به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین اول

نام و نام خانوادگی	محمدجواد رنجبر – بهراد موسایی
شماره دانشجویی	۸۱۰۱۰۱۱۷۳ - ۸۱۰۱۰۱۲۷۸
تاریخ ارسال گزارش	۱۴۰۱.۰۷.۰۱

فهرست

شکل‌ها

پاسخ ۱. تجزیه و تحلیل احساسات صورت مبتنی بر CNN

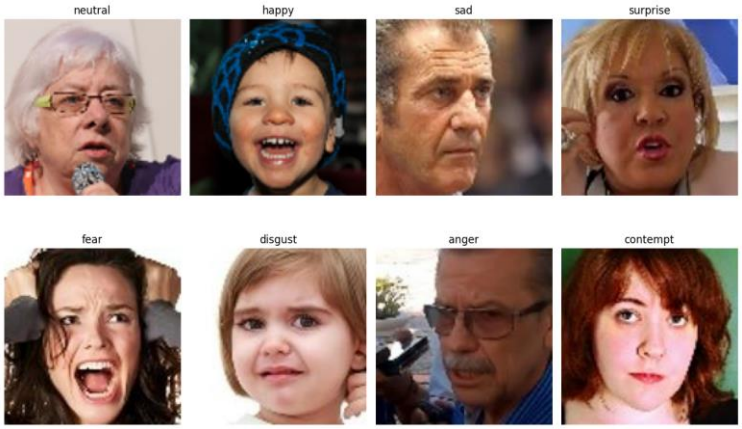
۱-۱. معرفی مقاله

این مقاله بر روی طراحی، استقرار و ارزیابی معماری‌های شبکه عصبی کانولوشنال[۱] برای تجزیه و تحلیل تأثیرات چهره در دستگاه‌های تلفن همراه تمرکز دارد. برخلاف رویکردهای سنتی CNN، مدل‌های مستقر در دستگاه‌های تلفن همراه باید نیازهای ذخیره‌سازی را به حداقل برسانند و در عین حال عملکرد بالا را حفظ کنند. بنابراین، ما سه نوع از معماری‌های CNN را پیشنهاد استفاده می‌کنیم.

۱-۲ پیش‌پردازش تصاویر و Data Augmentation

AffectNet یک مجموعه داده پرکاربرد در زمینه بینایی کامپیوتر و هوش مصنوعی است که به طور خاص بر روی تشخیص حالت چهره متمرکز است. این مجموعه شامل مجموعه وسیعی از تصاویر با حالات صورت مشروح شده است که طیف وسیعی از احساسات مانند شادی، غم، خشم، تعجب، ترس، انزجار و حالت های خنثی را به تصویر می کشد. این مجموعه داده به عنوان یک منبع ارزشمند برای آموزش و ارزیابی الگوریتم‌هایی است که برای درک و تفسیر احساسات انسان از طریق نشانه‌های چهره طراحی شده‌اند. محققان و توسعه‌دهندگان از AffectNet برای افزایش دقت و استحکام سیستم‌های تشخیص چهره استفاده می‌کنند و فناوری را قادر می‌سازد تا احساسات انسان را در برنامه‌های مختلف، از مراقبت‌های بهداشتی گرفته تا تعامل انسان و رایانه، بهتر درک کند و به آن پاسخ دهد.

[۱] Convolutional Neural Network



جدول ۱ نمونه داده‌های **AffectNet**

حال این تصاویر که دارای سه کانال (RGB) می‌باشد را برای آموزش باید آماده کنیم. علاوه بر نورمالیزیشن باید Augmentationهای مربوط به تعمیم‌پذیری را انجام دهیم. برای اینکار از ImageDataGenerator استفاده می‌کنیم.

«ImageDataGenerator» در Keras یک ابزار مهم برای تقویت داده‌های آموزشی است که تنوع مثال‌ها را از طریق تبدیل‌هایی مانند چرخش‌ها و جابجایی‌ها افزایش می‌دهد. این پیش پردازش بلادرنگ را با عادی سازی و استاندارد کردن مقادیر پیکسل ساده می کند و همگرایی سریع تر در طول آموزش را تسهیل می کند. با توانایی خود در تولید موثر دسته ای از تصاویر از پیش پردازش شده، روند آموزش را ساده می کند. علاوه بر این، ژنراتور از تنظیم اعتبار و داده‌های آزمایشی پشتیبانی می‌کند و به گردش کار یکپارچه کمک می‌کند. به طور کلی، «ImageDataGenerator» نقشی محوری در تهیه و تقویت مجموعه داده های تصویر برای آموزش شبکه عصبی، بهبود عملکرد و استحکام مدل ایفا می کند.

پیکربندی ImageDataGenerator برای آموزش:

- مقادیر پیکسل را در محدوده [۰،۱] عادی می کند (Rescale=1./255).
- تکنیک‌های تقویتی که برای تنوع بخشیدن به نمونه‌های آموزشی استفاده می‌شوند:
- چرخش تصادفی در ۲۰ درجه (rotation_range=20).
- تغییرات افقی و عمودی ۱۰٪ (width_shift_range=0.1، height_shift_range=0.1).
- ورق زدن افقی تصاویر (horizontal_flip=True).
- تقسیم ۸۰-۲۰ برای اعتبارسنجی (validation_split=0.2).

مشخصات تصویر:

- batch_size: روی ۴۰۰ تصویر در هر دسته تنظیم می‌کنیم (با توجه به مقاله).

- image_size: اندازه تصاویر به x128۱۲۸ پیکسل تغییر یافته است. بر اساس ابعاد تصویر خود را تنظیم کنید.

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    validation_split=0.2
)
```

شکل ۱ استفاده از **Imagegenrator**

حال نمونه‌ای از عکس‌ها را بعد از **Augment** کردن نمایش می‌دهیم:

شکل ۲ نمونه‌ای از **augmentation**

پیاده‌سازی مدل AlexNet

حال مدل AlexNet را با توجه به معماری مقاله پیاده‌سازی می‌کنیم:

جدول ۲ معماری شبکه **AlexNet**

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 128, 128, 16)      3888

 batch_normalization (Batch  (None, 128, 128, 16)      64
 Normalization)

 activation (Activation)     (None, 128, 128, 16)      0
```

```
max_pooling2d (MaxPooling2  (None, 64, 64, 16)        0
D)

gaussian_dropout (Gaussian  (None, 64, 64, 16)        0
Dropout)

conv2d_1 (Conv2D)           (None, 64, 64, 32)        25088

batch_normalization_1 (Bat  (None, 64, 64, 32)        128
chNormalization)

activation_1 (Activation)   (None, 64, 64, 32)        0

max_pooling2d_1 (MaxPoolin  (None, 32, 32, 32)        0
g2D)

gaussian_dropout_1 (Gaussi  (None, 32, 32, 32)        0
anDropout)

conv2d_2 (Conv2D)           (None, 32, 32, 64)        51200

batch_normalization_2 (Bat  (None, 32, 32, 64)        256
chNormalization)

activation_2 (Activation)   (None, 32, 32, 64)        0

max_pooling2d_2 (MaxPoolin  (None, 16, 16, 64)        0
g2D)

gaussian_dropout_2 (Gaussi  (None, 16, 16, 64)        0
anDropout)

conv2d_3 (Conv2D)           (None, 16, 16, 128)       73728

batch_normalization_3 (Bat  (None, 16, 16, 128)       512
chNormalization)

activation_3 (Activation)   (None, 16, 16, 128)       0

max_pooling2d_3 (MaxPoolin  (None, 8, 8, 128)         0
g2D)

gaussian_dropout_3 (Gaussi  (None, 8, 8, 128)         0
anDropout)

conv2d_4 (Conv2D)           (None, 8, 8, 128)         147456

batch_normalization_4 (Bat  (None, 8, 8, 128)         512
chNormalization)

activation_4 (Activation)   (None, 8, 8, 128)         0

max_pooling2d_4 (MaxPoolin  (None, 4, 4, 128)         0
g2D)

gaussian_dropout_4 (Gaussi  (None, 4, 4, 128)         0
anDropout)

flatten (Flatten)           (None, 2048)              0

dense (Dense)               (None, 1024)              2097152

batch_normalization_5 (Bat  (None, 1024)              4096
chNormalization)

activation_5 (Activation)   (None, 1024)              0

dropout (Dropout)           (None, 1024)              0
```

```
 dense_1 (Dense)             (None, 1024)              1048576

 batch_normalization_6 (Bat  (None, 1024)              4096
 chNormalization)

 activation_6 (Activation)   (None, 1024)              0

 dropout_1 (Dropout)         (None, 1024)              0

 dense_2 (Dense)             (None, 8)                 8200

=================================================================
Total params: 3464952 (13.22 MB)
Trainable params: 3460120 (13.20 MB)
Non-trainable params: 4832 (18.88 KB)
_________________________________________________________________
```

شبکه عصبی کانولوشنال الهام گرفته از الکس نت

این کد یک معماری شبکه عصبی کانولوشنال (CNN) شبیه AlexNet را ارائه می‌کند که با استفاده از Keras' Sequential API ساخته شده است. این ساختار مدل پیشگام AlexNet را منعکس می کند که دارای لایه های کانولوشن و کاملاً متصل برای طبقه بندی تصاویر است:

بلوک های کانولوشنال:

معماری: شبیه طرح AlexNet با پنج بلوک کانولوشن، که هر کدام شامل لایه های کانولوشن، نرمال سازی دسته ای، فعال سازی ReLU و max pooling است.

پیکربندی لایه: بلوک ها حاوی لایه های کانولوشنی با کاهش اندازه هسته (از ۹×۹ به ۳×۳) و تعداد فزاینده فیلترها (از ۱۶ به ۱۲۸) برای ثبت ویژگی های سلسله مراتبی هستند.

- Pooling: از حداکثر ادغام با یک هسته x۲ و padding برای نمونه برداری از نقشه های ویژگی بعد از هر بلوک استفاده می کند.
- Flatten: خروجی از لایه های کانولوشن را به یک بردار مسطح تبدیل می کند.
- لایه‌های تمام متصل: دو لایه کاملاً متصل متشکل از هر کدام از ۱۰۲۴ نورون را تعبیه می‌کند، همراه با Normalization دسته‌ای، فعال‌سازی ReLU و Dropout برای منظم‌سازی.
- طبقه‌بندی: این مدل در یک لایه متراکم با ۸ نورون با استفاده از فعال‌سازی softmax برای اهداف طبقه‌بندی چند طبقه، که ۸ کلاس مجزا را نشان می‌دهد، استفاده می‌کند.

- Dropout: تکنیک Regularization: از منظم‌سازی حذف با نرخ‌های مشخص شده به عنوان (۰.۲، ۰.۵) در لایه‌های کانولوشنال و متراکم استفاده می‌کند تا از برازش بیش از حد جلوگیری شود.

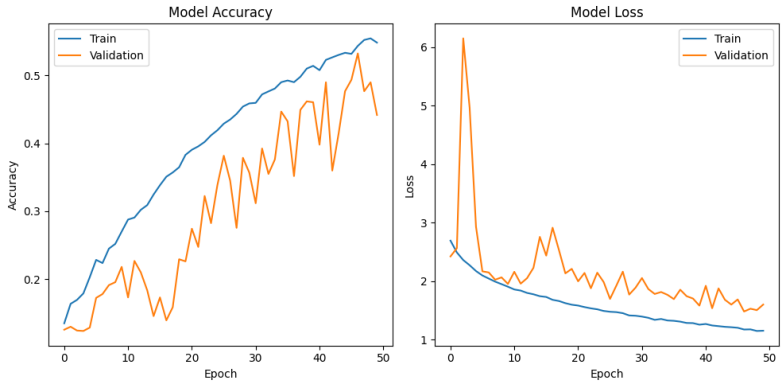
حال این مدل را با استفاده از داده‌های train آموزش داده و با استفاده از داده‌های validation ارزیابی می‌کنیم.

پیکربندی آموزش:

- اندازه دسته: برای پردازش یک دسته از ۴۰۰ نمونه در طول هر تکرار آموزشی، روی ۴۰۰ تنظیم کنید.
- دوره ها: آموزش برای ۵۰ دوره تنظیم شده است (epoch=50)، که نشان دهنده تعداد دفعاتی است که مدل در کل مجموعه داده در طول آموزش تکرار می شود.
- بهینه ساز: از بهینه ساز Adam با نرخ یادگیری ۰.۰۰۱، پارامترهای بتا و مقدار اپسیلون برای بهینه سازی استفاده می کند.
- Loss: مدل را با استفاده از categorical cross-entropy به عنوان تابع ضرر و دقت به عنوان متریک ارزیابی آماده شده است.
- بهینه ساز: «آدام» را به عنوان بهینه ساز آموزش مشخص شده است.

پس از پایان آموزش به ارزیابی مدل می‌پردازیم.

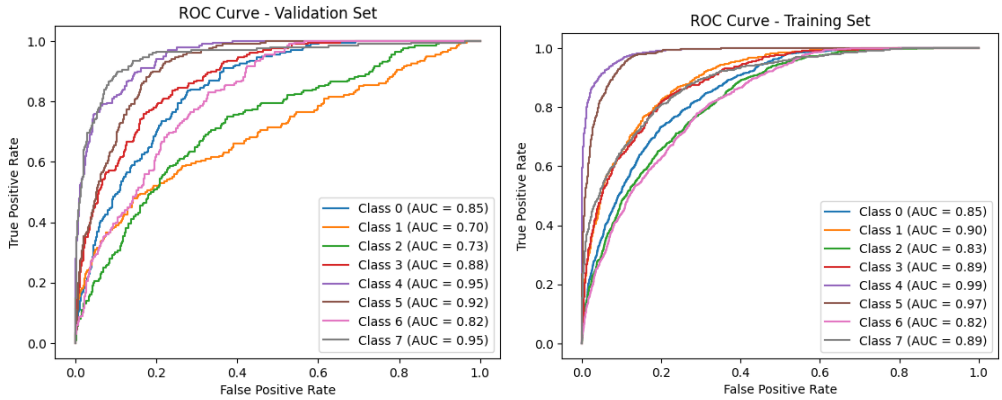
- نمودار دقت و خطا:



شکل ۳ نمودار دقت و خطا برای آموزش **AlexNet**

نمودار رسم شده که از loss و دقت مدل را در طول آموزش نشان می دهد، بینش‌های مهمی را نشان می‌دهد. یعنی پیشرفت معیارهای آموزشی و اعتبارسنجی را در طول دوره‌ها نشان می‌دهد. مشاهدات زیر را می توان انجام داد:

- پیشرفت آموزش:
  - بهبود دقت: دقت آموزش به طور پیوسته در طول دوره ها افزایش می یابد، که نشان می دهد مدل به طور مداوم از داده های آموزشی یاد می گیرد و قابلیت های پیش بینی خود را بهبود می بخشد.
  - کاهش loss: به طور همزمان، تلفات آموزشی به طور مداوم کاهش می یابد، که نشان دهنده توانایی مدل برای به حداقل رساندن خطاها در مجموعه آموزشی است.
- ارزیابی داده‌ی اعتبار سنجی:
  - دقت داده‌ی اعتبارسنجی: دقت در مجموعه داده اعتبارسنجی (ارزیابی) یک روند ثابت یا افزایشی را نشان می دهد، نه کاهشی را نشان می دهد. این دقت اعتبار سنجی پایدار یا صعودی نشان می دهد که مدل به خوبی به داده های دیده نشده تعمیم می یابد و با عملکرد آن در مجموعه آموزشی همسو می شود.
  - loss داده‌ی اعتبارسنجی: به طور مشابه، تلفات اعتبار سنجی ثابت می ماند یا به کاهش ادامه می یابد، و نشان می دهد که مدل در هنگام آزمایش بر روی داده های جدید با مسائل بیش از حد برازش یا واگرایی قابل توجهی از رفتار آموزشی مواجه نمی شود.
- تفسیر:
  - ظرفیت مدل: افزایش مداوم در دقت آموزش همراه با معیارهای اعتبارسنجی پایدار یا بهبود یافته نشان می‌دهد که مدل هنوز به ظرفیت یادگیری خود نرسیده است. هنوز پتانسیل برای یادگیری بیشتر بدون مواجهه با بیش از Overfit وجود دارد.
  - ادامه آموزش: از آنجایی که هم معیارهای آموزشی و هم اعتبارسنجی روندهای مثبتی را بدون نشانه‌های واگرایی یا بیش از حد برازش نشان می‌دهند، نشان می‌دهد که مدل در همچنان جای زیادی برای یادگیری دارد و می‌توانیم بدون نگرانی از overfit ادامه بدهیم.
- نمودار ROC مربوط به هرکلاس:



شکل ۴ نمودار **ROC** برای **AlexNet**

با توجه به ROC متوجه می‌شویم، مدل در تشخیص بعضی از کلاس‌ها مانند کلاس بنفش خوب عمل می‌کند و با آستانه‌ی بالا قابلیت تشخیص دارد، با این حال مدل در بعضی از کلاس‌ها مانند کلاس نارنجی مشکل دارد.

- مقادیر متریک‌های ارزیابی:

جدول ۳ عملکرد شبکه **AlexNet** روی داده‌های آموزش

Training Set Classification Report:

	precision	recall	f1-score	support
anger	0.59	0.15	0.24	800
contempt	0.82	0.14	0.23	800
disgust	0.44	0.20	0.28	800
fear	0.63	0.37	0.46	800
happy	0.89	0.82	0.85	800
neutral	0.83	0.61	0.70	800
sad	0.25	0.82	0.39	800
surprise	0.44	0.72	0.54	800
accuracy			0.48	6400
macro avg	0.61	0.48	0.46	6400
weighted avg	0.61	0.48	0.46	6400

Validation Set Classification Report:

	precision	recall	f1-score	support
anger	0.57	0.10	0.18	200
contempt	0.67	0.06	0.11	200
disgust	0.36	0.14	0.20	200
fear	0.57	0.32	0.41	200
happy	0.72	0.71	0.72	200
neutral	0.54	0.56	0.55	200
sad	0.28	0.74	0.40	200
surprise	0.45	0.93	0.60	200
accuracy			0.45	1600

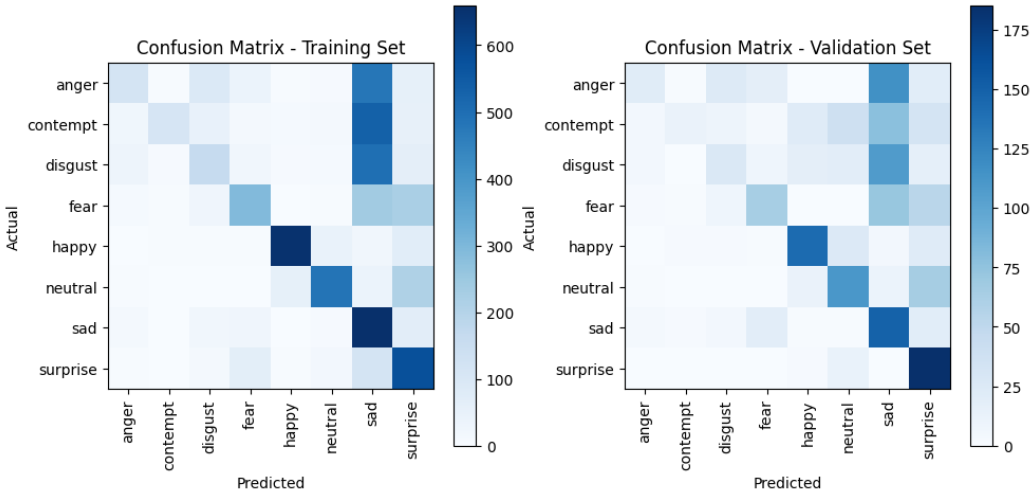
macro avg	0.52	0.45	0.40	1600
weighted avg	0.52	0.45	0.40	1600

- مجموعه آموزشی:
- precision : بالاترین برای "شاد" (۰.۸۹) و "خنثی" (۰.۸۳) که نشان دهنده پیش بینی های مثبت دقیق است. کمترین مقدار برای «تحقیر» (۰.۸۲) و «انزجار» (۰.۵۹)، که نشان دهنده مثبت کاذب بیشتر است.
- recall: «غمگین» (۰.۸۲) و «غافلگیرکننده» (۰.۷۲) recall بالایی دارند، که نشان دهنده موارد از دست رفته کمتر است. "خشم" (۰.۱۵) و "تحقیر" (۰.۱۴) recall کمی دارند که نشان دهنده بسیاری از موارد از دست رفته است.
- امتیاز F1: «شاد» بالاترین امتیاز f1 (۰.۸۵) را نشان می‌دهد که نشان‌دهنده تعادل بین دقت و یادآوری است. «تحقیر» (۰.۲۳) و «انزجار» (۰.۲۸) امتیازات پایین تری دارند که نشان دهنده عدم تعادل بین دقت و یادآوری است.
- به طور کلی: «خشم»، «تحقیر» و «انزجار» عملکرد نسبتاً ضعیفی دارند و precision، recall و امتیازات f1 کمتری را در مقایسه با سایر کلاس‌ها نشان می‌دهند.
- مجموعه اعتبار سنجی:
  - Recall: بالاترین برای «سورپرایز» (۰.۴۵) و «شاد» (۰.۷۲)، که نشان دهنده پیش بینی های مثبت دقیق است. کمترین میزان برای «انزجار» (۰.۳۶) و «خشم» (۰.۵۷) که مثبت کاذب بیشتری را نشان می دهد.
  - precision: «سورپرایز» (۰.۹۳) و «غمگین» (۰.۷۴) یادآوری بالایی را نشان می‌دهند که نشان‌دهنده موارد از دست رفته کمتر است. «خشم» (۰.۱۰) و «تحقیر» (۰.۰۶) یادآوری کمی دارند که نشان دهنده بسیاری از موارد از دست رفته است.
  - امتیاز "F1: "Happy دارای بالاترین امتیاز f1 (۰.۷۲) است که نشان دهنده تعادل بین precision و Recall است. «تحقیر» (۰.۱۱) و «انزجار» (۰.۲۰) نمرات پایین تری دارند که نشان دهنده عدم تعادل بین دقت و یادآوری است.
  - به طور کلی: «خشم»، «تحقیر» و «انزجار» دوباره عملکرد ضعیف‌تری را در مقایسه با سایر کلاس‌ها در دقت، یادآوری و امتیازات f1 نشان می‌دهند.

نتیجه‌گیری:

در هر دو مجموعه آموزشی و اعتبارسنجی، «خشم»، «تحقیر» و «انزجار» به طور مداوم معیارهای عملکرد پایین‌تری را نشان می‌دهند، که نشان می‌دهد مدل با این کلاس‌ها بیشتر مشکل دارد. دلیل این اتفاق می‌تواند، شباهت ظاهری این کلاس‌ها باشد که برای انسان نیز تشخیص آن‌های پیچیده خواهد بود.

- ماتریس درهمریختگی

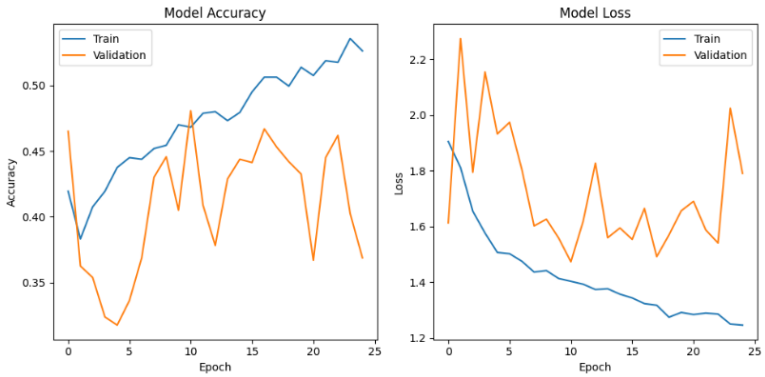


شکل ۵ ماتریس درهمریختگی برای آموزش مدل **AlexNet**

ماتریس سردرگمی چالش‌هایی را در توانایی مدل برای تمایز بین کلاس‌هایی که شباهت‌های مشترک دارند نشان می‌دهد. به طور خاص، با متمایز کردن احساساتی که نزدیک به یکدیگر مرتبط هستند مبارزه می کند. به عنوان مثال، احساساتی مانند ناراحتی، شبیه به خشم، حالت تهوع و تحقیر، برای مدل در جداسازی موثر این احساسات با مشکل مواجه می شود. این فقدان تمایز واضح در بین طبقات نزدیک به هم منعکس کننده محدودیت های مدل در دسته بندی دقیق این حالت های هیجانی ظریف است.

حال برای داده‌های ارزیابی و Tune مقادیر ارزیابی عملکرد مدل را محاسبه می‌کنیم:

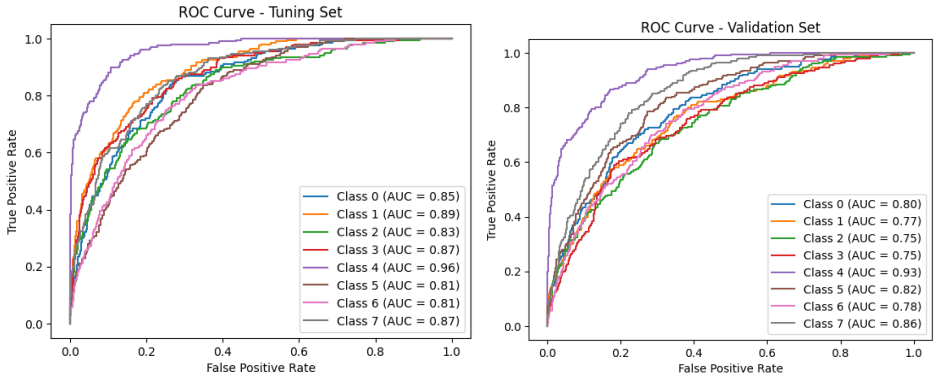
- نمودار loss و accuracy:



شکل ۶ نمودار دقت و خطا برای **AlexNet** روی داده‌ی **Tune**

پس از tunning، مدل ظرفیت یادگیری خود را حفظ می کند، که از افزایش مداوم دقت آموزش مشهود است. قابل ذکر است که دقت ارزیابی ثابت می ماند که نشان دهنده عدم وجود overfit است. این سازگاری نشان می‌دهد که مدل همچنان می‌تواند یاد بگیرد و بدون به خطر انداختن توانایی خود برای تعمیم به داده‌های جدید، پیشرفت کند. با ادامه روند صعودی دقت، آموزش بیشتر برای افزایش عملکرد مدل مفید به نظر می رسد.

- نمودار ROC مربوط به هرکلاس:



شکل ۷ نمودار **ROC** برای داده‌های **Tune** مدل **AlexNet**

با توجه به ROC متوجه می‌شویم، مدل در تشخیص بعضی از کلاس‌ها مانند کلاس بنفش خوب عمل می‌کند و با آستانه‌ی بالا قابلیت تشخیص دارد، با این حال مدل در بعضی از کلاس‌ها مانند کلاس نارنجی مشکل دارد. بعد از tune کردن مدل، بیشتر کلاس‌ها به سمت چپ گوشه این نمودار نزدیکتر می‌شوند که نشان از بهتر شدن مدل دارد.

- مقادیر متریک‌های ارزیابی:

جدول ۴ عملکرد شبکه **AlexNet** روی داده‌های **Tune**

```
Tuning Set Classification Report:
```

	precision	recall	f1-score	support
anger	0.32	0.76	0.45	200
contempt	0.59	0.52	0.55	200
disgust	0.44	0.41	0.42	200
fear	0.34	0.77	0.47	200
happy	0.96	0.39	0.55	200
neutral	0.74	0.07	0.13	200
sad	0.37	0.38	0.37	200
surprise	0.86	0.03	0.06	200
accuracy			0.41	1600
macro avg	0.58	0.41	0.37	1600
weighted avg	0.58	0.41	0.37	1600

Validation Set Classification Report:

	precision	recall	f1-score	support
anger	0.28	0.65	0.39	200
contempt	0.42	0.26	0.32	200
disgust	0.27	0.31	0.29	200
fear	0.26	0.62	0.37	200
happy	0.86	0.41	0.55	200
neutral	0.56	0.20	0.29	200
sad	0.36	0.26	0.30	200
surprise	0.71	0.03	0.05	200
accuracy			0.34	1600
macro avg	0.47	0.34	0.32	1600
weighted avg	0.47	0.34	0.32	1600

نقاط قوت: «شاد» precision نسبتاً بالاتری را نشان می‌دهد اما Recall کمتری را نشان می‌دهد که در نتیجه امتیاز F1 متوسطی به دست می‌آید. «ترس» تعادل خوبی بین precision و Recall نشان می‌دهد که منجر به امتیاز بالاتر F1 می‌شود.

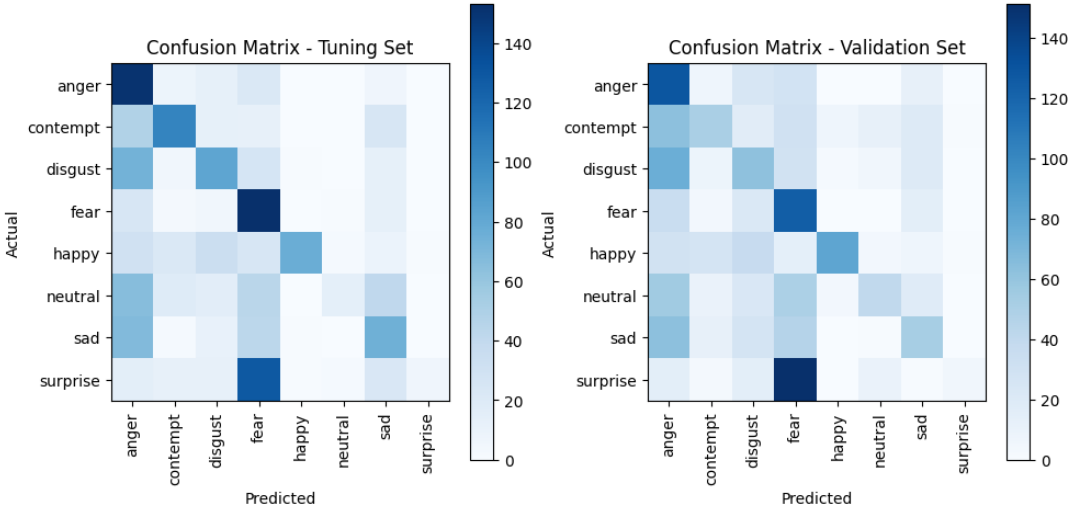
نقاط ضعف: "خنثی"، "غافلگیرانه" و "تحقیر" به طور قابل توجهی precision پایین، Recall و در نتیجه امتیازهای F1 پایین را نشان می دهد. این احساسات به دلیل پیش‌بینی‌های نامتعادل (دقت کم) و موارد از دست رفته (Recakk کم) چالش‌های مهمی را برای مدل ایجاد می‌کنند.

مجموعه اعتبار سنجی:

روندهای مشابه: مجموعه اعتبارسنجی الگوی سازگار با مجموعه تنظیم را نشان می دهد و نقاط قوت و ضعف مشابهی را در بین احساسات نشان می دهد.

در این مدل نیز همچنان در بعضی از کلاس‌ها عملکرد مدل بد است. دلیل این موضوع دوباره شباهت بین کلاس‌ها می‌تواند باشد. همچنین برای مثال، در کلاس‌هایی که ویژگی‌های خاصی وجود دارد مدل به شدت بهتر عمل کرده است، برای مثال کلاس شاد مدل خیلی خوب عمل می‌کند.

- ماتریس که ماتریس درهمریختگی به شکل زیر خواهد بود:



شکل ۸ ماتریس درهم‌ریختگی مدل **AlexNet** برای داده‌های **Tune**

با توجه به ماتریس درهم‌ریختگی متوجه می‌شویم که مدل در تشخیص بعضی از کلاس‌های نزدیک به هم مشکل دارد، برای مثال احساس سورپرایز را با ترس اشتباه می‌کند.

با این حال *tune* کردن مدل باعث بهتر شدن این مدل نسبت به مدل اصلی شده است که در این ماتریس نیز مشخص است.

پاسخ ۲ – پیاده‌سازی مدل VGGNet

۲-۱. مدل VGGNet

تمام مراحل پیش‌پردازش داده‌ها که در سوال قبل ذکر شد برای این سوال نیز پیاده‌سازی کرده‌ایم. پس معماری VGGNet را با توجه به مقاله پیاده‌سازی می‌کنیم. معماری معرفی شده برای این مدل به شکل زیر خواهد بود:

جدول ۵ معماری **VGGNet**

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 128, 128, 16)	432
batch_normalization (BatchNormalization)	(None, 128, 128, 16)	64
activation (Activation)	(None, 128, 128, 16)	0
conv2d_1 (Conv2D)	(None, 128, 128, 16)	2304
batch_normalization_1 (BatchNormalization)	(None, 128, 128, 16)	64
activation_1 (Activation)	(None, 128, 128, 16)	0
max_pooling2d (MaxPooling2D)	(None, 64, 64, 16)	0
gaussian_dropout (GaussianDropout)	(None, 64, 64, 16)	0
conv2d_2 (Conv2D)	(None, 64, 64, 32)	4608
batch_normalization_2 (BatchNormalization)	(None, 64, 64, 32)	128
activation_2 (Activation)	(None, 64, 64, 32)	0
conv2d_3 (Conv2D)	(None, 64, 64, 32)	9216
batch_normalization_3 (BatchNormalization)	(None, 64, 64, 32)	128
activation_3 (Activation)	(None, 64, 64, 32)	0
max_pooling2d_1 (MaxPooling2D)	(None, 32, 32, 32)	0
gaussian_dropout_1 (GaussianDropout)	(None, 32, 32, 32)	0

```
conv2d_4 (Conv2D)            (None, 32, 32, 64)        18432

batch_normalization_4 (Bat   (None, 32, 32, 64)        256
chNormalization)

activation_4 (Activation)    (None, 32, 32, 64)        0

conv2d_5 (Conv2D)            (None, 32, 32, 64)        36864

batch_normalization_5 (Bat   (None, 32, 32, 64)        256
chNormalization)

activation_5 (Activation)    (None, 32, 32, 64)        0

max_pooling2d_2 (MaxPoolin   (None, 16, 16, 64)        0
g2D)

gaussian_dropout_2 (Gaussi   (None, 16, 16, 64)        0
anDropout)

conv2d_6 (Conv2D)            (None, 16, 16, 128)       73728

batch_normalization_6 (Bat   (None, 16, 16, 128)       512
chNormalization)

activation_6 (Activation)    (None, 16, 16, 128)       0

conv2d_7 (Conv2D)            (None, 16, 16, 128)       147456

batch_normalization_7 (Bat   (None, 16, 16, 128)       512
chNormalization)

activation_7 (Activation)    (None, 16, 16, 128)       0

max_pooling2d_3 (MaxPoolin   (None, 8, 8, 128)         0
g2D)

gaussian_dropout_3 (Gaussi   (None, 8, 8, 128)         0
anDropout)

conv2d_8 (Conv2D)            (None, 8, 8, 128)         147456

batch_normalization_8 (Bat   (None, 8, 8, 128)         512
chNormalization)

activation_8 (Activation)    (None, 8, 8, 128)         0

conv2d_9 (Conv2D)            (None, 8, 8, 128)         147456

batch_normalization_9 (Bat   (None, 8, 8, 128)         512
chNormalization)

activation_9 (Activation)    (None, 8, 8, 128)         0

max_pooling2d_4 (MaxPoolin   (None, 4, 4, 128)         0
g2D)
```

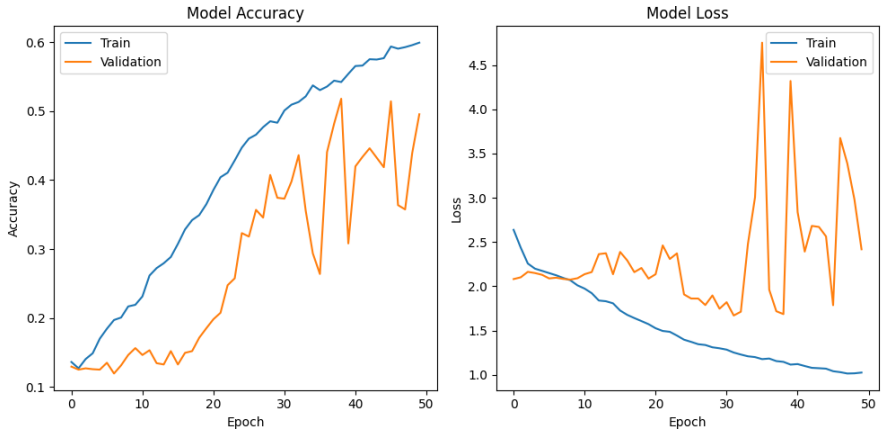
```
gaussian_dropout_4 (Gaussi  (None, 4, 4, 128)         0
anDropout)

flatten (Flatten)           (None, 2048)              0

dense (Dense)               (None, 1024)              2097152

batch_normalization_10 (Ba  (None, 1024)              4096
tchNormalization)

activation_10 (Activation)  (None, 1024)              0

dropout (Dropout)           (None, 1024)              0

dense_1 (Dense)             (None, 1024)              1048576

batch_normalization_11 (Ba  (None, 1024)              4096
tchNormalization)

activation_11 (Activation)  (None, 1024)              0

dropout_1 (Dropout)         (None, 1024)              0

dense_2 (Dense)             (None, 8)                 8200

=================================================================
Total params: 3753016 (14.32 MB)
Trainable params: 3747448 (14.30 MB)
Non-trainable params: 5568 (21.75 KB)
```

معماری مدل:

- بلوک های کانولوشن: شامل پنج مجموعه از دو لایه کانولوشن متوالی (هر کدام با هسته های ۳×۳) و نرمال سازی دسته ای و به دنبال فعال سازی ReLU است.
- لایه‌های ادغام: حداکثر لایه‌های ادغام با پنجره‌های ۲×۲ و لایه‌های «same» را بعد از هر جفت لایه‌های کانولوشن برای کم کردن پیچیدگی نقشه‌های ویژگی ترکیب می‌کند.
- Dropout: Gaussian را با نرخ ۰.۲ در هر بلوک کانولوشن و ۰.۵ در لایه های کاملاً متصل بعدی اعمال می کند.
- Flattening: خروجی را از لایه های کانولوشن به یک بردار مسطح تبدیل می کند.
- لایه‌های کاملاً متصل: از دو لایه متراکم با ۱۰۲۴ نورون استفاده می‌کند که از Normalization دسته‌ای، فعال‌سازی ReLU و Dropout برای Regularization استفاده می‌کند.
- لایه خروجی: با یک لایه متراکم از ۸ نورون با استفاده از فعال سازی softmax برای طبقه بندی چند کلاسه به پایان می رسد.

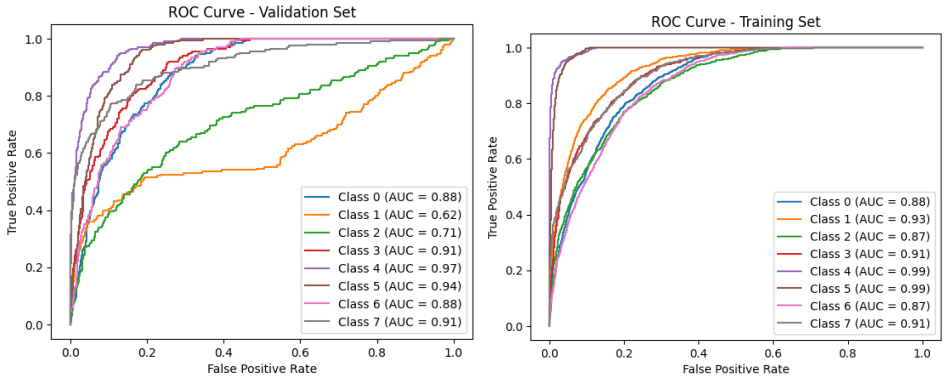
حال مدل را برای آموزش آماده می‌کنیم، پیکربندی این مدل نیز همانند مدل مطرح شده در سوال اول می‌باشد.

- ابتدا مدل را با استفاده از داده‌های آموزش، برای ۵۰ ایپاک آموزش می‌دهیم که نمودار دقت و خطای آن به صورت زیر خواهد بود:



شکل ۹ نمودار خطا و دقت برای مدل **VGGNet** برای داده‌های آموزش

- نموار ROC مربوط به هر کلاس به شکل زیر خواهد بود:



شکل ۱۰ نمودار **ROC** برای داده‌های آموزش مدل **VGGNet**

با توجه به ROC متوجه می‌شویم، مدل در تشخیص بعضی از کلاس‌ها مانند کلاس بنفش خوب عمل می‌کند و با آستانه‌ی بالا قابلیت تشخیص دارد، با این حال مدل در بعضی از کلاس‌ها مانند کلاس نارنجی مشکل دارد. که به شدت بد عمل می‌کند. با توجه به ROC داده‌های آموزش می‌توانیم متوجه شویم، مدل پترن داده‌های آموزش را تا حد خوبی یاد گرفته است و نزدیک به overfit شدن می‌باشد.

- حال متریک‌های ارزیابی شبکه را برای داده‌های ارزیابی و آموزش را نشان می‌دهیم:

روی داده‌های آموزش **VGGNet** عملکرد شبکه **6** جدول

```
Training Set Classification Report:
              precision    recall  f1-score   support
```

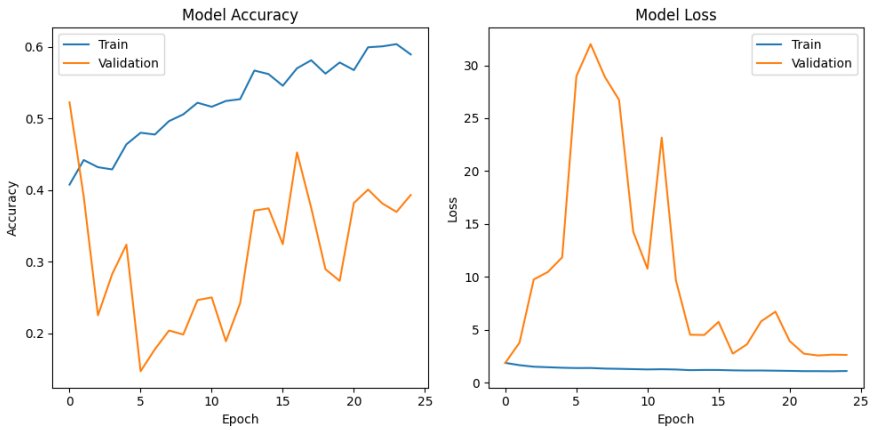
anger	0.43	0.57	0.49	800
contempt	0.61	0.58	0.59	800
disgust	0.69	0.22	0.33	800
fear	0.75	0.31	0.44	800
happy	0.98	0.67	0.80	800
neutral	0.64	0.98	0.77	800
sad	0.34	0.73	0.46	800
surprise	0.67	0.42	0.52	800
accuracy			0.56	6400
macro avg	0.64	0.56	0.55	6400
weighted avg	0.64	0.56	0.55	6400

Validation Set Classification Report:

	precision	recall	f1-score	support
anger	0.47	0.54	0.50	200
contempt	0.50	0.33	0.39	200
disgust	0.46	0.13	0.20	200
fear	0.65	0.23	0.33	200
happy	0.83	0.56	0.67	200
neutral	0.39	0.97	0.56	200
sad	0.41	0.68	0.51	200
surprise	0.70	0.55	0.61	200
accuracy			0.49	1600
macro avg	0.55	0.49	0.47	1600
weighted avg	0.55	0.49	0.47	1600

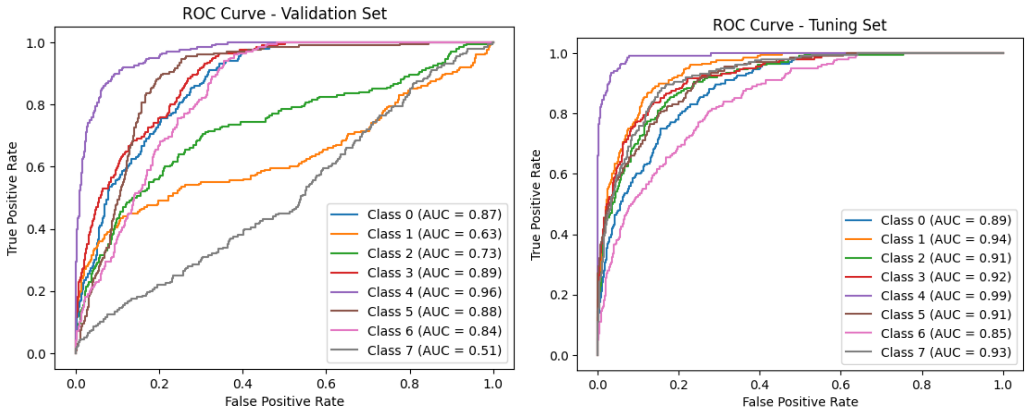
همچنین بعد از Finetune کردن نتایج به صورت زیر می‌باشد:

- نمودار خطا و دقت داده‌های آموزش و ارزیابی:



شکل ۱۱ نمودار خطا و دقت برای داده‌های **Tune** مدل **VGGNet**

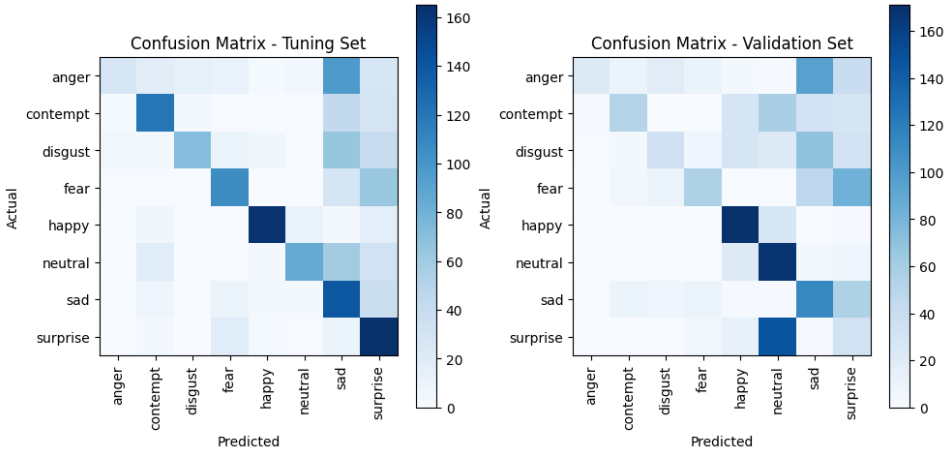
نمودار ROC برای داده‌های آموزش و ارزیابی:



شکل ۱۲ نمودار **ROC** برای داده‌های **Tune** مدل **VGGNet**

با توجه به ROC متوجه می‌شویم، مدل در تشخیص بعضی از کلاس‌ها مانند کلاس بنفش خوب عمل می‌کند و با آستانه‌ی بالا قابلیت تشخیص دارد، با این حال مدل در بعضی از کلاس‌ها مانند کلاس نارنجی مشکل دارد. با توجه به بدتر شدن نتیجه مدل نسبت به ROC قبلی می‌توانیم بفهمیم، داده‌های Tune میزان تعمیم‌پذیری لازم را نداشتند و مدل بیشتر به سمت overfit شدن پیش رفته است. نتایج یک سری کلاس‌ها بدتر شده است.

- ماتریس درهم‌ریختگی برای داده‌های آموزش و ارزیابی:



شکل ۱۳ ماتریس درهمریختگی مدل **VGGNet** برای داده‌های **Tune**

با توجه به ماتریس مشخص است که مدل بسیار بهتر از AlexNet عمل کرده است با این حال در کلاس‌های شبیه به هم همچون عصبانیت و ناراحتی اشتباه تشخیص می‌دهد.

تفاوت‌ها با مدل AlexNet:

- معماری: VGGNet معمولاً از اندازه‌های هسته کوچک‌تر (x۳۳) به طور مداوم در سراسر معماری استفاده می‌کند، در حالی که مدل AlexNet مانند از اندازه‌های هسته‌های مختلف (x۹۹ تا x۳۳) استفاده می‌کند.
- عمق: مشخصه کلیدی VGGNet ساختار یکنواخت آن با شبکه های عمیق تر به دلیل فیلترهای متوالی با اندازه کوچک است، در حالی که مدل AlexNet مانند لایه های کمتری در هر بلوک داشت.

استفاده از Dropout: هر دو مدل حذف را پیاده سازی می کنند. با این حال، نرخ های خاص و قرار دادن متفاوت است، که بر استراتژی منظم سازی مدل تاثیر می گذارد. در مدل AlexNet از dropout معمولی استفاده شده است، اما در مدل VGGNet از dropout گوسی.

- تفاوت dropout:
  - Dropout: غیرفعال سازی دودویی نورون ها بر اساس احتمال.
  - Gaussian Dropout: نویز گاوسی به صورت تصادفی به داده‌ها اضافه می کند.

با توجه به نتایج به نظر می‌آید که VGGNet از AlexNet بهتر عمل می‌کند، دلیل این موضوع می‌تواند عمیق‌تر بودن شبکه VGGNet نسبت به شبکه‌ی AlexNet می‌باشد که باعث شده است پترن‌های پیچیده در تصویر را بهتر یاد بگیرد، برای مثال در مدل AlexNet تشخیص بین کلاس‌هایی مانند غمگین و عصبانی سخت بود با این حال VGGNet این توانایی را بهتر فراهم می‌کند. همچنین dropout گوسی ممکن است که تعمیم‌پذیری بیشتری برای شبکه VGGNet فراهم کرده باشد و نتایج کلی آن بهتر شده باشد.

۲-۲ مدل MobileNet

در این مدل نیز تمام مراحل پیش‌پردازش مطرح شده در سوال یک را ابتدا انجام می‌دهیم. سپس مدل MobileNet را با توجه به مقاله پیاده‌سازی می‌کنیم. معماری معرفی شده برای این مدل به شکل زیر خواهد بود:

جدول ۷ عملکرد شبکه **VGGNet** روی داده‌های **Tune**

Layer (type)	Output Shape	Param #
conv2d_28 (Conv2D)	(None, 64, 64, 32)	864
batch_normalization_54 (BatchNormalization)	(None, 64, 64, 32)	128
activation_54 (Activation)	(None, 64, 64, 32)	0
depthwise_conv2d_26 (DepthwiseConv2D)	(None, 64, 64, 32)	288

```
batch_normalization_55 (Ba  (None, 64, 64, 32)        128
tchNormalization)

activation_55 (Activation)  (None, 64, 64, 32)        0

conv2d_29 (Conv2D)          (None, 64, 64, 64)        2048

batch_normalization_56 (Ba  (None, 64, 64, 64)        256
tchNormalization)

activation_56 (Activation)  (None, 64, 64, 64)        0
depthwise_conv2d_27 (Depth  (None, 32, 32, 64)        576
wiseConv2D)

batch_normalization_57 (Ba  (None, 32, 32, 64)        256
tchNormalization)

activation_57 (Activation)  (None, 32, 32, 64)        0

conv2d_30 (Conv2D)          (None, 32, 32, 128)       8192

batch_normalization_58 (Ba  (None, 32, 32, 128)       512
tchNormalization)

activation_58 (Activation)  (None, 32, 32, 128)       0

depthwise_conv2d_28 (Depth  (None, 32, 32, 128)       1152
wiseConv2D)

batch_normalization_59 (Ba  (None, 32, 32, 128)       512
tchNormalization)

activation_59 (Activation)  (None, 32, 32, 128)       0

conv2d_31 (Conv2D)          (None, 32, 32, 128)       16384

batch_normalization_60 (Ba  (None, 32, 32, 128)       512
tchNormalization)

activation_60 (Activation)  (None, 32, 32, 128)       0

depthwise_conv2d_29 (Depth  (None, 16, 16, 128)       1152
wiseConv2D)

batch_normalization_61 (Ba  (None, 16, 16, 128)       512
tchNormalization)

activation_61 (Activation)  (None, 16, 16, 128)       0

conv2d_32 (Conv2D)          (None, 16, 16, 256)       32768

batch_normalization_62 (Ba  (None, 16, 16, 256)       1024
tchNormalization)

activation_62 (Activation)  (None, 16, 16, 256)       0

depthwise_conv2d_30 (Depth  (None, 16, 16, 256)       2304
wiseConv2D)

batch_normalization_63 (Ba  (None, 16, 16, 256)       1024
tchNormalization)

activation_63 (Activation)  (None, 16, 16, 256)       0

conv2d_33 (Conv2D)          (None, 16, 16, 256)       65536

batch_normalization_64 (Ba  (None, 16, 16, 256)       1024
tchNormalization)

activation_64 (Activation)  (None, 16, 16, 256)       0

depthwise_conv2d_31 (Depth  (None, 8, 8, 256)         2304
wiseConv2D)

batch_normalization_65 (Ba  (None, 8, 8, 256)         1024
tchNormalization)

activation_65 (Activation)  (None, 8, 8, 256)         0

conv2d_34 (Conv2D)          (None, 8, 8, 512)         131072

batch_normalization_66 (Ba  (None, 8, 8, 512)         2048
tchNormalization)

activation_66 (Activation)  (None, 8, 8, 512)         0
```

```
depthwise_conv2d_32 (Depth  (None, 8, 8, 512)         4608
wiseConv2D)

batch_normalization_67 (Ba  (None, 8, 8, 512)         2048
tchNormalization)

activation_67 (Activation)  (None, 8, 8, 512)         0

conv2d_35 (Conv2D)          (None, 8, 8, 512)         262144

batch_normalization_68 (Ba  (None, 8, 8, 512)         2048
tchNormalization)

activation_68 (Activation)  (None, 8, 8, 512)         0

depthwise_conv2d_33 (Depth  (None, 8, 8, 512)         4608
wiseConv2D)

batch_normalization_69 (Ba  (None, 8, 8, 512)         2048
tchNormalization)

activation_69 (Activation)  (None, 8, 8, 512)         0

conv2d_36 (Conv2D)          (None, 8, 8, 512)         262144

batch_normalization_70 (Ba  (None, 8, 8, 512)         2048
tchNormalization)

activation_70 (Activation)  (None, 8, 8, 512)         0

depthwise_conv2d_34 (Depth  (None, 8, 8, 512)         4608
wiseConv2D)

batch_normalization_71 (Ba  (None, 8, 8, 512)         2048
tchNormalization)

activation_71 (Activation)  (None, 8, 8, 512)         0

conv2d_37 (Conv2D)          (None, 8, 8, 512)         262144

batch_normalization_72 (Ba  (None, 8, 8, 512)         2048
tchNormalization)

activation_72 (Activation)  (None, 8, 8, 512)         0

depthwise_conv2d_35 (Depth  (None, 8, 8, 512)         4608
wiseConv2D)

batch_normalization_73 (Ba  (None, 8, 8, 512)         2048
tchNormalization)

activation_73 (Activation)  (None, 8, 8, 512)         0

conv2d_38 (Conv2D)          (None, 8, 8, 512)         262144

batch_normalization_74 (Ba  (None, 8, 8, 512)         2048
tchNormalization)

activation_74 (Activation)  (None, 8, 8, 512)         0

depthwise_conv2d_36 (Depth  (None, 8, 8, 512)         4608
wiseConv2D)

batch_normalization_75 (Ba  (None, 8, 8, 512)         2048
tchNormalization)

activation_75 (Activation)  (None, 8, 8, 512)         0

conv2d_39 (Conv2D)          (None, 8, 8, 512)         262144

batch_normalization_76 (Ba  (None, 8, 8, 512)         2048
tchNormalization)

activation_76 (Activation)  (None, 8, 8, 512)         0

depthwise_conv2d_37 (Depth  (None, 4, 4, 512)         4608
wiseConv2D)

batch_normalization_77 (Ba  (None, 4, 4, 512)         2048
tchNormalization)

activation_77 (Activation)  (None, 4, 4, 512)         0

conv2d_40 (Conv2D)          (None, 4, 4, 1024)        524288

batch_normalization_78 (Ba  (None, 4, 4, 1024)        4096
tchNormalization)
```

```
activation_78 (Activation)   (None, 4, 4, 1024)        0

depthwise_conv2d_38 (Depth   (None, 4, 4, 1024)        9216
wiseConv2D)

batch_normalization_79 (Ba   (None, 4, 4, 1024)        4096
tchNormalization)

activation_79 (Activation)   (None, 4, 4, 1024)        0

conv2d_41 (Conv2D)           (None, 4, 4, 1024)        1048576

batch_normalization_80 (Ba   (None, 4, 4, 1024)        4096
tchNormalization)

activation_80 (Activation)   (None, 4, 4, 1024)        0

global_average_pooling2d_2   (None, 1024)              0
 (GlobalAveragePooling2D)

dense_2 (Dense)              (None, 8)                 8200

=================================================================
Total params: 3237064 (12.35 MB)
Trainable params: 3215176 (12.26 MB)
Non-trainable params: 21888 (85.50 KB)
```

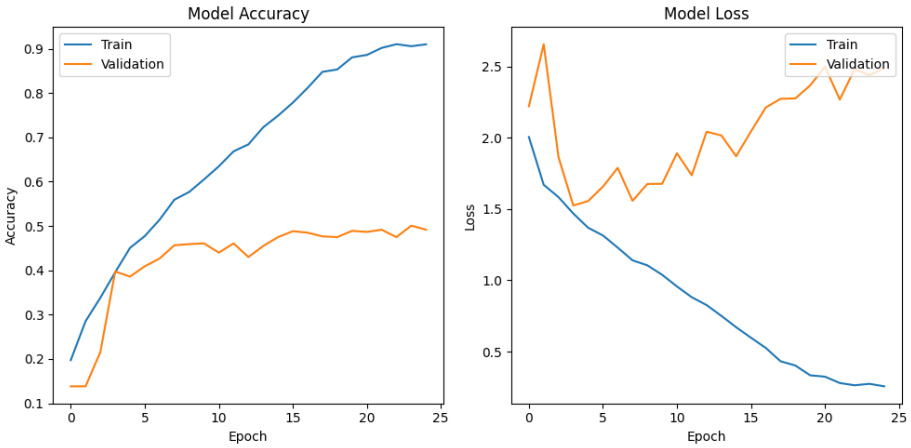
معماری MobileNet:

- بلوک‌های پیچیدگی عمقی: این بلوک‌ها برای استخراج ویژگی‌ها استفاده می‌شوند، این بلوک‌ها از پیچیدگی‌های قابل تفکیک عمیق و به دنبال نرمال‌سازی دسته‌ای و فعال‌سازی ReLU تشکیل شده‌اند.
- پیچیدگی های راه راه: در فواصل زمانی برای نمونه برداری از نقشه های ویژگی استفاده می شود.
- مقیاس بندی مدل: پارامتر آلفا عرض کانال های شبکه را مقیاس می کند.
- لایه خروجی: یک لایه متراکم با فعال سازی softmax، که پیش بینی هایی را در ۸ کلاس ایجاد می کند.

این معماری بر پیچیدگی‌های عمقی سبک وزن، کاهش پیچیدگی محاسباتی و در عین حال حفظ قدرت بیانی، برای محیط‌های موبایل و محدود به منابع، تأکید دارد.

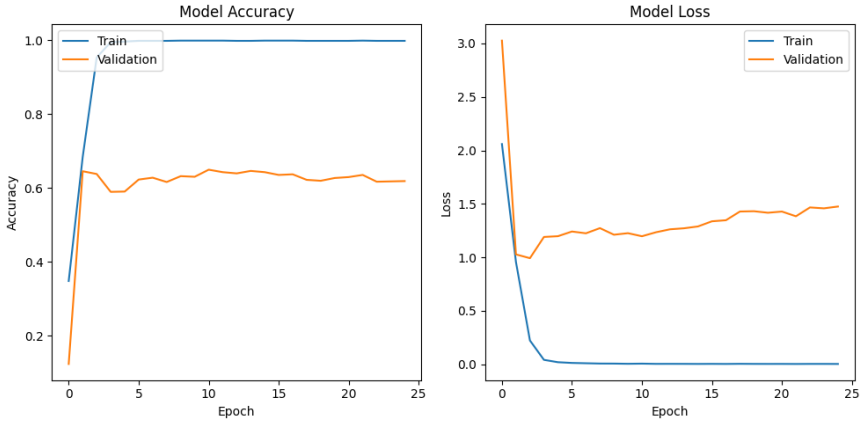
حال این مدل را با پیرکبندی‌های معرفی شده در سوال یک برای ۵۰ ایپاک آموزش می‌دهیم.

- نمودار دقت و خطا:



شکل ۱۴ نمودار دقت و خطا برای داده‌های آموزش مدل **MobileNet**

همچنین بعد از fine tuning نتیجه‌ی این مدل به شکل زیر خواهد بود:



شکل ۱۵ نمودار دقت و خطا برای داده‌های آموزش مدل **MobileNet**

این مدل با حجم کمترش نسبت به دو مدل قبلی به شدت سریع‌تر همگرا می‌شد. همچنین بعد از tuning دقت ۴۹ درصد روی داده‌های ارزیابی می‌داد که قابل مقایسه با مدل‌های قبلی می‌باشد.

```
38/38 [==============================] - 1s 18ms/step - loss: 2.4902
- accuracy: 0.4975
213/213 [==============================] - 4s 20ms/step - loss: 0.3200
- accuracy: 0.8878
```

AlexNet: این مدل به طور قابل توجهی با AlexNet متفاوت است، که دارای لایه های کانولوشن با حداکثر تجمع و لایه های متراکم است. AlexNet از پیچیدگی‌های استاندارد و عملیات ادغام بدون جداسازی پیچیدگی‌های عمقی و نقطه‌ای که در اینجا دیده می‌شود، استفاده می‌کند.

VGGNet: VGGNet شامل یک سری از لایه‌های کانولوشن استاندارد است که هر بلوک کانولوشن دارای چندین کانولوشن ۳×۳ است که به دنبال آن لایه‌های max-pooling قرار دارند. در مقابل، مدل ارائه شده

کانولوشن های قابل تفکیک عمیق را انتخاب می کند و هدف آن کاهش پیچیدگی محاسباتی با جداسازی اطلاعات فضایی و بین کانالی است.

مزایا مدل:

کارایی: پیچیدگی‌های قابل تفکیک عمیق، تعداد پارامترها و بار محاسباتی را کاهش می‌دهند و مدل را کارآمدتر می‌کنند، به خصوص در تنظیمات تلفن همراه یا محدودیت منابع.

قدرت بازنمایی: این مدل در حالی که کارآمد است، ممکن است در مقایسه با معماری‌های عمیق‌تری مانند VGGNet، بازنمایی کم‌عمق‌تری داشته باشد، که ممکن است بر توانایی آن در گرفتن ویژگی‌های بسیار پیچیده تأثیر بگذارد.

تفاوت این مدل با دو مدل قبلی در حجم کوچکتر آن نسبت به آن مدل‌ها است. و همچنین این مدل لایه‌ی *DepthwiseConv2D* را در تفاوت با مدل‌های قبلی دارد.

پاسخ ۳ – تشخیص بیماران مبتلا به کووید با استفاده از عکس ریه

۳-۱. معرفی مقاله

در تلاش برای شناسایی سریع و دقیق COVID-19، تصویربرداری با اشعه ایکس به عنوان یک ابزار تشخیصی ارزشمند برجسته می شود. این مقاله بر استفاده از تصاویر اشعه ایکس برای تشخیص ناهنجاری‌های ظریف ریوی مشخصه موارد COVID-19 از افراد دارای شرایط ریوی طبیعی تمرکز دارد. با استفاده از یادگیری ماشین و تخصص رادیولوژیکی، هدف مطالعه ما توسعه مدل‌های طبقه‌بندی قوی است که به شناسایی غیرتهاجمی و به موقع کمک می‌کند. ما بر ادغام تکنیک‌های یادگیری عمیق با دانش حوزه تأکید می‌کنیم و پتانسیل همکاری بین رشته‌ای را برای تقویت قابلیت‌های تشخیصی در بیماری‌های عفونی برجسته می‌کنیم.

۳-۲ جمع‌آوری داده و پیش پردازش تصاویر

ابتدا داده‌ها را به دو دسته‌ی ارزیابی و آموزش تقسیم می‌کنیم. یعنی ۰.۱۵ داده‌ها را برای ارزیابی کنار می‌گذاریم. سپس از آنجا که داده‌های ما کم هست نیاز داریم، روش‌هایی برای تعمیم داده استفاده کنیم. در مقاله‌ی معرفی شده، چهار نوع روش تعمیم داده شامل:

- چرخش ۹۰ درجه
- چرخش ۱۸۰ درجه

- چرخش ۲۷۰ درجه
- Flipping

حال معماری گفته شده در مقاله را پیاده‌سازی می‌کنیم که به شکل زیر خواهد بود:

جدول**8** معماری **MobileNet**

Layer (type)	Output Shape	Param #
conv2d_20 (Conv2D)	(None, 150, 150, 64)	1792
max_pooling2d_20 (MaxPooling2D)	(None, 75, 75, 64)	0
batch_normalization_28 (BatchNormalization)	(None, 75, 75, 64)	256
activation_20 (Activation)	(None, 75, 75, 64)	0
dropout_20 (Dropout)	(None, 75, 75, 64)	0
conv2d_21 (Conv2D)	(None, 75, 75, 128)	73856
max_pooling2d_21 (MaxPooling2D)	(None, 37, 37, 128)	0
batch_normalization_29 (BatchNormalization)	(None, 37, 37, 128)	512
activation_21 (Activation)	(None, 37, 37, 128)	0
dropout_21 (Dropout)	(None, 37, 37, 128)	0
conv2d_22 (Conv2D)	(None, 37, 37, 256)	295168
max_pooling2d_22 (MaxPooling2D)	(None, 18, 18, 256)	0
batch_normalization_30 (BatchNormalization)	(None, 18, 18, 256)	1024
activation_22 (Activation)	(None, 18, 18, 256)	0
dropout_22 (Dropout)	(None, 18, 18, 256)	0
conv2d_23 (Conv2D)	(None, 18, 18, 512)	1180160
max_pooling2d_23 (MaxPooling2D)	(None, 9, 9, 512)	0
batch_normalization_31 (BatchNormalization)	(None, 9, 9, 512)	2048

```
activation_23 (Activation)   (None, 9, 9, 512)         0

dropout_23 (Dropout)         (None, 9, 9, 512)         0

conv2d_24 (Conv2D)           (None, 9, 9, 512)         2359808

max_pooling2d_24 (MaxPooli   (None, 4, 4, 512)         0
ng2D)

batch_normalization_32 (Ba   (None, 4, 4, 512)         2048
tchNormalization)

activation_24 (Activation)   (None, 4, 4, 512)         0

dropout_24 (Dropout)         (None, 4, 4, 512)         0

flatten_4 (Flatten)          (None, 8192)              0

dense_12 (Dense)             (None, 512)               4194816

batch_normalization_33 (Ba   (None, 512)               2048
tchNormalization)

dense_13 (Dense)             (None, 256)               131328

batch_normalization_34 (Ba   (None, 256)               1024
tchNormalization)

dense_14 (Dense)             (None, 1)                 257

=================================================================
Total params: 8246145 (31.46 MB)
Trainable params: 8241665 (31.44 MB)
Non-trainable params: 4480 (17.50 KB)
```

- هر لایه کانولوشن شامل:

حداکثر ادغام (MaxPooling2D) برای نمونه برداری از نقشه های ویژگی.

نرمال سازی دسته ای (BatchNormalization) برای تثبیت.

فعال سازی ReLU (فعال سازی('relu')) برای معرفی غیر خطی.

Dropout (Dropout(0.2)) برای جلوگیری از برازش بیش از حد.

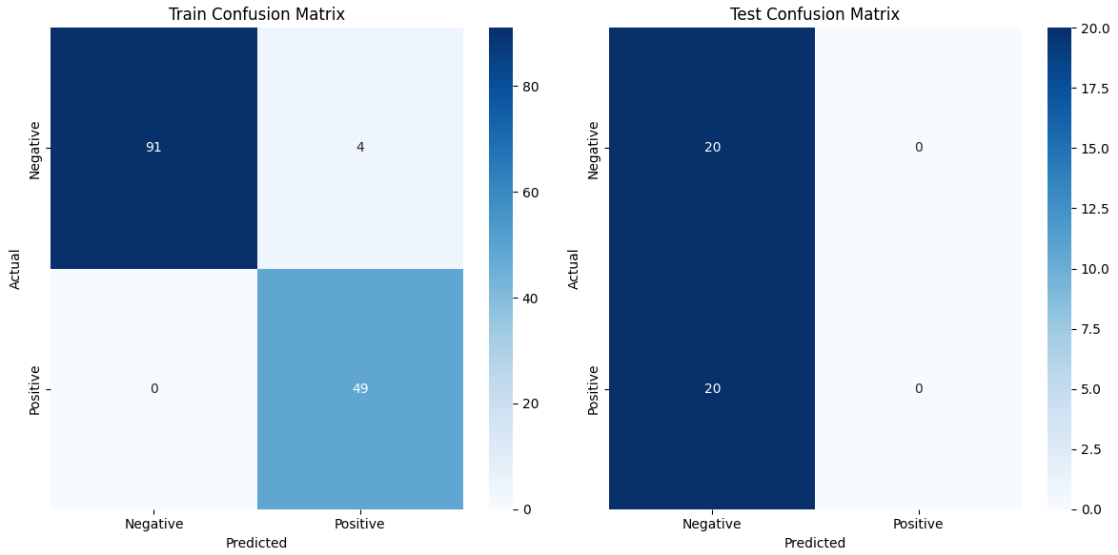
- لایه Flatten: خروجی از لایه های کانولوشن را به یک آرایه ۱ بعدی مسطح می کند.
- لایه های متراکم: متشکل از دو لایه پنهان Dense(512)، (activation='relu') و Dense(256, activation='relu') با فعال سازی ReLU.
- نرمال سازی دسته ای پس از هر لایه متراکم اعمال می شود.

- لایه خروجی: لایه نهایی با یک نورون و فعال سازی سیگموئید برای کارهای طبقه بندی باینری.

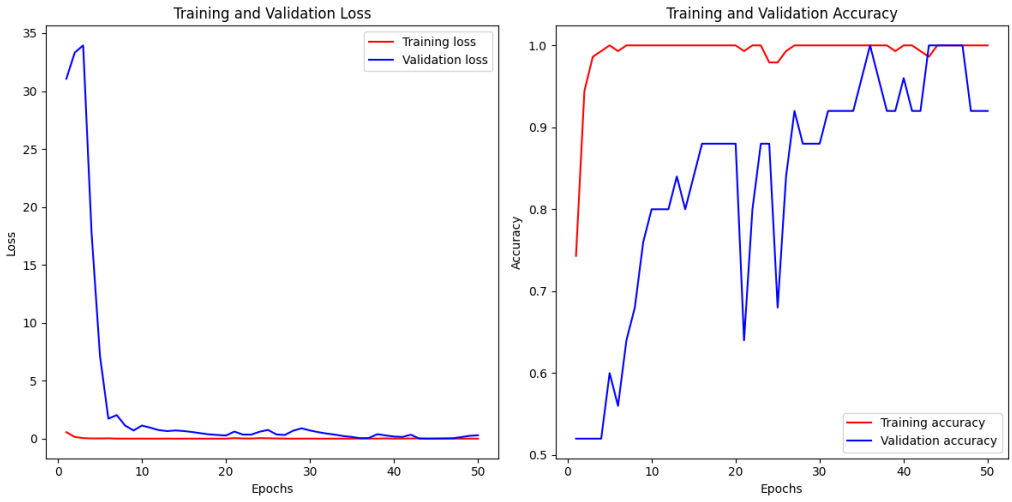
در ابتدا به ترتیب شروع به تعمیم داده‌ها می‌کنیم و نتایج بدست آمده را با یکدگیر مقایسه می‌کنیم:

- مدل با داده‌های اصلی و چرخش ۹۰ درجه:

ماتریس درهمریختگی:



دقت و خطا:



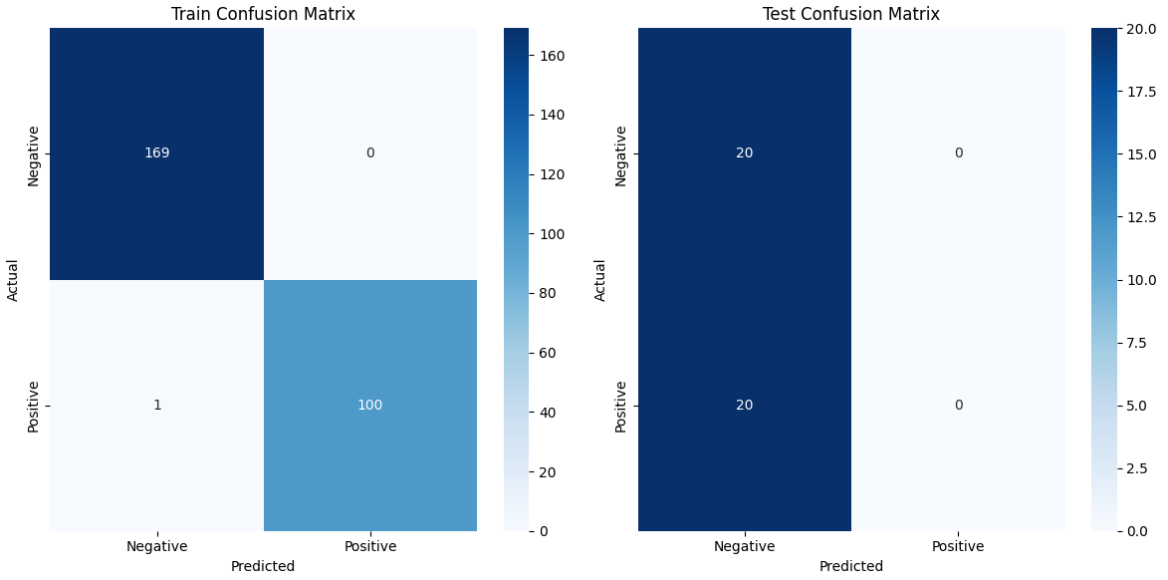
جدول**9** عملکرد شبکه **MobileNet** برای یک **Augmentation**

```
Classification Report for train:
              precision    recall  f1-score   support

           0       1.00      0.96      0.98        95
           1       0.92      1.00      0.96        49

    accuracy                           0.97       144
```

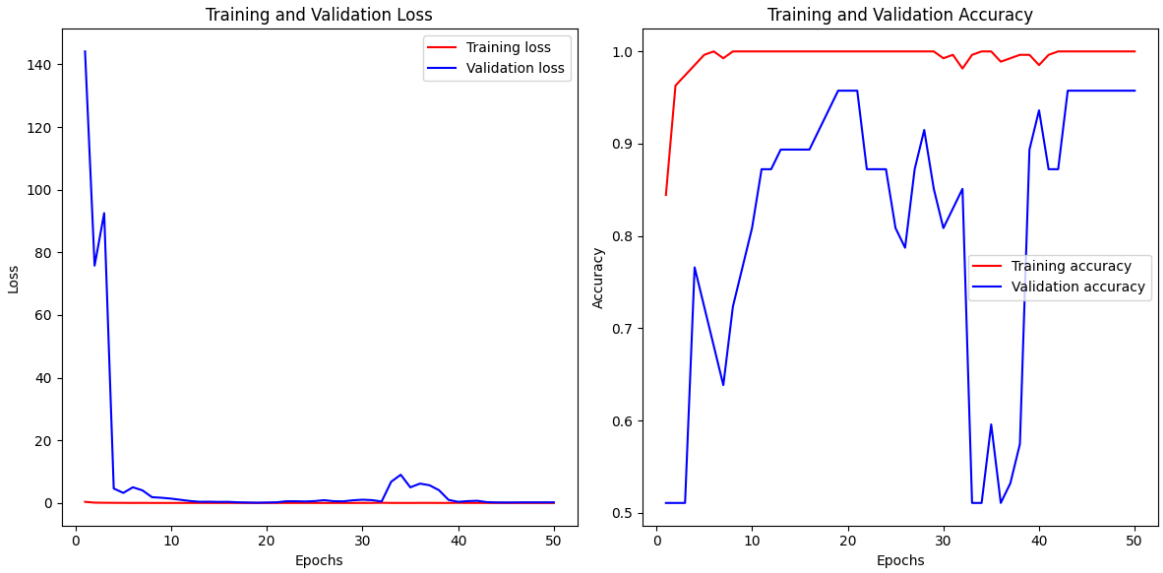
```
   macro avg       0.96      0.98      0.97       144
weighted avg       0.97      0.97      0.97       144

Train's Specificity: 0.9578947368421052

Classification Report for test:
              precision    recall  f1-score   support

           0       0.50      1.00      0.67        20
           1       0.00      0.00      0.00        20

    accuracy                           0.50        40
   macro avg       0.25      0.50      0.33        40
weighted avg       0.25      0.50      0.33        40

Test's Specificity: 1.0
```

- مدل با داده‌های اصلی و چرخش ۹۰ و ۱۸۰ درجه:

ماتریس درهمریختگی:



دقت و خطا:

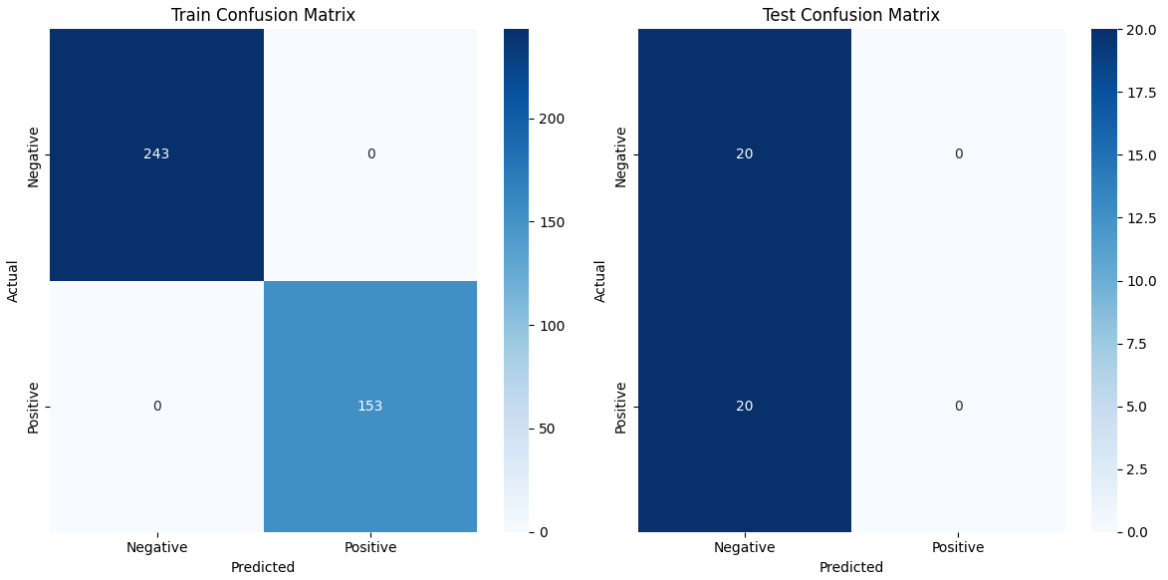


جدول**10** عملکرد شبکه **MobileNet** برای دو **Augmentation**

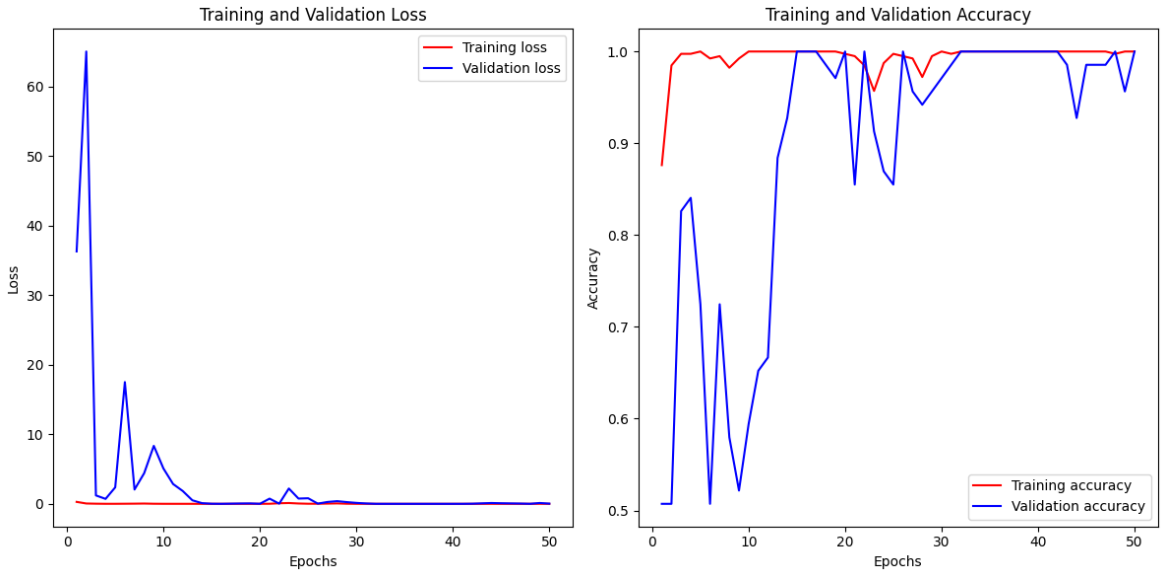
```
Classification Report for train:
              precision    recall  f1-score   support

           0       0.99      1.00      1.00       169
           1       1.00      0.99      1.00       101

    accuracy                           1.00       270
   macro avg       1.00      1.00      1.00       270
weighted avg       1.00      1.00      1.00       270

Train's Specificity: 1.0

Classification Report for test:
              precision    recall  f1-score   support

           0       0.50      1.00      0.67        20
           1       0.00      0.00      0.00        20

    accuracy                           0.50        40
   macro avg       0.25      0.50      0.33        40
weighted avg       0.25      0.50      0.33        40

Test's Specificity: 1.0
```

- مدل با داده‌های اصلی و چرخش ۹۰، ۱۸۰ و ۲۷۰ درجه:

ماتریس درهمریختگی:



دقت و خطا:



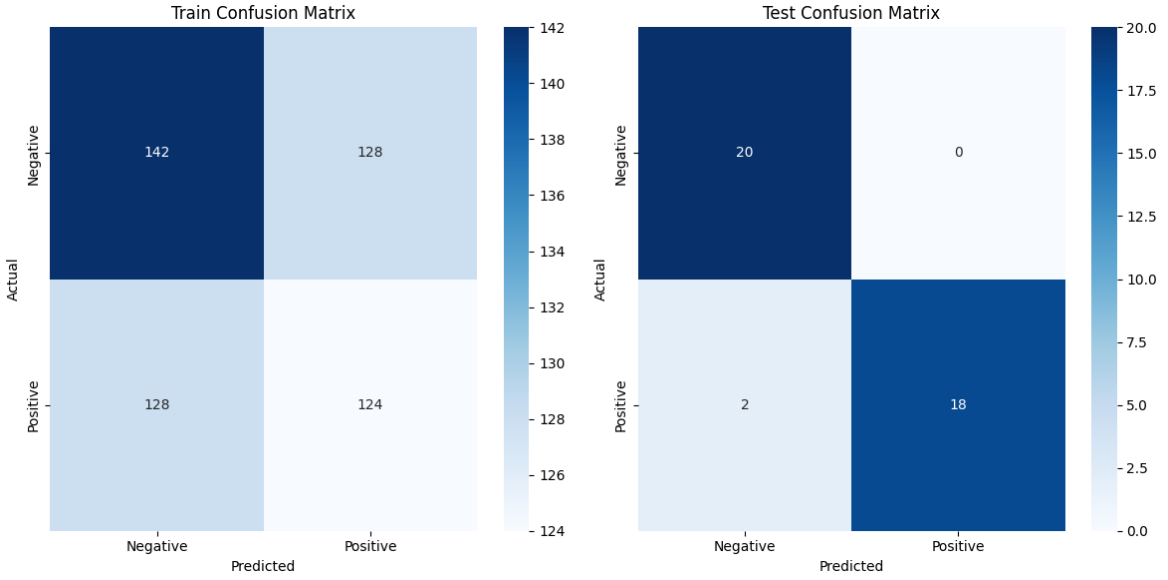
جدول ۱۱ عملکرد شبکه **MobileNet** برای سه **Augmentation**

```
Classification Report for train:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       243
           1       1.00      1.00      1.00       153

    accuracy                           1.00       396
   macro avg       1.00      1.00      1.00       396
weighted avg       1.00      1.00      1.00       396

Train's Specificity: 1.0

Classification Report for test:
              precision    recall  f1-score   support
```

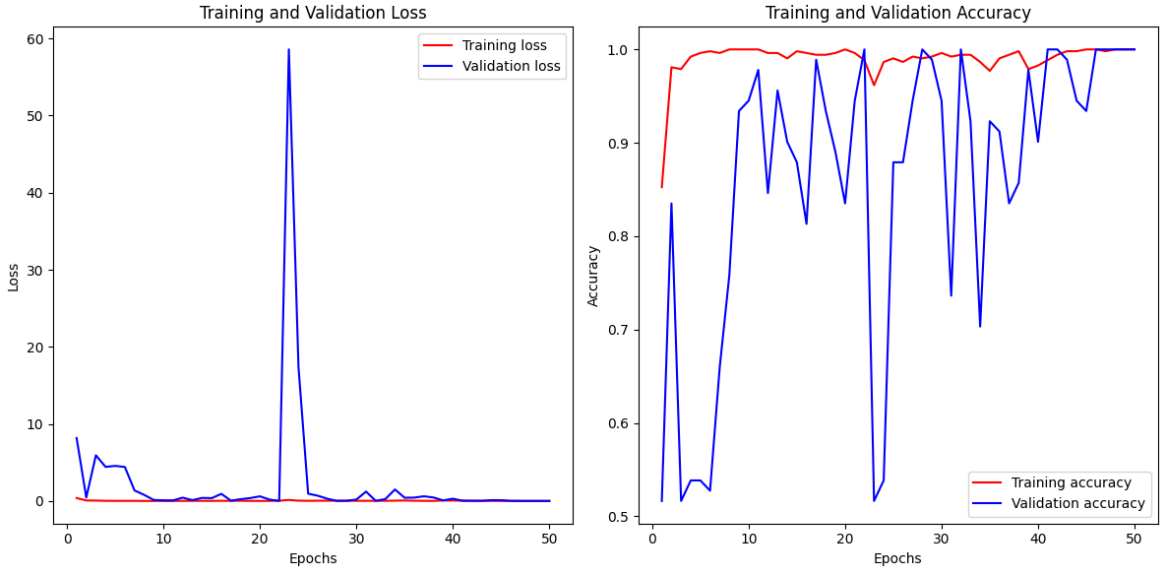
```
              0       0.50      1.00      0.67        20
              1       0.00      0.00      0.00        20

    accuracy                            0.50        40
   macro avg        0.25      0.50      0.33        40
weighted avg        0.25      0.50      0.33        40

Test's Specificity: 1.0
```

- مدل با داده‌های اصلی و چرخش ۹۰، ۱۸۰، ۲۷۰ درجه و Flipping:

ماتریس درهمریختگی:



دقت و خطا:



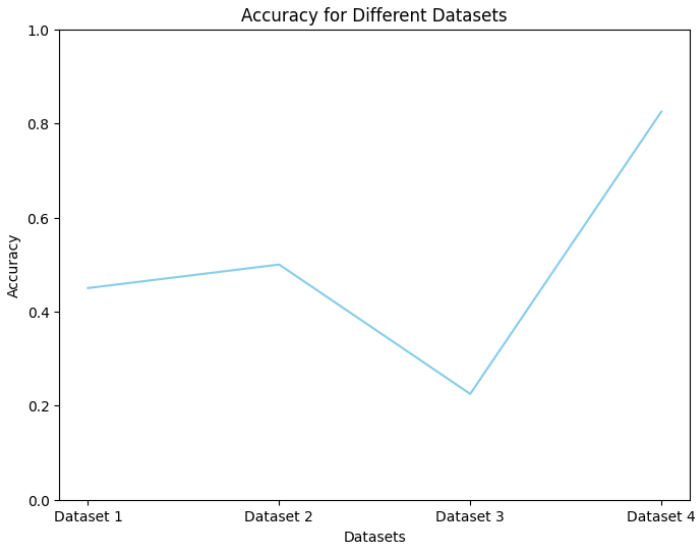
جدول ۱۲ عملکرد شبکه **MobileNet** برای چهار **Augmentation**

```
Classification Report for train:
              precision    recall  f1-score   support

           0       0.53      0.53      0.53       270
           1       0.49      0.49      0.49       252

    accuracy                           0.51       522
   macro avg       0.51      0.51      0.51       522
weighted avg       0.51      0.51      0.51       522

Train's Specificity: 0.5259259259259259

Classification Report for test:
              precision    recall  f1-score   support

           0       0.91      1.00      0.95        20
           1       1.00      0.90      0.95        20

    accuracy                           0.95        40
   macro avg       0.95      0.95      0.95        40
weighted avg       0.95      0.95      0.95        40

Test's Specificity: 1.0
```

نتیجه‌گیری:

با استفاده از Augmentation می‌توان داده‌ها را افزایش داده که باعث مزایایی همچون:

- ایجاد داده های متنوع تر برای آموزش، کاهش بیش از حد مناسب.
- آموزش شبکه برای تشخیص الگوها با وجود تغییرات..
- افزایش توانایی مدل برای تعمیم به داده های جدید و دیده نشده.



شکل ۱۶ افزایش **augmentation**

همانطور که مشخص است با افزایش augmentationها مدل رو به سمت بهتر شدن و تعمیم‌پذیر تر شدن رفته است.

۵-۳ ارزیابی شبکه:

حال شبکه‌هایی با تعداد لایه‌های مختلف برای این وظیفه آموزش می‌دهیم:

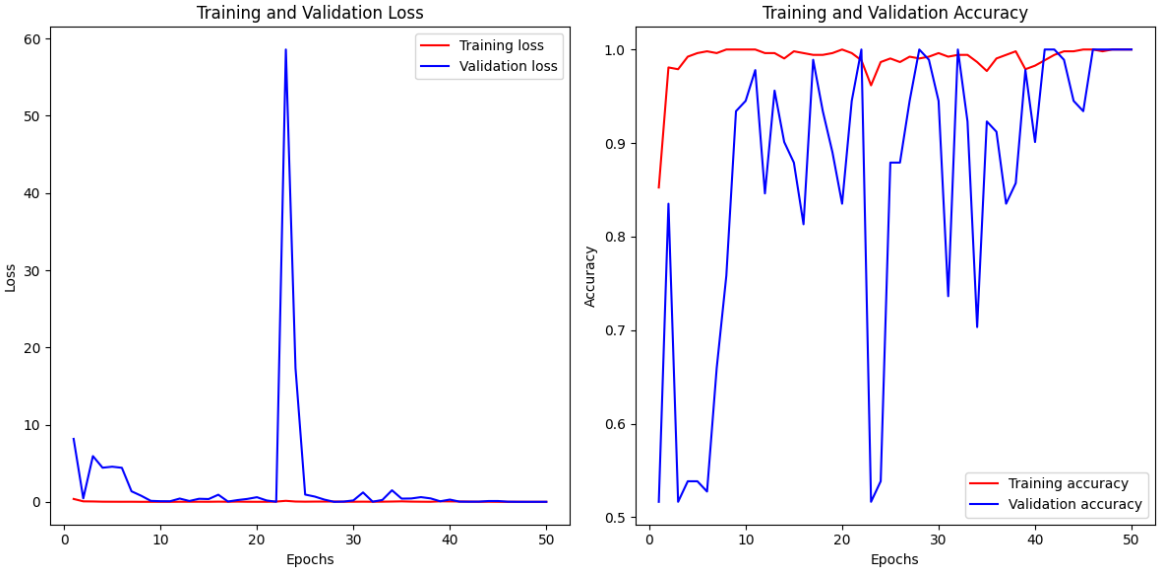
- یک لایه کانولوشنال:

جدول ۱۳ معماری شبکه **MobileNet** با یک لایه

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_40 (Conv2D)          (None, 150, 150, 64)      1792

 max_pooling2d_40 (MaxPooli  (None, 75, 75, 64)        0
 ng2D)

 batch_normalization_56 (Ba  (None, 75, 75, 64)        256
 tchNormalization)

 activation_40 (Activation)  (None, 75, 75, 64)        0

 dropout_40 (Dropout)        (None, 75, 75, 64)        0

 flatten_8 (Flatten)         (None, 360000)            0

 dense_24 (Dense)            (None, 512)               184320512

 batch_normalization_57 (Ba  (None, 512)               2048
 tchNormalization)
```

```
 dense_25 (Dense)            (None, 256)               131328

 batch_normalization_58 (Ba  (None, 256)               1024
 tchNormalization)

 dense_26 (Dense)            (None, 1)                 257

=================================================================
Total params: 184457217 (703.65 MB)
Trainable params: 184455553 (703.64 MB)
                                Non-trainable params: 1664 (6.50 KB)
```

نمودار دقت و خطای:



شکل ۱۷ نمودار دقت و خطا با یک لایه

همچنین دقت مدل برای داده‌ی تست و آموزش به شکل زیر می‌باشد:

```
loss: 0.1306 - accuracy: 0.9500
loss: 0.0014 - accuracy: 1.0000
```

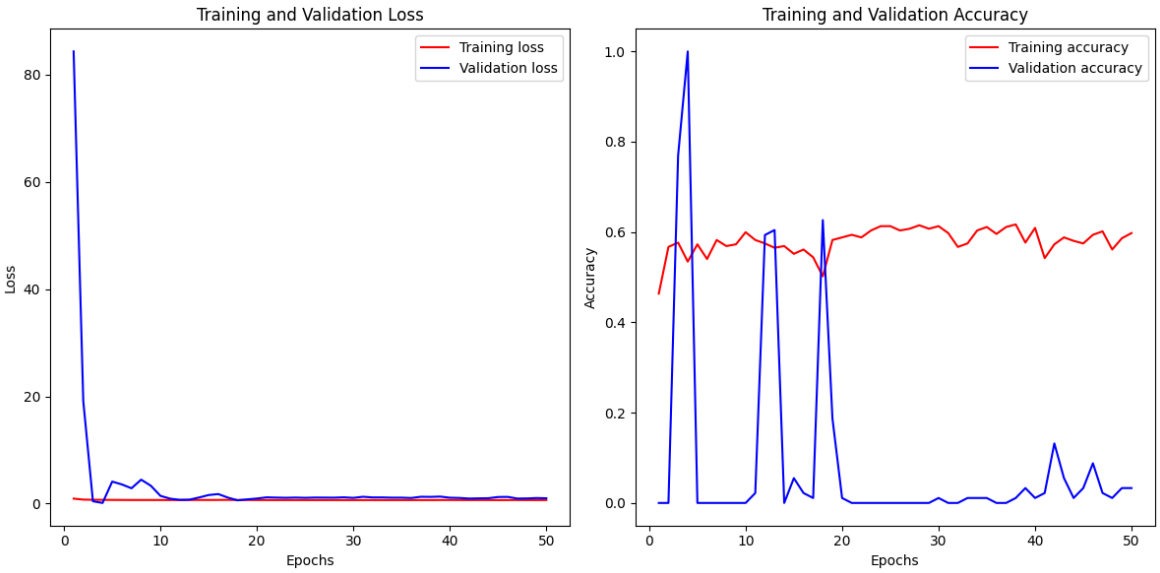
- دو لایه کانولشنال:

جدول ۱۴ معماری شبکه **MobileNet** با دو لایه

```
Layer (type)                Output Shape              Param #
=================================================================
 conv2d_41 (Conv2D)          (None, 150, 150, 64)      1792

 max_pooling2d_41 (MaxPooli  (None, 75, 75, 64)        0
 ng2D)

 batch_normalization_59 (Ba  (None, 75, 75, 64)        256
 tchNormalization)

 activation_41 (Activation)  (None, 75, 75, 64)        0
```

```
 dropout_41 (Dropout)        (None, 75, 75, 64)        0

 conv2d_42 (Conv2D)          (None, 75, 75, 128)       73856

 max_pooling2d_42 (MaxPooli  (None, 37, 37, 128)       0
 ng2D)

 batch_normalization_60 (Ba  (None, 37, 37, 128)       512
 tchNormalization)

 activation_42 (Activation)  (None, 37, 37, 128)       0

 dropout_42 (Dropout)        (None, 37, 37, 128)       0

 flatten_9 (Flatten)         (None, 175232)            0

 dense_27 (Dense)            (None, 512)               89719296

 batch_normalization_61 (Ba  (None, 512)               2048
 tchNormalization)

 dense_28 (Dense)            (None, 256)               131328

 batch_normalization_62 (Ba  (None, 256)               1024
 tchNormalization)

 dense_29 (Dense)            (None, 1)                 257

=================================================================
Total params: 89930369 (343.06 MB)
Trainable params: 89928449 (343.05 MB)
                                Non-trainable params: 1920 (7.50 KB)
```

نمودار دقت و خطای:



شکل ۱۸ نمودار دقت و خطا با دو لایه

همچنین دقت مدل برای داده‌ی تست و آموزش به شکل زیر می‌باشد:

```
loss: 0.0126 - accuracy: 1.0000
loss: 0.1456 - accuracy: 0.9655
```

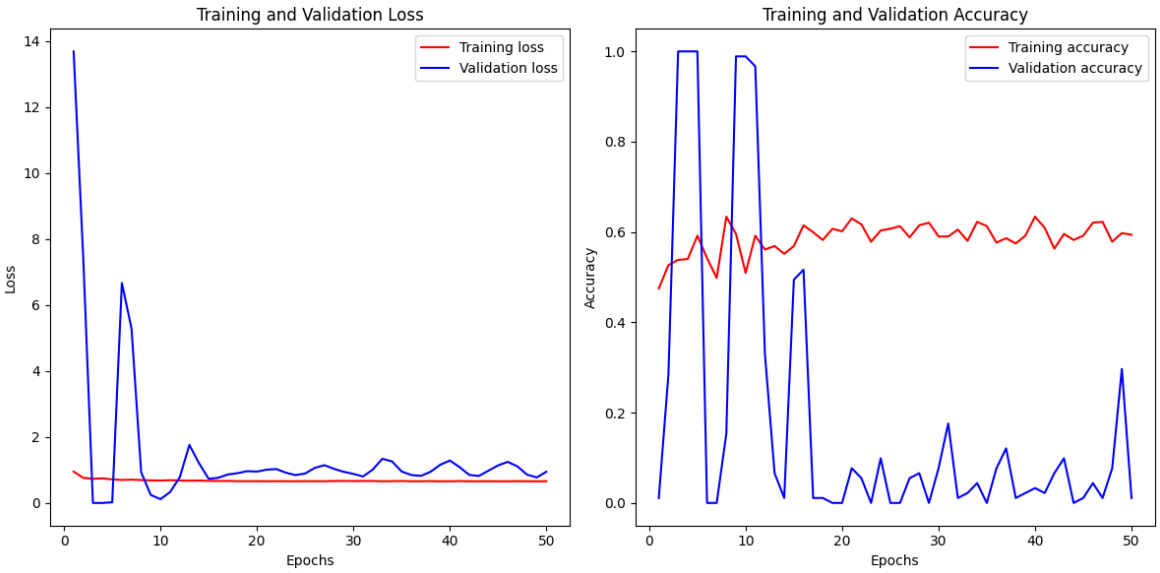
- سه لایه کانولوشنال:

جدول ۱۵ معماری شبکه **MobileNet** با سه لایه

Layer (type)	Output Shape	Param #
conv2d_43 (Conv2D)	(None, 150, 150, 64)	1792
max_pooling2d_43 (MaxPooling2D)	(None, 75, 75, 64)	0
batch_normalization_63 (BatchNormalization)	(None, 75, 75, 64)	256
activation_43 (Activation)	(None, 75, 75, 64)	0
dropout_43 (Dropout)	(None, 75, 75, 64)	0
conv2d_44 (Conv2D)	(None, 75, 75, 128)	73856
max_pooling2d_44 (MaxPooling2D)	(None, 37, 37, 128)	0
batch_normalization_64 (BatchNormalization)	(None, 37, 37, 128)	512
activation_44 (Activation)	(None, 37, 37, 128)	0

```
dropout_44 (Dropout)         (None, 37, 37, 128)       0

conv2d_45 (Conv2D)           (None, 37, 37, 256)       295168

max_pooling2d_45 (MaxPooli   (None, 18, 18, 256)       0
ng2D)

batch_normalization_65 (Ba   (None, 18, 18, 256)       1024
tchNormalization)

activation_45 (Activation)   (None, 18, 18, 256)       0

dropout_45 (Dropout)         (None, 18, 18, 256)       0

flatten_10 (Flatten)         (None, 82944)             0

dense_30 (Dense)             (None, 512)               42467840

batch_normalization_66 (Ba   (None, 512)               2048
tchNormalization)

dense_31 (Dense)             (None, 256)               131328

batch_normalization_67 (Ba   (None, 256)               1024
tchNormalization)

dense_32 (Dense)             (None, 1)                 257

=================================================================
Total params: 42975105 (163.94 MB)
Trainable params: 42972673 (163.93 MB)
                                Non-trainable params: 2432 (9.50 KB)
```

نمودار دقت و خطا:



شکل ۱۹ نمودار دقت و خطا با سه لایه

همچنین دقت مدل برای داده‌ی تست و آموزش به شکل زیر می‌باشد:

```
loss: 0.7348 - accuracy: 0.6250
loss: 0.7085 - accuracy: 0.5575
```

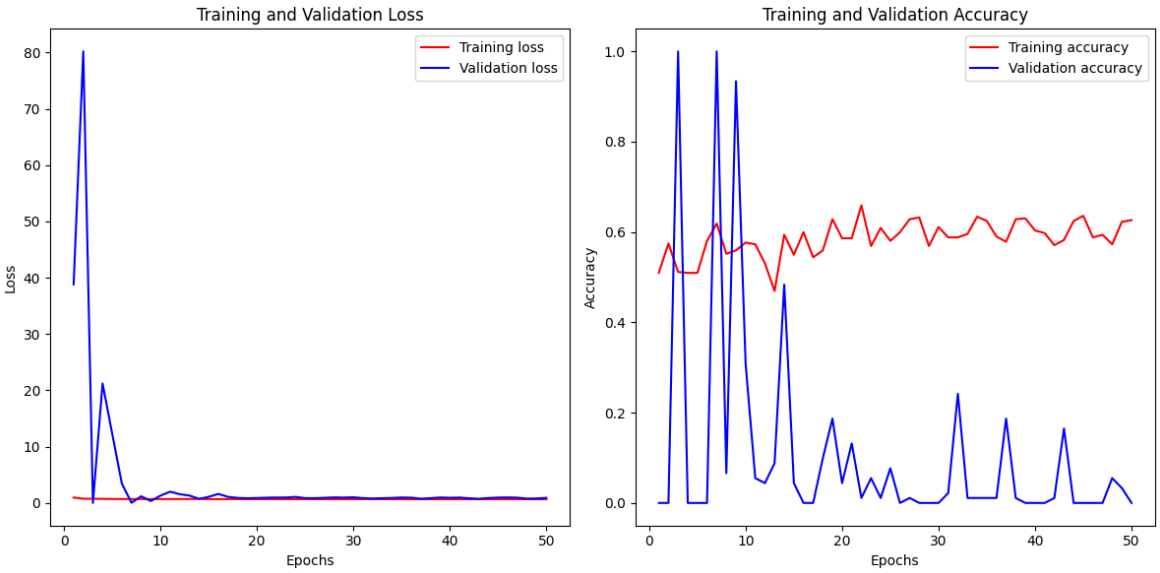
- چهار لایه کانولوشنال:

جدول ۱۶ معماری شبکه **MobileNet** با چهار لایه

conv2d_46 (Conv2D)	(None, 150, 150, 64)	1792
max_pooling2d_46 (MaxPooling2D)	(None, 75, 75, 64)	0
batch_normalization_68 (BatchNormalization)	(None, 75, 75, 64)	256
activation_46 (Activation)	(None, 75, 75, 64)	0
dropout_46 (Dropout)	(None, 75, 75, 64)	0
conv2d_47 (Conv2D)	(None, 75, 75, 128)	73856
max_pooling2d_47 (MaxPooling2D)	(None, 37, 37, 128)	0
batch_normalization_69 (BatchNormalization)	(None, 37, 37, 128)	512
activation_47 (Activation)	(None, 37, 37, 128)	0
dropout_47 (Dropout)	(None, 37, 37, 128)	0
conv2d_48 (Conv2D)	(None, 37, 37, 256)	295168
max_pooling2d_48 (MaxPooling2D)	(None, 18, 18, 256)	0
batch_normalization_70 (BatchNormalization)	(None, 18, 18, 256)	1024
activation_48 (Activation)	(None, 18, 18, 256)	0
dropout_48 (Dropout)	(None, 18, 18, 256)	0
conv2d_49 (Conv2D)	(None, 18, 18, 512)	1180160
max_pooling2d_49 (MaxPooling2D)	(None, 9, 9, 512)	0
batch_normalization_71 (BatchNormalization)	(None, 9, 9, 512)	2048
activation_49 (Activation)	(None, 9, 9, 512)	0
dropout_49 (Dropout)	(None, 9, 9, 512)	0
flatten_11 (Flatten)	(None, 41472)	0
dense_33 (Dense)	(None, 512)	21234176

batch_normalization_72 (BatchNormalization)	(None, 512)	2048
dense_34 (Dense)	(None, 256)	131328
batch_normalization_73 (BatchNormalization)	(None, 256)	1024
dense_35 (Dense)	(None, 1)	257

نمودار دقت و خطا:



شکل ۲۰ نمودار دقت و خطا با چهار لایه

همچنین دقت مدل برای داده‌ی تست و آموزش به شکل زیر می‌باشد:

```
loss: 0.7616 - accuracy: 0.4250
loss: 0.6600 - accuracy: 0.6054
```

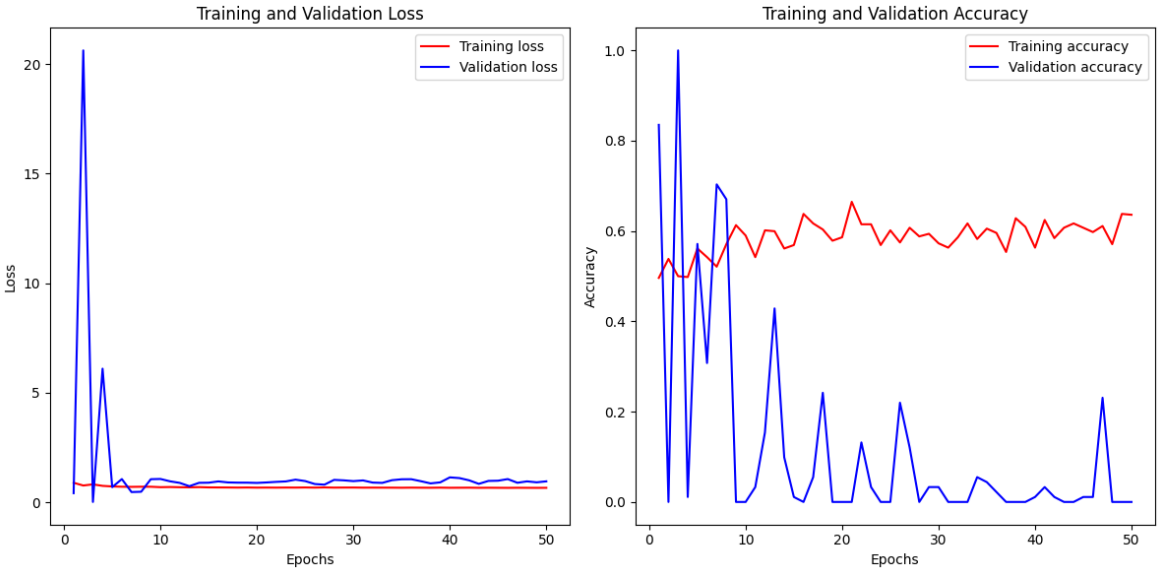
- پنج لایه کانولوشنال:

جدول ۱۷ معماری شبکه **MobileNet** با پنج لایه

conv2d_50 (Conv2D)	(None, 150, 150, 64)	1792
max_pooling2d_50 (MaxPooling2D)	(None, 75, 75, 64)	0
batch_normalization_74 (BatchNormalization)	(None, 75, 75, 64)	256
activation_50 (Activation)	(None, 75, 75, 64)	0
dropout_50 (Dropout)	(None, 75, 75, 64)	0
conv2d_51 (Conv2D)	(None, 75, 75, 128)	73856

```
max_pooling2d_51 (MaxPooli  (None, 37, 37, 128)       0
ng2D)

batch_normalization_75 (Ba  (None, 37, 37, 128)       512
tchNormalization)

activation_51 (Activation)  (None, 37, 37, 128)       0

dropout_51 (Dropout)        (None, 37, 37, 128)       0

conv2d_52 (Conv2D)          (None, 37, 37, 256)       295168

max_pooling2d_52 (MaxPooli  (None, 18, 18, 256)       0
ng2D)

batch_normalization_76 (Ba  (None, 18, 18, 256)       1024
tchNormalization)

activation_52 (Activation)  (None, 18, 18, 256)       0

dropout_52 (Dropout)        (None, 18, 18, 256)       0

conv2d_53 (Conv2D)          (None, 18, 18, 512)       1180160

max_pooling2d_53 (MaxPooli  (None, 9, 9, 512)         0
ng2D)

batch_normalization_77 (Ba  (None, 9, 9, 512)         2048
tchNormalization)

activation_53 (Activation)  (None, 9, 9, 512)         0

dropout_53 (Dropout)        (None, 9, 9, 512)         0

conv2d_54 (Conv2D)          (None, 9, 9, 512)         2359808

max_pooling2d_54 (MaxPooli  (None, 4, 4, 512)         0
ng2D)

batch_normalization_78 (Ba  (None, 4, 4, 512)         2048
tchNormalization)

activation_54 (Activation)  (None, 4, 4, 512)         0

dropout_54 (Dropout)        (None, 4, 4, 512)         0

flatten_12 (Flatten)        (None, 8192)              0

dense_36 (Dense)            (None, 512)               4194816

batch_normalization_79 (Ba  (None, 512)               2048
tchNormalization)

dense_37 (Dense)            (None, 256)               131328

batch_normalization_80 (Ba  (None, 256)               1024
tchNormalization)

dense_38 (Dense)            (None, 1)                 257
```

نمودار دقت و خطا:



شکل ۲۱ نمودار دقت و خطا با چهار لایه

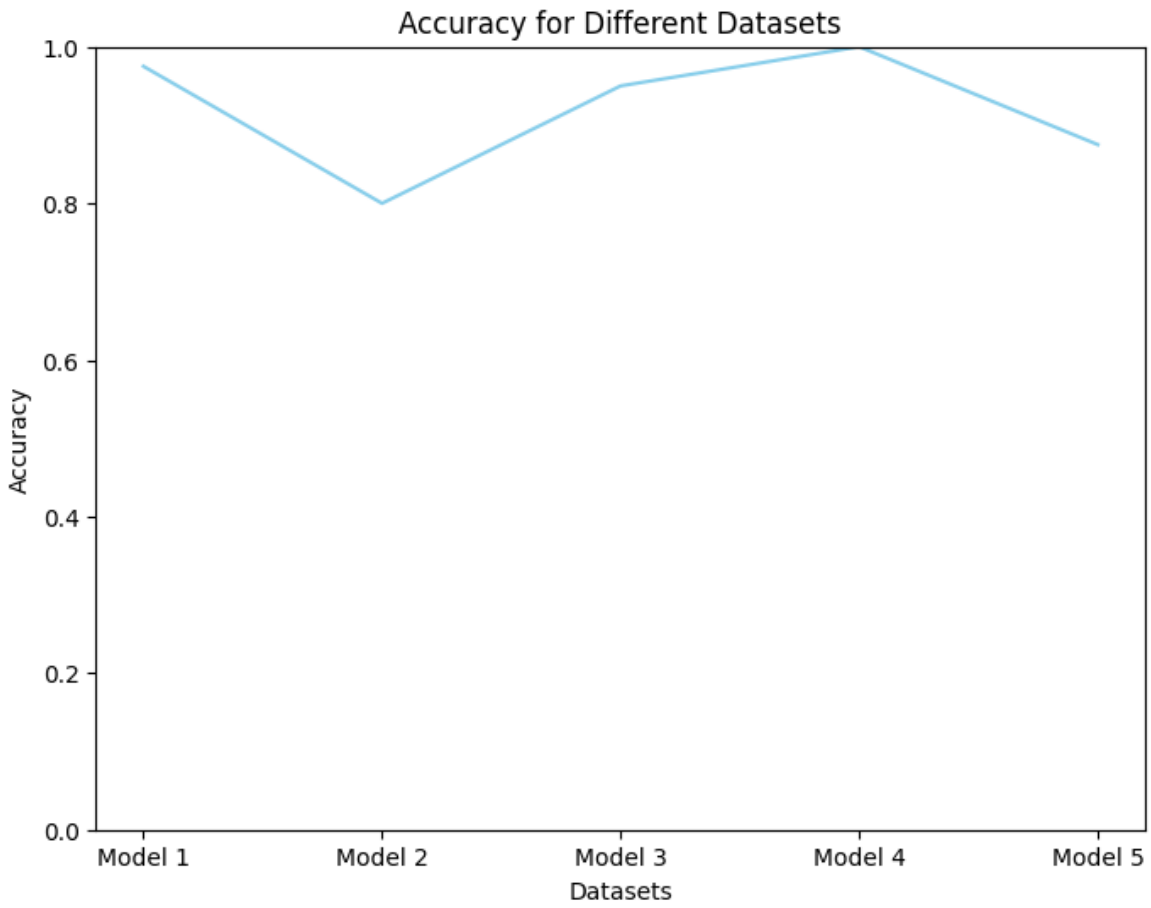
همچنین دقت مدل برای داده‌ی تست و آموزش به شکل زیر می‌باشد:

```
loss: 0.6848 - accuracy: 0.4750
loss: 0.6532 - accuracy: 0.6054
```

- شش لایه کانولوشنال:

همان مدل اصلی آموزش داده‌ی ما دارای ۶ لایه می‌باشد.

با توجه به تعداد لایه‌های مختلف مشخص است که بعضی از مدل‌ها به شدت ساده هستند و قابلیت تعمیم یادگیری پترن‌های داده‌ها را به خوبی ندارند. همچنین با افزایش لایه‌های مدل، سرعت overfit شدن به داده‌های آموزش افزایش می‌یابد و قابلیت تعمیم‌پذیری مدل کاهش می‌یابد.



شکل ۲۲ دقت مدل وابسته به تعداد لایه‌ها

لذا باید مدلی را با تعداد لایه نه بسیار زیاد و نه بسیار کم انتخاب کرد که هم قابلیت تعمیم‌پذیری مناسبی داشته باشد و هم قابلیت یادگیری داده‌های موجود را داشته باشد.