به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین دوم

نام و نام خانوادگی	محمدرضا نعمتی
شماره دانشجویی	810100226
نام و نام خانوادگی	محمدامین یوسفی
شماره دانشجویی	810100236

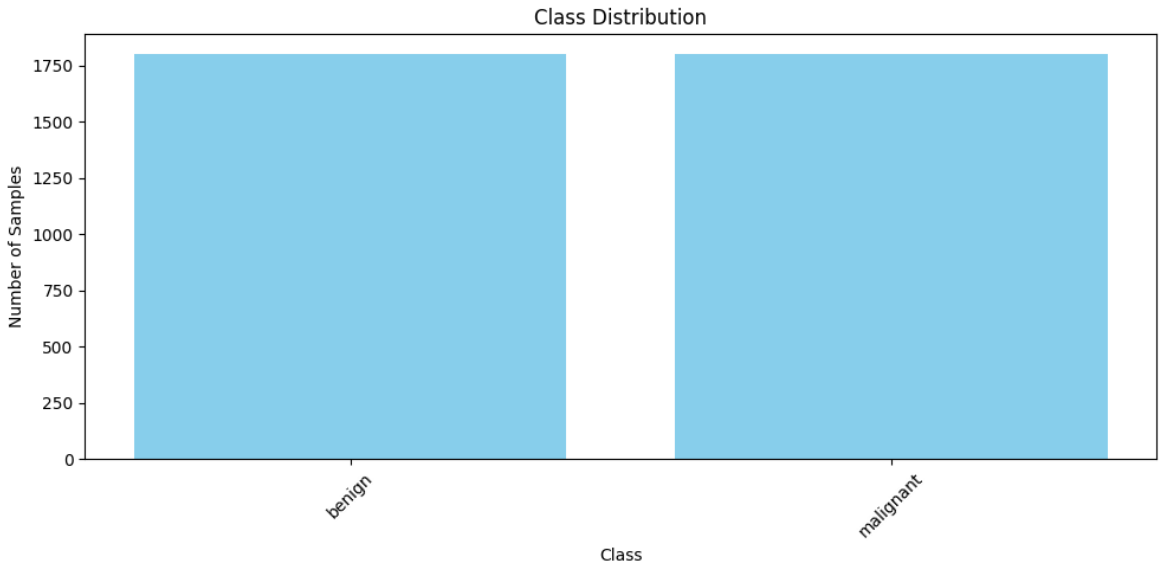
فهرست

شکل‌ها و جدول‌ها

پرسش 1. تشخیص ضایعه سرطانی با استفاده از CNN

۱-۲. پیش پردازش تصاویر

- **تغییر سایز تصاویر:** تغییر اندازه تصاویر برای مدل‌های CNN ضروری است زیرا این مدل‌ها نیاز به ورودی‌هایی با ابعاد یکسان دارند. این یکنواختی در ابعاد به مدل اجازه می‌دهد تا ساختار معماری خود را حفظ کند و فیلترها و عملیات‌های pooling به درستی اعمال شوند. همچنین، با استانداردسازی ورودی‌ها، شبکه قادر به استخراج ویژگی‌ها به‌صورت سیستماتیک از تمام نمونه‌ها می‌شود. بدون این مرحله، شبکه ممکن است با مشکلاتی در پردازش تصاویر با ابعاد مختلف مواجه شود، که می‌تواند منجر به کاهش کارایی مدل شود. ما ابعاد تصاویر بر اساس مدل مطرح شده در مقاله، 28 در 28 قرار دادیم.
- **تغییر سایز تصاویر:** نرمال‌سازی داده‌ها در یادگیری عمیق با مقیاس‌بندی مقادیر پیکسل‌ها به بازه $[0,1][0, 1]$ به بهبود فرآیند آموزش کمک می‌کند. این کار با کاهش واریانس مقادیر ورودی، همگرایی مدل را تسریع می‌کند و باعث پایداری گرادیان‌ها در طول بک‌پراپگیشن می‌شود. نرمال‌سازی همچنین به مدل کمک می‌کند تا ویژگی‌های مرتبط‌تری را از تصاویر استخراج کند و عملکرد کلی بهتری داشته باشد. در نتیجه، فرآیند یادگیری شبکه‌های عصبی کارآمدتر و دقت نهایی مدل بیشتر می‌شود. در این پروژه ما با همه پیکسل ها را بر 255 تقسیم کردیم.
- **متعادل کردن تعداد داده‌ها:** متعادل‌سازی داده‌ها نقش مهمی در مقابله با مشکل عدم توازن داده‌ها دارد، که می‌تواند باعث سوگیری مدل به سمت کلاس‌های پرجمعیت شود. این کد از تکنیک oversampling استفاده می‌کند تا تعداد نمونه‌های کلاس‌های کم‌تعداد را به تعادل با کلاس‌های پرجمعیت برساند.



شکل 1-1. توزیع داده‌ها پس از معادل سازی داده‌ها

```
Dataset Mean Colors (R, G, B): [0.7122631072998047, 0.508866548538208, 0.5083397030830383]
Dataset Std Dev Colors (R, G, B): [0.17689144611358643, 0.15635590255260468, 0.16519539058208466]
```

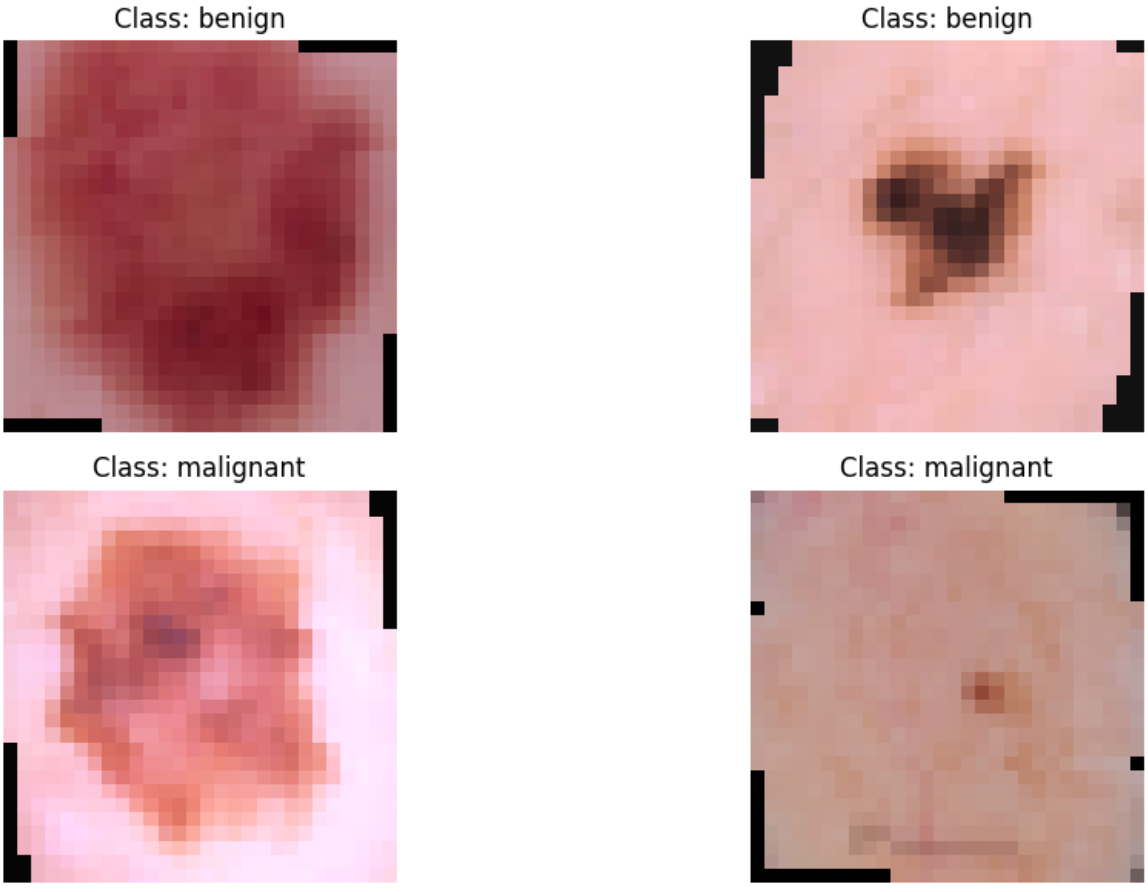
شکل 1-2. میانگین پیکسل‌ها پس از نرمال‌سازی داده‌ها

۱-۳. داده افزایی(Data augmentation)

Data augmentation به مدل‌های یادگیری عمیق کمک می‌کند تا عملکرد بهتری در مواجهه با داده‌های جدید و دیده‌نشده داشته باشند. با اعمال تغییراتی مانند چرخش، تغییر مقیاس، وارونه‌سازی، برش، و تنظیم روشنایی، تنوع داده‌ها افزایش می‌یابد، که این امر خطر بیش‌برازش (overfitting) را کاهش می‌دهد. همچنین، این روش موجب می‌شود مدل ویژگی‌های عمومی‌تری از داده‌ها یاد بگیرد و دقت و پایداری آن در شرایط مختلف بهبود یابد. این فرآیند بخشی کلیدی از پیش‌پردازش است و زمینه‌ای قوی برای آموزش مؤثر مدل فراهم می‌کند.

- **معکوس سازی افقی تصاویر:** از `RandomHorizontalFlip` برای چرخاندن تصاویر به صورت افقی با احتمال 30٪ استفاده شده است. این کار جهت‌گیری‌های مختلف توده‌ها را شبیه‌سازی می‌کند و تنوع داده‌ها را افزایش می‌دهد.
- **معکوس سازی عمودی تصاویر:** با RandomVerticalFlip، تصاویر به صورت عمودی و با احتمال 30٪ معکوس می‌شوند. این تکنیک، مقاومت مدل در برابر تغییرات زاویه‌ای را تقویت می‌کند.

- **چرخش تصادفی تصاویر:** RandomRotation تصاویر را در محدوده مثبت منفی می‌چرخاند. این عملیات تغییرات جزئی زاویه‌ای را شبیه‌سازی کرده و تعمیم‌پذیری مدل را بهبود می‌بخشد.
- **اعمال تغییرات هندسی:** RandomAffine تغییرات هندسی یا کج کردن تصاویر (shear) را تا حداکثر 10 درجه اعمال می‌کند. این کار به مدل کمک می‌کند تا با تحریفات هندسی کوچک در داده‌ها بهتر کنار بیاید.
- **تغییر روشنایی و کنتراست تصاویر:** ColorJitter مقدار روشنایی و کنتراست تصاویر را به صورت تصادفی تغییر می‌دهد. این عملیات شرایط نوری متغیر در داده‌های واقعی را شبیه‌سازی کرده و قابلیت تعمیم مدل را افزایش می‌دهد.



شکل 1-3. نمونه‌ای از داده‌ها پس از داده‌افزایی

۱-۴. پیاده سازی

در این پروژه، از دیتالودر با batch size 128 استفاده شده است. یکی از مهم‌ترین مزیت‌ها مدیریت بهینه حافظه است، زیرا داده‌ها به صورت batch بارگذاری می‌شوند و از بارگذاری کل مجموعه داده در حافظه جلوگیری می‌شود. همچنین، DataLoader با امکان استفاده از چندین worker به صورت موازی، سرعت بارگذاری داده‌ها را افزایش می‌دهد. قابلیت shuffle کردن داده‌ها نیز به بهبود تعمیم مدل و جلوگیری از overfitting کمک می‌کند. علاوه بر این، DataLoaderانعطاف‌پذیری بالایی دارد و می‌توان آن را برای داده‌های نامتعادل یا مجموعه داده‌های سنگین تنظیم کرد.

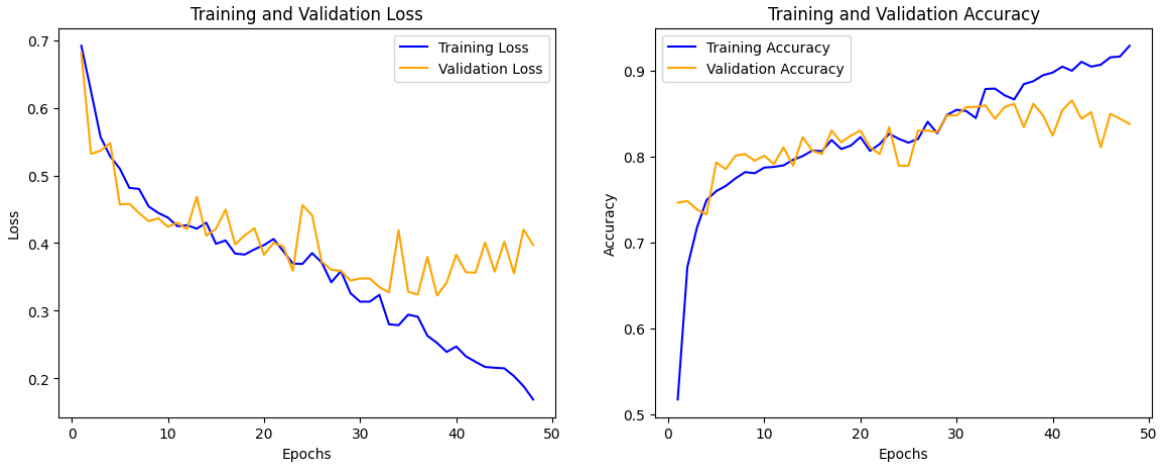
با این حال، این روش نقاط ضعفی نیز دارد. از جمله پیچیدگی بیشتر کد، که نیازمند آماده‌سازی خاص برای Dataset و DataLoader است. همچنین، استفاده از چندین worker ممکن است سربار پردازشی اضافی ایجاد کند. محدودیت در سفارشی‌سازی برای برخی کاربردهای خاص و نیاز به هماهنگی مناسب بین GPU و CPU از دیگر چالش‌های این روش هستند، زیرا عدم هماهنگی می‌تواند باعث ایجاد bottleneckدر عملکرد شود.

در این بخش از مدل و الگوریتم ارائه شده در مقاله استفاده شده با این تفاوت که در اینجا ما طبقه‌بندی باینری داریم پس تعداد نورون‌های لایه خروجی باید 2 باشد. داده‌ها نیز به صورتی به سه کلاس train و test و validation تقسیم شده‌اند که تعداد اعضای برابری از هر کلاس داشته باشند (شکل 1-5).

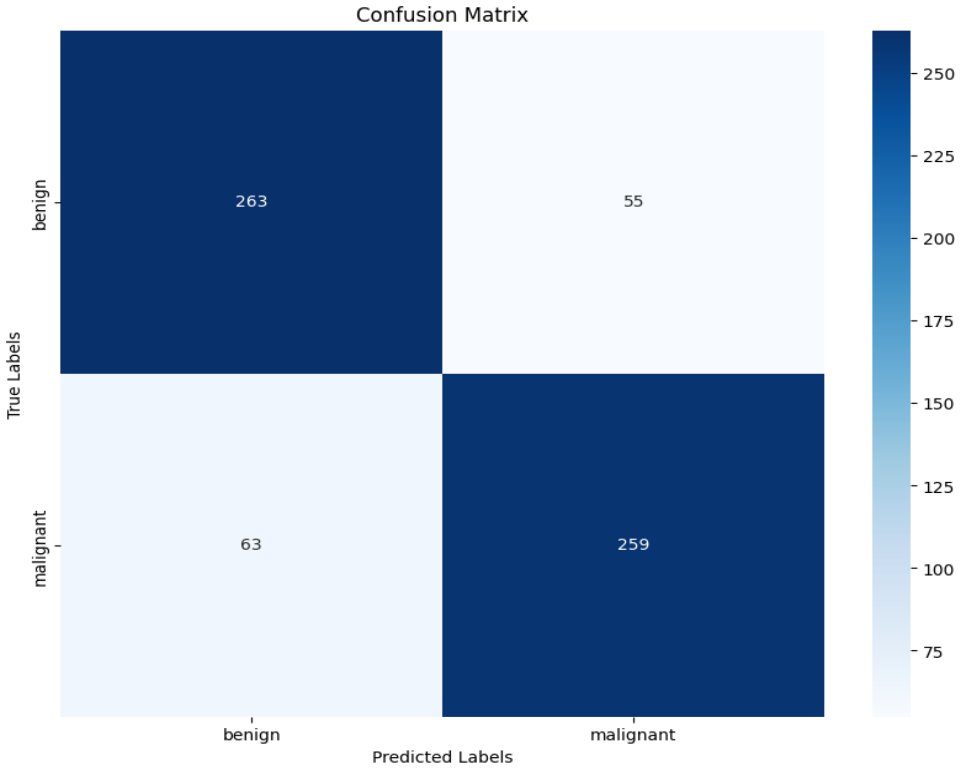
```
Training Set:
  benign: 1152
  malignant: 1152
Validation Set:
  benign: 288
  malignant: 288
Test Set:
  malignant: 360
  benign: 360
```

شکل 1-4. تعداد اعضای هر مجموعه داده

نتایج آموزش و تست مدل ابتدایی:



شکل 1-5. تغییرات دقت و هزینه داده‌های **train** و **validation** مدل ابتدایی



شکل 1-6. ماتریس آشفتگی داده تست مدل ابتدایی

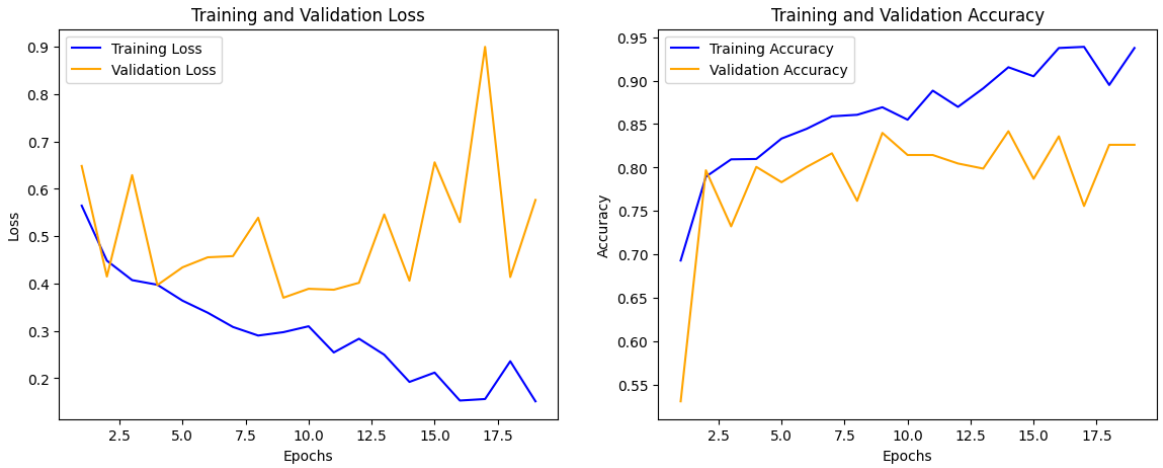
```
Evaluating on Test Set...
Test Loss: 0.5096, Test Accuracy: 0.8156
Precision: 0.8248, Recall: 0.8043, F1 Score: 0.8145
```

شکل 1-7. نتایج نهایی داده تست روی مدل ابتدایی

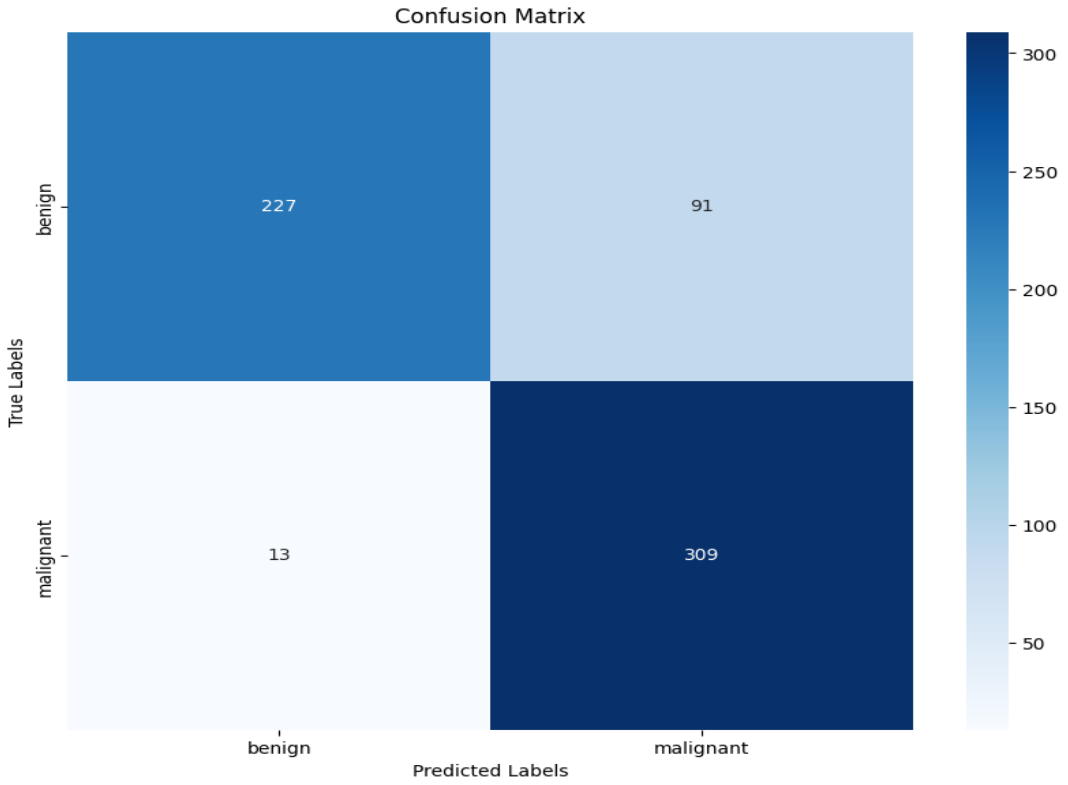
حال برای کاهش بیش‌پردازش و بهبود مدل، مدلی جدید ایجاد می‌کنیم با این تفاوت که پس از هر لایه convolutional، یک لایه bach normalization قرار می‌گیرد. همچنین پس از هر لایه کاملا متصل، یک لایه drop out قرار داده می‌شود.

- **Batch normalization:** با نرمال‌سازی داده‌های هر لایه در حین آموزش، مشکلاتی مانند تغییرات داخلی توزیع داده‌ها را کاهش می‌دهد. این کار باعث می‌شود فرآیند آموزش پایدارتر و سریع‌تر شود. همچنین، بچ نرمالیزیشن به مدل کمک می‌کند تا تعمیم‌پذیری بهتری بر روی داده‌های دیده‌نشده داشته باشد.
- **Drop Out:** به عنوان یک روش منظم‌سازی عمل می‌کند تا از بیش‌پردازش جلوگیری کند. این کار با غیرفعال کردن تصادفی تعدادی از نورون‌ها در هر مرحله از آموزش انجام می‌شود، که موجب می‌شود مدل نتواند بیش از حد وابسته به تعدادی از نورون‌های خاص شود و به این ترتیب ویژگی‌های کلی‌تری را یاد بگیرد.

نتایج آموزش و تست مدل بهبود یافته:



شکل 1-8. تغییرات دقت و هزینه داده‌های **train** و **validation** مدل بهبود یافته



شکل 1-9. ماتریس آشفتگی داده تست مدل بهبود یافته

```
Evaluating on Test Set...
Test Loss: 0.6087, Test Accuracy: 0.8375
Precision: 0.7725, Recall: 0.9596, F1 Score: 0.8560
```

شکل 1-10. نتایج نهایی داده تست روی مدل بهبود یافته

۱-۵. تحلیل نتایج

ماتریس آشفتگی برای هر دو مدل در بخش 1-5 و در شکل‌های 1-6 و 1-9 نمایش داده شده است. از روی این ماتریس می‌توان به معیارهای مهمی از جمله Accuracy, precision, recall, f1score دست یافت.

- Accuracy: نسبت تشخیص داده‌هایی که درست تشخیص داده شده اند به کل داده‌ها.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- Recall: نسبت تشخیص‌های مثبتی که واقعا مثبت هستند به آن‌هایی که درست تشخیص داده شده اند.

$$Recall = \frac{TP}{TP + FN}$$

- Precision: نسبت تشخیص‌های مثبتی که واقعا مثبت هستند به آن‌هایی که مثبت تشخیص داده شده اند.

$$Precision = \frac{TP}{TP + FP}$$

- F1 score: میانگین هارمونیک Precision و Recall.

$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

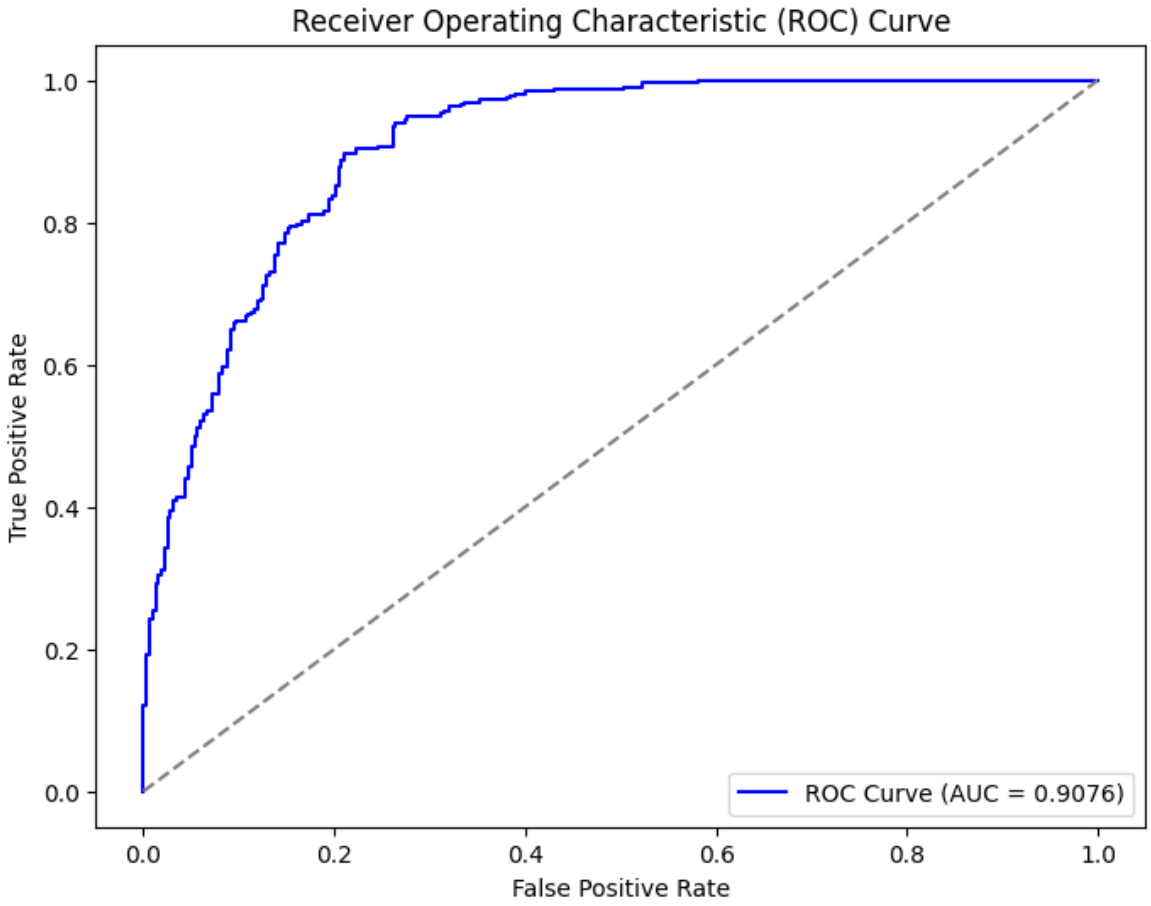
به طور کلی برای طبقه‌بندی، F1 score معیار خوبی برای تحلیل نتایج می‌باشد چرا که ترکیبی از سایر معیارها است و تحلیل ما به سمت خاصی جهتدار نمی‌شود. در بخش بعد مقایسه ای بین دو مدل انجام خواهیم داد.

نمودار ROC (Receiver Operating Characteristic) ابزاری قدرتمند برای ارزیابی عملکرد مدل‌های طبقه‌بندی است. این نمودار با نمایش رابطه بین نرخ مثبت‌های واقعی و نرخ مثبت‌های کاذب در آستانه‌های مختلف، به تحلیل کیفیت پیش‌بینی‌های مدل کمک می‌کند. ویژگی کلیدی این نمودار این است که به ما اجازه می‌دهد عملکرد مدل را در تمام سطوح آستانه بررسی کنیم و به وابستگی به یک آستانه خاص محدود نشویم. همچنین، ROC برای مقایسه مدل‌ها بسیار مفید است؛ مدلی که نمودارش به سمت بالا و چپ نمودار متمایل‌تر باشد، عملکرد بهتری دارد.

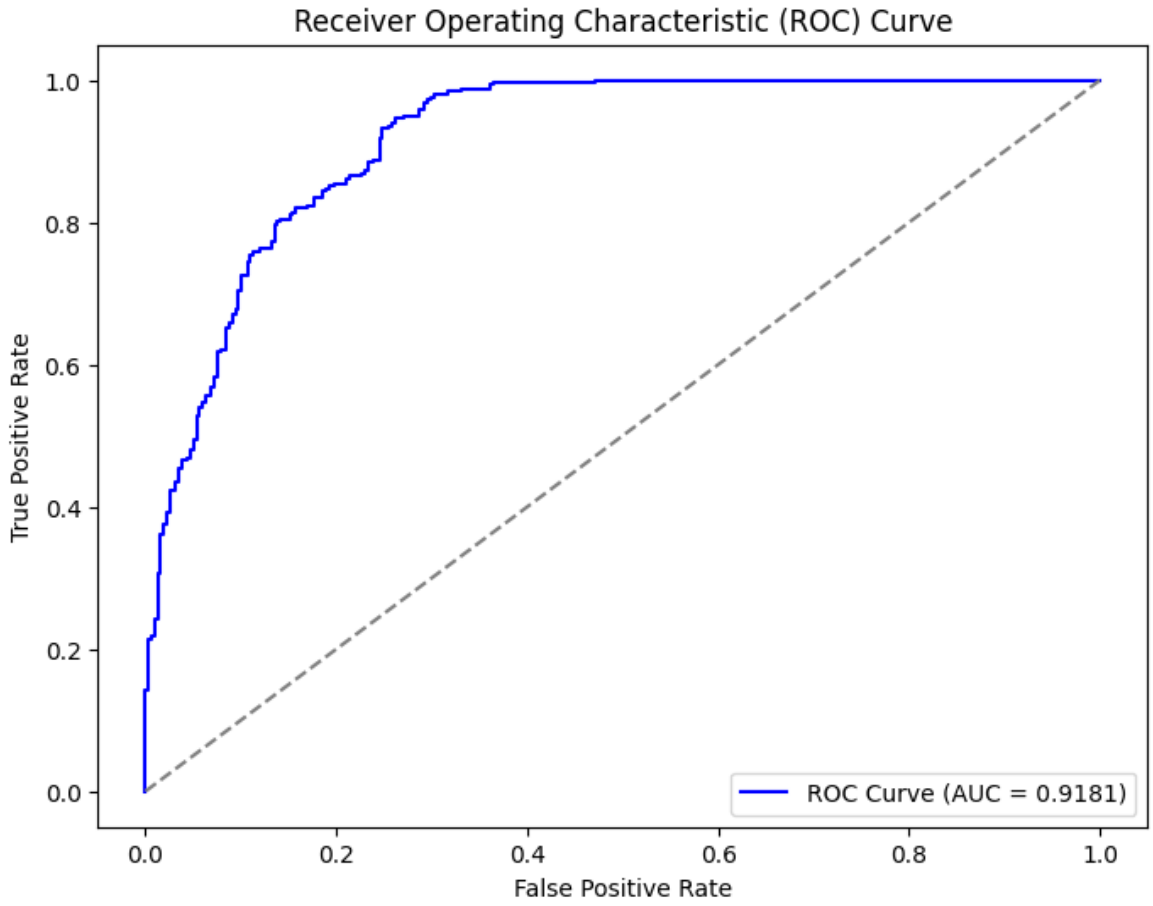
مساحت زیر نمودار ROC (AUC) معیاری عددی است که عملکرد کلی مدل را نشان می‌دهد. این مقدار بین ۰ تا ۱ است؛ هرچه AUC بیشتر باشد، مدل بهتر عمل کرده است. اگر نمودار ROC پایین‌تر از خط تصادفی باشد، به این معنی است که عملکرد مدل حتی از پیش‌بینی تصادفی نیز ضعیف‌تر است. این نمودار به‌ویژه در شرایطی که داده‌ها نامتوازن هستند مفید است، زیرا بدون تأثیرگذاری بر توزیع کلاس‌ها، تعامل بین دقت (Precision) و حساسیت (Recall) را بررسی می‌کند.

همانطور که در تصاویر 1-11 و 1-12 مشاهده می‌کنید، مساحت زیر نمودار برای هر دو مدل بیش از 0.9 است که این نشان‌دهنده این است که هر دو مدل یک طبقه بندی خوب انجام داده اند اما مساحت زیر

نمودار ROC در مدل دوم مقداری بیشتر از مدل اول است که این نشان‌دهنده بهبود عملکرد از جهت استفاده از Batch Normalization و Drop Out می‌باشد.



شکل 1-11. نمودار **ROC** مدل ابتدایی



شکل 1-12. نمودار **ROC** مدل بهبود یافته

۱-۶. مقایسه نتایج

در جدول 1-1 و 1-2 می‌توانید همه معیارهایی که در مورد آن‌ها در بخش قبل صحبت شد را مشاهده نمایید.

جدول 1-1. معیارهای آموزش دو مدل

Test Accuracy	Validation Accuracy	Train Accuracy	Model
0.8156	0.8379	0.9288	**Paper**
0.8375	0.8418	0.9154	**Improved**

جدول 1-2. معیارهای ارزیابی دو مدل

AUC	F1 Score	Recall	Precision	Accuracy	Model
0.9076	0.8145	0.8043	0.8248	0.8156	**Paper**
0.9181	0.8560	0.9596	0.7725	0.8375	**Improved**

همانطور که مشاهده می‌کنید، میزان بیش‌پردازش در مدل اول بیشتر از مدل دوم است. البته با افزایش میزان Drop Out این اختلاف بیشتر نیز می‌شد اما این کار باعث کاهش دقت نهایی می‌شود. مدل‌های زیادی با معماری‌های مختلف آزمایش شد و مدلی که نتایج آن گزارش شده، بهترین عملکرد را بین مدل‌های آزمایش شده دارا است. میزان Accuracy و F1 Score که معیار اصلی مقایسه ما است نیز در مدل دوم بهبود محسوسی داشته است. همچنین Early Stopping باعث شده تا مدل دوم در 17 مرحله آموزش ببیند در حالی که مدل اول 50 مرحله طول کشید که این نیز نشان دهنده درستی عملکرد لایه‌های Drop Out و Batch Normalization می‌باشد.

۱-۷. مدل عمیق‌تر

مدل ResNet (Residual Network) یک معماری عمیق در شبکه‌های عصبی است که برای حل مشکل "از بین رفتن گرادیان" در شبکه‌های بسیار عمیق طراحی شده است. ویژگی اصلی ResNet استفاده از Residual Connections است که به شبکه اجازه می‌دهد اطلاعات را از لایه‌های قبلی مستقیماً به لایه‌های بعدی منتقل کند. این ساختار باعث می‌شود آموزش مدل حتی با تعداد لایه‌های بسیار زیاد پایدار و مؤثر باشد .ResNet به دلیل عملکرد فوق‌العاده‌اش در تشخیص تصویر، طبقه‌بندی، و وظایف بینایی کامپیوتر شناخته شده است و نسخه‌هایی مانندResNet18 ، ResNet34و ResNet50 دارد که اعداد آن‌ها نشان‌دهنده تعداد لایه‌هاست.

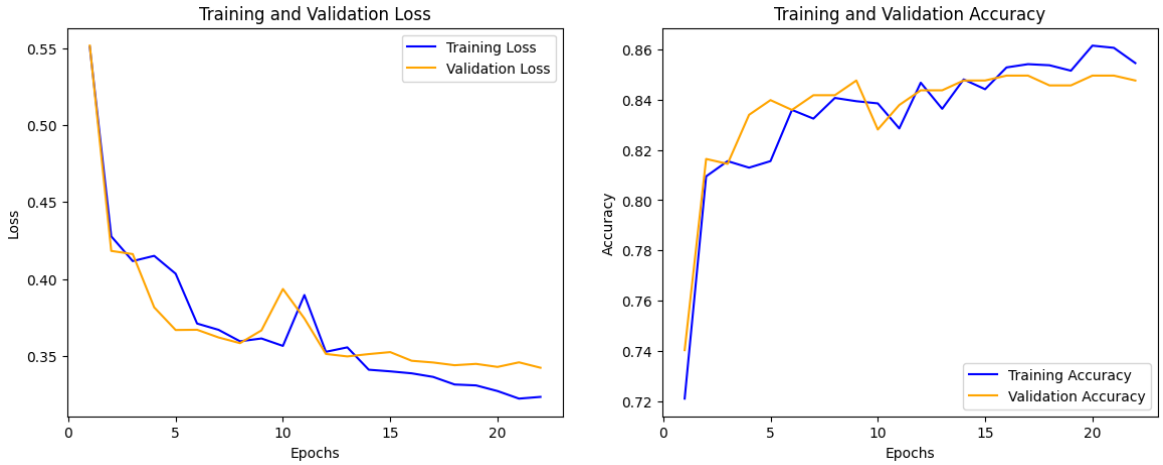
در این بخش ما از ResNet18 استفاده کردیم. معماری ResNet18 شامل 18 لایه آموزش‌پذیر است که از یک ترکیب متوالی لایه‌های کانولوشن، BatchNormalization، فعال‌سازی ReLU و Residual Connectionsتشکیل شده است.

ساختار اصلی آن شامل:

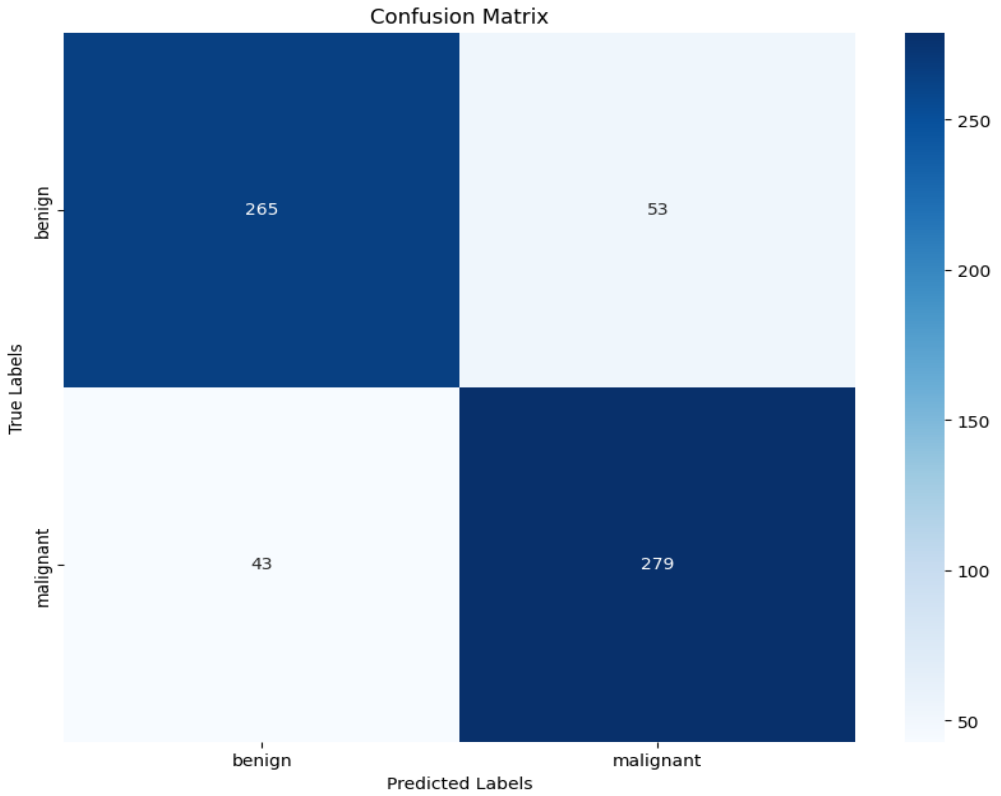
1. **لایه ورودی :**یک کانولوشن 7×7 با 64 فیلتر وstride=2 ، به‌همراه یک لایه MaxPooling با kernel 3×3 وstride=2.
2. **چهار بلوک باقی‌مانده :(Residual Blocks)**
   - بلوک اول: دو کانولوشن 3×3 با 64 فیلتر.
   - بلوک دوم: دو کانولوشن 3×3 با 128 فیلتر (stride=2 در اولین کانولوشن).
   - بلوک سوم: دو کانولوشن 3×3 با 256 فیلتر (stride=2 در اولین کانولوشن).
   - بلوک چهارم: دو کانولوشن 3×3 با 512 فیلتر (stride=2 در اولین کانولوشن).
3. **لایه پایانی :**یک Global Average Pooling و یک لایه Fully Connected با تعداد خروجی برابر با کلاس‌های موردنظر.

Residual Blocks در هر بلوک به‌طور مستقیم خروجی لایه قبلی را به لایه بعدی متصل می‌کنند، که موجب تسهیل جریان گرادیان و یادگیری عمیق‌تر مدل می‌شود.

ما این مدل را با فریز کردن لایه‌های ابتدایی و کاهش تعداد نورون‌های لایه آخر از 1000 به 2، fine tune کردیم تا لایه‌های convolutional دوباره آموزش نبینند.



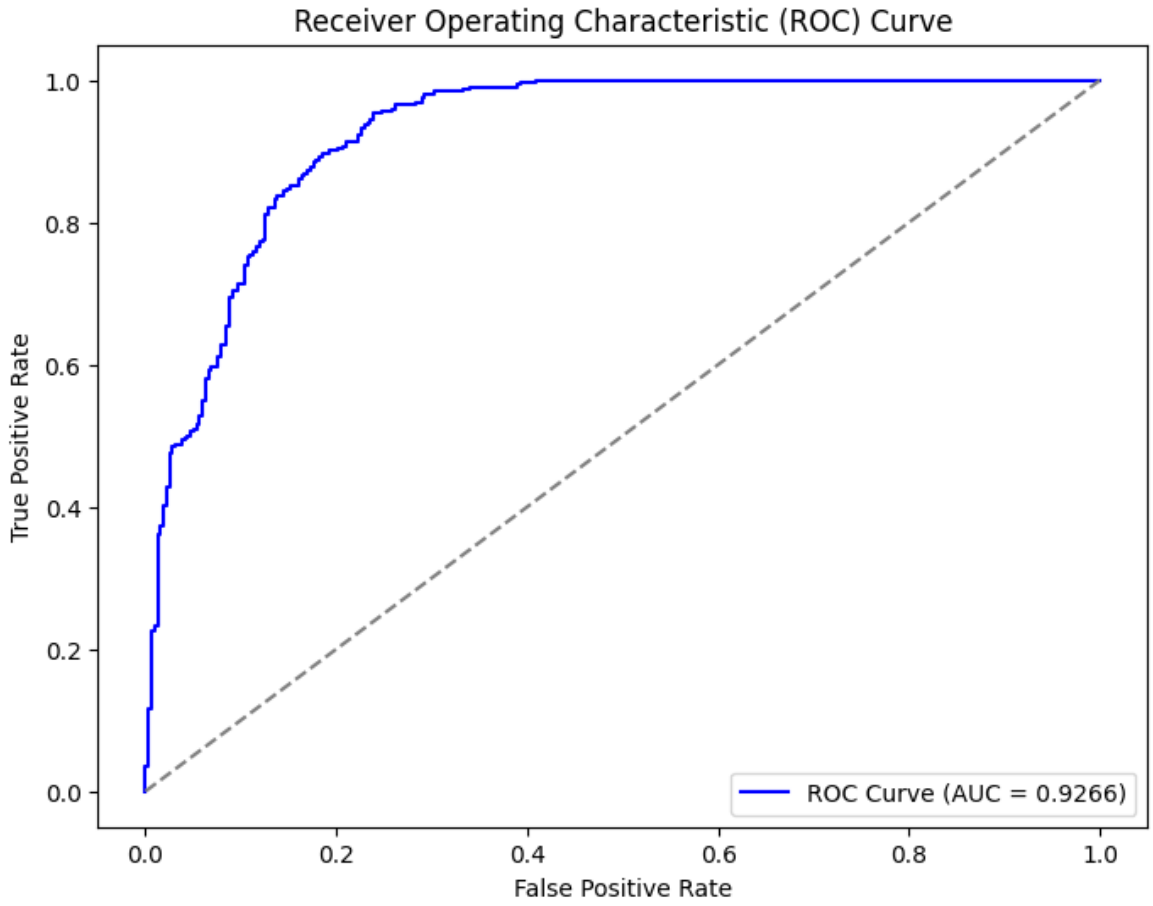
شکل 1-13. تغییرات دقت و هزینه داده‌های **train** و **validation** مدل **ResNet**



شکل 1-14. ماتریس آشفتگی داده تست مدل **ResNet**

```
Evaluating on Test Set...
Test Loss: 0.3362, Test Accuracy: 0.8500
Precision: 0.8404, Recall: 0.8665, F1 Score: 0.8532
```

شکل 1-15. نتایج نهایی داده تست روی مدل بهبود یافته

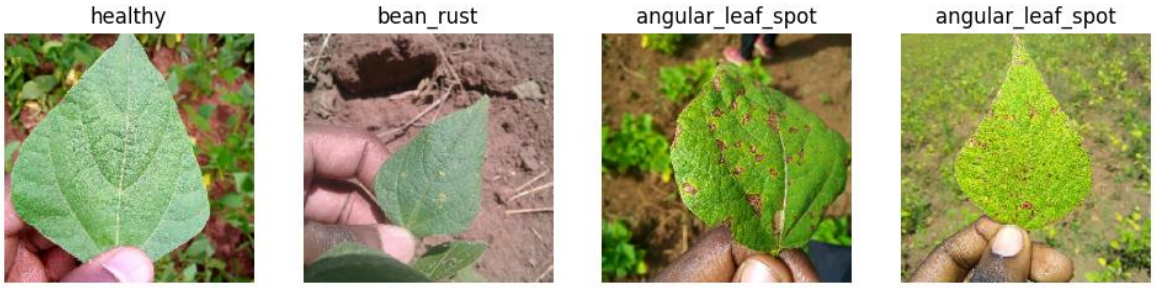


شکل 1-16. نمودار **ROC** مدل **ResNet**

همانطور که مشاهده می‌کنید، مدل ResNet از هر جهت عملکرد بهتری نسبت به دو مدل قبلی دارد که این به علت کشف ویژگی‌های بهتر توسط این مدل و به طور کلی، بهتر و پیچیده‌تر بودن این مدل است.

پرسش ۲ – تشخیص بیماری‌های برگ لوبیا با شبکه‌های عصبی

۲-۱. پیش‌پردازش تصاویر



شکل 2-1. نمونه‌هایی از دیتاست به همراه لیبل

تنها پیش‌پردازش انجام شده در مقاله، تغییر اندازه تصاویر به سایز 224 * 224 * 3 است. زیرا این اندازه با ساختار ورودی مدل‌های از پیش آموزش‌دیده مانند MobileNet ، EfficientNet و NasNet سازگار است. این کار باعث کاهش پیچیدگی محاسباتی، یکنواختی داده‌ها، و استفاده بهینه از ویژگی‌های استخراج‌شده توسط مدل شده و در نهایت عملکرد مدل را بهبود می‌بخشد. علاوه بر این، تصاویر برای مدل MobileNetV2 با وزن‌های ImageNet، نرمالایز کردن عکس‌ها ضروری است تا مقادیر پیکسل به بازه [0,1] برسند و سپس با میانگین [0.485,0.456,0.406] و انحراف معیار [0.229,0.224,0.225] نرمال شوند. این کار سازگاری داده‌های ورودی با تنظیمات از پیش آموزش‌دیده مدل را تضمین کرده و به بهبود دقت و پایداری در آموزش کمک می‌کند. همچنین برای مدل NasNet، که در این پیاده‌سازی از نسخه NasNetLarge استفاده شده است نیز از preprocess آماده در کتابخانه keras استفاده شده است تا تصاویر برای ورود به مدل آماده شوند.

۲-۲. پیاده‌سازی

۲-۲-۱. انتخاب مدل‌ها

- MobileNetV2

یک مدل شبکه عصبی سبک برای دستگاه‌های با منابع محدود است. این معماری بر پایه ایده Residual Connections طراحی شده است، اما تفاوت کلیدی آن در استفاده از Inverted Residuals است که در آن به‌جای کاهش ابعاد ویژگی در بلوک‌های Bottleneck، ابعاد افزایش می‌یابد. این روش باعث حفظ اطلاعات کلیدی در ویژگی‌ها و کاهش هزینه محاسباتی می‌شود.. این مدل با Depthwise Separable Convolution عملیات کانولوشن را بهینه می‌کند. ساختار

مدل شامل بلوک‌هایی است که ترکیبی از کانولوشن عادی، کانولوشن عمقی و تابع فعال‌سازی ReLU6 هستند. سایز ورودی پیش‌فرض 224 * 224 است و خروجی نهایی یک بردار پیش‌بینی دسته‌بندی می‌باشد که به اندازه تعداد کلاس‌ها است، که برای ImageNet این عدد برابر 1000 است.

- EfficientNetB6

یکی از مدل‌های سری EfficientNet است که با استفاده از تکنیک Neural Architecture Search (NAS) و روش Compound Scaling بهینه‌سازی شده است. این مدل با مقیاس‌دهی همزمان به عمق، عرض و رزولوشن شبکه، کارایی بالایی ارائه می‌دهد. EfficientNetB6 از بلوک‌های MBConv استفاده میکند که شامل کانولوشن‌های Depthwise و Pointwise همراه با SE Blocks (Squeeze-and-Excitation) است. این بلوک‌ها باعث افزایش توجه به ویژگی‌های مهم و کاهش نویز اطلاعاتی می‌شوند. سایز ورودی پیش‌فرض 528 * 528 است و خروجی نیز مشابه دیگر مدل‌های دسته‌بندی، به تعداد کلاس‌ها وابسته است.

- NasNet

توسط گوگل با استفاده از NAS توسعه یافته و بر اساس معماری‌های جستجو شده بهینه ساخته شده است. این مدل شامل دو نوع بلوک اصلی به نام Normal Cell و Reduction Cell است. Normal Cell به حفظ ابعاد ویژگی کمک می‌کند، درحالی‌که Reduction Cell ابعاد را کاهش می‌دهد تا پردازش در مراحل بعدی ساده‌تر شود. بلوک‌ها به صورت تکراری در کل شبکه استفاده می‌شوند و به طور خودکار برای بهینه‌سازی تعادل میان دقت و پیچیدگی محاسباتی طراحی شده‌اند. ویژگی قابل توجه NasNet، بهینه‌سازی از طریق جستجوی معماری است که تعادل بین دقت و پیچیدگی محاسباتی را فراهم می‌کند. سایز ورودی پیش‌فرض آن معمولاً 331 * 332 است و خروجی مانند مدل‌های دیگر، یک بردار ویژگی یا پیش‌بینی دسته‌بندی می‌باشد.

یادگیری تقویتی

برای پیاده‌سازی مدل با یادگیری انتقالی، نیاز است که head مدل pretrained را از آن حذف کنیم، زیرا این head برای classify کردن دیتاهایی است که شبکه قبلا با آن train شده است و نمی‌تواند به‌طور مستقیم برای دیتاست جدید استفاده شود. به همین دلیل، یک head جدید متناسب با دیتاست فعلی طراحی می‌کنیم. در این طراحی، ابتدا base model را freeze می‌کنیم، یعنی وزن‌های آن را ثابت نگه می‌داریم. این کار باعث می‌شود وزن‌های یادگرفته‌شده از داده‌های (در اینجا ImageNet) برای استخراج

ویژگی‌ها حفظ شوند و نیازی به تنظیم مجدد آن‌ها نباشد. این امر سرعت آموزش را افزایش داده و نیاز به داده‌های بیشتر را کاهش می‌دهد. پس از خروجی مدل پایه، از لایه GlobalAveragePooling2D استفاده می‌کنیم که وظیفه آن کاهش ابعاد ویژگی‌های خروجی مدل پایه است. این لایه، اطلاعات کلیدی را حفظ کرده و تعداد پارامترهای لایه‌های بعدی را کاهش می‌دهد، که به بهبود سرعت و کارایی مدل کمک می‌کند.

در مرحله بعد، یک لایه Dropout با مقدار 0.3 اضافه شده است. این لایه برای مقابله با overfitting اضافه شده است و با حذف تصادفی تعدادی از نورون‌ها در هر مرحله آموزش، از وابستگی بیش‌ازحد به برخی ویژگی‌ها جلوگیری می‌کند.

برای تقویت قدرت یادگیری، یک لایه Fully Connected با 64 نورون و تابع فعال‌سازی ReLU اضافه می‌شود. این لایه وظیفه ترکیب و پردازش ویژگی‌های استخراج‌شده از لایه پایه را دارد و با تابع فعال‌سازی ReLU، رفتار غیرخطی در مدل ایجاد می‌کند، که برای یادگیری ویژگی‌های پیچیده ضروری است.

در نهایت، یک لایه Dense دیگر به‌عنوان لایه خروجی اضافه شده است. این لایه شامل 3 نورون (برابر با تعداد کلاس‌های دیتاست) است و از تابع فعال‌سازی Softmax استفاده می‌کند که احتمال هر کلاس را محاسبه کرده و خروجی مدل را برای کلاس‌بندی نهایی فراهم می‌کند.

۲-۲-۲: تقویت داده

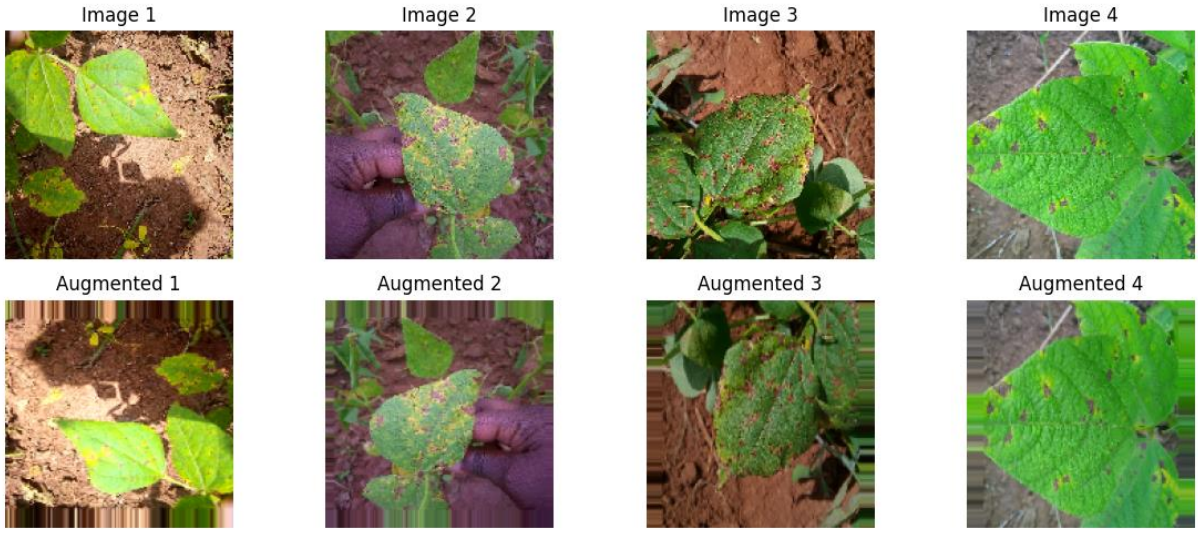
جدول 2-1. روش‌های انتخاب شده برای تقویت داده و توضیحات

Augmentation	Value	Description
zoom_range	0.2	این تکنیک بخش‌هایی از تصویر را بزرگ یا کوچک‌نمایی می‌کند که باعث می‌شود مدل بتواند ویژگی‌ها را در مقیاس‌های مختلف یاد بگیرد. این کار کمک می‌کند تا مدل در برابر تغییرات مقیاس مقاوم‌تر شود و ویژگی‌های مهم را در سطوح مختلف تصویر شناسایی کند.
horizontal_flip	True	این روش تصاویر را به صورت تصادفی افقی برمی‌گرداند. این کار برای داده‌هایی که ویژگی‌هایشان در دو طرف تصویر یکسان است (مانند برگ در این دیتاست) مفید است و باعث می‌شود مدل از یادگیری الگوهای جهت‌دار غیرعمومی جلوگیری کند.
vertical_flip	True	این روش تصاویر را به صورت تصادفی عمودی برمی‌گرداند. این تکنیک در مسائلی که جهت عمودی اهمیت زیادی ندارد

		(مانند برگ) مفید است. البته باید در استفاده از آن دقت کرد، زیرا برای داده‌هایی مانند چهره انسانی ممکن است غیرمنطقی باشد.
brightness_range	[0.8, 1.2]	تنظیم روشنایی تصاویر به صورت تصادفی کمک می‌کند تا مدل در شرایط نوری مختلف عملکرد خوبی داشته باشد. این تکنیک باعث افزایش مقاومت مدل در برابر تغییرات محیطی می‌شود، خصوصا همانطور که در نمونه‌ها دیده شد، تصاویر در نور های متفاوتی وجود دارند.

برای پیاده‌سازی این تقویت داده، از ImageDataGenerator در keras استفاده شد.

```
train_datagen = ImageDataGenerator(
    zoom_range=0.2
    horizontal_flip=True,
    vertical_flip=True,
    brightness_range=[0.8, 1.2]
)
```



شکل 2-2. نمونه‌هایی از تصاویر به همراه نسخه تقویت شده آنها

۲-۲-۳: اندازه‌های ورودی

همانطور که در بخش (2-2-1) گفته شد، هر کدام از مدل‌ها مقدار مشخص معمولی برای اندازه عکس‌های ورودی دارند. ولی خب این امکان نیز وجود دارد که هنگام لود کردن مدل از کتابخانه keras، ابعاد ورودی را مشخص کرد و این مورد آسیبی به مدل و آموزش و عملکرد آن نمیزند. در این تمرین چون

در ابتدا همه عکس‌ها به اندازه 224 * 224 تبدیل شدند، ابعاد ورودی همه مدل‌ها هم به این اندازه مقداردهی شد. همچنین عدد 3 هم به معنی وجود 3 کانال (channel) برای عکس‌ها است.

```
MobileNetV2(weights='imagenet', include_top=False,
input_shape=(224, 224, 3))

EfficientNetB6(weights='imagenet', include_top=False,
input_shape=(224, 224, 3))

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

تنظیم صحیح اندازه تصویر ورودی تأثیر قابل‌توجهی بر دقت و کارایی مدل‌های یادگیری عمیق دارد. اندازه تصویر باید با معماری مدل هماهنگ باشد. کوچک‌سازی تصاویر باعث کاهش پیچیدگی محاسباتی می‌شود؛ زیرا با کاهش تعداد پیکسل‌ها، پردازش سریع‌تر انجام می‌شود و مصرف منابع سخت‌افزاری بهینه می‌گردد. اما انتخاب اندازه نامناسب ممکن است جزئیات حیاتی را از بین ببرد، بنابراین اندازه‌ای باید انتخاب شود که اطلاعات کلیدی تصویر را حفظ کند. همچنین استفاده از تصاویر با اندازه یکسان یکنواختی داده‌ها را تضمین می‌کند، که این امر از بروز مشکلات ناشی از تفاوت اندازه‌ها در هنگام پردازش توسط مدل جلوگیری می‌کند. این عوامل در کنار هم، به بهبود عملکرد و دقت مدل کمک می‌کنند.

۲-۲-۴: بهینه‌سازها

- Adam

این بهینه‌ساز میانگین متحرک gradients (First Moment) و مربع gradients (Second Moment) را نگه می‌دارد و با استفاده از آن‌ها نرخ یادگیری را برای هر پارامتر به صورت تطبیقی تنظیم می‌کند. این روش ترکیبی از مزایای Momentum (برای سرعت‌بخشی به همگرایی) و RMSprop (برای نرخ یادگیری تطبیقی) است. Adam معمولا در مسائل مختلف به دلیل پایداری و سرعت همگرایی بالا عملکرد خوبی دارد و به دلیل تنظیم خودکار نرخ یادگیری، نیاز کمتری به تنظیم دستی هایپرامترها دارد.

- RMSprop

این بهینه‌ساز با تقسیم نرخ یادگیری به ریشه میانگین مربعات gradients، نرخ یادگیری را برای پارامترهایی که تغییرات زیادی دارند، کاهش می‌دهد. این ویژگی باعث می‌شود که مدل به سرعت در مسیر بهینه حرکت کند و در نواحی پرتلاطم پایدارتر باشد.RMSprop برای مسائل با داده‌های غیرایستا (مانند داده‌های سری زمانی) مناسب است. این روش می‌تواند در مسائل پیچیده‌ای که نیاز به تنظیم دقیق نرخ یادگیری دارند، عملکرد خوبی داشته باشد. با این حال، در برخی مسائل ممکن است نرخ یادگیری بیش از حد کاهش یابد و همگرایی کند شود.

- Nadam

Nadam نسخه بهبودیافته Adam است که از تکنیک Nesterov Momentum استفاده می‌کند. این تکنیک با اضافه کردن یک گام Lookahead به Adam، سرعت و دقت همگرایی را بهبود می‌بخشد. Nadam با ترکیب ویژگی‌های Adam و Nesterov، مسیر حرکت را در فضای پارامترها بهتر پیش‌بینی می‌کند و باعث کاهش نوسانات می‌شود و در مسائل حساس به نوسانات، عملکرد پایدارتری نسبت به Adam داشته باشد و به نقاط بهینه با تعمیم بهتر برسد.

چرا عملکرد متفاوت دارند؟

عملکرد متفاوت بهینه‌سازها به دلیل تفاوت در نحوه تنظیم نرخ یادگیری، حساسیت به هایپرامترها، و ساختار داده‌ها و مدل است. هر الگوریتم از روش‌های خاصی برای بهینه‌سازی استفاده می‌کند؛ برای مثال، Adam با استفاده از هر دو Moment، نرخ یادگیری را تطبیق می‌دهد، در حالی که RMSprop تنها بر پایه Second Moment عمل می‌کند، و این تفاوت باعث رفتارهای مختلف در همگرایی می‌شود. Nadam نیز با افزودن تکنیک Nesterov Momentum به Adam، مسیری نرم‌تر و دقیق‌تر در حرکت به سمت بهینه فراهم می‌کند. علاوه بر این، حساسیت الگوریتم‌ها به مقادیر اولیه هایپرامترها نقش مهمی دارد؛ Adam و Nadam معمولا کمتر تحت تأثیر مقادیر اولیه قرار می‌گیرند، در حالی که RMSprop نیاز به تنظیمات دقیق‌تری دارد. همچنین، ساختار داده و مدل مورد استفاده نیز بر عملکرد تاثیرگذار است. این عوامل باعث می‌شوند که انتخاب مناسب‌ترین الگوریتم برای هر مسئله نیازمند آزمایش و ارزیابی باشد.

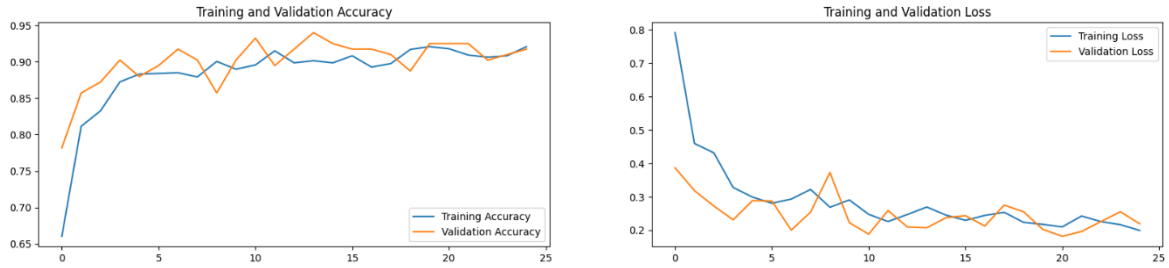
۲-۲-۵: آموزش مدل

حال مدل را با حالت‌های مختلف و گفته شده در مقاله آموزش می‌دهیم و نتایج را در تصاویر زیر گزارش می‌کنیم. همچنین early-stopping هم استفاده شده است و مشاهده می‌شود که برخی مدل‌ها زودتر از 25 epoch متوقف شده‌اند.

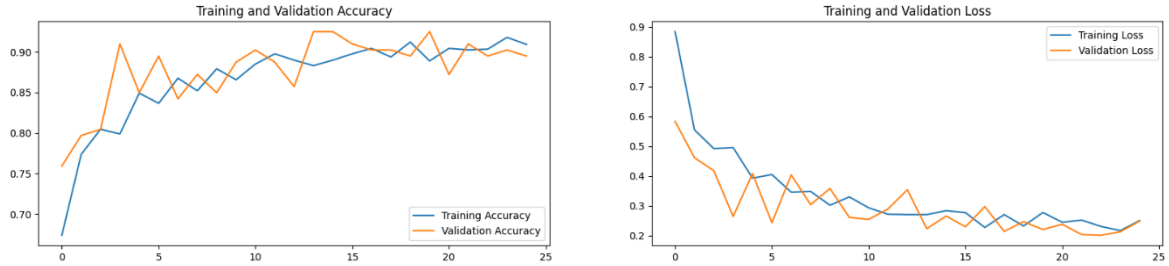
جدول **2-2**. مقادیر هایپرپارامترها بر اساس مقاله

Learning Rate	0.001
Epochs	25
Batch Size	32
Dropout	0.3
Optimizers	Adam, RMSprop, Nadam
Early Stopping Patience	10

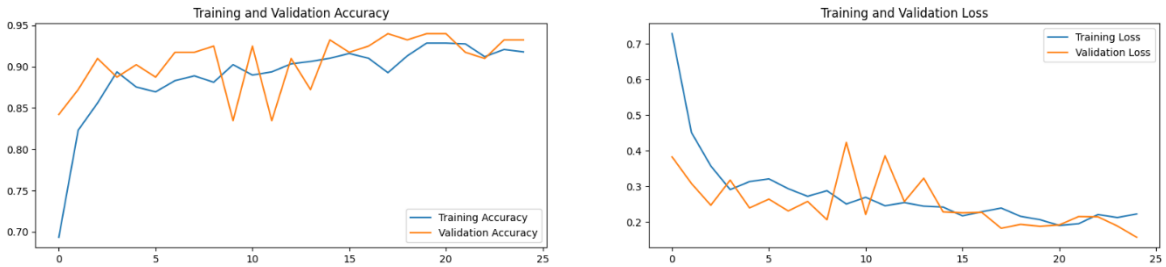
- MobileNetV2



شکل 2-3. عملکرد **MobileNetV2 - Adam**

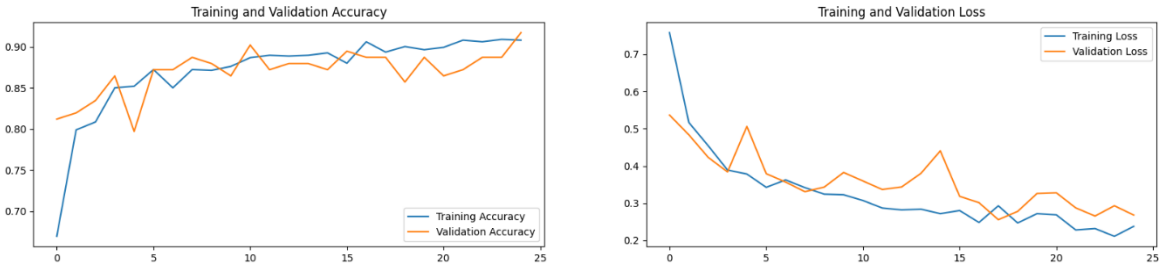


شکل 2-4. عملکرد **MobileNetV2 - RMSprop**

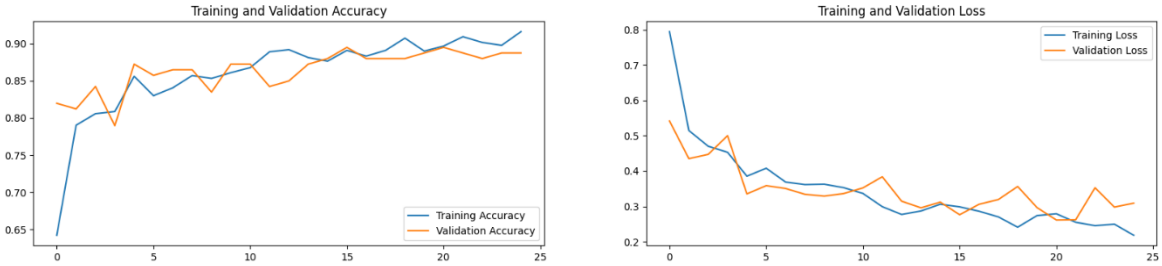


شکل 2-5. عملکرد **MobileNetV2 - Nadam**

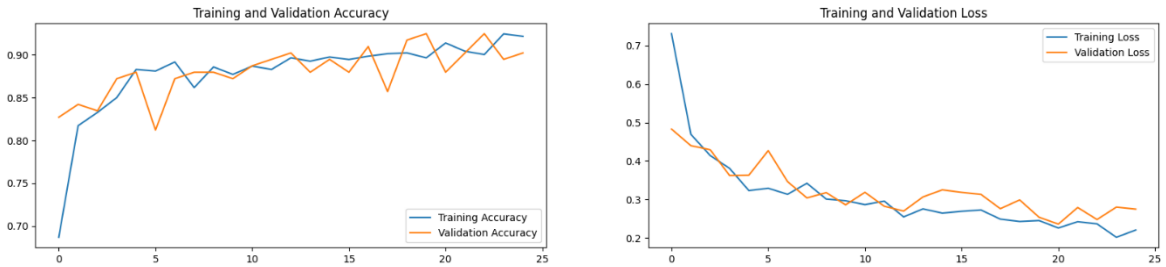
- EfficientNetB6



EfficientNetB6 - Adam عملکرد 2-**6**. شکل

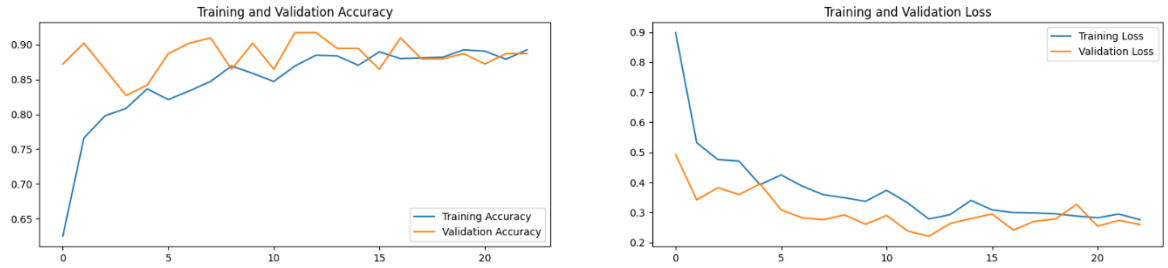


EfficientNetB6 - RMSprop عملکرد 2-**7**. شکل

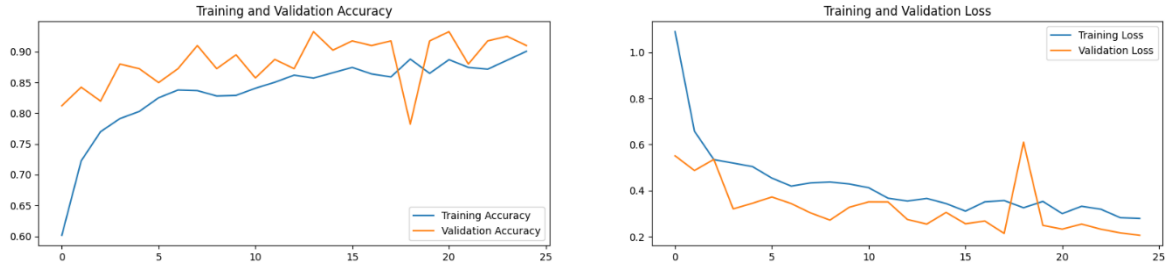


EfficientNetB6 - Nadam عملکرد 2-**8**. شکل

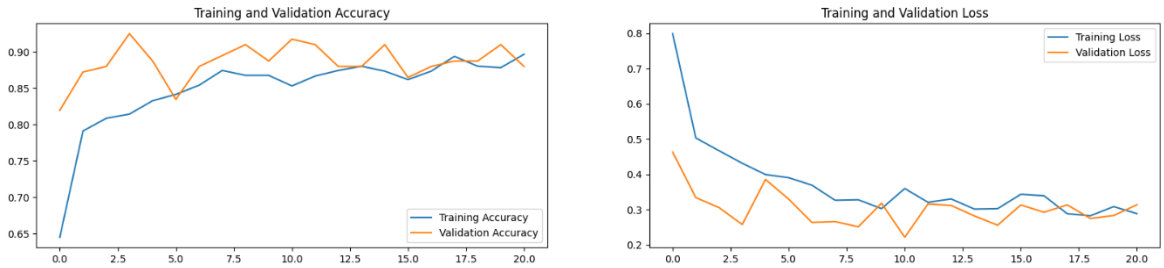
- NasNet (NasNetLarge)



شکل 2-**9**. عملکرد **NasNet - Adam**



شکل 2-**10**. عملکرد **NasNet - RMSprop**



شکل 2-**11**. عملکرد **NasNet - Nadam**

۲-۳: تحلیل نتایج

در ابتدا پس از آموزش مدل MobileNetV2 با بهینه‌ساز Nadam، تعدادی نمونه از دیتای ارزیابی را برای پیش‌بینی به آن می‌دهیم.

شکل 2-12. نتیجه پیش‌بینی مدل بر روی تعدادی نمونه ارزیابی

مشاهده می‌شود که یکی از نمونه‌ها به اشتباه پیش‌بینی شده است. البته این مورد که در 5 عکس، یک نمونه اشتباه شده است را نمیتوان تعمیم داد و با بررسی‌های بسته‌های دیگر از دیتا ارزیابی، تقریبا درصد زیادی به درستی پیش‌بینی شدند که در جدول (2-3) مشاهده می‌شوند.

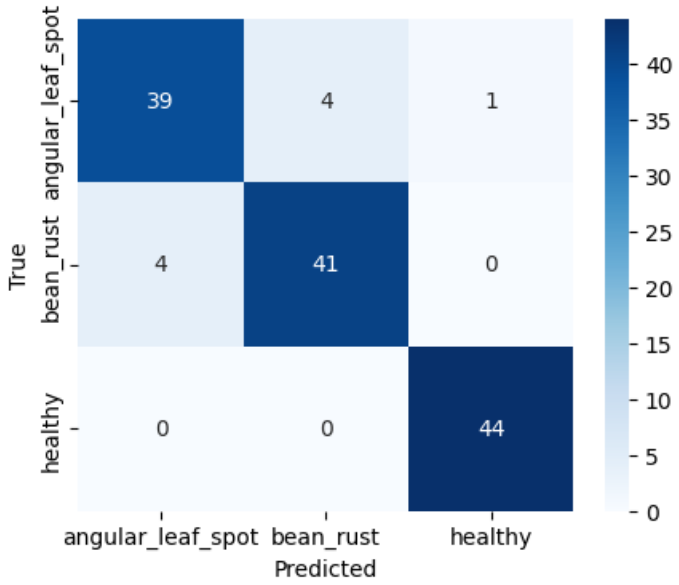
جدول 2-3. دقت و خطای مدل‌ها در داده آموزش و ارزیابی با بهینه‌سازهای مختلف

Optimizer	CNN Model	Tr-Acc(%)	Val-Acc(%)	Tr-Loss	Val-Loss
Adam	EfficientNetB6	90.14	91.73	0.2476	0.2680
	MobileNetV2	93.07	91.73	0.1761	0.2193
	NasNet	88.66	88.72	0.2886	0.2592
RMSprop	EfficientNetB6	91.81	88.72	0.2095	0.3092
	MobileNetV2	92.07	89.47	0.2185	0.2493
	NasNet	89.11	90.98	0.2728	0.2053
Nadam	EfficientNetB6	92.28	90.23	0.2209	0.2753
	MobileNetV2	92.26	93.23	0.2036	0.1570
	NasNet	90.71	87.97	0.2727	0.3141

همانطور که مشاهده می‌کنید، به جز مدل MobileNetV2 با بهینه‌ساز RMSprop، بقیه مدل‌ها Validation Accuray مانند مقاله یا بهتر از آن را بدست آورده اند؛ ولی تقریبا اکثر Train Accuracy های بدست آمده از مقاله کمتر است. که دلیل هر دو را می‌توان به Data Augmentation ارتباط داد که در مقاله از آن استفاده نشده بود، ولی در این پروژه استفاده شد که باعث می‌شود مقدار دقت در دیتای

آموزش کاهش یابد، اما از طرفی مدل generalizability بدست آورد و بتواند بر روی دیتای جدید و مشاهده نشده، عملکرد بهتری را نشان دهد.

مدل MobileNetV2 با بهینه‌ساز Nadam بیشترین دقت با 93.23 درصد و کمترین خطا با 0.1570 در داده ارزیابی داشته است. در شکل (2-13) می‌بینیم که برخی از نمونه‌های کلاس‌های bean_rust و angular_leaf_spot که کلاس‌های بیماری هستند با یکدیگر اشتباه پیش‌بینی شده‌اند. اما این را میتوان دید که کلاس healthy کاملا درست پیش‌بینی شده است و این نشان‌دهنده این است ک مدل به خوبی فرق بین برگ‌های سالم و غیرسالم را تشخیص داده و صرفا در انتخاب نوع بیماری، کمی دچار ضعف است.



شکل 2-13. ماتریس درهم‌ریختگی دیتای ارزیابی با مدل MobileNetV2 و Nadam

در بررسی نمودارهای دقت و خطا در بخش (2-2-5)، می‌بینیم که در اکثر موارد overfit رخ نداده و به جز مدل EfficientNetB6 با RMSprop و Nadam، دقت‌ها و خطاهای آموزش و ارزیابی بسیار نزدیک به همدیگر هستند. این overfit نشدن را همانطور که گفته شد، میتوان از Data Augmentation و لایه Dropout نشات گرفت. همچنین روند کلی همه نمودارها به سمت افزایش دقت و کاهش خطا است، با این نکته که مدل NasNet با Adam و Nadam باعث فعال شدن early-stopping شده است و بیشتر از 20 epoch پیش نرفته است. دلیل اینکه بقیه مدل‌ها 25 epoch را کامل کردند این است که با توجه به آزمایش‌های مختلف، patience در early-stopping مقدار 10 قرار داده شد تا مدل‌ها بیشتر آموزش ببینند و این باعث شد که اکثر مدل‌ها early-stopping را فعال نکنند.

نمودار تعدادی از مدل‌ها، مثل MobileNetV2 با Nadam یا EfficientNetB6، در حین آموزش نوسان‌های نسبتا زیادی دارند. با بررسی و آزمایش‌های متعدد، بنظر میرسد که این نوسانات ارتباط قوی‌ای

با لایه Fully Connected ای قبل از head قرار دادیم دارد و اگر تعداد لایه‌ها را بیشتر می‌کردیم یا همین یک لایه، نورون‌های بیشتری داشت، بشدت این نوسانات بیشتر میشد و در آخر هم به دقت خوبی نمی‌رسید. در حالت کنونی با یک لایه Fully Connected 64 نورونی، این نوسانات همچنان وجود دارند اما خیلی شدید نیستند و مشاهده هم شد که در آخر به دقت خوبی میتوانند برسند.

همچنین بعضی از مدل‌ها مانند NasNet با RMSprop، در یک epoch ناگهان یک peak خطای بالا می‌دهند که باعث افت شدید دقت در آن epoch می‌شود. دلیل واضحی برای این رفتار پیدا نشد و حتی با چندین بار آزمایش متعدد با پارامترهای مختلف، ممکن بود که باز این peak را داشته باشیم یا گاهی نداشته باشیم. در کل الگویی در رخ دادن این peak ها دیده نشد و باید بررسی‌های دقیق‌تر و حرفه‌ای‌تری بر روی ساختار شبکه و وزن‌ها انجام شود تا دلیل را متوجه شویم.