بهار ۱۴۰۰، دوره ۶، شماره ۱، پیاپی ۲۰
DOR :20.1001.1.2538161.1400.6.1.4.2

تاریخ دریافت مقاله: ۹۸/۱۲/۲۴
تاریخ پذیرش مقاله: ۹۹/۰۸/۰۵

نوع مقاله: مروری

مروری بر شبکه عصبی پیچشی و معماری‌های مختلف آن

فاطمه باجلان
دانشکده علوم و فنون نوین- دانشگاه تهران - ایران
پست الکترونیکی: bajelan.fateme@ut.ac.ir

هادی ویسی*
استادیار دانشکده علوم و فنون نوین- دانشگاه تهران - ایران
پست الکترونیکی: h.veisi@ut.ac.ir

محمد خوانساری
دانشیار دانشکده علوم و فنون نوین- دانشگاه تهران - ایران
پست الکترونیکی: m.khansari@ut.ac.ir

چکیده

شبکه عصبی پیچشی (CNN) یکی از روش‌های عمیق یادگیری ماشین است که با توجه به کارایی بالای آن، امروزه استفاده از آن در استخراج ویژگی و دسته‌بندی تصاویر امر بسیار رایجی شده است. در این مقاله برای آشنایی بیشتر پژوهشگران با این حوزه و نوآوری‌های مرتبط با این شبکه، ساختار اصلی این شبکه و تعداد ۱۰ معماری مهم و رایج آن مرور شده است. با مطالعه مقالات اصلی ارائه‌دهنده هر یک از معماری‌های شبکه پیچشی، نکات مهم و اصلی هر معماری که موجب بهبود عملکرد آن نسبت به موارد پیشین خود شده، جمع‌آوری و بررسی شده است. علاوه بر آن، مقایسه دقت معماری‌های مختلف این شبکه بخش دیگری از این مقاله مروری است. با توجه به حجم بالای مطالب در این حوزه و رشد سریع آن، نیاز به یک مرجع که همه مطالب ارائه شده تاکنون را در کنار هم مرور کرده باشد، انگیزه نوشتن این مقاله بوده است. مطالعه ساختارها و تکنیک‌های استفاده شده برای بهبود این شبکه می‌تواند به پژوهشگران در ایجاد ساختارهای بهینه‌تر و دقیق‌تر کمک کند.

واژه‌های کلیدی: یادگیری عمیق، شبکه عصبی پیچشی (کانولوشنال)، معماری‌های CNN (الکس نت، گوگل نت، رزنت)

مقدمه

یادگیری ماشین مجموعه روش‌هایی هستند که ابداع می‌شوند تا کامپیوترها بتوانند وظایفی را به‌صورت خودکار یاد بگیرند. روش‌های ابتدایی یادگیری ماشین توانایی پردازش داده‌های طبیعی به فرم خام را نداشتند و نیاز به بخش مجزایی وجود داشت تا داده‌های خام تبدیل به ویژگی‌هایی شود که برای ماشین قابل استفاده باشد، برای این کار از دانش متخصصین کمک می‌گرفتند و ویژگی‌ها با الگوریتم‌هایی که مختص آن نوع داده در

* نویسندهٔ مسئول

حوزه مشـــخص بود، استخراج می‌شـــدند تا بتواند مبنای تصمیم‌گیری ماشین باشد. امّا در روش‌های جدید یادگیری ماشـــین، تلاش می‌شـــود یادگیری نحـــوه بازنمایی[1] داده خـــام ورودی (تبدیـــل ورودی به ویژگی‌هـــا) نیز در خود الگوریتم گنجانده شود. در این‌گونه روش‌ها، داده می‌تواند به‌صورت خام به‌عنوان ورودی مورد استفاده قرارگیرد و نحوه بازنمایی لازم برای تصمیم‌گیری به‌صورت خودکار یافته می‌شود و نیازی به مهندسی ویژگی‌های آن‌ها نیست. روش‌هـــای یادگیری عمیق[2] عموما از این دســـته روش‌ها هســـتند که چندین لایه برای یافتن و ترکیب بازنمایی‌های مختلف از داده ورودی دارند. این لایه‌ها مســـتقیما و بدون نیاز به طراحی نیروی انسانی متخصص شکل می‌گیرند و با افزایش تعداد لایه‌ها می‌تواننـــد توابع پیچیده‌تری را یاد بگیرند. شبکه عصبی پیچشی (CNN)[3] یکی از این روش‌ها است که هر دو بخش استخراج ویژگی و دسته‌بندی را در خود دارد و در صورتی که در ســـایر روش‌های یادگیری ماشـــین نیاز به ماژولی برای استخراج ویژگی از تصاویر وجود داشـــته باشد نیز می‌توان از بخشی از شبکه عصبی پیچشی استفاده کرد.

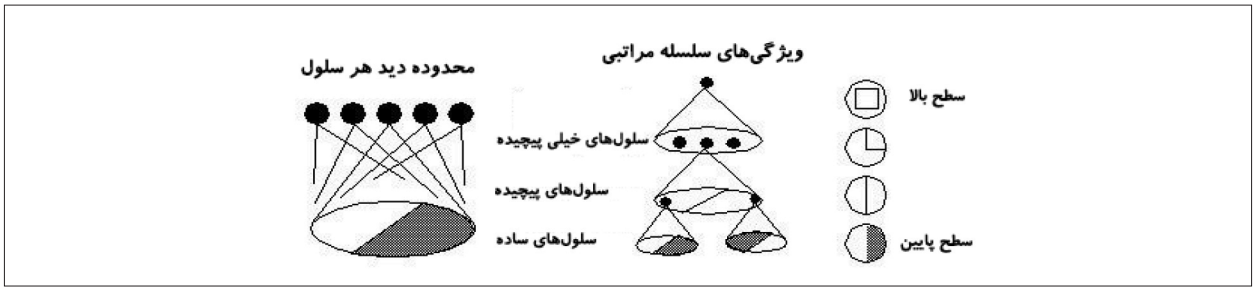
شبکه عصبی پیچشـــی عمیق کاربردهای گسترده‌ای در حوزه‌های مختلف دارد که تشـــخیص اشیاء در تصویر که کاربرد عام آن اســـت و به همین جهت سالانه چالشی به نام ILSVRC[4] با استفاده از دادگان ImageNet [۱] جهت تشخیص تصاویر وجود داشته که اکثر معماری‌های مهم شبکه عصبی پیچشی در روند حل این چالش ابداع شده و یا ارتقا یافته‌اند. کاربردهای خاص این شـــبکه‌ها در علوم مختلف نیز بسیار گســـترده اســـت و ما در اینجا به چند مورد از پژوهش‌های مرتبط به آن اشـــاره می‌کنیم: تشخیص چهره[5] [۲]، پردازش متن[۳]، در علم پزشکی تشخیص بیماری‌های مختلف[٤] مانند سرطان سینه[٥]، بیماری‌های دندانی[٦]، و سایر بیماری‌هایی که به کمک تصاویر قابل تشخیص است. در علم کشاورزی تشخیص بیماری‌ها یا کمبودهای شایع گیاهان در سطح کلان و با کمک تصاویر ماهواره‌ای[۷] و یا در سطح خرد به کمک تصاویر معمول[۸]. در علوم محیط زیست تشخیص الگوهای بوم‌شناسی[۹] و حتی در حوزه‌های مرتبط به بازی و بازیکن هوشـــمند مانند انجام بازی گو[6] [۱۰] از شـــبکه‌های عصبی پیچشی استفاده شده است.

افزایش حجم دادگان و گســـتردگی نیاز به پردازش و دســـته‌بندی تصاویر، نیاز به افزایش دقت در این حوزه را ایجاد کرده و هر ســـاله در دنیا معماری‌های مختلفی برای شبکه عصبی پیچشی ارائه می‌شود. با توجه به کاربردهای گسترده این‌گونه شـــبکه‌ها، نیاز به اطلاع از آن‌ها و مرور تکنیک‌های استفاده شده برای بهبود کارکردشان موردنیاز اســـت و انگیزه این مقاله ارائه مرجعی برای معرفی علمی این شبکه و انواع توسعه‌های مرتبط با آن در کنار هم بوده است. در ادامه مقاله، علاوه بر معرفی شبکه عصبی پیچشی و لایه‌هـــای رایج در آن در بخش دوم، معماری‌های مهم و ویژگی‌های اصلی آن‌ها در بخش ســـوم مرور شده است. در بخش چهـــارم تکنیک‌های مورد اســـتفاده برای بهبود معماری‌ها شـــرح داده شده و در فصل آخر نیز بر اساس پژوهش‌های موجود، مقایســـه‌ای میـــان معماری‌های ذکر شده آمده است.

۲- ساختار شبکه عصبی پیچشی

موضوع شبکه‌های عصبی پیچشی از سال ۱۹۹۰ مطرح بوده اســـت[۱۱]، امّا دلایل اصلی این‌که شبکه‌های عصبی پیچشی در چند سال اخیر امکان استفاده و ارتقاء یافته‌اند و به روشی غالب در فضای یادگیری ماشین تبدیل شده‌اند، افزایش حجم داده در دســـترس بـــرای آموزش و همچنین توســـعه توان پردازشـــی کامپیوترها بوده اســـت. این دو موضوع امکان افزایش تعداد لایه‌های شبکه‌های عصبی را ایجاد کرده تا بتوان ویژگی‌هـــای پیچیده‌تری را از داده‌ها استخراج نموده و بر آن اساس دسته‌بندی را با دقت بالاتر انجام داد.

1- Representation
2- Deep Learning
3- Convolutional Neural Networks (CNN)
4- ImageNet Large Scale Visual Recognition Challenge (ILSVRC)
5- Face Recognition
6- Go



شکل ۱: ساختار سلسه مراتبی سیستم بینایی و مغز انسان برای تشخیص اشیاء [۱۳]

شـــبکه عصبی پیچشـــی، نوعی از شـــبکه‌های عصبی مصنوعی جلورو[۷] اســـت که برای پردازش و دســـته‌بندی داده‌های چندبعدی مانند تصاویر ایجاد شده است. ساختار آن در تلاش برای شبیه‌سازی کارکرد سیستم بینایی انسان به‌صورت فعلی در آمده اســـت. همان‌طور که در شـــکل ۱ نشان داده شده است، بر اساس نتایج پژوهش‌های هیوبل و ویزل[۸] در سال ۱۹۵۹ سلول‌های چشم و مغز برای تشخیص اشیاء یک ساختار سلسه‌مراتبی از ویژگی‌ها را از ساده تا پیچیده تشـــخیص می‌دهند [۱۲]، بدین گونه که ساده‌ترین ویژگی‌های تصاویر مانند مـــرز به‌عنوان ویژگی‌های لایه اول و در لایه‌هـــای بالاتر مواردی ماننـــد انحنا و زاویه و در ادامه شـــکل‌ها و ویژگی‌های پیچیده‌تر تشـــخیص داده می‌شوند. علاوه بر آن، ویژگی دیگر این شبکه‌ها این است که هر ســـلول، نه تنها یک نقطه بلکه یک محدوده دوبعدی را مشـــاهده می‌کند، که به آن ناحیه دریافت[۹] می‌گویند. این نواحـــی دریافت که اطلاعات همســـایگی هر نقطه را تامین می‌کنند، غالبا با هم هم‌پوشانی دارند.

شـــبکه‌های عصبی پیچشـــی مانند ســـایر شبکه‌های عصبی مصنوعی، از نورون‌هایی ساخته‌شده‌اند که وزن‌ها و بایاس[۱۰]هـــا را ذخیـــره می‌کننـــد و در لایه‌های بعد خود روشی برای تصمیم‌گیری را به همراه دارند. اگر بخواهیم از شبکه‌های عصبی معمولی و لایه‌های تمام متصل[۱۱] برای دسته‌بندی تصاویر استفاده کنیم حجم هر لایه مخفی بسیار بزرگ و روند به‌روزرسانی آن بسیار طولانی خواهد بود.

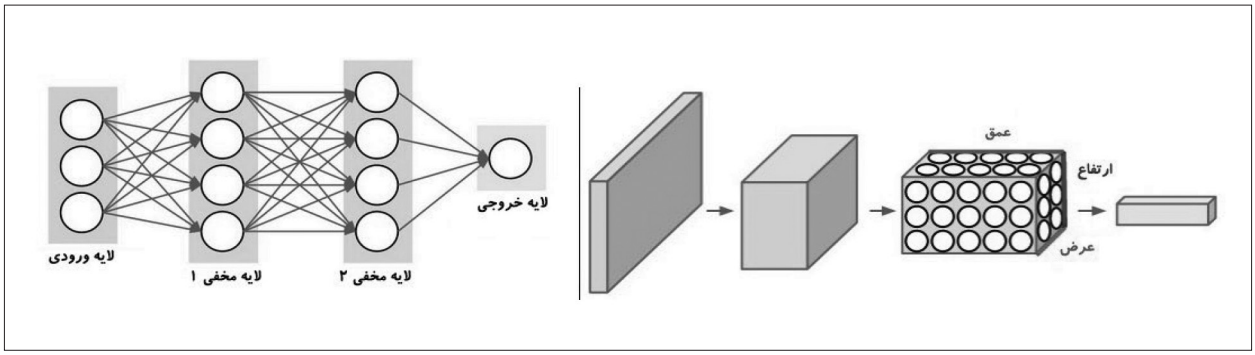
تفاوت این شـــبکه با سایر شـــبکه‌های عصبی این است که ورودی آن یک تصویر در نظر گرفته شـــده و روش انجام اعمـــال بر اســـاس ورودی تصویر تطبیق پیـــدا کرده‌اند و ساختاری را پیشـــنهاد می‌دهد که بتوان بهترین ویژگی‌ها را از تصاویر استخراج کرد. در ساختار این شبکه ورودی ســـه‌بعدی در نظر گرفته می‌شـــود؛ تصویـــری که دارای عرض، ارتفاع، و عمق است (شکل ۲).

یان لـــکان[۱۲]، به‌عنـــوان یکـــی از اولین افـــرادی که در خصوص شبکه‌های عصبی پیچشـــی صحبت کرده است [۱۵]، چند دلیل را برای رســـیدن به چنین شـــبکه‌ای عنوان می‌کنـــد: ۱. کاهش تعداد پارامترها بـــرای کاهش پیچیدگی شبکه ۲. استفاده از ویژگی محلی بودن مقادیر در تصاویر ۳. عدم وابســـتگی ویژگی‌ها به یک نقطه خاص از تصویر، که تحت عنوان اشتراک پارامترها مطرح می‌شود. در ادامه ساختار این شبکه و لایه‌های مختلف آن مرور می‌شوند.

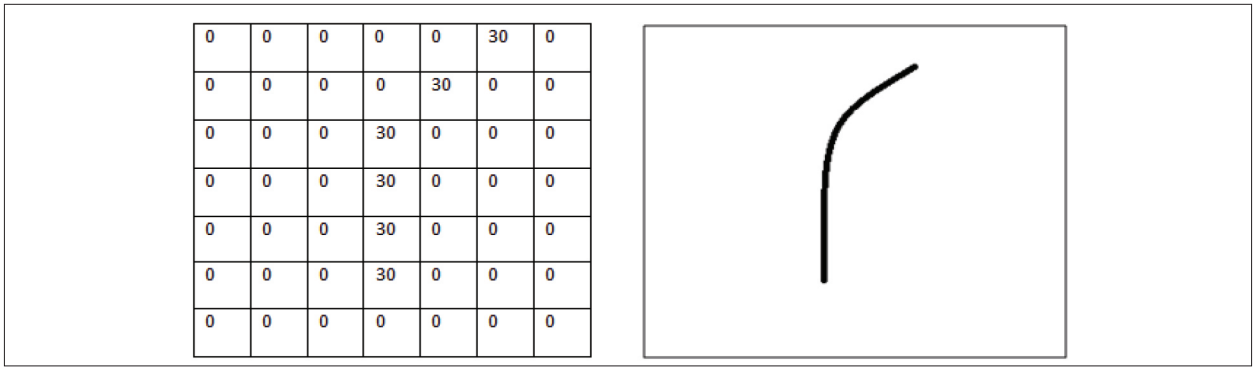
۲-۱- لایه‌های شبکه عصبی پیچشی

شبکه عصبی پیچشی مانند سایر شبکه‌های جلورو از تعدادی لایه متوالی تشکیل شده است که علاوه بر ورودی و خروجی، لایه‌های اصلی برای تشـــکیل این شـــبکه لایه پیچش[۱۳]، لایه ادغام[۱۴] (یـــا نمونه‌برداری)، لایه تمام متصل (مشابه آنچه در شـــبکه‌های عصبی پرسپترون چند لایه[۱۵] داریم) هســـتند که همـــراه آن‌ها موارد دیگری هســـتند که گاهی لایه خوانده می‌شـــوند، در ادامه مورد بررسی قرار می‌گیرند.

7- Feedforward
8- Hubel and Wiesel
9- Receptive Field
10- Bias
11- Fully Connected
12- Yann Lecun
13- Convolution
14- Pooling
15- Multi-Layer Perceptron (MLP)



شکل ۲: ساختار کلی شبکه‌های عصبی (چپ) - ساختار تغییریافته برای ورودی‌های تصویری یا چندبعدی (راست)[۱۴]



شکل ۳: بازنمایی یک پالایه تشخیص‌دهنده انحنا [۱۶]

ورودی

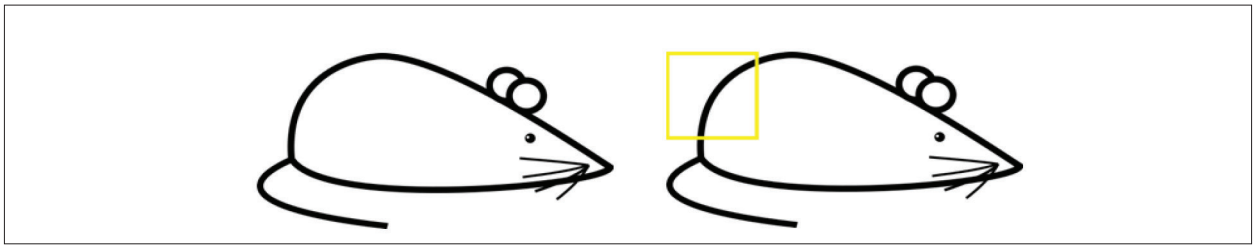
ورودی ایـــن شـــبکه یک تصویر اســـت که یـــک داده ســـه‌بعدی دارای طول و عرض و عمق به شـــمار می‌رود. عمـــق به معنای تعداد مؤلفه‌هایی اســـت که با آن مقدار هر پیکسل را نمایش می‌دهیم، مثلاً درصورتی‌که تصویر سیاه و ســـفید باشد عمق برابر ۱ و درصورتی‌که تصویر رنگی با سه مؤلفه‌ی قرمز، سبز و آبی باشد، عمق برابر ۳ خواهد بود و اگر در فضای دیگـــری رنگ‌ها را نمایش دهیم عمق برابر تعداد مؤلفه‌های آن فضا خواهد بود. مقادیر پیکسل‌ها در داده‌هـــای تصویری طبیعـــی به‌صورت محلـــی با هم ارتباط دارند، یعنی اگر ناحیه‌ای دارای رنگ خاصی باشد، پیکســـل‌های اطراف آن نیز دارای رنگ مشابه خواهند بود. به همین دلیل حفظ ســـاختار مقادیر در تصویر و پردازش آن‌ها در ساختار اصلی حائز اهمیت است. در عمل، بخشی از تصویر ورودی معادل ورودی شبکه است که یک پنجره محدود از تصویر است که روی تصویر لغزانده می‌شود تا کل تصویر را پیمایش کند.

پیچش

این لایه شبیه‌سازی عملکرد ســـلول‌های چشم انسان را بـــر عهـــده دارد و در واقع لایه‌ای اســـت که ویژگی‌های شبکه را تشخیص می‌دهد و در خود ذخیره می‌کند. در این لایـــه مفهومی به نام پالایش تعریف می‌شـــود که مجموعه نورون‌های لایه مخفی شـــبکه عصبی هستند، با این تفاوت که در شـــبکه عصبی مصنوعی هر نـــورون مخفی متصل به کل ورودی‌هـــای لایه قبل خود یا بخش مشـــخصی از آن‌هاســـت، امّـــا در لایه پیچـــش، این پالایش‌هـــا به بخش خاصـــی اختصاص ندارند و یک ویژگی را در کل ماتریس جســـت‌وجو می‌کنند (مانند آنچه در شکل‌های ۳ و ٤ نشان داده شده است).

یکـــی از مزایای محاســـباتی این ســـاختار، اشـــتراک پارامترها[16]ســـت که به این دلیل رخ می‌دهد که پالایه‌ها به ازای کل نواحی تصویر ساخته و به‌روزرسانی می‌شوند. مزیت دیگر این ســـاختار به‌عنوان اتصال تُنُک[17] شـــناخته

16- Parameter Sharing
17- Sparse Connectivity



شکل ۴: بازنمایی از تطبیق پالایه بر انحنای تصویر [۱۶]

شده است. کاهش تعداد اتصال‌ها – نسبت به اتصال کامل– و ایجاد نظم بخصوص، موجب کاهش تعداد پارامترها و در نتیجه کاهش محاسبات می‌شود.

پالایه‌ها، یا همان پارامترهای لایه پیچش، ماتریس‌هایی سه‌بعدی هستند که وزن‌های شبکه در این لایه را در خود ذخیره می‌کنند. این وزن‌ها باید در مرحله آموزش از روی داده آموزشی تخمین زده شوند. این پالایه‌ها با پیچش در ورودی لایه خـــود، در تمامی ابعاد، یک خروجی دو بعدی تولید می‌کنند که آن را نگاشـــت فعال‌ســـازی[۱۸] یا نگاشت ویژگـــی[۱۹] می‌نامیم. در واقع هر پالایـــه، پنجره ورودی از تصویر را پالایش کرده و آن پنجره از تصویر به یک عدد تبدیـــل می‌گردد. اندازه هر پالایه کـــه برابر با اندازه پنجره لغزان روی تصویر است، از جمله پارامترهای شبکه است که توســـط طراح شبکه تعیین می‌شـــود. بعد از یادگیری، هرکدام از ایـــن پالایه‌های لایه پیچـــش بیانگر یک ویژگی از تصویر می‌شـــوند. بـــه همین دلیل مـــا در هر لایه بیش از یـــک پالایه داریم، و عمق خروجـــی یک لایه همان تعداد پالایه‌هاســـت. هر پنجره از ورودی با عبـــور از هر پالایه (ضرب شـــدن مقادیر متناظر در همدیگر و جمع شدن) یک پیکســـل از تصویر خروجی ایجاد می‌کند که این موضوع در رابطه (۱) نشـــان داده شده است. در این رابطه I_j^l بیانگر ویژگی (پیکسل) jم از تصویر لایه l، w_{ij}^l وزن‌های پالایهٔ لایه l، و b_j^l بایاس اســـت. در اینجا f(.) تابع فعال‌سازی است که معمولا ReLU است.

$$I_j^l=f(\sum_i * I_j^{l-1}\times w_{ij}^l + b_j^l) \qquad (۱)$$

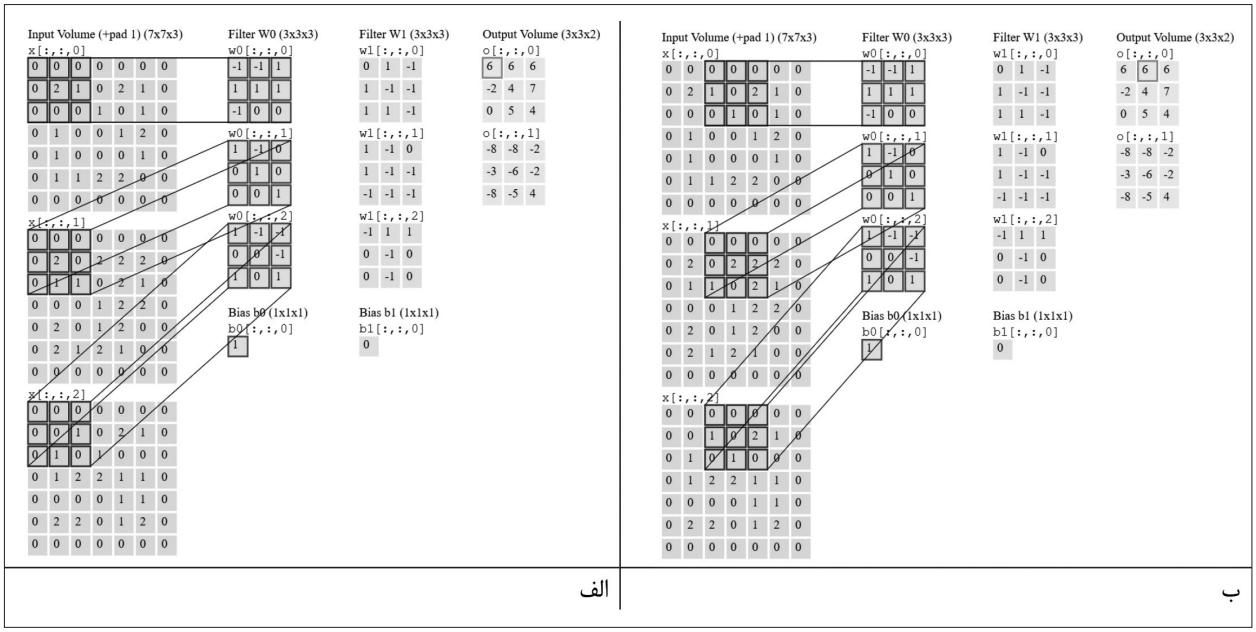
در خروجـــی لایه پیچش، بجز اندازه پالایه ســـه عامل مؤثر هستند، عمق یا همان تعداد پالایه‌ها، گام و حاشیه، که در ادامه شرح هر یک آمده است.

عمق: این پارامتر نشـــان‌دهندهٔ تعداد پالایه‌هاست. هر پالایـــه به یک ویژگی در تصویر اختصاص دارد، در نتیجه افزایش تعداد پالایه‌ها ویژگی‌های استخراج‌شده را افزایش می‌دهد. پالایه‌ها اختصاص به یک لایه از تصویر ندارند و از لحاظ عمق، هم عمق ورودی خود هســـتند، همانطور که چشم انسان نیز لایه‌های رنگی را مجزا از یکدیگر نمی‌بیند.

گام[۲۰]: مشـــخص‌کننده انـــدازه گام در پیچش پالایه بر روی تصویر است و در واقع بیانگر تعداد پیکسل‌هایی است که در لغزاندن پنجره ورودی روی تصویر در هر مرحله، پنجـــره را جابجا می‌کنیم. در حالـــت پیش‌فرض این عدد ۱ اســـت، یعنی پالایه یک پیکســـل یک پیکسل حرکت می‌کند. هر چه مقدار گام بزرگ‌تر باشـــد اندازه خروجی کوچک‌تر خواهد بود. گام هم در راستای محور طول هست و هم در راستای محور عرض. پنجره ورودی ابتدا در راستای افقی روی تصویر لغزانده می‌شـــود و زمانـــی که به انتهای آن ردیف از تصویر رسید به ابتدای ردیف بعدی با اندازه گام عمودی جابجا می‌شود.

حاشـــیه[۲۱]: یکی دیگر از پارامترهای مؤثـــر در اندازه خروجی، عددی است که با عنوان حاشیه تعریف می‌شود. این متغیر مشخص‌کننده تعداد ردیف صفرهایی است که به حاشـــیه تصویر ورودی اضافه می‌کنیم. این پارامتر برای تعیین انـــدازه تصویر خروجی از لایه پیچش مهم اســـت. با فرض این‌که اندازه تصویـــر ورودی W، اندازه پالایه F، اندازه گام S و اندازه حاشـــیه P باشد، آنگاه اندازه تصویر

18- Activation Map
19- Activation Map
20- Stride
21- Padding



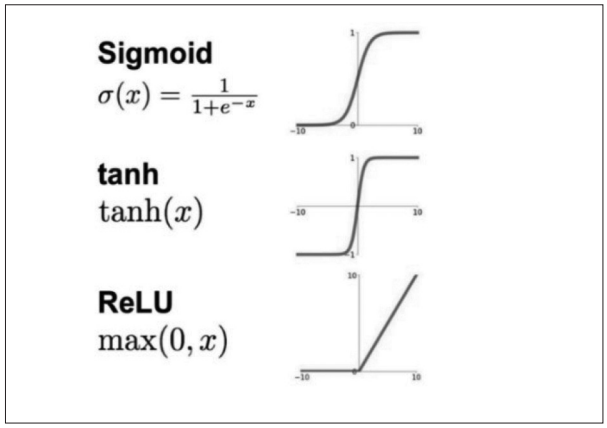
شکل ۵: مثال پیچش پالایه ۳ در ۳ در یک ورودی ۵ در ۵ با اندازه گام ۲ و مقدار حاشیه ۱[۱۴]

خروجی (W−F+2P)/S+1 اســت. ایــن متغیرها می توانند اندازه طول یا عرض تصویر باشــند. بنابراین، به‌صورت عمومی اثر پالایش کردن لایه پیچش، تولید ماتریس‌هایی با اندازه کوچک‌تر از تصویر ورودی است و برای حفظ اندازه و یا تولید اعداد صحیح (غیراعشاری) برای اندازه ماتریس خروجی این لایه از پارامتر حاشیه استفاده می‌شود.

در شــکل ۵ یک مثال آورده شده است که در آن اندازه تصویر ورودی ۷×۷ و رنگی (با عمق ۳) است و اندازه هر پالایه (پنجره روی تصویر) ۳×۳ است؛ دو بار عمل پیچش در یک پالایه نشان داده شده است. اندازه گام در این مثال ۲ و مقدار حاشــیه ۱ اســت. در این مثــال، اندازه تصویر خروجی ۳×۳ است و به تعداد پالایه‌ها (که در این مثال دو پالایه است) ماتریس خروجی داریم.

ReLU

اگر لایه را، برای مفهومی اســتفاده کنیم که در هر بار آموزش دارای فرآیند به‌روزرسانی باشد، ReLU را نمی‌توان یک لایه به حساب آورد و صرفاً می‌توان به دید یک تبدیل بر روی مقادیر خروجی لایه پیچش به آن نگاه کرد، تبدیلی کــه ویژگی خطی بودن را در شــبکه از بین می‌برد. از بین رفتــن این ویژگی مزیتی اســت که در شــبکه‌های عصبی به کمــک توابع فعال‌ســازی مختلف ایجاد می‌شــود زیرا اکثر دســته‌بندی‌های دنیای طبیعــی دارای مرزهای خطی مشــخص نیستند و ما نیاز به شبیه‌سازی توابع به‌صورت غیرخطی داریم. در واقع ReLU یک تابع فعال‌ســازی است که روی مقادیر پیکســل‌های تصویــر حاصل از خروجی لایه پیچش اعمال می‌شــود و می‌توانســت به جای آن هر تابع فعال‌ســازی غیرخطی دیگر مانند سیگموید هم باشد امّا یکی از مزایای استفاده از ReLU نسبت به توابعی مانند سیگموید این است که محاسبه آن بسیار سریع‌تر است، و این موجب می‌شود استفاده از آن برای آموزش شبکه‌های عمیق کم‌هزینه‌تر و رایج‌تر باشــد. نکته دیگر این است که آموزش شبکه با سیگموید به سرعت به حد اشباع می‌رسد، به این دلیل که گرادیان‌ها تنها در ناحیه کوچکی، غیرصفر و تأثیرگذار در به‌روزرســانی هســتند، و در ســایر نقاط مقادیری بســیار نزدیک به صفر می‌شوند که ضرب شدن آن‌ها در هم موجب می‌شود عملاً به گرادیان صفر برسیم، و دیگر شــاهد تغییراتی در نورون‌ها نباشیم. به این دلایل، ReLU به‌عنوان تابع فعال‌ســازی رایجی در این شبکه‌ها و برخی شــبکه‌های عمیق دیگر تبدیل شــده است. رابطه و نمودار این تابع و مقایســه آن با دو تابع فعال‌سازی رایج



شکل ۶: تابع فعال‌سازی ReLU و دو تابع معمول دیگر در شبکه عصبی عمیق

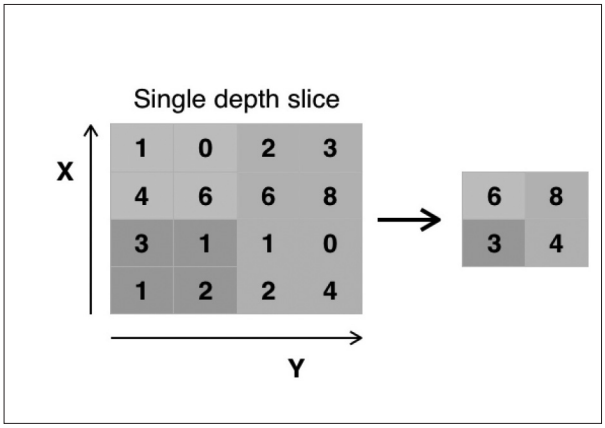
دیگر (ســیگموید و تانژانت هایپربولیک) در شکل ٦ آورده شده است.

محاسبات لایه‌های بعد از اعمال این تابع فعال‌سازی، که می‌تواند مربوط به یک لایه ادغام باشد و یا یک لایه پیچش دیگر، بر روی ماتریس‌های خروجی عبور داده شده از این تابع اعمال می‌شود.

ادغام

کاربــرد این لایــه، کاهش انــدازه نقشــه‌های ویژگی، بدون افزودن پارامتری به شــبکه است. برای کاهش تعداد پارامترها و کاهش میزان پردازش در شــبکه از لایه ادغام اســتفاده می‌شــود. ادغام به‌صورت مجزا بــر هر لایه از نگاشت فعال‌سازی اعمال می‌شــود. هدف مهم دیگری که از این کار دنبال می‌شود جلوگیری از بیش‌برازش[22] شبکه اســت. این لایه معادل با اعمال یک پالایه (معمولا با اندازه ۲×۲) روی ماتریــس حاصل از لایه پیچــش برای کاهش اندازه اســت و عملگر پالایهٔ بیشینه‌گیری یا میانگین‌گیری اســت. این پالایه با گام (معمولا برابر با ۲) جلو می‌رود و با این کار طول و عرض ماتریس کوچک می‌شــود. بدیهی اســت در این لایه پارامتری برای یادگیری (مانند وزن‌های شــبکه) وجود ندارد. در شکل ۷ یک مثال از عملکرد این با پالایهٔ ۲×۲ و با گام ۲ نشــان داده شده است که باعث شده اندازه تصویر نصف شود.

22- Overfitting



شکل۷: عملکرد لایه ادغام با عملگر بیشینه‌گیری با پالایهٔ ۲×۲ و گام ۲

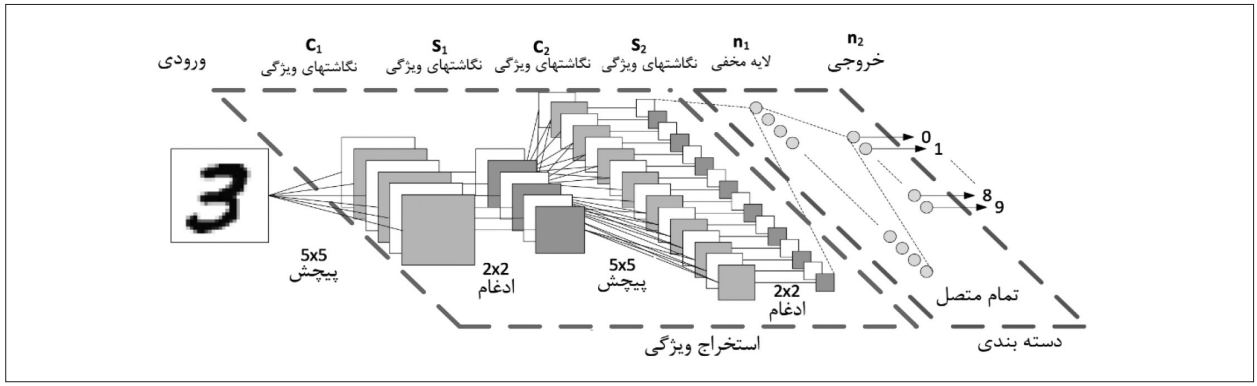
نرمال‌سازی دسته‌ای[23]

نرمال‌ســازی دســته‌ای مقادیر ماتریس‌ها را که بازه نامشــخصی دارند، به بازه مشــخصی مانند ۰ تا ۱ تبدیل می‌کند. برای این کار مزایای متفاوتی ذکر شده است: افزایش سرعت یادگیری شبکه، افزایش قدرت و کاهش بیش‌برازش در شــبکه. البته با توجه به محل قرارگیری آن در شبکه و لایه‌هایــی که پیش و پس از آن قرار می‌گیرند برخی از این مزایا را ایجاد می‌کند. این عملگر هم که نمی‌توان آن را لایه نامید، فقط در برخی از معماری‌ها استفاده می‌شود.

تمام متصل

لایه تمام متصل که شــاکله اصلی شــبکه‌های عصبی معمولی مانند پرســپترون چندلایه است، عموماً در انتهای ساختار شــبکه عصبی پیچشی به‌عنوان دســته‌بند به کار گرفته می‌شــود. این لایه خود شامل یک لایه خروجی و یک یا چند لایه ماقبل آن (مشابه لایه مخفی در شبکه پرسپترون چندلایه) است. هرکدام از این لایه‌ها می‌توانند تابع فعال‌سازی متفاوتی مانند تانژانت هایپربولیک و سیگموید داشته باشند. تابع فعال‌ســازی لایه آخر معمولا بیشــینه‌نرم (Softmax) اســت که همان‌طور کــه در رابطه (۲) آمده اســت، این لایه مشــابه سیگموید است با این تفاوت که مقادیر خروجی آن به‌صورت احتمال اســت و برای کاربردهای دسته‌بندی که دنبال یافتن دسته‌ای با بیشترین احتمال برای ورودی شبکه هستیم، کاربرد فراوانی دارد. با توجه به استفاده زیاد شبکه

23- Batch Normalization



شکل ۸: نمونه‌ای از یک شبکه عصبی پیچشی برای دسته‌بندی اعداد دست نوشته

پیچشـــی در کاربردهای دسته‌بندی، از این تابع در لایه آخر این شبکه استفاده می‌شود.

$$s(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \qquad (۲)$$

در نهایت ساختار یک شبکه عصبی پیچشی به‌صورت کلی شـــامل دو زیربخش اســـتخراج ویژگی و دسته‌بندی است که در زیربخش اســـتخراج ویژگـــی از تعدادی لایه پیچش و ادغام اســـتفاده می‌شـــود و زیربخش دسته‌بندی که در واقع همان پرســـپترون چند لایه است، فقط شامل لایه‌های تمام متصل است. شکل ۸ مثالی از ساختار کلی این شبکه برای دسته‌بندی تصاویر اعداد دست‌نوشته (۱۰ دسته) با دو لایـــه پیچش و ادغام متوالی و لایـــه تمام متصل با یک لایه خروجی و یک لایه مخفی به همراه اندازه‌های تصاویر هر لایه آورده شـــده اســـت. همان‌گونه که پیداست، اندازه تصاویر در هر لایه کوچک شـــده امّا تعداد آن‌ها بیشـــتر می‌شود. در واقع هر لایه مسئول استخراج سطحی از انتزاع ویژگی‌های ماتریس ورودی را برعهده دارد و هرچه از لایه ورودی دورتر می‌شـــویم و به سمت لایه خروجی حرکت می‌کنیم، سطح انتزاع بیشتر می‌شـــود؛ این مفهوم کارکرد اصلی استخراج ویژگی در شـــبکه عصبی عمیق است. در آخرین لایه زیربخش اســـتخراج ویژگی (که در شکل ۸ یک لایه ادغام است)، همه مقادیر در کنار هم تبدیل به یک بردار شده و به‌عنوان ویژگی استخراج شده توسط شبکه پیچشی به لایه تمام متصل داده می‌شوند تا روی آن‌ها دسته‌بندی صورت گیرد.

در زیربخش استخراج ویژگی در این شبکه، چیدمان دو لایه پیچش و ادغام و تعداد آن‌ها از جمله پارامترهایی است که باید در زمان طراحی شبکه در مورد آن‌ها تصمیم‌گیری شود. ســـاختار کلی چیدمان این لایه را می‌توان به‌صورت زیر بنویســـیم که بیان می‌کند N لایه پیچش متوالی که بعد از هر کدام از آن‌ها تابع ReLU اعمال شـــده، یک لایه ادغام به‌صورت اختیاری (بدین معنی که می‌تواند وجود نداشته باشد) می‌آید و کل این ساختار می‌تواند M بار تکرار شود.

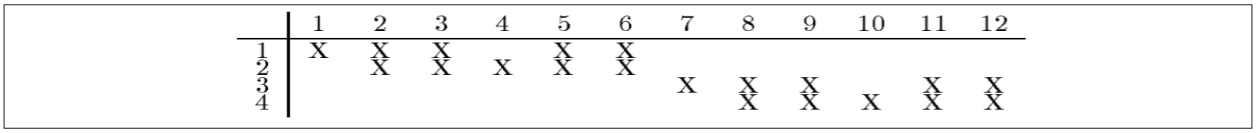
[[CONV →RELU]*N → POOL?]*M

در زیربخش دســـته‌بندی نیز، لایـــه خروجی یک بار و لایه (مخفی) ماقبـــل آن می‌تواند به تعداد K بار (حداقل یک بار) با تابع فعال‌سازی غیرخطی مرتبط (مانند تابع ReLU) تکرار شود. لذا کل ساختار این شبکه را می‌توان به‌صورت زیر نوشت.

INPUT →[[CONV → RELU]*N → POOL?]*M → [FC → RELU]*K → FC

برای آموزش شـــبکه‌های عصبی پیچشی که معادل با تنظیم و به‌روزرسانی مقادیر پالایه‌ها در لایه‌های پیچش و وزن‌های لایه‌های تمام متصل است، مشابه یادگیری شبکه های عصبی پرسپترون چندلایه از الگوریتم یادگیری پس انتشار خطا[۲۴] استفاده می‌شود [۱۱] [۲۰]. در این الگوریتم، ابتدا نمونه داده آموزش به شبکه داده می‌شود تا خروجی آن محاســـبه شود، سپس با استفاده از یک تابع خطا مانند میانگین مربعات خطا[۲۵] یا آنتروپی متقابل[۲۶] خطای شـــبکه

24- Error Back-Propagation
25- Mean Square Error (MSE)
26- Cross Entropy



شکل ۹: اتصالات بین لایه H2 و H3 در شبکه لینت [۱۱]

(در مقایسه با خروجی واقعی) محاسبه شده و با به‌روز کردن پارامترهای شبکه در جهت کاهش گرادیان[۲۷] سعی در کمینه کردن خطا می‌شود. برای انجام این کار، در هر لایه لازم است خطا محاسبه شود که به این منظور خطای لایه آخر به لایه ماقبل خود پس انتشار داده می‌شود تا خطای لایه ماقبل آخر به‌دست آید و از پس‌انتشار خطای این لایه، خطای مربوط به لایه قبل‌تر و این فرایند تا رسیدن به لایه ورودی ادامه پیدا می‌کند.

۳- معماری‌های شبکه عصبی پیچشی

بعد از مورد توجه قرار گرفتن شبکه عصبی پیچشی از سال ۲۰۱۲ به بعد، توسعه‌های مختلفی روی آن‌ها انجام شده است که منجر به معماری‌های نوینی از این شبکه‌ها شده است. این معماری‌ها هرچند در مفاهیم پایه بیان شده در بخش پیشین مشترکند امّا هر کدام دارای ایده‌هایی برای بهبود کارایی هستند. در این بخش تعداد ۱۰ مورد از معماری‌های مهم شبکه عصبی پیچشی مرور می‌شوند و ایده‌های پر استفاده‌تر و تکرار شده‌ی آن‌ها در فصل بعد به تفکیک قابل مطالعه است.

۱- لینت[۲۸]

یان لکان که ارائه‌دهنده اولین معماری از شبکه عصبی پیچشی است، در ابتدا شبکه لینت-۱ [۱۱] را در سال ۱۹۹۰ معرفی نمود. فرض مطرح شده برای طراحی این ساختار این است که یک شبکه عصبی تمام متصل با ورودی تصویر، به دلیل تعداد بالای پارامترها، قدرت تعمیم پایینی دارد، پس نیاز است شبکه‌ای با اتصالات محدودتر طراحی شود. همچنین با این فرض که ورودی‌ها تصویر هستند، از ویژگی محلی بودن داده‌ها استفاده شده و کاهش پارامترها را متناسب با این ویژگی نظم داده است. روش‌های کلاسیک تشخیص الگو در تصاویر، عموماً ویژگی‌های سطح پایین را تشخیص داده و با ترکیب آن‌ها باهم به ویژگی‌های سطح بالاتر دست پیدا می‌کنند. در این ساختار نیز با همین ایده، دو لایه پیچش قرار داده شده است که برای دست‌یابی به ویژگی‌های گفته‌شده، نورون‌های این لایه‌ها، اطلاعات یک همسایگی از ماتریس ورودی را در خود دارند.

این شبکه دارای چهار لایه، H1 و H2 و H3 و H4 است. که لایه اول و سوم لایه‌های پیچش با ماتریس وزن‌های ۵×۵ هستند، و لایه‌های دوم و چهارم لایه‌های نمونه‌برداری[۲۹] هستند -که بعداً به نام لایه ادغام شناخته شدند- و میانگین‌گیری آن‌ها میانگین وزنی، روی نواحی ۲×۲ است. ورودی لایه H1، تصویر اصلی است، از ٤ پالایه مورد استفاده، ٤ نقشه ویژگی[۳۰] -البته در آن زمان واژه پالایه برای این کاربرد استفاده نمی‌شده است- به دست می‌آید. ورودی H3، ٤ ماتریس است که نقشه‌های ویژگی به‌دست‌آمده از دو لایه قبلی هستند و با توجه به محدودیت توان پردازشی کامپیوترها در آن سال‌ها، این لایه ۱۲ پالایه یک یا دو لایه دارد و این تعداد برای پوشش دادن تمام ترکیبات نقشه‌های ویژگی کم است. به همین جهت هرکدام از پالایه‌ها روی یک یا دو تا از نقشه ویژگی‌ها اعمال می‌شوند. شکل ۷ تطبیق پالایه‌ها به نقشه‌های ویژگی لایه H2 را نشان می‌دهد.

این شبکه که برای تشخیص اعداد دست‌نوشته طراحی شده بود، در انتهای شبکه، مستقیماً لایه تصمیم‌گیری را داریم که دارای ۱۰ نورون است (برابر با تعداد ۱۰ رقم) و تمامی واحدهای خروجی لایه H4 به آن اتصال کامل دارند. تابع فعال‌سازی که پس از ادغام اعمال می‌شود، تانژانت هایپربولیک، و تابع فعال‌سازی لایه آخر، سیگموید است.

27- Gradient Descent
28- LeNet
29- Sub-Sampling
30- Feature Map

نتایج به‌دست آمده از لینت-۱ نشان داد شبکه پیچشی بزرگ‌تری برای اســتفاده کامل از داده ورودی موردنیاز است. به همین دلیل شبکه لینت-٤ [۱۷] با لایه‌ها و پالایه‌های بیشتر ارائه شد. این شبکه با ورودی‌های ۳۲×۳۲، دارای ٤ لایه اســت. لایــه اول آن پیچش با اندازه ۵×۵، با ٤ نقشــه ویژگی به‌عنوان خروجی اســت. در لایه دوم با ادغام دوتا دوتا از نقشــه‌های لایه قبل به ۸ نقشــه ویژگی دست پیدا می‌کنیم، در اینجا نیز ادغام به‌صورت میانگین‌گیری انجام می‌شــود و تابع فعال‌سازی پس از آن اعمال می‌شود. لایه ســوم پیچش با اندازه ۵×۵ به خروجی ۱٦ نقشــه ویژگی، و لایــه چهارم نیز ادغام به تعداد لایه قبل اســت، یعنی هر ماتریس لایه قبل، با میانگین‌گیری از واحدهای ۲×۲، به یک ماتریس در این لایه تبدیل می‌شــود و سپس دوباره اعمال تابع فعال‌ســازی انجام می‌شــود. در انتهای شبکه این بار پیش از لایه تصمیم‌گیری یک لایه مخفی تمام متصل با ۱۲۰ نــورون وجود دارد، و در انتها لایه تصمیم‌گیری که دارای ۱۰ نورون است، قرار دارد.

نســخه تقویت‌شــده[31] لینت-٤ از تکنیک تقویت شبکه اســتفاده می‌کند، که در این مورد لایه‌ها به شــکل موازی آمــوزش داده نمی‌شــوند و در هر لایه به موارد اشــتباه تشخیص‌داده‌شده در لایه‌های پیشــین وزن بالاتری داده می‌شــود تا شــبکه بتواند آن‌ها را بهتر یــاد بگیرد. اولین شــبکه به‌صورت عادی آموزش داده می‌شــود، دومی ۵۰ درصد از دادگان آموزش را از میان تشخیص‌های صحیح و ۵۰ درصد را از میان تشخیص‌های غلط می‌گیرد. شبکه ســوم نیز بر روی دادگانی کار می‌کند که دو شــبکه قبلی در خصــوص نتیجه آن توافق نداشــته‌اند. عــلاوه بر این، داده‌هایی که در دســته‌بند ابتدایی با درصد اطمینان بالایی تشخیص داده شوند دیگر وارد مراحل بعدی نمی‌شوند. این تصمیم‌ها موجب شده با این‌که این شبکه دارای ۳ دسته‌بند است، امّا میزان پردازش آن تنها ۱,۷۵ برابر لینت-٤ معمول است. به خاطر تعدد شبکه‌ها و به تبع آن، بیشتر بودن تعداد پارامترها، نیاز به دادگان ورودی بیشــتری برای آموزش

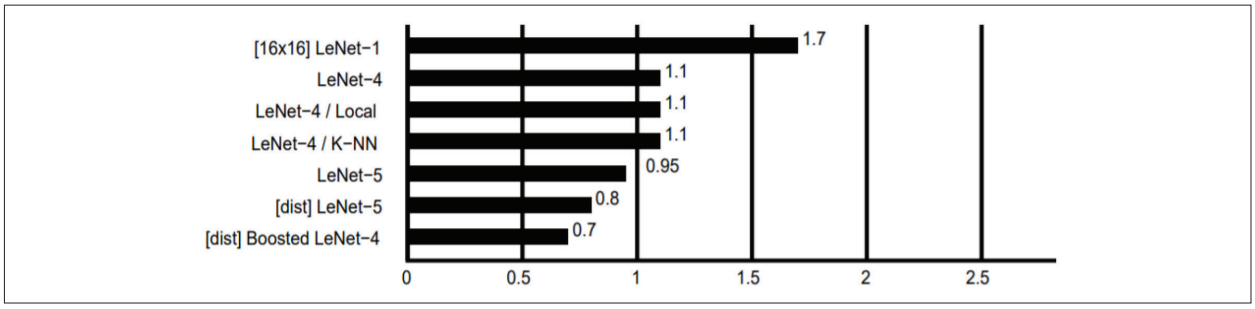
31- Boosted

شکل ۱۰: نویسه‌های ساده شده با اندازه ۱۲×۷ به‌عنوان مرجع محاسبه فاصله برای خروجی لینت۵ [۱۷]

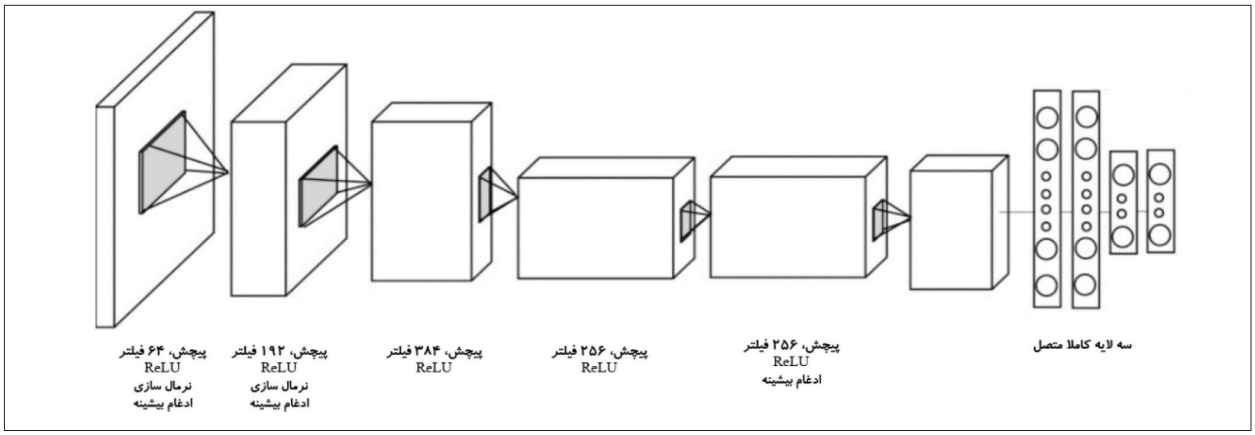
شــبکه است، که با اعمال تبدیل‌هایی مانند چرخش، مقیاس کردن و جابجایی، تعداد نمونه‌های مجموعه دادگان افزایش می‌یابد.

لینت-۵، [۱۷] که نسخه ارتقایافته لینت مربوط به سال ۱۹۹۵ است و در مقالات متعددی به آن اشاره‌شده و مرجع مقایسه قرار گرفته است، از لحاظ ساختاری شباهت زیادی به لینــت-٤ دارد. موارد تفاوت آن در تعداد لایه‌های پنهان پیش از لایه تصمیم‌گیری اســت که یک لایه افزایش داشته اســت، یعنی در انتهای شــبکه، دو لایه مخفی تمام متصل ۱۲۰تایی و ۸٤تایی قرار دارند. همچنین در لایه اول پیچش، ٦ نقشــه ویژگی دارد که این تعــداد در لینت-٤، برابر با ٤ نقشــه بود. این تغییرات کمک می‌کند میــزان ویژگی‌های استخراج‌شــده در لایه اول افزایــش یابد و از ترکیب آن‌ها در لایه ســوم ویژگی‌های متنوع‌تری به دست آید. در این شــبکه برای حرکت از لایه دوم -که در آن ادغام و اعمال تابع فعال‌سازی می‌شود- به لایه سوم، از ٦ نقشه ویژگی به ۱٦ نقشــه ویژگی می‌رسیم که برای این کار، طبق روال مشخصی سه، چهار یا شــش نقشه ویژگی از لایه دوم با همدیگر ترکیب‌شــده‌اند. در این ساختار، عمل ادغام به این شکل انجام می‌شود که ابتدا مقادیر داخل ناحیه ۲×۲ با هم جمع شده و سپس در یک متغیر قابل آموزش، ضرب و با یک بایاس قابل آموزش، جمع می‌شــود. در لایه خروجی این شــبکه، برای تصمیم‌گیری از فاصله اقلیدسی تابع پایه شعاعی[32] استفاده می‌شود و خروجی دارای کمترین فاصله از ۸٤ ویژگی لایه آخر، تصمیم شــبکه را مشخص می‌کند.

32- Euclidian RBF



شکل ۱۱: مقایسه نرخ خطای نسخه‌های مختلف معماری‌های لینت [۱۸]



شکل ۱۲: ساختار شبکه الکس‌نت

برای محاسبه فاصله، ماتریس‌های دوبعدی با اندازه ۷×۱۲ ساخته شده است که نماینده نویسه‌های مختلف هستند (شکل ۸) و فاصله از آن‌ها محاسبه می‌شود. در آموزش این شبکه نیز برای بالا بردن قدرت تعمیم‌پذیری، تبدیل‌هایی مانند چرخش، مقیاس و جابجایی بر روی برخی ورودی‌ها اعمال می‌شود.

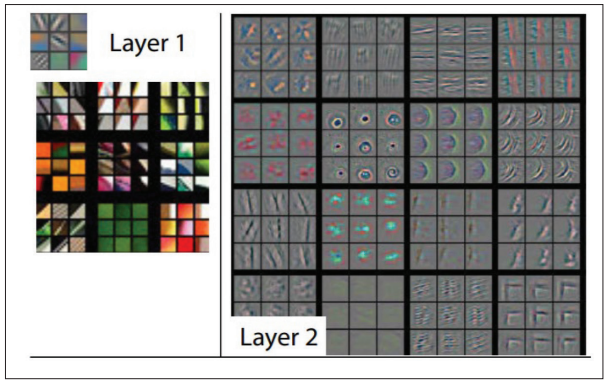
در مقاله [۱۸] بر روی مجموعه داده NIST [۱۹] و همچنین نسخه تغییریافته آن[۳۳]، که با توجه به نیاز شبکه از روی داده‌های اصلی ایجاد شده‌اند، مقایسه‌ای میان انواع لینت انجام‌شده است که نتیجه در شکل ۹ (بر حسب مقدار خطا) قابل مشاهده است.

۲- الکس‌نت

این شبکه [۲۰] که در سال ۲۰۱۲ معرفی شده است، به دلیل کارایی بالای آن در کاربرد دسته‌بندی تصاویر، شروع توجه جدی پژوهشگران به شبکه‌های عصبی پیچشی است. در این شبکه چند تغییر نسبت به کارهای قبلی صورت گرفته است مانند افزایش تعداد لایه‌ها، استفاده از تابع ReLU، نرمال کردن و استفاده از ادغام بیشینه. دلیل استفاده از توابع فعال‌سازی در شبکه‌های عصبی، ایجاد قابلیت غیرخطی بودن در شبکه است که بتوان با آن تصمیمات دقیق‌تری را ایجاد کرد. در شبکه الکس‌نت، به دو دلیل از ReLU به‌عنوان تابع فعال‌سازی استفاده شده است و این یک تغییر اساسی است که در معماری‌های پس از الکس‌نت هم در بسیاری از شبکه‌ها استفاده‌شده است. دلیل اول اشباع‌پذیری توابع فعال‌سازی مانند سیگموید و تانژانت هایپربولیک است و دلیل دوم ساده‌تر شدن محاسبات و افزایش سرعت آموزش شبکه است.

نرمال کردن خروجی‌های یک لایه، برای ورود به لایه بعد، به خاطر رفع مشکل اشباع توسط ReLU موردنیاز نیست، امّا استفاده از آن به تعمیم‌پذیری شبکه کمک می‌کند. این نوع از نرمال‌سازی، کار نرمال کردن را در یک نقطه، به‌صورت میان‌لایه‌ای، و نه در یک لایه، در همسایگی آن نقطه انجام می‌شود. هدف آن نیز تقویت مقادیر بزرگ و

33- Modified NIST (MNIST)



شکل ۱۳: ویژگی‌های استخراج‌شده توسط الکس‌نت در لایه اول و دوم [۲۲]

کـم تاثیر کردن مقادیر کوچک اسـت. اعمال این روش در دسـته‌بندی مجموعه داده CIFAR-10، خطا را از ۱۳ درصد به ۱۱ درصد کاهش داده است.

همچنیـن در لایـه ادغام این شـبکه، برخلاف شـبکه لینت از ادغام بیشـینه استفاده می‌شود. ویژگی دیگری که برای کاهش بیش‌برازش در شـبکه، هم‌پوشانی داشتن در بلوک‌هایی اسـت که روی آن‌ها عمل ادغام انجام می‌شود. ساختار الکس‌نت در شکل ۱۰ قابل‌مشاهده است.

لایـه اول پیچش آن دارای ۹۶ پالایهٔ ۳×۱۱×۱۱ اسـت. دلیل بعدی بودن پالایه‌ها، سه بعدی بودن تصویر ورودی شـبکه اسـت. عمل پیچش با اندازه گام ۳ روی آن اعمال می‌شـود که بر روی خروجی آن ادغام بیشـینه، با اندازه ناحیه ۳ و گام ۲ اعمال‌شـده است. سپس تابع غیرخطی و نرمال‌سـازی اجرا می‌شـود. لایه دوم پیچش شامل ۲۵۶ پالایهٔ ۴۸×۵×۵ است که بر روی خروجی آن نیز مانند لایه قبل ادغام بیشینه با همان ویژگی‌ها اعمال‌شده است. دلیل عدد ۴۸ این اسـت که ۹۶ نقشه ویژگی خروجی مرحله قبل به دو دسـته ۴۸تایی تقسیم‌شده است. سومین لایه پیچش ۳۸۴ پالایه با اندازه ۱۲۸×۳×۳، و لایه چهارم نیز ۳۸۴ پالایه با انـدازه ۱۹۲×۳×۳ را دارا اسـت. لایه پنجم پیچش، ۲۵۶ پالایه با انـدازه ۱۹۲×۳×۳ دارد که بـر روی خروجی آن ادغام بیشـینه با گام ۴ انجام می‌شـود و خروجی آن پس از پردازش در دو لایه تمام متصل، به لایه نهایی می‌رسد.

اسـتفاده از ترکیب کردن نتیجه چندین شـبکه مختلف برای تصمیم‌گیری نهایی، مانند لینت-۴ ارتقایافته، روشی اسـت که نتایج مثبتی را در بهبود دقت نشـان داده است امّا بـه دلیل افزایـش قابل‌توجـه میزان پـردازش از آن صرف‌نظر شده است. ایده‌ای که الهام گرفته از ایده ترکیب شبکه‌هاست و به‌عنوان جایگزین مطرح شده، تکنیک حذف تصادفی[34] [۲۱] اسـت که شرح آن در فصل بعد آمده و در این مقاله احتمـال فعال یا غیرفعال‌سـازی نورون‌ها ۰/۵ در نظر گرفته شـده است. با این روش، میزان پردازش به حدود دو برابر افزایش می‌یابد. این میزان پردازش بیشتر نیز صرف همگرا شـدن شبکه می‌شود که بیشتر از حالت عادی به طول می‌انجامد.

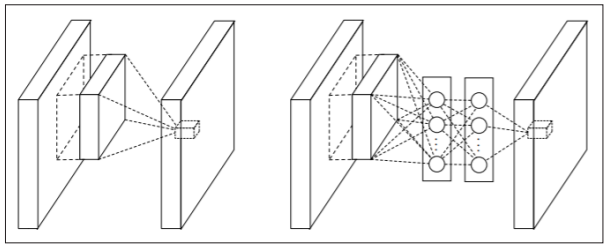
شـبکه الکس‌نت در ۲۰۱۲ با ۸ لایه در مسابقه ILSVRC و روی دادگان ImageNet [۱] بـا خطای حدود ۱۶ درصد و با فاصله زیاد از نفر دوم، برنده مسـابقه شد و از آن به بعد توجه به استفاده از شبکه عصبی پیچشی در پردازش تصویر و توسعه این شبکه‌ها سرعت گرفته است.

۳- ZFnet

شـبکه ZF در سـال ۲۰۱۳ برنده مسابقه ILSVRC شده است که در آن پژوهشگران به تکنیکی برای بصری‌سازی ویژگی‌هـای استخراج‌شـده در هر لایه از شـبکه عصبی پیچشـی رسـیده‌اند [۲۲] و بـه کمـک آن ویژگی‌هـای به‌دست‌آمده از اجرای الکس‌نت را به تصویر کشیده‌اند. از نتایج به‌دسـت‌آمده که در شکل ۱۱ قابل مشاهده است، دو مشکل در ساختار الکس‌نت مشاهده‌شده که برای حل آن‌ها تغییراتی در ساختار ایجاد شده است.

مشـکل اول این‌که ویژگی‌های استخراج‌شـده توسط الکس‌نت در لایه اول (شـکل ۱۳: ویژگی‌های استخراج‌شده توسـط الکس‌نت در لایه اول و دوم [۲۲])، بیشـتر شامل تغییرات با فرکانس‌های پایین و فرکانس‌های بالا هستند و ویژگی‌هایی با فرکانس میانه کمتر در آن‌ها وجود دارد. برای رفع این مشکل اندازه پالایهٔ لایه اول را از اندازه ۱۱×۱۱ به اندازه ۷×۷ تغییر داده‌اند. مشکل دوم در خصوص تداخل[35] ویژگی‌های استخراج‌شـده در لایه دوم است که به خاطر

34- Drop Out
35- Aliasing



شکل ۱۴: پیچش در شبکه‌های معمولی (چپ) جایگزینی پیچش با پرسپترون چندلایه در ساختار شبکه در شبکه (راست) [۲۳]

اندازه گام بزرگ پیچش رخداده، به همین دلیل اندازه گام از ٤ به ۲ تغییر یافته است. در نتیجه این تغییرات ویژگی‌های متنوع‌تر و متمایزتری با روش ارائه‌شده به‌دست‌آمده‌اند و کارایی شبکه را در مقایسه با الکس‌نت بهبود داده‌اند.

٤- شبکه در شبکه

ایده این پژوهش [۲۳]، مبنای شکل‌گیری ایده استفاده از پیچش ۱×۱ است، که در شبکه‌هایی مانند VGG، گوگل‌نت[۳۶] [۲٤] و رزنت[۳۷] [۲۵]، برای تغییر تعداد لایه‌ها و عموماً کاهش میزان پردازش مورد استفاده قرارگرفته است. این پژوهش استفاده از ساختار پرسپترون چندلایه را به جای استفاده از لایه پیچش پیشنهاد می‌دهد و دلیل آن را این اعلام می‌کند که شبکه عصبی پیچشی یک عملگر خطی است که ما پس از انجام آن یک عملیات جهت غیرخطی‌سازی انجام می‌دهیم، درصورتی‌که استفاده از یک تخمین‌گر جهانی توابع[۳۸] مانند پرسپترون چندلایه که توانایی بالاتری برای ساخت توابع غیرخطی دارد، کارایی شبکه را افزایش می‌دهد. نحوه عملکرد نیز بدین‌صورت است که پرسپترون چندلایه، به‌عنوان یک میکروشبکه، جایگزین عملگر پیچش می‌شود و ناحیه ورودی پیچش، ورودی آن قرار می‌گیرد. مثال ارائه‌شده در مقاله، میکروشبکه‌ای سه لایه است که در لایه‌های آن از ReLU برای غیرخطی‌سازی استفاده می‌شود (شکل ١٤).

همچنین ایده دیگری که در این مقاله مطرح می‌شود، استفاده از ادغام کلی میانگین[۳۹] است، که برای جایگزینی لایه‌های تمام متصل در انتهای شبکه‌ها استفاده شده است.

36- GoogleNet
37- ResNet
38- Universal Function Approximator
39- Global Average Pooling

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

شکل ۱۵: نسخه‌های مختلف معماری VGGnet [۲۶]

۵- VGGnet

در این معماری مربوط به سال ۲۰۱٤ [۲٦] که برنده چالش ILSVRC در همان سال شده است، طراحان تصمیم گرفته‌اند با افزایش عمق شبکه به دسته‌بندی بهتری دست پیدا کنند و درعین‌حال میزان پارامترها را نیز کاهش دهند. ایده این معماری شکستن پالایه‌ها به پالایه‌های کوچک‌تر است که هر دو هدف را برآورده می کند. در این ساختار پیچش اصلی مورد استفاده با پالایه ۳×۳ صورت می‌گیرد. در این مقاله چینش‌های متنوعی از لایه‌ها طراحی و سنجیده شده‌اند تا به یک نمونه معماری بهتر دست یابند. شش نوع معماری سنجیده شده در این مقاله در شکل ۱۳ آمده‌اند.

تمامی لایه‌های پیچش ۳×۳ استفاده‌شده در این شبکه‌ها دارای اندازه گام یک و یک ردیف حاشیه هستند تا اندازه تصویر تا انتهای بلوک ثابت بماند. ادغام بیشینهٔ موجود در جدول نیز با اندازه بلوک ۲×۲ با اندازه گام ۲ است، یعنی بلوک‌ها هم‌پوشانی ندارند. نکته دیگری که در این مقاله و در سنجش مدل‌های ذکرشده در بالا قابل‌توجه است این است که دو نوع آموزش و سه نوع سنجش برای آن در نظر گرفته شده است.

انواع آموزش اعمال‌شده، تک مقیاس[40] و چند مقیاس[41] هستند، آموزش تک مقیاس مشابه حالت عادی آموزش است و آموزش چند مقیاسی، با این هدف انجام می‌شود که یک جسم خاص در تصاویر مختلف، برحسب فاصله می‌تواند در اندازه‌های متفاوتی ظاهر شود، به همین دلیل نیاز است شبکه بتواند در اندازه‌های مختلف آن را تشخیص دهد. بدین خاطر اندازه تصاویر ورودی را به یک مقدار در بازه [کمینه: بیشینه] تغییر داده و سپس بر روی قطعه‌ای از آن به اندازه ورودی شبکه، مرحله آموزش را انجام می‌دهند.

در خصوص ارزیابی و استفاده، سه روش مطرح‌شده تک‌برش[42]، چندبرش[43] و ارزیابی چگال[44] است. استفاده از روش چندبرش برای ازریابی، بدین‌صورت انجام می‌شود که بر روی برش‌های متفاوتی از ورودی و نسخه‌های دَوَران‌یافته آن‌ها ارزیابی صورت می‌پذیرد و تصمیم نهایی از میانگین نتایج قطعات به دست می‌آید. روش چگال بدین‌صورت است که شبکه آموزش‌دیده را نه بر روی برش‌هایی از تصویر که بر روی کل تصویر اعمال می‌کنند، و لایه‌های تمام متصل انتهای شبکه را نیز به لایه‌های پیچش تبدیل می‌نمایند. درنهایت برای هر تصویر به‌جای یک خروجی و یک تصمیم، چند تصمیم خواهیم داشت که با میانگین و بیشینه گرفتن بر روی نتایج، نتیجه نهایی به دست می‌آید. یک مزیت این روش نسبت به برش دادن تصویر ورودی جلوگیری از محاسبه‌ی چندبارهٔ شبکه بر روی برخی پیکسل‌ها است، ضمن این‌که در واقع تمام حالات برش‌های مختلف را نیز پوشش می‌دهیم، نه یک تعداد معین از آن‌ها را. مزیت دیگر آن این است که در حالت برش خورده، لایه پیچش در اطراف تصویر حاشیه صفری که ما داده‌ایم را برای محاسبات استفاده می‌کند امّا در اینجا مقادیر واقعی در آن دخیل هستند و در واقع اصل داده بیشتر در تصمیم‌گیری موثر است.

40- Single Scale
41- Multi Scale
42- Single Crop
43- Multi Crop
44- Dense Evaluation

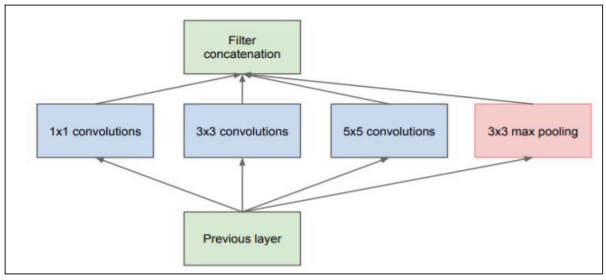
۶- گوگل‌نت

ساده‌ترین راه برای بهبود دقت شبکه‌های عصبی پیچشی، افزایش عمق و عرض آن‌هاست که افزایش عمق، به معنای افزایش تعداد لایه‌ها و افزایش عرض، به معنای بالا بردن تعداد ویژگی‌ها در هر لایه است. که این می‌تواند یک روش سریع -از منظر طراحی- برای بهبود دقت باشد. امّا این کار دو مشکل اصلی دارد. یکی افزایش تعداد پارامترهاست که می‌تواند به بیش‌برازش در شبکه بیانجامد. خصوصاً درصورتی‌که تعداد نمونه‌های برچسب‌دار در مجموعه دادگان محدود باشد، این موضوع تبدیل به مشکلی جدی در یادگیری شبکه می‌شود. مشکل دیگری که افزایش مداوم عمق شبکه‌ها به دنبال دارد، افزایش نیاز به منابع پردازشی است. مثلاً در یک شبکه عصبی پیچشی، هر تعداد افزایش در تعداد فیلترهای دو لایه پیچش پشت سر هم، به‌صورت درجه‌دو منجر به افزایش میزان پردازش می‌شود. که اگر ساختار اضافه شده کارایی بالایی نداشته باشد و منجر به ایجاد ماتریس‌های تنک[45] با درایه‌های متعدد نزدیک به صفر شود، هدر رفتن منابع پردازشی را به همراه خواهد داشت.

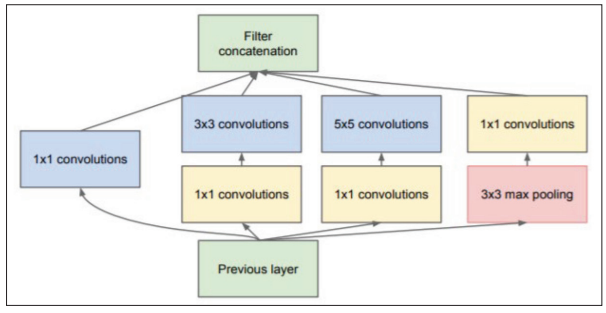
راه اصلی برای رفع هر دو مشکل می‌تواند استفاده بیشتر از ساختارهای تنک، به‌جای ساختارهای تمام متصل باشد، در این راستا لایه‌های تمام متصل انتهای شبکه‌های عصبی پیچشی رایج حذف شده و تنها یک لایه باقی می‌ماند. با توجه به این‌که پردازنده‌ها ساختار مناسب برای استفاده از این ساختارهای تنک را ندارند، با تبدیل آن‌ها به عملیات چگال انجام می‌شود. برای همین پژوهشگران این مقاله در سال ۲۰۱۵ [۲۴]، به دنبال ساختارهایی هستند که عملیات تنک را به کمک ساختارهای چگال پیاده‌سازی کنند. در واقع به دنبال بلوکی هستند با ویژگی مذکور طراحی شود و قابلیت تکرار شدن پشت‌سرهم برای ساختن یک شبکه عمیق را در کنار استفاده بهینه از منابع پردازشی داشته باشد. حالت پایه بلوک ارائه‌شده در شکل ۱۴ قابل‌مشاهده است.

در یک شبکه عصبی پیچشی معمولی، پالایه‌ها ثابت و

45- Sparse



شکل ۱۶: بلوک پایه معماری گوگل‌نت [۲۴]



شکل ۱۷: نسخه دوم از ماژول پایه معماری گوگل‌نت، که به نام طراحی گلوگاهی شناخته می‌شود [۲۴]

از پیش انتخاب‌شده هستند، امّا در شبکه گوگل، در داخل هر بلاک که ماژول اینسپشن[46] نام دارد سه عمل پیچش با اندازه پالایه‌های ۱×۱، ۳×۳ و ۵×۵ را در کنار هم داریم، و در کنار آن‌ها لایه ادغام نیز وجود دارد. لایه ادغام موجود مانند انجام عمل ادغام بر روی خروجی لایه قبل است. خروجی تمامی این عملیات به‌صورت چگال در کنار هم قرارگرفته خروجی بلوک را می‌سازند. شبکهٔ لایه بعد می‌تواند از هرکدام از این بخش‌ها استفاده نماید و این بدان معناست که شبکهٔ آموزش داده‌شده لایه‌های فعال ثابت ندارد و بر اساس نیاز دادگان و دسته‌بندی‌های آن‌ها، از خروجی انواع پیچش استفاده می‌شود.

در نسخه بعدی این شبکه از ماژول اینسپشن که در شکل ۱۵ تصویر شده است، برای کاهش تعداد پارامترها از پیچش ۱×۱ استفاده شده است. این ماژول با حفظ ابعاد طول و عرض، تعداد لایه‌های به‌دست آمده از پیچش را، کاهش می‌دهد.

ابتدای ساختار شبکه گوگل، مانند ساختارهای معمول شبکه عصبی پیچشی دارای دو لایه پیچش و دو لایه ادغام در میان آن‌هاست و پس‌ازآن ۹ ماژول اینسپشن قرار دارد که طبق شکل ۱۶ پس از بلوک‌های دوم، هفتم و نهم لایه ادغام اجباری به‌صورت مجزا از لایه ادغام موازی که درون ماژول‌ها موجود است، قرارگرفته است. مورد دیگر استفاده از ادغام کلی میانگین به‌جای لایه‌های تمام متصل میانی است که افزایش دقت را به دنبال دارد.

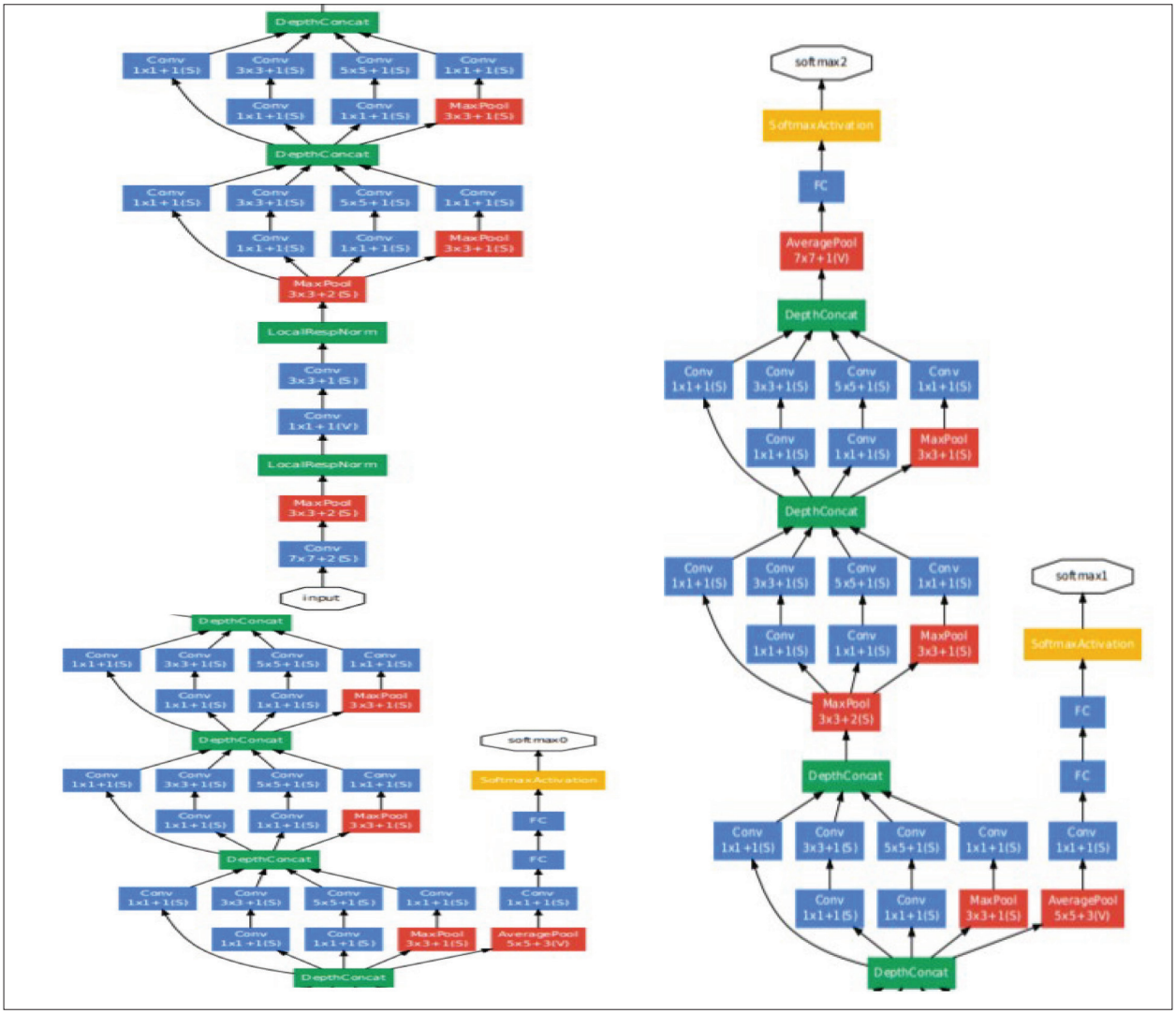
با توجه به عمیق بودن این ساختار، انتشار مؤثر گرادیان‌ها از لایه آخر به اولین لایه یک مسئله به‌حساب می‌آید. طبق پژوهش [۲۴] از دسته‌بندهای میانی برای بهبود آموزش شبکه استفاده شده است.

نسخهٔ بعدی معماری گوگل با نام اینسپشن-۲ در مقاله [۲۷] ارائه شده است. در این مقاله ابتدا اصولی برای طراحی شبکه‌ها ذکر شده و سپس معماری موردنظر ارائه شده است، این اصول عبارتند از: ۱. شبکه نباید دچار کمبود اطلاعات ورودی شود، بدین معنا که کاهش اندازه ورودی، خصوصا در مراحل ابتدایی نباید به‌یک‌باره انجام شود، بلکه این کاهش باید در طول شبکه و به‌صورت تدریجی صورت پذیرد. ۲. افزایش تعداد نقشه‌های ویژگی در یک لایه کمک می‌کند ویژگی‌های به‌دست‌آمده وابستگی کمتری به یکدیگر داشته باشند که این موضوع، سرعت آموزش ویژگی‌ها را افزایش می‌دهد. ۳. کاهش عمق ورودی پیش از اعمال لایه‌ها، مشکل جدی در بازنمایی دادگان ایجاد نمی‌کند و موجب کاهش دقت زیادی نمی‌شود. ٤. برای دست‌یابی به شبکه‌ای بهینه، نیاز است تناسبی از عمق و عرض شبکه -به معنای تعداد پالایه‌ها در هر لایه- را در نظر بگیریم. بر اساس یک قاعده کلی انتظار می‌رود افزایش عمق و عرض شبکه ما را به شبکه‌ای باکیفیت‌تر برساند، امّا با توجه به محدود بودن توان پردازشی، باید به حالت بهینه‌ای از عرض و عمق متناسب با آن دست یافت.

در معماری ارائه‌شده چند ایده برای بهبود ساختار به کار گرفته‌شده است. ایده اول، همان شکستن پیچش‌ها به پیچش‌های کوچکتر است که هم به شکل متقارن و هم نامتقارن مورد استفاده قرار گرفته است.

ایده دیگری که به کار گرفته شده است، کاهش اندازه

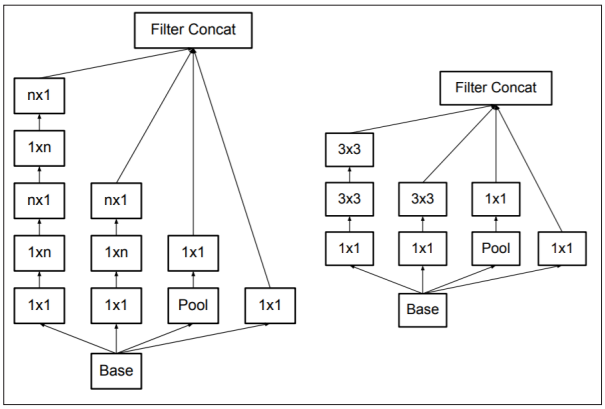
46- Inception



شکل ۱۸: ساختار کلی گوگل‌نت – نسخه یک [۲۴]

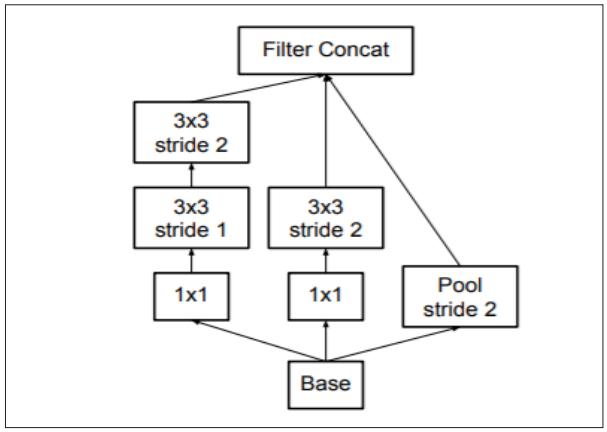
نقشه‌های ویژگی به‌گونه‌ای است که میزان پردازش مورد نیـــاز در عمل پیچش را کاهش دهد. اســـتفاده از پیچش و سپس ادغام، یک چینش رایج در شبکه‌های عصبی پیچشی اســـت، با فرض این‌که ادغام انجام شـــده ۲×۲ باشد عملا ۷۵٪ پردازش انجام شـــده بی‌استفاده باقی می‌ماند و توان پردازشـــی هدر می‌رود. همچنین قرار دادن ادغام پیش از پیچش، با اصل یک تناقـــض دارد چون حجم قابل توجهی از دادگان را حذف می‌کند و یک گلوگاه اطلاعاتی حســـاب می‌شود. راه‌حل ارائه‌شده استفاده از پیچش با اندازه گام ۲ به جای این دو لایه پشت سر هم است (شکل ۱۸).

معماری اینسپشن-۲ دارای سه نسخه است که تفاوت

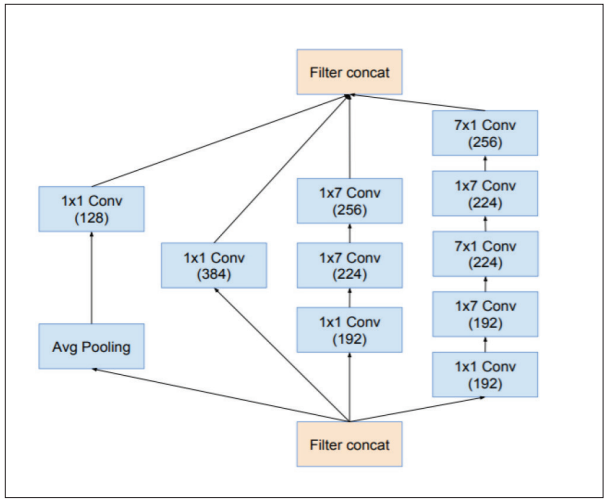


شکل ۱۹: ماژول پایه با پیچش‌های ۳×۳ (سمت راست) استفاده از پیچش‌های n×1 و 1×n به‌جای n×n در ماژول پایه (سمت چپ) [۲۷]

آن‌ها در اســـتفاده از برخی از ایده‌های بالا در ماژول‌های سازنده است. نسخه معماری که تمامی ایده‌های مطرح‌شده



شکل ۲۰: استفاده از پیچش با گام ۲ به‌جای ادغام [۲۷]

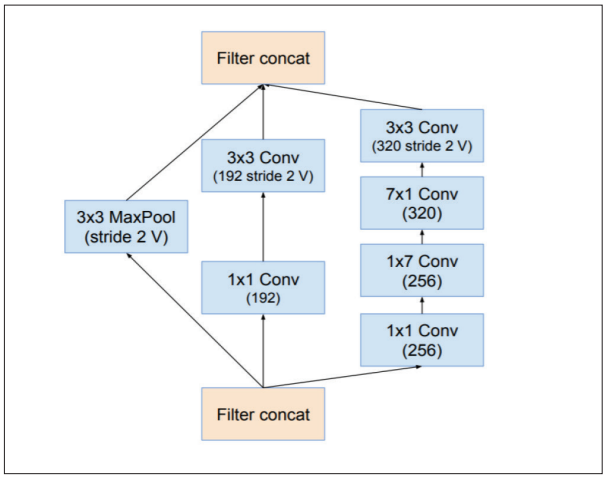


شکل ۲۱: بلوک اینسپشن B مورد استفاده در ساختار اینسپشن-۴ [۲۸]

در این مقاله را در خود دارد، اینسپشن-۳ [۲۷] است.

مقاله دیگری که در آن نسخه‌های بعدی این معماری ارائه می‌شود، [۲۸] است که نسخه چهارم اینسپشن و نسخه‌های بعدی که در ترکیب با معماری رِزنت ارائه‌شده‌اند را معرفی می‌کند. در نسخه چهارم، تلاش شده است شبکه یکدست‌تری ساخته شود. در نتیجه پایه شبکه[۴۷]، یعنی بخش ابتدایی آن، دچار تغییر شده است و علاوه بر آن، سه نوع بلوک اینسپشن A، B و C برای ورودی با اندازه‌های ۳۵×۳۵، ۱۷×۱۷ و ۸×۸ طراحی و در ساختار مورد استفاده قرارگرفته است. در هر بلوک پیچش‌ها با اندازه‌های متفاوتی قرار دارند که تعداد پالایه‌های هر کدام به‌صورتی انتخاب شده است که توان پردازشی به‌صورت متعادل در شاخه‌ها مورد استفاده قرار گیرد. در این بلوک‌های اینسپشن، اندازه

47- Stem



شکل ۲۲: بلوک کاهش از اندازه ۱۷×۱۷ به ۸×۸ [۲۸]

ورودی دستخوش تغییر نمی‌شود، برای کاهش اندازه ورودی بلوک‌های دیگری تحت عنوان بلوک‌های کاهش[۴۸] طراحی شده‌اند که ایده مطرح شده در مقاله [۲۷] در خصوص افزایش اندازه گام پیچش در آن به کار رفته است. در شکل ۱۹ و ۲۰ دو نمونه از ماژول‌های گفته‌شده قابل مشاهده هستند.

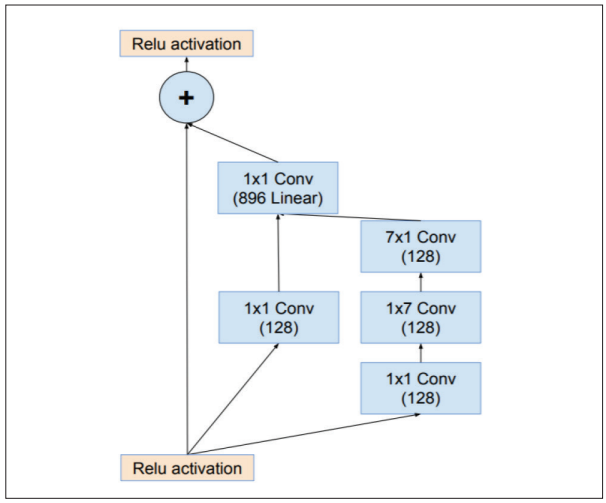
در معماری اینسپشن-رِزنت [۲۸] نیز بلوک‌های اینسپشن، با میان‌برهای باقیمانده ترکیب‌شده و بلوک‌های پایه را مانند شکل ۲۱ درآورده‌اند. همان‌طور که قابل‌مشاهده است، پیش از انجام عمل جمع ورودی، به کمک یک پیچش ۱×۱ تعداد لایه‌های منتج از سایر عملیات انجام‌شده در بلوک را با ابعاد ورودی یکسان‌سازی می‌کنیم تا عمل جمع به‌صورت نظیر به نظیر قابل انجام باشد.

۷- رِزنت

در سال‌های اخیر، در حوزه شبکه‌های عصبی پیچشی، افزایش عمق به‌عنوان راهکاری برای افزایش دقت این شبکه‌ها در نظر گرفته می‌شود. امّا مسئله‌ای که در خصوص افزایش عمق ما را دچار مشکل می‌سازد، کوچک شدن گرادیان‌ها و به‌روز نشدن آن‌ها طی مراحل آموزش است، که به آن محو گرادیان[۴۹] می‌گویند.

در مقاله [۲۵] مربوط به سال ۲۰۱۶ که ارائه‌دهندهٔ

48- Reduction
49- Vanishing Gradient



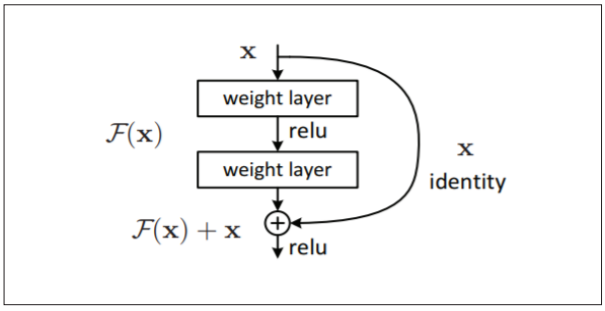
شکل ۲۳: بلوک اینسپشن B ترکیب‌شده بااتصال میان‌بر [۲۸]

معماری رِزنت است، بیان شده است که برای رفع این مشکل، می‌توان از ساختارهای مبتنی بر باقیمانده[50] استفاده کرد، و این فرضیه بر مبنای بهبودهایی است که در استفاده از روش‌های مبتنی بر باقیمانده در مسائل دیگر و در روش‌های مبتنی بر ایجاد میان‌بر در شبکه‌ها دیده شده است. راهکار ارائه شده برای رفع این مشکل در این پژوهش، استفاده از اتصال میانبر و ساختن شبکه‌ای مبتنی بر بازنمایی باقیمانده است.

ساختار اولیه این شبکه شامل ۳٤ لایه است و ساختار آن با الهام از VGGnet، به جز در لایه اول، متشکل از پیچش‌هایی با اندازه ۳×۳ است. این شبکه شامل ۱٦ بلوک است. بلوک اولیه سازنده شبکه، شامل یک جفت پیچش ۳×۳، با یک لایه ReLU –برای غیرخطی کردن– در میان آن‌ها و اتصال میان‌بر است که پس از جمع شدن مقدار ورودی با خروجی پیچش‌ها، بار دیگر ReLU بر آن اعمال می‌شود (شکل ۲۲). نکته‌ای که در خصوص این ساختار وجود دارد این است که اتصال‌های میان‌بر، پارامتر جدیدی به شبکه اضافه نمی‌کنند، که این موضوع، امکان مقایسه با شبکه‌های ساده با عمق مشابه را امکان‌پذیر می‌سازد.

ساختار کلی شبکه رزنت در شکل ۲۳ نشان داده شده است. همان‌طور که در این شکل دیده می‌شود، برخی از اتصال‌های میان‌بر به‌صورت خط‌چین ترسیم‌شده‌اند که این

50- Residual



شکل ۲۴: ساختار پایه بلوک رزنت [۲۵]

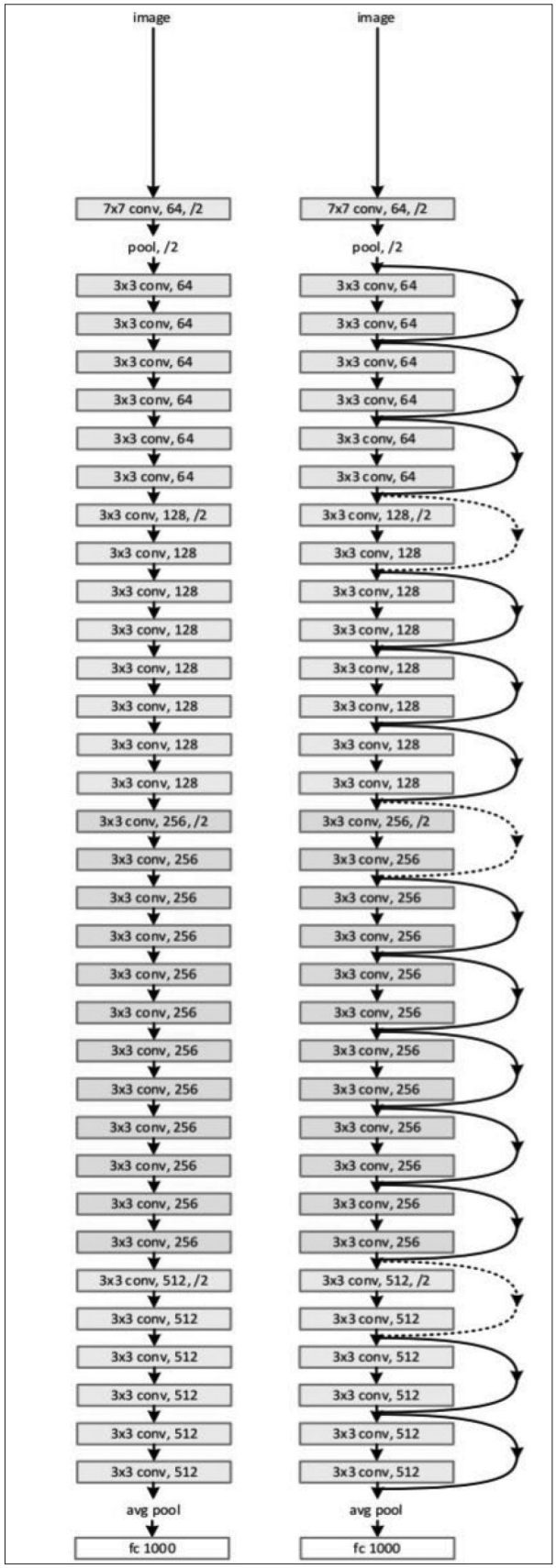
موارد، تعداد لایه‌های نقشه ویژگی‌ها را افزایش می‌دهند. در انتهای این شبکه نیز یک لایه ادغام کلی میانگین و یک لایه تمام متصل برای تصمیم‌گیری، به همراه softmax وجود دارد.

در این مقاله [۲۸] نوع دیگری از ساختار رِزنت و سه نمونه از آن نیز ارائه می‌شود. این نوع از ساختار را، ساختار گلوگاهی[51] نامیده‌اند. ایده آن این است که در پیچش‌های ۳×۳ که پرهزینه‌ترین فعالیت ساختار رِزنت به حساب می‌آید، تعداد نقشه‌های ویژگی را به کمک پیچش ۱×۱ کاهش داده و پس از انجام پیچش ۳×۳، آن را به تعداد ابعاد موردنیاز بازگردانند. شکل ۲٤ نشان‌دهنده بلوک‌های ساختار گلوگاهی شامل پیچش ۱×۱ برای کاهش ابعاد، پیچش ۳×۳ برای استخراج ویژگی و یک پیچش ۱×۱ دیگر برای بازگرداندن ابعاد به تعداد موردنظر است. اگر بخواهیم همان ساختار ۱٦ بلوکی (۳٤ لایه) را با بلوک‌های جدید بسازیم، دارای ۵۰ لایه خواهد بود.

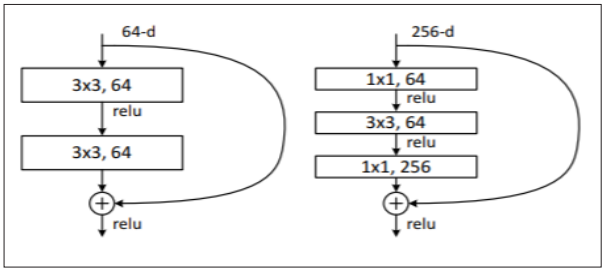
در این پژوهش، برای سنجش نتایج، یک ساختار ۳٤ لایه بدون اتصال‌های میان‌بر، ساختار ۳٤ لایه اولیه رِزنت و ساختار ۵۰ لایه گلوگاهی را بر روی مجموعه دادگان CIFAR-10 اعمال کرده‌اند که بهبود دقت را به میزان ۰,۵٤ درصد نشان داده است.

همچنین دو نمونه دیگر از ساختار گلوگاهی با تعداد لایه‌های ۱۰۱ و ۱۵۲، برای دسته‌بندی مجموعه دادگان ImageNet که حجم بالاتری دارد ساخته شده است. تا اینجا با افزایش بلوک‌ها شاهد افزایش دقت بوده‌ایم. امّا نمونه‌ای از ساختار گلوگاهی با ۱۲۰۲ لایه ساخته شده است که

51- Bottleneck



شکل ۲۵: ساختار یک شبکه معمولی ۳۴ لایه (چپ) در مقایسه با رزنت ۳۴ لایه (راست) [۲۵]



شکل ۲۶: ماژول اولیه رزنت (چپ) و ماژول گلوگاهی رزنت (راست) [۲۵]

میزان دقت آن از ساختار ۱۵۲ لایه کمتر بوده است، و حدس پژوهشگران مقاله بر این است که این افزایش بالای تعداد پارامترهای شبکه، موجب بیش‌برازش شده است.

نکته‌ای که در خصوص استفاده از میان‌برها در شبکه وجود دارد این است که خروجی هر بلوک باید ابعاد مساوی با ورودی آن داشته باشد تا بتوان جمع نظیر به نظیر را در آن انجام داد. درصورتی‌که تعداد لایه‌های ورودی و خروجی برابر نباشند، با ضرب یک ماتریس تصویر خطی[۵۲] از ماتریس ورودی را، با ابعاد موردنیاز می‌سازیم.

۷-۱- ایده‌های بهبود ساختار رزنت

در سال‌های اخیر اکثر پژوهش‌ها به بهبود ساختارهای مطرح‌شده مانند رِزنت و گوگل‌نت می‌پردازند و کمتر مواردی یافت می‌شود که ساختار را از پایه به نوع دیگری می‌سازند. برخی از این موارد در زیر ادامه بررسی می‌شوند.

در ساختار رِزنت، خروجی هر بلوک، ترکیب مشخصی از ورودی و نتیجه عملیات درون بلوک است. ایده مطرح‌شده در مقاله [۲۹] که هایوِی‌نت[۵۳] نام‌گذاری شده است، پارامتر دیگری را به هر بلوک می‌افزاید و آن ضریبی است که مشخص می‌کند چه نسبتی از خروجی متعلق به ورودی و چه نسبتی متعلق به سایر عملیات درون بلوک باشد. این ایده که الهام گرفته از معماری شبکه عصبی حافظه کوتاه‌مدت ماندگار[۵۴] است، در واقع یک سازوکار دریچه‌ای[۵۵] پیش از خروجی بلوک قرار می‌دهد (فرمول ۳).

52- Linear Projection
53- HighwayNet
54- Long Short-Term Memory (LSTM)
55- Gating Mechanism

$$y_i = H_i(X) * T_i(X) + x_i * (1 - T_i(X)) \quad (۳)$$

این ســـازوکار برای بهینه‌سازی شبکه‌های عمیق ارائه شده است، بدین‌صورت که سیگنال را به‌صورت تطبیق‌پذیر برای عمق مربوطه در شـــبکه عبور می‌دهد، که این امر به جریان بهتر گرادیان در مرحله پس‌انتشار کمک می‌کند.

در مقاله دیگری [۳۰] انواع تغییریافته‌ای از بلوک رِزنت مورد بررسی قرار گرفته و نهایتاً تغییر در چینش تبدیل‌ها ارائه شـــده است که به جریان بهتر گرادیان‌ها در آن کمک می‌کند. این چینش جدید امکان آموزش شبکه‌های عمیق‌تر از ۱۰۰۰ لایه را پدید می‌آورد. به بلوک ساخته‌شده در این مقاله رِزنت با پیش‌فعال‌ســـازی[۵۶] گفته می‌شود، که در آن عملیات نرمال‌سازی دسته‌ای و ReLU پیش از عمل پیچش انجام می‌شـــوند، این عملیات نرمال‌ســـازی تأثیری مانند منظم‌ســـازی[۵۷] بر روی داده‌ها دارد. همچنین در این مقاله سعی شـــده اســـت کمترین میزان تغییر بر روی خروجی بلوک صورت پذیرد، به همیـــن دلیل عملیات ReLU پس از جمع شـــدن دو مقدار خروجی از بلوک حذف شـــده و در شاخه‌ی مربوط به عمل پیچش قرارگرفته است (شکل ۲۵).

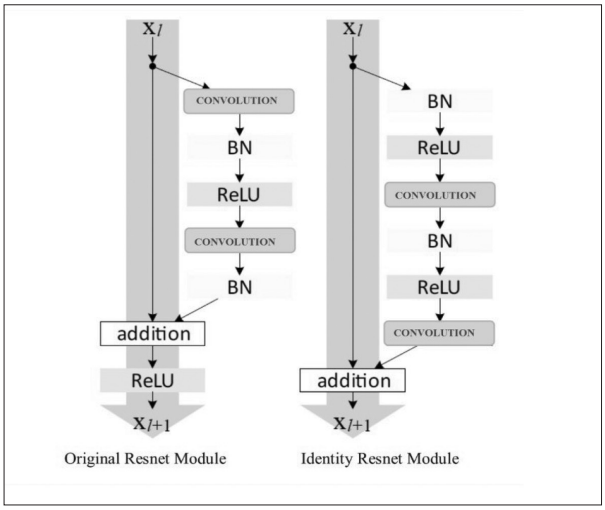
مقاله دیگری با عنوان شبکه‌های باقیمانده عریض[۵۸] [۳۱]، تلاش کرده اســـت با تغییر متغیرهای معمول در بلوک‌های رِزنت مانند عمق، عرض و نوع پیچش مورداســـتفاده تأثیر این عوامل را در بهبود کارایی بررســـی کنـــد. موارد زیر برخی از دستاوردهای این مقاله هستند:

a. از نظـــر عمق بلوک، یا همان تعداد پیچش‌های درون بلوک، دو پیچش بهترین عملکرد را داشته و با افزایش آن، تعداد اتصال‌های میان‌بر در شـــبکه کاهش یافته و به دلیل مشکل شدن بهینه‌سازی، دقت شبکه کاهش می‌یابد.

b. از نظر نـــوع پیچش‌های داخل بلوک، حالات مختلفی در نظر گرفته شـــده است مانند که حالت ساده یعنی (۳×۳) که بلوک شامل دو پیچش با اندازه ۳ در ۳ است به بالاترین دقت رسیده است (جدول ۱).

c. افزایش عرض بلوک، بـــا ضرایب ۱ الی ۱۲ بر روی شـــبکه‌هایی با عمق ۱۶، ۲۲، ۲۸ و ۴۰ آزمایش شده است. در شـــبکه‌هایی بـــا عمق ۱۶ و۲۲ و ۲۸ بـــا افزایش ضریب دقت کاهش یافته است، امّا در شبکه با عمق ۴۰، با افزایش عرض با ضریب ۸، دقتی کمتر از شـــبکه ۲۲ لایه با ضریب ۸ کسب شده اســـت. در مقایسه شبکه‌های این آزمایش با شـــبکه‌های رِزنت عادی، می‌توان اشـــاره داشت به شبکه ۴۰ لایه با عرض ۴ برابر، که توانســـته است به دقتی بالاتر از رِزنـــت ۱۰۰۱ لایـــه بـــه دســـت آورد درحالی‌که میزان پارامترهای آن نیز از شبکه هزار لایه کمتر بوده است. این تغییر میزان پارامترهای شـــبکه را با ضریب مشخص‌شده افزایش می‌دهد امّا زمان آموزش و بهینه شـــدن پارامترها نسبت به شبکه‌هایی با عمق بیشتر، به دلیل امکان پردازش موازی بسیار کمتر است.

مقالـــه بعدی [۳۲]، در تلاش برای آموزش شـــبکه‌های باقیمانـــده بســـیار عمیق، از ایـــده حـــذف تصادفی برای بلوک‌های رِزنت اســـتفاده کرده اســـت کـــه در هر دور از

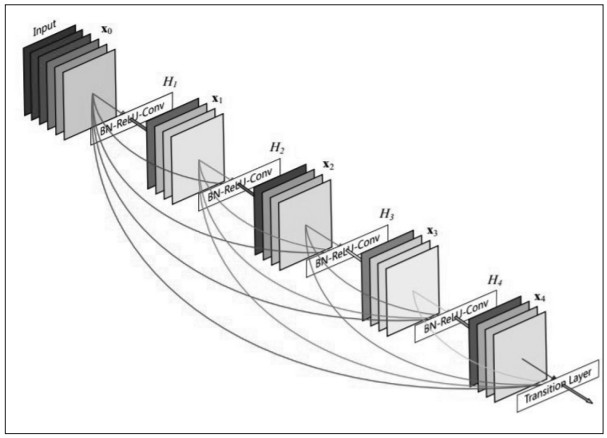


شکل ۲۷: ساختار ماژول رزنت تغییریافته در مقاله [۳۰]

جدول ۱: دقت‌های به‌دست‌آمده از سنجش انواع مختلف پیچش درون بلوک رِزنت [۳۱]

block type	depth	# params	time,s	CIFAR-10
$B(1,3,1)$	40	1.4M	85.8	6.06
$B(3,1)$	40	1.2M	67.5	5.78
$B(1,3)$	40	1.3M	72.2	6.42
$B(3,1,1)$	40	1.3M	82.2	5.86
$B(3,3)$	28	1.5M	67.5	5.73
$B(3,1,3)$	22	1.1M	59.9	5.78

56- Pre-Activation
57- Regularization
58- Wide Residual Networks



شکل ۲۸: ساختار درون یک بلوک چگال با نرخ رشد ۴ [۳۴]

آموزش، تعدادی از بلوک‌های رِزنت را به‌صورت تصادفی غیرفعال می‌کند و ورودی بلوک از میان‌بر به‌صورت بدون تغییر عبور می‌کند. البته این صرفاً مربوط به زمان آموزش است، و در زمان سنجش تمام شبکه مورداستفاده قرار می‌گیرد.

پژوهش دیگری با ارائه ساختاری به نام ResNext [۳۳]، تلاش کرده است با افزایش عرض شبکه بدون تغییر میزان پارامترها به دقت بالاتری دست یابد. در اینجا متغیر دیگری با عنوان قدرتمندی[59] مطرح می‌شود، که هر بلوک از رِزنت را تبدیل به بلوکی با تعدادی مسیر موازی می‌کند که هر یک از این مسیرها به میزان عدد قدرتمندی دارای نقشه ویژگی هستند. این مقاله با اعمال تابعی برای غیرخطی سازی در هرکدام از این مسیرها، سعی در افزایش کارایی آن‌ها دارد. این موضوع موجب افزایش دقت شبکه ارائه‌شده می‌گردد.

۸- دِنس‌نت

این شبکه [۳۴]، با ایده‌ای شبیه به رِزنت امّا متفاوت از آن ساختاری را طراحی کرده است که ویژگی بارز آن باز استفاده از ویژگی‌های ساخته‌شده در لایه‌های پیشین است. ایده این مقاله برگرفته از مشاهداتی است که در مقالات پس از معماری رِزنت در تلاش برای بهینه‌سازی شبکه‌های عمیق‌تر از ۱۰۰ یا هزار لایه دیده شده است. مقالاتی مانند [۲۹] و [۳۲]، که تغییری که در شبکه -برای آموزش آن- ایجاد می‌کنند موجب گذر و نادیده گرفته شدن برخی لایه‌ها می‌شود. این موضوع نویسندگان این مقاله را به این نتیجه رسانده است که عمق خیلی زیاد ایده‌آل نیست و احتمال افزونگی در شبکه‌های بسیار عمیق بالاست. به همین دلیل سعی شده است به کمک باز استفاده از ویژگی‌ها، شبکه‌ای با دقت بالاتر ساخته شود.

بلوک چگال ۶۰ که در این مقاله طراحی می‌شود، دارای چند مرحله عملیات نرمال‌سازی – غیرخطی سازی و پیچش است که در هر مرحله تمامی نقشه‌های ویژگی مراحل پیشین به‌عنوان ورودی در نظر گرفته می‌شوند، و در واقع دانش شبکه به‌صورت تجمعی در اختیار آن مرحله قرار می‌گیرد، بدین‌صورت شبکه در هر لایه می‌تواند ویژگی‌هایی از طیف ساده تا پیچیده را برحسب نیاز حدس بزند. یکی از پارامترهای مهم در این شبکه نرخ رشد است که در واقع همان تعداد نقشه‌های ویژگی در هر مرحله است که به موارد پیشین خود افزوده می‌شود. نرخ رشد عموماً عدد بزرگی نیست و این نکته‌ای است که موجب کاهش میزان پارامترهای این معماری نسبت به سایر معماری‌ها می‌شود. اعداد پیشنهادشده برای نرخ رشد اعدادی مابین ۱۲ و ۴۰ هستند که در مقایسه با تعداد نقشه‌های ویژگی در هر لایه از سایر شبکه‌ها عدد کوچکی است. در میان بلوک‌های چگال این شبکه لایه‌های گذار قرار دارند که شامل پیچش ۱×۱ و لایه ادغام با اندازه ۲×۲ هستند.

۹- اسکوییزنت[61]

در نوعی دیگر از مقالات این حوزه، به جای تلاش برای دستیابی به دقت بالاتر، تلاش می‌شود میزان پارامترها و حجم مدل کاهش یابد [۳۵]it is typically possible to identify multiple DNN architectures that achieve that accuracy level. With equivalent accuracy, smaller DNN architectures offer at least three advantages: (۱. برخی مزایا که برای ساخت مدل‌های کوچک ذکر شده است عبارتند از: سرعت بالاتر و سربار کمتر در آموزش مدل روی کارسازهای موازی، امکان انتقال سریع‌تر به‌روزرسانی‌ها به کامپیوترهای کاربران و امکان

59- Cardinality

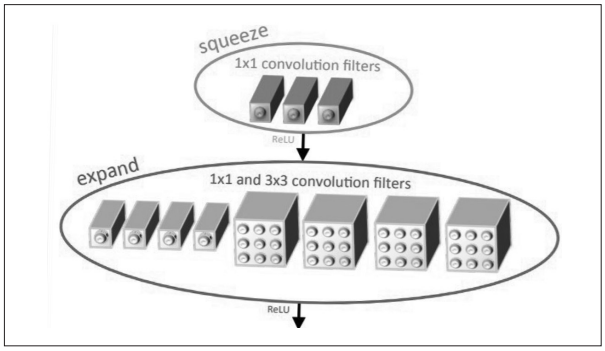
60- Dense Block
61- SqueezeNet

پیاده‌ســـازی مدل روی FPGAهایی با حجم کوچک. در این مقاله، بلوک‌های ســـازنده شبکه‌های عصبی پیچشی مانند بلوک رِزنت و بلوک اینسپشن را ریزساختار[62] شبکه‌ها، و ســـاختار کلی آن‌ها را که از چینش ریزساختارها در کنار هم تشـــکیل می‌شود کلان‌ســـاختار[63] می‌نامد. ریزساختار ارائه‌شده در این مقاله ماژول آتش٦٤ نام دارد که با چیدن آن‌ها در کنار هم، اسکوییزنت ساخته می‌شود.

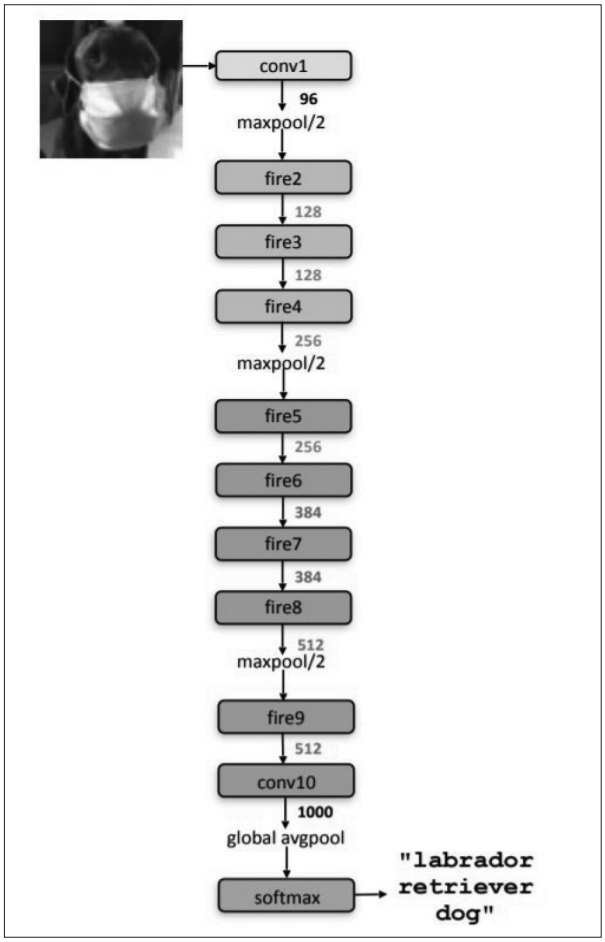
در ایـــن پژوهش چنـــد راهبرد پایه بـــرای طراحی این ماژول جدید ذکر شده است: ۱. جایگزینی پالایه‌های ۳×۳ با پالایه‌هـــای ۱×۱ برای کاهش ۹ برابری پارامترها ۲. کاهش تعداد لایه‌هایی کـــه ورودی پیچش ۳×۳ قرار می‌گیرند، به کمک لایه فشرده‌ســـازی ۳. نمونه‌برداری دیرهنگام، برای این‌که ورودی لایه‌های پیچش اطلاعات بیشـــتری را شامل باشـــند که این موضوع در افزایش دقت شبکه موثر است. این قانون که به معنای اســـتفاده کمتـــر از اندازه گام ۲ در پیچش‌ها، و اســـتفاده دیرهنگام از لایه ادغام است، موجب می‌شـــود در بخش عظیم شبکه، نقشـــه‌های ویژگی اندازه بزرگی داشته باشند. ساختار ماژول آتش (شکل ٢٦) شامل دو بخش فشردن[65] و گستردن[66] است. در بخش فشردن، با استفاده از پیچش‌های ۱×۱ تعداد لایه‌های ورودی به بخش گستردن محدود می‌شود. در بخش گستردن، با استفاده از پیچش‌های ۱×۱ و ۳×۳ ویژگی‌هایی را استخراج می‌کنند و به‌عنوان خروجی ماژول در کنـــار هم قرار می‌دهند. برای هم‌اندازه شدن خروجی ماژول، به اندازه یک درایه حاشیه برای ورودی‌های پیچش ۳×۳ در نظر گرفته می‌شود.

کلان‌ســـاختار این شبکه که در شـــکل ۲۸ قابل‌مشاهده اســـت، از چینش ماژول‌های آتش و تعداد کمی لایه ادغام شکل گرفته و در انتهای آن ادغام کلی میانگین و softmax برای دسته‌بندی استفاده شده است. پژوهشگران با طراحی و پیاده‌سازی این ساختار، که میزان پارامترهای آن تنها دو درصد الکس‌نت است، بر روی مجموعه داده ImageNet به

62- Micro Architecture
63- Macro Architecture
64- Fire Module
65- Squeeze
66- Expand



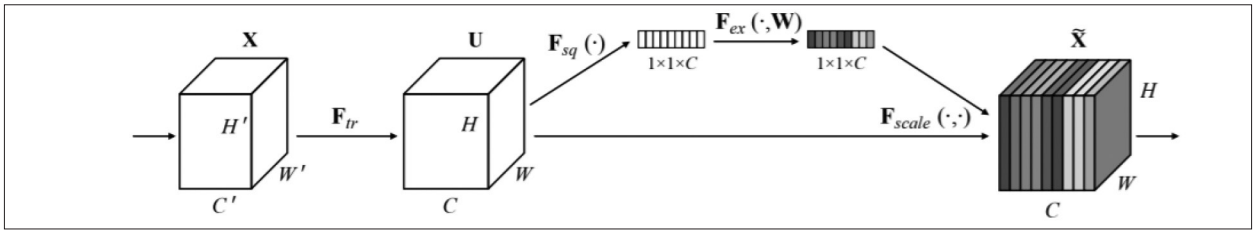
شکل ۲۹: ساختار ماژول آتش، ریزساختار سازنده اسکوییزنت [۳۵]



شکل ۳۰: ساختار اسکوییزنت [۳۵]

دقت مشابه الکس‌نت دست پیدا می‌کنند. حجم مدل ساخته شـــده حـــدود ٤/٨ مگابایت اســـت که با ایـــن وجود تلاش شده اســـت با استفاده از روش‌های فشرده‌سازی به حجم پایین‌تری دست یابند. حجم کوچکترین مدل به‌دست آمده در این پژوهش ۰/٤٧مگابایت است که به روش فشرده‌سازی عمیق[67] فشرده شده است. این روش فشرده‌سازی ترکیبی

67- Deep Compression



شکل ۳۱: عملیات فشردن و برانگیختن در ساختار SEnet [۳۶]

از هرس شـــبکه و چندی سازی[۶۸]، به همراه استفاده از کد هافمن است.

۱۰- SEnet

هـــدف از ارائه این ســـاختار، بهبود تـــوان بازنمایی[۶۹] شبکه برای دســـت‌یابی به دقت بالاتر اســـت [۳۶]. تمرکز ایـــن پژوهش برای این بهبود، تمرکز بر ایجاد ارتباط میان لایه‌های دادگان اســـت. راهکار انتخاب شده اضافه کردن ســـازوکاری برای مدل کردن ارتبـــاط پویا و خطی مابین لایه‌ها، با استفاده از اطلاعات کلی شبکه است. در این شبکه، تبدیل SE طراحی شـــده با اســـتفاده از اطلاعات شبکه، یک سازوکار دریچه‌ای است که میزان اطلاعات مورد استفاده هر لایه را کنترل می‌کند. این تبدیل قابل اعمال به بلوک‌های سایر معماری‌های شـــبکه عصبی پیچشی است. تبدیل SE شـــامل دو بخش فشردن[۷۰] و برانگیختن[۷۱] است. در مرحله فشرده‌سازی، با استفاده از ادغام کلی میانگین هر کانال از داده تبدیل به یک عدد به‌عنوان توصیف‌گر کانال[۷۲] می‌شود. این مقادیر یک شمای کلی از اطلاعات تصویر را می‌سازند. در مرحله برانگیختن، درجه‌بندی تطبیقی[۷۳] مقادیر کانال‌ها با اســـتفاده از توصیف‌گرهای مرحله قبل انجام می‌شود. درجه‌بندی تطبیقی تعیین وزن‌های کانال‌های داده است که بر اساس ارتباط میان کانال‌ها و میزان اهمیت هر لایه تعیین می‌شـــود. برای شبیه‌ســـازی این ارتباطات از دو لایه تمام متصل استفاده می‌شود که ورودی‌های آن توصیف‌گرهای کانال‌ها هستند و تعداد مقادیر خروجی و ورودی آن برابر است. تابع فعال‌سازی لایه اول ReLU و لایه دوم سیگموید است. مقادیر به‌دست آمده از این تبدیل به‌عنوان ضریبی در کانال‌های متناظر خود ضرب می‌شـــوند. شبکه طی مراحل آمـــوزش این ضرایـــب را به بهترین مقادیـــر خود نزدیک می‌سازد (شکل ۲۹).

برای سنجش میزان کارایی سازوکار طراحی شده، آن را به نسخه‌های مختلف ساختار رِزنت، اینسپشن و موارد دیگر اعمال شده اســـت که در تمامی موارد بهبود دقت را نشان داده اســـت. بهترین نتیجه به‌دست آمده SEnet-154 اســـت که اعمال ســـازوکار SE بر نمونـــه تغییریافته‌ای از ساختار ResNeXt است.

۴- خلاصه روش‌های بهبود معماری‌ها

٤-۱- افزایش عمق شبکه

افزایش عمق شـــبکه، یکی از ساده‌ترین ایده‌های مورد اســـتفاده برای افزایش دقت شبکه است که در بسیاری از معماری‌ها از این روش کمک گرفته شـــده است، خصوصا در ابتدای کار اســـتفاده از شـــبکه‌های عصبی پیچشی که توان پردازشـــی رایانه‌ها رو به افزایش بود روش بســـیار موثری به حساب می‌آمد. این روش موجب می‌شود شبکه بتواند ویژگی‌هـــای پیچیده‌تری را یاد بگیـــرد و در نتیجه دقت افزایـــش پیدا کند. البته افزایـــش عمق همواره موجب افزایش دقت نمی‌شـــود، مســـائلی که ممکن است پیش آید یکی افزایش تعداد پارامترهاست که در صورت کافی نبودن حجم دادگان می‌تواند منجر به بیش‌برازش شـــود. مشکل دیگر این است که وقتی عمق خیلی زیاد شود، حین آموزش شـــبکه و در مرحله پس‌انتشـــار، گرادیان ناشی از خطاها بســـیار کوچک می‌شوند که موجب می‌شـــود پارامترهای ابتدایی شبکه به‌روزرسانی نشوند و شبکه نتواند به خوبی

68- Quantization
69- Representational Power
70- Squeeze
71- Excitation
72- Channel Descriptor
73- Adaptive Re-Calibration

از عمق افزایش یافته استفاده کند.

٤- ٢- ایجاد افزونگی در دادگان

برای بهبود آموزش شـبکه، می‌توان تغییراتی بر روی دادگان اولیه انجام داد و دادگان بیشـتری را از روی آن‌ها سـاخت. این کار عـلاوه بر افزایش حجـم دادگان موجب افزایش تنوع می‌شـود. افزایش تنـوع دادگان کمک می‌کند شبکه با وابستگی کمتری به حالت ورودی بتواند نتایج را به درستی پیش‌بینی کند.

٤- ٣- حذف تصادفی

در ایـن روش در زمـان آموزش شـبکه، بـا احتمال مشـخص، در هر گذر برخی نورون‌هـا را فعال و برخی دیگـر را غیرفعال در نظر می‌گیریم، به همین جهت شـبکه در هر دور سـاختار متفاوتی پیـدا می‌کند، و به دلیل عدم اتکای شـبکه به تمام ویژگی‌ها برای تصمیم‌گیری، تک تک ویژگی‌های به‌دست آمده قوی‌تر و بهتر انتخاب می‌شوند. ویژگی‌هـای که می‌تـوان با تعداد نورون‌هـای کمتری بر روی آن‌ها تصمیم گرفـت، و در واقع ویژگی‌هایی انتخاب می‌شوند که وابستگی کمتری بین نورون‌ها ایجاد می‌کنند. البته این کار می‌تواند در سـطح بالاتر از نورون‌ها –مانند بلوک‌های شـبکه– هم استفاده شـود که در معماری‌های مختلف با آن مواجه خواهیم بود.

٤- ٤- تقویت شبکه

منظور از تقویت، این است که چند شبکه آموزش داده شـوند و نتیجه نهایی از تصمیم مجموع شبکه‌ها به‌دست آورده شود. ترتیب استفاده از شبکه‌ها و نحوه تعیین نتیجه می‌تواند متفاوت باشـد، برای مثال می‌شود چند شبکه را موازی آمـوزش داد و برای نتیجه از رای‌گیری اسـتفاده نمود.

٤- ٥- افزایش عرض شبکه

منظور از افزایش عرض شبکه به‌طور کلی قرار گرفتن تعداد بیشـتری از پالایه‌ها در یک لایه از شـبکه است، که موجب می‌شـود در هر لایه تعداد ویژگی‌های بیشتری را ذخیره نماییم و افزایش تنوع ویژگی‌ها می‌تواند به افزایش دقـت کمک کند. همچنین افزایش عرض پردازش موازی را امکان‌پذیر می‌سـازد و موجب می‌شود از توان پردازشی رایانه‌ها بهتر و بهینه‌تر استفاده شود. البته افزایش عرض نیز مانند برخی تکنیک‌های گفته شـده می‌تواند روش‌های مختلفی داشته باشد که ساده‌ترین آن افزایش تعداد پالایه‌ها در یک معماری موجود است، و یک مورد پرکاربرد دیگر، افزایش تعداد مسـیرها در یک سـاختار یا بلوک سازندهٔ ساختار است.

٤- ٦- پیچش ۱×۱

این روش که الهام گرفته از معماری شبکه در شبکه [۲۳] اسـت، عملیاتی است که با حفظ طول و عرض نگاشت‌های ویژگی بر روی تعداد آن‌ها موثر اسـت و دادگان را با هم ترکیـب می‌کند. این نوع پیچـش می‌تواند هم برای افزایش تعداد نگاشت‌ها و هم برای کاهش آن‌ها مورد استفاده قرار گیرد. یکی از کاربردهای آن در جایی است که ما بخواهیم خروجی مسـیرهای مختلفی از شـبکه را در کنار هم قرار دهیم و برای عملیات مشـترکی آماده کنیم که نیاز باشـد ابعاد یکسـانی داشته باشـند، در اینجا می‌توان با استفاده از پیچـش ۱×۱ بعد تعداد را با هـم برابر کرد. همچنین در برخی ساختارها که نیاز به کاهش ابعاد برای کاهش میزان پردازش هسـت، از این نوع پیچش اسـتفاده می‌شود زیرا در عین کاهش بعد، بخشـی از محتوا بطور کامل از دست نمی‌رود و ترکیبی از آن در لایه‌های دیگر نگاشـت موجود است.

٤- ٧- ادغام کلی میانگین

ایـن روش بـرای تبدیل نگاشـت‌های ویژگـی به یک عدد اسـت کـه عموما در لایه‌های آخر شـبکه اسـتفاده می‌شـود. بدین‌صورت که در لایه‌هـای آخر، پیش از لایه تصمیم‌گیری، تعداد نقشه‌های ویژگی را به تعداد دسته‌های موجود در دسته‌بندی می‌رسانیم، و سپس میانگین هرکدام

از این نقشـــه‌های ویژگی را به‌عنوان مقادیر نهایی به لایه تصمیم‌گیری می‌دهیـــم تا با روش‌هایـــی مانند softmax، دسته منتخب را مشخص نماید.

٤- ٨- استفاده از دسته‌بندهای میانی

در مرحله آموزش شـــبکه، وزن‌های شـــبکه در مسیر پس‌انتشار خطا به روزرسانی می‌شوند تا به مقادیر بهینه برای تصمیم‌گیری صحیح برســـند. با افزایش یافتن عمق شـــبکه‌ها، خطای انتشـــار یافته از انتهای شـــبکه به سمت ابتـــدای آن مقادیـــر بســـیار کوچکی می‌شـــود که موجب می‌شود برخی وزن‌ها بخصوص وزن‌های ابتدای شبکه- به روزرسانی نشوند.

یک راه‌حل برای این مشـــکل اســـتفاده از دسته‌بندهای میانی اســـت کـــه پیش از رســـیدن به انتهای شـــبکه قرار می‌گیرند و هر کدام بر اســـاس مقادیـــری که تا آن بخش از شـــبکه دیده شده تصمیم‌گیری انجام می‌دهند، در نتیجه می‌توان بر اســـاس ایـــن تصمیم‌ها نیز مقادیـــر خطایی را محاســـبه کرد که در مرحله پس انتشار از این مقادیر برای به‌روزرسانی وزن‌های شبکه استفاده می‌شوند.

٤- ٩- شکستن عمل پیچش به پیچش‌های کوچکتر

این روش که در معماری‌های مختلفی اســـتفاده شـــده اســـت، اســـتفاده از چند پالایهٔ کوچک به‌صورت متوالی و بدون لایه ادغام در بین آن‌ها، می‌تواند جایگزین پالایه‌هایی با انـــدازه بزرگ‌تر اســـت. دلایل اصلی برای اســـتفاده از پالایه‌های کوچک -برای مثـــال ۳×۳-، یکی افزایش دفعات اســـتفاده از غیرخطی‌سازی است که به شبکه کمک می‌کند توابع و ویژگی‌های پیچیده‌تری را یاد بگیرد و دومین دلیل، کاهش تعداد پارامترهاست. برای نمونه در یک لایه پیچش با پالایه‌ای با انـــدازه 7×7×C- که C تعداد کانال‌ها یا همان نقشـــه ویژگی‌های ورودی یک لایه است- تعداد پارامترها $7^2\ C^2$ است، امّا در شبکه‌ای با سه عمل پیچش 3×3 به‌عنوان جایگزین، $3(3^2C^2)$ پارامتر وجود دارد که ۸۱٪ کمتر اســـت. کاهـــش پارامترها به همگرایی ســـریع‌تر شـــبکه و کاهش بیش‌برازش کمک می‌کند.

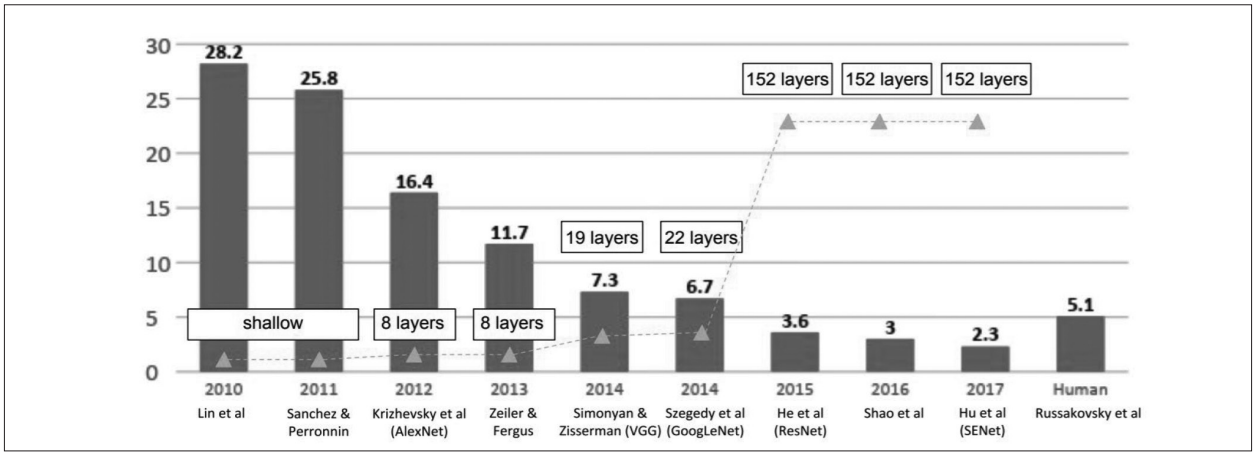
موجب کاهش میزان پردازش در شبکه‌ها و البته موجب افزایش عمـــق آن‌ها هم می‌شـــود. این شکســـت می‌تواند شکســـت متقارن یا نامتقارن باشـــد، در بند قبل یک مثال متقارن ذکر شد. در شکست نامتقارن می‌شود پیچش n×n را به دو پیچش n×1 و 1×n شکست. برای مثال به‌جای یک فیلتر ۳×۳ با ۹ پارامتر، از دو پیچش 3×1 و 1×3 -هر کدام شامل ۳ پارامتر- استفاده می‌شود که با ۳۳ درصد کاهش تعداد پارامترها همراه اســـت. البته در مقاله [۲٦] ذکر شده است که این کار در لایه‌های ابتدایی شبکه عملکرد خوبی را نشان نمی‌دهد و در لایه‌هایی با نگاشت‌های ویژگی به ابعاد ۱۲ الی ۲۰ نتیجه خوبی را به همراه دارد.

٤- ۱۰- استفاده از اتصال‌های میانبر

یکی دیگر از راه‌حل‌های ارائه شـــده برای رفع مشـــکل محو گرادیان در شـــبکه‌های عمیق، استفاده از اتصال‌های میانبر اســـت که ســـاختارهای مبتنی بر باقیمانده را ایجاد می‌کنند. اگر خروجی شـــبکه در حالـــت معمول، تابع H(x) باشد، ما با افزودن میان‌بر به بلوک سازندهٔ معماری، مقدار x یا همان ورودی بلوک را به خروجی آن اضافه می‌کنیم، در نتیجه شـــبکه باید بتواند به کمک پارامترهای خود، تابع H(x)-x را بسازد که همان بازنمایی باقیمانده است.

۵- مقایسه و جمع‌بندی

در این پژوهش مروری داشـــتیم بر ســـاختار شـــبکه عصبی پیچشـــی و لایه‌های اصلی تشکیل‌دهنده آن، سپس معماری‌هـــای مهم شـــبکه عصبی پیچشـــی کـــه تغییرات جدی‌تری در این زمینه ایجاد کرده بودند، بررسی و نکات اصلی تفاوت آن‌ها ذکر شـــد. تلاش‌های انجام شـــده برای افزایش دقت عمدتا در راستای افزایش عمق، تغییر ساختار در جهت بهبود پس‌انتشـــار خطا و ایجاد انعطاف در شبکه بوده اســـت. برخی پژوهشـــگران نیز با هدف کاهش تعداد پارامترهـــا تلاش در تغییر ســـاختار برای بازاســـتفاده از ویژگی‌ها دارند. این حوزه، یکی از حوزه‌ها فعال پژوهشی

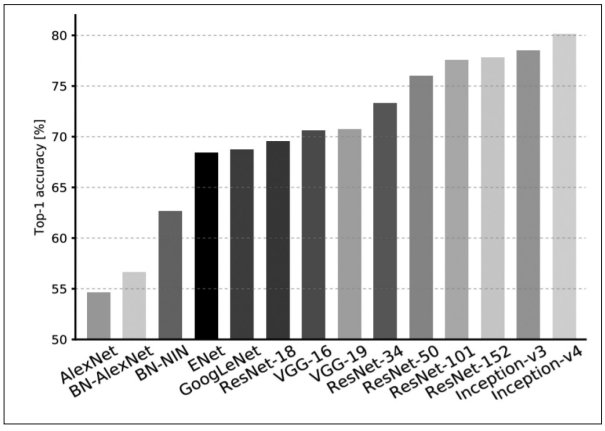


شکل ۳۲: خطای تشخیص و تعداد لایه‌های معماری‌های برنده چالش دسته‌بندی ImageNet [۳۷]

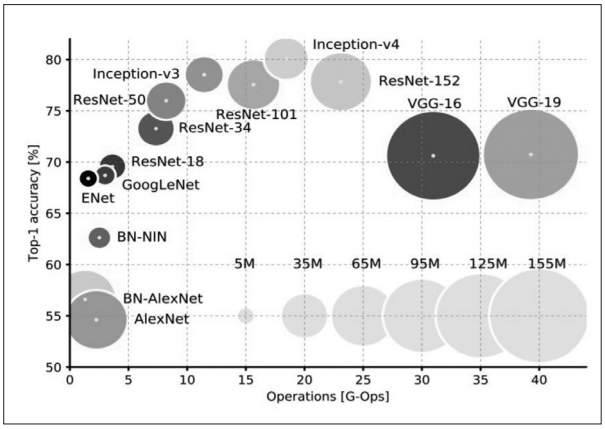
است که هر روزه نوآوری‌های تازه‌ای را در آن شاهد هستیم. ارائه ساختارهای جدید، بهبود ساختارهای موجود، کاهش تعداد پارامترها و موارد دیگر می‌توانند حوزه‌های فعالیت برای پژوهشگران علاقه‌مند باشد.

به‌عنوان بخشی از جمع‌بندی این مقاله، مقایسه‌ای بین معماری‌های معرفی‌شده ارائه می‌شود که برای این هدف، می‌توان به میزان کارایی این معماری‌ها برای چالش دسته‌بندی ImageNet [۱] اشاره نمود که تا سال ۲۰۱۷ جریان داشته است و هر ساله برای حل این چالش معماری‌های جدید شکل گرفته و معرفی شده‌اند. همان‌طور که در شکل ۳۲ قابل مشاهده است، برندهٔ این چالش از سال ۲۰۱۲ و پس از آن به ترتیب الکس‌نت، VGGnet، ZFnet و گوگل‌نت (به ترتیب در چالش دسته‌بندی و دیگری در چالش تشخیص محل شیء[۷۴]) و رزنت بوده است، برنده سال ۲۰۱۶ نیز با چینش چند نوع از معماری‌های پیشین در کنار هم و تجمیع نتایج به این میزان خطا دست یافته است. خطای گزارش‌شده در شکل ۳۰ مربوط به چالش تشخیص دسته صحیح از میان ۵ حدس اول است. این چالش در سال ۲۰۱۷ با ارائه ساختار SEnet و دست‌یابی به خطای ۲,۳ درصد به پایان رسیده و دیگر ادامه پیدا نکرده است.

همچنین برای مقایسه می‌توان به پژوهش [۳۸] اشاره نمود که معماری‌ها را از دو منظر بررسی کرده است. یکی دقت اولین تشخیص صحیح، بر روی مجموعه داده



شکل ۳۳: مقایسه دقت اولین تشخیص صحیح معماری‌های مختلف بر روی مجموعه داده ImageNet [۳۸]



شکل ۳۴: مقایسه حجم پردازش و دقت اولین تشخیص صحیح معماری‌های مختلف بر روی مجموعه داده ImageNet [۳۸]

ImageNet که در شکل ۳۱ نتایج آن آمده است و دومی میزان پردازش موردنیاز برای آموزش هریک از ساختارها در کنار دقت تشخیص است (شکل ۳۲).

به‌عنوان جمع‌بندی معماری‌های مختلف، خلاصه‌ای از

74- Localization

جدول ۲: مقایسه معماری‌های مختلف شبکه‌های عصبی پیچشی و نوآوری‌های آن‌ها

نام معماری		خلاصه نوآوری‌ها نام ساختار	ساختار برتر دسته				سال انتشار مقالات
			عمق شبکه	تعداد پارامترها	خطا روی ImageNet		
لینت [۱۱] [۱۷]		استفاده از پیچش در لایه‌های شبکه عصبی، برای کاربرد دسته‌بندی تصاویر	لینت-۵	۷	۶۰۰۰۰	-	۱۹۹۸-۱۹۹۰
الکس‌نت [۲۰]		افزایش تعداد لایه‌ها و نقشه‌های ویژگی در هر لایه (افزایش دقت)، استفاده از ReLU به‌عنوان تابع فعال‌سازی (کاهش میزان پردازش و مشکل اشباع گرادیان)، استفاده از تکنیک حذف تصادفی (بهبود ویژگی‌های آموخته شده در شبکه)	الکس‌نت	۸	۶۲ میلیون	* 16.4%	۲۰۱۲
ZFnet [۲۲]		تغییر اندازه گام پیچش از ۴ به ۲ و اندازه فیلتر از ۱۱ به ۷ در اولین پیچش برای به دست آمدن ویژگی‌های دقیق‌تر و بهتر (افزایش دقت)	ZFnet با ۵۱۲، ۱۰۲۴ و ۵۱۲ فیلتر در لایه ۳ و ۴ و ۵	۸	ذکر نشده	* 16.1%	۲۰۱۴
شبکه در شبکه [۲۳]		استفاده از یک شبکه عصبی کوچک به جای عمل پیچش، بر روی همان ناحیه دید پیچش (در نظر گرفتن روابط بین لایه‌ای دادگان)	شبکه در شبکه	۴	ذکر نشده	-	۲۰۱۳
VGGnet [۲۶]		تبدیل پیچش‌های با اندازه بزرگ به پیچش‌های ۳×۳ متوالی (افزایش عمق، کاهش پارامترها و افزایش دقت)	VGG-E	۱۹	۱۴۴ میلیون	7.1%	۲۰۱۴
گوگل‌نت (اینسپشن) [۲۴][۲۷] [۲۸]		افزایش عرض بلوک‌ها برای استفاده بهینه از منابع، ارائه مسیرهای مختلف با اندازه‌های مختلف پیچش داخل بلوک (آموزش ویژگی‌ها با اندازه‌های متنوع‌تر)، تفکیک عمل پیچش معمول به پیچش‌های ساده‌تر و با پردازش کمتر	اینسپشن-رزنت۲	۱۶۴ [۳۹]	۵۵ میلیون [۴۰]	3.7%	۲۰۱۶ -۲۰۱۴
رزنت [۲۵]		استفاده از اتصال میانبر در بلوک‌های شبکه (ایجاد امکان افزایش عمق، جریان یافتن بهتر گرادیان‌ها در پس‌انتشار)	رزنت۱۵۲	۱۵۲	۶۰ میلیون [۴۱]	3.6%	۲۰۱۵
ایده های بهبود رزنت	هایوی نت [۲۹]	استفاده از سازوکار دریچه‌ای برای کنترل میزان عبور داده از اتصال میانبر و بلوک پیچش (بهبود دقت، امکان آموزش شبکه‌های عمیق‌تر)	هایوی۲	۱۹	۲،۳ میلیون	-	۲۰۱۵
	پیش فعال‌سازی رزنت [۳۰]	جابجایی تبدیلات اعمال شده داخل بلوک‌های رزنت برای بهبود جریان گرادیان‌ها (بهبود دقت)	رزنت۲۰۰ پیش‌فعال‌شده۱	۲۰۰	۶۴،۲ میلیون [۳۱]	4.8%	۲۰۱۶
	شبکه با عمق تصادفی [۳۲]	غیرفعال کردن تصادفی بلوک‌هایی از شبکه در مرحله آموزش (دست یافتن به شبکه‌ای با ویژگی‌های بهتر)	ندارد	۱۲۰۲	ذکر نشده	4.9%	۲۰۱۶
	رزنت عریض [۳۱]	بررسی چند پارامتر مهم بلوک رزنت مانند تعداد ویژگی‌ها (عرض شبکه) و تغییر آن (بهبود دقت)	WRN-۵۰-۲	۵۰	۶۸،۹ میلیون	6.03%	۲۰۱۶
	ResNeXt [۳۳]	ایجاد مسیرهای موازی داخل بلوک رزنت (بهبود دقت)	ResNeXt ۱۰۱-۴×۶۴	۱۰۱	۸۳،۶ میلیون	4.4%	۲۰۱۷
دِنس‌نت [۳۴]		بازاستفاده از ویژگی‌ها با استفاده از اتصال‌های میانبر میان تمامی لایه‌ها در درون بلوک‌ها (کاهش عرض و بهبود دقت)	دنس‌نت-۲۶۴ (نرخ رشد ۳۲)	۲۶۴	۴-۳۳ میلیون	5.29%	۲۰۱۷
اسکوییزنت [۳۵]		صرفا استفاده از پیچش‌های ۱×۱ و ۳×۳ در ماژول‌ها (کاهش حجم بسیار زیاد)	اسکوییزنت + اتصال ساده	۶۹ [۴۱]	۱،۲ میلیون [۴۱]	17.5%	۲۰۱۶
SEnet [۳۶]		ایجاد ارتباط میان لایه‌های اطلاعات و در واقع وزن‌دهی به هر لایه به کمک اطلاعات خود لایه‌ها (بهبود دقت)	SEnet-154	۱۵۴	۱۴۶ میلیون [۴۰]	* 3.58%	۲۰۱۷

1- ResNet200 (pre-act)

مشخصات شـــبکه‌ها و معماری برتر ذکر شده در مقالات آن دســـته در جدول ۲ آورده شده است. در این جدول، در بخش ساختار برتر، صرفا ساختارهای تکی در نظر گرفته شده‌اســـت و مواردی که با گروه کردن چند ســـاختار به خطای کمتری می‌رسند در جدول نیامده است. خطای اعلام شـــده در این جدول، مربوط به ۵ تشخیص است. همچنین مواردی که با * مشـــخص شده‌اند، بر روی مجموعه داده آزمایشی گزارش شده‌اند و در سایر موارد خطای گزارش شده بر روی داده اعتبارسنجی[۷۵] است.

مراجع

[1] "ImageNet." [Online]. Available: http://www.image-net.org/. [Accessed: 03-Aug-2019].

[2] Wang M., Deng W., "Deep Face Recognition: {A} Survey," CoRR, vol. abs/1804.0, 2018.

[3] Wang T., Wu D. J., Coates A., Ng A. Y., "End-to-end text recognition with convolutional neural networks," in Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012), 2012, pp. 3304–3308.

[4] Yamashita R., Nishio M., Do R. K. G., Togashi K., "Convolutional neural networks: an overview and application in radiology," Insights Imaging, vol. 9, no. 4, pp. 611–629, 2018.

[5] Ekici S., Jawzal H., "Breast cancer diagnosis using thermography and convolutional neural networks," Med. Hypotheses, vol. 137, p. 109542, 2020.

[6] Schwendicke F., Golla T., Dreher M., Krois J., "Convolutional neural networks for dental image diagnostics: A scoping review," J. Dent., vol. 91, p. 103226, 2019.

[7] Gruszczyński W., Puniach E., Ćwiąkała P., Matwij W., "Application of convolutional neural networks for low vegetation filtering from data acquired by UAVs," ISPRS J. Photogramm. Remote Sens., vol. 158, pp. 1–10, 2019.

[8] Brahimi M., Arsenovic M., Laraba S., Sladojevic S., Boukhalfa K., Moussaoui A., "Deep Learning for Plant Diseases: Detection and Saliency Map Visualisation," in Human and Machine Learning: Visible, Explainable, Trustworthy and Transparent, Zhou J., Chen F., Eds. Cham: Springer International Publishing, 2018, pp. 93–117.

[9] Brodrick P. G., Davies A. B., Asner G. P., "Uncovering Ecological Patterns with Convolutional Neural Networks," Trends Ecol. Evol., vol. 34, no. 8, pp. 734–745, 2019.

[10] Silver D.,Huang A.,Maddison Ch. J.,Guez A.,Sifre L., van den Driessche G., Schrittwieser J., Antonoglou I., Panneershelvam V., Lanctot M., Dieleman S., Grewe D., Nham J., Kalchbrenner N., Sutskever I., Lillicrap T., Leach M., Kavukcuoglu K., Graepel Th., Hassabis D.,, "Mastering the game of Go with deep neural networks and tree search," Nature, vol. 529, no. 7587, pp. 484–489, 2016.

[11] LeCun, Yann, Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., Jackel, L. D., "Handwritten Digit Recognition with a Back-Propagation Network," in Advances in Neural Information Processing Systems 2, D. S. Touretzky, Ed. Morgan-Kaufmann, 1990, pp. 396–404.

[12] HUBEL, D. H., WIESEL, T. N., "Receptive fields of single neurones in the cat's striate cortex.," J. Physiol., vol. 148, no. 3, pp. 574–91, Oct. 1959.

[13] "Plato's Cave: Hubel & Wiesel." [Online]. Available: http://cns-alumni.bu.edu/~slehar/webstuff/pcave/hubel.html. [Accessed: 12-Sep-2019].

[14] "CS231n Convolutional Neural Networks for Visual Recognition." [Online]. Available: http://cs231n.github.io/convolutional-networks/. [Accessed: 12-Sep-2019].

[15] Lecun, Y., Bengio, Y., Hinton, G., "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[16] "A Beginner's Guide To Understanding Convolutional Neural Networks – Adit Deshpande – Engineering at Forward | UCLA CS '19." [Online]. Available: https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/. [Accessed: 12-Sep-2019].

[17] LeCun, Y., LeCun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., Drucker, H., Guyon, I., Müller, U., Säckinger, E., Simard, P., Vapnik, V., "Comparison of Learning Algorithms for Handwritten Digit Recognition," Int. Conf. Artif. NEURAL NETWORKS, pp. 53--60, 1995.

[18] Lecun, Y., "Gradient-Based Learning Applied to Document Recognition," proc. IEEE, 1998.

[19] "The EMNIST Dataset | NIST." [Online]. Available: https://www.nist.gov/node/1298471/emnist-dataset. [Accessed: 09-Aug-2019].

[20] Krizhevsky, A., Sutskever, I., Hinton, G. E., "Imagenet classification with deep convolutional neural networks," Adv. Neural Inf. Process. Syst., p. 2012.

[21] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. R., "Improving neural networks by preventing co-adaptation of feature detectors," Jul. 2012.

[22] Zeiler, M. D., Fergus, R., "Visualizing and Understanding Convolutional Networks," Springer, Cham, 2014, pp. 818–833.

[23] Lin, M., Chen, Q., Yan, S., "Network In Network," CoRR, vol. abs/1312.4, pp. 1–10, 2013.

[24] Simonyan, K., Zisserman, A., "Very Deep Convolutional Networks for Large-Scale Image Recognition," pp. 1–14, 2014.

[25] He, K., Zhang, X., Ren, S., Sun, J., "Deep Residual Learning for Image Recognition," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[26] Simonyan, K., Zisserman, A., "Very Deep Convolutional Networks for Large-Scale Image Recognition," pp. 1–14, 2014.

[27] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna,

75- Validation

tion (CVPR), 2017, pp. 2261–2269.
[35] Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., Keutzer, K., Han, S., Dally, W. J., “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size,” CoRR, vol. abs/1602.0, pp. 1–13, 2016.
[36] Hu, J., Shen, L., Albanie, S., Sun, G., Wu, E., “Squeeze-and-Excitation Networks,” CoRR, vol. abs/1709.0, pp. 7132–7141, Sep. 2017.
[37] Li, F.-F., Johnson, J., Yeung, S., “Lecture 9: CNN Architectures,” 2019. [Online]. Available: http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture09.pdf. [Accessed: 03-Aug-2019].
[38] Canziani, A., Paszke, A., Culurciello, E., “An Analysis of Deep Neural Network Models for Practical Applications,” CoRR, vol. abs/1605.0, 2016.
[39] “Pretrained Inception-ResNet-v2 convolutional neural network - MATLAB inceptionresnetv2.” [Online]. Available: https://www.mathworks.com/help/deeplearning/ref/inceptionresnetv2.html. [Accessed: 03-Jan-2020].
[40] “State-of-the-art table for Image Classification on ImageNet.” [Online]. Available: https://paperswithcode.com/sota/image-classification-on-imagenet. [Accessed: 03-Jan-2020].
[41] “Image - Wolfram Neural Net Repository.” [Online]. Available: https://resources.wolframcloud.com/NeuralNetRepository/inputdomain/Image/. [Accessed: 03-Jan-2020].
Z., “Rethinking the Inception Architecture for Computer Vision,” in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2818–2826.
[28] Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A. A., “Inception-v4, inception-ResNet and the impact of residual connections on learning,” Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence. AAAI Press, pp. 4278–4284, 2017.
[29] Srivastava, R. K., Greff, K., Schmidhuber, J., “Highway Networks,” CoRR, vol. abs/1505.0, May 2015.
[30] He, K., Zhang, X., Ren, S., Sun, J., “Identity Mappings in Deep Residual Networks,” CoRR, vol. abs/1603.0, pp. 630–645, 2016.
[31] Zagoruyko, S., Komodakis, N., “Wide Residual Networks,” CoRR, vol. abs/1605.0, 2016.
[32] Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K. Q., “Deep Networks with Stochastic Depth,” in Computer Vision -- ECCV 2016, 2016, pp. 646–661.
[33] Xie, S., Girshick, R., Dollar, P., Tu, Z., He, K., “Aggregated Residual Transformations for Deep Neural Networks,” in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 5987–5995.
[34] Huang, G., Liu, Z., Maaten, L. van der, Weinberger, K. Q., “Densely Connected Convolutional Networks,” in 2017 IEEE Conference on Computer Vision and Pattern Recogni-