به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

شبکه های عصبی و یادگیری عمیق

مینی پروژه یک

نام و نام خانوادگی	علی عدالت
شماره دانشجویی	۸۱۰۱۹۹۳۴۸
تاریخ ارسال گزارش	

فهرست گزارش سوالات (لطفاً پس از تکمیل گزارش، این فهرست را به‌روز کنید.)

سوال 1 – –مفاهیم تئوری

۱

الف

روش گرادیان کاهشی به صورت mini-batch نمی‌تواند همگرایی خوبی را تضمین کند. مشکلات این روش در زیر آمده است.

تعیین نرخ یادگیری دشوار است. اگر این نرخ بسیار کم باشد، همگرایی کند و دردناکی خواهیم داشت. اگر این نرخ خیلی زیاد باشد، باعث می شود اطراف مینیمم محلی نوسان کنیم و حتی می‌تواند باعث واگرایی شود.

ثابت بودن نرخ یادگیری روش گرادیان کاهشی مناسب نیست. در طول یادگیری نیاز است که نرخ به روز شود. در ابتدا ما برای نزدیک شدن به مینیم به گام‌های بلندی نیاز داریم تا به سرعت بتوانیم به اطراف مینیمم برسیم. در حالی که در نزدیکی مینیمم ما به گام‌های کوتاهی نیاز داریم تا همگرا شویم. اگر در اطراف مینیم گام بزرگی داشته باشیم، نوسان و حتی واگرایی ممکن است که رخ دهد. این تغییر اندازه نرخ یادگیری باید با توجه به روند یادگیری و حرکت به سوی هدف باشد. برای کاهش نرخ می‌توان از زمانبندی از پیش تعریف شده استفاده کرد. بر اساس این که تغییر مفدار تابع هدف میان epoch ها از یک استانه کمتر شود نیز می توان نرخ را کاهش داد. باز در اینجا نیز باید آستانه از پیش تعریف شده باشد. به همین دلیل از قبل تعیین شدن این موارد، این روش‌ها نمی توانند خود را با ویژگی‌های مجموعه داده مورد بررسی تطبیق دهند. به همین دلیل استفاده از این روش‌ها سخت و در بیشتر مواقع ناکارآمد خواهد بود. دلیل آن عدم توجه به ویژگی‌های مجموعه داده است.

نرخ یادگیری یکسانی برای همه به روزرسانی های پارامتر اعمال می شود. اگر داده های ما پراکنده باشد و ویژگی های ما فرکانس های بسیار متفاوتی داشته باشد ، ممکن است ما نخواهیم همه آنها را به همان اندازه به روز کنیم ، اما برای ویژگی هایی که به ندرت اتفاق می افتد یک به روزرسانی بزرگتر انجام دهیم.

گرادیان کاهشی در پیمایش دره ها مشکل دارد، یعنی مناطقی که سطح آنها در یک بعد بسیار شیب دارتر از ابعاد دیگر است. در اطراف اپتیمم های محلی این موضوع معمول است. در این سناریوها، گرادیان کاهشی در دامنه های دره در حال نوسان است در حالی که فقط با تردید در امتداد پایین و به سمت مطلوب محلی پیشرفت می کند.

چالش اصلی دیگر کمینه کردن توابع خطای بسیار غیر محدب رایج برای شبکه های عصبی و جلوگیری از گرفتار شدن در حداقل های محلی زیر بهینه بیشمار آنها است. دشواری در واقع از حداقل محلی نیست بلکه از نقاط زین ناشی می شود. یعنی نقاطی که در یک بعد با شیب سمت بالا و در بعد دیگری با شیب پایین روبرو می شود. این نقاط زین معمولاً با فلات خطای یکسان احاطه شده اند که فرار گرادیان کاهشی را دشوار می کند. زیرا شیب در تمام ابعاد نزدیک به صفر است. این باعث می‌شود در این نقاط گرادیان کاهشی گیر کند و به نقطه کمینه بهینه نرسد.

ب

Momentum روشی است که به تسریع گرادیان کاهشی در جهت مینیم محلی کمک می کند و نوسانات را کاهش می دهد. در این روش از میزان تغییرات گذشته در بروز رسانی پارامتر برای تعیین میزان تغییر کنونی پارامتر استفاده می‌کنیم. در Figure 1 نحوه بروزرسانی در این روش آمده است.

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta)$$
$$\theta = \theta - v_t$$

Figure 1 نحوه بروزرسانی پارامتر در **momentum**

عبارت Momentum برای ابعادی که گرادیان هم جهت است و تغییر جهت نداریم، افزایش می‌یابد. در ابعادی که تغییر جهت گرادیان داریم، مقدار عبارت کم می‌شود. برای تعیین میزان تغییر هر پارامتر از جمع ضریب دار جابه جایی های گذشته پارامتر هم استفاده می‌کنیم. اگر تاکنون گرادیان در جهت پایین و کمینه محلی بوده است، اگر الان گرادیان در جهت بالا چپ برای مثال باشد. تا حد ممکن به کمک سابقه قبلی حرکت به سمت بالا چپ را تضعیف می‌کنیم. این کار باعث می‌شود میزان تغییرات کنونی ما کمتر شود که حاصل کم شدن عبارت Momentum است. اگر گرادیان کنونی در جهت قبلی ها باشد، جمع سابقه با گرادیان کنونی حرکت در جهت گرادیان کنونی را تقویت می‌کند. این باعث می‌شود در جهت گرادیان بیشتر جابه جا شویم. باعث می‌شود سریعتر به سمت کمینه محلی حرکت کنیم. در جایی که نوسان داریم، قبلا پایین رفتیم حالا بالا می‌ریم یا قبلا راست رفتیم و حالا چپ، به کمک جمع سابقه با گرادیان کنونی جلوی نوسان تا حد ممکن گرفته می‌شود. جمع آثار در این شرایط به شکلی است که حرکت تقریبا نزدیک به جهتی است که ما را به کمینه می‌رساند. حرکت در جهت کمینه قاطعیت دارد و

در جهت های دیگر در حد تردید ناشی از گرادیان کنونی است. این موارد باعث می‌شود مشکل گرادیان کاهشی در پیمایش دره ها برطرف شود.

Adagrad الگوریتمی برای بهینه سازی مبتنی بر شیب است که دقیقاً این کار را انجام می دهد: با انجام به روزرسانی های کوچکتر، میزان یادگیری را با پارامترها تطبیق می دهد. از میزان یادگیری پایین (نرخ یادگیری پایین) برای پارامترهای مرتبط با ویژگی های مکرر و به روزرسانی های بزرگتر (نرخ یادگیری بالا) برای پارامترهای مرتبط با ویژگی های نادر استفاده می‌کند. به همین دلیل، برای پرداختن به داده های پراکنده بسیار مناسب است. در گرادیان کاهشی ما برای تمام پارامترها و در تمام زمان‌ها از یک نرخ یادگیری استفاده می‌کردیم. در Adagrad برای هر پارامتر ما از نرخ یادگیری متفاوتی استفاده می‌کنیم. همچنین نرخ یادگیری برای هر پارامتر در طی زمان تغییر می‌کند. روش محاسبه نرخ یادگیری در Figure 2 آمده است.

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i}.$$

Figure 2 نحوه بروزرسانی پارامتر در **adagrad**

در Figure 2 $g_{t,i}$ میزان گرادیان کنونی تابع هزینه و ماتریس G یک ماتریس قطری است. هر المان بر روی قطر اصلی آن مانند i,i برابر مجموع مربعات گرادیان پارامتر θ_i تا زمان t است. با این روش نرخ یادگیری با گذشت زمان و افزایش اندازه مجموع مربعات گرادیان یک متغییر برای آن متغییر، کاهش می‌یابد. برای پارامترهای مربوط به ویژگی‌های مکرر ما همیشه گرادیان غیر صفر داریم و اندازه مجموع مربعات گرادیان از یک پارامتر مربوط به ویژگی‌های نادر بیشتر است. این باعث می‌شود نرخ یادگیری برای این پارامترها کمتر باشد. این موضوع مشکل تعیین نرخ یادگیری و ثابت بودن نرخ یادگیری را برطرف می‌کند. همچنین مشکل نرخ یادگیری یکسان برای تمام پارامترها هم با این روش حل می‌شود.

ضعف اصلی Adagrad جمع شدن مربع شیب‌ها در مخرج است: از آنجا که هر مقدار اضافه شده مثبت است، مقدار جمع شده در طول آموزش رشد می کند. این به نوبه خود باعث می شود که میزان یادگیری کوچک شود و در نهایت بینهایت کوچک شود. در آن زمان الگوریتم دیگر قادر به کسب دانش اضافی نیست. یک روش برای حل این مشکل AdaDelta است که توسعه یافته Adagrad است. در این روش تمام خاصیت های Adagrad را داریم و قدرت حل مشکل ما مانند Adagrad است. در AdaDelta توانایی بیشتری داریم.

این روش به دنبال کاهش میزان تهاجم و یکنواختی در کاهش نرخ یادگیری است. در این روش به جای جمع تمام مربع شیب‌های گذشته از پنجره محدودی برای جمع شیب‌های گذشته استفاده می‌کند. برای این کار از decaying average مربع تمام شیب‌های گذشته استفاده می‌کنیم. نحوه محاسبه متوسط مربع شیب‌ها در زیر Figure 3 آمده است. در این روش شیبهای نزدیک ضریب بیشتری دارند و اهمیت بیشتری می‌گیرند. شیب‌های بسیار دور ضریب‌های بسیار کمی دارند که تاثیر آنها را بسیار ناچیز می‌کند.

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2$$

Figure 3 نحوه محاسبه **decaying average** مربع تمام شیب‌های گذشته

به این روش مشکل کاهش نمایی نرخ یادگیری و کاهش ثابت آن در روش Adagrad حل می‌شود. از متوسط در Figure 3 به جای ماتریس G در محرج برای تعیین نرخ یادگیری استفاده می‌شود.

روش دیگر Adam یا Adaptive Moment Estimation است. این روش از ترکیب روش AdaDelta و Momentum استفاده می‌کند. از decaying average مربع تمام شیب‌های گذشته مانند AdaDelta به عنوان v_t برای تعیین نرخ یادگیری استفاده می‌کند. از decaying average تمام شیب‌های گذشته مانند Momentum به عنوان m_t برای گرادیان کنونی استفاده می‌کند. عبارت به روز رسانی این میانگین‌ها در Figure 4 آمده است.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

Figure 4 به روز رسانی متوسط مربع شیب‌ها و خود شیب‌های گذشته

نحوه به روز رسانی پارامتر در این روش در Figure 5 آمده است.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t.$$

Figure 5 نحوه به روز رسانی پارامتر در روش **Adam**

همانطور که دیده می‌شود این روش ترکیب روش‌های روش AdaDelta و Momentum است و خاصیت‌های هر دو روش را دارا است. این باعث می‌شود تمام مشکل‌های قابل حل در این دو روش برای

گرادیان کاهشی، در روش Adam نیز قابل حل باشد. . این روش مشکل تعیین نرخ یادگیری و ثابت بودن نرخ یادگیری را برطرف می‌کند. همچنین مشکل نرخ یادگیری یکسان برای تمام پارامترها هم با این روش حل می‌شود. این روش مشکل گرادیان کاهشی در پیمایش دره ها را برطرف می‌کند. در عمل نشان داده شده است که روش adam به خوبی عمل می‌کند و تفاوت عملکرد آن با روش دیگر قابل مقایسه است.

۲

Overfit زمانی رخ می‌دهد که عملکرد مدل بر روی داده دیده نشده اختلاف زیادی با عملکرد بر روی داده آموزش داشته باشد. این موضوع نشان می‌دهد که قدرت تعمیم مدل پایین است و جنرالیزیشن خوبی نداریم. در این حالت عملا مدل داده‌های دیده شده در آموزش را به خاطر سپرده است و به همین دلیل برای داده جدید عملکرد خوبی ندارد.

برای حل این مشکل از سه روش dropout ، penalty norm و early stopping استفاده میشود. در روش penalty norm اندازه وزن‌های پارامترها را محدود می‌کنیم. مدلی با وزن های بزرگ می تواند نشانه یک مدل ناپایدار باشد که در آن تغییرات کوچک ورودی می تواند منجر به تغییرات زیادی در خروجی شود. این می تواند نشانه این باشد که مدل بر مجموعه داده های آموزشی منطبق است و هنگام پیش بینی داده های جدید عملکرد ضعیفی خواهد داشت. یک راه حل برای این مشکل به روزرسانی الگوریتم یادگیری است تا مدل را به کوچک نگه داشتن وزن تشویق کند. این تنظیم وزن نامیده می شود و می تواند به عنوان یک تکنیک کلی برای کاهش overfit بر مجموعه داده های آموزش و بهبود تعمیم مدل مورد استفاده قرار گیرد. از روش‌های مانند L1 norm و L2 norm برای این کار استفاده می‌شود. در اولی مجموع اندازه وزن‌ها و در دومی مجموع مربعات وزن‌ها را کمینه می‌کنیم. برای این کار این مجموعها به تابع هزینه اضافه می‌شود. این کمینه کردن باعث می‌شود که وزنها کوچک باقی بمانند.

چالش اساسی در آموزش شبکه های عصبی ، مدت زمان آموزش آنها است. آموزش زیاد به معنای این است که مدل بر مجموعه داده های آموزشی منطبق است و عملکرد ضعیفی در مجموعه آزمون دارد. یعنی overfit رخ داده است. روش حل این است که روی مجموعه داده های آموزشی، آموزش دهید اما در مرحله‌ای که عملکرد در یک مجموعه داده اعتبار سنجی شروع به کاهش می کند، آموزش را متوقف کنید. این رویکرد ساده، موثر و گسترده مورد استفاده برای آموزش شبکه های عصبی، توقف زودرس یا early stoppng نامیده می شود.

شبکه های عصبی یادگیری عمیق به احتمال زیاد به سرعت با یک مجموعه داده آموزشی با تعداد مثال کم دچار overfit می‌شود. ensembles مجموعه‌ای از شبکه‌های عصبی با پیکرهای مختلف باعث کاهش overfit می‌شود. اما برای این کار محاسبات بالا و بیشتری بر روی داده آموزش نیاز است. یک راه برای استفاده از این روش با محاسبات پایین استفاده از dropout است. می توان از یک مدل واحد برای شبیه سازی داشتن تعداد زیادی از معماری های مختلف شبکه با dropout تصادفی گره‌ها در حین آموزش استفاده کرد. این یک روش regularization بسیار ارزان و کاملاً محاسباتی و موثر برای کاهش اتصالات اضافی و بهبود خطای تعمیم در شبکه های عصبی عمیق از هر نوع ارائه می دهد. در طول آموزش ، تعدادی از خروجی های لایه به طور تصادفی نادیده گرفته می شوند. این باعث می‌شود تعدادی گره در هر لایه به طور تصادفی حذف شود. از شبکه حاصل که تعداد نورون کمتری دارد برای یادگیری کامل بر روی داده‌های آموزش استفاده می‌شود. عملا یک شبکه با یک معماری خاص بر روی داده train آموزش داده ایم. هر بار ساختار شبکه به طور تصادفی تغییر می‌کند و شبکه‌های با ساختار مختلف ایجاد می‌کنیم که همگی به طور موازی روی train آموزش داده شده اند. Ensemble این مدل‌ها مدل نهایی را می‌سازد که باعث می‌شود قدرت تعمیم شبکه افزایش یابد و overfit کم شود.

۳

گاهی اوقات ما نیاز داریم که از داده‌های خام ویژگی استخراج کنیم و بر اساس آنها به طبقه بندی یا رگرسیون بپردازیم. در این موارد ما یک شبکه طراحی می‌کنیم که داده خام را به آن اعمال می‌کنیم. این شبکه به صورت خودکار ویژگی‌های مهم در داده‌ها را یاد می‌گیرد و می‌تواند از داده‌های خام آنها را استخراج کند. همچنین بر اساس ویژگی‌های یادگرفته به یادگیری و تخمین تابع می‌پردازد. برای یادگیری استخراج ویژگی از داده‌های خام ما نیاز داریم تا تعدادی لایه در ابتدا داشته باشیم. بعد از این لایه‌ها ما تعدادی لایه برای تخمین تابع خواهیم داشت. در این شرایط ما از بیش از دو لایه مخفی استفاده می‌کنیم. گاهی اوقات ما ورودی‌های با تعداد بالای ویژگی و ابعاد داریم، در این موارد نیز از تعداد لایه مخفی بیشتر استفاده می‌کنیم. در این موارد ما با این لایه‌های بیشتر ویژگی‌های مهم رو از ویژگی‌های ورودی برای حل مسئله استخراج می‌کنیم.

۴

دلیل آن جلوگیری از زیاد شدن پارامتر است. اگر تعداد پارامترها زیاد شود و over parameterization رخ دهد، برای یک مجموعه داده ثابت با تعداد نمونه محدود دچار overfit می‌شویم. تعداد پارامترها آنقدر زیاد شده که کاملا داده آموزش و نویزهای موجود در آن را را به خاطر می‌سپاریم. تعداد پارامتر زیاد قدرت مدل سازی بسیار بالایی را به ما می‌دهد که این قدرت را به ما می‌دهد که داده‌ها را به خاطر بسپاریم. این موضوع قدرت تعمیم را پایین می‌آورد و یادگیری خوبی نخواهیم داشت.

۵

تمام نورون ها در صورتی که وزن آنها یکسان باشد در نهایت یک چیز را می آموزند. حال بگذارید ببینیم چرا این اتفاق می افتد. برای یک لایه معین L بگویید ما M نورون داریم و لایه قبلی آن L-1 دارای N نورون است. بیایید بگوییم وزنهای مرتبط با دو نورون لایه L برابر W [1] و W [2] هستند. حال هر نورون این لایه مجموع حاصل از خروجی های فعال سازی a [l-1] از لایه قبلی و وزن های مرتبط با آن نورون را به عنوان ورودی می گیرد. اگر وزن W [1] و W [2] یکسان باشد ، در طی تکثیر رو به جلو ، دو نورون ورودی یکسانی را در یافت می‌کنند. ضرب وزن و خروجی لایه قبل برای هر دو نورون یکسان خواهد بود و خروجی فعال سازی یکسانی را تولید می کنند. به طور مشابه در هنگام backpropagation، وزن های یکسان نیز به روشی مشابه تغییر می کنند و در نهایت دوباره یکسان می شوند. بنابراین در نهایت دو نورون دقیقاً به یک شکل رفتار می کنند و یاد می گیرند فقط یک ویژگی را شناسایی کنند. حال اگر تمام وزن های یک شبکه عصبی نیز یکسان باشد، تمام نورون ها فقط یک ویژگی خاص را یاد می گیرند. دلیل اینکه ما از چند سلول عصبی در یک شبکه عصبی استفاده می کنیم این است که می خواهیم نورون های مختلف ویژگی های مختلف داده های ورودی را بیاموزند. اگر همه نورون ها فقط یک چیز را یاد بگیرند پس هیچ دلیلی برای استفاده از شبکه عصبی با چندین نورون وجود ندارد. برای دستیابی به نتایج خوب می توان از روشهای مختلف مقداردهی اولیه مانند Xavier Initialization و غیره استفاده کرد.

۶

گرادیان خطا جهت و بزرگی محاسبه شده در حین آموزش یک شبکه عصبی است که برای به روزرسانی وزن های شبکه در جهت درست و با مقدار مناسب استفاده می شود. در شبکه های عمیق یا شبکه های عصبی مکرر ، شیب های خطا می توانند در هنگام بروزرسانی ترکیب شوند و منجر به شیب های بسیار

بزرگ شوند. اینها به نوبه خود منجر به بروزرسانی گسترده در وزن شبکه و در عوض ایجاد یک شبکه ناپایدار می شود. در نهایت ، مقادیر وزن می توانند آنقدر بزرگ شوند که سرریز (overflow) رخ دهد و منجر به مقادیر NaN شوند. که در این صورت می‌گوییم انفجار حاصل از گرادیان (exploding gradient) رخ داده است. این انفجار از طریق رشد نمایی با ضرب مکرر شیب ها از طریق لایه های شبکه که مقادیر بزرگتر از یک دارند، رخ می دهد.

مشکل گرادیان در حال از بین رفتن (vanishing gradients) یکی از نمونه رفتارهای ناپایدار است که ممکن است در آموزش شبکه عصبی عمیق با آن روبرو شوید. این وضعیت را توصیف می کند که در آن یک شبکه feed forward چند لایه یا یک شبکه عصبی مکرر قادر به انتشار اطلاعات شیب مفید از انتهای خروجی مدل به لایه های نزدیک ورودی مدل نیست. در شبکه های عمیق یا شبکه های عصبی مکرر، شیب های خطا می توانند در هنگام بروزرسانی ترکیب شوند و منجر به شیب های بسیار کوچک شوند. این شیب‌ها می‌توانند آنقدر کوچک شوند که underflow رخ دهد و شیب صفر داشته باشیم. این شیب‌های بسیار کوچک به نوبه خود منجر به بروزر نشدن وزن شبکه می شوند. در این صورت می‌گوییم vanishing gradients رخ داده است. چون گرادیان بدست آمده در خروجی در لایه‌های اولیه ناپدید شده است و در این لایه‌ها تغییری رخ نمی‌دهد. این موضوع از طریق ضرب مکرر شیب ها از طریق لایه های شبکه که مقادیر کوچکتر از یک دارند، رخ می دهد.

سوال ۲ – CNN

ابتدا مجموعه داده را دانلود می‌کنیم. داده آموزش به ۵ قسمت تقسیم شده است. این ۵ قسمت را به صورت جدا لود می‌کنیم و به هم متصل می‌کنیم. از اتصال این قسمت‌ها داده آموزش با ۵۰۰۰۰ تا داده بدست می‌آید. هر عکس را نیز به شکل $(3 \times 32 \times 32)$ در می‌آوریم. همه‌ی اینکارها را برای داده تست به تعداد ۱۰۰۰۰ تا هم انجام می‌دهیم. در Figure 6 تعداد ۲۵ نمونه از داده‌های آموزش به نمایش در آمده است. برای انجام طبقه بندی نیاز است که اطلاعات هر پیکسل در بازه صفر تا یک به نمایش درآید. برای این نرمالسازی تایپ تمام اعداد یک عکس را float32 می‌کنیم و تمام اعداد را بر 255 تقسیم می‌کنیم. برای طبقه بندی چند کلاسه ما نیاز داریم که تمام لیبل‌های عکس‌ها را به صورت one-hot تبدیل کنیم. با توجه به این موضوع ما ۱۰ کلاس به ترتیب زیر داریم.

('plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck')

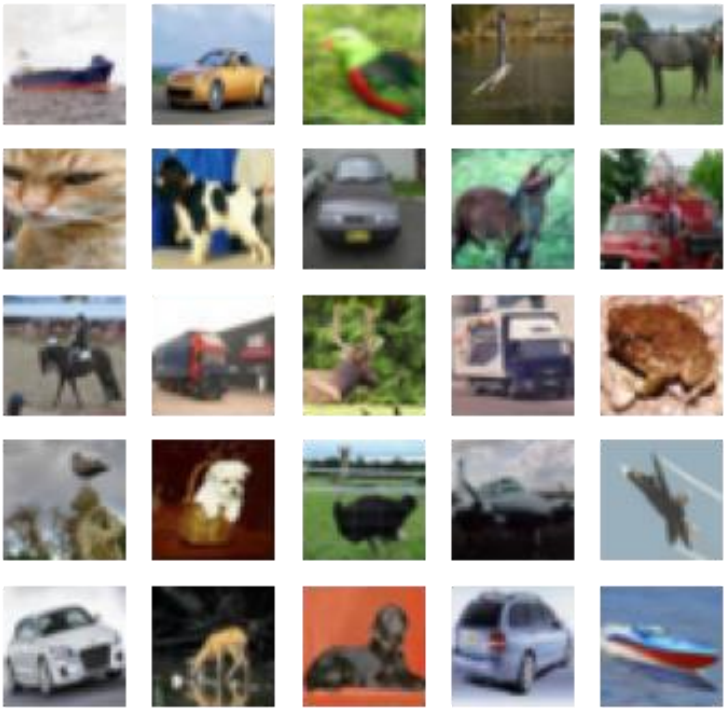

Figure 6 تعداد ۲۵ داده راندم از مجموعه آموزش

۱

ساختار شبکه پایه در Figure 7 آمده است.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 32)        896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 64)        36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 128)         73856
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 128)         147584
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 128)               262272
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 550,570
Trainable params: 550,570
Non-trainable params: 0
_________________________________________________________________
```

Figure 7 ساختار شبکه پایه

در شبکه از سه لایه کانولوشن پشت هم استفاده می‌کنیم. هر لایه شامل دو عمل کانولوشن پشت هم و یک عمل max-pooling به صورت pool_size=2 بعد دو عمل کانولوشن است. بعد از این لایه‌های کانولوشنی ویژگی‌های استخراجی به کمک flatten به یک بردار تبدیل می‌شوند. بعد آن از یک لایه مخفی لایه خروجی استفاده می‌کنیم. اندازه پنجره در تمام عمل‌های کانولوشن (3×3) است. در تمام عمل‌های کانولوشن از stride یک (1×1) و padding به شکل same استفاده می‌کنیم. در لایه اول کانولوشنی در تمام عمل‌های کانولوشن از فیلتر برابر ۳۲ استفاده می‌کنیم. در لایه دوم کانولوشنی در تمام عمل‌های کانولوشن از فیلتر برابر ۶۴ استفاده می‌کنیم. در لایه سوم کانولوشنی در تمام عمل‌های کانولوشن از فیلتر برابر ۱۲۸ استفاده می‌کنیم.

در تمام لایه‌های مخفی fully-connected و عمل‌های کانولوشن از تابع فعالساز relu استفاده می‌کنیم. در لایه خروجی از تابع فعالساز softmax استفاده می‌کنیم. در این شبکه ما از یک لایه مخفی یا fully-connected استفاده می‌کنیم که اندازه آن برابر ۱۲۸ است. جنس لایه مخفی dense است. لایه خروجی dense اندازه ۱۰ دارد. اگر بخواهیم یک لایه مخفی دیگر هم استفاده کنیم و دو لایه مخفی داشته باشیم اندازه آن ۶۴ خواهد بود.

در اینجا برای تابع هزینه از categorical_crossentropy و برای بهینه ساز از Adam استفاده می‌کنیم. برای آموزش از mini batch به اندازه‌ی ۶۴ استفاده می‌کنیم.

۲

مدل پیشنهادی را بر روی داده train آموزش می‌دهیم. نمودار تغییرات دقت بر حسب ایپاک بر روی داده‌های آموزش و تست در Figure 8 آمده است.

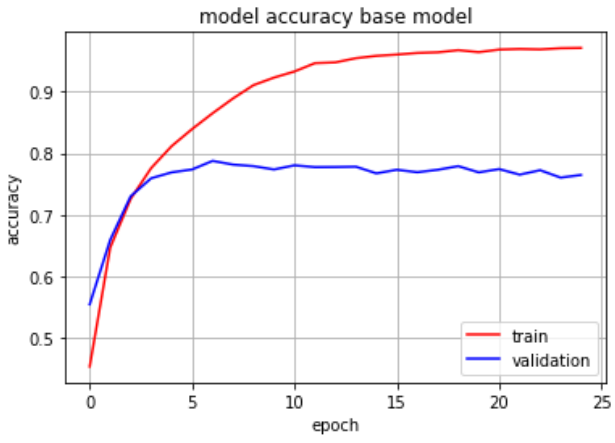

Figure 8 نمودار تغییرات دقت برحسب **epoch** بر روی داده آموزش و تست

همانطور که در Figure 8 آمده است، بعد از ۲۵ تا ایپاک نمودار دقت داده آموزش ثابت شده است و همگرایی رخ داده است. در طول این ایپاک ها نمودار دقت داده آموزش در حال افزایش بوده است و در انتها به یک مقدار بسیار نزدیک یک همگرا شده است. این روند نشان می‌دهد که مدل همواره در حال آموزش و بیشتر fit شدن به داده‌های آموزش بوده است. این درحالی است که نمودار دقت بر روی داده تست بعد از ۷ ایپاک روند صعودی به 78.78 درصد همگرا شده است و بعد آن اطراف این مقدار بوده است. بعد از همگرایی بهبودی در دقت بر روی داده تست نداشته ایم. اما بعد از ۷ تا ایپاک نمودار آموزش همچنان صعودی است. این یعنی بعد از ۷ تا ایپاک بیشتر بر روی داده آموزش fit شده ایم اما بر روی تست عملکردمان تغییری نداشته است. بعد از ۷ تا ایپاک، افزایش فاصله دو نمودار نشان دهنده افزایش overfit و کاهش قدرت تعمیم است. برای جلوگیری از overfit ما از روش ModelCheckpoint استفاده می‌کنیم. در این روش مقدار loss بر روی داده تست مانیتور می‌شود. زمانی که loss بر روی تست کمینه است، وزن‌های مدل را ذخیره می‌کنیم. این وزن‌ها تعیین کننده‌ی بهترین مدل بر روی داده تست در طول فرآیند آموزش است. این مدل زمانی بدست می‌آید که بعد از آن مدل بر روی تست یا بهبودی ندارد یا عملکردش بدتر می‌شود. این زمان overfit و فاصله نمودار تست از آموزش کم است و overfit نداریم.

بهترین دقت بر روی داده تست برابر 0.7878 است که در ایپاک ۷ ام رخ می‌دهد. بهترین دقت بر روی داده تست برابر 0.9709 است که در ایپاک ۲۵ ام رخ می‌دهد. در Figure 9 شماره ایپاک بهترین دقت با شروع از صفر و بهترین دقت در تست و آموزش آمده است.

```
train: 0.9709399938583374 24
test: 0.7878000140190125 6
```

Figure 9 بهترین دقت بر روی تست و آموزش و تعداد ایپاک برای رسیدن به این مقادیر

۳

در این قسمت عملکرد شبکه را با تغییر تعداد لایه مخفی (لایه fully-connected) بررسی می‌کنیم. مدل پایه ما در قسمت ۱ شامل یک لایه مخفی با ۱۲۸ نورون بود. نمودار تغییرات دقت بر روی داده تست و آموزش برای حالت یک لایه مخفی در Figure 8 آمده است. برای بررسی صفر لایه مخفی باید تمام پارامترهای شبکه و نحوه آموزش و بقیه ساختار شبکه ثابت و با حالت مدل پایه در قسمت ۱ یکسان باشد. ما نیز این شرایط را برقرار می‌کنیم و فقط لایه مخفی مدل پایه را حذف می‌کنیم و بردار ویژگی حاصل

بعد از flatten را مستقیما به لایه dense خروجی با ۱۰ نرون متصل می‌کنیم تا طبقه بندی انجام دهیم. نمودار تغییرات دقت بر روی داده تست و آموزش برای حالت صفر لایه مخفی در Figure 10 آمده است.

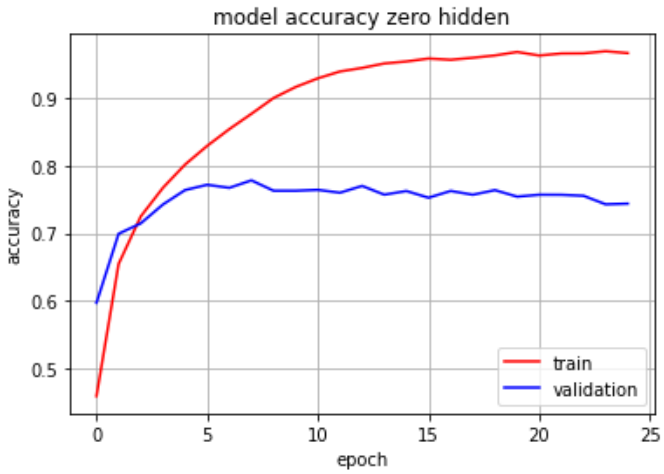

Figure 10 نمودار تغییرات دقت بر روی داده تست و آموزش با صفر لایه مخفی

همانطور که در Figure 10 دیده می‌شود، روند تغییرات دو نمودار تست و آموزش مانند حالت یک لایه در Figure 8 است. نمودار تست در ایپاک ۸ ام به مقدار حدکاثر خود یعنی 0.7779 می‌رسد. در Figure 11 مقدار حداکثر دقت در تست و آموزش در حالت صفر لایه مخفی و ایپاکی که در آن این مقادیر اتفاق می‌افتد، آمده است.

```
train: 0.9684000015258789 23
test: 0.777999997138977 7
```

Figure 11 حداکثر دقت در تست و آموزش در حالت صفر لایه مخفی و ایپاکی که در آن این مقادیر اتفاق می‌افتد

نمودار Figure 10 نشان می‌دهد که بعد از ۸ ایپال نمودار داده تست به 77 درصد میل کرده است و بعد از آن بهبودی نداشته است. این در حالی است که نمودار آموزش بعد از ۸ ایپاک در حال صعود بوده است. مقداری که در تست در این حالت به آن میل شده است از حالت یک لایه مخفی کمتر است. یعنی با افزایش یک لایه مخفی ما توانسته ایم به دقت بیشتری بر روی تست برسیم. در ایپاک ۵ ام آخرین بار loss بر روی تست کم می‌شود و در این حالت بهترین مدل را داریم. در این زمان دقت بهترین مدل بر روی تست 76.37 است. در حالت یک لایه مخفی دقت بهترین مدل بر روی تست در ایپاک ۶ ام برابر 77.39 درصد است. این مورد نشان می‌دهد که دقت بهترین مدل با یک لایه مخفی یک درصد بیشتر از حالت بدون لایه مخفی بر روی داده تست است. یعنی افزایش یک لایه مخفی عملکرد را بهتر کرده است. این درحالی است که اختلاف دقت تست و آموزش بهترین مدل بدون لایه مخفی 4.1 درصد است. اختلاف

دقت تست و آموزش بهترین مدل با یک لایه مخفی 6.65 درصد است. این اختلاف‌ها نشان می‌دهد با افزایش یک لایه مخفی تعداد پارامترها بیشتر شده و همچنین میزان overfit هم بیشتر شده است. اختلاف بیشتر بین تست و آموزش در حالت یک لایه نشان دهنده تعمیم کمتر این حالت نسبت به حالت بدون لایه مخفی است. همچنین دیدیم که در حالت یک لایه مخفی در یک ایپاک بیشتر بهترین مدل بدست می‌آید. که می‌تواند به خاطر پیچیده تر بودن و تعداد پارامتر بیشتر حالت یک لایه مخفی باشد. البته این اختلاف زمان چندان چشم‌گیر و قابل توجه نیست.

برای مقایسه حالت دو لایه مخفی با دو حالت دیگر نیاز است که فقط یک لایه مخفی به مدل پایه اضافه کنیم. این لایه اندازه ۶۴ نورون دارد. نمودار تغییرات دقت در آموزش و تست برای این حالت در Figure 12 آمده است.

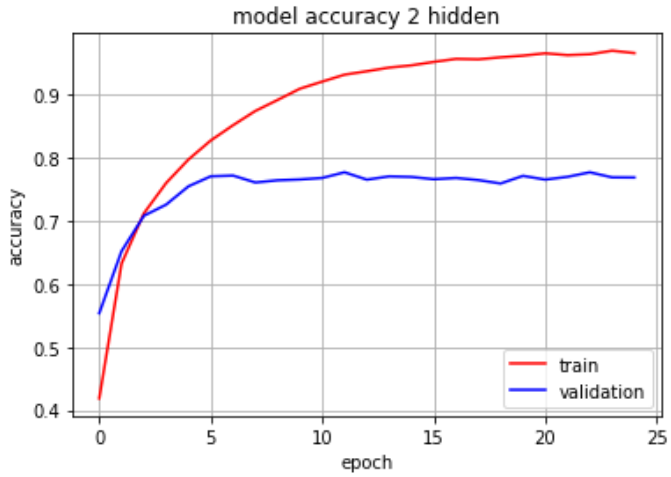

Figure 12 نمودار تغییرات دقت در آموزش و تست برای حالت دو لایه مخفی

روند تغییرات در Figure 12 مانند حالات قبل است. نمودار دقت بر روی داده تست بعد از ۶ ایپاک همگرا می‌شود و از آن به بعد اطراف آن حرکت می‌کند. نمودار آموزش در ۲۵ ایپاک کاملا صعودی حرکت می‌کند و در انتها همگرا می‌شود. بهترین مدل در این حالت دو لایه مخفی در ایپاک ۶ ام بدست می‌آید. دقت این بهترین مدل بر روی داده تست 0.7711 است. اختلاف دقت این بهترین مدل در تست و آموزش برابر 5.7 درصد است. این اختلاف بین تست و آموزش از حالت بدون لایه مخفی بیشتر است. یعنی قدرت تعمیم در آن از حالت بدون لایه مخفی کمتر است. اختلاف دقت تست و آموزش 0.9 در صد کمتر از حالت یک لایه مخفی است. این اختلاف 0.9 کم است و نشان می‌دهد قدرت تعمیم حالت دو لایه و یک لایه نزدیک هم است. دقت بهترین مدل با دو لایه مخفی بر روی داده تست کمتر از حالت یک لایه مخفی است. اختلاف دقت بر روی تست بهترین مدل در این دو حالت 0.3 درصد است. این اختلاف نیز کم است و نشان می‌دهد دقت بر روی تست با افزودن یک لایه مخفی بهتر نمی‌شود. این یعنی در حالت دو لایه مخفی با

تعداد بیشتر پارامتر بهبود عملکردی نسبت به حالت یک لایه مخفی نداریم. بیشترین مقدار دقت در تست و آموزش و ایپاک اتفاق افتادن این موارد با دو لایه مخفی در Figure 13 آمده است.

```
train: 0.9705399870872498 23
test: 0.7774999737739563 11
```

Figure 13 بیشترین مقدار دقت در تست و آموزش و ایپاک اتفاق افتادن این موارد با دو لایه مخفی

در Figure 13 بهترین دقت بر روی تست در حالت دو لایه مخفی از یک لایه مخفی کمتر و تقریبا برابر حالت بدون لایه مخفی است. این یعنی دو لایه مخفی نتوانسته حداکثر دقت روی تست را بهتر از حالات دیگر کند. همچنین می‌بینیم بهترین دقت روی تست در حالت دو لایه بسیار دیرتر و در تعداد بیشتری ایپاک بدست می‌آید. این موضوع می‌تواند به دلیل پیچیده تر بودن مدل در این حالت باشد.

دیدیم حالت یک لایه مخفی دقت بر روی تست بیشتری از حالت بدون لایه مخفی دارد و ما با افزودن یک لایه مخفی بهبود عملکرد داریم. با توجه به این موارد استفاده از دو لایه مخفی با توجه به افزودن پارامتر و نداشتن بهبود عملکرد به صرفه نیست. بهترین عملکرد برای حالت یک لایه مخفی است و بهترین ساختار، ساختار با یک لایه مخفی است.

۴

در این بررسی از مدل پایه استفاده می‌کنیم. در اینجا برای بررسی تابع فعالساز کانولوشن‌ها و لایه مخفی فقط تابع فعالساز را در هر مورد تغییر می‌دهیم و دیگر موارد در ساختار مدل و نحوه یادگیری را ثابت نگه می‌داریم. مدل پایه از تابع فعال ساز relu استفاده می‌کرد که نمودار تغییرات دقت بر داده تست و آموزش برای آن در Figure 8 آمده است. نمودار تغییرات دقت بر روی داده تست برای سه حالت تابع فعالساز relu و tanh و sigmoid در Figure 14 آمده است. همانطور که دیده می‌شود به ازای این ۱۰ ایپاک مقدار دقتی که مدل در حالت relu به آن میل کرده بیشتر از حالات دیگر است. در حالت tanh نیز دقتی که نمودار به آن میل کرده از حالت sigmoid بیشتر است. با توجه به بیشتر بودن دقت بعد از ۱۰ ایپاک در حالت relu این تابع فعالساز بهتر است. در تمام ایپاک‌ها نمودار مربوط به sigmoid زیر دو نمودار دیگر است. در نهایت نیز به مقدار دقت کمتر از دو حالت دیگر می‌رسد. در نمودار sigmoid تغییرات ۵۰ درصدی در دقت در این ۱۰ ایپاک داریم. در نهایت نیز این نمودار حالت صعودی خود را از دست می‌دهد و ثابت می‌شود. این شرایط نشان می‌دهد که دقت بر روی تست در همین حدود خواهد بود. نمودار مربوط به tanh تغییرات

بسیار کمی حداکثر ۱۰ درصد دارد. در ابتدا صعود اندکی دارد و در آخر تقریبا ثابت است و در ادامه نزول می‌کند. در کل نیز تغییرات نمودار در این حالت ناچیز است. این شرایط نشان می‌دهد که در ادامه نیز دقت بر روی تست کمتر مساوی ۷۵ درصد است. نمودار مربوط به relu تغییر ۲۰ درصدی در دقت دارد و بعد از ۱۰ ایپاک به دقت بالای ۷۷ درصد میل می‌کند.

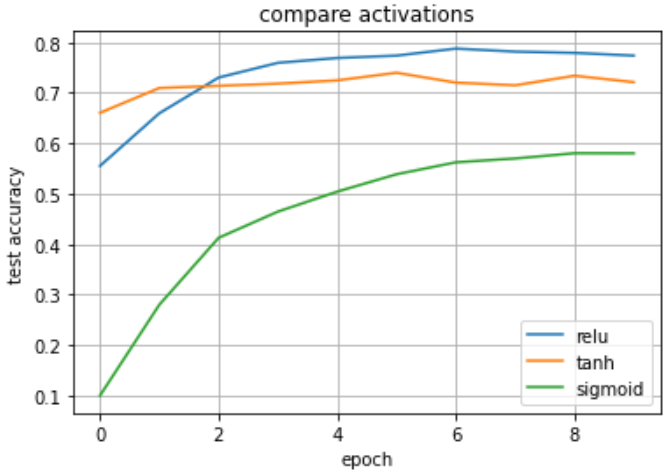

Figure 14 نمودار تغییرات دقت بر روی داده تست برای سه حالت تابع فعالساز

نمودار تغییرات دقت بر روی تست و آموزش در ۱۰ ایپاک برای حالت tanh در Figure 15 و در حالت sigmoid در Figure 16 آمده است.

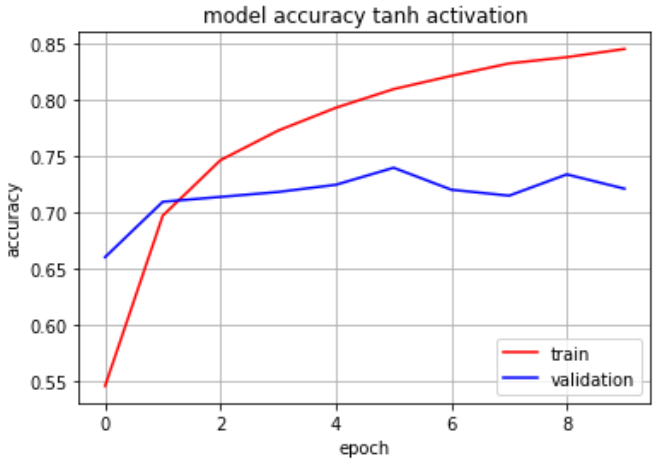

Figure 15 تغییرات دقت بر روی تست و آموزش در ۱۰ ایپاک برای حالت **tanh**

همانطور که در Figure 15 دیده می‌شود نمودار تغییرات آموزش در انتهای ۱۰ ایپاک در حال ثابت شدن است. این یعنی مقداری که نمودار تست به آن میل کرده که کمتر از ۷۵ درصد است، نهایت دقت بر روی تست این مدل است. چون مقداری که در حالت relu به آن میل شده بیشتر است، پس relu بهتر از

tanh است. دلیل تغییر کم نمودار تست در حالت tanh می‌تواند gradient vanishing و گیر کردن در مینیمم محلی نامطلوب باشد چرا که مشتق این تابع کمتر از یک و مثبت است.

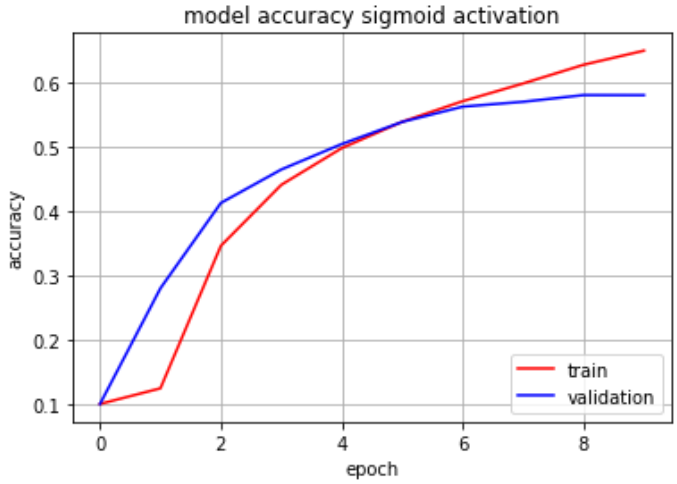

Figure 16 تغییرات دقت بر روی تست و آموزش در ۱۰ ایپاک برای حالت **sigmoid**

در حالت sigmoid در Figure 16 دیده می‌شود که نمودار تست ثابت شده است و به مقدار کمتر از ۶۰ میل کرده است. از آنجا که شیب نمودار آموزش در Figure 16 نیز کم شده است و نمودار نزدیک ثابت شدن است، به نظر نمی رسد که در ادامه دقت بر روی داده تست خیلی بیش تر شود. پس در این حالت نیز مقدار دقت میل شده در تست در ۱۰ ایپاک خیلی کمتر از مقدار میل شده در حالت relu است. این یعنی relu از sigmoid هم بهتر است.

۵

در این حالت برای مقایسه دو روش بهینه سازی از مدل پایه استفاده می‌کنیم. در هر دو حالت مدل یکسان و دیگر شرایط یادگیری مانند تابع هزینه، اندازه batch و ... یکسان است. در یادگیری مدل پایه ما از روش Adam برای بهینه سازی استفاده کردیم. در اینجا فقط کافی است از روش gradient descent برای بهینه سازی استفاده کنیم. نمودار تعییرات دقت بر روی داده تست و آموزش برای روش Adam در Figure 8 آمده است. نمودار تعییرات دقت بر روی داده تست و آموزش برای روش gradient descent در Figure 17 آمده است. همانطور که در Figure 17 دیده می‌شود، نمودار دقت داده آموزش صعودی است و در انتها شیب آن کم می‌شود و به ثابت شدن نزدیک می‌شود. این یعنی یادگیری نزدیک به تمام شدن است. با این شرایط نمودار تست در انتها تقریبا ثابت شده و نزول کرده است. مقدار بیشینه نمودار تست نزدیک ۵۵ درصد است. با توجه به نزدیک بودن اتمام یادگیری، مقدار دقت بر روی داده تست افزایش

زیادی نخواهد داشت. پس مقدار بیشینه نمودار تست معیار خوبی برای بیشینه دقت بر روی تست در این حالت است.

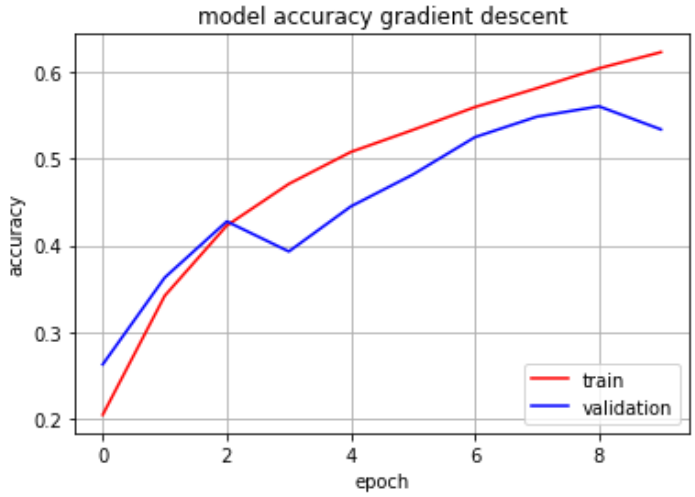

Figure 17 نمودار تغییرات دقت بر روی داده تست و آموزش برای روش **gradient descent**

در Figure 18 نمودار تغییرات دقت این دو روش بهینه سازی بر روی داده تست آمده است. همانطور که دیده می‌شود، نمودار حالت gradient descent همیشه زیر نمودار adam قرار دارد و فاصله این دو نمودار زیاد است و در هر ایپاک فاصله بیشتر از ۲۰ درصد است. بیشینه نمودار gradient descent برابر ۵۷ درصد است. با توجه به این که نمودار adam به مقدار ۷۸ درصد میل کرده است و پیشرفت نمودار gradient descent آنچنان زیاد نیست که این اختلاف ۲۰ درصد را در ادامه جبران کند، پس عملکرد روش adam بهتر است. به همین دلیل نمودار adam همیشه بالاتر است و اختلاف زیادی با حالت دیگر دارد.

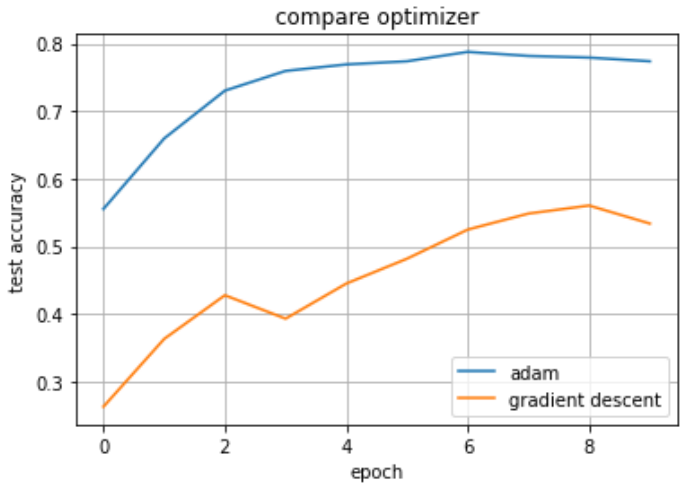

Figure 18 نمودار تغییرات دقت دو روش بهینه سازی در ۱۰ ایپاک

همانطور که در سوال اول دیدیم، روش گرادیان کاهشی دارای مشکلاتی است که روش adam آنها را حل کرده است. این موضوع باعث می‌شود که با روش adam ما یادگیری بهتری داشته باشیم و بتوانیم به کمینه بهتری برسیم. این داشتن یادگیری بهتر باعث می‌شود که دقت ما بر روی داده تست در روش adam در فرآیند یادگیری بهتر از روش گرادیان کاهشی باشد. به همین دلیل نمودار adam همیشه بالای نمودار گرادیان کاهشی است.

۶

در این جا تعداد نمونه‌های هر کلاس را در داده آموزش به ۶۰۰ تا می‌رسانیم. بعد از این کار مجموعه آموزش جدید را shuffle می‌کنیم تا داده‌های هر دسته پشت هم نباشند. این موضوع جلوی ایجاد بایاس در فرآیند یادگیری را می‌گیرد. وقتی داده‌های یک دسته پشت هم باشد مدل به خوبی به یادگیری بر اساس داده‌های آن کلاس می‌پردازد. در اینجا وزن‌ها طوری تعیین می‌شود که داده‌های یک دسته به خوبی مدل شوند. حال بعد از این داده‌ها، داده‌های دسته‌های دیگر داده می‌شود. در اینجا مدل نسبت به یک کلاس دیده شده بایاس است. هر داده را با احتمال بالایی عضو کلاس دیده شده می داند. با دیدن داده‌های دسته‌های دیگر به تدریج بایاس مدل کمتر می‌شود اما اینکار به سختی انجام می‌شود و ممکن است باعث شود یادگیری در کل به خوبی انجام نشود. حتی ممکن است بایاس آنقدر قوی باشد که مدل بعد از دیدن داده‌های دیگر نتواند این بایاس را برطرف کند. بعد از این کاهش نمونه، از مدل پایه برای آموزش بر روی داده آموزش جدید استفاده می‌کنیم. نمودار تغییرات دقت بر روی داده آموزش جدید و تست در Figure 19 آمده است.

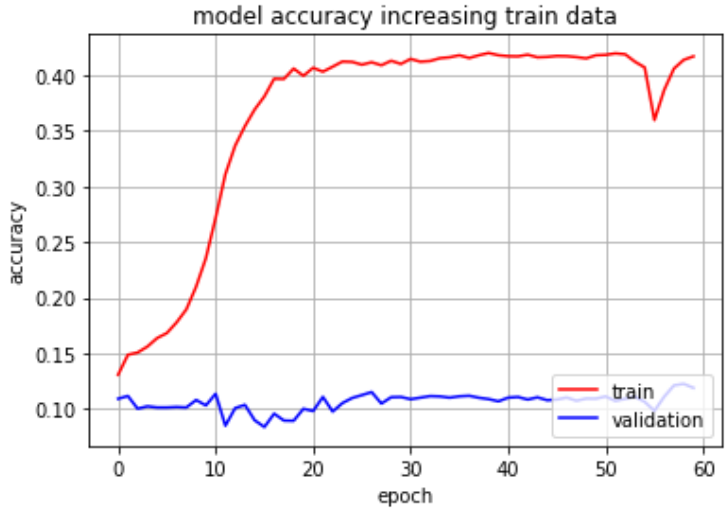

Figure 19 نمودار تغییرات دقت بر روی داده آموزش جدید و تست

شرایط آموزش مانند قسمت ۱ است. همانطور که در Figure 19 دیده می‌شود، نمودار دقت بر روی داده‌های آموزش به ۴۲ درصد بعد از ۲۲ ایپاک میل می‌کند. دقت بر روی داده آموزش بیش از این ۴۲ درصد نمی‌شود. همچنین می‌بینیم که دقت بر روی تست روی ۱۰ درصد تقریبا ثابت می‌ماند. همانطور که دیده می‌شود ابتدا نمودار آموزش افزایش می‌یابد و سپس به ۴۲ درصد میل می‌کند. بعد از ۶۰ تا ایپاک هم دقت بر روی داده آموزش نزدیک یک نمی‌شود و ثابت روی ۴۲ درصد مانده است. این شرایط نمودار آموزش و تست نشان دهنده‌ی underfit است. یعنی تعداد داده‌ها خیلی کم است و نمی‌توانیم مدل دقیقی را برای fit شدن خوب بر روی داده آموزش یاد بگیریم. عملا کمبود دیتا باعث شده یادگیری خوبی نداشته باشیم. به همین دلیل عملکرد ما بر روی داده دیده نشده تقریبا ثابت و بسیار پایین است.

۷

در مدل پایه ما در هر لایه کانولوشنی از دو عمل کانولوشن پشت هم استفاده می‌کنیم. دو عمل کانولوشن پشت هم دارای کرنل هم ساز (3×3) است. برای تبدیل این دو عمل به یک عمل، باید از کرنل (5×5) استفاده کنیم. در Figure 20 نحوه ارتباط یک خانه از feature map خروجی کانولوشن دوم به خانه‌های ماتریس ورودی آمده است.

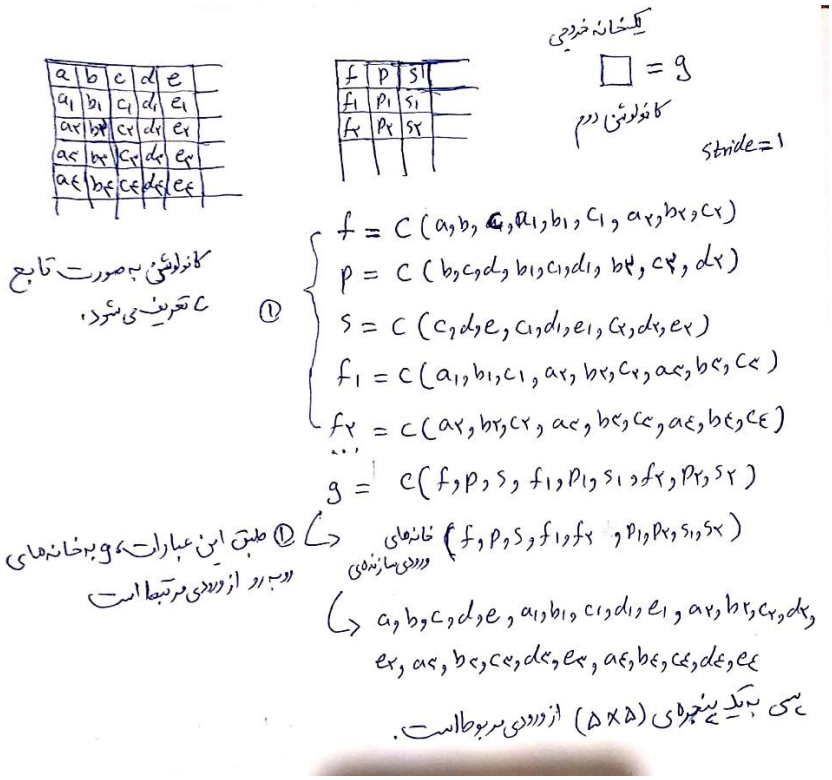

Figure 20 نحوه ارتباط یک خانه از **feature map** خروجی کانولوشن دوم به خانه‌های ماتریس ورودی در دو عمل کانولوشن پشت هم

بر اساس همین ارتباط دو عمل کانولوشن پشت هم با کرنل ۳ در ۳ را باید به یک کانولوشن ۵ در ۵ تبدیل کرد. در مدل پایه در تمام لایه ها به همین شکل از یک لایه کانولوشن استفاده می‌کنیم تا مدل جدید بدست آید. دیگر بخش‌های ساختار مدل پایه را بدون تغییر می‌گذاریم. مدل جدید و پایه را در یک شرایط بر روی داده train آموزش می‌دهیم. نمودار تغییرات دقت روی تست و آموزش برای مدل با یک عمل کانولوشن در هر لایه در Figure 21 آمده است.

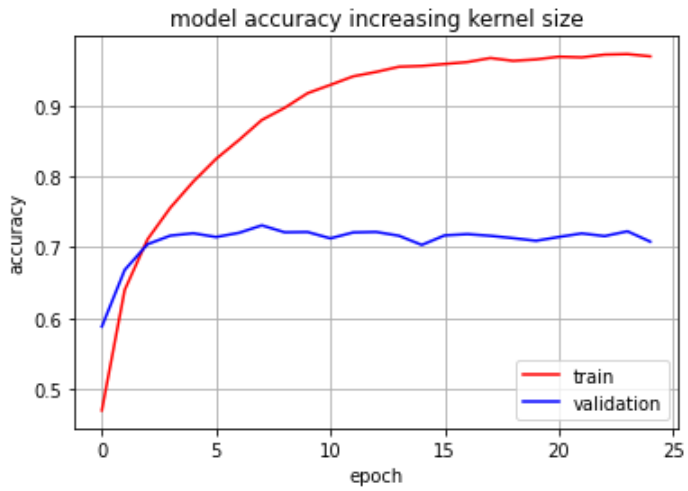

Figure 21 نمودار تغییرات دقت بر روی آموزش و تست با تبدیل دو عمل کانولوشن پشت هم به یک عمل

همانطور که در Figure 21 دیده می‌شود، با تبدیل دو کانولوشن به یکی باز روند تغییر دقت بر روی تست و آموزش مانند قسمت ۱ است. نمودار تست بعد از افزایش ابتدایی به مقدار ۷۲ درصد میل کرده است. دقت بهترین مدل بر روی داده تست برابر 0.7166 است و اختلاف دقت این بهترین مدل که با model checkpoint بدست آمده روی آموزش و تست برابر 11.85 درصد است. این اختلاف از حالت مدل پایه بسیار بیشتر است و یعنی از قسمت ۱ و مدل پایه قدرت تعمیم کمتری داریم. همچنین دقت بهترین مدل با این تبدیل بر روی داده تست از بهترین مدل با ساختار مدل پایه به اندازه 7.2 درصد کمتر است. این یعنی این تبدیل عملکرد مدل را بدتر از مدل پایه کرده است. این موضوع در Figure 22 دیده می‌شود که تغییرات دقت این مدل جدید و مدل پایه بر روی داده تست است. همانطور که دیده می‌شود مقدار دقتی که نمودار مدل جدید به آن میل کرده پایین تر از مقداری است که مدل پایه به آن میل کرده است. اختلاف این دو مقدار بالای ۵ درصد می‌باشد و قابل توجه است.

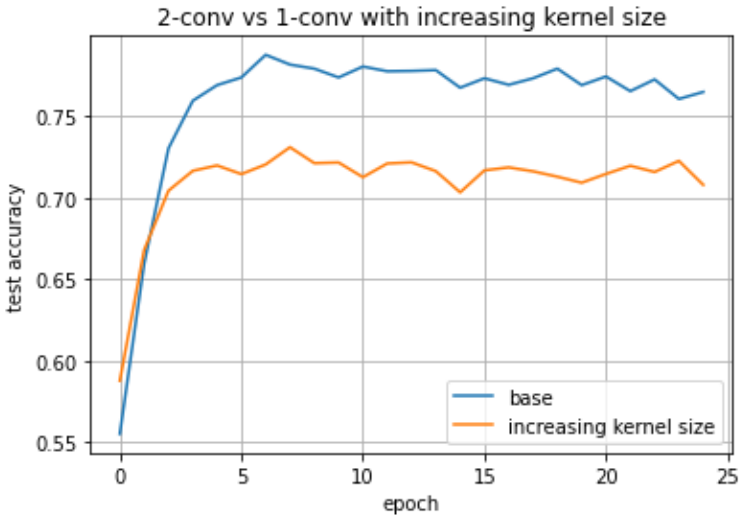

Figure 22 تغییرات دقت این مدل جدید و مدل پایه بر روی داده تست

جایگزین کردن کانولوشن با کرنل بزرگ با دو کانولوشن با کرنل کوچکتر را دیدیم. دیدیم که این کار باعث افزایش دقت مدل بر روی داده تست و افزایش قدرت تعمیم مدل می‌شود. این کار عملکرد مدل را بهبود می‌بخشد. با تبدیل کانولوشن با کرنل بزرگ به تعدادی کانولوشن با کرنل کوچکتر، ما عمق شبکه عصبی را بیشتر می‌کنیم. با این کار تعداد لایه‌های خلاصه سازی در مدل بیشتر می‌شود و در این شرایط ویژگی‌های استخراجی، ویژگی‌های کلی تر و جنرالتری خواهند بود. دلیل این جنرالتر بودن خلاصه تر کردن ورودی است. این کار باعث می‌شود که به ویژگی‌های کلی برسیم که در جامعه اصلی داده‌ها باعث جداپذیری خوب دسته ها می‌شوند. رسیدن به ویژگی‌های جنرالتر باعث افزایش قدرت تعمیم و بهبود عملکرد می‌شود.

۸

برای بررسی این موضوع از مدل پایه استفاده می‌کنیم. در مدل پایه بعد از هر تابع فعالساز relu از dropout استفاده می‌کنیم تا مدل جدید را بسازیم. احتمال تمام dropout ها را یکسان در نظر می‌گیریم. برای پیداکردن مقدار بهینه احتمال، این مدل جدید را با احتمال 0.2 و 0.3 و 0.5 بر روی داده train آموزش می‌دهیم. عملکرد مدل جدید در این سه حالت را مقایسه می‌کنیم. مقدار احتمالی بهینه است که دقت روی داده تست را بیشتر می‌کند. برای این سه حالت احتمال dropout نمودار تغییرات دقت بر روی داده تست و آموزش به ترتیب در Figure 23 و Figure 24 و Figure 25 آمده است. همانطور که در Figure 23 دیده می‌شود، نموار تست در حالت احتمال ۰.۲ به مقدار دقت ۸۰ درصد میل کرده است. در حال احتمال ۰.۳ در Figure 24 می‌بینیم که نمودار تست به مقدار بالای ۸۰ درصد و نزدیک ۸۲ میل

کرده است. در حالت احتمال ۰.۵ در Figure 25 می‌بینیم که نمودار تست به مقداری کمی بالاتر از ۷۰ درصد میل کرده است. بر اساس مقادیری که نمودارهای تغییرات دقت بر روی تست در این سه حالت به آن میل کرده‌اند، احتمال ۰.۳ بهترین احتمال است. چون باعث شده است که مدل به دقت بالای ۸۰ و بالا تر از حالات دیگر میل کند و بر روی آن ثابت شود.

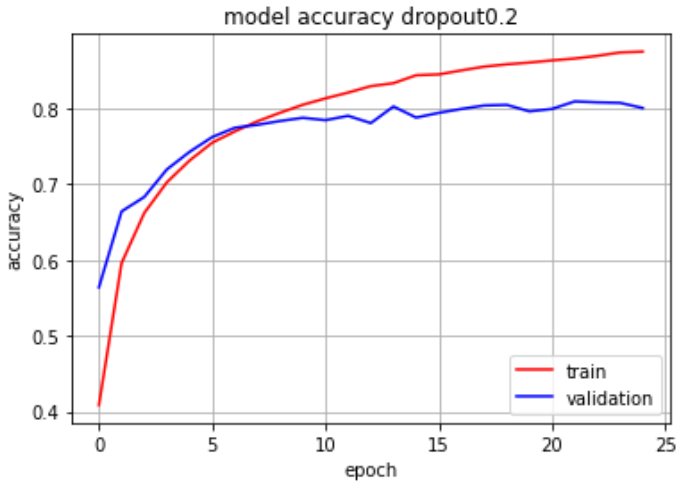

Figure 23 تغییرات دقت روی تست و آموزش برای مدل پایه با **dropout** به احتمال ۰.۲

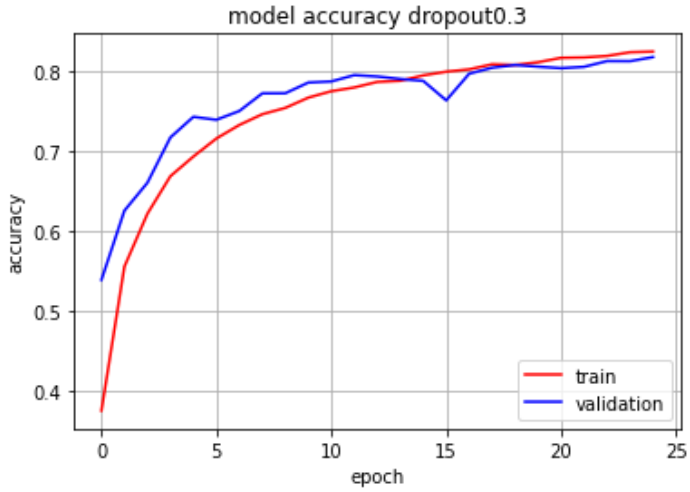

Figure 24 تغییرات دقت روی تست و آموزش برای مدل پایه با **dropout** به احتمال ۰.۳

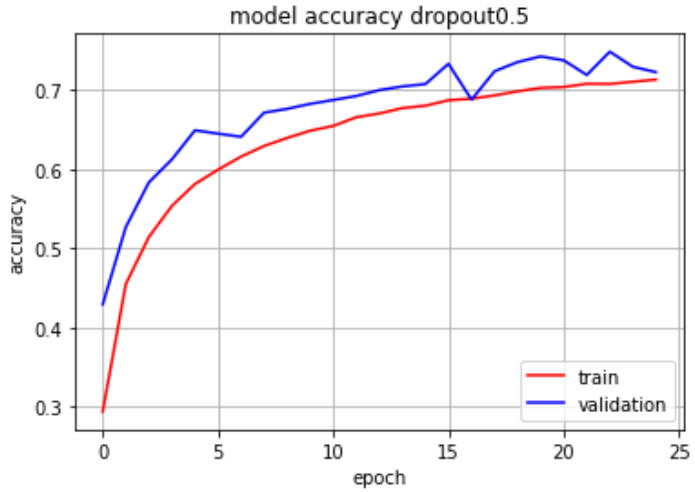

Figure 25 تغییرات دقت روی تست و آموزش برای مدل پایه با **dropout** به احتمال ۰.۵

پس از احتمال ۰.۳ برای ساخت مدل جدید استفاده می‌کنیم. مدل جدید را به صورت یکسان با مدل پایه آموزش می‌دهیم. نمودار تغییرات دقت این دو مدل بر روی داده تست و آموزش به ترتیب در Figure 26 و Figure 27 آمده است.

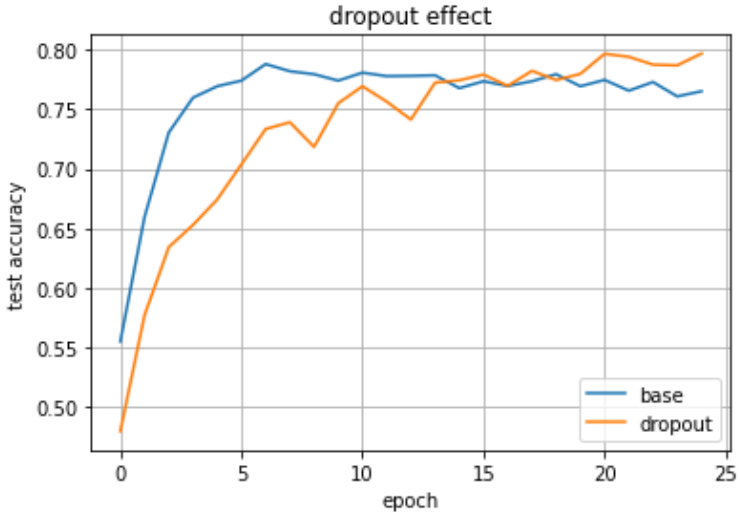

Figure 26 تغییرات دقت مدل پایه با و بدون **dropout** روی داده تست

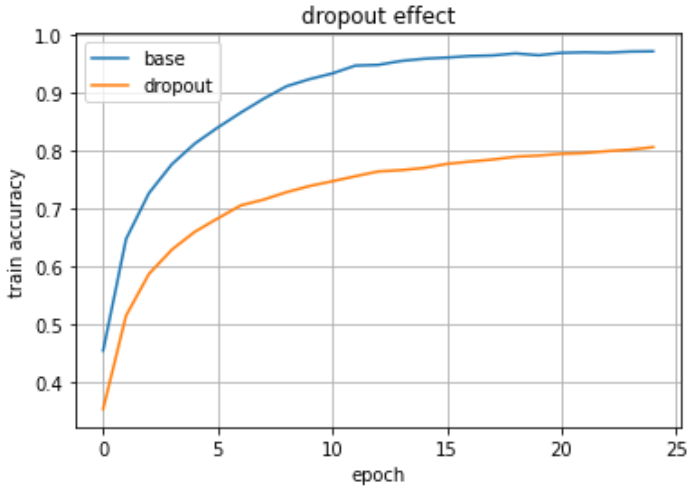

Figure 27 تغییرات دقت مدل پایه با و بدون **dropout** روی داده آموزش

همانطور که در Figure 27 دیده می‌شود، مقدار میل شده توسط مدل با dropout در آموزش کمتر است. این یعنی مدل با dropout کمتر توانسه بر روی داده‌های آموزش fit شود. دلیل این موضوع حذف شدن تعدادی از نورون‌های شبکه به خاطر dropout است. با این کار dropout تعدادی نورون را عملا در زمان یادگیری حذف می کنیم که باعث می‌شود پیچیدگی قابل پوشش و پارامترهای مدل در هنگام آموزش کمتر شود. به همین دلیل مدل ساده شده نمی تواند به خوبی قبل بر روی داده‌ها fit شود. در Figure 26 می‌بینیم که نمودار مدل پایه بعد از ۷ ایپاک تقریبا روی ۷۷ ثابت می‌شود. این درحالی است که نمودار مدل با dropout بعد از ۲۰ ایپاک بر روی عددی نزدیک ۸۰ ثابت می‌شود. این یعنی در حالت dropout ما به دقت بیشتری بر روی داده تست می‌رسیم. این نشان می‌دهد افزودن dropout عملکرد را بهبود می‌دهد و قدرت تعمیم را افزایش می‌دهد. در حالت dropout بهترین مدل دقت 79.66 روی داده تست دارد و اختلاف دقت تست و آموزش در این مدل برابر یک درصد است. این اختلاف کم میزان کم overfit و تعمیم بالا در این حالت را نشان می‌دهد. همانطور که در سوال اول تمرین بررسی کردیم، dropout باعث می‌شود که ما از ensemble چند مدل با ساختار شبکه متفاوت استفاده کنیم. این موضوع باعث می‌شود که از overfit جلوگیری شود و قدرت تعمیم مدل بدست آمده از ensemble تعدادی مدل را داشته باشیم. یعنی dropout قدرت تعمیم مدل را بهبود می‌بخشد. به همین دلیل ما عملکرد بهتری با dropout داشتیم.

سوال 3 – Data Augmentation

۱

افزایش داده های تصویری یا Image data augmentation تکنیکی است که می تواند با ایجاد نسخه های اصلاح شده و تغییر یافته تصاویر در مجموعه داده ، به طور مصنوعی اندازه مجموعه داده های آموزشی را گسترش دهد. آموزش مدل های شبکه عصبی با یادگیری عمیق در مورد داده های بیشتر می تواند به مدل های ماهرانه تری منجر شود و تکنیک های افزایش می توانند تغییراتی در تصاویر ایجاد کنند که می‌تواند توانایی مدل های متناسب را برای تعمیم آنچه که آموخته اند به تصاویر جدید بهبود بخشد. تغییرات تصاویر به گونه‌ای هستند که ماهیت تصویر و لیبل آن را تغییر نمی‌دهند.

عملکرد شبکه های عصبی یادگیری عمیق اغلب با افزایش مقدار داده موجود بهبود می یابد. افزایش داده ها تکنیکی است برای ایجاد مصنوعی داده های آموزشی جدید از داده های آموزش موجود. این کار با استفاده از تکنیک های خاص دامنه بر روی مثال هایی از داده های آموزشی که نمونه های جدید آموزشی را ایجاد می کند، انجام می شود. افزایش داده های تصویری شاید شناخته شده ترین نوع افزایش داده ها باشد و شامل ایجاد نسخه های تغییر شکل یافته از تصاویر در مجموعه آموزش است که متعلق به همان کلاس تصویر اصلی است. تحولات شامل طیف وسیعی از عملیات در زمینه دستکاری تصویر ، مانند شیفت ، فلیپ ، بزرگنمایی و موارد دیگر است. هدف این است که مجموعه داده های آموزشی را با مثال های جدید و قابل قبول گسترش دهیم. این به معنای تغییراتی در مجموعه تصاویر آموزشی است که احتمالاً توسط مدل دیده می شود. به عنوان مثال ، یک فلیپ افقی از یک عکس از یک گربه ممکن است منطقی باشد ، زیرا ممکن است عکس از سمت چپ یا راست گرفته شده باشد. فلیپ عمودی عکس گربه منطقی نیست و با توجه به اینکه بعید به نظر می رسد که مدل عکس گربه وارونه را ببیند، مناسب نخواهد بود. به همین ترتیب ، واضح است که انتخاب تکنیک های خاص افزایش داده مورد استفاده برای یک مجموعه داده آموزشی ، باید با دقت و در چارچوب مجموعه داده آموزش و دانش حوزه مسئله انتخاب شود. بعلاوه، آزمایش روشهای افزایش داده به طور جداگانه با یک نمونه اولیه از مجموعه داده آموزش برای دیدن اینکه آیا منجر به بهبود قابل اندازه گیری در عملکرد مدل می شوند ، می‌تواند مفید باشد. الگوریتم های مدرن یادگیری عمیق ، مانند شبکه عصبی کانولوشن ، یا CNN ، می توانند ویژگی هایی را بیابند که از نظر موقعیت مکانی در تصویر ثابت نیستند. با این وجود ، این افزایش داده آموزش می تواند به مدل در یادگیری ویژگی هایی کمک کند که در تبدیل هایی مانند ترتیب از چپ به راست به بالا به پایین ، سطح نور در عکس ها و موارد دیگر نیز بی تغییر هستند.

افزایش داده های تصویر معمولاً فقط برای مجموعه داده های آموزشی اعمال می شود و نه برای داده های ارزیابی یا تست. این کار متفاوت از آماده سازی داده ها مانند تغییر اندازه تصویر و اندازه گیری پیکسل است. آماده سازی داده ها باید به طور مداوم در تمام مجموعه های داده ای که با مدل تعامل دارند انجام شود.

Data Augmentation فقط بر روی داده‌های آموزش انجام می‌شود. علت آن دلیل استفاده از این روش است. ما از این روش برای افزایش داده برای انجام یادگیری بهتر استفاده می‌کنیم. به همین دلیل زمانی که مجموعه آموزش تعداد داده کمی داشته باشد یا داده‌های دسته‌های آن متوازن و نامرغوب باشد، از این روش برای افزایش داده‌های آموزش و بالانس کردن دسته‌ها در آموزش و افزایش مرغوبیت داده آموزش استفاده می‌کنیم. اگر داده‌های یک دسته در آموزش از دیگر دسته‌ها خیلی کمتر باشد، مدل یادگیری خوبی نخواهد داشتت و نسبت به کلاس با داده کم بایاس پیدا می‌کند. برای عکس جدید به علت تعداد کم داده آموزش کلاس مورد نظر، احتمال تعلق عکس به کلاس با تعداد کم را بسیار پایین در نظر می‌گیرد. این موضوع مستقل از ویژگی‌ها و خود عکس است. داده‌های ارزیابی داده‌هایی دیده نشده برای بررسی عملکرد مدل آموزش دیده و تعیین پارامترهای ساختاری آن مدل هستند. با توجه به این هدف نیازی به بالانس کردن و افزایش این داده‌ها نداریم. داده‌های تست داده‌هایی برای مقایسه عملکرد دو مدل مختلف هستن که باید برای تمام مدل‌ها ثابت و یکسان باشد. به همین دلیل نباید تغییری در این داده‌ها ایجاد کنیم و باید این داده‌ها دیده نشده باقی بمانند. به همین دلیل از data augmentation برای داده‌های تست استفاده نمی‌کنیم.

تبدیل‌های که در این روش انجام می‌شوند و ماهیت تصویر را تغییر نمی‌دهند: دوران، شیفت در جهت افقی به چپ و راست، شیفت در جهت عمودی به بالا و پایین، روشنایی تصویر (brightness)، shear با جابه جایی هر نقطه در جهت مبنا با اندازه ثابت یا برش (باعث کشیده شدن در جهت یک زاویه می‌شود)، zoom ، channel_shift که باعث تغییر راندم در کانال‌های (رنگ‌های) تصویر می‌شود، flip عمودی و افقی(در حالت افقی تصویز اپ چپ به راست را راست به چپ می‌کند. در جهت عمودی به صورت مشابه بالا به پایین را پایین به بالا می‌کند) و rescale که برای اسکیل کردن است.

۲

در اینجا از ترکیبی از تبدیل‌های موجود برای augmentation برای تولید تصویر جدید استفاده می‌کنیم. ترکیب تبدیل‌های مورد استفاده در Figure 28 آمده است.

```
# create image data augmentation generator
datagen = ImageDataGenerator(width_shift_range=0.2,
                             height_shift_range=0.2,
                             horizontal_flip=True, vertical_flip=True,
                             rotation_range=90,
                             brightness_range=[0.2,1.0],
                             zoom_range=[0.5,1.0])
```

Figure 28 ترکیب تبدیل‌ها برای **augment** کردن تصویر تست

تصویر اصلی به همراه ۱۰ تصویر تولید شده با augmentation و ترکیب تبدیل‌های بیان شده از آن در Figure 29 آمده است.

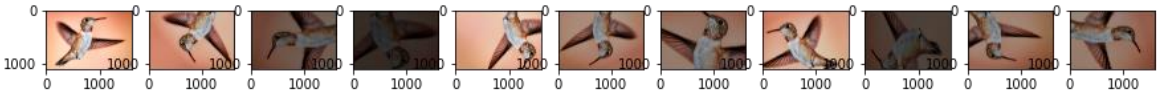

Figure 29 تصویر اصلی به همران ۱۰ تصویر ایجاد شده با **augmentation**

۳

در این بخش داده cifar10 را مانند سوال سه لود می‌کنیم. تمام پیش پردازش و کارهای لازم برای آمده سازی این مجموعه داده برای طبقه بندی را مانند سوال دو انجام می‌دهیم. بعد از این موارد در داده آموزش، داده‌های مربوط به کلاس سگ و گربه را پیدا می‌کنیم. این دو کلاس هر یک ۵۰۰۰ داده دارند. از هز یک از کلاس‌ها ۴۵۰۰ داده به صورت راندم حذف می‌کنیم. سپس داده‌های آموزش بعد از این مرحله را shuffle می‌کنیم. این کاهش داده در Figure 30 آمده است.

```
import random

c = []
d = []
for i, v in enumerate(label):
  if np.argmax(v)==3:
    c.append(i)
  elif np.argmax(v)==5:
    d.append(i)

c = random.sample(c, 4500)
d = random.sample(d, 4500)
label = np.delete(label, c+d, 0)
train = np.delete(train, c+d, 0)
```

Figure 30 کاهش نمونه‌های سگ و گربه

بعد از این کاهش داده آموزش، بهترین مدل سوال دو را بر روی داده آموزش کاهش یافته، آموزش می‌دهیم. بهترین مدل مدل پایه در سوال دو به همراه dropout با احتمال 0.3 است. بعد از آموزش مدل بدست آمده را روی داده تست ارزیابی می‌کنیم. ماتریس آشفتگی برای داده‌های تست در Figure 31 آمده است. مقدار متریک‌های ارزیابی برای طبقه بندی نیز برای این حالت در Figure 32 آمده است.

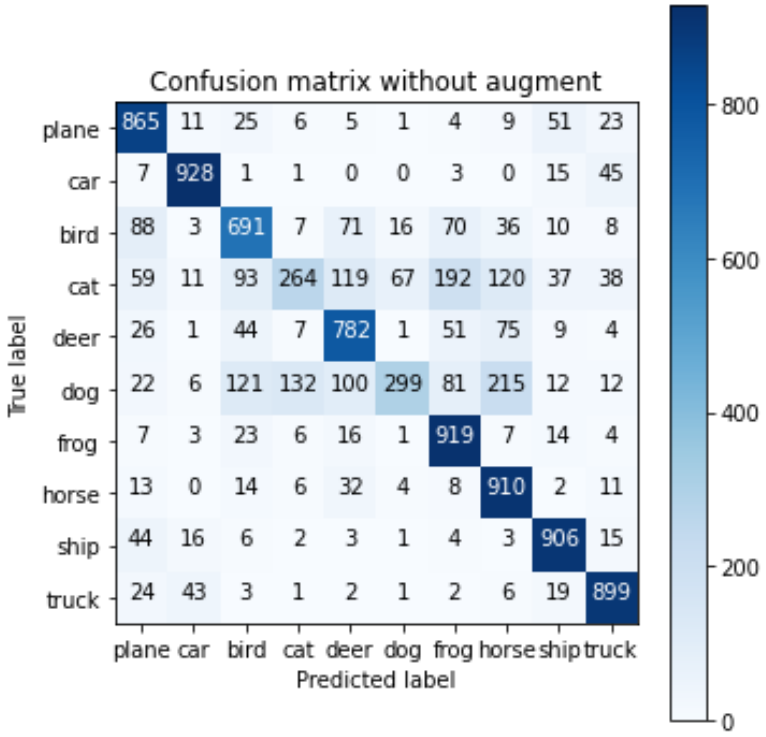

Figure 31 ماتریس آشفتگی مدل با داده آموزش کاهش یافته روی داده تست

```
Test Loss 0.8340916037559509
Test Accuracy 0.7462999820709229
              precision    recall  f1-score   support

           0       0.75      0.86      0.80      1000
           1       0.91      0.93      0.92      1000
           2       0.68      0.69      0.68      1000
           3       0.61      0.26      0.37      1000
           4       0.69      0.78      0.73      1000
           5       0.76      0.30      0.43      1000
           6       0.69      0.92      0.79      1000
           7       0.66      0.91      0.76      1000
           8       0.84      0.91      0.87      1000
           9       0.85      0.90      0.87      1000

    accuracy                           0.75     10000
   macro avg       0.74      0.75      0.72     10000
weighted avg       0.74      0.75      0.72     10000

f1 0.7235767556224577
precision 0.7441134197103221
recall 0.7463
accuracy 0.7463
```

Figure 32 متریک‌های ارزیابی طبقه بندی برای مدل روی داده آموزش کاهش یافته

همانطور که در Figure 31 دیده می‌شود، این تعداد کمتر داده در دو دسته گربه و سگ باعث شده که مدل نتواند این دو دسته را به خوبی از دیگر دسته‌ها متمایز کند. همانطور که در ماتریس آشفتگی دیده می‌شود، از ۱۰۰۰ داده در هر یک از این دسته‌ها تعداد حدود 0.75 از داده‌های هر دسته به دسته‌های دیگر نسبت داده شده است. این موضوع در ردیف هر یک از کلاس‌ها قابل مشاهده است. در ردیف مربوط به هر یک از این دو کلاس دیده می‌شود که در ستون‌های غیر از ستون مربوط به کلاس ردیف، اعداد بزرگی وجود دارد. برای مثال در ردیف گربه می‌بینیم که ۱۹۲ داده گربه به اشتباه قورباغه گفته شده است. ۱۲۰ داده هم به اشتباه اسب گفته شده است. ۴ دسته از دسته‌های اشتباه گرفته شده با گربه عدد بالای ۹۰ دارند. در دسته سگ نیز ۴ دسته اشتباه گرفته شده با آن عدد بالای ۹۰ دارند. پس تعداد زیادی نزدیک 0.75 داده از هر دو دسته سگ و گربه به اشتباه به دسته‌های دیگر نسبت داده شده است. همچنین می‌بینیم که داده‌های اشتباه طبقه بندی شده در هر دو گروه در تمام دسته‌های غیر مرتبط توزیع شده است و اشتباه های هر دسته فقط مربوط به اشتباه گرفتن با یک یا تعداد بسار کمی دسته نیست. این توزیع شدگی اشتباهات در این دو دسته و تعداد بالای اشتباهات در این دو دسته نشان می‌دهد که یادگیری برای گربه و سگ به خوبی نبوده است و می‌توان گفت یادگیری برای تمایز دو دسته سگ و گربه از دیگر دسته‌ها نداشته ایم. کمبود داده در این دو دسته باعث شده نتوانیم این دو دسته را مدل کنیم و از دیگر دسته‌ها متمایز کنیم. این اختلال به وجود آمده به خاطر داده‌های کمتر این دسته ها از دیگر دسته‌ها است. این یادگیری ضعیف برای دو دسته باعث پایین بودن recall و precision این دسته نسبت به دیگر دسته‌ها می‌شود. این موضوع در Figure 32 آمده است.

۴

در اینجا از ترکیبی از تبدیل‌ها برای augmentation استفاده می‌کنیم و داده‌های هر دسته سگ و گربه را که در داده آموزش کاهش یافته قسمت ۳ برابر ۵۰۰ بود را به ۵۰۰۰ داده می‌رسانیم. ترکیب مورد استفاده از تبدیل‌ها برای augmentation داده کاهش یافته cifar10 در Figure 33 آمده است. برای افزایش داده، از هر داده‌ی هر یک از دسته‌ها ۹ داده جدید با ترکیب تبدیل‌ها برای augmentation ایجاد می‌کنیم.

بعد از افزایش داده‌ها از مدل و شرایط یادگیری مشابه به قسمت قبل برای یادگیری استفاده می‌کنیم و مدل بدست آمده را بر روی داده تست مشابه قسمت قبل ارزیابی می‌کنیم.

```
# create image data augmentation generator
datagen = ImageDataGenerator(width_shift_range=0.2,
                             height_shift_range=0.2,
                             horizontal_flip=True, vertical_flip=True,
                             rotation_range=90,
                             brightness_range=[0.2,1.0],
                             zoom_range=[0.5,1.0])
```

Figure 33 ترکیب مورد استفاده از تبدیل‌ها برای **augmentation** داده کاهش یافته **cifar10**

ماتریس آشفتگی برای مدل آموزش دیده با داده augment شده در Figure 34 و متریک‌های ارزیابی طبقه بند در Figure 35 آمده است.

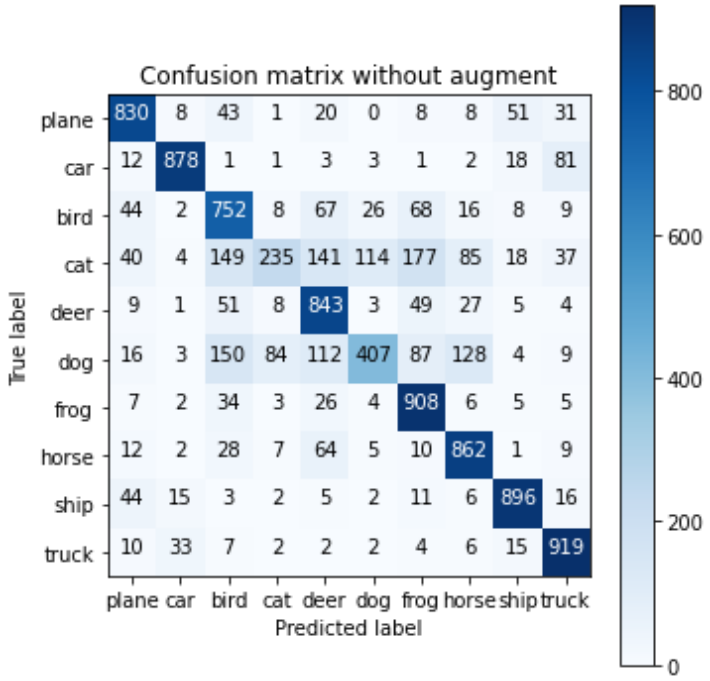

Figure 34 ماتریس آشفتگی برای مدل آموزش دیده با داده **augment** شده

همانطور که در Figure 34 دیده می‌شود، تعداد داده‌های اشتباه دسته بندی شده از دیتاها با لیبل سگ به ۶۰ درصد نسبت به قسمت قبل کاهش یافته است. تعداد دسته‌های اشتباه گرفته شده با دسته سگ که تعداد بالای ۹۰ دارند به سه دسته کاهش یافته است. این موارد نشان می‌دهد با دیتای augment شده مدل توانسته دسته سگ را بهتر از دیگر دسته‌ها متمایز کند. این یعنی یادگیری بهتری صورت گرفته است. در مورد دسته گربه تعداد داده درست تشخیص داده کمی به اندازه ۲۹ تا کمتر شده است. تعداد داده گربه که اشتباه به عنوان سگ تشخیص داده شده افزایش یافته است. تعداد دسته اشتباه تشخیص داده شده با تعداد بالای ۹۰ برای داده‌های گربه مانند قبل است. نکته قابل توجه برای اشتباهات در زمینه

داده‌های گربه، نزدیکتر شدن دسته‌های اشتباه با دسته گربه است. در قسمت قبل تعداد ۱۲۰ تا اشتباه در تشخیص گربه به عنوان اسب داشتیم. در اینجا تعداد این اشتباه به ۸۵ مورد رسیده است. به جای اشتباه با دسته دور اسب، اشتباه با دسته نزدیگ گربه بیشتر شده است. این موارد در مورد دسته گربه داده‌ها نشان می‌دهد که یادگیری بهتری درباره گربه‌ها داشته ایم که دلیل آن نزدیکتر شدن لیبل اشتباه برای داده‌های گربه به گربه است. همچنین می‌بینیم که دعداد پیش بینی اشتباه به عنوان اسب نسبت به قبل کمتر شده است. بیشتر پیش بینی اشتباه اسب مربوط به داده‌ها با دسته واقعی سگ و گربه بوده است. با بهبود یادگیری این دو دسته و متمایز کردن بهتر این دو دسته از سایر دسته‌ها، تعداد پیش بینی اشتباه اسب نسبت به بخش قبل کمتر شده است. بیشتر پیش بینی اشتباه قورباغه مربوط به داده‌ها با دسته واقعی سگ و گربه بوده است. با بهبود یادگیری این دو دسته و متمایز کردن بهتر این دو دسته از سایر دسته‌ها، تعداد پیش بینی اشتباه قورباغه نسبت به بخش قبل کمتر شده است. تعداد اشتباه پیش بینی قورباغه برای داده‌ه با لیبل واقعی گربه از ۱۹۲ به ۱۷۷ کاهش یافته است. این بهبود یادگیری باعث افزایش متریک‌های طبقه بندی شده است. این موضوع در Figure 35 قابل مشاهده است.

```
Test Loss 0.8424445390701294
Test Accuracy 0.753000020980835
              precision    recall  f1-score   support

           0       0.81      0.83      0.82      1000
           1       0.93      0.88      0.90      1000
           2       0.62      0.75      0.68      1000
           3       0.67      0.23      0.35      1000
           4       0.66      0.84      0.74      1000
           5       0.72      0.41      0.52      1000
           6       0.69      0.91      0.78      1000
           7       0.75      0.86      0.80      1000
           8       0.88      0.90      0.89      1000
           9       0.82      0.92      0.87      1000

    accuracy                           0.75     10000
   macro avg       0.75      0.75      0.73     10000
weighted avg       0.75      0.75      0.73     10000

f1 0.7344645631493587
precision 0.7536370713285152
recall 0.7529999999999999
accuracy 0.753
```

Figure 35 متریک‌های ارزیابی طبقه بند آموزش دیده با داده **augment** شده

سوال 4 – Transfer Learning

در اینجا با توجه به اولین رقم سمت راست شماره دانشجویی خود یعنی ۸ از مدل Xception استفاده می‌کنم.

۱

معماری مدل در Figure 36 آمده است. داده ها ابتدا از entry flow عبور داده می‌شوند و سپس از طریق جریان میانی (middle flow) که هشت بار تکرار می شود، عبور می‌کنند. سرانجام داده‌ها از exit flow عبور می‌کنند و خروجی به ازای هر داده تعیین می‌شود. توجه داشته باشید که همه لایه های Convolution و SeparableConvolution با نرمال سازی دسته ای (batch normalization) دنبال می شوند که در نمودار نیامده است. تمام لایه های SeparableConvolution از ضریب عمق 1 استفاده می‌کنند (بدون گسترش عمق).

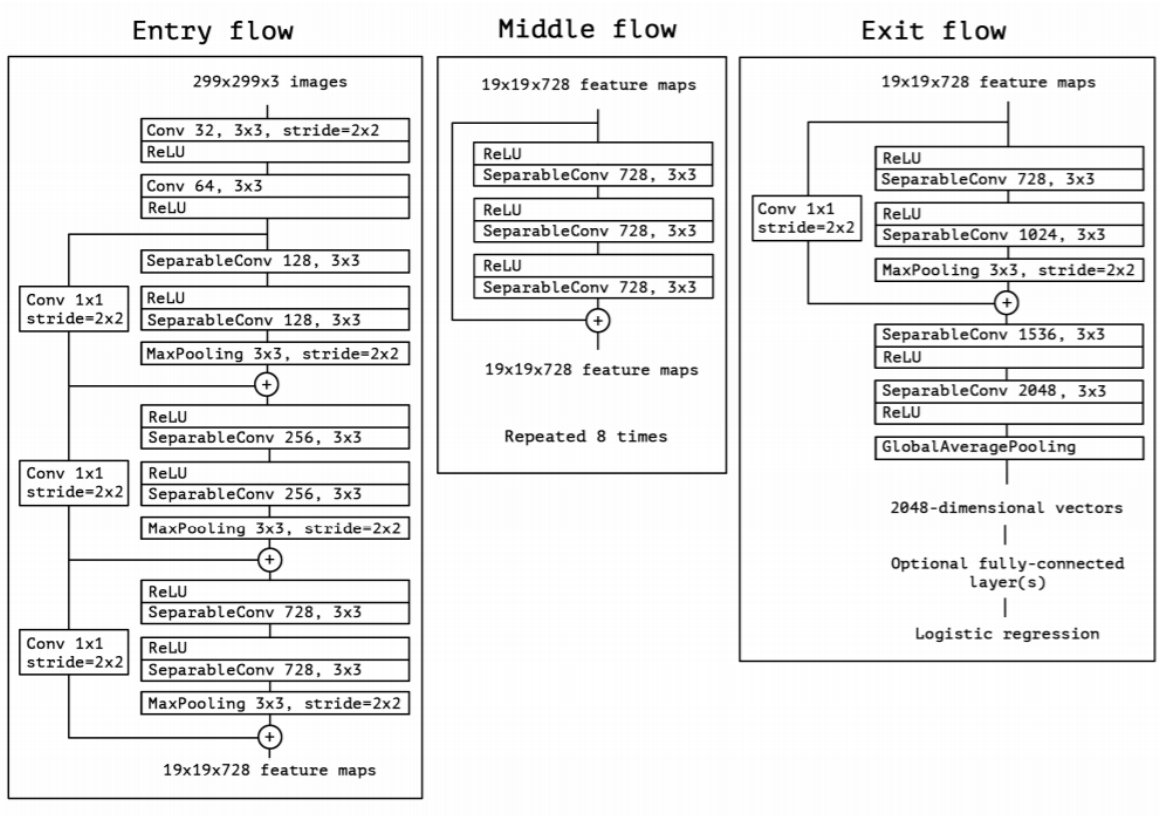

Figure 36 معماری مدل **Xception**

کل معماری از ۱۴ ماژول کانولوشنی تشکیل شده است. دو ماژول اول و آخر linear residual connection وجود ندارد. اتصال residual مستقیما ورودی را با خروجی کانولوشن‌های ماژول ترکیب می‌کند. در ماژول‌های کانولوشنی از ساختارهای کانولوشن عادی و Depthwise separable Convolution استفاده شده است. در Depthwise separable Convolution عملیات کانولوشن عادی به دو مرحله depthwise convolution و pointwise convolution شکسته شده است. در depthwise convolution بدون گسترش عمق ما از ضریب عمق یک استفاده می‌کنیم. در این جا ما کانولوشن بر روی ورودی انجام می‌دهیم ولی عمق را تغییر نمی‌دهیم. به همین دلیل برای هر کانال ورودی یک کرنل مجزا با اندازه مشخص شده در نظر می گیریم که عمق آن یک است. هر یک از این کرنل ها فقط روی کانال متناظر عملیات کانولوشن انجام می‌دهد. خروجی کانال‌ها را باهم در انتها stach می‌کنیم. بعد از این مرحله از خروجی آن برای pointwise convolution استفاده می‌کنیم. اگر ورودی ($3 \times 12 \times 12$) باشد، در صورت اعمال ۲۵۶ فیلتر با کرنل (5×5)، بعد از مرحله depthwise convolution ما خروجی به شکل ($3 \times 8 \times 8$) داریم. در pointwise convolution می‌خواهیم تعداد کانال خروجی را افزایش دهیم و ۲۵۶ تا برسانیم. در این مرحله از ۲۵۶ تا فیکتر استفاده می‌کنیم که هر یک کرنل ($3 \times 1 \times 1$) دارد که ۳ تعداد کانال ورودی این مرحله است. خروجی این مرحله خروجی Depthwise separable Convolution است. برای مثال ما حروجی به صورت ($256 \times 8 \times 8$) خواهد بود. در این مثال ما از stride برابر یک استفاده کردیم.

بعد از مازول‌های کانولوشنی ما ویژگی‌های مفید را از تصویر ورودی بیرون کشیده‌ایم و بعد از آن باید این ویژگی‌ها را به صورت بردار درآوریم و بر اساس آن به طبقه بندی بپردازیم. مدل مورد استفاده ما برای طبقه بندی طراحی شده است به همین دلیل در آخرین لایه از logistic regression استفاده شده است. ای لایه یک بردار به اندازه تعداد کلاس‌ها خروجی می‌دهد که هر المان احتمال تعلق ورودی به کلاس با index آن المان است. مدل به صورت پیش فرض برای دسته بندی مجموعه داده imagenet طراحی شده است و روی این مجموعه داده آموزش یافته است. تعداد کلاس‌ها برای این داده‌ها برابر 1000 است. پس به ازای هر عکس یک بردار ۱۰۰۰ تایی خروجی می‌دهیم.

سایز تصویر ورودی برای این شبکه به صورت (299×299) پیکسل است. همانطور که گفته شد، این مدل برای طبقه بندی عکس‌ها در کلاس‌های مجموعه داده imagenet طراحی و آموزش داده شده است. این مدل با مدل‌های دیگر مثل VGG-16 و ResNet-152 و Inception V3 در مقاله[1] مقایسه شده است که عملکرد بهتری از تمام آنها داشته است. همچنین در این مقاله می‌بینیم که تعداد پارامتر این مدل

[1] https://arxiv.org/pdf/1610.02357.pdf

تقریبا برابر Inception V3 است ولی توانسه عملکرد بهتری از آن داشته باشد. (تعداد پارامتر Xception کمی کمتر از Inception V3 است) در دیتاست imagenet مدل طراحی شده توانسته بهبود کمی نسبت به Inception V3 داشته باشد. در مجموعه داده JFT توانسته بهبود زیادی نسبت به Inception V3 در عملکرد ایجاد کند. نکته قابل توجه دیگر معرفی Depthwise separable Convolution است. این ساختار ویژگی‌های مشابهی با Inception ماژول دارد و نکته مهم دیگر این است که به سادگی کانولوشن عادی می‌توان از آن استفاده کرد. استفاده از Depthwise separable Convolution آسان است و ویژگی مشابه ماژول Inception دارد.

ورودی نیاز به پیش پردازش دارد. باید ابتدا عکس را به اندازه‌ی ورودی در بیاوریم تا بتوانیم مدل را روی آن اعمال کنیم. برای پیش پردازش از tf.keras.applications.xception.preprocess_input استفاده می‌کنیم که عدد هر پیکسل را بین 1- و 1 اسکیل می‌کند. دلیل این پیش پردازش این است که ورودی مدل CNN برای استخراج ویژگی و دسته بندی باید همیشه یکسان و ثابت باشد به همین دلیل ابتدا باید عکس را به اندازه ورودی مشخص شده برای مدل در بیاوریم. از اسکیل پیکسل‌ها استفاده می‌کنیم، چون می‌خواهیم تمام پیکسل‌ها رنج یکسان و برابر بازه 1- تا 1 داشته باشند. این بازه حتما باعث سرعت بخشیدن به یادگیری می شود و از بایاس نسبت به اندازه ورودی جلوگیری می‌کند. چون در ابتدا بازه پیکسل‌ها برابر صفر تا ۲۵۵ است که بازه بزرگی است. برای مثال اگر بازه ورودی را تغییر ندهیم، برخی از توابع فعالسازی برای اعداد بزرگ و بازه بزرگی از این بازه دچار اشباع می‌شوند که تغییر در این نواحی این توابع صفر است. این مورد باعث کند شدن یادگیری می‌شود. همچنین بازه ورودی خام بازه بزرگی از اعداد نامنفی است. این موضوع می‌تواند قرار گیری در ناحیه منفی اعداد را کم رنگ کند. در توابع فعالساز مانند relu که در ناحیه منفی خروجی با ناحیه مثبت متفاوت است، امکان قرار گیری در ناحیه منفی را کم می‌کنیم. این موضوع سرعت یادگیری را می‌تواند تحت تاثیر قرار دهد.

۲

انتقال یادگیری (Transfer learning) یک روش یادگیری ماشین است که در آن مدلی که برای یک کار ایجاد شده است، به عنوان نقطه شروع یک مدل برای کار دوم مورد استفاده مجدد قرار می گیرد. دلیل استفاده از این روش این است که آموزش یک کار وقت گیر است و نیاز به یک مجموعه داده بزرگ آموزش دارد. زمانی از این روش استفاده می‌کنیم که مجموعه آموزش بزرگ و مرغوبی نداریم و زمان یادگیری کمی در اختیار داریم یا منابع محاسباتی پایینی داریم. در این شرایط از یک مدل از قبل آموزش دیده بر

روی یک مجموعه بزرگ و مرغوب برای یک تسک که تسک آن با تسک موردنظر ما تجانس و نزدیکی دارد، استفاده می‌کنیم. ما از بخش استخراج ویژگی مدل از پیش اموزش یافته به عنوان استخراج ویژگی در شروع مدل خود استفاده می‌کنیم. به کمک دانش بالای موجود در مدل از پیش آموزش دیده بهترین و مهمترین ویژگی‌ها را استخراج می‌کنیم و بر اساس آنها به حل تسک خود می‌پردازیم. با این کار در صورت نداشتن مجموعه آموزش بزرگ و غنی می‌توانیم از مهارت بالای مدل از پیش آموزش دیده در استخراج ویژگی استفاده کنیم که کیفیت استخراج ویژگی ما را بسیار بالا می‌آورد. با مجموعه کوچک در اختیار ما نمی‌توان استخراجگر با کیفیت ویژگی آموزش داد. همچنین آموزش دادن استخراجگر ویژگی زمانبر است و به منابع محاسباتی بزرگی نیاز دارد که با transfer learning، نیاز ما به صرف زمان برای آموزش بسیار کاهش می‌یابد. این موضوع باعث می‌شود کل فرآیند یادگیری مدل ما برای تسک مورد نظرمان سریعتر و با منابع محاسباتی کمتری انجام شود.

۳

برای این کار از keras.applications استفاده می‌کنیم و معماری مدل Xception را لود می‌کنیم. در اینجا از وزن‌های این مدل که بر روی مجموعه داد imagenet آموزش داده شده است، استفاده می‌کنیم. کاری که ما می‌خواهیم انجام دهیم، طبقه بندی و بدست آوردن احتمال تعلق عکس به هر یک از کلاس‌های مجموعه داده imagenet است. پس کافی است معماری مدل Xception برای طدقه بندی را لود کنیم که در لایه آخر از logistic regression استفاده می‌کند. مانند معماری قسمت یک این سوال. خروجی این ساختار با وزن‌های حاصل از آموزش روی imagenet به ازای هر عکس برداری از احتمالات است. بردار ۱۰۰۰ تایی است به تعداد لیبل‌های مجموعه داده است. هر درایه احتمال تعلق عکس به کلاس به شماره index درایه است. در Figure 37 نحوه لود مدل و وزن‌های مدل و پیاده سازی مدل موردنظر آمده است.

```
# the required input dimensions for the ImageNet Xception pre-trained network
inputShape = (299, 299)
preprocess = preprocess_input
```

```
model = Xception(weights="imagenet")
```

```
image = load_img('IMG_20210507_121338_3[1].jpg', target_size=inputShape)
image = img_to_array(image)

# add batch size
image = np.expand_dims(image, axis=0)
# pre-process the image using the appropriate function based on the
# model that has been loaded (i.e., mean subtraction, scaling, etc.)
image = preprocess(image)
```

```
preds = model.predict(image)
```

Figure 37 نحوه لود مدل و وزن‌های موردنظر

در Figure 37 علاوه بر نحوه لود مدل از پیش آموزش دیده و پیاده سازی مدل موردنظر، نحوه تغییر اندازه عکس ورودی و لود آن و نحوه پیش پردازش بر روی عکس ورودی آمده است.

۴

تعداد اشیا قابل تشخیص مدل ۱۰۰۰ تا برابر کلاس‌های مجموعه داده imagenet است که در زیر لیست نام تعدادی از این اشیا آمده است.

```
n02119789 1 kit_fox
n02100735 2 English_setter
n02110185 3 Siberian_husky
n02096294 4 Australian_terrier
n02102040 5 English_springer
n02066245 6 grey_whale
n02509815 7 lesser_panda
n02124075 8 Egyptian_cat
n02417914 9 ibex
n02123394 10 Persian_cat
n02125311 11 cougar
n02423022 12 gazelle
n02346627 13 porcupine
n02077923 14 sea_lion
n02110063 15 malamute
n02447366 16 badger
n02109047 17 Great_Dane
n02089867 18 Walker_hound
n02102177 19 Welsh_springer_spaniel
n02091134 20 whippet
n02092002 21 Scottish_deerhound
n02071294 22 killer_whale
n02442845 23 mink
n02504458 24 African_elephant
n02092339 25 Weimaraner
n02098105 26 soft-coated_wheaten_terrier
n02096437 27 Dandie_Dinmont
n02114712 28 red_wolf
```

در https://gist.github.com/aaronpolhamus/964a4411c0906315deb9f4a3723aac57 لیست کامل نام اشیا آمده است.

۵

عکس گرفته شده و مورد استفاده در Figure 38 آمده است.

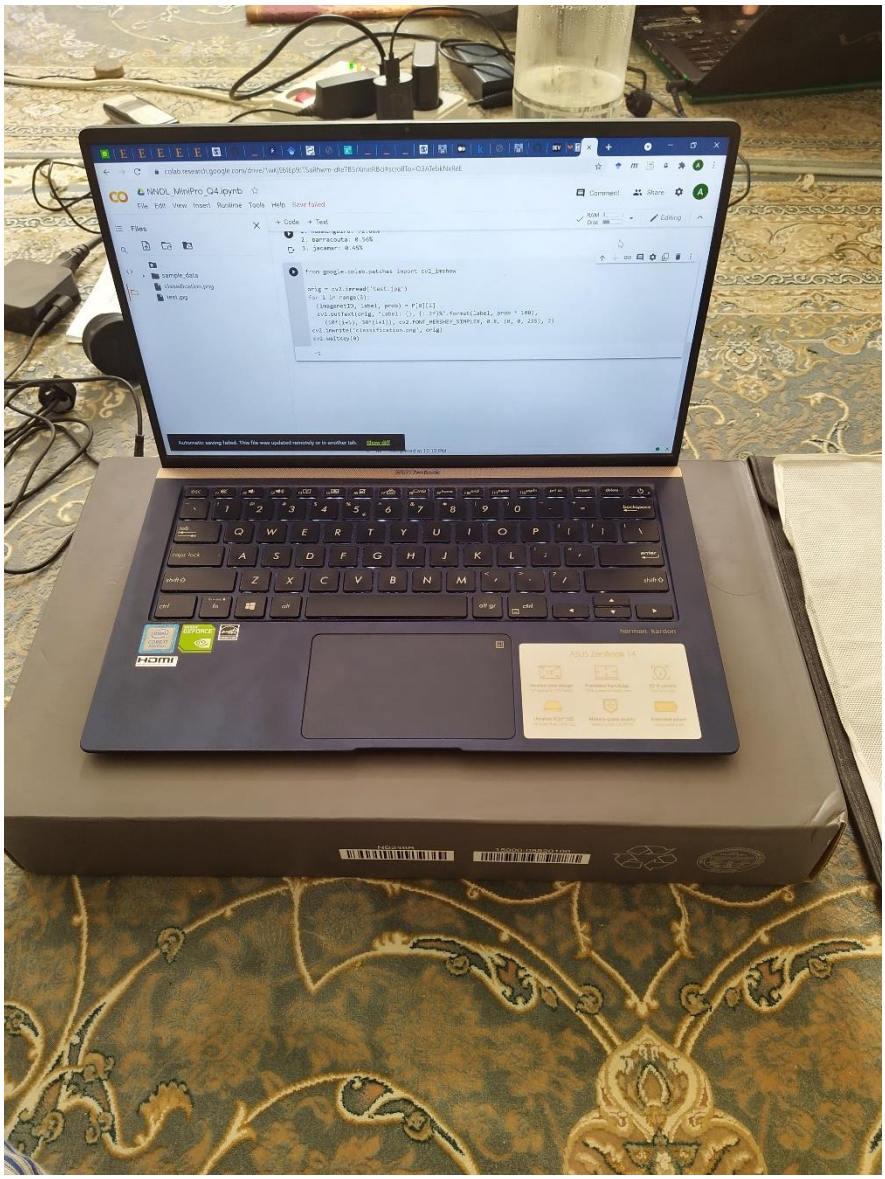

Figure 38 عکس گرفته شده

مانند Figure 37 عکس گرفته شده را لود می‌کنیم و پیش پردازش لازم را روی آن انجام می‌دهیم. با استفاده از مدل بخش ۳ که در Figure 37 هم پیاده سازی آن آمده به طبقه بندی عکس گرفته شده می‌پردازیم. سه کلاس با بیشترین احتمال‌ها در Figure 39 آمده است.

```
1. monitor: 14.34%
2. desktop_computer: 10.41%
3. modem: 10.28%
```

Figure 39 سه شی با بیشترین احتمال در عکس گرفته شده

سه شی با بیشترین احتمال مربوط به عکس بر روی عکس نوشته شده و در Figure 40 آمده است.

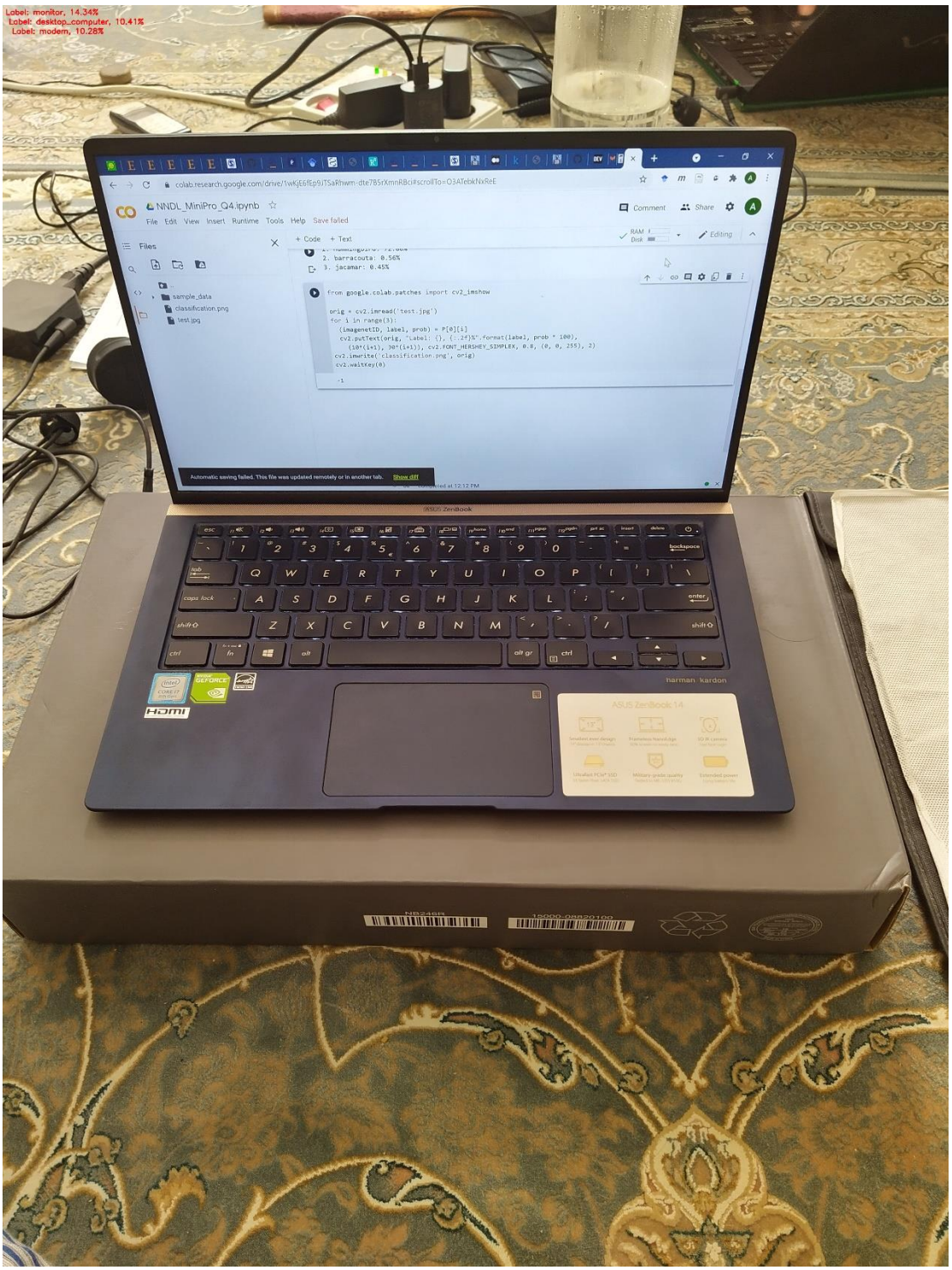

Figure 40 سه شی با بیشترین احتمال مربوط به عکس بر روی عکس نوشته شده