دارد که به خوبی آموزش ببیند و پیش‌بینی را با دقت انجام دهد. طبق مطالعات انجام شده، اگر اندازه‌ی داده‌های آموزشی ما چیزی حدود ۱۰۰ میلیون تصویر باشد، معمولا CNN عملکرد بهتری از خود نشان می‌دهد. اما اگر تعداد تصاویر از این میزان فراتر رود، دقت ViT در پیش‌بینی‌ها بهتر و بیشتر است. برای اینکه عملکرد شبکه با دیتای کم بهتر شود، میتوان از لایه‌های augmentation در معماری آن استفاده کرد و تصاویر بیشتری تولید کرد. همچنین، میتوان از روش‌های self-supervised استفاده کرد تا دیتای بیشتری تولید کنیم.
2. استفاده از مکانیزم attention بار محاسباتی شبکه را تا ۴ برابر بیشتر می‌کند و بنابراین به منابع محاسباتی زیادی برای این پیاده سازی نیاز داریم. راهکاری که برای بهبود این مشکل در شبکه‌های ترنسفورمری می‌توان ارائه داد، استفاده از attentionهای خطی، اسپارس و یا سلسله مراتبی است که پیچیدگی محاسبات در آنها کمتر است.
3. این مدل‌ها نسبت به شبکه‌های کانولوشنی مدت زمان بیشتری برای یادگیری احتیاج دارند و تنظیم هایپرپارامترها برای آنها زمان بیشتری می‌طلبد. برای مواجهه با این مسئله، میتوان از روش‌های بهینه‌ساز بهتری مثل زمانبتدی نرخ یادگیری تطبیقی، gradient clipping و یا روش‌های مقداردهی اولیه‌ی پیشرفته استفاده کرد.
4. Scaleکردن این مدل‌ها به اندازه‌های بزرگ، دشوار است و منابع محاسباتی زیادی می‌طلبد. برای رفع این مشکل، می‌توان از معماری های scalable مثل Scalable Vision Transformers استفاده کرد که عرض، عمق و رزولوشن را بالانس می‌کند تا عملکرد شبکه بهینه شود.
5. یکی دیگر از مشکلات این شبکه‌ها این است که برخلاف CNNها قادر به درک سلسله مراتب فضایی نیستند که این مسئله منجر به همگرایی کندتر آن‌ها می‌شود. برای بهبود این مشکل، از ترکیب معماری دو شبکه‌ی ترنسفورمری و کانولوشنی می‌توان بهره برد. مثلا می‌توان استخراج ویژگی‌های اولیه را با استفاده از CNN پیاده سازی کرد و پس از آن از ترنسفورمر برای context modeling سراسری بهره برد.

۲-۲. لود و پیش‌پردازش دیتاست

برای این‌کار لازم است تمام تصاویر مجموعه دادگان CIFAR-10 را resize کرده و آنها را به تصاویری با اندازه‌ی ۲۲۴×۲۲۴ تبدیل کنیم. همچنین، طبق گفته‌های مقاله، تصاویر آموزشی را اندکی augment می‌کنیم. برای augmentation از RandomHorizontalFlip در تبدیل متناظر با دادگان آموزشی استفاده کرده‌ایم. همچنین، تمام تصاویر مجموعه دادگان را به تنسور تبدیل کرده و در بازه‌ی [0, 1] scale می‌کنیم. نهایتا، تصویر CIFAR-10 را با میانگین $(0.4914, 0.4822, 0.4465)$ و انحراف معیار $(0.2470, 0.2435, 0.2616)$ برای کانال‌های RGB در بازه‌ی [1 ,1-] نرمال می‌کنیم. پس از اعمال تبدیلات گفته شده روی داده‌های آموزشی و آزمایشی، دادگان آموزشی را به دو بخش train و validation تقسیم می‌کنیم. نهایتا، loader متناظر با هر سه بخش داده‌ها را ایجاد می‌کنیم.

نکته‌ی قابل توجه در این بخش این است که برای مدل کانولوشنی با batch size=512 با memory limit مواجه می‌شدیم. لذا برای هر دو مدل از batch size=256 استفاده کردیم.

۲-۳. fine-tuning شبکه‌ی کانولوشنی

۲-۳-۱. لود مدل کانولوشنی با وزن‌های pretrained دیتاست ImageNet1K و modify کردن مدل

در این بخش، ابتدا مدل کانولوشنی VGG19 را با وزن‌های pretrained روی داده‌های ImageNet1K در دو حالت به عنوان مدل پایه لود می‌کنیم و وزن‌های آخرین بلاک کانولوشنی آن را unfreeze می‌کنیم، درحالیکه سایر لایه‌ها فریز هستند. سپس، بخش fully-connected آن را متناظر با مدل مقاله modify میکنیم. تفاوت دو مدل در pooling است. در اولین مدل، AdaptiveAvgPool2d(7, 7) (بنامیم مدل کانولوشنی ۱) و در مدل دوم AdaptiveAvgPool2d(1, 1) را در نظر گرفته‌ایم (بنامیم مدل کانولوشنی ۲) که در حالت اول حدود ۱۵ میلیون پارامتر آموزش پذیر داریم اما در مدل دوم، تعداد پارامترها برابر تعداد پارامترهای مقاله است.

head مدل به شکل زیر خواهد بود:

- Flatten: خروجی لایه‌های کانولوشنی را flat می‌کند و تنسور چند بعدی را به تنسور یک بعدی تبدیل می‌کند تا بخش کانولوشنی را به بخش fully-connected متصل کند.
- Linear: یک لایه‌ی fully-connected که خروجی لایه‌ی flatten را با اندازه‌ی ۵۱۲× ۷ × ۷ دریافت می‌کند و آن را به اندازه‌ی ۲۵۶ کاهش می‌دهد.
- ELU: از یک فعال‌ساز برای افزایش ویژگی غیرخطی مدل استفاده می‌شود که به یادگیری الگوهای پیچیده کمک می‌کند.
- Dropout: برای جلوگیری از overfitting در حین یادگیری، به طور تصادفی در هر بروزرسانی، نیمی از واحدهای آموزشی غیرفعال می‌شوند.
- Linear: یک لایه‌ی fully-connected که ۲۵۶ ویژگی را از لایه‌ی قبلی میگیرد و آن را به ۱۰ ویژگی کاهش می‌دهد چراکه در دیتاست، ۱۰ کلاس داشتیم.

نکته‌ی قابل توجه این است که نیاز به اضافه کردن softmax به مدل نیست، چرا که از تابع هزینه‌ی cross entropy استفاده می‌کنیم که خود، به طریقی از softmax استفاده میکند و اعمال مجدد آن، موجب redundancy می‌شود.

۲-۳-۲. تعداد پارامترهای trainable مدل کانولوشنی

Model	Trainable parameters
CNN (VGG19-based with the last conv block modified)	15864586

جدول 3. تعداد پارامترهای **trainable** مدل کانولوشنی **modified** شده با پایه‌ی **VGG19** و **AdaptiveAvgPool2d(7, 7)**

Model	Trainable parameters
CNN (VGG19-based with the last conv block modified)	9573130

جدول 4. تعداد پارامترهای **trainable** مدل کانولوشنی **modified** شده با پایه‌ی **VGG19** و **AdaptiveAvgPool2d(1, 1)**

۲-۳-۳. fine-tune کردن مدل کانولوشنی روی دادگان CIFAR-10

هایپرپارامترهای زیر در فرایند fine-tune کردن شبکه با دادگان CIFAR-10 به صورت زیر است:

Epochs	Initial lr	Minimum lr	Loss function	Optimizer	Patience	Batch size
20	0.0001	0.0000001	Cross Entropy	Adam	5	256

جدول 5. هایپرپارامترها در مدل کانولوشنی

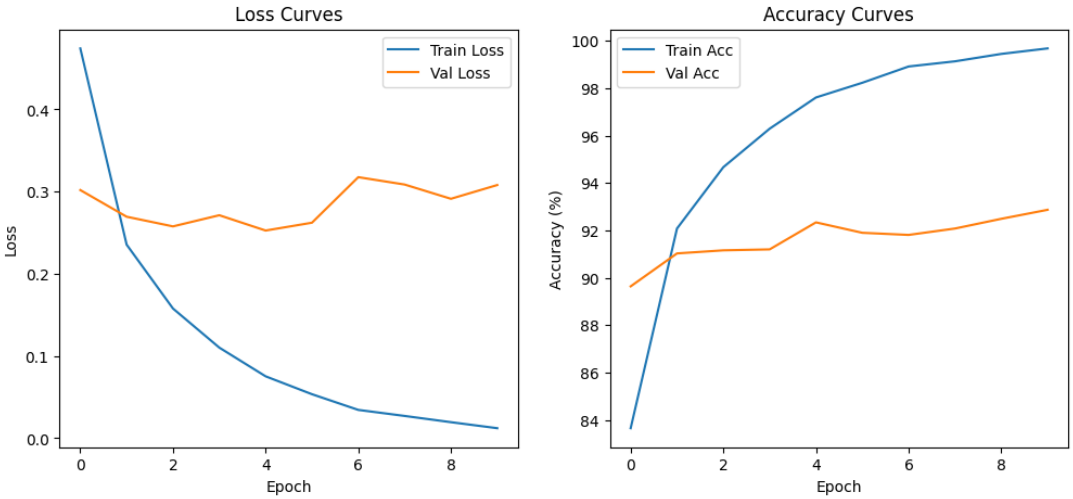
در آموزش مدل کانولوشنی، از تکنیک early stopping استفاده کرده‌ایم طوریکه اگر در ۵ ایپاک متوالی، validation loss بهبود پیدا نکند، اموزش مدل متوقف شود تا دچار overfitting نشود.

همچنین، یک زمانبند نیز در نظر گرفته شده که در صورتی که نرخ یادگیری از minimum learning rate کمتر نبود، آن را در هر ایپاک کاهش دهد.

۲-۳-۴. نمودار تابع هزینه و دقت برای مدل کانولوشنی

شکل زیر، نمودارهای خواسته شده را برای مدل کانولوشنی اول نشان می‌دهد. همانطورکه مشخص است، با استفاده از تکنیک early stopping، با وجود اینکه تعداد ایپاک‌ها روی ۲۰ تنظیم شده بود، اما پس از گذشت ۹ ایپاک، آموزش مدل متوقف شده تا از overfit شدن آن جلوگیری شود. در شکل مشخص است که دقت مدل روی داده‌های آموزشی با گذشت زمان افزایش می‌یابد طوریکه از دقت اولیه‌ی ۸۴٪، به ۹۹.۶۸٪ می‌رسد. همچنین، روی داده‌های اعتبارسنجی، دقت مدل از ۸۹.۶۴٪ به ۹۲.۸۷٪ افزایش می‌یابد. اما از آنجا که early stopping را طوری تعریف کردیم که اگر پس از گذشت ۵ ایپاک، هزینه اعتبارسنجی کاهش نیافت، آموزش متوقف شود، چون کمترین هزینه در ایپاک ۴ بوده و پس از آن به طور نوسانی هزینه به میزان کم افزایش و کاهش یافته، پس در ایپاک ۹ یادگیری متوقف می‌شود.

lossبرای داده‌های آموزشی به طور پیوسته در فرایند یادگیری درحال کاهش است که مورد انتظار بود. همچنین، loss برای داده‌های اعتبارسنجی، تا ایپاک ۴ در حال کاهش است. اما از آن به بعد رفتارش شبیه به overfitting می‌شود که برای جلوگیری از این مسئله، مدل را از یادگیری بازمی‌داریم.



برای داده‌های آموزشی و اعتبارسنجی روی مدل کانولوشنی اول **loss** و **accuracy** شکل 17. نمودار تغییرات

نتایج عملکرد مدل کانولوشنی اول ساخته شده روی مجموعه دادگان، به صورت زیر است:

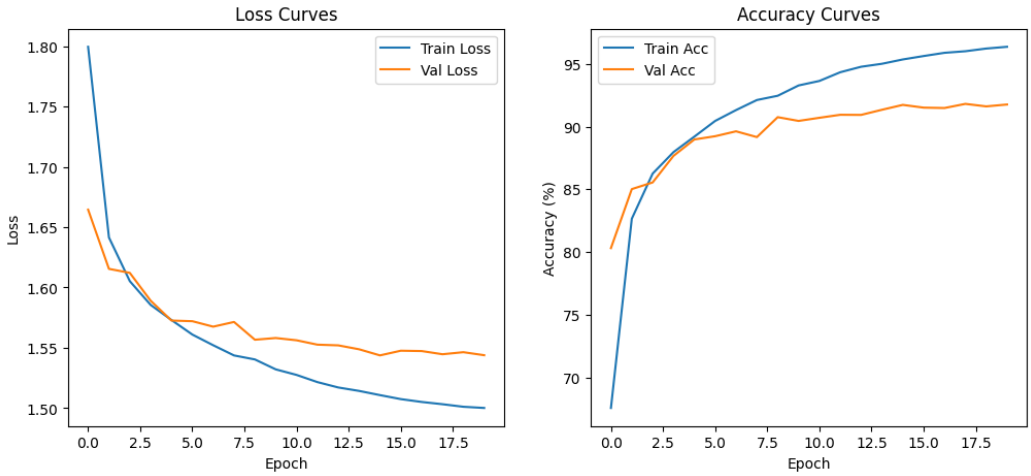
Train accuracy	Train loss	Validation accuracy	Validation loss
99.68%	0.0122	92.87%	0.3080

CIFAR-10 جدول 6. نتایج عملکرد آموزش مدل کانولوشنی اول روی مجموعه دادگان

همچنین، با ارزیابی این مدل روی دادگان آزمایشی، نتایج به صورت زیر حاصل شده است:

Test accuracy: 92.69%.

در مدل دوم کانولوشنی، نتایج به شکل زیر است:



شکل 18. نمودار تغییرات **accuracy** و **loss** برای داده‌های آموزشی و اعتبارسنجی روی مدل کانولوشنی دوم

Train accuracy	Train loss	Validation accuracy	Validation loss
96.38%	1.5002	91.77%	1.5439

جدول 7. نتایج عملکرد آموزش مدل کانولوشنی دوم روی مجموعه دادگان **CIFAR-10**

همچنین، با ارزیابی مدل کانولوشنی دوم روی دادگان آزمایشی، نتایج به صورت زیر حاصل شده است:

Test accuracy: 91.86%.

همانطور که مشخص است، مدل دوم تا انتهای ایپاک ۲۰ اجرا شده و early stopping اعمال نشده است. به طور کلی دقت و هزینه روی داده‌های آموزشی به ترتیب در حال افزایش و کاهش است. روی دادگان اعتبارسنجی نیز به طور کلی همین روند برقرار است. اما در ایپاک های نهایی، تغییر چندان مشهود نیست و مدل در حال همگرایی است. پس برای جلوگیری از overfitting، مدل در جای خوبی متوقف شده است.

۲-۳-۵. میانگین زمان آموزش و اعتبارسنجی در شبکه‌ی کانولوشنی

جدول زیر، میانگین زمان صرف شده برای آموزش و اعتبار سنجی در مدل کانولوشنی اول را نمایش می‌دهد:

Average training time	Average validation time
262.763s	59.575s

جدول 8. میانگین مدت زمان آموزش و اعتبارسنجی مدل اول کانولوشنی

برای مدل دوم کانولوشنی نیز میانگین زمان صرف شده برای آموزش و اعتبار سنجی به صورت زیر است:

Average training time	Average validation time
270.012s	61.278s

جدول 9. میانگین مدت زمان آموزش و اعتبارسنجی مدل دوم کانولوشنی

از آنجا که مجموعه دادگان آموزشی ۴ برابر دادگان اعتبارسنجی است، اعداد فوق منطقی به نظر می‌رسند.

۲-۴. fine-tuning شبکه‌ی ترنسفورمر

۲-۴-۱. لود مدل ترنسفورمری با وزن‌های pretrained دیتاست ImageNet1K و modify کردن مدل

مدل ترنسفورمری انتخابی برای این بخش، DeiTBaseDistilled است. طبق مقاله، وزن‌های اخرین بلاک ترنسفورمری این مدل باید unfreeze شود، درحالیکه سایر لایه‌ها فریز هستند. طبق معماری این شبکه، بلاک ۱۲، آخرین بلاک ترنسفورمری است که باید unfreeze شود. همچنین، classifier شبکه را نیز unfreeze و modify می‌کنیم. تغییرات classifier دقیقا مشابه مدل کانولوشنی و بخش ۲-۳-۱ است، با این تفاوت که اندازه‌ی ورودی لایه‌ی Flatten برابر ۷۶۸ است که مرتبط با معماری شبکه‌ی ترنسفورمری مورد استفاده است.

۲-۴-۲. تعداد پارامترهای trainable مدل ترنسفورمر

Model	Trainable parameters
Transformer(DeiTBaseDistilled -based with the last transformer block modified)	7302686

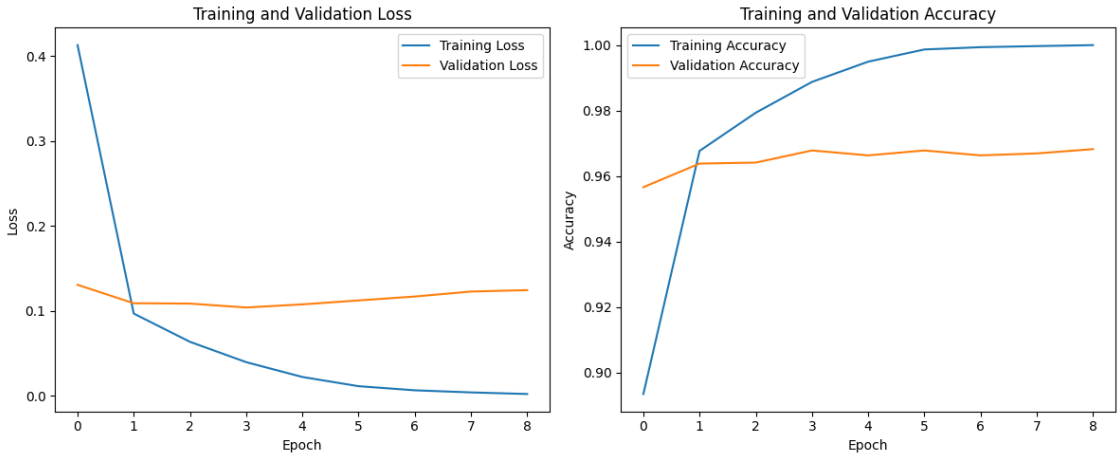
جدول 10 .تعداد پارامترهای **trainable** مدل ترنسفورمری **modified** شده با پایه‌ی **DeiTBaseDistilled**

۲-۴-۳. fine-tune کردن مدل ترنسفورمری روی دادگان CIFAR-10

هایپرپارامترها در این مدل نیز عینا مشابه مدل کانولوشنی تنظیم شده‌اند. همچنین، در این حالت نیز مانند مدل کانولوشنی، از تکنیک‌های زمانبندی و early stopping بهره برده‌ایم تا در فرایند یادگیری، learning rate کاهش یافته و در صورت عدم بهبود عملکرد مدل، آموزش متوقف شود.

۲-۴-۴. نمودار تابع هزینه و دقت برای مدل ترنسفورمری

شکل زیر، نمودارهای accuracy و loss متناظر با دادگان آموزشی و اعتبارسنجی را نشان می‌دهد. آموزش این شبکه، پس از گذشت ۹ ایپاک متوقف شده چراکه از تکنیک early stopping استفاده شده و پس از ایپاک چهارم، validation loss به مرور شروع به رشد می‌کند. بنابراین، برای جلوگیری از overfitting، آموزش شبکه را متوقف کرده ایم.



شکل 19. نمودار تغییرات **accuracy** و **loss** برای داده‌های آموزشی و اعتبارسنجی روی مدل ترنسفورمر

در مورد دقت مدل روی دادگان آموزشی و اعتبارسنجی نیز میتوان گفت که دقت روی دادگان آموزشی با گذشت زمان افزایش یافته و از ۸۹.۳۵٪ به ۹۹.۹۹٪ رسیده است که مطلوب است. روی دادگان اعتبارسنجب نیز دقت افزایش پیدا کرده و از ۹۵.۶۶٪ به ۹۶.۸۲٪ رسیده است. اما همانطور که مشخص است، در ایپاک های نهایی که منجر به early stopping شده‌اند، دقت تغییر چندانی نمیکند.

همانطور که از نمودارها مشخص است، با گذشت زمان در فرایند یادگیری، loss روی دادگان آموزشی به طور پیوسته در حال کاهش است و در ایپاک های نهایی، به صفر میل می‌کند.

نتایج عملکرد این مدل روی دادگان نیز به صورت زیر است:

Train accuracy	Train loss	Validation accuracy	Validation loss
99.99%	0.0022	96.82%	0.1245

جدول 11. نتایج عملکرد آموزش مدل ترنسفورمر روی مجموعه دادگان **CIFAR-10**

همچنین، با ارزیابی مدل ترنسفورمری روی دادگان تست داریم:

Test accuracy: 96.89%

۲-۴-۵. میانگین زمان آموزش و اعتبارسنجی در شبکه‌ی ترنسفورمر

جدول زیر، میانگین زمان صرف شده برای آموزش و اعتبار سنجی در مدل ترنسفورمری را نمایش می‌دهد:

Average training time	Average validation time
496.143s	109.225s

جدول 12. میانگین مدت زمان آموزش و اعتبارسنجی مدل ترنسفورمری

از آنجا که مجموعه دادگان آموزشی ۴ برابر دادگان اعتبارسنجی است، اعداد فوق منطقی به نظر می‌رسند.

۲-۵. مقایسه نتایج

همانطور که واضح است، دقت مدل ترنسفورمری نسبت به هر دو مدل کانولوشنی بررسی شده، به طور قابل توجهی (حدود ۴ الی ۵ درصد) افزایش یافته است. (مدل کانولوشنی اول ۹۲.۸۷٪، مدل کانولوشنی دوم ۹۱.۷۷٪ و ترنسفورمری، ۹۶.۸۲٪) مقدار loss نیز در مقایسه با مدل کانولوشنی، هم برای دادگان آموزشی و هم برای اعتبارسنجی، کاهش یافته است.

در مدل کانولوشنی اول بعد از گذشت ۹ ایپاک، فرایند یادگیری متوقف شده است و تا ایپاک ۴ عملکرد هر دو روی دادگان اعتبارسنجی رو به بهبود بوده است. اما از آن به بعد، برای هر دو مدل، بهبود چشم گیری صورت نمی‌گیرد. اما مدل کانولوشنی دوم تمام ۲۰ ایپاک را به انتها می‌رساند و در ایپاک‌های نهایی تقریبا همگرا می‌شود.

نکته‌ی قابل توجه این است که علاوه بر دقت نهایی مدل ترنسفوری، دقت ابتدایی آن نیز در مواجهه با دقت مدل کانولوشنی در اولین ایپاک بیشتر است و به وضوح در همان ایپاک اول می‌توان تاثیر attention و معماری ترنسورمری را مشاهده کرد و برتری مدل ترنسفورمری به کانولوشنی برای این دیتاست را نتیجه گرفت.

با مقایسه‌ی نتایج این مدل‌ها، میتوان دریافت که مدل ترنسفورمری هم در آموزش و هم در اعتبارسنجی، به طور میانگین حدودا دوبرابر زمان صرف می‌کند. این مسئله، مطابق انتظار است چرا که تکنیک attention پیچیدگی محاسبات را بالا می‌برد و بنابراین، این فرایند مدت زمان بیشتری به طول می‌انجامد.

همچنین، دقت مدل ترنسفورمری روی دادگان آموزشی ۹۶.۸۹٪ است درحالیکه این دقت در مدل کانولوشنی ۹۲.۶۹٪ و ۹۱.۷۷٪ بود.

در مورد تعداد پارامترها در دو مدل نیز میتوان گفت که با وجود اینکه پارامترهای مدل کانولوشنی اول حدودا دو برابر پارامترهای مدل ترنسفورمری هستند، اما میانگین زمان آموزش و اعتبارسنجی مدل ترنسفورمری، نزدیک دو برابر زمان مورد نیاز برای آموزش و اعتبارسنجی مدل کانولوشنی است.

مدل دوم کانولوشنی نیز تعداد پارامترهای بیشتری نسبت به مدل ترنسفورمری دارد اما هم دقت کمتری دارد و هم مدل زمان کمتری نسبت به مدل ترنسفورمری زمان می‌برد.

با توضیحات فوق میتوان نتیجه گرفت علی‌رغم بار محاسباتی بیشتر شبکه‌ی ترنسفورمری نسبت به کانولوشنی، دقت حدود ۴ تا ۵ درصد افزایش پیدا کرده که این میزان افزایش، قابل توجه است و صرف حدود دوبرابر منابع برای استفاده از مدل ترنسفورمری به جای کانولوشنی روی این دیتاست، منجر به دستیابی به مدلی قوی تر و robust تر می‌شود که قدرت تعمیم بیشتری دارد و در مواجهه با دادگان دیده نشده، عملکرد بهتری را از خود نشان می‌دهد.

در پایان، به مقایسه‌ی نتایج به دست آمده با ارقام متناظر آنها در مقاله می‌پردازیم. جدول زیر به همین منظور تدوین شده است:

	Model	Model type	Trainable parameters	Validation accuracy	Pretrained on
Paper	VGG-19	CNN	9,573,130	92.784%	ImageNet1K
	DeiTBaseDistilled	Transformer	7,488,276	96.450%	ImageNet1K
Obtained	VGG-19(1)	CNN	15,864,586	92.87%	ImageNet1K
	VGG-19(2)	CNN	9,573,130	91.77%	ImageNet1K
	DeiTBaseDistilled	Transformer	7,302,686	96.82%	ImageNet1K

جدول 13. مقایسه‌ی عملکرد مدل‌های ساخته شده با مقاله

با بررسی مقادیر در جدول فوق، متوجه می‌شویم در مدل ترنسفورمری، تعداد پارامترها حدودا ۱۸۰ هزار عدد کمتر از مدل مقاله است. دلیل این تفاوت میتواند پارامترهای دیگری باشد که در مدل ساخته شده توسط ما به صورت دیفالت فرض شده‌اند اما در مقاله دیفالت نبوده. یا اینکه ممکن است ورژن متفاوتی از مدل پایه در مقاله استفاده شده است.

اما مدل کانولوشنی دوم تعداد پارامترهایش دقیقا مشابه مقاله است.

دقت مدل ترنسفورمری در مقایسه با مدل مقاله، حدود ۰.۴٪ بیشتر است.

در مورد مدل‌های کانولوشنی نیز می‌توان گفت که مدل اول که تعداد پارامترهای بیشتری دارد، دقت بهتری از مقاله دارد اما مدل دوم که تعداد پارامترهای آن برابر مدل مقاله است، حدود ۱ درصد کمتر است. دلیل این تفاوت‌های ناچیز، میتواند به علت مقدارهی اولیه‌ی تصادفی مدل باشد. همچنین، از آنجا که از early stopping استفاده کرده‌ایم، ممکن است به علت تفاوت در عملیات و مقداردهی های اولیه‌ی تصادفی، فرایند یادگیری مدل متفاوت باشد و به نتایج اندکی متفاوت دست پیدا کنیم. همچنین، augmentation نیز به صورت رندوم انجام شده که این نیز میتواند دلیلی برای این تفاوت اندک باشد.