**تمرین ۲ – عرب‌زاده (810102206) و فرتوت(؟)**

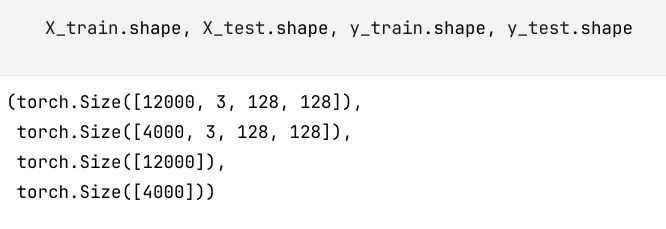
**۲-۱)**

برای Augmentation برای هر عکس یک عکس اضافی در نظر می‌گیریم (به دلیل کمبود حافظه). عکس‌ها را از نوع uint8 در نظر می‌گیریم تا کمترین حافظه ممکن را بگیرند و آن‌ها را تک به تک از دیسک می‌خوانیم.

روی هر عکس دو transform تعریف می‌کنیم تا تبدیل عکس به Tensor انجام گیرد، که common\_transform برای عکس‌های عادی و تغییر نیافته و augmented\_transform برای ایجاد دیتا مضاعف است.

augmented\_transform به این‌گونه تعریف شده است که برای هر عکس حول محور z تا 20 درجه ساعتگرد یا پادساعتگرد به صورت رندوم چرخیده شود، تا اندازه 10 درصد طول و 10 درصد عرض تصویر shift پیدا کند (translation) و در نهایت حول محور x خوابانده شود.

عکس‌ها تبدیل می‌شوند و به داده آموزش و تست تقیسم می‌شوند به نسبت 2 به 1 و اندازه آن‌ها به صورت زیر است:

در نهایت برای به تصویر کشیدن عکس‌های خوانده شده برای اطمینان از درست خواندن آن‌ها، آن‌ها را Plot می‌کنیم:



**۱-۲)**

عکس‌ها را با جزییاتی که گفته شد می‌خوانیم و برای تسریع خواندن آن‌ها در دفعات بعدی اجرا، یک فایل cache از tensor عکس‌ها روی دیسک می نویسیم که آن را مستقیم بخوانیم.

Batch size را طبق فرضیات مقاله برابر ۴۰۰ در نظر می‌گیریم.

معماری مدل را توصیف مقاله در نظر می‌گیریم:

Conv2D → ReLU →Conv2D → ReLU → BatchNorm2D → MaxPool2D → Dropout2D

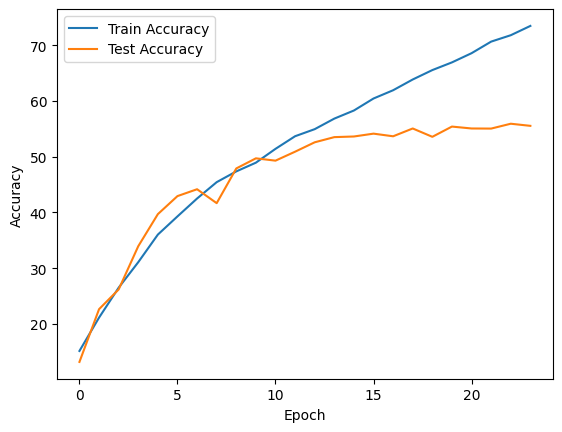
که دسته بالا بایستی 5 دفعه تکرار شود تا لایه‌های Convolutional را تشکیل دهد. در مرحله بعد باید Flatten قرار گیرد تا لایه‌ی Fully Connected آن را تحویل بگیرد و طبقه‌بندی انجام شود.

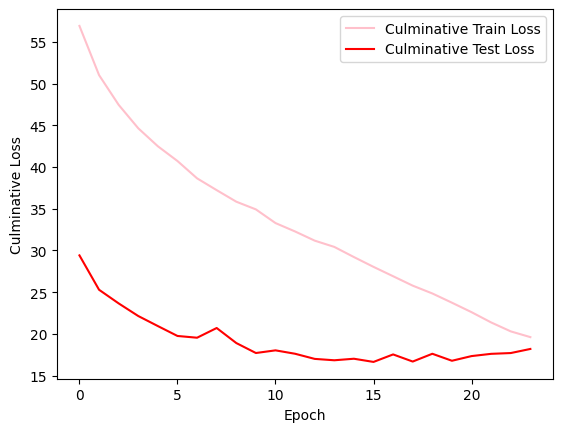
وارد فاز آموزش می‌شویم. اگر قبلاً Train کرده‌ایم،مدلی که روی دیسک ذخیره کرده‌ایم را می‌خوانیم (برای انجام Train فایل مدل را پاک کنید) و آموزش را skip می‌کنیم.

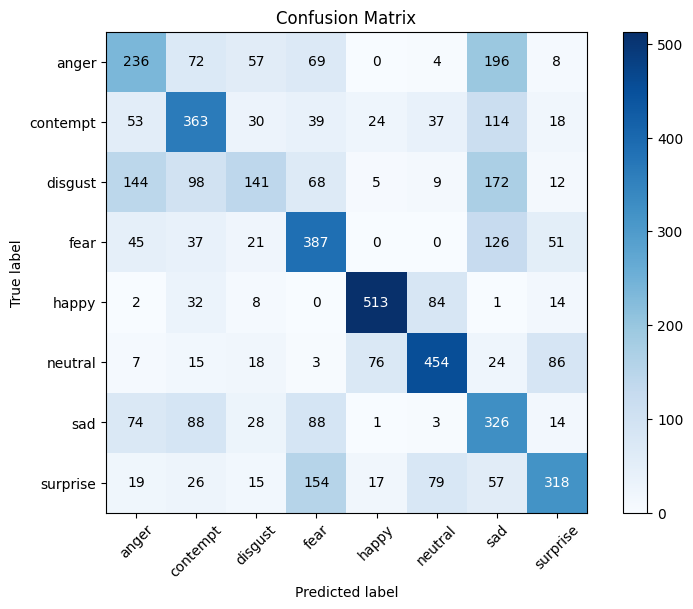
پیکربندی آموزش به روال مقاله است: تعداد دور ها 24 است، optimizer از Adam استفاده می‌کند و سرعت یاد‌گیری ۰.۰۰۱ است.

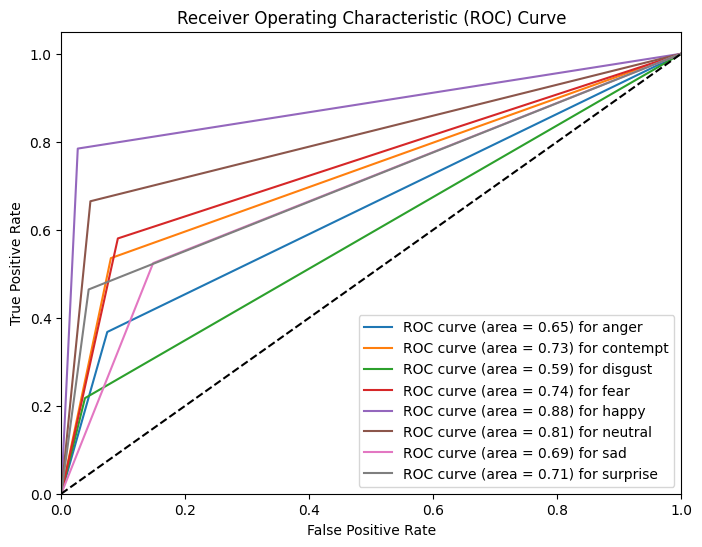
در هر دسته، ورودی از uint8 به float تبدیل می‌شود تا شبکه بتواند روی آن محاسبه گرادیان انجام دهد. تخمین ها از softmax رد می‌شوند و محتمل ترین آن‌ها (بیشترین) انتخاب می‌شود. در آخر دسته tensor ها را detach می‌کنیم تا گرادیان آن‌ها از محاسبات حذف شود. در پایان هر epoch هم تست را به صورت مرسوم انجام می‌دهیم. دقت های تست و آموزش در لیستی ریخته شده تا نمودارشان ترسیم گردد. بعد از آموزش، مدل را روی دیسک ذخیره می‌کنیم.

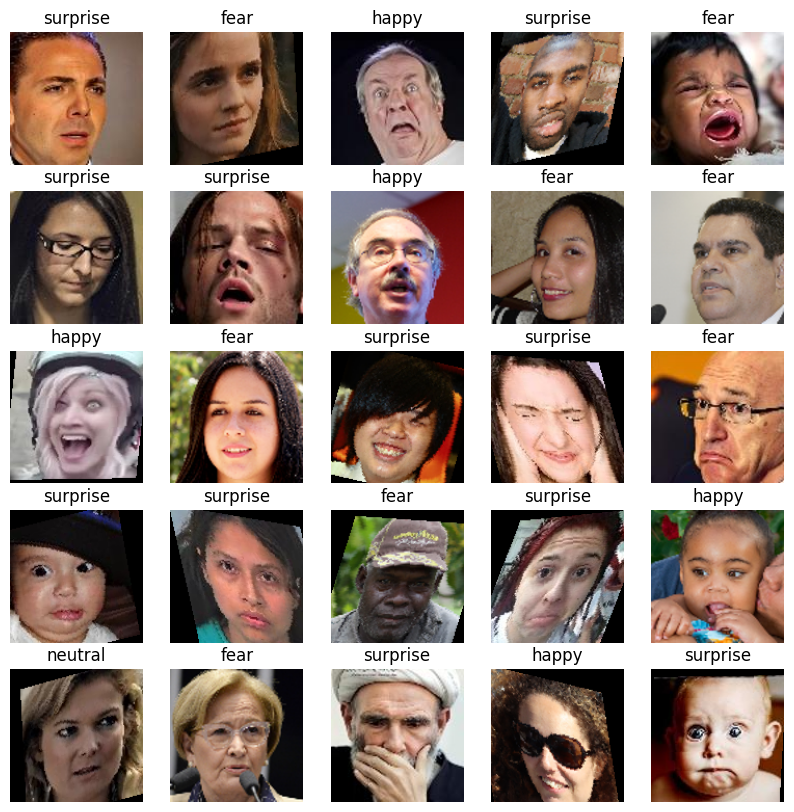
در سلول بعد آموزش، نمودار ها وجود دارد:



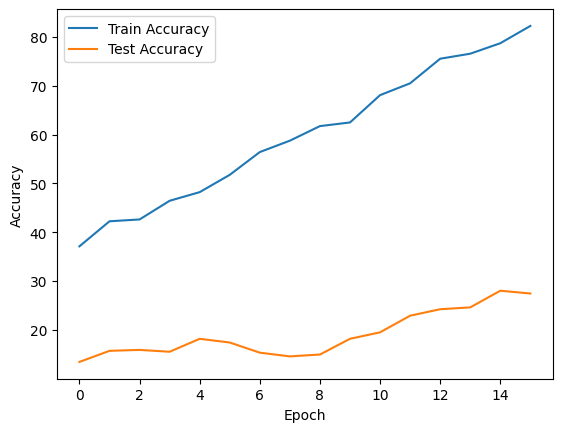
در سلول بعد ماتریس آشفتگی داده تست رسم می‌شود:

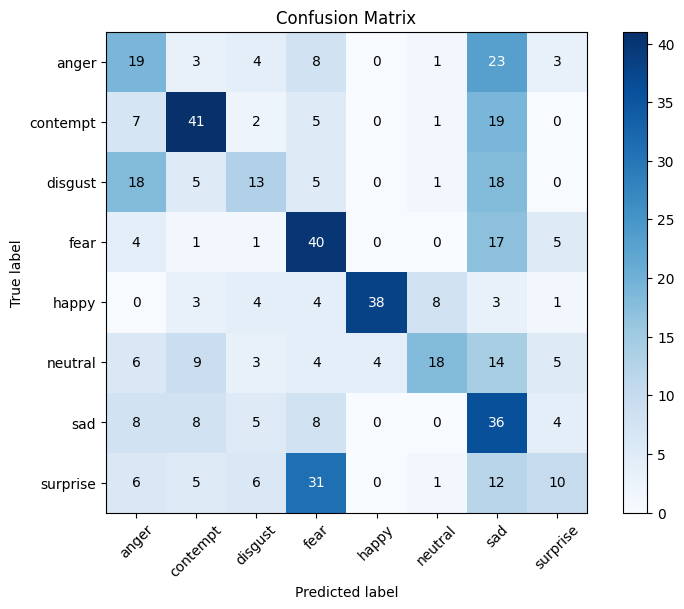
و نمودار ROC هم به صورت زیر است:

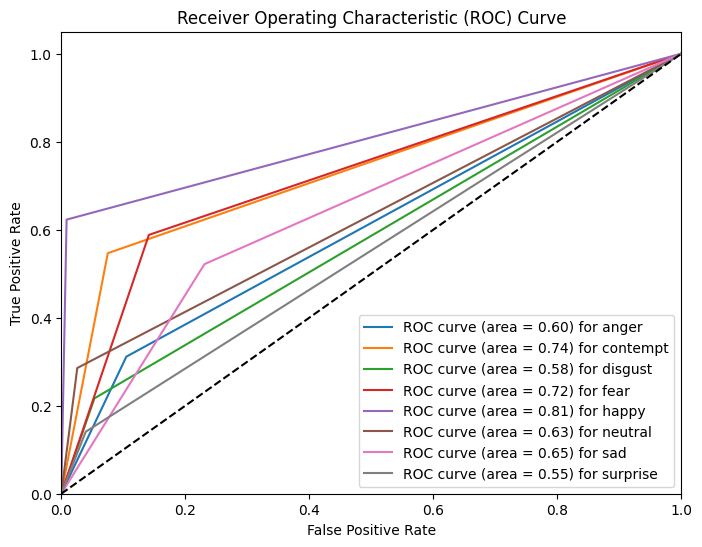
و در نهایت عمل‌کرد تصویری مدل Train شده را بررسی می‌کنیم:



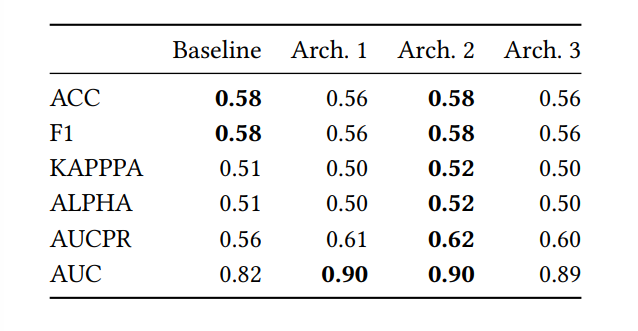
پروسه بالا فقط برای آموزش شبکه بود و نه finetune آن، پروسه finetune آن کاملاً مشابه آموزش است و نمودار های آن به شرح زیر است(مشاهده می‌شود که عمل‌کرد مدل با این finetune تضعیف شده است):







در قیاس بین VGGNet و ‌AlexNet، همانطور که نمودار زیر نشان می‌دهد معماری VGG برتری کلی ای نسبت به AlexNet دارد:

دقت کلی VGG دو درصد بیشتر از AlexNet است و در سایر مترکس ها نیز پیشرو است.

دلائل عمل‌کرد بهتری که VGG ارائه می‌دهد : که لایه‌های Convolutional آن دو برابر AlexNet است و همچنین فیلتر های VGG با سایز ۳x۳ ریز-دانه تر ویژگی را استخراج می‌کنند و در نتیجه Feature extraction بیشتری به کار می‌گیرد.