به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین دوم

نام عضو اول	پریسا محمدی
شماره دانشجویی	۸۱۰۱۰۱۵۰۹
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 آلزایمر با استفاده از تصویر برداری مغزی(ADNI)

۱-۱. معرفی مقاله

در دهه کنونی، پیشرفت‌های مراقبت‌های بهداشتی به دلیل کمک طولانی‌تر به مردم، توجه گسترده‌ای را به خود جلب کرده است. زنده ماندن و زندگی بهتر بیماری آلزایمر (AD) شایع ترین بیماری تخریب کننده عصبی و زوال عقل است. این ارزش پولی مراقبت از بیماران مبتلا به آلزایمر به طور چشمگیری افزایش می یابد. ضرورت داشتن یک سیستم کامپیوتری برای طبقه بندی اولیه و دقیق AD بسیار مهم می شود. الگوریتم های یادگیری عمیق مزایای قابل توجه به جای روش های یادگیری ماشینی دارند. بسیاری از مطالعات تحقیقاتی اخیر که از اسکن MRI مغز و شبکه های عصبی کانولوشنال (CNN) استفاده کرده اند به نتایج امیدوارکننده ای برای تشخیص بیماری آلزایمر دست یافتند. بر این اساس، در این مطالعه از مجموعه داده های تصویربرداری عصبی بیماری آلزایمر (ADNI) برای طبقه بندی دودویی AD و MCI استفاده شده است.

۱-۲. پیش پردازش تصاویر

در ابتدا سایز تصاویر را اندازه می گیریم ومشاهده می کنیم که تصاویر سایز های متفاوتی دارند در نتیجه در اولین گام پیش پردازش تصاویر ،طبق مقاله، ابتدا سایز تصاویر را به 128 در 128 تبدیل می کنیم.

```
The size of the image Alzheimer_s_Disease_Neuroimaging_ADNI_Dataset/MCI\ADNI_002_S_0729_MR_Axial_T2-Star__br_raw_201
30819102121023_8_S198151_I385950.jpg is 7265 bytes.
The size of the image Alzheimer_s_Disease_Neuroimaging_ADNI_Dataset/MCI\ADNI_002_S_0729_MR_Axial_T2-Star__br_raw_201
30819102123509_29_S198151_I385950.jpg is 6922 bytes.
The size of the image Alzheimer_s_Disease_Neuroimaging_ADNI_Dataset/MCI\ADNI_002_S_0729_MR_Axial_T2-Star__br_raw_201
30819102125431_36_S198151_I385950.jpg is 5930 bytes.
The size of the image Alzheimer_s_Disease_Neuroimaging_ADNI_Dataset/MCI\ADNI_002_S_0729_MR_Axial_T2-Star__br_raw_201
30819102128757_23_S198151_I385950.jpg is 7820 bytes.
The size of the image Alzheimer_s_Disease_Neuroimaging_ADNI_Dataset/MCI\ADNI_002_S_0729_MR_Axial_T2-Star__br_raw_201
30819102129676_22_S198151_I385950.jpg is 6887 bytes.
The size of the image Alzheimer_s_Disease_Neuroimaging_ADNI_Dataset/MCI\ADNI_002_S_0729_MR_Axial_T2-Star__br_raw_201
30819102132924_31_S198151_I385950.jpg is 6785 bytes.
The size of the image Alzheimer_s_Disease_Neuroimaging_ADNI_Dataset/MCI\ADNI_002_S_0729_MR_Axial_T2-Star__br_raw_201
30819102220973_21_S198151_I385950.jpg is 6734 bytes.
The size of the image Alzheimer_s_Disease_Neuroimaging_ADNI_Dataset/MCI\ADNI_002_S_0729_MR_Axial_T2-Star__br_raw_201
30819102242234_37_S198151_I385950.jpg is 5446 bytes.
The size of the image Alzheimer_s_Disease_Neuroimaging_ADNI_Dataset/MCI\ADNI_002_S_0729_MR_Axial_T2-Star__br_raw_201
30819102245145_11_S198151_I385950.jpg is 6200 bytes.
```

شکل 1. سایز تعدادی از تصاویر

سپس برای افزایش سرعت یادگیری عکس ها را با تقسیم بر 225 نرمال می کنیم. مقیاس گذاری مقادیر پیکسل با ضریب 255/1 یک مرحله پیش پردازش رایج در کار با داده های تصویر است و برای نرمال سازی انجام می شود. علت این کار این است که :

- محدوده ارزش پیکسل : در تصاویر دیجیتال، مقادیر پیکسل معمولاً اعداد صحیحی از 0 تا 255 هستند. مقدار پیکسل 0 معمولاً سیاه و 255 نشان دهنده سفید است.
- نرمال سازی: شبکه‌های عصبی اغلب زمانی که داده‌های ورودی نرمال می‌شوند بهتر عمل می‌کنند و سریع‌تر همگرا می‌شوند. نرمال سازی شامل مقیاس بندی مقادیر پیکسل به یک محدوده استاندارد، معمولا بین 0 و 1 است.
- ضریب مقیاس 255/.1 : تقسیم هر پیکسل بر 255، مقادیر را به محدوده [0، 1] تغییر می دهد. به عنوان مثال، یک مقدار پیکسل 128 پس از مقیاس بندی به 128 / 255 = 0.50196 تبدیل می شود.
- مزایای نرمال سازی : نرمال سازی فرآیند بهینه سازی را در طول آموزش پایدارتر می کند. این به جلوگیری از مشکلاتی مانند ناپدید شدن یا گرادیان ها کمک می کند. این تضمین می کند که ویژگی ها مقیاس مشابهی دارند، که می تواند برای الگوریتم های بهینه سازی خاص مهم باشد.
- تاثیر بر آموزش شبکه عصبی: شبکه‌های عصبی اغلب از توابع فعال‌سازی استفاده می‌کنند که زمانی که مقادیر ورودی در محدوده خاصی قرار دارند به عنوان مثال، توابع sigmoid یا tanh مؤثرتر هستند. با نرمال سازی مقادیر پیکسل ورودی در محدوده [0، 1]، اطمینان حاصل می کنید که ورودی شبکه عصبی در محدوده مناسبی برای این توابع فعال سازی قرار می گیرد.

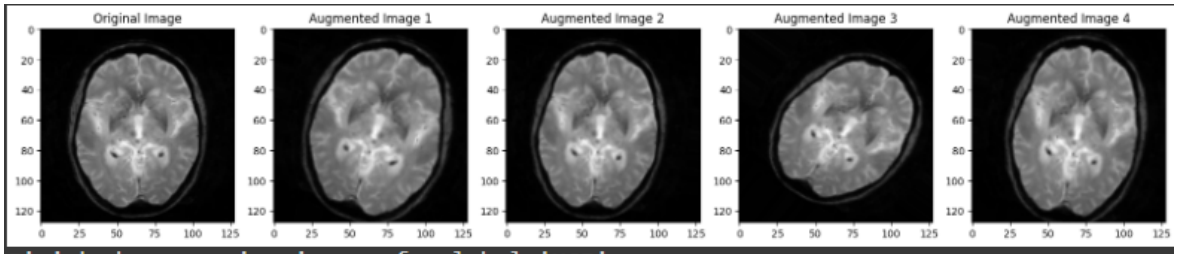
۱-۳. داده افزایی

در این مرحله عملیات داده افزایی را با استفاده از پارمتر های مقاله انجام می دهیم.این پارامتر ها عبارت‌اند از:

```
horizontal_flip=True, shear_range=0.2, height_shift_range=0.1,
rotation_range=45, zoom_range=[0.9, 1.1]
```

اعمال افزایش داده ها بر روی تصاویر اصلی و ایجاد 4 تصویر جدید افزوده شده از هر تصویر می تواند تأثیرات متعددی بر مجموعه داده ها و آموزش آنها داشته باشد.

- افزایش اندازه مجموعه داده : اندازه مجموعه داده به میزان 5 برابر افزایش می یابد (1 تصویر اصلی + 4 تصویر افزوده شده). این می تواند مفید باشد زیرا داشتن داده های بیشتر می تواند منجر به نتایج آموزشی بهتر شود.
- تنوع در داده : تصاویر افزوده شده تنوع در مجموعه داده را معرفی می کند. تغییراتی که تعریف کرده‌ایم (تغییر افقی، برش، تغییر ارتفاع، چرخش و بزرگ‌نمایی) تغییراتی از تصاویر اصلی ایجاد می‌کند که می‌تواند سناریوها یا دیدگاه‌های مختلف ممکن را نشان دهد. این می تواند به مدل کمک کند تا ویژگی های قوی تر و کلی تر را بیاموزد.
- کاهش overfitting: افزایش داده ها یک تکنیک منظم سازی است، به این معنی که می تواند به کاهش اضافه overfitting کمک کند. تطبیق بیش از حد زمانی اتفاق می‌افتد که یک مدل داده‌های آموزشی را خیلی خوب یاد می‌گیرد، تا جایی که در داده‌های دیده نشده (مانند اعتبارسنجی یا داده‌های آزمایشی) ضعیف عمل می‌کند. با ایجاد و استفاده از تصاویر افزوده شده، اساساً تنوع داده های آموزشی خود را افزایش می دهیم و به تعمیم بهتر مدل خود کمک می کنیم.
- نیازهای محاسباتی: با افزایش اندازه مجموعه داده ها، نیازهای محاسباتی برای آموزش مدل نیز افزایش می یابد. برای ذخیره تصاویر افزوده شده به حافظه بیشتر و قدرت پردازش بیشتری برای آموزش مدل روی آنها نیاز داریم.
- زمان آموزش: با افزایش حجم داده ها، زمان لازم برای آموزش مدل نیز افزایش می یابد.

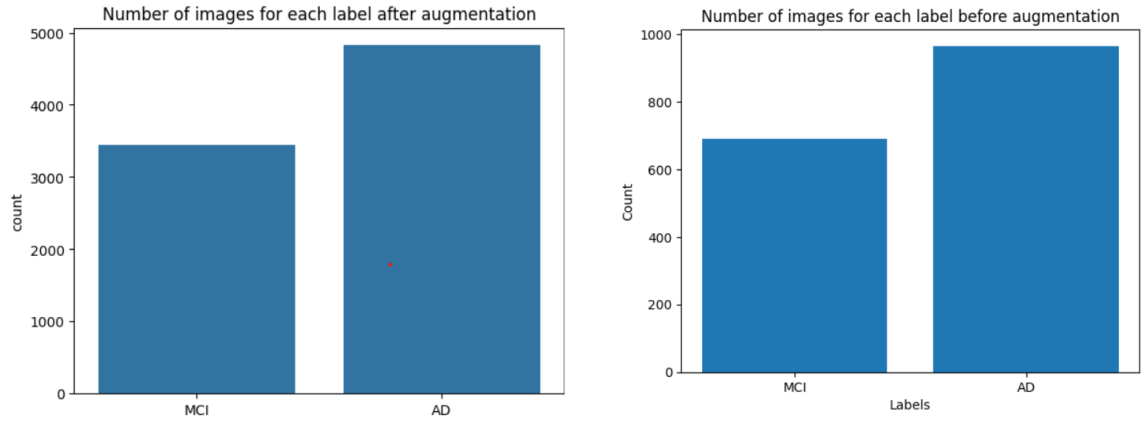



```
Finished processing images for label 'MCI'.
Number of images for label 'MCI': 3445
Finished processing images for label 'AD'.
Number of images for label 'AD': 4825
There are 8270 images in the 'images' variable.
There are 8270 image labels in the 'image_labels' variable.
There are 2 unique labels in the 'image_labels_encoded' variable.
The unique labels are: [0 1]
The counts of each label are: [4825 3445]
```

شکل 2. تعداد داده های هرکلاس بعد از داده افزایی

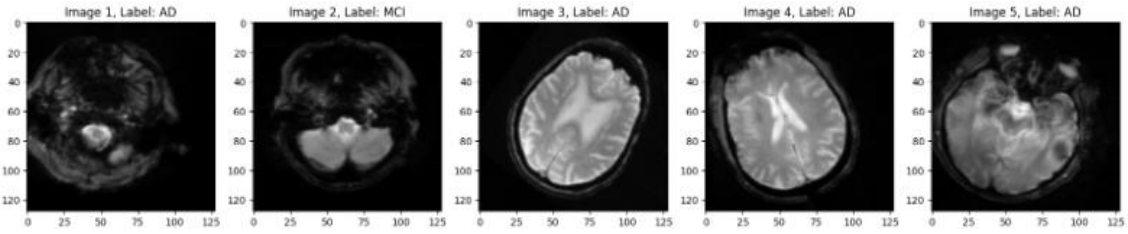
نمایش توزیع آماری کلاس ها قبل و بعد از اضافه شدن تصاویر را در پلات زیر می بینیم. که همان طور که از تعداد هر کلاس مشخص است می بینیم که تعداد هر کلاس قبل از داده افزایی برابر است با تعداد تصاویر در هر پوشه کلاس های MCI و AD (#MCI = 689 , #AD = 965)برابر است.همچنین پس از داده افزایی نیز می بینیم که تعداد داده ها برابر با ۵ (4_augmented +1_original = 5_total_images) برابر داده های هر کلاس است.

(#MCI = 689*5 = 3445, #AD = 965*5 = 4825) البته به علت استفاده از پلات هیستوگرام تعداد دقیق عکس ها مشخص نیست ولی با پلات هیستوگرام می توان به خوبی مشاهده کرد که نسبت تعداد عکس های هر کلاس ، به کل عکس ها قبل و بعد از داده افزایی ثابت است.



شکل 3. نسبت کلاس‌های داده اصلی و داده آگمنت شده

پنج تصویر را به صورت تصادفی از مجموعه دادگان جدید را نیز نمایش می دهیم و همان طور که انتظار داریم عکس های جدید با عکس های اصلی بر حسب پارامترهای داده افزایی مثل تغییر افقی، برش، تغییر ارتفاع، چرخش و بزرگ‌نمایی تفاوت دارند.



شکل 4. پنج عکس بعد از داده افزایی

۱-۴. پیاده سازی

Glorot initialization: مقداردهی اولیه Glorot که با نام اولیه Xavier نیز شناخته می شود، روشی است که برای مقداردهی اولیه وزن ها در شبکه عصبی استفاده می شود. این نام به افتخار Xavier Glorot گرفته شده است.

ایده اصلی پشت مقداردهی اولیه Glorot، حفظ واریانس در فعال‌سازی‌ها و گرادیان‌های منتشر شده در لایه‌های یک شبکه عصبی عمیق است. اگر وزن ها در یک شبکه خیلی کوچک شروع شوند، سیگنال با عبور از هر لایه منقبض می شود تا زمانی که برای مفید بودن خیلی کوچک باشد. اگر وزن‌ها در یک شبکه خیلی بزرگ شروع شوند، آنگاه سیگنال با عبور از هر لایه رشد می‌کند تا زمانی که بیش از حد عظیم باشد که مفید نباشد.

مقداردهی اولیه Glorot این مشکل را با مقیاس بندی واریانس وزن یک لایه بر اساس تعداد اتصالات ورودی و خروجی برطرف می کند. وزن ها به طور تصادفی مقداردهی اولیه می شوند، اما در محدوده خاصی که توسط تعداد نورون های ورودی و خروجی تعیین می شود.

Loss function: Categorical Crossentropy یک تابع ضرر است که در وظایف طبقه بندی چند کلاسه استفاده می شود. دلیل استفاده از آن در ارتباط با تابع softmax رای وظایف چند کلاسه این است که softmax بردار اعداد را به توزیع احتمال تبدیل می کند، که لازم است زیرا خروجی مدل ما یک توزیع احتمال بر روی کلاس ها است.

اگر از CategoricalCrossentropy به عنوان تابع ضرر خود استفاده کنیم، باید از تابع فعال سازی softmax در لایه خروجی خود استفاده کنیم. تابع softmax برای مسائل طبقه بندی چند کلاسه استفاده می شود که در آن هر نمونه می تواند تنها به یک کلاس تعلق داشته باشد. یک توزیع احتمال بر روی کلاس ها را خروجی می دهد، به این معنی که مقادیر خروجی در محدوده (0، 1) هستند و مجموع آنها 1 است.

از سوی دیگر، تابع sigmoid معمولا برای مسائل طبقه بندی باینری یا مسائل طبقه بندی چند برچسبی استفاده می شود که در آن هر نمونه می تواند به چندین کلاس تعلق داشته باشد. مقداری را در محدوده (0، 1) خروجی می دهد که نشان دهنده احتمال تعلق نمونه به کلاس مثبت است.

بنابراین، در مورد داده ما ، از آنجایی که با یک مشکل طبقه‌بندی چند کلاسه (با کلاس‌های ADو MCI سر و کار داریم، استفاده از softmax در لایه خروجی و CategoricalCrossentropy به‌عنوان تابع ضرر، رویکرد صحیحی است.

همچنین می بایستی اشاره کنم که من در ابتدا از learining rate = 0.1 که در مقاله استفاده شده بود استفاده کردم ولی دقت مدل به مراتب پایین بود ولی با قرار دادن learining rate = 0.001، دقت نهایی مدل ها به حدود 0.99 رسید.

پیاده سازی مدل های طبق مقاله :

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 126, 126, 32)      320

 batch_normalization_4 (Batc  (None, 126, 126, 32)     128
 hNormalization)

 max_pooling2d_2 (MaxPooling  (None, 63, 63, 32)       0
 2D)

 conv2d_5 (Conv2D)           (None, 61, 61, 32)        9248

 batch_normalization_5 (Batc  (None, 61, 61, 32)       128
 hNormalization)

 max_pooling2d_3 (MaxPooling  (None, 30, 30, 32)       0
 2D)

 flatten_1 (Flatten)         (None, 28800)             0

 dense_3 (Dense)             (None, 128)               3686528

 dense_4 (Dense)             (None, 2)                 258

=================================================================
Total params: 3,696,610
Trainable params: 3,696,482
Non-trainable params: 128
```

شکل 5. خلاصه مدل **teset_1**

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 126, 126, 32)      320

 batch_normalization_6 (Batc  (None, 126, 126, 32)     128
 hNormalization)

 conv2d_7 (Conv2D)           (None, 124, 124, 32)      9248

 batch_normalization_7 (Batc  (None, 124, 124, 32)     128
 hNormalization)

 max_pooling2d_4 (MaxPooling  (None, 62, 62, 32)       0
 2D)

 flatten_2 (Flatten)         (None, 123008)            0

 dense_5 (Dense)             (None, 128)               15745152

 dense_6 (Dense)             (None, 64)                8256

 dense_7 (Dense)             (None, 2)                 130

=================================================================
Total params: 15,763,362
Trainable params: 15,763,234
Non-trainable params: 128
```

شکل 6. خلاصه مدل **teset_2**

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 32)      320

 batch_normalization (BatchN  (None, 126, 126, 32)     128
 ormalization)

 conv2d_1 (Conv2D)           (None, 124, 124, 32)      9248

 batch_normalization_1 (Batc  (None, 124, 124, 32)     128
 hNormalization)

 max_pooling2d (MaxPooling2D  (None, 62, 62, 32)       0
 )

 conv2d_2 (Conv2D)           (None, 60, 60, 32)        9248

 batch_normalization_2 (Batc  (None, 60, 60, 32)       128
 hNormalization)

 conv2d_3 (Conv2D)           (None, 58, 58, 32)        9248

 batch_normalization_3 (Batc  (None, 58, 58, 32)       128
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 29, 29, 32)       0
 2D)
```

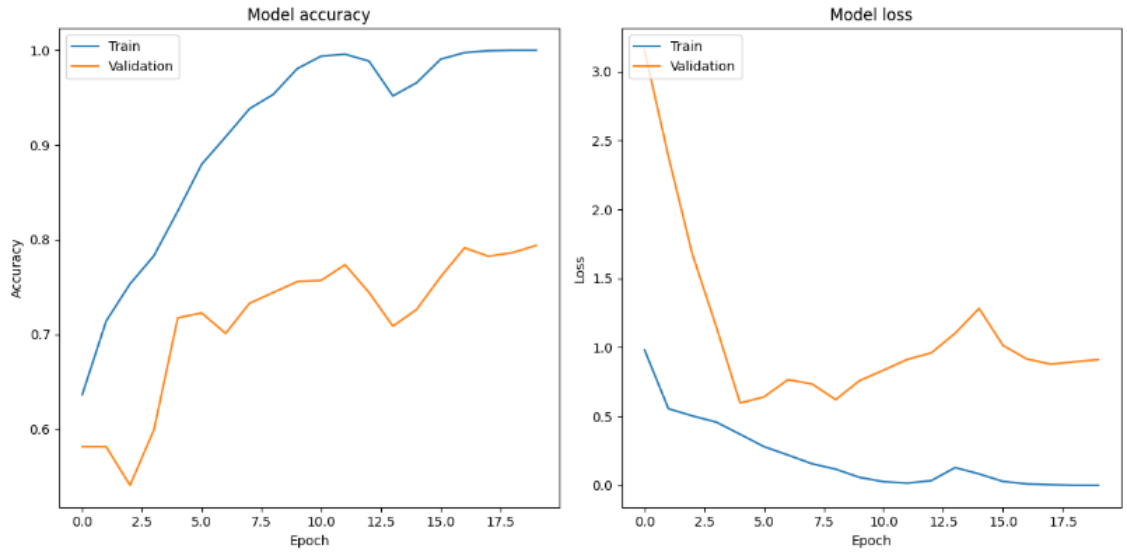
```
flatten (Flatten)            (None, 26912)             0

dense (Dense)                (None, 128)               3444864

dense_1 (Dense)              (None, 64)                8256

dense_2 (Dense)              (None, 2)                 130

=================================================================
Total params: 3,481,826
Trainable params: 3,481,570
Non-trainable params: 256
```

شکل 7. خلاصه مدل **proposed**

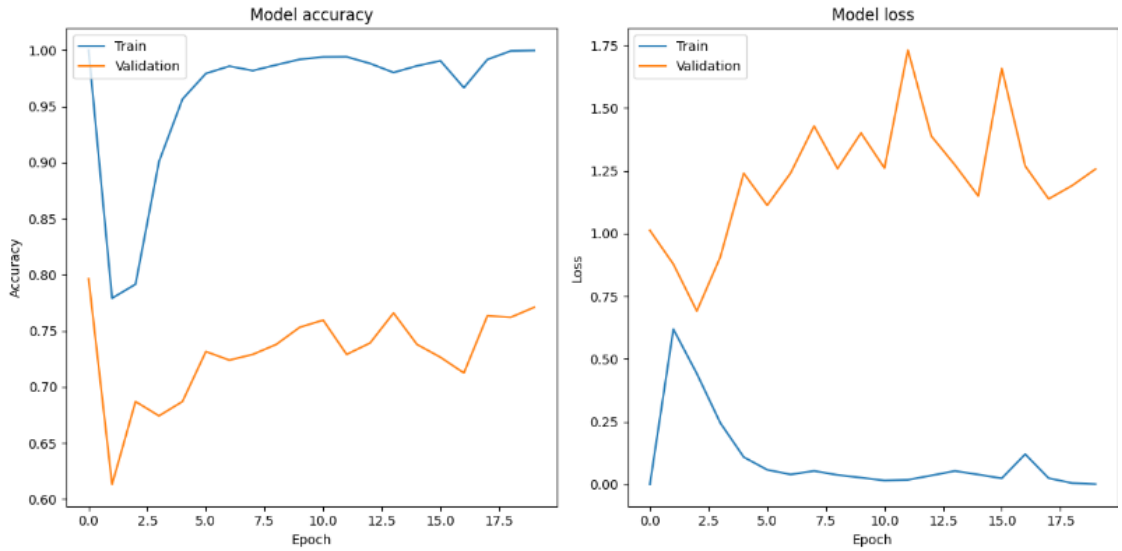
۱-۵. تحلیل نتایج

بر اساس تصویری که می در پایین می بینیم ، به نظر می رسد که مدل واقعاً overfit برایش رخ داده است . تطبیق بیش از حد زمانی اتفاق می‌افتد که یک مدل داده‌های آموزشی را خیلی خوب یاد می‌گیرد و علاوه بر الگوهای زیربنایی، نویز و نقاط پرت را نیز ثبت می‌کند. وقتی این اتفاق می‌افتد، مدل روی داده‌های آموزشی خوب عمل می‌کند اما روی داده‌های دیده نشده (مانند اعتبارسنجی یا داده‌های آزمایشی) ضعیف عمل می‌کند. این با واگرایی بین معیارهای آموزشی و اعتبارسنجی در نمودارهایمان نشان داده شده است.

همچنین می بینیم که نمودار loss validation و اساسا عملکرد مدل به batch size هم به شدت وابسته است. batch size تعداد نمونه هایی است که به طور همزمان از طریق شبکه منتشر می شوند. اگر مدل مان به batch size حساس است، ممکن است به این معنی باشد که مدل ناپایدار است. batch size کوچکتر گاهی اوقات می‌توانند به دلیل نویز در تخمین‌های گرادیان منجر به overfit شوند.



شکل 8. اجرای مدل به ازای **batch_size = 64**



شکل 9. اجرای مدل به ازای **batch_size = 32**

ROC curve: منحنی ROC (Receiver Operating Characteristic) یک نمایش گرافیکی است که عملکرد یک طبقه بندی کننده باینری را به عنوان آستانه تمایز آن نشان می دهد. این با ترسیم نرخ مثبت واقعی TPR در برابر نرخ مثبت کاذب FPR در تنظیمات آستانه های مختلف ایجاد می شود.

نرخ مثبت واقعی TPR، همچنین به عنوان حساسیت یا یادآوری شناخته می شود، نسبت مشاهدات مثبت واقعی (در این مورد، پیش بینی های صحیح یک کلاس توسط مدل مان) است که به درستی به عنوان چنین شناسایی شده اند.

نرخ مثبت کاذب FPRنسبت مشاهدات منفی واقعی است که به اشتباه به عنوان مثبت شناسایی شده اند.

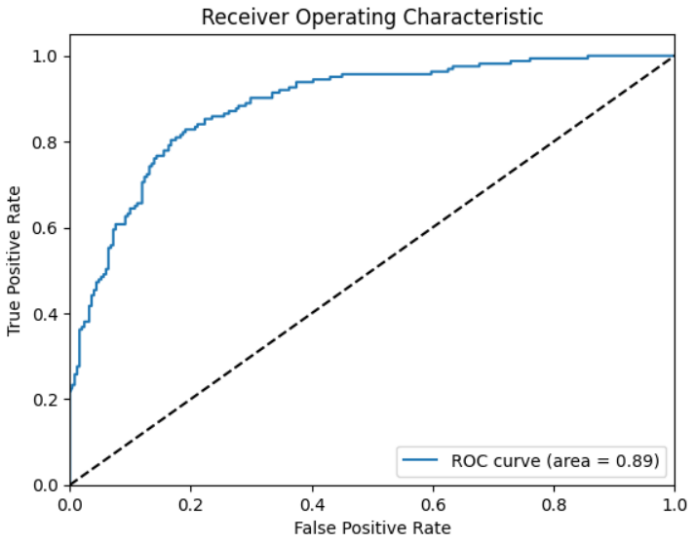
AUC مساحت زیر منحنی ناحیه زیر منحنی ROC است. این مقدار اسکالر ، عملکرد کلی طبقه‌بندی‌کننده باینری را خلاصه می‌کند. AUC از 0 تا 1 است که در آن:

AUC = 1 یک طبقه بندی کننده کامل را نشان می دهد (به خوبی می تواند بین تمام موارد مثبت و منفی تمایز قائل شود).

AUC = 0.5 نشان می دهد که طبقه بندی کننده بهتر از حدس زدن تصادفی نیست.

AUC کمتر از 0.5 نشان دهنده عملکرد بدتر طبقه بندی کننده نسبت به حدس زدن تصادفی است.

در مورد مدل مان ، AUC 0.89 نشان می دهد که مدل شما عملکرد نسبتا خوبی دارد. به این معنی که 89٪ احتمال دارد که مدل بتواند بین طبقه مثبت و طبقه منفی تمایز قائل شود.



شکل 10. **ROC curve** مدل **proposed**

- Accuracy: نسبت نمونه های به درستی پیش بینی شده به کل نمونه ها در مجموعه داده است. تعداد دفعات صحیح بودن مدل را اندازه می‌گیرد.

- Percission: دقت نسبت مشاهدات مثبت پیش بینی شده صحیح به کل مثبت های پیش بینی شده است. این اندازه گیری می کند که چه تعداد از نمونه های مثبت پیش بینی شده در واقع مثبت هستند.
- Recall(Sensivity): یادآوری نسبت مشاهدات مثبت پیش بینی شده صحیح به همه موارد مثبت واقعی است. با پیش‌بینی مثبت، تعداد نمونه‌های مثبت واقعی را اندازه‌گیری می‌کند.

- F1_Score: امتیاز F1 معیاری است که هم دقت و هم یادآوری را در یک معیار واحد ترکیب می‌کند، آنها را متعادل می‌کند و دید جامع‌تری از عملکرد مدل ارائه می‌کند.

- AUC: AUC به ناحیه زیر منحنی ROC اشاره دارد. AUC بالاتر نشان‌دهنده عملکرد بهتر مدل است، با 1 که پیش‌بینی کامل و 0.5 شانس تصادفی است.

- Confusion Matrix: این ماتریس نتایج طبقه بندی را برای هر کلاس در مجموعه داده شما نشان می دهد که مثبت درست، مثبت کاذب، منفی درست و منفی کاذب را نشان می دهد.

حال، نتایج مدل و مجموعه داده مان را برای این پارامترها تجزیه و تحلیل می کنیم:

Confusion Matrix مان [[110 53] ,[30 221]] را نشان می دهد، که نشان می دهد 221 مثبت درست، 30 مثبت کاذب، 53 منفی کاذب، و 110 منفی درست وجود دارد.

- accuracy برای AD بالا در 0.81 است به این معنی که وقتی AD را پیش‌بینی می‌کند معمولاً درست است در حالی که MCI دقت کمی پایین‌تر در 0.79 دارد.

- recall برای AD نیز بالا است و 0.88 است، به این معنی که بیشتر موارد واقعی AD را ثبت می کند، اما برای MCI کمتر است و تنها حدود دو سوم (0.67) موارد واقعی را ثبت می کند.

- امتیاز F1 که دقت و یادآوری را متعادل می‌کند، عملکرد خوبی را در AD نشان می‌دهد اما در MCI به دلیل ارزش یادآوری پایین‌تر آن، کمی کمتر است.

- accuracy کلی برابر با 0.80 است که نشان‌دهنده قابلیت پیش‌بینی کلی خوب در هر دو کلاس است.

```
13/13 [==============================] - 0s 10ms/step
13/13 [==============================] - 0s 8ms/step
Confusion Matrix:
[[216  35]
 [ 40 123]]
Classification Report:
              precision    recall  f1-score   support

          AD       0.84      0.86      0.85       251
         MCI       0.78      0.75      0.77       163

    accuracy                           0.82       414
   macro avg       0.81      0.81      0.81       414
weighted avg       0.82      0.82      0.82       414

AUC:  0.8859347884535478
```

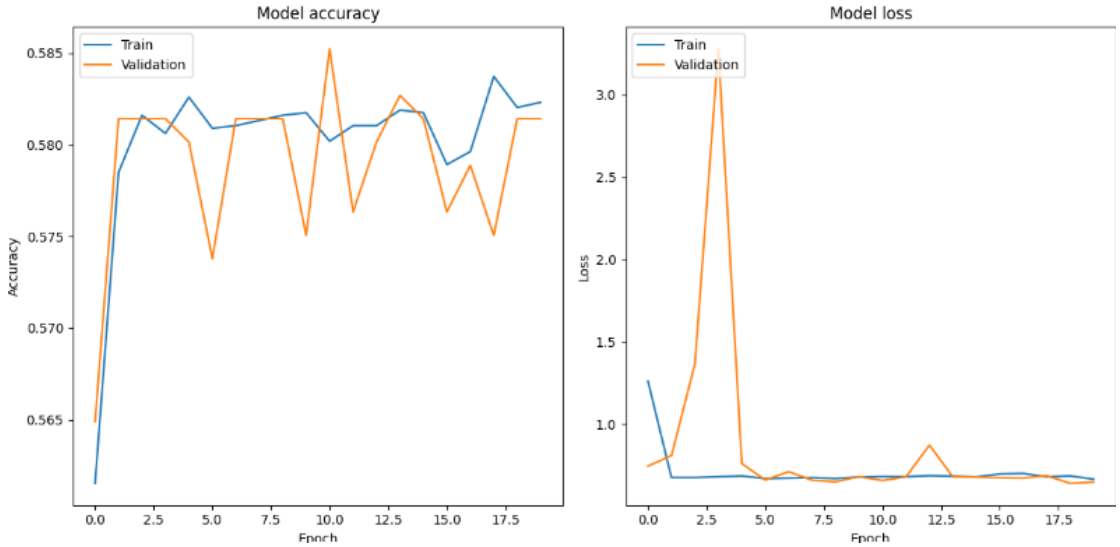
شکل 11. تحلیل مدل **proposed** بدون **dropout**

۱-۶. مقایسه نتایج

Dropout: یک تکنیک regularization برای کاهش overfitting در شبکه های عصبی است. با تنظیم تصادفی کسری از واحدهای ورودی در هر به‌روزرسانی در طول زمان آموزش کار می‌کند، که به جلوگیری از overfitting کمک می‌کند.

با استفاده از لایه‌های dropout، اساساً نویز را به شبکه خود وارد می‌کنیم که می‌تواند به جلوگیری از تناسب بیش از حد مدل با داده‌های آموزشی (بیش از حد) کمک کند. این می تواند به بهبود توانایی مدل برای تعمیم به داده های دیده نشده کمک کند.

و همین طور که در مدل با dropout میبینیم که loss validation دیگر صعودی نبوده و تا حد خوبی توانسته ایم که از overfitting جلوگیری کنیم.



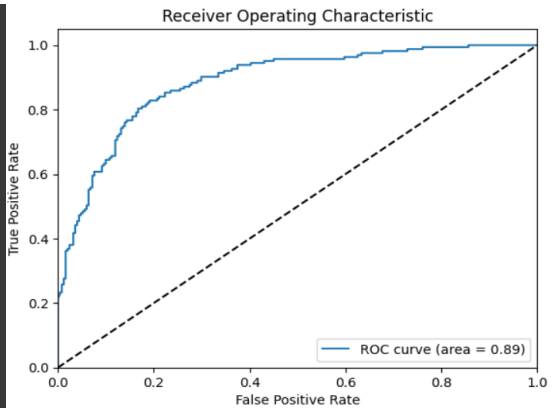
شکل 12. تحلیل **loss** بعد از اعمال لایه های **dropout**

مقایسه بین معماری مدل های model Testing 1 و model Testing 2 و proposed model:

با مقایسه ROC و AUC و F1_score که معیاری است که در خود هم recall و precision را بررسی می کند، متوجه می شویم که مدل proposed بهتر test_2 و test_2 هم از test_1 بهتر است :

نتایج مدل proposed: Roc curve = 0.89 و F1_score_AD = 0.85 و F1_score_MCI = 0.77

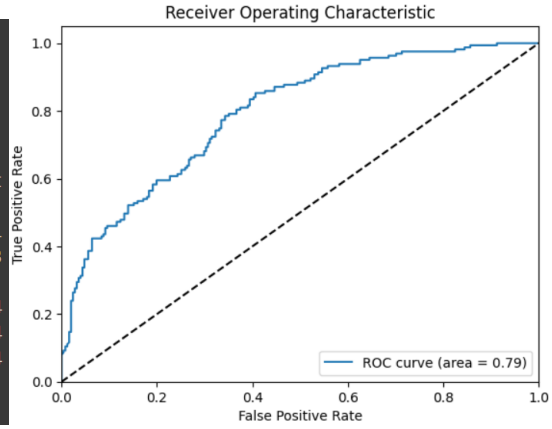
```
13/13 [==============================] - 0s 10ms/step
13/13 [==============================] - 0s 8ms/step
Confusion Matrix:
[[216  35]
 [ 40 123]]
Classification Report:
              precision    recall  f1-score   support

          AD       0.84      0.86      0.85       251
         MCI       0.78      0.75      0.77       163

    accuracy                           0.82       414
   macro avg       0.81      0.81      0.81       414
weighted avg       0.82      0.82      0.82       414

AUC:  0.8859347884535478
```



شکل 13. تحلیل loss مدل **proposed** بعد از اعمال لایه های **dropout**

نتایج مدل test-2: Roc curve = 0.79 و F1_score_AD = 0.76 و F1_score_MCI = 0.61

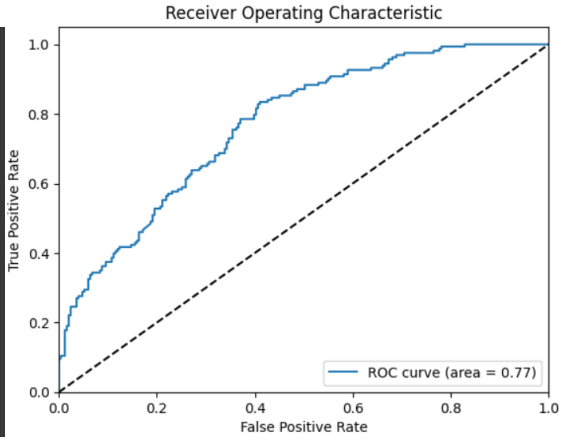
```
13/13 [==============================] - 0s 8ms/step
13/13 [==============================] - 0s 8ms/step
Confusion Matrix:
[[194  57]
 [ 66  97]]
Classification Report:
              precision    recall  f1-score   support

          AD       0.75      0.77      0.76       251
         MCI       0.63      0.60      0.61       163

    accuracy                           0.70       414
   macro avg       0.69      0.68      0.69       414
weighted avg       0.70      0.70      0.70       414

AUC:  0.792633148388043
```



شکل 14. تحلیل loss مدل **test_2** بعد از اعمال لایه های **dropout**

نتایج مدل test-1: Roc curve = 0.77 و F1_score_AD = 0.75 و F1_score_MCI = 0.59

```
13/13 [==============================] - 0s 5ms/step
13/13 [==============================] - 0s 5ms/step
Confusion Matrix:
[[190  61]
 [ 69  94]]
Classification Report:
              precision    recall  f1-score   support

          AD       0.73      0.76      0.75       251
         MCI       0.61      0.58      0.59       163

    accuracy                           0.69       414
   macro avg       0.67      0.67      0.67       414
weighted avg       0.68      0.69      0.68       414

AUC:  0.7677327499816684
```



شکل 15. تحلیل **loss** مدل **test_1** بعد از اعمال لایه های **dropout**

اثر نسبت های مختلف تقسیم بندی داده ها با در نظر گرفتن نسبتهای ۰.۳ و ۰.۵ ضمن اطمینان از توزیع کلاس ها:

تقسیم بندی داده ها، که اغلب به عنوان train-test split از آن یاد می شود، نقش مهمی در عملکرد یک مدل شبکه عصبی ایفا می کند. در اینجا تحلیلی از نسبت‌های مختلفی آورده شده است:

- Train-Test Split = 0.95-0.05: در این سناریو، 95 درصد از داده ها برای آموزش و تنها 5 درصد برای آزمایش استفاده می شود. در حالی که این مقدار زیادی داده را برای مدل فراهم می کند تا از آن یاد بگیرد، داده های بسیار کمی برای آزمایش باقی می گذارد. این می تواند منجر به overfitting شود، جایی که مدل در داده های آموزشی عملکرد خوبی دارد اما در داده های دیده نشده ضعیف است. همچنین، معیارهای ارزیابی به‌دست‌آمده از مجموعه آزمون ممکن است به دلیل اندازه کوچک مجموعه آزمون قابل اعتماد نباشد.

- Train-Test Split = 0.5-0.5 or 0.7-0.3 : این تقسیمات متعادل تر هستند. آنها حجم خوبی از داده ها را هم برای آموزش مدل و هم برای ارزیابی عملکرد آن فراهم می کنند. تقسیم 70-30 اغلب مورد استفاده قرار می گیرد، زیرا تعادل خوبی بین آموزش و تست ایجاد می کند.

- تقسیم آزمون Train-Test با اندازه آزمون بزرگتر: اگر اندازه داده های آزمون افزایش یابد و اندازه داده های آموزشی کاهش یابد، دقت مدل می تواند به میزان قابل توجهی کاهش یابد. این به این دلیل است که مدل داده‌های کمتری برای یادگیری در طول مرحله آموزش دارد، که می‌تواند توانایی آن را برای تعمیم خوب به داده‌های دیده نشده محدود کند. مدل ممکن است الگوهای زیربنایی در داده‌ها را به‌طور مؤثری ثبت نکند، که منجر به عملکرد ضعیف‌تر در مجموعه آزمایشی می‌شود.

از نظر تأثیر بر recall یا هر معیار دیگر، مدلی که در یک مجموعه تمرینی بزرگ‌تر (مانند تقسیم 95-5) آموزش داده شده است، ممکن است recall بالایی را در مجموعه تمرینی نشان دهد، اما اگر بیش از حد مناسب باشد، recall ضعیفی را در مجموعه آزمایشی نشان دهد. از سوی دیگر، مدلی که با تقسیم متعادل‌تری (مانند 50-50 یا 70-30) آموزش داده شده است، احتمالاً مقادیر recall قابل اعتمادتر و ثابت‌تری را بین مجموعه‌های آموزشی و آزمایشی نشان می‌دهد.

همچنین شایان ذکر است که استفاده از تکنیک‌هایی مانند cross-validation می‌تواند تخمین قوی‌تری از عملکرد مدل ارائه دهد، زیرا وابستگی به نحوه تقسیم داده‌ها به مجموعه‌های آموزشی و آزمایشی را کاهش می‌دهد. این می تواند به ویژه در هنگام برخورد با مجموعه داده های کوچکتر (مانند مجموعه داده های ما که شامل کمتر از 2000 تصویر هستند) مفید باشد.

```
13/13 [==============================] - 0s 10ms/step
13/13 [==============================] - 0s 8ms/step
Confusion Matrix:
[[216  35]
 [ 40 123]]
Classification Report:
              precision    recall  f1-score   support

          AD       0.84      0.86      0.85       251
         MCI       0.78      0.75      0.77       163

    accuracy                           0.82       414
   macro avg       0.81      0.81      0.81       414
weighted avg       0.82      0.82      0.82       414

AUC:  0.8859347884535478
```

شکل 16. تحلیل مدل **proposed** بعد از اعمال لایه های **dropout** با **(test) split size = 0.05**

```
78/78 [==============================] - 1s 8ms/step
78/78 [==============================] - 1s 7ms/step
Confusion Matrix:
[[1091  320]
 [ 717  353]]
Classification Report:
              precision    recall  f1-score   support

          AD       0.60      0.77      0.68      1411
         MCI       0.52      0.33      0.41      1070

    accuracy                           0.58      2481
   macro avg       0.56      0.55      0.54      2481
weighted avg       0.57      0.58      0.56      2481

AUC:  0.5711374911410347
```

شکل 17. تحلیل مدل **proposed** بعد از اعمال لایه های **dropout** با **(test)** **split size = 0.3**

```
130/130 [==============================] - 1s 8ms/step
130/130 [==============================] - 1s 7ms/step
Confusion Matrix:
[[1828  551]
 [1192  564]]
Classification Report:
              precision    recall  f1-score   support

          AD       0.61      0.77      0.68      2379
         MCI       0.51      0.32      0.39      1756

    accuracy                           0.58      4135
   macro avg       0.56      0.54      0.54      4135
weighted avg       0.56      0.58      0.56      4135

AUC:  0.5806769153211327
```

شکل 18. تحلیل مدل **proposed** بعد از اعمال لایه های **dropout** با **(test)** **split size = 0.5**

اثر Glorot initializer: یک تکنیک مقداردهی اولیه وزن است که می تواند به طور قابل توجهی بر عملکرد یک مدل یادگیری عمیق تأثیر بگذارد.از جمله مزایای ان می توان به موارد زیر اشاره کرد :

- کاهش Vanishing/Exploding Gradients: آغازگر Glorot به کاهش مشکل گرادیان های ناپدید و انفجار کمک می کند، که در شبکه های عمیق با توابع فعال سازی sigmoid یا tanh رایج است. این مشکل زمانی رخ می دهد که شیب ها خیلی کوچک (ناپدید می شوند) یا خیلی بزرگ (منفجر می شوند) و آموزش شبکه را سخت می کند. آغازگر Glorot تضمین می کند که واریانس خروجی هر لایه با واریانس ورودی آن برابر است و گرادیان ها قبل و بعد از عبور از یک لایه در جهت معکوس واریانس یکسانی دارند.

- آموزش کارآمد را ترویج می کند: آغازگر Glorot آموزش پایدار و کارآمد را تسهیل می کند و منجر به همگرایی بهتر می شود. این به حفظ انحراف استاندارد فعال‌سازی لایه‌ها در حدود 1 کمک می‌کند و به ما امکان می‌دهد چندین لایه دیگر را در یک شبکه عصبی عمیق بدون انفجار یا ناپدید شدن گرادیان‌ها قرار دهیم.

- تقارن را می شکند: راه اندازی کننده Glorot تقارن بین واحدهای یک لایه را می شکند، این مهم است زیرا اگر همه وزن ها به یک مقدار مقداردهی اولیه شوند، همه واحدهای یک لایه در طول آموزش ویژگی های یکسانی را یاد خواهند گرفت.

- تعمیم‌سازی را بهبود می‌بخشد: مطالعات نشان داده‌اند که مدل‌هایی که با مقداردهی اولیه Glorot راه‌اندازی می‌شوند، اغلب بهتر از مدل‌هایی که با سایر اولیه‌سازها تعمیم می‌یابند.

پرسش ۲. بررسی تاثیر افزایش داده بر عملکرد شبکه‌های کانولوشنی Fine-Tune شده

۲-۱. معرفی مقاله

تا سال‌های اخیر، یادگیری عمیق نتایج قابل توجهی را در زمینه یادگیری ماشین در وظایف Computer vision به دست آورد.

در میان معماری‌های مختلف، معماری‌های مبتنی بر شبکه‌های عصبی عمیق که به عنوان شبکه‌های عصبی کانولوشنال شناخته می‌شوند، اخیراً به طور گسترده برای تشخیص و طبقه‌بندی تصاویر استفاده می‌شوند.

اگرچه این ابزار بسیار مناسبی برای وظایف بینایی کامپیوتری است، اما نیازمند حجم زیادی از داده‌های آموزشی برای دست‌یابی به عملکرد بالا است. در این مقاله، روش افزایش داده پیشنهاد شده است تا چالش‌های مواجهه با کمبود داده‌های آموزشی را برطرف سازد.

برای تحلیل اثر افزایش داده، روش پیشنهادی از دو معماری شبکه‌ی عصبی کانولوشنال استفاده می‌کند. برای کاهش زمان آموزش بدون کاستن از دقت، مدل‌ها با استفاده از تنظیم دقیق شبکه‌های پیش‌آموزش دیده شده VGG16 و ResNet50 ساخته شده‌اند.

برای ارزیابی عملکرد مدل‌ها، تابع‌های خطای و دقت استفاده می‌شوند. مدل‌های پیشنهادی با استفاده از چارچوب یادگیری عمیق Keras ساخته شده‌اند و مدل‌ها روی مجموعه داده‌ی سفارشی ایجاد شده از پایگاه داده CAT vs DOG در Kaggle آموزش داده شده‌اند. نتایج آزمایشی نشان داد که هر دو مدل هنگام استفاده از افزایش داده، دقت آزمون بهتری دارند.

۲-۲. پیش پردازش تصاویر

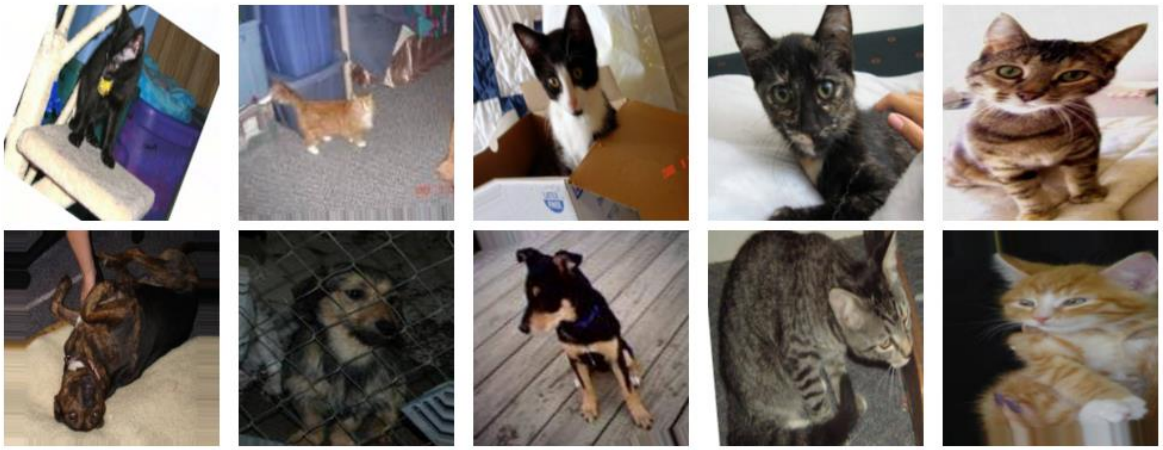
در این بخش، ابتدا تصاویر آموزش و ارزیابی را لود می‌کنیم. در این بخش به محض لود کردن دیتا، همزمان آن‌ها را نرمالایز (با تقسیم هر پیکسل بر ۲۵۵ این نرمالایزیشن اتفاق می‌افتد چرا که در RGB، هر پیکسل مقداری بین ۰ و ۲۵۵ دارد). در این بین چون عکس‌های ما اندازه‌های متفاوتی دارند، چون برای آموزش مدل نیاز است که همه ورودی‌ها، یک سایز داشته باشند، آن‌ها را به سایز (224, 224, 3) می‌بریم.

حال همه آن‌ها را در یک pandas DataFrame ذخیره می‌کنیم. سپس خروجی‌های مدنظر مدل را تعریف می‌کنیم، به این صورت که گربه‌ها مقدار ۰ و سگ‌ها مقدار ۱ داشته باشند.

سپس مانند مقاله، مجموعه داده‌ای که برای آموزش اختصاص یافته، را به نسبت ۷۰ به ۳۰ برای داده‌های آموزش و اعتبارسنجی به ترتیب تقسیم می‌کنیم. برای این‌کار از تابع train_test_split از کتابخانه sklearn بخش model_selection استفاده می‌کنیم.

برای بخش افزایش داده‌ها (augmented)، از ImageDataGenerator استفاده می‌کنیم. طبق مقایه سه روش برای افزایش داده در نظر می‌گیریم. به این صورت که هر کدام با یک احتمال تصادفی اتفاق می‌افتد:

1. Horizontal flip: به این صورت که به صورت تصادفی، تصاویر به صورت افقی معکوس میکند. برای انجام این بخش، horizontal_flip را برابر True قرار می‌دهیم.
2. Random rotation: به صورت تصادفی حداکثر به میزان ۳۰ درجه به صورت ساعتگرد و پادساعتگرد می‌چرخانیم. برای انجام این کار، مقدار rotation_range را برابر ۳۰ قرار می‌دهیم.
3. Scale: در این روش به میزان ۰.۷۵ تا ۱.۲۵ عکس را به ترتیب کوچک و یا بزرگ می‌کنیم. برای این کار، zoom_range را بازه [0.75, 1.25] قرار می‌دهیم.
4. همچنین در استفاده از این تابع، fill_mode را برابر ’nearest‘ قرار داده تا در صورت از دست رفتن قسمتی از تصویر، با پیکسلی مشابه پیکسل‌های اطراف آن، جایگزین شود.



شکل 19. نمونه تصاویر تولید شده برای افزایش داده‌ها

در حالی که تعداد داده‌های اصلی، برابر ۴۹۱ عدد (طبق جداسازی ۷۰، ۳۰ برای داده‌های آموزش و اعتبار سنجی در مقاله) است، تعداد داده‌های تولید شده به این روش، مانند محاسباتی که در بخش ۵.۲ مقاله انجام شده، حدود ۲۴۵۵۰۰ عدد در طول فرآیند آموزش برای هر مدل که از دیتای آگمنت شده استفاده می‌کند، خواهد بود. به این صورت که ما در کل به اندازه:

$$M = Batch\_Size \times Iterations \times N$$

داده برای آموزش خواهیم داشت. به این صورت که M تعداد کل دیتا مختلف تولید شده در طول فرآیند آموزش و N تعداد دیتا اولیه است. همچنین مانند پارامترهای تعریف شده در مقاله، *batch_size* برابر ۱۰ و *Iterations* (*num_epochs*) برابر ۵۰ در کل خواهد بود (۲۵ برای حالت freezed و ۲۵ ایپاک برای حالت unfreezed).

بنابراین افزایش داده‌ها به طور صحیح انجام شده. یکی از خوبی‌های استفاده از ImageDataGenerator این است که نیازی به ذخیره دیتا تولید شده نبوده و می‌توان همان لحظه دیتا مورد نیاز را تولید کرد.

از طرفی یکی از مشکلات آن این است که تقریبا با یک احتمال بالایی، هر داده تنها یک بار توسط مدل دیده شده و تقریبا در هر دوره آموزش، داده‌ها متفاوت خواهند بود.

	Train	*Test*	*Validation*
Augmented Data	*245500*	*100*	*219*
Original Data	*491*	*100*	*219*

جدول 1. تعداد داده‌های استفاده شده برای آموزش و ارزیابی مدل بعد و قبل آگمنت

۲-۳. پیاده‌سازی

VGG16

در ابتدا backbone مدل را که در اینجا از vgg16 استفاده می‌کنیم، لود می‌کنیم. این کار به روش زیر انجام می‌شود:

```
vgg16_base = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
```

در اینجا، وزن‌های شبکه vgg16 را که با دیتا معروف imagenet آموزش دیده‌اند، لود می‌کنیم. همچنین با استفاده از دستور include_top برابر False، قسمت fully connected انتهایی این شبکه را لود نمی‌کنیم، چرا که می‌خواهیم لایه fully connected مناسب برای داده‌ها و تسک خود را به آن اضافه کنیم. لایه‌های مناسب را به شبکه به صورت زیر اضافه کرده و در نهایت آن را با همین ساختار، آموزش می‌دهیم.

- ابتدا لایه Flatten را به مدل اضافه می‌کنیم تا خروجی چند بعدی از شبکه vgg16، را به صورت یک وکتور، داشته باشیم.
- سپس یک لایه Batch Normalization خواهیم داشت تا نورون ها مقادیری در یک scale داشته باشند.
- حال اولین لایه Dense خود را که ۵۱۲ نورون دارد، اضافه می‌کنیم. activation function این لایه، ReLU است.
- حال یک لایه DropOut با نرخ ۲۰ درصد داریم تا شبکه به صورت general تر آموزش ببیند و متکی به تعدادی از نورون‌های خاص نباشد.
- سپس دوباره یک لایه Dense با ۵۱۲ نورون اضافه می‌کنیم. activation function این لایه نیز ReLU است.
- دوباره یک لایه DropOut استفاده می‌کنیم. این لایه نیز با نرخ ۲۰ درصد، نورون‌ها را به صورت رندوم خاموش می‌کند.
- یک لایه Batch Normalization خواهیم داشت تا نورون‌ها نرمالایز شوند.
- در نهایت برای هندل کردن مسئله classification خود، چون که target دیتا به صورت ۰ و ۱ دخیره شده، در این لایه تنها یک نورون خواهیم داشت که activation function آن، sigmoid خواهد بود.

آموزش مدل با استفاده از داده آگمنت شده.

1 . برای آموزش مدل، ابتدا backbone مدل که vgg16 است را **freeze** می‌کنیم. در واقع با اینکار اجازه ترین شدن لایه‌های کانولوشنی vgg16 را در مدل تعریف شده نمی‌دهیم تا تنها در این مرحله، لایه‌های fully connected تازه اضافه شده ترین شوند. این کار را به روش زیر انجام می‌دهیم:

```
model.trainable = False
```

2. بار بررسی خلاصه مدل، تعداد پارامترها در کل **27,925,825** و پارامترهایی که قابلیت یادگیری در این مرحله را دارند، به تعداد **13,159,937** است.

3. حال قبل از آموزش مدل، ابتدا آن را compile می‌کنیم. در اینجا برخی از پارامترهای مربوط به بخش آموزش مدل را تعریف می‌کنیم که در آموزش شبکه و بروزرسانی وزن‌های شبکه دخیل هستند.

a. Optimizer: برای بروزرسانی نرخ یادگیری، از SGD Optimizer استفاده می‌کنیم. همچنین پارامترهای آن را مانند مقاله ست می‌کنیم.

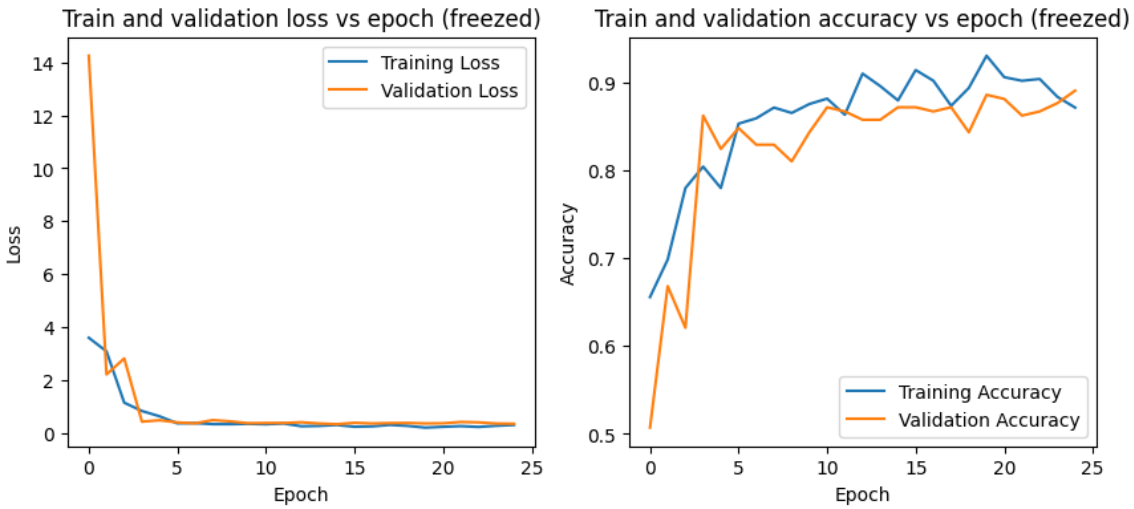
i. Learning rate = 0.1
ii. Momentum = 0.9
iii. Learning rate decay = 0.002

b. Loss function: برای تابع loss، binary cross entropy را قرار می‌دهیم، چرا که مقادیر خروجی ما به صورت باینری هستند.

4. سپس مدل را با استفاده از هایپر پارامترهای زیر مطابق با مقاله آموزش می‌دهیم.

a. Train Data = Augmented Data
b. Epoch = number of epochs (25 here)
c. Batch Size = batch_size (10 here)
d. Validation Data = splited data from train set

در این بخش از داده‌های آگمنت شده با استفاده از ImageDataGenerator، برای آموزش مدل استفاده می‌کنیم. می‌توانیم نتایج بدست آمده از آموزش را در نمودارهای موجود در شکل زیر بررسی کنیم.



شکل 20. نمودار دقت و **loss** برای حالت **freezed** مدل **vgg16**

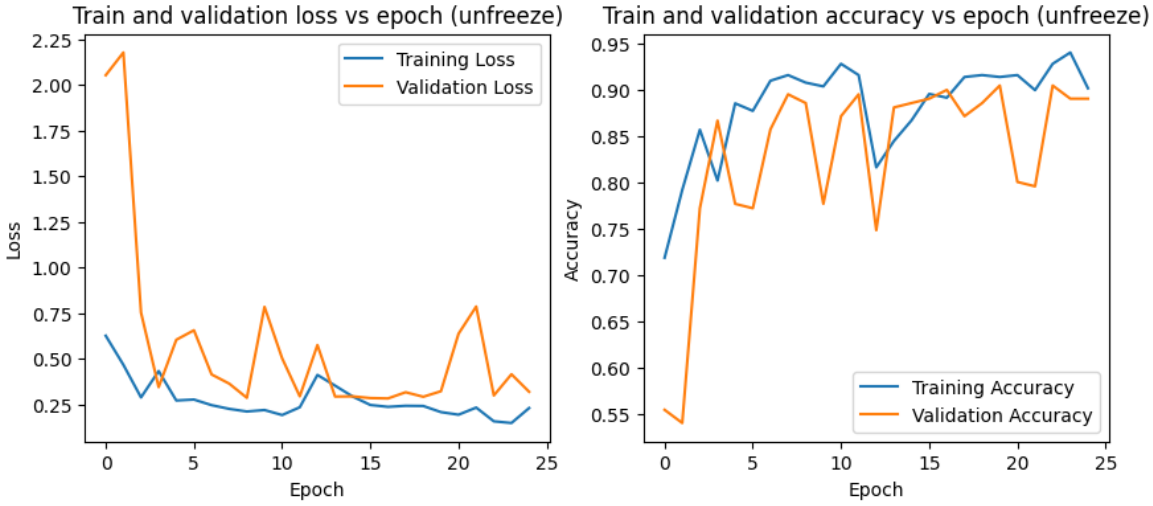
با توجه به نمودارهای بدست آمده، متوجه فرآیند یادگیری مدل می‌شویم. این مدل با ۲۵ ایپاک آموزش، به دقت:

- ۹۳ درصد برای داده‌های آموزش در بهترین حالت،
- حدود ۸۹.۱ درصد برای داده‌های اعتبار سنجی و
- ۸۹ درصد برای داده‌های تست

رسیده است.

حال طبق روش fine-tune مقاله، بلاک آخر (در واقع بلاک ۵ ام) vgg16 را قابل آموزش می‌کنیم. بنابراین در حال حاضر، 17,879,553 پارامتر قابل آموزش خواهیم داشت.

حال دوباره مدل را با همان پارامترها در شرایط آنفریز آموزش می‌دهیم. که نمودار دقت و loss آن به صورت زیر است.



شکل 21. نمودار دقت و **loss** برای حالت **unfreezed** مدل **vgg16**

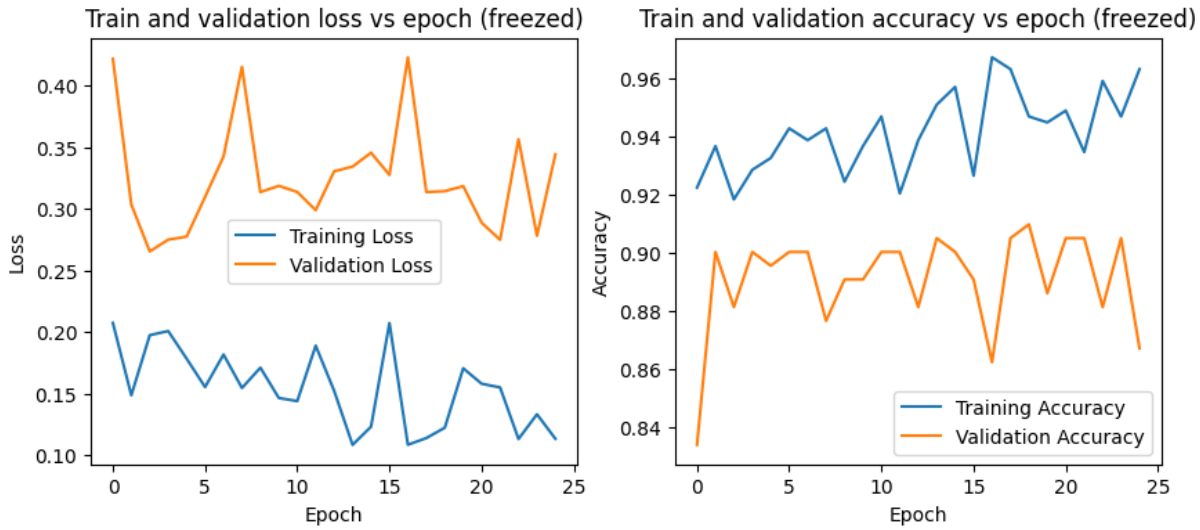
با توجه به نمودارهای داده شده، در این ۲۵ ایپاک آموزش مدل آنفریز شده در بلاک آخر vgg16، دقت‌ها به صورت زیر خواهند بود:

- ۹۴ درصد برای داده‌های آموزش در بهترین حالت،
- ۹۰ درصد برای داده‌های اعتبارسنجی و
- ۹۳ دردصد برای داده‌های تست

نکته قابل توجه کاهش خوب loss مدل برای داده‌های آموزش و اعتبارسنجی است، همچنین دقت مدل برای این داده‌ها افزایش یافته است. که این اتفاق هدف ما از Fine-Tune کردن مدل در شرایط آنفریز است. بنابراین یک مدل خوب با داده‌های آگمنت شده بدست آمد که تقریبا مدل بهتری از مدل ارائه شده در مقاله می‌باشد.

آموزش مدل با استفاده از داده اصلی شده.

دوباره مدل را در بخش vgg16 فریز کرده و سپس مدل را با همان پارامترهای قبل (مانند مقاله) آموزش می‌دهیم. نمودار آموزش در بخش آموزش فریز شده را می‌توانیم در شکل زیر مشاهده کنیم.



شکل 22. نمودار دقت و **loss** برای داده‌های اصلی در حالت **freezed**

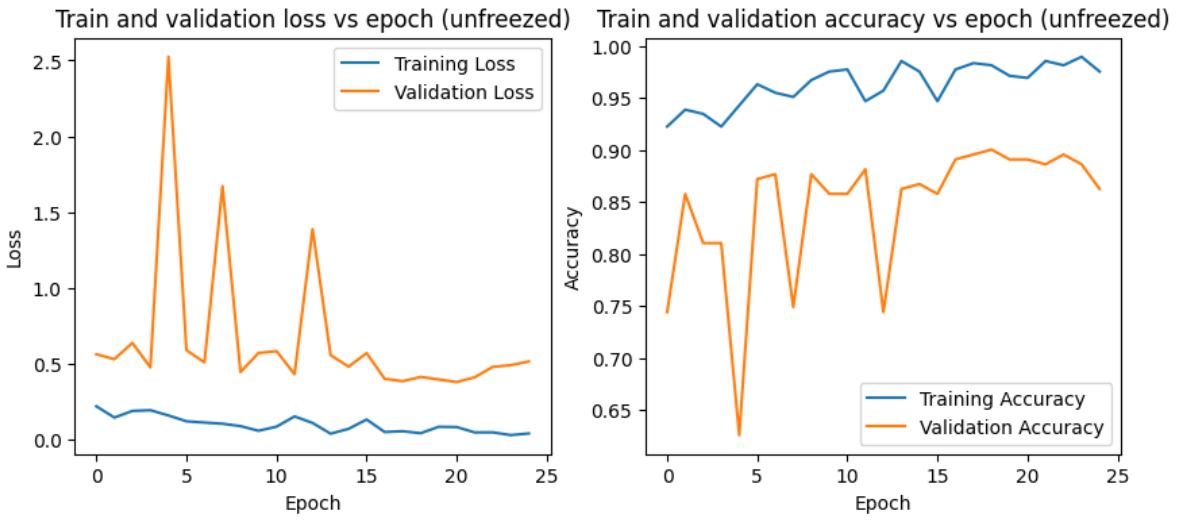
با توجه به نمودار فرآیند آموزش متوجه می‌شویم که همان اتفاقی که انتظار داشتیم، می‌افتد. اینکه مدل شروع به overfit روی داده‌های کم آموزش کرده و دیگر روی داده‌های اعتبارسنجی و همچنین داده‌های تست، به خوبی عمل نخواهد کرد. در واقع انجام آگمنتیشن برای افزایش دیتا، به منظور جلوگیری از همین overfit انجام می‌شود.

دقت‌های این آموزش به صورت زیر است:

- در بهترین حالت، دقت ۹۷ درصد روی داده‌های آموزش،
- دقت ۹۱ درصد برای داده‌های اعتبارسنجی و
- دقت ۹۳ برای داده‌های تست

در حال حاضر به نظر می‌آید که نتایج بهتر از حالت augment شده هستند، در صورتی که این اتفاق تنها در ظاهر است و دلیل آن در بسیار کوچک بودن دیتاستی که در اختیار داریم است. عکس‌ها حدودا شمایلی مانند یک دیگر دارند (بدون آگمنت شدن)، و به همین دلیل دقت روی تست و اعتبارسنجی به صورت سوری بالا خواهد بود. ولی از طرفی خاصیت تعمیم پذیری روی داده‌های دیده نشده، نخواهد داشت.

حال دوباره بلاک آخر (بلاک ۵ ام) vgg16 را unfreeze کرده و مدل را با همان پارامترها، آموزش می‌دهیم. نتایج آن در نمودارهای زیر قابل مشاهده است.



شکل 23. نمودار دقت و **loss** در حالت **Unfreezed** برای **vgg16**

با توجه به نمودار صفحه قبل، به طور مشخص در طول فرآیند آموزش، overfit اتفاق افتاده. همچنین این برازش بیش از حد، نسبت به دفعه قبل در آموزش شبکه freezed، بیشتر شده و فاصله دقت بر داده‌های آموزش از داده‌های اعتبارسنجی، بیشتر شده. حتی این دقت‌ها کاهش نیز داشته اند (تنها شاهد افزایش دقت در داده‌های آموزش بودیم) که این از نتایج برازش بیش از حد و کم بودن دیتا است. قابل ذکر است که اگر این مدل با تعداد بیشتری ایپاک اجرا می‌شد و آموزش می‌دید، در این صورت میزان overfit به طور غیر قابل چشم پوشی افزایش میافت. به این صورت که از روند آموزش مشخص است، احتمالا با اجرای بیشتر آن، دقت مدل بر روی داده‌های آموزش تا ۱۰۰ درصد هم رفته و دقت بر داده‌های اعتبارسنجی و تست کاهش میابد.

ResNet50

در ابتدا backbone مدل را که در اینجا از ResNet50 استفاده می‌کنیم، لود می‌کنیم. این کار به روش زیر انجام می‌شود:

```
resnet_base = ResNet50(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
```

در اینجا، وزن‌های شبکه ResNet50 را که با دیتا معروف imagenet آموزش دیده‌اند، لود می‌کنیم. همچنین با استفاده از دستور include_top برابر False، قسمت fully connected انتهایی این شبکه را لود نمی‌کنیم، چرا که می‌خواهیم لایه fully connected مناسب برای داده‌ها و تسک خود را به آن اضافه کنیم. لایه‌های مناسب را به شبکه به صورت زیر اضافه کرده و در نهایت آن را با همین ساختار، آموزش می‌دهیم.

- ابتدا لایه GlobalMaxPooling2D را به مدل اضافه می‌کنیم تا خروجی چند بعدی از شبکه ResNet50، را به صورت یک وکتور، داشته باشیم. اگر می‌خواستیم از لایه Flatten استفاده کنیم، تعداد پارامترها بسیار زیاد می‌شد، اما با استفاده از این لایه، اطلاعات مهم‌تر انتقال یافته و همچنین تعداد پارامترهای مدل به اندازه معقولی باقی می‌ماند.
- سپس یک لایه Batch Normalization خواهیم داشت تا نورون ها مقادیری در یک scale داشته باشند.
- حال اولین لایه Dense خود را که ۵۱۲ نورون دارد، اضافه می‌کنیم. activation function این لایه، ReLU است.
- یک لایه Batch Normalization خواهیم داشت تا نورون‌ها نرمالایز شوند.
- حال یک لایه DropOut با نرخ ۱۰ درصد داریم تا شبکه به صورت general تر آموزش ببیند و متکی به تعدادی از نورون‌های خاص نباشد.
- در نهایت برای هندل کردن مسئله classification خود، چون که target دیتا به صورت ۰ و ۱ دخیره شده، در این لایه تنها یک نورون خواهیم داشت که activation function آن، sigmoid خواهد بود.

آموزش مدل با استفاده از داده آگمنت شده.

1. برای آموزش مدل، ابتدا backbone مدل که ResNet50 است را **freeze** می‌کنیم. در واقع با اینکار اجازه ترین شدن لایه‌های کانولوشنی ResNet50 را در مدل تعریف شده نمی‌دهیم تا تنها در این مرحله، لایه‌های fully connected تازه اضافه شده ترین شوند. این کار را به روش زیر انجام می‌دهیم:

```
model.trainable = False
```

2. بار بررسی خلاصه مدل، تعداد پارامترها در کل **24,647,553** و پارامترهایی که قابلیت یادگیری در این مرحله را دارند، به تعداد **1,054,721** است.

3. حال قبل از آموزش مدل، ابتدا آن را compile می‌کنیم. در اینجا برخی از پارامترهای مربوط به بخش آموزش مدل را تعریف می‌کنیم که در آموزش شبکه و بروزرسانی وزن‌های شبکه دخیل هستند.

a. Optimizer: برای بروزرسانی نرخ یادگیری، از SGD Optimizer استفاده می‌کنیم. همچنین پارامترهای آن را مانند مقاله ست می‌کنیم.

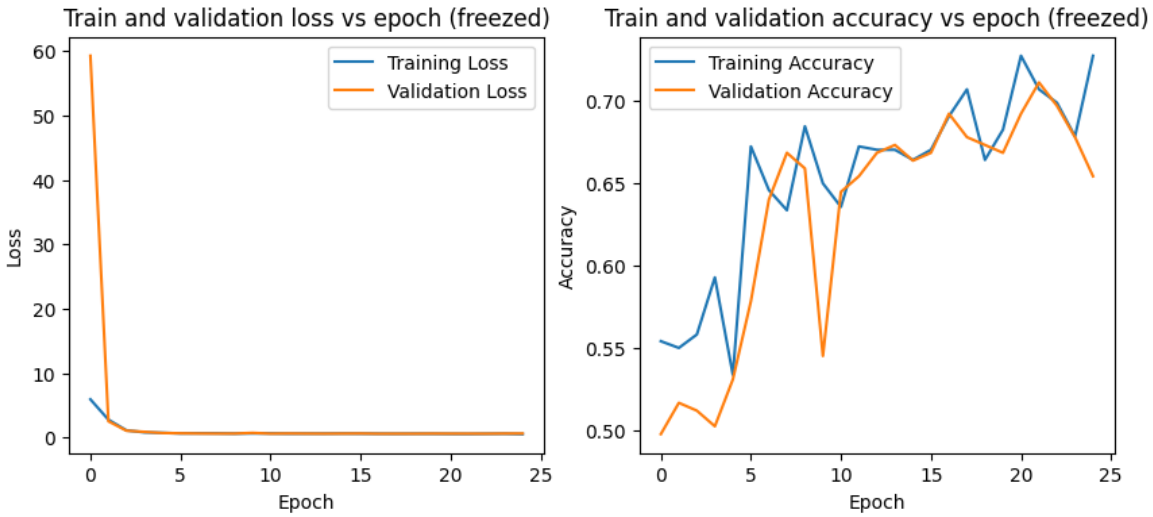
i. Learning rate = 0.1
ii. Momentum = 0.9
iii. Learning rate decay = 0.002

b. Loss function: برای تابع loss، binary cross entropy را قرار می‌دهیم، چرا که مقادیر خروجی ما به صورت باینری هستند.

4. سپس مدل را با استفاده از هایپر پارامترهای زیر مطابق با مقاله آموزش می‌دهیم.

a. Train Data = Augmented Data
b. Epoch = number of epochs (25 here)
c. Batch Size = batch_size (10 here)
d. Validation Data = splited data from train set

در این بخش از داده‌های آگمنت شده با استفاده از ImageDataGenerator، برای آموزش مدل استفاده می‌کنیم. می‌توانیم نتایج بدست آمده از آموزش را در نمودارهای موجود در شکل زیر بررسی کنیم.



شکل 24. نمودار دقت و **loss** برای **ResNet50** در حالت **freezed**

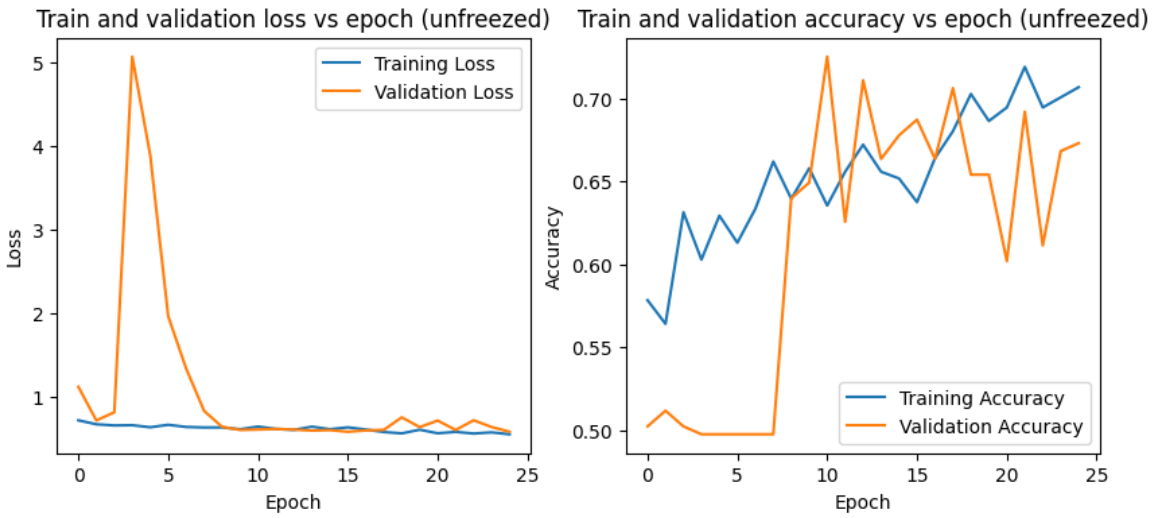
با توجه به نمودار متوجه می‌شویم آموزش مدل تقریبا خوب بوده و overfit اتفاق نیافتاده. در واقع دقت مدل روی داده‌های آموزش و اعتبارسنجی تقریبا روند مشابهی داشته. با توجه به این نمودارها، دقت مدل روی داده‌های ما به این صورت است:

- دقت ۷۲.۷ درصد در بهترین حالت بر روی داده‌های آموزش،
- دقت ۷۱ درصد در بهترین حالت برای داده‌های اعتبارسنجی،
- دقت ۶۵ درصد برای داده‌های ارزیابی.

دقت مدل خیلی کم‌تر از چیزی بوده که با استفاده از مدل VGG16 بدست آمده. همچنین این دقت کمتر از دقتی که مقاله مرجع بدست آورده می‌باشد.

حال طبق روش fine-tune مقاله، بلاک آخر (در واقع Conv5_block3) ResNet50 را قابل آموزش می‌کنیم. بنابراین در حال حاضر، 4,471,297 پارامتر قابل آموزش خواهیم داشت.

حال دوباره مدل را با همان پارامترها در شرایط آنفریز آموزش می‌دهیم. که نمودار دقت و loss آن به صورت زیر است.



شکل 25. نمودار دقت و **loss** برای **ResNet50** در شرایط **unfreeze**

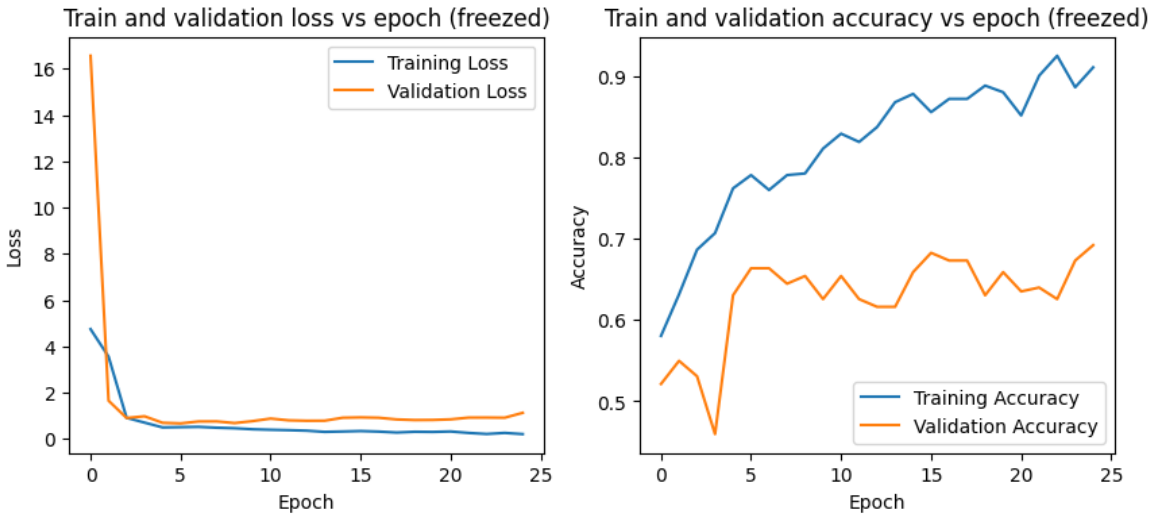
مشاهده می‌کنیم که دقت مدل برای داده‌های آموزش و اعتبارسنجی بعد از آموزش در حالت قابل آموزش بودن بلاک آخر، کمی کاهش یافته، اما دقت برای داده‌های تست افزایش داشته. (البته بهترین حالت دقت برای داده‌های اعتبارسنجی تفاوت کمی با حالت freezed داشته، در حالی که آخرین نتیجه مدل کمی، بیشتر اختلاف دارد). به صورت خلاصه دقت مدل برای داده‌ها به صورت زیر است:

- دقت ۷۲ درصد در بهترین حالت برای داده‌های آموزش،
- دقت ۷۰ درصد در بهترین حالت برای داده‌های اعتبارسنجی و
- دقت ۶۷ درصد برای داده‌های تست

نتیجه دیگری که می‌توان از این نمودار گرفت، این است که بعد از ایپاک ۱۰، مدل تقریبا شروع به overfit کرده و دقت مدل روی داده‌های اعتبارسنجی کم کم شروع به کاهش پیدا کردن می‌کند.

آموزش مدل با استفاده از داده اصلی شده.

دوباره مدل ResNet50 را فریز کرده و برای آموزش آماده می‌کنیم. سپس با همان پارامترهای مقاله، مدل را آموزش می‌دهیم. نتیجه آموزش مدل بر دیتا اصلی را می‌توانیم در نمودار زیر، ببینیم.



شکل 26. نمودار دقت و **loss** مدل **ResNet50** در شرایط **freezed**

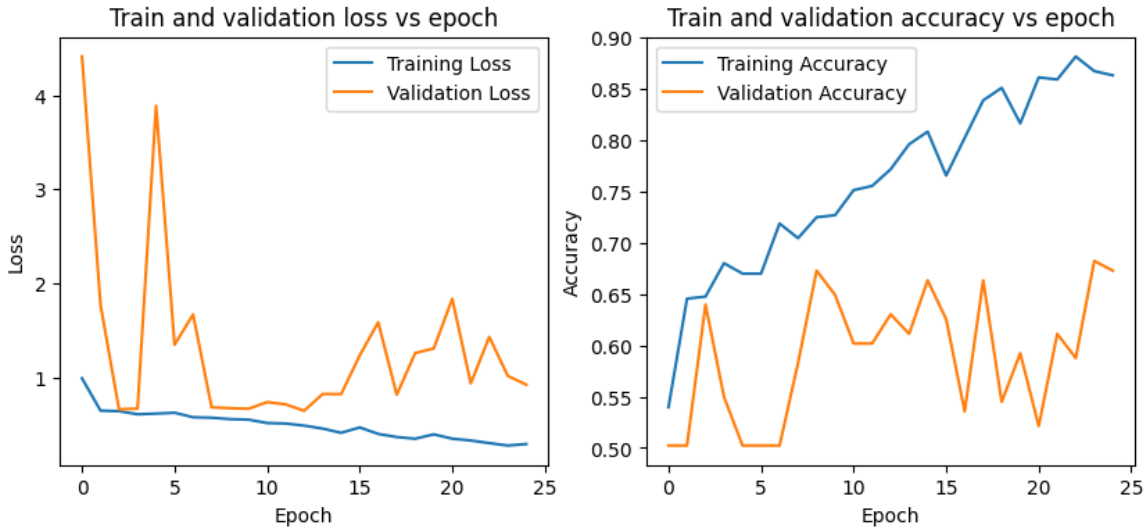
دقت بدست آمده مدل در حالتی که با دیتا اصلی آموزش دیده، برای دیتا مختلف به این صورت است:

- ۹۱ درصد در بهترین حالت برای داده آموزش،
- ۷۰ درصد برای داده‌های اعتبارسنجی و
- ۵۸ درصد برای داده تست

به طور واضح در این حالت، overfit اتفاق افتاده که دلیل بر کم بودن دیتا ورودی است. بنابراین مدل صرفا دیتا آموزش را یاد گرفته (در واقع حفظ می‌کند) و خاصیت تعمیم پذیری ندارد و نمی‌تواند دیتا دیده نشده را به خوبی classify کند. این برداشت را می‌توان از دقت پایینی که مدل روی داده تست گرفته نیز برداشت کرد.

حالا با آنفریز کردن بلاک Conv5_block3 که در واقع آخرین بلاک کانولوشنی است، مدل را طبق روش موجود در مقاله در بخش ۴.۱ آماده ادامه آموزش خود بر روی دیتا آموزش اصلی، می‌کنیم. در این حالت مدل جدید ما 4,471,297 پارامتر آماده آموزش خواهد داشت.

نتایج بدست آمده در آموزش مدل برای دیتا اصلی (آگمنت نشده) به صورت زیر است.



شکل 27. نمودار دقت و **loss** برای **ResNet50** در حالت **unfreezed**

همان گونه که از نمودار مشخص است، دقت مدل بعد از Fine-Tune کردن، برای دیتا مختلف به این صورت است:

- حدود ۹۰ درصد در بهترین حالت برای داده آموزش،
- ۶۷ درصد برای داده اعتبارسنجی و
- ۵۳ درصد برای داده تست

بنابراین بعد از Fine-Tune کردن مدل، میزان overfit کردن مدل روی داده‌های تست از جایی به بعد بیشتر شده و مدل تنها داده‌های تکراری آموزش را به خوبی حفظ می‌کند. Augmentation روش جلوگیری از overfit کردن مدل برای دیتاست کوچک‌تر از حد نیاز است که ما تاثیر آن را دیدم.

۲-۴. نتایج و تحلیل آن

در ابتدا هر کدام از مدل‌ها را برای داده‌های آگمنت شده و نشده بررسی می‌کنیم و سپس یک مقایسه کلی بین مدل‌هایی که از VGG16 و ResNet50 استفاده کردند، انجام می‌دهیم.

VGG16

طبق جدول شماره ۲، می‌توانیم تاثیر Data augmentation را به طور واضح ببینیم. به این صورت که برای داده‌های آگمنت نشده، دقت مدل روی داده‌های آموزش بسیار بیشتر بوده و همچنین loss نهایی آن به میزان قابل توجهی کم‌تر است. Overfit شدن مدل در حالت استفاده از داده اصلی آنجایی نمایان می‌شود که دقت داده تست و همچنین اعتبارسنجی در این حالت، کم‌تر از مدل آموزش دیده با داده آگمنت شده است. شدت این موضوع آنجایی مشخص می‌شود که loss برای هر دو داده تست و اعتبارسنجی، چیزی در حدود ۰.۲ افزایش یافته است.

بنابراین خاصیت انجام Data augmentation، که هدف از انجام آن افزایش داده برای جلوگیری از overfit کردن مدل و یا در واقع حفظ کردن داده محدود ورودی به جای یاد گرفتن آن است، در اینجا به خوبی از overfit کردن مدل جلوگیری کرده است.

همچنین اگر به خاطر داشته باشیم، مدل معرفی شده، نسبت به مدلی که در مقاله معرفی شده و از آن برای یادگیری این دیتا استفاده شده است، تعداد پارامترهای کمتری دارد (در حدود ۱۳ میلیون پارامتر کم‌تر). که این نشان دهنده بهینه بودن تعریف لایه‌های fully connected متصل به بلاک آخر کانولوشنی VGG16، تا زمان classify کردن ورودی است.

بنابراین توانستیم با یک مدل نسبتا بهینه و همچنین استفاده از یک مدل قبلا آموزش دیده (pre-trained) و Fine-Tune کردن آن، دقت خوب و قابل قبولی روی داده‌های در اختیار بدست بیاوریم و همچنین تاثیر مثبت افزایش دیتا و همچنین تغییر آن‌ها تا حد مشخصی (Data augmentation) را به خوبی شاهد باشیم.

	With data augmentation	Without data augmentation
Final training accuracy	90.22%	97.56%
Final validation accuracy	89.10%	86.26%
Test accuracy	93%	92%
Final training loss	0.2298	0.0392
Final validation loss	0.3189	0.5139
Test loss	0.2561	0.4516

جدول 2. خلاصه نتایج برای مدل **VGG16**

ResNet50

طبق جدول شماره ۳، می‌توانیم نتایجی شبیه به نتایج مدلی که از VGG16 استفاده کرده، بگیریم. در اینجا نیز دقت مدل روی داده‌های آموزش به شدت افزایش پیدا کرده (حتی بیشتر از VGG16)، و همچنین loss آن به شدت کاهش پیدا کرده. از طرفی دقت مدل روی داده‌های اعتبارسنجی تفاوت خاصی نداشته، اما loss آن به شدت زیاد و نزدیک به ۶۰ درصد افزایش پیدا کرده است. همچنین همان‌طور که شاهد هستیم، دقت مدل روی داده‌های تست، ۱۴ درصد کاهش داشته که کاهش شدیدی محسوب می‌شود و loss آن چیزی حدود ۲ برابر شده است.

بنابراین در اینجا نیز شاهد جلوگیری از overfit شدن مدل در حالت استفاده از داده اصلی با استفاده از داده augment شده، هستیم. اصولا augmentation زمانی اتفاق می‌افتد که ما داده بسیار کمی در اختیار داریم یا حتی اینکه داده‌های در دسترس قابلیت تعمیم پذیری به مدل نداده و تنها آن را با یک سری داده خاص آموزش می‌دهد. پس augmentation برای ایجاد خاصیت generalization مدل در شرایط کمبود داده، استفاده می‌شود.

	With data augmentation	Without data augmentation
Final training accuracy	70.67%	86.35%
Final validation accuracy	67.30%	67.30%
Test accuracy	67%	53%
Final training loss	0.5539	0.2936
Final validation loss	0.5842	0.9259
Test loss	0.6176	1.3322

جدول 3. خلاصه نتایج برای مدل **ResNet50**

مقایسه کلی نتایج بدست آمده

ابتدا برخی از تفاوت‌های دو مدل VGG16 و ResNet50 را بررسی کنیم تا ببینیم دلیل این نتایج بر اساس ساختار این دو شبکه، چگونه است. چرا که ممکن است تفاوت عملکرد آن‌ها به دلیل تفاوت آن‌ها در ساختار باشد.

- عمق شبکه‌ها: ResNet50 یک شبکه عمیق‌تر نسبت به VGG16 است. شبکه‌های عمیق‌تر به طور کلی ظرفیت بیشتری برای یادگیری ویژگی‌های پیچیده از داده دارند. این موضوع ممکن است مزیتی باشد زمانی که با مجموعه داده‌های پیچیده‌تر یا زمانی که نیاز به گرفتن جزئیات دقیق‌تر از تصاویر است، سودمند باشد.
- Skip connection: ResNet50 اتصالات ردیابی یا اتصالات کوتاه مسیر (Skip connection) را معرفی می‌کند که به کاهش مشکل کاهش گرادیان کمک می‌کند. این کار، این امکان را می‌دهد که گرادیان‌ها بهتر در طول آموزش منتقل شوند، به ویژه در شبکه‌های عمیق‌تر. در واقع استفاده از skip connectionها، در برای جلوگیری از gradient vanishing اتفاق استفاده می‌شوند. این اتصالات ردیابی ممکن است ResNet50 را قادر به یادگیری بهتر از داده‌های محدود موجود در مجموعه داده ما کند.
- Parameter efficiency: ResNet50 نسبت به VGG16 با استفاده از بلوک‌های ماندگاری کارآیی پارامتری بیشتری دارد. این بدان معناست که ResNet50 ممکن است با استفاده از کمترین پارامترها عملکرد بهتری داشته باشد که این موضوع زمانی که با مجموعه داده‌های کوچک مانند دیتا در اختیار کار می‌کنید، مفید است.

با توجه به این عوامل، ممکن استResNet50 ، با وجود عمق بیشتر، برای ویژگی‌های خاص مجموعه داده در اختیار، مناسب‌تر باشد و برتری را نسبت به VGG16 از خود نشان دهد. این اتفاق برای مقاله افتاده و همانطور که شاهد هستیم، مقاله نتایج بهتری با استفاده از ResNet50 بدست آورده تا استفاده از VGG16. بنابراین این مشکل ریشه در جای دیگری دارد وگرنه ساختار ResNet50 نسبت به VGG16 ساده تر بوده و پیچیدگی کم‌تری دارد (تعداد پارامترها در پیاده سازی ما حدود ۴ میلیون و در مقاله حدود ۱۷ میلیون کم‌تر است).

حال که ResNet50 در کمال تعجب عملکرد بهتری ندارد، ممکن است برخی از دلایل زیر باعث این عملکرد بد شوند.

- عدم تعادل یا تغییرپذیری داده: مجموعه داده ما ممکن است دارای کلاس‌های نامتعادل یا تغییرپذیری ذاتی باشد که بر توانایی ResNet50 برای عمومی‌سازی تأثیر بگذارد. ResNet50 ممکن است الگوهای خاصی در داده‌ها را حفظ کرده باشد که نماینده‌ی توزیع کلی نباشند. و در این صورت قادر به عملکرد بهتر نباشد. که می‌دانیم در دیتاستی که در اختیار داریم، این موضوع دیده نمی‌شود.
- با اینکه ResNet50 یک مدل قدرتمند است، ممکن است برای یادگیری انتقالی روی این مجموعه داده خاص بهترین انتخاب نباشد. وزن‌های پیش‌آموزش دیده ممکن است برای وظیفه در دست به اندازه‌ی مشابه یا مفید نباشند.

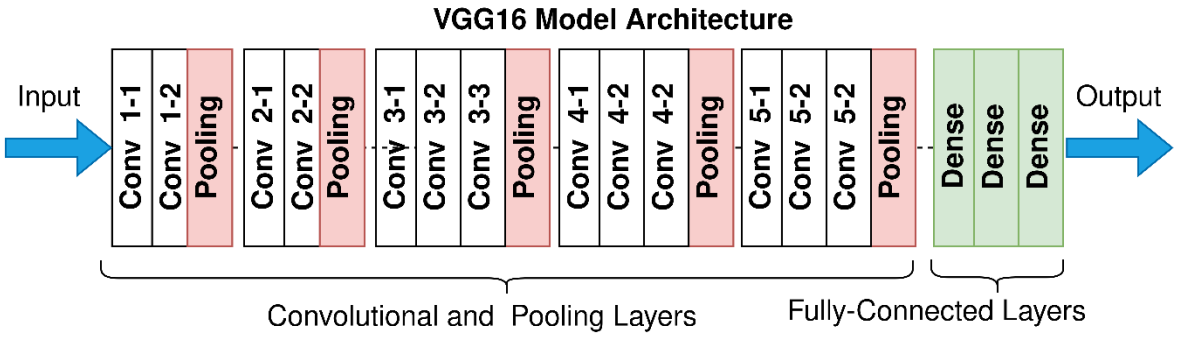
- کمبود داده: با توجه به اندازه کوچک مجموعه داده، ممکن است این مسئله باعث شود که هر دو مدل به طور کلی مشکل داشته باشند در یادگیری و عملکرد بهتری از آنها نتیجه ندهند.

- پیچیدگی مدل: مدل‌های عمیق مانند ResNet50 ممکن است بیش از حد پیچیده باشند برای یک مجموعه داده کوچک، که باعث بیش‌برازش شوند. در این صورت، ممکن است VGG16 که ساده‌تر است، عملکرد بهتری از خود نشان دهد. (البته در اینجا منظور از پیچیدگی تعداد پارامترها نیست، چرا که می‌دانیم ResNet50، دارای تعداد کم‌تری پارامتر است).

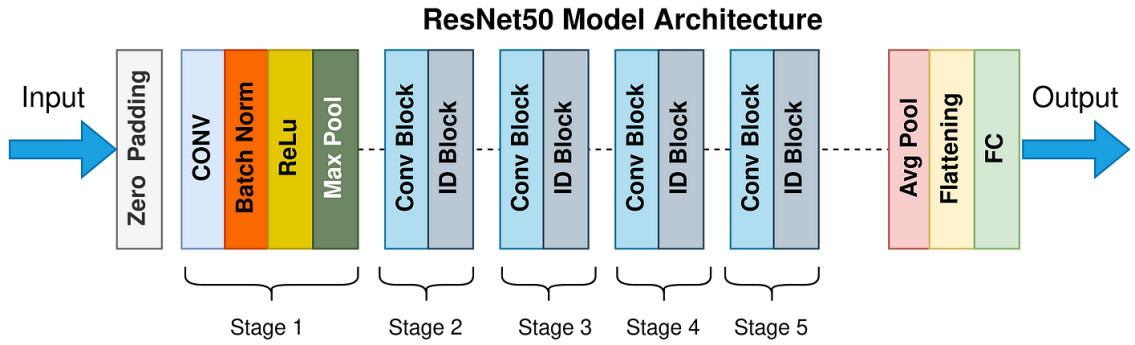
- استفاده از وزن‌های پیش‌آموزش دیده: با توجه به اندازه کوچک مجموعه داده، استفاده از وزن‌های پیش‌آموزش دیده ممکن است کمک کند تا مدل بهبود یابد. این موضوع ممکن است برای VGG16 مفیدتر باشد زیرا این مدل به اندازه کافی ساده است که از بیش‌برازش جلوگیری کند.

ممکن است به این دلایل ResNet50 علمکرد خوبی نداشته. اما با توجه به هدف اصلی مقاله، شاهد بودیم که Data Augmentation برای هر دو شبکه خوب بوده. همچنین استفاده از transfer learning بسیار کار را برای آموزش راحت و سریع می‌کند، در حالی که اجازه استفاده از یک شبکه بهتر و جامع‌تر داده شده است. از طرفی Data Augmentation نیز از هزینه‌های اضافی برای ساخت و تولید دیتا جلوگیری می‌کند.

در این صفحه می‌توانیم ساختار هر کدام از این مدل‌های از قبل آموزش دیده را به طور خلاصه ببینیم.



شکل 28. ساختار خلاصه **VGG16**



شکل 29. ساختار خلاصه **ResNet50**