

محمدرضا غفرانی  
۴۰۰۱۳۱۰۷۶  
۲۳ اردیبهشت ۱۴۰۱

## شبکه‌های عصبی

### تمرین دوم

### سوال یک

#### شبکه LeNet-5

این شبکه عصبی از ۵ لایه تشکیل شده است. سه لایه کانولوشنی بوده و باقی لایه‌ها از نوع fully connected هستند. در ادامه ساختار هر یک از این لایه‌های این شبکه عصبی توضیح داده می‌شود.

۱. لایه کانولوشنی: این لایه کانولوشنی دارای ۶ فیلتر با ابعاد  $5 \times 5$  بوده و با اندازه قدم یک بر روی تصویر ورودی اعمال می‌شود. تابع فعال‌ساز این لایه  $\tanh$  است. با توجه به این که این لایه ۶ فیلتر را بر روی تصویر ورودی اعمال می‌کند، بنابراین این لایه تصویر تک کاناله ورودی را به ۶ کانال مجزا نگاشت می‌کند.

(A) Pooling: در این قسمت یک فیلتر  $2 \times 2$  روی خروجی لایه اعمال می‌شود. فیلتر اعمال شده از نوع average-pooling بوده و تعداد کانال‌های ورودی را تغییر نمی‌دهد. اما ابعاد هر ورودی دریافتی را نصف می‌کند.

۲. لایه کانولوشنی: در این لایه ۱۶ فیلتر با ابعاد  $5 \times 5$  و اندازه قدم یک به خروجی‌های لایه قبلی اعمال می‌شود. تابع فعال این لایه نیز  $\tanh$  است.

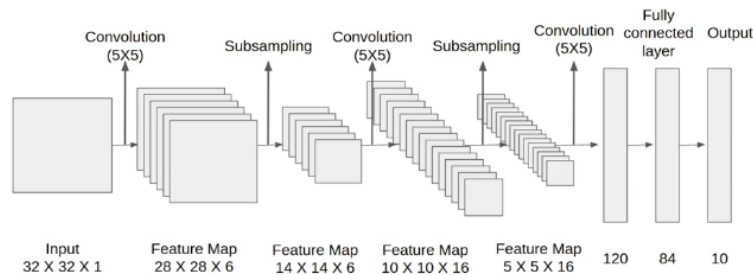
(A) Pooling: در این قسمت نیز یک فیلتر  $2 \times 2$  بر روی خروجی لایه قبلی اعمال شده و ابعاد خروجی‌های لایه قبلی را نصف می‌کند.

۳. لایه کانولوشنی: این لایه با اعمال ۱۲۰ فیلتر با ابعاد  $5 \times 5$  ابعاد خروجی تمام کانال‌ها را به یک بعد می‌رساند. تابع فعال‌ساز این لایه نیز مشابه لایه‌های قبلی  $\tanh$  است.

۴. لایه fully-connected: این لایه خروجی لایه قبلی را گرفته و عمل دسته‌بندی را آغاز می‌کند. همان‌طور که از نام این لایه مشخص است، هر گره این لایه به تمامی گره‌های لایه بعدی متصل است.

۵. لایه fully-connected: عملکرد این لایه نیز مشابه لایه fully-connected قبلی است. معمولاً خروجی این لایه را به یک لایه softmax می‌دهند تا خروجی نهایی مدل مشخص شود.

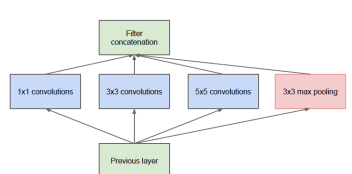
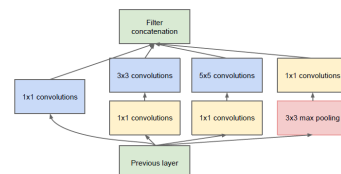
در شکل ۱ ساختار این شبکه در هنگام اعمال بر روی یک تصویر سیاه و سفید با ابعاد  $32 \times 32$  مشاهده می‌شود.



شکل ۱: نمونه‌ای از شبکه عصبی LeNet

## شبکه Inception

با افزایش تعداد لایه‌های کانولوشنی احتمال بیش‌برازش مدل روی داده‌های آموزشی بیشتر می‌شود. برای رفع این مشکل در نسخه اول شبکه Inception چند فیلتر کانولوشنی در یک سطح از شبکه استفاده شد. این ایده باعث شد مدل نسبت به مدل‌های پیشین عملکرد بهتری داشته باشد. از طرف دیگر در این مدل به جهت کاهش ابعاد و همچنین افزایش سرعت به هر لایه یک فیلتر کانولوشنی  $1 \times 1$  اضافه شده است. در شکل ۲ نوآوری‌های انجام شده این مدل نشان داده شده است.

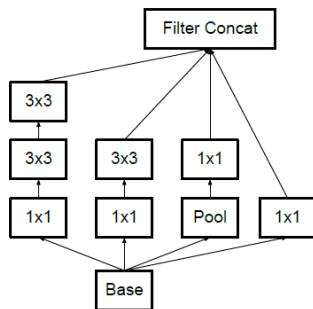
(ب) شیوه قرارگیری فیلترها قبل از اضافه کردن فیلتر  $1 \times 1$ (آ) شیوه قرارگیری فیلترها با اضافه کردن فیلتر  $1 \times 1$ 

شکل ۲

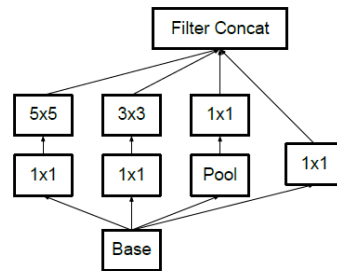
در مدل Inception v3 ایده‌های بالا به همراه دیگر ایده‌هایی که در ادامه توضیح داده می‌شود، برای بهبود عملکرد مدل به کار گرفته شده است.

- **تبدیل فیلترهای با ابعاد بالا به چند فیلتر با ابعاد کوچک‌تر** کاهش بعد یکی از ویژگی‌های مثبت مدل Inception v1 بود. نسخه سوم این مدل هم این ویژگی خوب را به ارث برد و حتی آن را بهبود داد. در تصویر ۳ ایده نسخه سوم مدل مشاهده می‌شود. در شبکه Inception v3 مطابق شکل فیلترهای با ابعاد  $5 \times 5$  به دو فیلتر با ابعاد  $3 \times 3$  تبدیل می‌شود. این تبدیل باعث کاهش تعداد پارامترهای مدل و افزایش سرعت مدل می‌شود. در شکل ۴ نمونه‌ای از عملکرد دو فیلتر  $3 \times 3$  که به صورت متوالی قرار داده شده است، آورده شده است.

- **تبدیل فیلترها به فیلترهای نامتقارن** با توجه به مزیت‌هایی که کاهش ابعاد فیلترها در کنار افزایش تعداد آن‌ها برای ما به همراه داشته است به نظر می‌رسد که ایده خوبی

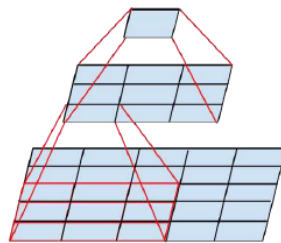


Inception v3 مدل معماری فیلترهای مدل (ب)



Inception v1 مدل معماری فیلترهای مدل (آ)

شکل ۳: شیوه کاهش ابعاد فیلترهای کانولوشنی در مدل inceptionv3



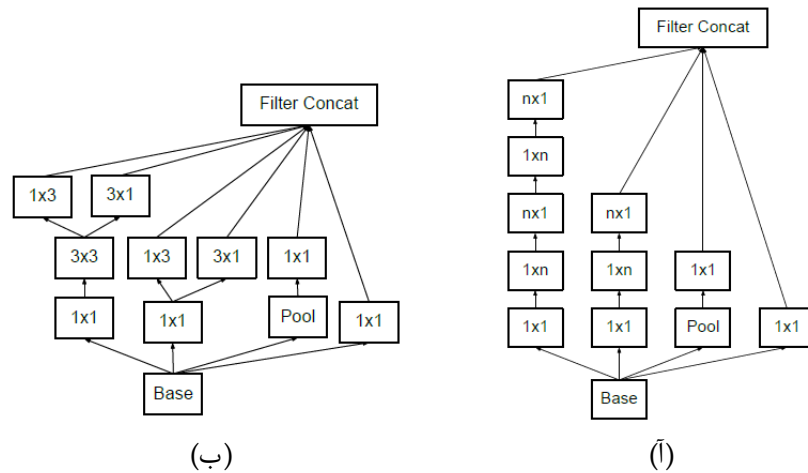
شکل ۴

باشد که فیلترهای  $3 \times 3$  نیز به فیلترهای  $2 \times 2$  تبدیل شود. اما این تبدیل کاهش بعد چندانی نداشته و مزیتی ایجاد نمی‌کند. روش دیگر آن است که ابعاد فیلترها را نامتقارن در نظر بگیریم. این ایده در شکل ۵آ دیده می‌شود. با این کار پارامترهای مدل به شکل چشم‌گیری کاهش می‌یابد. در شکل ۵ب نمونه‌ای از اعمال این ایده بر روی شکل ۳ دیده می‌شود.

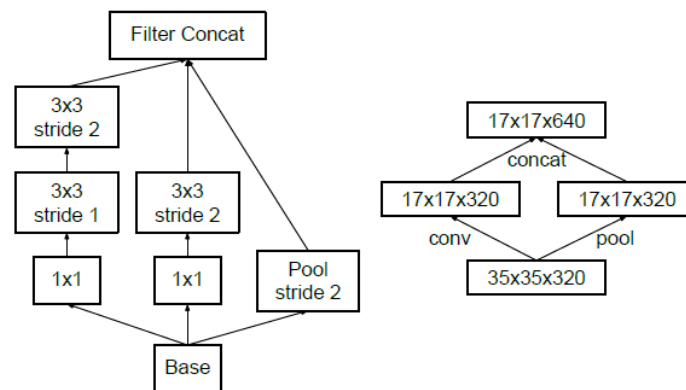
- استفاده از دسته‌بندهای کمکی هدف از استفاده از دسته‌بندهای کمکی افزایش همگرایی مدل‌های عصبی عمیق است. این دسته‌بندها بیشتر به منظور جلوگیری از مشکل vanishing gradient استفاده می‌شوند. در شبکه Inception این دسته‌بند در گام‌های اولیه یادگیری بهبودی ایجاد نمی‌کند. اما با نزدیک شدن به گام‌های نهایی، عملکرد شبکه‌ای که از دسته‌بند کمکی استفاده کرده است بسیار بهتر از شبکه بدون دسته‌بند است. به عبارتی دیگر این دسته‌بندی کمکی نظیر یک هموارساز عمل کرده و عملکرد مدل را بهبود می‌دهد.

- کاهش بهینه ابعاد (Efficient Grid Size Reduction) در شبکه Inception v3 بر خلاف روال معمول که از دو تابع max pooling و average pooling برای کاهش ابعاد مدل استفاده می‌شود. از دو ساختار موجود در شکل ۶ برای کاهش ابعاد استفاده می‌شود.

شکل کلی مدل Inception v3 در شکل ۷ آورده شده است. همان‌طور که مشاهده می‌شود معماری مدل LeNet نسبت به مدل Inception بسیار کوچک‌تر است. مدل Inception

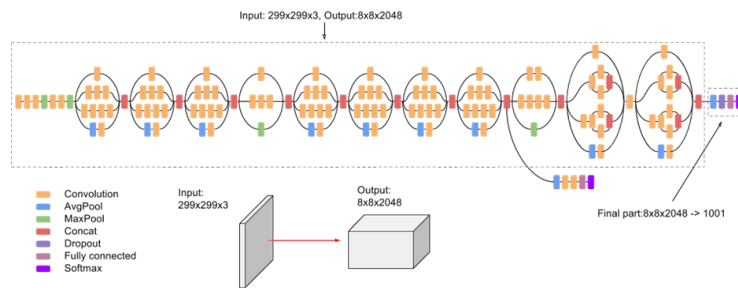


شکل ۵: شیوه فاکتور کردن نامتقارن و مثالی از آن



شکل ۶: شبکه‌هایی که برای کاهش بعد در مدل Inception v3 استفاده می‌شود.

تعداد لایه‌های بیشتری داشته و در نتیجه مدل عمیق‌تری است. عمیق‌تر بودن باعث شده است که به راحتی بتواند الگوی داده‌ها را کشف کرده و به نتایج بسیار خوبی برسد. البته عمیق بودن هم مزیت و هم عیب مدل محسوب می‌شود. به دلیل عمیق بودن ممکن است روی داده‌های آموزشی بیش‌برازش شود، در نتیجه راهکارهای فراوانی را هم برای جلوگیری از وقوع مشکلاتی نظیر *vanishing problem* و بیش‌برازش پیاده‌سازی کرده است. به خاطر همین عمیق بودن به زمان بیشتری برای آموزش مدل نیاز داشته و در هنگام تست نیز مدت زمان بیشتری برای پاسخ دادن به داده ورودی نیاز دارد. اما در حالت کلی از آن جا که عملکرد بهتری نسبت به مدل LeNet دارد در نتیجه نسبت به آن بهتر است.



شکل ۷: معماری کلی شبکه Inception

## سوال دو

شبکه LeNet انتظار دارد ورودی یک تصویر تک کاناله با ابعاد  $32 \times 32$  باشد. اما تصاویر مجموعه داده ما  $500 \times 500$  بوده و سه کاناله هستند. به همین دلیل با انجام پیش‌پردازش روی داده‌ها آن‌ها را به تصاویر سیاه‌وسفید تبدیل کرده و اندازه تصاویر را نیز به  $32 \times 32$  تبدیل می‌کنیم.

شبکه LeNet از آن جا که برای شناسایی رقم‌های دست‌نویس ارائه شده است، دارای ۱۰ نود در لایه خروجی است. اما در این جا می‌خواهیم داده‌ها را به سه کلاس تقسیم کنیم؛ بنابراین لایه خروجی شبکه را تغییر داده و تعداد نودهای خروجی آن را به ۳ کاهش می‌دهیم. به جز این موارد پیش‌پردازش یا تغییر دیگری در ساختار مدل انجام نشده است. عملکرد مدل در این حالت در جدول ۱ آورده شده است.

Table 1: جدول عملکرد شبکه LeNet در حالت پایه

| train |      | validation |     | test |      |
|-------|------|------------|-----|------|------|
| loss  | acc  | loss       | acc | loss | acc  |
| 0.84  | 0.62 | 0.95       | 0.6 | 0.96 | 0.55 |

## قسمت الف

در ادامه تاثیر افزودن هر یک از لایه‌های منظم‌ساز بررسی می‌شود.

- **منظم‌ساز L1:** در هنگام منظم‌سازی با استفاده از پارامتر  $L1$  هنگامی که پارامتر منظم‌سازی عدد کوچکی نظیر  $10^{-6}$  است عملکرد مدل اندکی بهتر می‌شود. اما اگر مقادیر بزرگی نظیر 0.1 را برای تنظیم مدل در نظر بگیریم، عملکرد مدل هم در داده‌های ارزیابی و هم داده‌های آموزش افت می‌کند. (جدول ۲)

- **منظم‌ساز L2:** نتایج حاصل شده در هنگام استفاده از این منظم‌ساز بسیار شبیه نتایج حاصل شده در حالت استفاده از منظم‌ساز  $L1$  بوده و تفاوت فاحشی بین آن‌ها نیست. این شباهت قابل توجه است. فرمول هموارسازی  $L2$  بسیار شبیه  $L1$  است بنابراین منطقیست که عملکرد این دو تابع هموارساز شبیه یکدیگر باشند. (جدول ۳)

Table 2: عملکرد شبکه LeNet با منظم‌ساز L1

| L1 value  | train |      | validation |      | test |      |
|-----------|-------|------|------------|------|------|------|
|           | loss  | acc  | loss       | acc  | loss | acc  |
| $10^{-6}$ | 0.77  | 0.66 | 0.91       | 0.59 | 0.88 | 0.59 |
| 0.1       | 1.32  | 0.34 | 1.32       | 0.34 | 1.32 | 0.34 |

Table 3: عملکرد شبکه LeNet با منظم‌ساز L2

| L2 value  | train |      | validation |      | test |      |
|-----------|-------|------|------------|------|------|------|
|           | loss  | acc  | loss       | acc  | loss | acc  |
| $10^{-6}$ | 0.79  | 0.59 | 0.95       | 0.59 | 0.92 | 0.55 |
| 0.1       | 1.19  | 0.63 | 1.28       | 0.56 | 1.24 | 0.57 |

- **منظم‌ساز Dropout:** تاثیر وجود این لایه همانند تاثیر وجود منظم‌سازهای L1 و L2 است. بدین معنی که هنگامی که مقدار نرخ dropout مدل کم است (مثلا در حد  $10^{-2}$  یا  $10^{-4}$ ) عملکرد مدل در داده‌های ارزیابی اندکی بهتر از حالت پایه می‌شود. با زیاد کردن نرخ dropout نیز (در حد 0.7) مدل نمی‌تواند روی داده‌های ارزیابی و آموزشی به نتایج خوبی برسد.

Table 4: عملکرد شبکه LeNet با منظم‌ساز Dropout

| dropout value | train |      | validation |      | test |      |
|---------------|-------|------|------------|------|------|------|
|               | loss  | acc  | loss       | acc  | loss | acc  |
| $10^{-4}$     | 0.78  | 0.66 | 0.94       | 0.6  | 0.95 | 0.5  |
| 0.7           | 0.97  | 0.54 | 0.98       | 0.54 | 0.99 | 0.48 |

با توجه به یکسان بودن نتایج و عملکرد تقریباً یکسان مدل با اضافه کردن هموارسازی‌های مختلف نتایج حاصل شده را یکجا توجیه می‌کنیم. هدف ما از اضافه کردن هموارسازی‌های مختلف به مدل جلوگیری از بیش‌برازش مدل بر روی داده آموزشی و رسیدن به نتیجه بهتر در داده‌های ارزیابی بود. در تمامی تکنیک‌های هموارسازی استفاده شده، هنگامی که پارامتر هموارسازی عدد کوچکی بود همان‌طور که دیدیم عملکرد مدل در داده‌ها اندکی بهتر می‌شود. این مورد نشان می‌دهد که به ازای اضافه کردن این هموارسازی‌ها مدل دست از بیش‌برازش روی داده‌های آموزشی برداشته که در نتیجه به نتایج بهتر روی داده‌های ارزیابی رسیده است. اما هنگامی که پارامتر هموارسازی را بیشتر می‌کردیم، عملکرد مدل هم در داده‌های ارزیابی و هم در داده‌های آموزشی بدتر می‌شد که نشان می‌داد مدل به دلیل وجود لایه هموارسازی نمی‌تواند دانش چندانی در رابطه با داده‌ها بیاموزد.

## قسمت ب

با توجه به جدول ۵ به صورت کلی می‌توان گفت که افزایش تعداد فیلترهای کانولوشنی باعث بهتر شدن عملکرد مدل می‌شود. برای مثال در حالتی که تعداد فیلترها برابر (100, 100, 20) است، مدل توانسته است صحت خوبی را در داده‌های آزمون داده با کم کردن نرخ loss ارائه

دهد. البته این افزایش عملکرد بدون هزینه نیز نبوده و مدل به مدت زمان بیشتری نیاز دارد تا هر گام یادگیری را طی کند. همچنین با افزایش لایه‌های کانولوشنی به دلیل آزادی عملکرد بیشتر، استعداد مدل برای بیش‌برازش روی داده‌های آموزشی بیشتر می‌شود. بنابراین برای آن که هر مدل بهترین عملکرد خود را داشته باشد، از یک EarlyCallback با حداکثر تعداد صبر ۳ واحد استفاده کرده‌ایم. اگر نرخ loss در داده اعتبارسنجی در طی سه گام متوالی کاهش نیابد، حداکثر تعداد گام برای آموزش مدل برابر ۲۰ است.

Table 5: LeNet جدول بررسی تاثیر تعداد فیلتر در عملکرد شبکه

| # of filter  | train |      | validation |      | test |      |
|--------------|-------|------|------------|------|------|------|
|              | loss  | acc  | loss       | acc  | loss | acc  |
| (1,1,1)      | 1.1   | 0.33 | 1.1        | 0.33 | 1.1  | 0.34 |
| (3,3,2)      | 1     | 0.49 | 1.02       | 0.47 | 1.01 | 0.48 |
| (20,10,5)    | 0.86  | 0.6  | 0.97       | 0.49 | 0.92 | 0.54 |
| (50,40,30)   | 0.53  | 0.8  | 0.88       | 0.62 | 0.84 | 0.61 |
| (100,100,20) | 0.4   | 0.88 | 0.88       | 0.61 | 0.82 | 0.62 |

## قسمت ج

با توجه به نتایج که در جدول ۶ آورده شده است، با بیشترکردن ابعاد فیلترهای کانولوشن صحت عملکرد مدل کاهش می‌یابد. این اتفاق را می‌توان این گونه توجیه کرد که هنگامی که ابعاد فیلتر بزرگ است نمی‌تواند ویژگی‌های کلی مدل را از تصویر استخراج کند، چرا که اندازه تصویر  $32 \times 32$  بوده و در نتیجه با اعمال یک فیلتر  $16 \times 16$  در لایه اول ویژگی‌های حاصل شده بسیار کلی می‌شود، که در لایه‌های بعدی با اعمال فیلترهای کوچک‌تر نیز ویژگی‌ها استخراج نمی‌شود. نکته بعدی که وجود دارد محل اعمال این فیلتر است برای مثال هنگامی که در لایه اول یک فیلتر  $5 \times 5$  و در لایه بعدی یک فیلتر  $3 \times 3$  اعمال می‌کنیم نتایج بسیار بهتری نسبت به حالتی که فیلتر  $3 \times 3$  در لایه اول است می‌گیریم.

از دیگر پیامدهای افزایش ابعاد فیلترها می‌توان به افزایش تعداد پارامترهای مدل اشاره کرد. با افزایش تعداد پارامترها نیاز به داده بیشتری دارد تا بتواند الگوی کلی داده‌ها را به خوبی پیدا کرده و مقدار بهینه هر پارامتر را به دست آورد.

در این حالت برای آن که تمامی مدل‌ها به بهترین عملکرد خود دست یابند، یک EarlyCallback با مشخصات توضیح داده شده در قسمت قبل آورده شده است.

Table 6: LeNet تاثیر اندازه فیلتر در عملکرد شبکه

| size of filter | train |      | validation |      | test |      |
|----------------|-------|------|------------|------|------|------|
|                | loss  | acc  | loss       | acc  | loss | acc  |
| (16, 7, 1)     | 1.05  | 0.43 | 1.09       | 0.38 | 1.09 | 0.43 |
| (14, 5, 2)     | 0.97  | 0.43 | 1.04       | 0.47 | 1.08 | 0.43 |
| (8, 5, 2)      | 0.8   | 0.66 | 0.99       | 0.59 | 0.93 | 0.56 |
| (5, 3, 2)      | 0.81  | 0.64 | 0.98       | 0.53 | 0.92 | 0.59 |
| (3, 5, 2)      | 0.8   | 0.64 | 0.98       | 0.48 | 1.07 | 0.45 |

## سوال سه

در فرآیند انتقال یادگیری ابتدا مدلی بر پایه مجموعه داده بزرگ ساخته می‌شود. در این مرحله مدل دانش خوبی از کلیات مسئله دارد اما نمی‌تواند در یک وظیفه خاص به خوبی عمل کند. تکنیک‌های انتقال یادگیری به منظور بهتر کردن عملکرد مدل در حوزه‌های خاص و یا انتقال آموخته‌های مدل به حوزه‌های مشابه استفاده می‌شود. از تکنیک‌های این دانش معمولاً در دو سناریو استفاده می‌شود.

- هنگامی که تعداد داده در دسترس در حوزه مقصد محدود است. در این حالت با تنظیم مدل آموزش دیده در داده‌های حجیم با داده‌های حوزه مقصد می‌توان به نتایج قابل قبولی رسید.
- سناریو دیگر هنگامی است که می‌خواهیم برای حوزه مدنظر مدلی را به سرعت ارائه دهیم. در این حالت به جای آن که مدل را از پایه روی تمامی داده‌ها آموزش دهیم، تلاش می‌کنیم با تغییرات جزئی در مدل حجیم، به نتایج خوبی در حوزه مقصد برسیم.

با ارائه شبکه‌های عصبی کارهای فراوانی در زمینه انتقال یادگیری صورت گرفت. معمولاً هنگامی که می‌خواهند یک مدل شبکه عصبی را روی داده‌های دیگری تنظیم کنند، شبکه را اصطلاحاً freeze کرده و لایه‌های آخری که برای تولید خروجی نهایی هستند را حذف می‌کنند. وزن‌های آموخته شده برای این لایه‌ها در طی فرآیند تنظیم<sup>۱</sup> تغییر نکرده و ثابت می‌ماند. حال لایه‌های نهایی جدید را اضافه کرده و آموزش را با یادگیری مقادیر وزن این لایه‌ها روی داده‌های حوزه مقصد، پیش می‌برند. در نهایت با در نظر گرفتن مقادیر بسیار کوچکی برای نرخ یادگیری و آموزش مجدد کل مدل (با unfreeze کردن لایه‌ها) آموزش مدل را نهایی می‌کند.

## سوال چهار

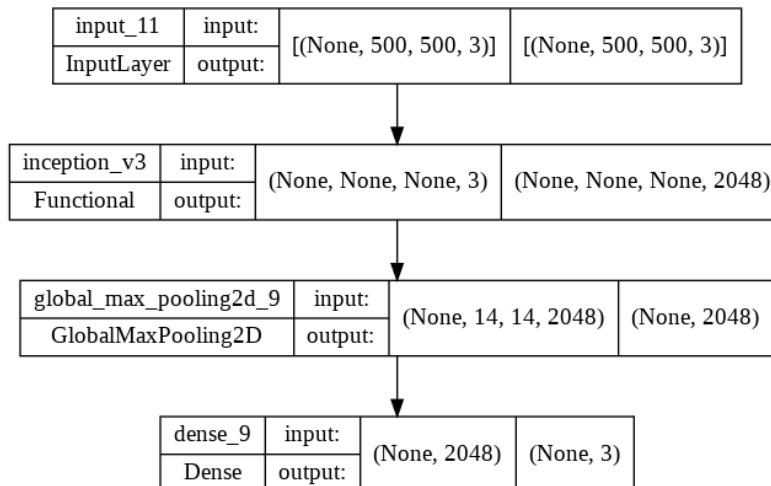
### قسمت الف

شکل کلی شبکه آموزش دیده برای آموزش مجدد مدل Inception در شکل ۸ آورده شده است. همان‌طور که مشاهده می‌شود برای آموزش مجدد مدل یک لایه ورودی به همراه

<sup>۱</sup>Fine tune



یک لایه MaxPooling و یک لایه خروجی از نوع Softmax به مدل اضافه کرده‌ایم. برای مشخص کردن تعداد بهینه لایه‌های فریز شده از آزمون و خطا استفاده می‌کنیم. بدین شکل که هر بار درصد مشخصی از لایه‌های ابتدایی مدل inception فریز شده و باقی لایه‌ها آموزش می‌بینند.



شکل ۸: شبکه طراحی شده برای آموزش مجدد مدل Inception

سعی و خطاهای انجام شده برای آموزش مدل در جدول ۷ آورده شده است. همچنین لازم به ذکر است که به دلیل زمان‌بر بودن آموزش مدل تعداد گام آموزش را برابر ۷ در نظر گرفته‌ایم. با توجه به نتایج، مدل به ازای این تعداد گام فرصت کافی برای آموزش و یافتن مقدار بهینه برای پارامترهای خود را داشته است. در توضیح نتایج باید گفت که مدل هنگامی که بیشتری آزادی عمل را داشته است توانسته به بهترین نتیجه برسد، اما در ادامه با محدود شدن آزادی عمل مدل، توانایی شبکه ضعیف و ضعیف‌تر شده است تا جایی که عملکرد مدل حتی از مدل LeNet هم ضعیف‌تر شده است. بنا بر این توضیح فاکتورهای زیر را می‌توان در هنگام افزایش تعداد لایه‌های فریز شده مدل دخیل دانست.

- **تعداد گام و محاسبات.** مشخص است که هر چه تعداد لایه‌های فریز شده کمتر باشد، تعداد پارامترهای مدل بیشتر بوده و در نتیجه مدل به مدت زمان و تعداد گام بیشتری برای یافتن بهترین مقدار این پارامترها نیاز دارد.
- **حوزه آموزش مدل از پیش آموزش دیده.** اگر مدل از پیش آموزش دیده پیش‌تر در حوزه دیگری آموزش دیده باشد که با حوزه مقصد تفاوت داشته باشد برای آن که بتواند روی حوزه مقصد به خوبی عمل کند نیاز به آزادی عمل بیشتر و تعداد لایه‌های فریز شده کمتری دارد تا بتواند پارامترها را به خوبی در حوزه جدید آموزش دهد. اما اگر حوزه عملکرد مدل چندان متفاوت نباشد، می‌توان با فریز کردن تعداد لایه‌های کمتر به دقت‌های خوبی رسید.

نکته جالبی که در خصوص Fine-tune این شبکه وجود دارد این است که اگر ابتدا بخواهیم تمامی لایه‌های شبکه Inception را فریز کرده و پس از آموزش لایه‌های جدید

اضافه شده، بخش‌هایی از مدل ازپیش‌آموزش دیده را unfreeze کرده و Fine-tune کنیم. مدل نهایی عملکرد ضعیف‌تری خواهد داشت. به همین دلیل ما در هنگام آموزش لایه‌های جدید، بخش‌هایی از مدل Inception را نیز unfreeze کرده و آموزش می‌دهیم. نتایج موجود در جدول ۷ با همین روش حاصل شده‌اند.

Table 7: درصد لایه‌های فریز شده مدل ازپیش‌آموزش دیده

| % of frozen layer | train |       | validation |       | test |       |
|-------------------|-------|-------|------------|-------|------|-------|
|                   | acc   | loss  | acc        | loss  | acc  | loss  |
| 0                 | 1     | 0     | 0.95       | 0.14  | 0.9  | 0.32  |
| 10                | 1     | 0     | 0.8        | 0.71  | 0.81 | 0.56  |
| 20                | 1     | 0.02  | 0.66       | 0.9   | 0.67 | 0.86  |
| 30                | 1     | 0.01  | 0.62       | 1.28  | 0.7  | 0.91  |
| 40                | 1     | 0.01  | 0.62       | 1.17  | 0.65 | 1.2   |
| 50                | 0.99  | 0.03  | 0.65       | 0.99  | 0.73 | 0.83  |
| 60                | 0.99  | 0.03  | 0.55       | 1.23  | 0.62 | 0.99  |
| 70                | 0.99  | 0.01  | 0.65       | 1.11  | 0.7  | 0.99  |
| 80                | 1     | 0.01  | 0.57       | 1.12  | 0.62 | 0.93  |
| 90                | 1     | 0.02  | 0.68       | 0.91  | 0.71 | 0.72  |
| 100               | 0.41  | 11.74 | 0.38       | 12.44 | 0.34 | 13.08 |

## قسمت ب

با توجه به نتایج مدل Inception با تنظیم درست تعداد لایه‌های فریز شده بهتر از مدل LeNet عمل می‌کند. از نظر دقت دسته‌بندی مدل LeNet به حداکثر دقت ۷۰ درصد می‌رسید در حالی که این مدل می‌تواند به دقت ۹۵ درصد روی داده‌های ارزیابی برسد. از نظر سرعت همگرایی مدل LeNet در طی حدود ۲۰ گام به همگرایی می‌رسید در حالی که این مدل در طی ۷ گام به همگرایی می‌رسد. از نظر میزان تعمیم‌پذیری نیز این مدل بهتر است چرا که با تنظیم درست تعداد لایه‌های تنظیم‌شونده می‌تواند هم داده‌ها را با صحت فراوان دسته‌بندی کند و هم نرخ loss در آن کم‌تر است. مزیت‌های شبکه LeNet از جنس زمان مورد نیاز برای انجام محاسبات هستند. مدل LeNet از آن جا که مدل سبک‌تری است هر گام یادگیری را سریع‌تر طی کرده و در زمان تست سریع‌تر پاسخ تولید می‌کند. همچنین مدل LeNet روی سخت‌افزارهای کوچک راحت‌تر پیاده می‌شود در حالی که مدل Inception به فضای بیشتری نیاز دارد.

با توجه به توضیحات هر یک از مدل‌های Inception و LeNet مزیت‌هایی دارند اما با توجه به این که مزیت اصلی برای ما دقت دسته‌بندی است بنابراین از نظر ما مدل Inception بهتر عمل کرده است.