## 



دانشگاه صنعتي امیرکبیر  
(پلی تکنیک تهران)

دانشكده مهندسی برق

ﮔﺰﺍﺭﺵکار پروژه نهایی

مقدمه ای بر ﻫﻮﺵ ﻣﺤﺎﺳﺒﺎﺗﻲ

پردازش سیگنال

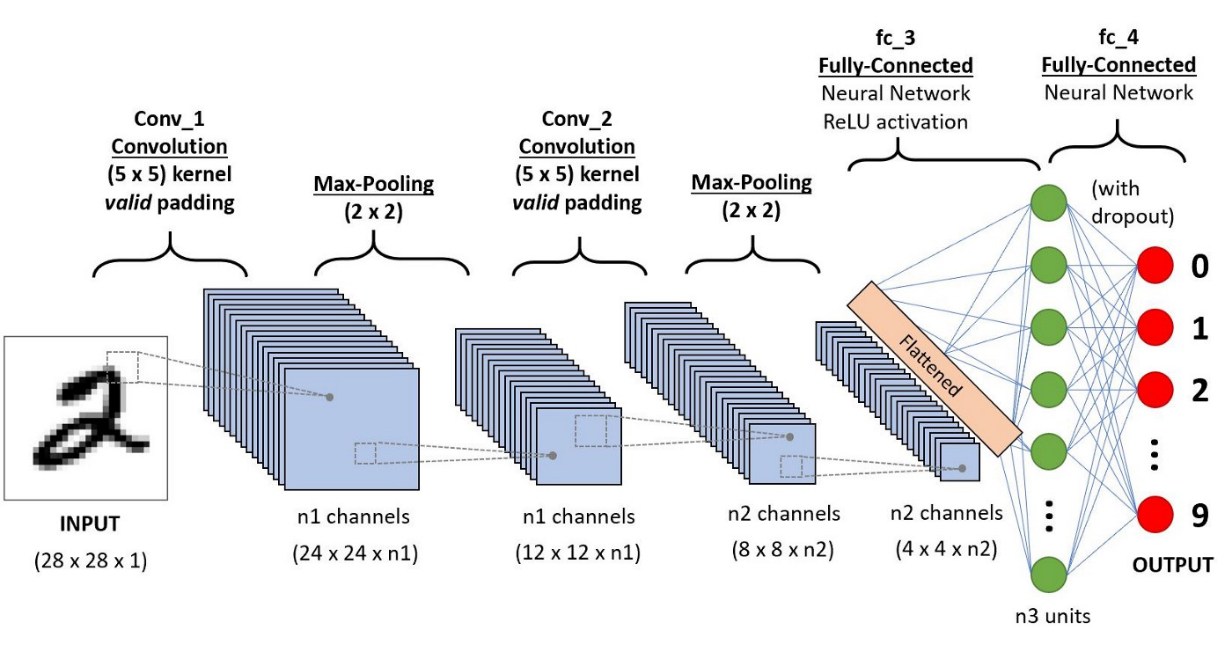
نگارش

علی بابالو

پویا شریفی

# پیش گزارش

## شبکه های CNN

در یادگیری عمیق، شبکه عصبی کانولوشنال (CNN یا ConvNet) یک کلاس از شبکه عصبی مصنوعی (ANN) است که بیشتر برای تجزیه و تحلیل تصاویر بصری استفاده می شود. CNN ها همچنین به عنوان شبکه های عصبی مصنوعی تغییر ناپذیر(SIANN) شناخته می شوند که بر اساس معماری وزن مشترک هسته ها یا فیلترهایی است که در امتداد ویژگی های ورودی اسلاید می شوند و پاسخ های معادل ترجمه معروف به نقشه های ویژگی را ارائه می دهند. برخلاف شهود، اکثر شبکه‌های عصبی کانولوشن به دلیل عملیات نمونه‌برداری پایینی که در ورودی اعمال می‌کنند، نسبت به ترجمه ثابت نیستند. آنها در تشخیص تصویر و ویدئو، سیستم های توصیه کننده، طبقه بندی تصویر، تقسیم بندی تصویر، تجزیه و تحلیل تصویر پزشکی، پردازش زبان طبیعی، رابط های مغز و کامپیوتر و سری های زمانی مالی کاربرد دارند.

شکل 1) شماتیک کلی یک شبکه CNN

CNN ها نسخه های منظم پرسپترون های چندلایه هستند. پرسپترون های چندلایه معمولاً به معنای شبکه های کاملاً متصل هستند، یعنی هر نورون در یک لایه به تمام نورون های لایه بعدی متصل است. "اتصال کامل" این شبکه ها آنها را مستعد به تناسب بیش از حد داده ها (Over Fitting) می کند. روش‌های معمول منظم‌سازی یا جلوگیری از تطبیق بیش از حد عبارتند از: جریمه‌کردن پارامترها در طول تمرین (مانند کاهش وزن) یا کوتاه کردن اتصال (اتصالات نادیده گرفته شده، ترک تحصیل، و غیره). و الگوهای با پیچیدگی فزاینده را با استفاده از الگوهای کوچکتر و ساده تری که در فیلترهای آنها نقش بسته است، جمع آوری کنید. بنابراین، در مقیاس اتصال و پیچیدگی، CNN ها در سطح پایین تر قرار دارند.

شبکه‌های کانولوشن از فرآیندهای بیولوژیکی الهام گرفته شده‌اند که الگوی اتصال بین نورون‌ها شبیه سازماندهی قشر بینایی حیوانات است. تک تک نورون‌های قشری تنها در ناحیه‌ای محدود از میدان بینایی به محرک‌ها پاسخ می‌دهند که به عنوان میدان گیرنده شناخته می‌شود. میدان های گیرنده نورون های مختلف تا حدی با هم همپوشانی دارند به طوری که کل میدان بینایی را پوشش می دهند.

CNN ها از پیش پردازش نسبتا کمی در مقایسه با سایر الگوریتم های طبقه بندی تصویر استفاده می کنند. این بدان معنی است که شبکه یاد می گیرد که فیلترها (یا هسته ها) را از طریق یادگیری خودکار بهینه کند، در حالی که در الگوریتم های سنتی این فیلترها به صورت دستی طراحی شده اند. این استقلال از دانش قبلی و مداخله انسان در استخراج ویژگی یک مزیت عمده است.

شبکه‌های عصبی کانولوشن، نوع تخصصی از شبکه‌های عصبی مصنوعی هستند که از یک عملیات ریاضی به نام کانولوشن به جای ضرب ماتریس عمومی حداقل در یکی از لایه‌های خود استفاده می‌کنند. آنها به طور خاص برای پردازش داده های پیکسلی طراحی شده اند و در تشخیص و پردازش تصویر استفاده می شوند.

در CNN، ورودی یک تانسور با شکل: (تعداد ورودی) × (ارتفاع ورودی) × (عرض ورودی) × (کانال های ورودی) است. پس از عبور از یک لایه کانولوشن، تصویر به یک نقشه ویژگی انتزاعی می شود که به آن نقشه فعال سازی نیز می گویند: (تعداد ورودی ها) × (ارتفاع نقشه ویژگی) × (عرض نقشه ویژگی) × (کانال های نقشه ویژگی).

لایه های کانولوشن ورودی را در هم می پیچند و نتیجه آن را به لایه بعدی منتقل می کنند. این شبیه به پاسخ یک نورون در قشر بینایی به یک محرک خاص است. هر نورون کانولوشنال داده ها را فقط برای میدان گیرنده خود پردازش می کند. اگرچه شبکه‌های عصبی فید فوروارد کاملاً متصل می‌توانند برای یادگیری ویژگی‌ها و طبقه‌بندی داده‌ها استفاده شوند، این معماری به طور کلی برای ورودی‌های بزرگ‌تر مانند تصاویر با وضوح بالا غیر عملی است. این به تعداد بسیار زیادی نورون نیاز دارد، حتی در یک معماری کم عمق، به دلیل اندازه ورودی بزرگ تصاویر، جایی که هر پیکسل یک ویژگی ورودی مرتبط است. به عنوان مثال، یک لایه کاملاً متصل برای یک تصویر (کوچک) با اندازه 100 × 100 دارای 10000 وزن برای هر نورون در لایه دوم است. در عوض، پیچیدگی تعداد پارامترهای آزاد را کاهش می‌دهد و به شبکه اجازه می‌دهد عمیق‌تر شود. برای مثال، صرف نظر از اندازه تصویر، استفاده از یک ناحیه کاشی کاری 5×5، که هر کدام دارای وزن های مشترک یکسانی هستند، تنها به 25 پارامتر قابل یادگیری نیاز دارد. استفاده از وزن‌های منظم‌شده بر روی پارامترهای کمتر، از شیب‌های ناپدید شدن و مشکلات Exploding Gradients که در طول پس انتشار خطا در شبکه‌های عصبی سنتی دیده می‌شود، جلوگیری می‌کند. علاوه بر این، شبکه‌های عصبی کانولوشن برای داده‌هایی با توپولوژی شبکه مانند (مانند تصاویر) ایده‌آل هستند زیرا روابط فضایی بین ویژگی‌های جداگانه در طول کانولوشن و/یا ادغام در نظر گرفته می‌شود.

## پیاده سازی شبکه CNN با TensorFlow

در TensorFlow، هر تصویر ورودی معمولاً به عنوان یک تنسور سه بعدی شکل [ارتفاع، عرض، کانال ها] نشان داده می شود. یک دسته کوچک به عنوان یک تنسور 4 بعدی شکل [اندازه mini-batch، ارتفاع، عرض، کانال ها] نشان داده می شود. وزن یک لایه کانولوشن به صورت یک تنسور 4 بعدی نشان داده می شود. شرایط بایاس یک لایه کانولوشن به سادگی به عنوان یک تنسور 1 بعدی شکل نشان داده می شود.

## Transfer Learning

یادگیری انتقالی به استفاده از دانش به دست آمده از یک مسئله یادگیری ماشین در مشکل دیگر اشاره دارد. به عنوان مثال، استفاده از دانش به دست آمده از تشخیص گربه/سگ برای شناسایی ساختمان ها. مؤلفه اصلی یادگیری انتقالی، استفاده از مدل های از پیش آموزش دیده برای جمع آوری دانش از یک کار و اعمال آن در سایر وظایف است.

مهمتر از همه، Transfer Learning یک جهش بزرگ برای توسعه دهندگان هوش مصنوعی در نظر گرفته می شود، زیرا به ما اجازه می دهد تا برنامه ها را سریعتر و کارآمدتر توسعه دهیم.

## مزایای استفاده از مدل از قبل آموزش دیده شده

اگر ابتدا مزایا را در نظر بگیریم، قابل توجه ترین مزیت پیش تمرین، سهولت استفاده است. فرض کنید ما یک کار یادگیری ماشینی داریم که باید روی آن کار کنیم. تنها کاری که باید انجام دهیم این است که یک مدل از پیش آموزش دیده را پیدا کنیم که در کارهای مشابه آموزش دیده باشد و آن را در کاری که روی آن کار می کنیم اعمال کنیم. نیازی به ساخت مدل از ابتدا نیست.

قبل از آموزش به مدل ها اجازه می دهد تا به سرعت بهینه شوند. این بدان معناست که اگر یک مدل از قبل آموزش دیده استفاده شود، یک مدل می‌تواند سریع‌تر به عملکرد بهینه دست یابد. مدلی که در دانستن اینکه کدام پارامترها احتمالاً به نتایج خوبی دست می یابند می تواند سریعتر در مقایسه با شروع از صفر بهینه شود.

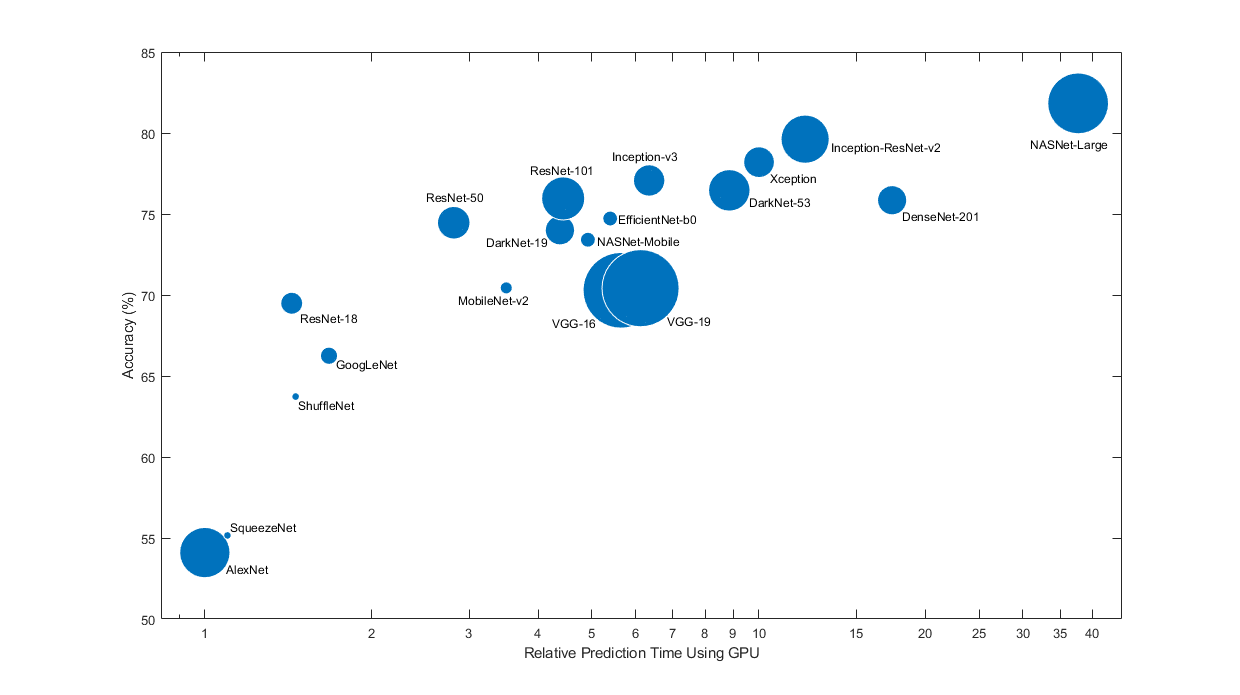
علاوه بر این، مدل‌های از قبل آموزش‌دیده شده این مزیت را دارند که به اندازه ساخت یک مدل از ابتدا به داده‌ نیاز ندارند. این به این دلیل است که اکثر مدل های از پیش آموزش دیده موجود در اینترنت تا به امروز بر روی مجموعه داده های بسیار بزرگ آموزش داده شده اند. از این رو، استفاده از چنین مدلی برای یک کار متفاوت به داده های کمتری برای همگرایی نیاز دارد.

## معایب استفاده از مدل از قبل آموزش دیده شده

اگرچه مفید است، اما قبل از آموزش باید با کمی احتیاط اعمال شود. اولا، ما همیشه به نتایج خوبی که توسط مدل از پیش آموزش دیده به دست می آید، نخواهیم رسید. عوامل متعددی ممکن است باعث این اتفاق شود. به عنوان مثال، استفاده از مجموعه داده از یک دامنه کاملاً متفاوت ممکن است نتایج یکسانی نداشته باشد. علاوه بر این، پارامترهای شبکه، نسبت تقسیم آزمون قطار و سخت افزار برای آموزش مورد استفاده عوامل تعیین کننده هستند.

علاوه بر این، تنظیم دقیق مدل های از پیش آموزش دیده می تواند کار دشواری باشد. برای تنظیم دقیق آنها به زمان و منابع CPU نیاز دارند.

## مقایسه شبکه های از پیش آموزش داده شده استفاده شده

می‌توان یک شبکه طبقه بندی تصاویر از پیش آموزش دیده را انتخاب کرد که قبلاً آموخته است ویژگی های قدرتمند و آموزنده را از تصاویر طبیعی استخراج کند و از آن به عنوان نقطه شروع برای یادگیری یک کار جدید استفاده کرد. اکثر شبکه های از پیش آموزش دیده بر روی زیرمجموعه ای از پایگاه داده ImageNet آموزش داده می شوند که در چالش تشخیص تصویری در مقیاس بزرگ ImageNet (ILSVRC) استفاده می شود. این شبکه ها بر روی بیش از یک میلیون تصویر آموزش دیده اند و می توانند تصاویر را در 1000 دسته شی مانند صفحه کلید، لیوان قهوه، مداد و بسیاری از حیوانات طبقه بندی کنند. استفاده از یک شبکه از پیش آموزش دیده با یادگیری انتقال معمولاً بسیار سریعتر و ساده تر از آموزش شبکه از ابتدا است.

شکل 2)مقایسه شبکه های pretrained شده از منظر Accuracy و زمان پیش بینی

در شکل فوق مقایسه ای از شبکه های مختلف از پیش آموزش دیده به تصویر کشیده شده است که در آن مساحت هر دایره با سایز شبکه بر روی دیسک نیز متناسب است.

همچنین می توان با آموزش شبکه بر روی مجموعه داده های جدید خود با شبکه از پیش آموزش دیده به عنوان نقطه شروع، لایه های عمیق تر را در شبکه تنظیم کرد. Fine-Tune شبکه با یادگیری انتقال اغلب سریعتر و آسانتر از ساخت و آموزش یک شبکه جدید است. شبکه قبلاً مجموعه ای غنی از ویژگی های تصویر را یاد گرفته است، اما وقتی شبکه را Fine-Tune شود، می تواند ویژگی های خاص مجموعه داده جدید را یاد بگیرد. اگر مجموعه داده‌ای بسیار بزرگ باشد، ممکن است انتقال یادگیری سریع‌تر از آموزش از ابتدا نباشد.

Fine-Tune یک شبکه کندتر است و به تلاش بیشتری نسبت به استخراج ویژگی های ساده نیاز دارد، اما از آنجایی که شبکه می تواند استخراج مجموعه متفاوتی از ویژگی ها را بیاموزد، شبکه نهایی اغلب دقیق تر است. Fine-Tune معمولا بهتر از استخراج ویژگی عمل می کند تا زمانی که مجموعه داده های جدید خیلی کوچک نباشد، زیرا در این صورت شبکه داده هایی برای یادگیری ویژگی های جدید دارد.  
ما در این قسمت، از سه شبکه EfficientNetB7، ResNet50 و همچنین Custom VGG استفاده کردیم که در ادامه به توضیحات آن ها می‌پردازیم.

## شبکه ResNet50

ResNet مخفف Residual Network است و نوع خاصی از شبکه عصبی کانولوشنال (CNN) است که در مقاله سال 2015 "یادگیری عمیق باقیمانده برای تشخیص تصویر" توسط He Kaiming، Zhang Xiangyu، Ren Shaoqing و Sun Jian معرفی شده است. CNN ها معمولا برای تقویت برنامه های بینایی کامپیوتری استفاده می شوند.

ResNet-50 یک شبکه عصبی کانولوشن 50 لایه است (48 لایه کانولوشن، یک لایه MaxPool و یک لایه استخر متوسط). شبکه‌های عصبی باقی‌مانده نوعی شبکه عصبی مصنوعی (ANN) هستند که با چیدن بلوک‌های باقیمانده، شبکه‌ها را تشکیل می‌دهند.

معماری اولیه ResNet ResNet-34 بود که شامل 34 لایه وزنی بود. با استفاده از مفهوم اتصالات میانبر، روش جدیدی را برای افزودن لایه‌های کانولوشنال بیشتر به یک CNN، بدون مواجهه با مشکل گرادیان ناپدید، ارائه کرد. یک اتصال میانبر از روی برخی از لایه ها عبور می کند و یک شبکه معمولی را به یک شبکه باقی مانده تبدیل می کند.

شبکه معمولی مبتنی بر شبکه های عصبی VGG (VGG-16 و VGG-19) بود - هر شبکه کانولوشن دارای یک فیلتر 3×3 بود. با این حال، ResNet فیلترهای کمتری دارد و پیچیدگی کمتری نسبت به VGGNet دارد. یک ResNet 34 لایه می تواند به عملکرد 3.6 میلیارد FLOP دست یابد و یک ResNet 18 لایه کوچکتر می تواند به 1.8 میلیارد FLOP دست یابد که به طور قابل توجهی سریعتر از یک شبکه VGG-19 با 19.6 میلیارد FLOP است (در مقاله ResNet بیشتر بخوانید. و همکاران، 2015).

معماری ResNet از دو قانون اساسی طراحی پیروی می کند. ابتدا تعداد فیلترها در هر لایه بسته به اندازه نقشه ویژگی خروجی یکسان است. دوم، اگر اندازه نقشه ویژگی نصف شود، تعداد فیلترهای آن دو برابر برای حفظ پیچیدگی زمانی هر لایه است.

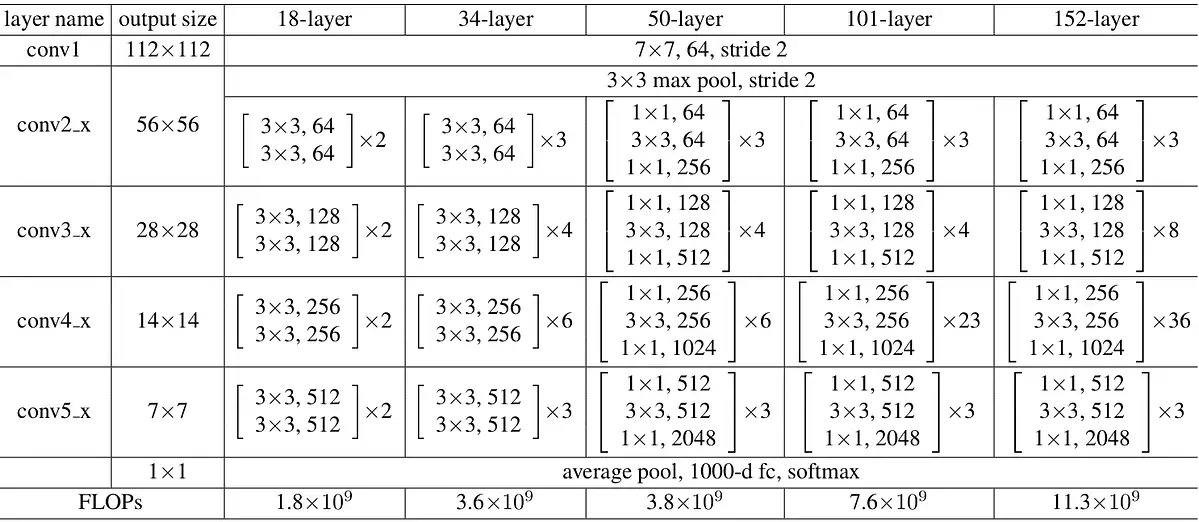
ResNet-50 دارای معماری مبتنی بر مدلی است که در بالا نشان داده شده است، اما با یک تفاوت مهم. ResNet 50 لایه از یک طرح گلوگاه برای بلوک ساختمان استفاده می کند. یک بلوک باقیمانده گلوگاه از پیچیدگی‌های 1×1 استفاده می‌کند که به نام «گلوگاه» شناخته می‌شود، که تعداد پارامترها و ضرب‌های ماتریس را کاهش می‌دهد. این امکان آموزش سریعتر هر لایه را فراهم می کند. از یک پشته سه لایه به جای دو لایه استفاده می کند.

معماری ResNet 50 لایه شامل عناصر زیر است که در جدول زیر نشان داده شده است:

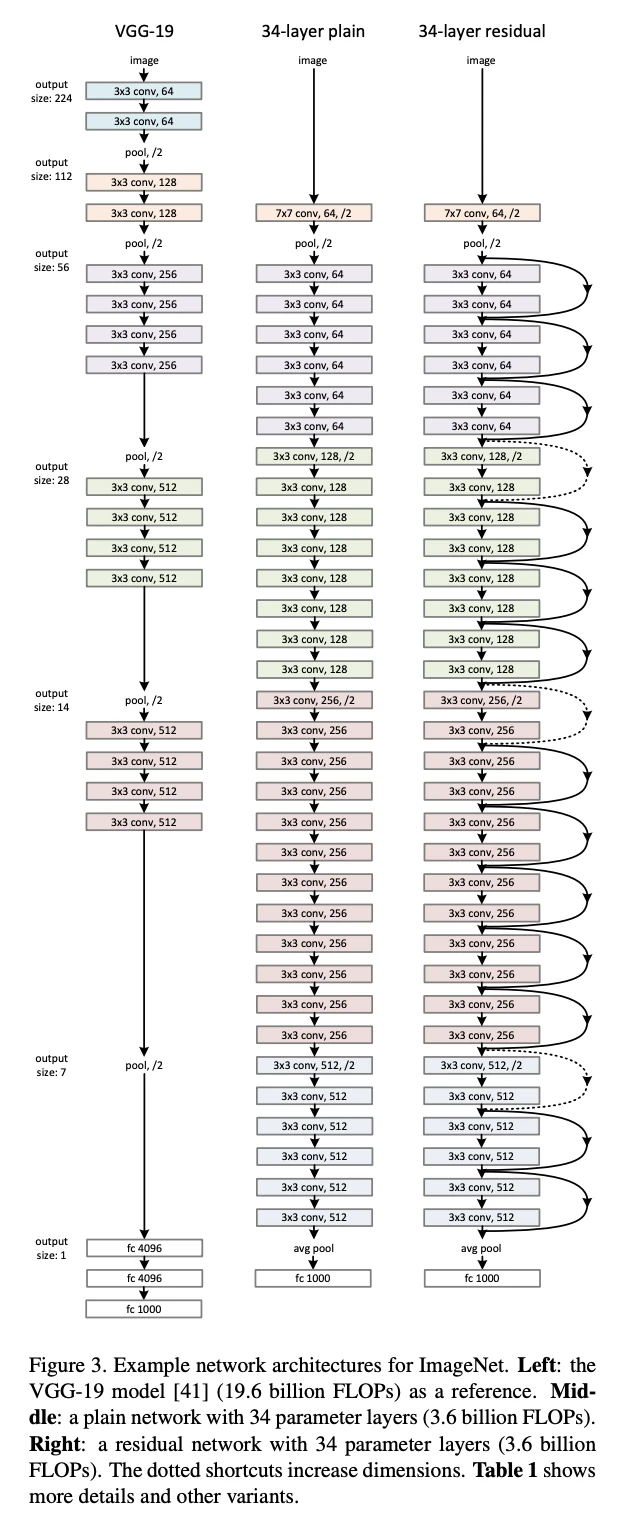
* **یک هسته کانولوشن 7×7** در کنار 64 هسته دیگر با یک stride در اندازه 2.
* **یک لایه Max pooling** حداکثر با اندازه stride 2.
* **9 لایه دیگر** 3 در 3 با 64 هسته کانولوشن، دیگری با اندازه 1 در 1 و 64 هسته، و لایه سوم با با اندازه 1 در 1 با 256 هسته. این 3 لایه 3 بار تکرار می شوند.
* **12 لایه دیگر** با 128 هسته با اندازه 1 در 1، 128 هسته با اندازه 1 در 1 و 512 هسته با اندازه 1 در 1، که 4 بار تکرار شده است.
* **18 لایه دیگر** با 256 هسته 1 در 1و 2 هسته 3 در 3، 256 و یک 1 در 1 با 1024 هسته که 6 بار تکرار شده است.
* **9 لایه دیگر** با 512 هسته 1 در 1، 512 هسته 3 در 3 و 2048 هسته 1 در 1 که 3 بار تکرار شده است.

(تا این لحظه شبکه دارای 50 لایه است)

* **Average Pooling**، به دنبال آن یک لایه کاملا متصل با 1000 گره، با استفاده از تابع فعال ساز softmax.



شکل 9) ساختار شبکه ResNet



شکل 10) مقایسه ساختار ResNet با شبکه معمولی CNN و VGG

به طور خلاصه ResNet-50 یک شبکه عصبی کانولوشن است که عمق 50 لایه دارد. یک نسخه از پیش آموزش دیده شبکه آموزش داده شده بر روی بیش از یک میلیون تصویر از پایگاه داده ImageNet قابل بارگیری است. شبکه از پیش آموزش دیده می تواند تصاویر را به 1000 دسته شی، مانند صفحه کلید، ماوس، مداد و بسیاری از حیوانات طبقه بندی کند.

**شبکه EfficientNet B7**

در معماري هاي طراحي شده در شبكه هاي كانولوشني، سه روش براي افزايش دقت استفاده مي­شود.اين سه روش شامل: افزايش عمق شبكه، ارتفاع شبكه و همچنين افزايش رزولوشن ورودي مي­باشد. كه افزايش هر كدامم از اين ويژگي­ها مي­تواند باعث بهبود عملكرد شبكه شود.

در اين مقاله به بررسي رابطه اين سه ويژگي مي­پردازد و بديهي مي­باشد كه اين سه ويژگي با يكديگر ارتباط مستقيمي دارند، بدين صورت كه با افزايش رزولوشن، ويژگي بيشتري براي بررسي وجود دارد بنابراين شبكه مي­تواند عمق بيشتري داشته باشد.

در واقع مي توان به effecientNet به عنوان يك نوع جستجو براي كارآمد ترين شبكه ي عصبي با توجه به ميزان توان محاسباتي نگاه كرد.

 در اين مقاله ايده اي در ر‌ابطه با طراحي شبكه جديد مطرح نشده است. بلكه با توجه به اينكه دستگاه هاي مختلف از توان پردازشي متفاوتي بهره مند هستند مي‌خواهيم شيوه اي داشته باشيم كه با توجه به دستگاه در دسترس و توانايي پردازش موجود چگونه يك شبكه را Scale كنيم. همچنين اگر از نظر زماني، ميزان زمان لازم براي آموزش شبكه را در نظر بگيريم با scale down شبكه زود تر و با scale up شبكه به مدت طولاني تري نياز به آموزش دارد. به طور مثال اگر بخواهيم شبكه ي ما سريع‌تر آموزش ببيند و كمي كاهش دقت در نتايج شبكه مسئله ي خيلي مهمي نباشد ميتوان از اين روش استفاده نمود.

بنابراين Efficient-net يك راهي براي به دست آوردن بهينه ترين ميزان براي scale up كردن با توجه به شرايط موجود مي باشد.

سه روش براي scale up كردن يك شبكه ي كانوولوشني به صورت زير مي‌باشد:

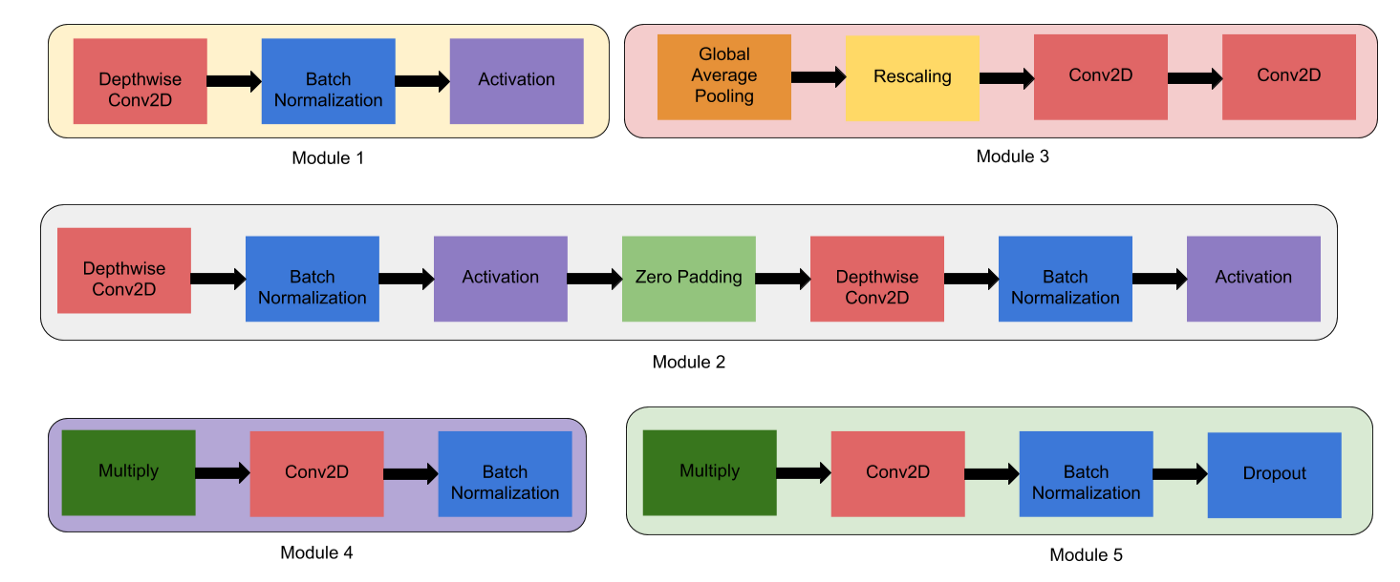
روش اول افزاش عمق: اين روش بيشترين استفاده را در معماري هاي موجود تا كنون داشته است. منظور از افزايش عمق، افزايش تعداد لايه هاي يك شبكه مي باشد.

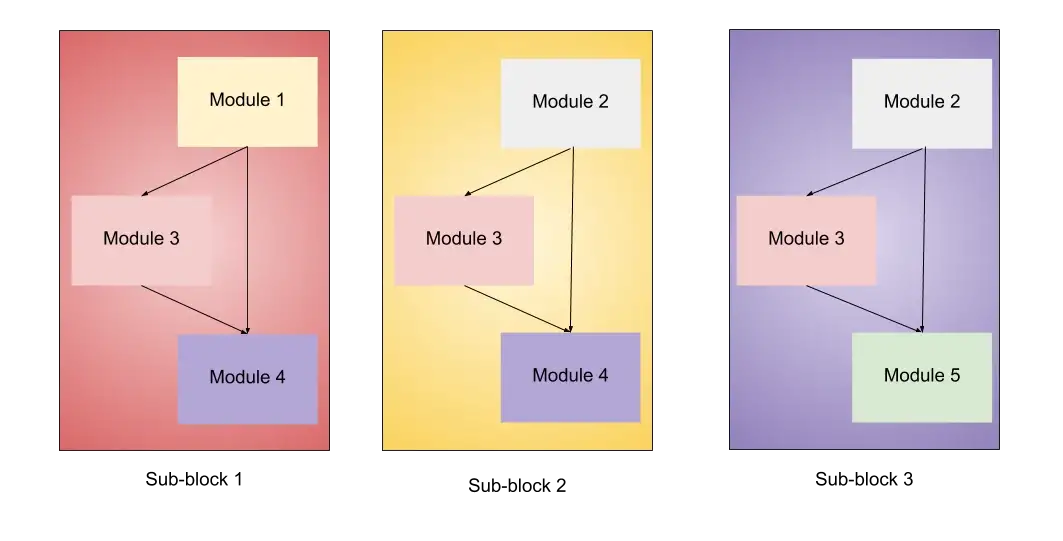
روش دوم افزايش عرض: از اين روش نسبت به عمق كمتر استفاده مي‌شود. منظور از عرض نيز مقدار كانال هاي يك شبكه مي ‌باشد.

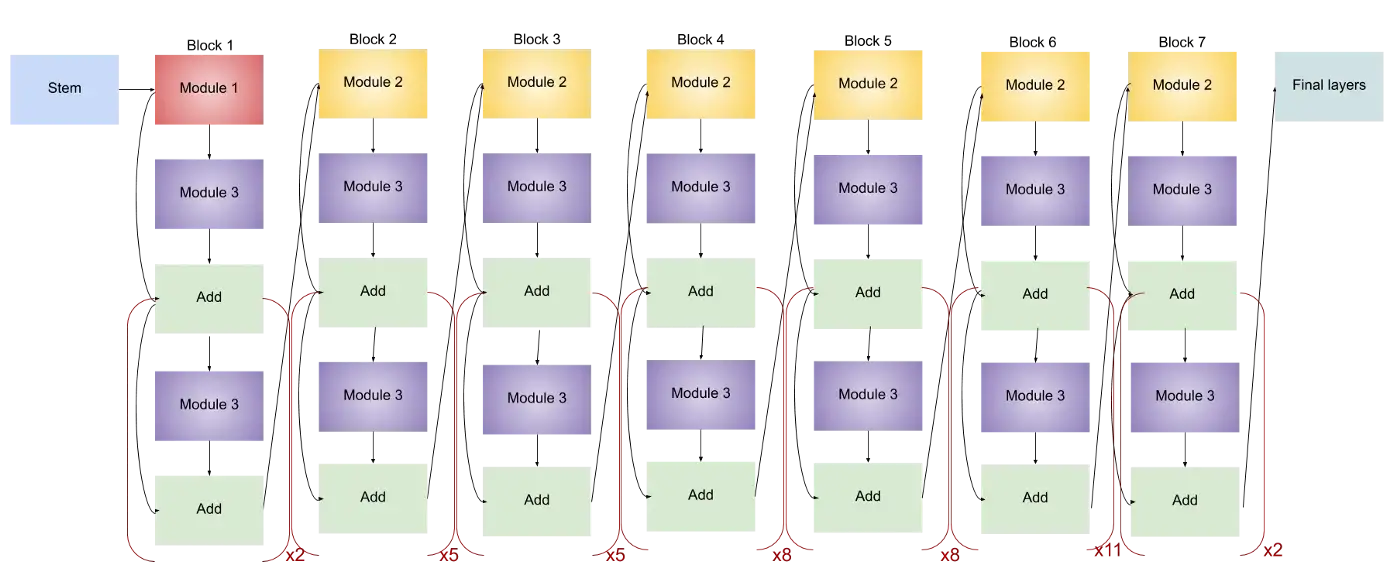
روش سوم افزايش resolution عكس ورودي: از اين روش نيز گاها در مقالات مشاهده شده كه استفاده شده است.

معماري كه به عنوان baseline در نظر گرفته مي شود نيز از اهميت بسياري برخوردار است، زيرا efficient netمعماري baseline را تغييري نمي دهد و صرفا آن را scale مي‌كند. و علاوه بر آن به طور مثال اگر ‌Alexnet به عنوان ‌baseline استفاده شود، و بعد از تغيير scale با استفاده از عرض، طول و رزولوشن روي همان Alexnet مقايسه مي‌شود و ب شبكه‌ي ديگري به طور مثال مانند REsnet مقايسه نمي شود. پس بنابراين داشتن يك baseline مناسب نيز از اهميت ويژه اي برخوردار است.

معماری شبکه Efficientnet B7 شامل چند ماژول است که ساب ماژول هایی را تشکیل می­دهند که آن ساب­ماژول های در طراحی شبکه استفاده می­شوند که تصاویر آنرا در زیر مشاهده می­کنید:

شکل۱۱) ماژول های طراحی EfficientNet

شکل۱۲)ساب ماژول های شبکه

شکل۱۳) شبکه نهایی EfficientNet B7

**توضیح کد**

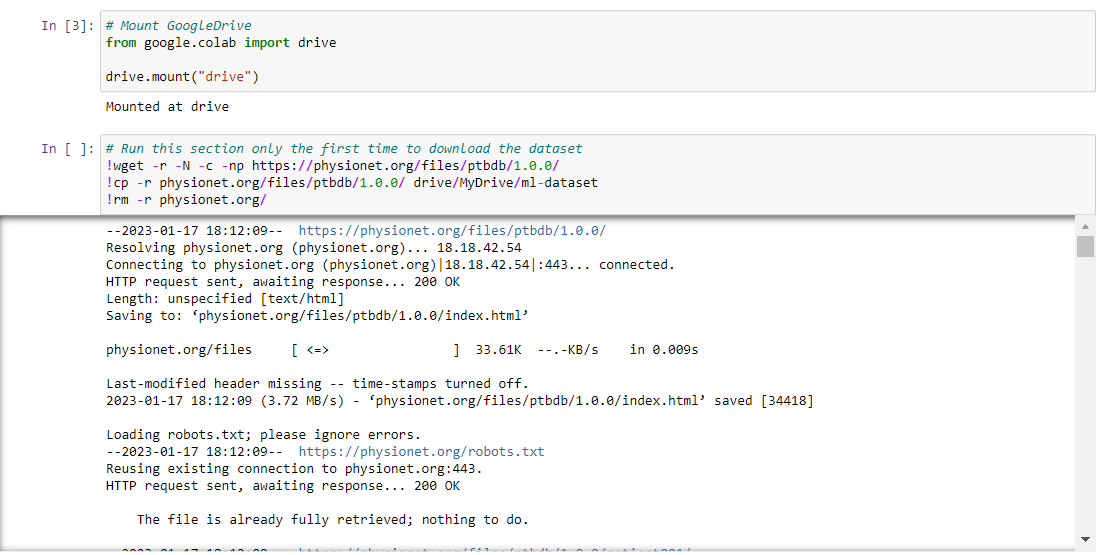
* Importing Libraries

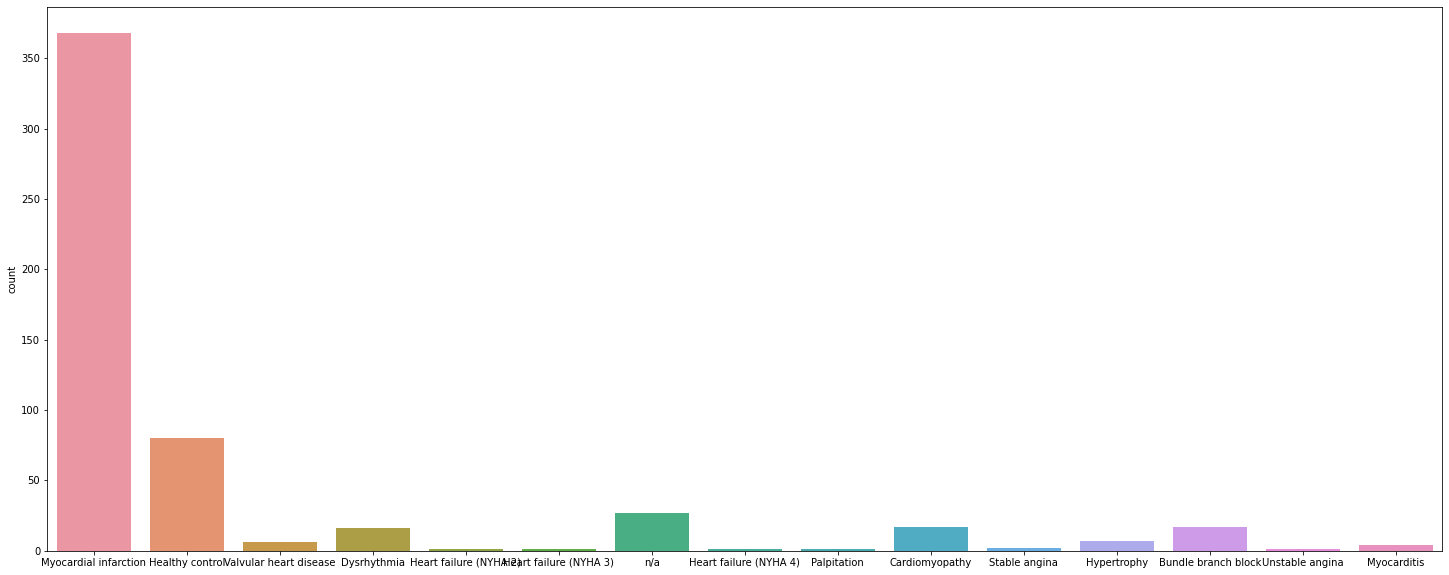


شکل۱۳) پکیج های مورد نیاز را اضافه می­کنیم.

* Dataset Loading

در این قسمت با دستور win get دیتاست را دانلود و به درابو خودمان اضافه و از آنجا آن را لود می­کنیم.

شکل۱۴) خواندن دیتاست

بعد از خواندن دیتاست برخی از توزیع های مربوط به بیماران و پراکندگی داده های مربوط به آنان را چاپ می­کنیم.

شکل ۱۵) توزیع بیماری های قلبی

Chart, bar chart, histogram

Description automatically generated

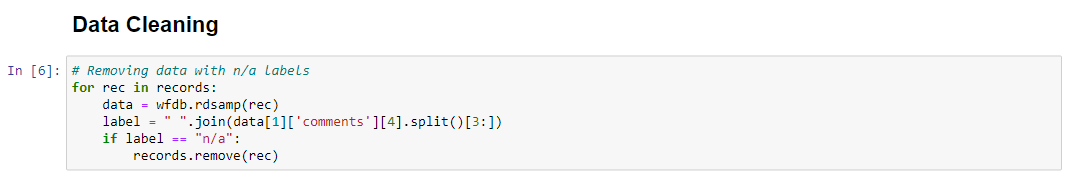
شکل۱۶) توزیع سن بیماران

Chart, bar chart

Description automatically generated

شکل۱۷) توزیع جنسیت بیماران

سپس دیتا هایی که مقدار nan دارند با کد زیر از دیتاست حذف می­کنیم.



شکل۱۸) Data Cleaning

پس از حذف کردن دیتا های نامناسب دیتای سیگنال را بوسیله فیلتر های FIR, IIR نویزشان را می­گیریم

شکل۱۹) فیلتر IIR

Graphical user interface

Description automatically generatedشکل۲۰) فیلتر FIR

* Generator

بعلت حجیم بودن دیتاست بهتر است از جنریتور ها استفاده کنیم این تایپ متغیر به اندازه batch size دیتا هارا گرفته و به حافظه موقت فرستاده و شبکه را با آن آموزش می­دهد در واقع این توابع با yield کردن تابع به جای return کردن و بستن تابع و کپی کردن کامل دیتاست بر روی ram بصورت کلی باعث عدم استفاده از حجم زیاد سخت افزار می­شوند و آموزش را ممکن می­کنند.

شکل۲۱) دیتا جنریتور

Graphical user interface, text, application

Description automatically generated

شکل۲۲) تقسیم دیتاست به داده أموزش و تست و ولیدیشن

* Training – ResNet

در این مرحله ما از رزنت استفاده کردیم اما برای حجیم نشدن شبکه از ۳- لایه اول این شبکه استفاده کردیم و نکته قابل ذکر دیگر این است که ما ابتدا از برای تابع فعال ساز لایه آخرمان از softmax استفاده کردیم سپس با توصیه مهندس کریمی از تابع فعالساز sigmoid نیز برای لایه أخر استفاده کردیم که نتایج را در اشکال زیر میبینید ( سافت­مکس عملکرد بهتری از سیگموید نشان داد):

مدلی که از آن برای پردازش صوت در این پروژه استفاده کردیم ,مدل زیر است:

from keras import layers

from keras.losses import BinaryCrossentropy

input\_s = layers.Input((10,1000,1))

base\_res = tf.keras.applications.resnet50.ResNet50(include\_top=False,

                                         weights=None,

                                         input\_tensor = input\_s)

pretrained\_model\_res = tf.keras.Model(inputs=base\_res.input, outputs=base\_res.layers[30].output)

model\_res = keras.Sequential([

    pretrained\_model\_res,

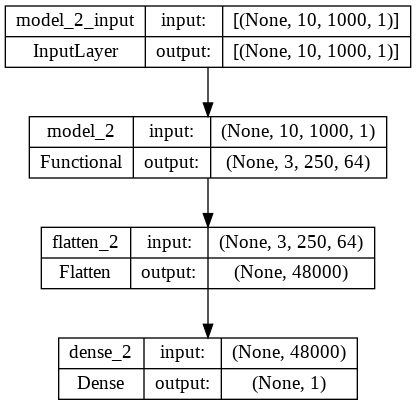
    tf.keras.layers.Flatten(),

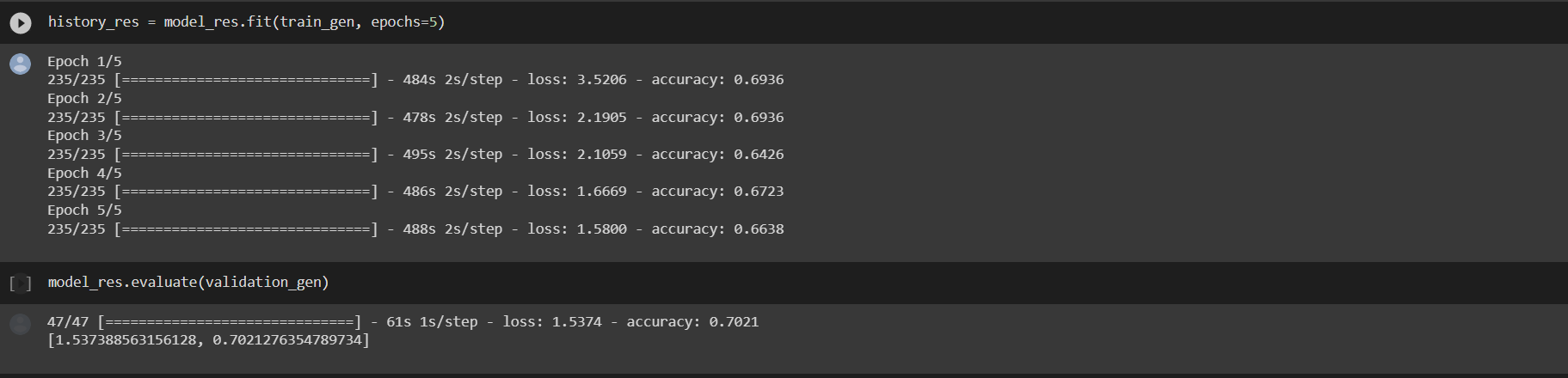
    tf.keras.layers.Dense(1, activation="softmax")

])

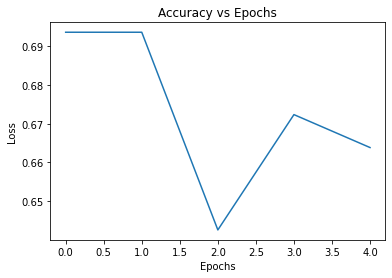
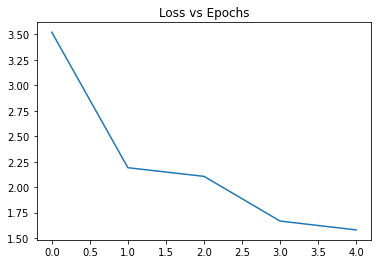
model\_res.compile(optimizer='adam', loss=BinaryCrossentropy(), metrics=['accuracy'])

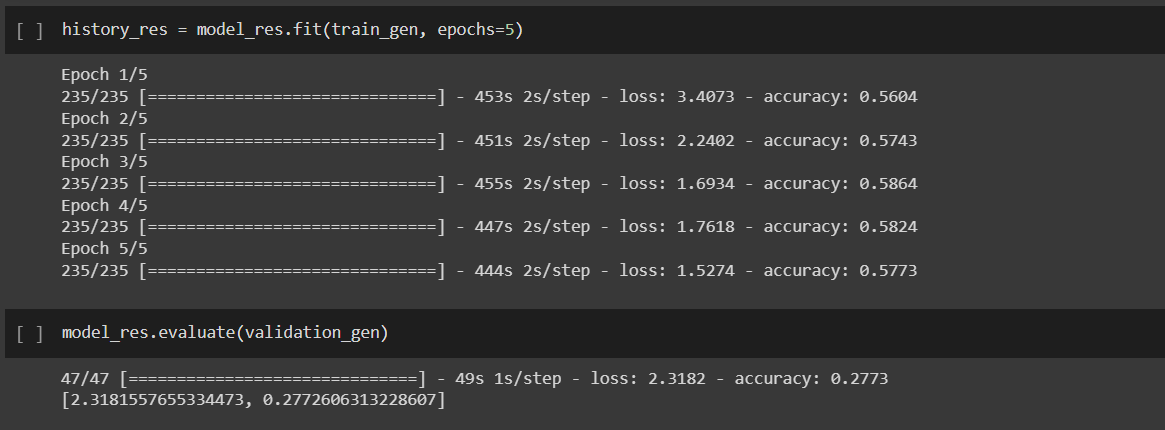
model\_res.summary()

شکل۲۳) معماری شبکه رزنت

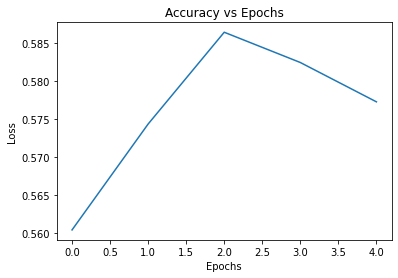
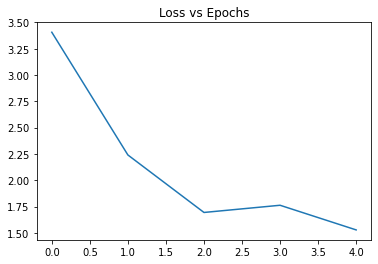
که نتیجه این شبکه با فعال ساز لایه أخر سافت­مکس بصورت زیر می­باشد:

شکل۲۴) فیت کردن مدل رزنت

****شکل۲۵) دفت و خطا مدل رزنت با سافت­مکس

در قسمت پایین هم نتایح با تابع فعال ساز سیگموید را مشاهده می­کنید:

شکل۲۶) آموزش شبکه رزنت با سیگموید

شکل۲۷) نتایح رزنت با تابع فعالساز سیگموید

* Training – EfficientNet B7

مانند مدل قبلی در اینجا از ۱۰۰ لایه اول EfficientNet B7 بهره بردیم که کد مدل همراه با نتایج در زیر مشخص هستند:

from keras import layers

input\_s = layers.Input((10,1000,1))

base\_effB7 = tf.keras.applications.efficientnet.EfficientNetB7(include\_top=False,

                                         weights=None,

                                         input\_tensor = input\_s)

pretrained\_model\_effB7 = tf.keras.Model(inputs=base\_effB7.input, outputs=base\_effB7.layers[100].output)

model\_effB7 = keras.Sequential([

    pretrained\_model\_effB7,

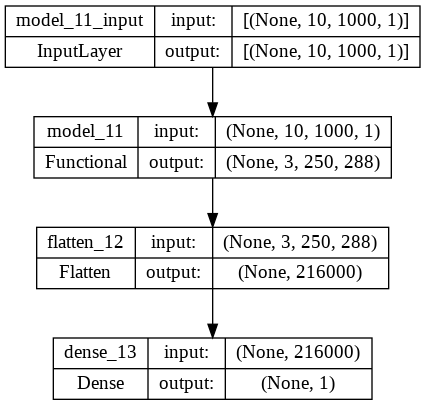
    tf.keras.layers.Flatten(),

    tf.keras.layers.Dense(1, activation=”softmax”)

])

model\_effB7.compile(optimizer=’adam’, loss=tf.keras.losses.BinaryCrossentropy(), metrics=[‘accuracy’])

model\_effB7.summary()

****شکل۲۸) معماری شبکه EfficientNet B7

A screenshot of a computer

Description automatically generatedشکل۲۹) آموزش شبکه EfficientNet B7

## C:\Users\shadi\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\F66E551B.tmpC:\Users\shadi\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\B48BD3D4.tmpشکل۳۰) نتایح معماری EfficientNet

## Custom VGG

در اینجا نیز یک مدل VGG کاستوم خودمان طراحی کردیم که نتایح و معماری را در پایین مشاهده می­کنید.

model\_VGG = tf.keras.Sequential()

# VGG\_1

model\_VGG.add(tf.keras.layers.Conv2D(32, 3, activation='relu', kernel\_initializer='he\_uniform', padding='same', input\_shape=(10, 1000, 1)))

model\_VGG.add(tf.keras.layers.BatchNormalization())

model\_VGG.add(tf.keras.layers.Conv2D(32, 3, activation='relu', kernel\_initializer='he\_uniform', padding='same', input\_shape=(10, 1000, 1)))

model\_VGG.add(tf.keras.layers.BatchNormalization())

model\_VGG.add(tf.keras.layers.MaxPooling2D(pool\_size=(2, 2)))

model\_VGG.add(tf.keras.layers.Dropout(0.2))

# VGG\_2

model\_VGG.add(tf.keras.layers.Conv2D(64, 3, activation='relu', kernel\_initializer='he\_uniform', padding='same', input\_shape=(10, 1000, 1)))

model\_VGG.add(tf.keras.layers.BatchNormalization())

model\_VGG.add(tf.keras.layers.Conv2D(64, 3, activation='relu', kernel\_initializer='he\_uniform', padding='same', input\_shape=(10, 1000, 1)))

model\_VGG.add(tf.keras.layers.BatchNormalization())

model\_VGG.add(tf.keras.layers.MaxPooling2D(pool\_size=(2, 2)))

model\_VGG.add(tf.keras.layers.Dropout(0.3))

# VGG\_3

model\_VGG.add(tf.keras.layers.Conv2D(128, 3, activation='relu', kernel\_initializer='he\_uniform', padding='same', input\_shape=(10, 1000, 1)))

model\_VGG.add(tf.keras.layers.BatchNormalization())

model\_VGG.add(tf.keras.layers.Conv2D(128, 3, activation='relu', kernel\_initializer='he\_uniform', padding='same', input\_shape=(10, 1000, 1)))

model\_VGG.add(tf.keras.layers.BatchNormalization())

model\_VGG.add(tf.keras.layers.MaxPooling2D(pool\_size=(2, 2)))

model\_VGG.add(tf.keras.layers.Dropout(0.4))

# Output

model\_VGG.add(tf.keras.layers.Flatten())

model\_VGG.add(tf.keras.layers.Dense(128, activation='relu', kernel\_initializer='he\_uniform'))

model\_VGG.add(tf.keras.layers.BatchNormalization())

model\_VGG.add(tf.keras.layers.Dropout(0.5))

model\_VGG.add(tf.keras.layers.Dense(1, activation='softmax'))

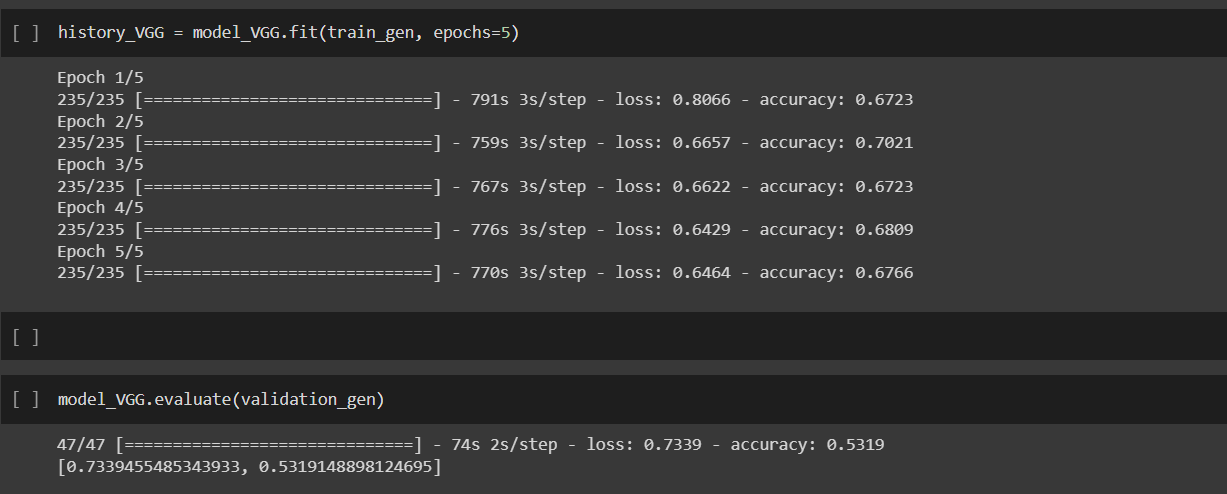
Text

Description automatically generated with medium confidence

شکل30) معماری شبکه 3 Layer VGG custom

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedشکل۳۱) آموزش شبکه VGG

شکل۳۲) نتایح شبکه VGG

* **نتیجه گیری**

در مقاله رفرنس ما که در منابع ذکر شده محققان شبکه resnet شان را در ۱۰۰۰۰اپوک با کارت گرافیک GTX1080 Ti آموزش دادند که به دقت ۹۳ درصدی در ۲ ساعت رسیدند با توجه به سخت افزار محدود ما و بهینه نبودن مدلمان در طی ۵ اپوک به دقت تقریبا ۷۰ درصدی رسیدیم نشان از عملکرد خوب شبکمان است.

* Reference

ECG Heartbeat Classification: A Deep Transferable Representation – [Mohammad Kachuee, Shayan Fazeli, Majid Sarrafzadeh University of California, Los Angeles (UCLA) Los Angeles, USA]