نسخه‌ی اصلاح شده‌ی مقاله‌ی زیر

کاربرد بسط تیلور در کاهش حجم شبکه‌های عصبی پیچشی برای طبقه‌بندی نقاشی‌های سبک امپرسیونیسم و مینیاتور

محمود امین‌طوسی

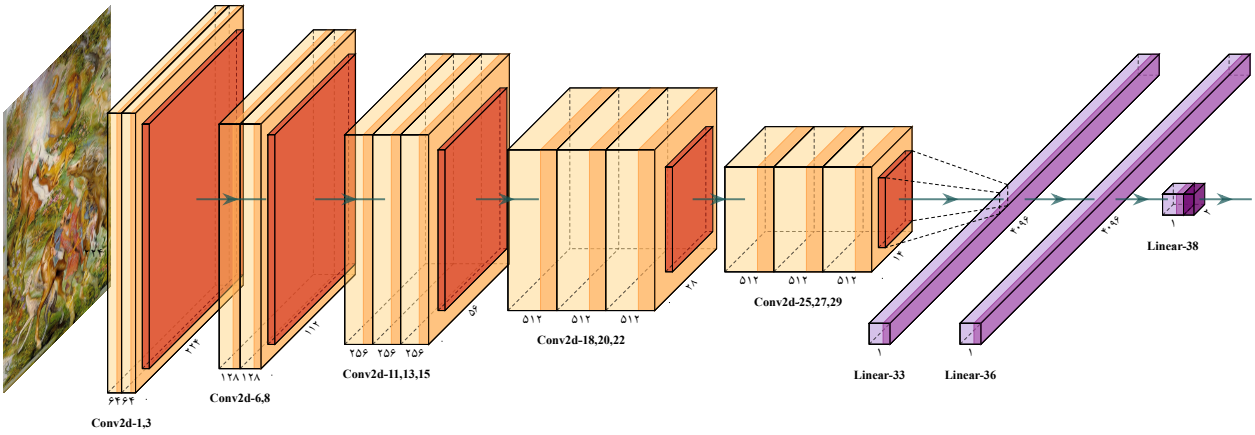
چکیده. بسط تیلور یکی از روش‌های تقریب توابعی است که از هر مرتبه‌ای مشتق‌پذیر هستند. روال اصلی یادگیری در شبکه‌های عصبی، مبتنی بر مشتق‌گیری از تابع هدف و استفاده از گرادیان کاهشی برای نیل به پاسخ بهینه است. شبکه‌های عصبی پیچشی از مهمترین ابزار حوزه یادگیری عمیق هستند. عمده‌ی این شبکه‌ها متضمن مدل‌هایی با اندازه‌های بزرگ بوده و کاهش حجم این مدل‌ها از موضوعات تحقیقاتی به‌روز می‌باشد. شیوه‌ی اصلی روش‌های کاهش حجم مدل‌ها، هرس کردن اتصالات زائد شبکه‌های عصبی است، که عموماً مبتنی بر اندازه‌ی وزن اتصالات می‌باشند. از جمله‌ی این شیوه‌ها، استفاده از بسط تیلور تابع هدف در محاسبه‌ی اولویت اتصالات، برای حذف آنها از شبکه است. در این نوشتار، این شیوه به صورت مبسوط مورد بررسی قرار گرفته و کاربرد جدیدی از آن در تفکیک تابلوهای نقاشی با سبک‌های امپرسیونیسم (برداشت‌گرایی) و مینیاتور (خُردنگارگری) ارائه شده است. نتایج آزمایش‌ها نشان داده است که با روش مبتنی بر بسط تیلور می‌توان ۸۳ درصد از اتصالات شبکه را انتخاب و حذف نمود، بدون آنکه دقت مدل در این کاربرد خاص کاهش پیدا کند.

۱. مقدمه

مدل‌های یادگیری عمیق[۱] در سالیان اخیر کارایی فوق‌العاده‌ای در شناسایی اشیاء به نمایش گذاشته‌اند [۱]. روشهای RCNN [۲]، Mask R-CNN [۳] و YOLO [۴] از جمله تحقیقات پر ارجاع و کارا در این حوزه هستند. شبکه‌های عصبی پیچشی[۲] یکی از ابزارهای اصلی مورد استفاده در این زمینه است که در این تحقیقات به‌کار گرفته شده‌اند. این شبکه‌ها را می‌توان توسیعی از شبکه‌های عصبی مرسوم از نوع پرسپترون چند لایه[۳] دانست که عمل پیچش[۴] یا فیلتر کردن تصاویر را برای استخراج ویژگی‌های مفید انجام می‌دهند و معمولاً متضمن هزاران نورون و میلیونها اتصال هستند. داشتن مجموعه دادگان بزرگ از الزامات مدلهای پیچیده‌ی جدید این حوزه است. این مجموعه داده‌ها یا به صورت مستقیم در خود مدل استفاده می‌شود و یا در ایجاد معماری‌هایی مانند VGG [۵]، ResNet [۶]، AlexNet [۷] و DarkNet[۵] [۴] به‌کار برده شده‌اند، که مبنای بیشتر مدل‌های مرتبط است. هر یک از این مدل‌ها تفاوت‌هایی از منظر تعداد نورون‌ها، نحوه‌ی اتصالات و نوع لایه‌های شبکه با هم دارند. اندازه این مدل‌ها معمولاً بسیار بزرگ بوده و شامل میلیون‌ها پارامتر هستند. شکل ۱ ساختار مدل VGG16 را نشان می‌دهد که بالغ بر ۱۳۰ میلیون پارامتر دارد. در روال یادگیری یک شبکه عصبی، در واقع وزن اتصالات هستند که باید به نحوی تنظیم شوند که عمل خواسته شده انجام شود. حجم یک مدل متناسب با تعداد نورون‌ها و اتصالات آن می‌باشد که جزء پارامترهای مدل می‌باشند.

عبارات و کلمات کلیدی. شبکه‌های عصبی پیچشی، بسط تیلور، یادگیری عمیق، هرس شبکه، طبقه‌بندی تصویر.
محمود امین‌طوسی
تاریخ دریافت: ۱۳yy/xx/xx تاریخ پذیرش: ۱۳yy/xx/xx .

[1]Deep Learning [2]Convolutional Neural Networks (CNNs/ConvNets) [3]Multi Layer Perceptron [4]Convolution
[5]https://pjreddie.com/darknet/



شکل ۱: معماری مدل VGG16 [۵] با بیش از ۱۳۰ میلیون پارامتر (وزن). ورودی شبکه، یک تصویر و خروجی آن مشخص‌کننده‌ی طبقه‌ی تصویر ورودی است.

برای آموزش چنین مدل‌هایی، حجم زیاد داده و توان پردازشی بالا مورد نیاز است. کاهش حجم این مدل‌ها که از طریق حذف تعدادی از نورون‌ها و اتصالات آنها یا حذف برخی از اتصالات انجام می‌شود، از موضوعات مورد علاقه محققین یادگیری عمیق است. هرس و کاهش اندازه‌ی مدل می‌تواند حجم یک مدل اولیه‌ی مثلاً ۵۰۰ مگابایتی را به ۵۰ مگابایت کاهش دهد، که برای استفاده در یک برنامه‌ی اندرویدی مناسب‌تر است. روش‌های مختلفی برای کاهش تعداد نورون‌ها یا تعداد اتصالات شبکه‌ها ارائه شده است. اولین بار ایده دراپ‌اوت[۶] توسط هینتون[۷] و همکاران [۸] برای شبکه‌های عصبی مطرح شد؛ در این شیوه نورون‌هایی از شبکه به صورت تصادفی حذف می‌شوند. گرچه ایده‌ی اصلی ساده است اما در عمل کارایی بسیار خوبی داشته و عموم مدل‌های جدید، متضمن چندین لایه دراپ‌اوت هستند. ایده‌ی دراپ‌اوتِ هینتون توسط وان [۸] و همکاران [۹] به حذف تصادفی وزن‌ها تعمیم داده شد و بعداً کارایی این شیوه بر روی شبکه‌های پیچشی مورد بررسی قرار گرفت [۱۰]. شیوه‌ی کلی دیگری که در هرس وزن‌های شبکه‌ها استفاده می‌شود، منظم‌سازی[۹] است که توسط محققین مختلف در قالب یک مسئله‌ی بهینه‌سازی تُنُک[۱۰] فرموله و حل شده است [۱۱، ۱۲، ۱۳، ۱۴]؛ تابع هدف مسئله‌ی بهینه‌سازی مدنظر در بخش ۳ آمده است. در این شیوه‌ها عموماً نُرم صفر وزن‌های اتصالات شبکه به عنوان جمله‌ی منظم‌ساز، به تابع هدف مسئله افزوده می‌شود. نُرم صفر یک بردار، تعداد عناصر غیرصفر آن است. وزن‌های با مقادیر کوچکِ نزدیک به صفر، هرس شده و حجم مدل کاهش می‌یابد. مولچانوف[۱۱] و همکاران از پژوهشگران شرکت انوی‌دیا[۱۲] یک شیوه‌ی مبتنی بر استفاده از بسط تیلور تابع هدف ارائه و برتری آن را نسبت به چندین روش مرسوم در این حوزه نشان داده‌اند [۱۵].

هدف این نوشتار، مقایسه‌ی این روش‌ها نیست؛ بلکه هدف اصلی، بیان چگونگی روش کار مولچانوف و همکاران [۱۵] است که در آن از بسط تیلور برای هرس اتصالات شبکه استفاده شده است. در مقاله‌ی مذکور، شیوه‌ی مورد استفاده به اختصار بیان شده است؛ در این نوشتار این روش به تفصیل بیان شده و در ضمنِ یک کاربرد جدید، کارایی آن نشان داده شده است. به عنوان کاربردی جدید، مسئله‌ی تفکیک دو سبک نقاشی امپرسیونیسم[۱۳] (برداشت‌گرایی) و مینیاتورهای ایرانی با سبک محمود فرشچیان در نظر گرفته شده است که در بخش ۴ بیان می‌شود.

---

[۷] جفری هینتون (Geoffrey Hinton) روانشناس شناختی، دانشمند علوم کامپیوتر و یکی از افرادی است که از آنها به عنوان پدران یادگیری عمیق یاد می‌شود.

[6]Dropout [8]L. Wan [9]Regularization [10]Sparse Optimization [11]P. Molchanov [12]Nvidia [13]Impersionism

از آنجا که برای بیان مسئله، یک نگرش مقدماتی بر شبکه‌های عصبی پیچشی لازم است، در بخش ۲، به مرور کوتاهی بر شبکه‌های عصبی پرداخته شده است. مدل ساده پرسپترون، که سنگ بنای شبکه‌های عصبی است، نحوه‌ی آموزش آن و مدل‌های شبکه‌های عصبی پیچشی در این بخش بیان شده‌اند. در بخش ۳، به معضل حذف اتصالات اضافی در شبکه‌های عصبی پیچشی، بسط تیلور و چگونگی استفاده از آن در روال حذف اتصالات زائد پرداخته شده است. در بخش ۴، مسئله‌ی تفکیک سبک‌های نقاشی به عنوان کاربردی جدید از شبکه‌های عصبی پیچشی عنوان شده و نتایج استفاده از شیوه‌ی مبتنی بر بسط تیلور برای کاهش حجم مدل مورد استفاده ذکر شده است. نتایج آزمایش‌های انجام شده نشان داده است که با شیوه‌ی مورد بحث، می‌توان در عین حفظ دقت مدل اصلی، ۸۳ درصد از اتصالات شبکه را حذف کرد. آخرین بخش به جمع‌بندی اختصاص یافته است.

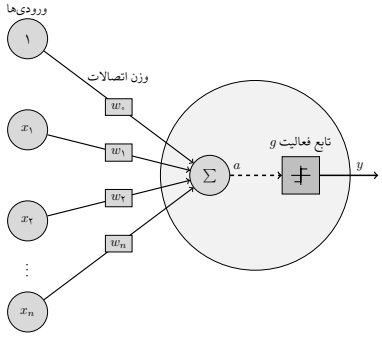
۲. مروری بر شبکه‌های عصبی

مدل‌های شبکه‌های عصبی مصنوعی، سال‌ها به امید ایجاد عملکردی مشابه مغز انسان در موضوعاتی نظیر تشخیص صحبت و تصویر در شاخه‌های هوش مصنوعی مورد مطالعه قرار گرفته‌اند. ساختار این مدل‌ها که الهام‌گرفته از شبکه‌های عصبی زیستی است، متشکل از تعدادی عنصر محاسباتی (نورون‌ها، گره‌ها، نودها) است که از طریق وزن‌هایی تطبیقی[۱۴] به هم متصل شده و به موازات یکدیگر کار می‌کنند. ساده‌ترین ساختار گره، معمولاً به صورت غیرخطی است که در آن، هر کدام از ورودی‌ها در وزن خاصی ضرب شده و حاصل‌جمع آنها پس از عبور از یک تابع، خروجی را به دست می‌آورد (شکل ۲). هر شبکه‌ی عصبی، علاوه بر معماری (توپولوژی) شبکه و مشخصات گره یا نود (یعنی نوع تابع آن، موسوم به تابع فعالیت[۱۵])، دارای یک روال یادگیری یا آموزشی نیز می‌باشد. عمل یادگیری[۱۶] در شبکه‌های عصبی، به معنی تنظیم درست وزن‌ها است به گونه‌ای که شبکه با ورود داده‌های مشخص، پاسخ مورد انتظار را تولید کند. روش‌های طراحی شبکه و قواعد آموزش آن، موضوع بسیاری از تحقیقات گذشته و اکنون است. با دسترسی به داده‌های آموزشی حجیم و افزایش تعداد نودها و اتصالات، تنظیم وزن اتصالات بسیار زمان‌بر و نیازمند توان پردازشی زیاد است. ظهور کارت‌های گرافیکی قدرتمند در کنار ابداع روش‌های بهتر بهینه‌سازی و انواع جدیدتر توابع فعالیت، باعث اوج‌گیری مجدد شبکه‌های عصبی در قالب یادگیری عمیق[۱۷] شده است.

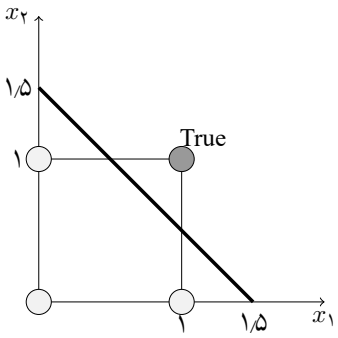
یکی از ابزارهای حوزه یادگیری عمیق، شبکه‌های عصبی پیچشی[۱۸] است که با دقت زیاد قادر به تفکیک نمونه‌ها در قالب دسته‌های از پیش مشخص است. مثال مشهوری که امروزه در حوزه یادگیری عمیق استفاده می‌شود، دسته‌بندی تصاویر سگ و گربه است. با استفاده از صدها تصویر متفاوت سگ و گربه، یک شبکه عصبی پیچشی آموزش می‌بینند تا مشخص کند که یک تصویر جدید (سگ یا گربه) متعلق به کدام دسته است. روال یادگیری در بسیاری از انواع شبکه‌ها از جمله شبکه‌های عصبی پیچشی، بر پایه‌ی مدل پرسپترون[۱۹] است. در ادامه، مدل پرسپترون و شبکه‌های عصبی پیچشی به صورت مختصر بیان می‌شوند.

۱.۲. مدل پرسپترون. روال اصلی یادگیری در شبکه‌های پرسپترونی مبتنی بر کمینه کردن خطای شبکه است. اگر هدف شبکه، طبقه‌بندی باشد، تابع هدف می‌تواند کاهش تعداد نمونه‌هایی باشد که به صورت نادرست دسته‌بندی شده‌اند[۲۰]. فرض کنید که مسئله‌ی موردنظر، طبقه‌بندی داده‌ها به دو کلاس است که برچسب دو کلاس (طبقه) به ترتیب صفر و یک درنظر گرفته شده و خروجی شبکه، صفر یا یک است. اگر y_i[۲۱] برچسب درست طبقه‌ی نمونه‌ی iام

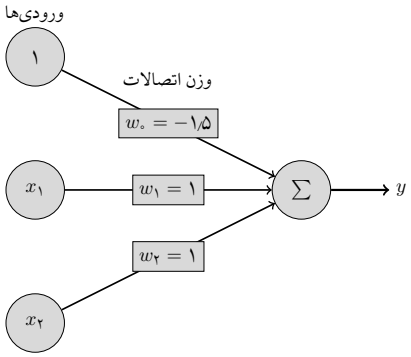
[14] Adaptive Weights [15] Activation Functions [16] Training or Learning Rule [17] Deep Learning [18] Convolutional Neural Networks
[19] Perceptron [20] Miss-Classification Error [21] Target



شکل ۲: مدل پرسپترون



شکل ۳: ترکیب عطفی x_1 و x_2. بالای خط پررنگ، ناحیه‌ای است که $x_1 \wedge x_2$، True است.



شکل ۴: مدل پرسپترون برای ترکیب عطفی x_1 و x_2؛ نمایش‌دهنده‌ی خط جداساز شکل ۳ است.

و $\hat{y}_i$ خروجی شبکه برای این نمونه باشد، مجموع زیر بیانگر تعداد نمونه‌هایی است که به اشتباه طبقه‌بندی شده‌اند:

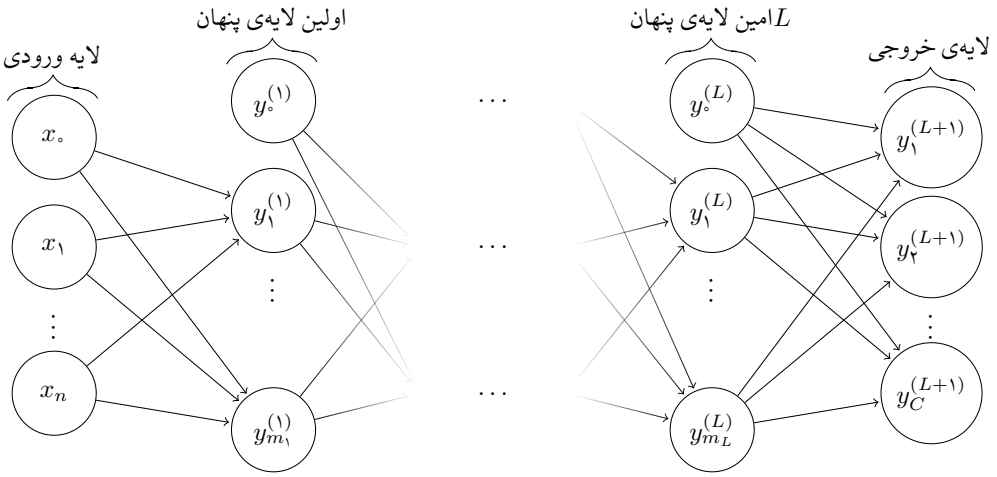
$$\sum_i |y_i - \hat{y}_i| \qquad (۱)$$

یعنی $\hat{y}_i$ که خروجی شبکه‌ی عصبی است ممکن است منطبق بر مقدار مورد انتظار y_i باشد (طبقه‌بندی درست) یا نباشد (طبقه‌بندی نادرست). نحوه‌ی محاسبه‌ی خروجی شبکه عصبی، در ادامه آمده است.

همان‌گونه که در ابتدای بخش اشاره شد یک نود در شبکه‌های عصبی مصنوعی، سیگنالهای ورودی را از n نورون دیگر یا ورودی‌های شبکه دریافت کرده و مجموع وزن‌دار این سیگنال‌ها، ورودی تابع فعالیت نورون را شکل می‌دهد (شکل ۲). یک مدل ساده با تابع فعالیت همانی، در واقع مشخص‌کننده‌ی یک مرز خطی برای دو ناحیه در صفحه است که هر ناحیه، یک دسته را مشخص می‌کند. با یک مثال، نحوه‌ی پیاده‌سازی ترکیب عطفی x_1 و x_2 با مدل پرسپترون را نشان خواهیم داد. خط $w_0 + w_1x_1 + w_2x_2 = 0$ با $w_0 = -1.5$ و $w_1 = w_2 = 1$ را درنظر بگیرید (خط پررنگ در شکل ۳). با این فرض که True و False به ترتیب با یک و صفر نشان داده شوند، خط فوق، مشخص‌کننده‌ی مرز تصمیم‌گیری برای $x_1 \wedge x_2$ است. اگر x_1 و x_2 هر دو یک (True) باشند، $x_1 + x_2 - 1.5$، مثبت و در سه حالت دیگر، منفی خواهد شد. اگر مثبت و منفی را به ترتیب متناظر با True و False فرض کنیم، خط فوق عمل ترکیب عطفی دو ورودی خود را انجام می‌دهد. مدل پرسپترون نمایش داده شده در شکل ۴، حاصل $w_0 + w_1x_1 + w_2x_2$ را برای x_1 و x_2 ورودی محاسبه می‌کند که با توضیح داده شده، در واقع عمل ترکیب عطفی ورودی‌های خود را انجام می‌دهد. معادله‌ی ساده‌ی خط فوق که در اینجا مشخص‌کننده‌ی یک مرز خطی بود، در فضاهای با ابعاد بالاتر (تعداد مؤلفه‌ها یا تعداد ویژگی‌های بیشتر) یک ابرصفحه‌ی جداساز را مشخص می‌کنند. فرض کنید مطابق رابطه‌ی زیر، a مجموع وزن‌دار ورودی‌های نورون باشد:

$$a = \sum_{j=1}^{n} w_j x_j + b \qquad (۲)$$

که در آن n تعداد ویژگی‌ها (مؤلفه‌ها)ی هر نمونه، b عرض از مبدأ (بایاس)، x_j، jامین سیگنال ورودی و w_j وزن اتصال ورودی jام به این نورون است. برای راحتی، عموماً یک ورودی $x_0 = 1$ درنظر گرفته شده و $w_0 = b$ به عنوان وزن



شکل ۵: شبکه عصبی پرسپترون چند لایه

اتصال این ورودی منظور می‌شود. به این ترتیب، رابطه بالا به صورت زیر نوشته می‌شود:

$$a = \sum_{j=1}^{n} w_j x_j + b = \sum_{j=0}^{n} w_j x_j = \mathbf{W}^T \mathbf{x} \qquad (۳)$$

که در آن، $\mathbf{x} = [x_0 = 1, x_1, \cdots, x_n]^T$ بردار ورودی و $\mathbf{W} = [w_0 = b, , w_1, \cdots, w_n]^T$ وزن اتصالات می‌باشد. سیگنال خروجی $\hat{y}$ یا پاسخ نورون، تابعی از ورودی آن است:

$$\hat{y} = g(a) = g\left(\sum_{j=0}^{n} w_j x_j\right) = g(\mathbf{W}^T \mathbf{x}) \qquad (۴)$$

تابع $g(.)$ تابع فعالیت نامیده می‌شود. توابع مختلفی به عنوان تابع فعالیت درنظر گرفته می‌شوند که تابع سیگموید با ضابطه‌ی $g(x) = \frac{1}{1+e^{-x}}$ از مرسوم‌ترین آنهاست. این ساختار ساده «مدل پرسپترون» نامیده می‌شود که در شکل ۲ نشان داده شده است.

۲.۲. روال آموزش پرسپترون. مجموعه‌ی آموزشی، متشکل از N زوج از الگوهای $\{(\mathbf{x}_i, t_i),\ i = 1, \cdots, N\}$ است که در آن $\mathbf{x}_i \in \mathbb{R}^n$ بردار نشان دهنده‌ی iامین ورودی و $y_i \in \mathbb{R}$ طبقه‌ی درست متناظر با این نمونه است. اگر مسأله‌ی مدنظر، طبقه‌بندی نمونه‌ها به دو کلاس باشد، یعنی $y_i \in \{0, 1\}$، که در آن، صفر و یک برچسب دو کلاس هستند.

تابع هدف (۱) را می‌توان در قالب مجموع مربعات خطا به صورت زیر نوشت که برای کمینه‌سازی مناسب‌تر است:

$$\frac{1}{2} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \qquad (۵)$$

اگر iامین نمونه از داده‌ها به مدل داده شود، خروجی مدل به صورت $\hat{y}_i = g(a_i) = g(\sum_{j=0}^{n} w_j x_j)$ خواهد بود. با جایگزینی $\hat{y}_i$ در تابع هدف (۵)، آن را می‌توان به صورت زیر برحسب w_i نوشت:

$$E(\mathbf{W}) = \frac{1}{2}\sum_i (y_i - \hat{y}_i)^2 = \frac{1}{2}\sum_i \left(y_i - g(\sum_{j=0}^{n} w_j x_j) \right)^2 \qquad (۶)$$

در این تابع خطا[۲۲]، x_j و y_i معلوم بوده و هدف، پیدا کردن وزن‌های بهینه $(\mathbf{W})$ برای کمینه‌سازی خطاست. پایه‌ی همه‌ی روش‌های مرسوم کمینه‌سازی در شبکه‌های عصبی الگوریتم *گرادیان کاهشی* می‌باشد. در هر مرحله از یک روال تکراری، وزن‌های جدید بر اساس گرادیان تابع هدف، به صورت زیر بهنگام می‌شوند:

$$\mathbf{W}^{new} = \mathbf{W}^{old} - \eta \nabla E \qquad (۷)$$

که η نرخ یادگیری و ∇E گرادیان تابع خطاست. کافی است گرادیان تابع خطا برحسب وزن‌های اتصالات، محاسبه شده و وزن‌ها، مطابق رابطه‌ی بالا بروزرسانی شوند، که در ادامه بیان خواهد شد. برای ساده‌سازی، خطای یک نمونه (بدون اندیس i) را درنظر بگیرید:

$$e = \frac{1}{2}(y - \hat{y})^2$$

بنا به قاعده‌ی مشتق زنجیره‌ای داریم:

$$\frac{\partial e}{\partial w_j} = \frac{\partial e}{\partial y}\frac{\partial y}{\partial a}\frac{\partial a}{\partial w_j} = -(y - \hat{y})g'(a)x_j = (\hat{y} - y)g'(a)x_j \qquad (۸)$$

لذا:

$$\frac{\partial E}{\partial w_j} = \sum_i (\hat{y}_i - y_i)g'(a)x_j \qquad (۹)$$

بنابراین گرادیان تابع هزینه را به صورت زیر خواهیم داشت که می‌تواند در رابطه‌ی (۷) برای بهنگام سازی وزن‌ها به‌کار گرفته شود:

$$\nabla E = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n} \right]^T \qquad (۱۰)$$

به عنوان مثال اگر $g(.)$ تابع همانی $g(a) = a$ باشد، مشتق آن بر حسب a یک شده و لذا روابط (۸) و (۱۰) برای نمونه‌ی iام به صورت زیر درخواهد آمد:

$$\frac{\partial e_i}{\partial w_j} = (\hat{y}_i - y)x_j \qquad (۱۱)$$

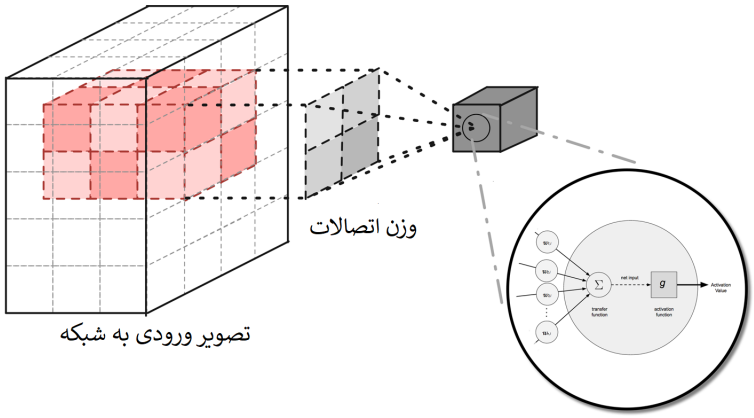
که

$$\nabla e_i = \left[\frac{\partial e_i}{\partial w_0}, \frac{\partial e_i}{\partial w_1}, \ldots, \frac{\partial e_i}{\partial w_n} \right]^T \qquad (۱۲)$$

در رابطه‌ی (۱۰) خطای همه‌ی داده‌های ورودی محاسبه می‌شود. اما به صورت معمول در روال آموزش شبکه‌های عصبی، خطای داده‌های آموزشی به صورت تکی یا گروهی در بروزرسانی وزن‌ها بکار برده می‌شوند:

$$\mathbf{W}^{new} = \mathbf{W}^{old} - \eta \nabla e_i \qquad (۱۳)$$

[22] Error Function



شکل ۶: مدل شبکه‌ی عصبی پیچشی. عموماً در نمایش شبکه‌های پیچشی از مکعب مستطیل برای نمایش نورون‌ها و از ماتریس (دوبعدی) یا مکعب مستطیل (سه‌بعدی) برای نمایش وزن‌ها استفاده می‌شود. مکعب خاکستری در واقع یک نورون با مدل پرسپترون نمایش داده شده در شکل ۲ است. اتصالات شبکه حکم فیلترهای یک شبکه‌ی پیچشی را دارند.

عنصر اصلی شبکه‌های چندلایه‌ی پرسپترونی[۲۳] همین مدل پرسپترون ساده است. یک شبکه‌ی چند لایه معمولاً شامل یک لایه‌ی ورودی، یک لایه‌ی خروجی و یک یا چند لایه‌ی پنهان است. شکل ۵، یک نمونه شبکه عصبی چند لایه را نشان می‌دهد. ساختار هر نود که با y نشان داده شده است، همان ساختار مدل پرسپترون شکل ۲ است. کلیات روش یادگیری در این شبکه‌ها و شبکه‌های پیچشی (شکل ۶) که با نام **پس انتشار خطا**[۲۴] [۱۶] معروف است، مشابه روال فوق‌الذکر است.

۳.۲. شبکه‌های عصبی پیچشی. در شکل ۶، بخشی از یک شبکه‌ی پیچشی که با نام شبکه‌های پیچشی عمیق هم نامیده می‌شوند ملاحظه می‌شود. برخلاف شبکه‌های چندلایه‌ی پرسپترونی که ورودی در قالب بردار به شبکه داده می‌شود، ورودی شبکه‌های پیچشی یک تصویر است. اگر تصویر رنگی (مرکب از سه رنگ قرمز، سبز و آبی) باشد، مطابق شکل فوق‌الذکر، ورودی یک مکعب مستطیل خواهد بود. در عموم شبکه‌های عصبی مرسوم، هر نود به همه نودهای لایه‌های مجاور متصل است، اما در شبکه‌های عصبی پیچشی، هر نود فقط از تعدادی از نودهای لایه‌ی قبل تاثیر می‌پذیرد. این بخش کار، مشابه عمل پیچش (کانولوشن) در پردازش سیگنال‌هاست که وجه تسمیه‌ی این شبکه‌ها به شبکه‌های عصبی پیچشی شده است. همچنین برای نمایش‌ها نورون‌ها برخلاف روال معمول نمایش شبکه‌های پرسپترونی که نورون‌ها با دایره نمایش داده می‌شوند، در شبکه‌های پیچشی از مکعب استفاده می‌شود. مکعب خاکستری نمایش داده شده در شکل ۶ همان مدل پرسپترون نمایش داده شده در شکل ۲ است. از آنجا که تعداد اتصالات در شبکه‌های پیچشی بسیار زیاد است، به صورت خط نمایش داده نمی‌شوند، بلکه گروهی از اتصالات دارای وزن، در قالب ماتریس یا مکعب مستطیل نمایش داده می‌شوند و یا اصلاً نمایش داده نشده و فقط تعداد آنها ذکر می‌شود (شکل ۱).

از معروف‌ترین توپولوژی‌های ابداع شده از شبکه‌های پیچشی، مدل VGG [۵] است که در شکل ۱ معماری مدلی از آن با نام VGG16 نشان داده شد. این مدل بیش از ۱۳۴ میلیون اتصال دارد. پنج گروه لایه‌ی پیچشی وظیفه‌ی استخراج ویژگی‌ها را دارند و سه لایه‌ی پرسپترونی آخر، کار طبقه‌بندی ویژگی‌های استخراج شده را انجام می‌دهند. این لایه‌ها در شکل ۱ به ترتیب با رنگ‌های زرد و بنفش مشخص شده‌اند. لایه‌های نارنجی رنگ بیانگر تابع فعالیت هستند و لایه‌های

[23] Multi Layer Perceptron (MLP) [24] Error Backpropagation

قرمز رنگ، لایه نمونه بردار یا انتخاب بیشینه (Max Pooling) نامیده می‌شوند که کار کوچک‌سازی تصویر را انجام داده و بدون وزن هستند. آخرین لایه به تعداد کلاس‌های مورد طبقه‌بندی، نورون دارد. در مدل اصلی آموزش دیده، هزار طبقه وجود دارد، اما برای کاربرد مدنظر این نوشتار، دو نورون خروجی برای دو دسته سبک نقاشی درنظر گرفته شده است.

جدول ۱ تعداد وزن‌های مدل شکل ۱ را نشان می‌دهد. ستون Layer بیانگر نام لایه‌هاست که در شکل ۱ در ضلع زیرین مکعب مستطیل‌ها نوشته شده است. ستون Output Shape اندازه خروجی هر لایه را نشان می‌دهد. به عنوان نمونه، خروجی اولین لایه‌ی پیچشی (اولین لایه زرد رنگ شکل ۱) در اولین سطر جدول ۱ به صورت $[\mathbf{64}, 224, 224]$ بیانگر آن است که ۶۴ فیلتر بر روی تصویر ورودی — که در اینجا 224×224 بوده است — اعمال شده و خروجی هر کدام یک تصویر 224×224 می‌باشد. ستون آخر، تعداد پارامترهای هر لایه را نشان می‌دهد. برای اولین سطر، اگر مشابه با شکل ۶، اندازه هر فیلتر 3×3 درنظر گرفته شود، هر فیلتر برای یک تصویر رنگی $3 \times 3 \times 3 = 27$ وزن اتصالی دارد که با احتساب بایاس ۲۸ وزن خواهد شد. لذا برای اولین لایه که ۶۴ فیلتر داریم، به تعداد $64 \times 28 = 1792$ پارامتر خواهیم داشت[۲۵]. چون تصویر مربعی درنظر گرفته شده است، فقط در کنار ضلع پایین مربع‌های شکل ۱ اندازه‌ی آنها نوشته شده است. لایه‌هایی که بدون وزن بوده‌اند نمایش داده نشده‌اند؛ به همین دلیل شماره لایه‌ها پی‌درپی نیست. تعداد پارامترها در اینجا همان تعداد اتصالات است که هر یک دارای یک وزن هستند همان‌گونه که ملاحظه می‌شود این مدل بیش از ۱۳۰ میلیون اتصال دارد. روال کلی آموزش شبکه، همان روشی است که در بخش ۲.۲ توضیح داده شد. صرفنظر از برخی ظرافت‌ها و پیچیدگی‌های پیاده‌سازی عملی، یک مسئله مهم در مورد این شبکه‌ها زمان‌بر بودن آموزش آنهاست. با هر داده آموزشی (یا گروهی از داده‌ها) که به شبکه داده می‌شود، همه‌ی وزن اتصالات باید بهنگام شوند. استفاده از کارت‌های گرافیکی با توان پردازشی بالای موازی امکان انجام چنین حجم زیادی از پردازش‌ها را مهیا کرده است؛ با این حال، کاهش حجم مدل، موجب کاهش میزان محاسبات و کمتر شدن حافظه‌ی مورد نیاز خواهد شد.

۳. کاهش حجم شبکه‌های عصبی پیچشی

همانگونه که در جدول ۱ ملاحظه شد، مدلی مانند VGG میلیون‌ها اتصال دارد. مدل اولیه بر روی ۱۴ میلیون تصویر و برای طبقه‌بندی هزار موجودیت متفاوت آموزش دیده است. یکی از مباحث حوزه یادگیری عمیق، *یادگیری انتقالی*[۲۶] است که هدف در آن به‌کارگیری یک مدل آموزش دیده در کاربردهای جدید است. به عنوان مثال اگر قرار باشد تصاویری متعلق به اشیاء جدیدی دسته‌بندی شوند که در هزار دسته‌ی اولیه‌ی VGG نبوده‌اند، می‌توان با داشتن یک مدل آموزش دیده و فقط چند صد تصویر از اشیاء جدید، مدل قبلی را با صرف هزینه‌ی محاسباتی نسبتاً کم، دوباره آموزش داد تا برای شناسایی اشیاء جدید قابل استفاده باشد. اگر مدل جدید فقط برای شناسایی چند دسته‌ی محدود موردنیاز باشد، احتمالاً بتوان بدون کاهش کارایی شبکه، تعدادی از اتصالات زائد[۲۷] را حذف کرد. این کاری است که به عنوان کاهش مدل انجام می‌شود.

۱.۳. چالش‌ها. راه حل اولیه‌ای که ممکن است به ذهن متبادر شود این است که به نوبت، هر یک از اتصالات حذف شده و تاثیر آن در کاهش خطای شبکه محاسبه شود. اتصالاتی که حذف آنها مؤثر باشد، هرس خواهند شد. در پیاده‌سازی‌های معمول شبکه‌های پیچشی، امکان حذف یک اتصال به صورت مجزا فراهم نیست، اما می‌توان یک دسته از آنها (یک فیلتر یا یک لایه) را حذف کرد. اگر قرار به حذف یک اتصال باشد، به جای حذف آن، ضریب وزنی آن صفر

[۲۵] هدف این نوشتار بیان تفصیلی شبکه‌های پیچشی نیست و به همین مقدار بسنده می‌شود. برای آشنایی بیشتر با مدل‌های شبکه‌های عصبی پیچشی و چگونگی محاسبه تعداد پارامترهای آنها، مطالعه‌ی کتاب «یادگیری عمیق با پایتون» [۱۷] پیشنهاد می‌شود.

[26] Transfer Learning [27] Redundant

جدول ۱: خلاصه مدل و پارامترهای مدل VGG16 نمایش داده شده در شکل ۱ برای یک تصویر ورودی فرضی با ابعاد 224×224. این مدل $134,268,738$ وزن دارد. لایه‌هایی که بدون وزن بوده‌اند نمایش داده نشده‌اند. اعداد پررنگ بیانگر تعداد فیلترهای (مجموعه‌ای از وزن‌ها) آن لایه است. تعداد پارامترها ربطی به طول و عرض تصویر ورودی ندارد. کاهش تعداد اتصالات (پارامترها) برای لایه‌های پیچشی بالای نقطه‌چین انجام خواهد شد.

Layer (type)	Output Shape	Param #
Conv2d-1	[**64**, 224, 224]	1,792
Conv2d-3	[**64**, 224, 224]	36,928
Conv2d-6	[**128**, 112, 112]	73,856
Conv2d-8	[**128**, 112, 112]	147,584
Conv2d-11	[**256**, 56, 56]	295,168
Conv2d-13	[**256**, 56, 56]	590,080
Conv2d-15	[**256**, 56, 56]	590,080
Conv2d-18	[**512**, 28, 28]	1,180,160
Conv2d-20	[**512**, 28, 28]	2,359,808
Conv2d-22	[**512**, 28, 28]	2,359,808
Conv2d-25	[**512**, 14, 14]	2,359,808
Conv2d-27	[**512**, 14, 14]	2,359,808
Conv2d-29	[**512**, 14, 14]	2,359,808
Linear-33	[4096]	102,764,544
...	...	...
Linear-36	[4096]	16,781,312
Linear-38	[2]	8,194
	Total params: 134,268,738	

قرار داده می‌شود. به این ترتیب برای انتخاب فقط یک اتصال برای حذف، همه‌ی اتصالات به نوبت باید به صورت موقت حذف شده و اتصالِ با کمترین تاثیر در کارایی شبکه، انتخاب و به صورت دائمی حذف شود.

روش فوق در عمل کارایی نخواهد داشت؛ عموماً به جای حذف تکی، انتخاب و حذف نورون‌ها به صورت گروهی انجام می‌شود؛ اما انتخاب و حذف وزن‌ها با این روش، یک مسئله‌ی بهینه‌سازی ترکیبیاتی است. اگر در هر مرحله k اتصال از بین n اتصال انتخاب شوند، $\binom{n}{k}$ حالت برای انتخاب وجود دارد. به عنوان نمونه اگر مدل شامل یک میلیون اتصال بوده و هدف، انتخاب و حذف ده اتصال باشد، $\binom{10^6}{10} \approx 2.75 \times 10^{53}$ حالت ممکن برای انتخاب این ده اتصال وجود دارد که بررسی همه آنها عملی نیست.

پس از حذف تعدادی از اتصالات، شبکه نیازمند آموزش مجدد است؛ که این عمل نیز زمان‌بر است. اگر شبکه شامل n اتصال باشد و در هر مرحله تعداد کمی از اتصالات هرس شوند، بلافاصله نباید اقدام به انتخاب و حذف گروه بعدی اتصالات نمود. چون حذف اتصالات قبلی، ساختار شبکه را مقداری تغییر داده است و شبکه برای یک مجموعه داده‌ی مشخص، همان خروجی قبل از حذف را تولید نمی‌کند. پس از هرس تعدادی از اتصالات، شبکه باید مجدداً مقداری آموزش داده شود تا اثر ناشی از حذف اتصالات جبران گردد.

ممکن است این پرسش مطرح شود که درصورت حذف تعدادی از اتصالات، وضعیت شبکه به چه صورت درخواهد آمد؟ همان‌گونه که پیش‌تر ذکر شد، عموماً در عمل، گروهی از اتصالات حذف می‌شوند. هر لایه در یک شبکه‌ی عصبی پیچشی، متضمن چندین فیلتر (گروهی از اتصالات) است. به عنوان نمونه، اولین لایه‌ی مدل VGG16 مطابق جدول ۱

دارای ۶۴ فیلتر است. حذف یک یا چند فیلتر از این لایه، خروجی و تعداد پارامترهای این لایه و ورودی به لایه‌ی بعدی را تحت تاثیر قرار خواهد داد. کاهش تعداد فیلترها، مترادف با کاهش حجم مدل است که هدف این نوشتار هم همین است.

همان‌گونه که در بخش ۳.۲ اشاره شد، بسیاری از مدل‌های پیچشی عمیق دارای تعداد بسیار زیادی نورون و اتصالات بین نورون‌ها می‌باشند. حذف اتصالات زائد می‌تواند منجر به ایجاد مدلی کوچک‌تر شود که حجم حافظه و محاسبات کمتری نیاز داشته باشد. مولچانوف و همکاران ایشان شیوه‌ای مبتنی بر بسط تیلور تابع هدف (۶) برای انتخاب وزن‌های زائد پیشنهاد داده‌اند [۱۵] که در این بخش این روش به صورت مبسوط بیان می‌شود.

۲.۳. استفاده از بسط تیلور در انتخاب وزن‌ها برای کاهش حجم مدل. فرض کنید $\mathbf{W}$ بردار وزن‌های شبکه و $E(\mathbf{W})$ تابع هزینه‌ی مسئله‌ی طبقه‌بندی باشد. کاهش حجم مدل، به معنی کاستن تعداد درایه‌های $\mathbf{W}$ است. برای بیان ریاضی مسئله، به جای حذف درایه‌ها، مؤلفه‌های مورد هرس از $\mathbf{W}$ را صفر کرده و بردار جدید را $\mathbf{W}'$ می‌نامیم. صفر شدن وزن یک اتصال مترادف با حذف (هرس) ارتباط بین دو نود مربوطه است. اگر فرض کنیم هیچ‌کدام از درایه‌های $\mathbf{W}$ برابر با صفر نباشد، نرم صفر $\mathbf{W}'$ ($\|\mathbf{W}'\|_0$)، تعداد مؤلفه‌های غیرصفر مدل، بعد از هرس را نشان می‌دهد. به این ترتیب میزان تغییر خطای شبکه، ناشی از هرس کردن تعدادی از اتصالات آن را می‌توان به صورت $|E(\mathbf{W})-E(\mathbf{W}')|$ نشان داد. فرض کنید میزان کاهشِ حجم نهایی مشخص است، یعنی به عنوان مثال برای B مفروض، مایل هستیم $\|\mathbf{W}'\|_0 \leq B$. کمتر بودن B معادل کمتر بودن مؤلفه‌های غیرصفر بردار وزن و به عبارت دیگر کمتر بودن اتصالات فعال مدل است. اگر اتصالاتِ دارای وزن برابر با صفر — به عنوان اتصالات غیرفعال — از بردار وزن حذف شوند، مدلی کوچک‌تر خواهیم داشت. حجمِ مدل جدید ناشی از هرس اتصالات، برابر با تعداد درایه‌های غیرصفر باقیمانده (نرم صفر بردار) است. با مفروضات فوق، مسئله‌ی کاهش تعداد اتصالات شبکه را می‌توان به صورت زیر نوشت:

$$\min_{\mathbf{W}'} |E(\mathbf{W}) - E(\mathbf{W}')| \quad \text{s.t. } \|\mathbf{W}'\|_0 \leq B \tag{۱۴}$$

وقتی یک اتصال (وزن) حذف شود، میزان تغییر تابع خطا با استفاده از بسط تیلور تقریب زده می‌شود. ابتدا مرور کوتاهی بر بسط تیلور داشته و سپس نحوه‌ی انتخاب اتصالاتی از شبکه که حذف آنها تاثیر کمی بر کارایی شبکه داشته باشد، بیان خواهد شد.

اگر تابع f در نقطه‌ی a (نزدیک به x) بی‌نهایت بار مشتق‌پذیر باشد، داریم:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots \tag{۱۵}$$

که در آن $f^{(n)}(.)$ مشتق مرتبه nام تابع $f(.)$ است. اگر $f(.)$ یک تابع حقیقی مقدار چند متغیره باشد، بسط تیلور آن به صورت زیر است:

$$T(\mathbf{x}) = f(\mathbf{a}) + (\mathbf{x}-\mathbf{a})^{\mathsf{T}}\nabla f(\mathbf{a}) + \frac{1}{2!}(\mathbf{x}-\mathbf{a})^{\mathsf{T}}\left\{\nabla^2 f(\mathbf{a})\right\}(\mathbf{x}-\mathbf{a}) + \cdots \tag{۱۶}$$

در حالت دو متغیره، اگر از جملات مربوط به مشتقات دوم به بعد صرفنظر کنیم، داریم:

$$f(x,y) \approx f(a,b) + \frac{\partial}{\partial x}f(a,b)(x-a) + \frac{\partial}{\partial y}f(a,b)(y-b) \tag{۱۷}$$

بسط فوق، فقط در راستای محور xها حول نقطه‌ی (a,y) به صورت زیر است:

$$f(x,y) \approx f(a,y) + \frac{\partial}{\partial x}f(a,y)(x-a)$$

بر این اساس، اگر به فرض، مدل شبکه عصبی دارای دو وزن w_1, w_2 باشد، تقریب خطی فوق در راستای w_1 حول نقطه‌ی $(0, w_2)$ را به صورت خواهیم داشت:

$$E(w_1, w_2) \approx E(0, w_2) + \frac{\partial}{\partial w_1} E(0, w_2)(w_1 - 0) \tag{۱۸}$$

پیش از این، بردار $\mathbf{W}$ که تعدادی از درایه‌های آن صفر باشند، با $\mathbf{W}'$ نمایش داده شد؛ برای حالت دو متغیره‌ی فوق‌الذکر و با فرض $\mathbf{W} = [w_1, w_2]$ و $\mathbf{W}' = [0, w_2]$، رابطه‌ی (۱۸) را می‌توان به صورت زیر نوشت:

$$E(\mathbf{W}) \approx E(\mathbf{W}') + \frac{\partial}{\partial w_i} E(\mathbf{W}')(w_i - 0), \qquad i = 1 \tag{۱۹}$$

به عبارت دیگر در حالت کلی داریم:

$$E(\mathbf{W}) - E(\mathbf{W}') \approx \frac{\partial}{\partial w_i} E(\mathbf{W}') w_i \tag{۲۰}$$

و لذا:

$$|E(\mathbf{W}) - E(\mathbf{W}')| \approx \left| \frac{\partial}{\partial w_i} E(\mathbf{W}') w_i \right| \tag{۲۱}$$

به این ترتیب مسئله‌ی کمینه‌سازی (۱۴) را می‌توان به صورت زیر نوشت:

$$\min \left| \frac{\partial}{\partial w_i} E(\mathbf{W}') w_i \right| \quad \text{s.t. } \|\mathbf{W}'\|_0 \leq B \tag{۲۲}$$

هدف اصلی در هرس شبکه‌ها، یافتن اتصالاتی بود که حذف آنها کمترین تاثیر را بر کارایی شبکه داشته باشد. در رابطه‌ی (۲۱)، مقدار $|E(\mathbf{W}) - E(\mathbf{W}')|$ بیانگر تغییر کارایی شبکه در صورت صفر شدن iامین مؤلفه‌ی $\mathbf{W}$ است که با بسط تیلور، معادل بودن تقریبی آن با $\left| \frac{\partial}{\partial w_i} E(\mathbf{W}') w_i \right|$ نشان داده شد. حال برای یافتن اتصالی که حذف آن کمترین *اثر در تغییر کارایی شبکه* را داشته باشد، کافی است آن w_i انتخاب شود که کمترین مقدار $\left| \frac{\partial}{\partial w_i} E(\mathbf{W}') w_i \right|$ را داشته باشد. محاسبه‌ی این عبارت فقط نیازمند وزن مربوطه و مشتق تابع هدف است. ابزارهای مرسوم مورد استفاده در حوزه یادگیری عمیق در روال تنظیم وزن‌ها، مقادیر وزن‌ها و مشتق تابع هدف را در اختیار برنامه‌نویس قرار می‌دهند. به آسانی[۲۸] می‌توان وزن‌ها را بر اساس حاصل‌ضرب مشتق تابع در وزن اتصال به صورت صعودی مرتب نموده و وزن با کمترین مقدار $\left| \frac{\partial}{\partial w_i} E(\mathbf{W}') w_i \right|$ را انتخاب و حذف کرد. در ادامه، کاربردی از این شیوه در مسئله‌ی تفکیک دو سبک نقاشی بیان خواهد شد.

۴. کاربرد در طبقه‌بندی نقاشی‌های سبک امپرسیونیسم و مینیاتور

در این بخش در قالب یک کاربرد جدید، تاثیر هرس وزن‌ها بر پایه‌ی بسط تیلور را خواهیم دید. روش مورد بررسی در بستر پای‌تورچ[۲۹] پیاده‌سازی شده است که به همراه تصاویر به‌کار برده شده، از گیت‌هاب نگارنده[۳۰] قابل دسترس است. کاربرد مدنظر، تفکیک تابلونگاره‌های دو سبک نقاشی مشهور است. فرض کنید چند صد تصویر از دو نوع سبک نقاشی —به عنوان تصاویر آموزشی— و مدلی از قبل آموزش دیده[۳۱] بر روی اشیایی دیگر در دسترس است. یک مدل VGG16 حدود ۵۰۰ مگابایت حجم دارد. هدف، ایجاد یک مدل کم حجم‌تر است که بتواند نقاشی‌های دیگری از این دو

[۲۸] البته این «به آسانی»، از آن به آسانی‌هاست که گاهی در منابع ریاضی ملاحظه می‌کنیم که «به آسانی نتیجه می‌شود ...». مرتب‌سازی مشکلی ندارد، اما دسترسی به مشتق تابع در هر مرحله و صفر کردن وزن اتصالات، نیازمند ریزه‌کاری‌های فراوان برنامه‌نویسی است.

[29] PyTorch: https://pytorch.org/ [30] https://github.com/mamintoosi/CNN-pruning-using-Taylor-expansion

[31] Pre-trained Model

شکل ۷: تصاویر کلود مونه (نقاش با سبک امپرسیونیسم) و محمود فرشچیان (مینیاتور) در کنار برخی از آثار ایشان.

سبک که در مجموعه آموزشی نبوده‌اند را با دقت خوبی به دو دسته‌ی مربوطه کلاسه‌بندی نماید. به این منظور دو سبک نقاشی امپرسیونیسم (برداشت‌گرایی) و مینیاتور با سبک فرشچیان انتخاب شدند. در زمان نگارش این متن (شهریور ۹۹) مسابقه‌ی تولید نقاشی با سبک برداشت‌گرایی مونه در کاگل[۳۲] آغاز شده بود[۳۳]. هدف مسابقه‌ی مذکور ایجاد یک شبکه‌ی مولد رقابتی[۳۴] برای تولید خودکار نقاشی‌های جدید با سبک امپرسیونیسم کلود مونه[۳۵] بوده است. شیوه‌ی برداشت‌گرایی به عنوان سبکی از نقاشی توسط گروهی از هنرمندان ساکن پاریس آغاز شده است. نام این سبک از نام یک نقاشی از کلود مونه بنام *برداشت گرایی، طلوع خورشید* (به فرانسوی: Impression, soleil levant) گرفته شده‌است[۳۶]. مسابقه‌ی فوق نقطه شروع ایده‌ی این کاربرد بود. به عنوان سبک دیگری از نقاشی که یک سیستم تفکیک دو سبک مدل‌سازی شود، مینیاتورهای (خُرد نگارگری‌های) محمود فرشچیان انتخاب شد. شکل ۷، دو نمونه از نقاشی‌های کلود مونه و فرشچیان را نشان می‌دهد. تصویر فرضی ورودی به مدل نمایش داده شده در شکل ۱، مینیاتور *ضامن آهوی* فرشچیان بوده است.

تا آنجا که نگارنده می‌داند هنوز کاری درخصوص طبقه‌بندی تابلونگاره‌های سبک‌های نقاشی انجام نشده است. البته هدف از این نوشتار، ایجاد بهترین مدل برای طبقه‌بندی این سبک‌های نقاشی نیست؛ هدف اصلی آن است که نحوه‌ی استفاده از بسط تیلور در کاهش حجم مدل‌های یادگیری عمیق بیان شده و کاربرد جدیدی از آن بدینوسیله نشان داده شود. لذا موضوع مقایسه با سایر روش‌های طبقه‌بندی مطرح نیست.

برای جمع‌آوری تصاویر موردنیاز، با یک خزنده‌ی وب[۳۷]، با جستجوی عبارات «impressionism-claude-monet» و «مینیاتورفرشچیان» در گوگل، حدود هزار تصویر برداشت شد. با ملاحظه‌ی بصری، تعداد زیادی از نتایج نامطلوب حذف شدند و برای هر دسته ۳۱۳ تصویر نگه داشته شد. تصاویر به دو دسته‌ی آموزش و آزمون تقسیم شدند. از مجموع ۶۲۶ تصویر، ۹۰درصد (۵۲۶ تصویر) به عنوان آموزش و ۶۲ تصویر باقیمانده به عنوان داده آزمون درنظر گرفته شدند. اندازه تصاویر ورودی کوچک و در حدود 150×150 پیکسل بوده است.

ابتدا مدل اصلی از قبل آموزش دیده‌ی شکل ۱ با پارمترهای جدول ۱ بر روی داده‌های آموزشی مورد آموزش مجدد قرار گرفت. برای ارزیابی کارایی مدل از معیار صحت[۳۸] استفاده شده است. «صحت» عبارت از نسبت تعداد نمونه‌هایی

---

[۳۲] سایت kaggle.com از معروف‌ترین سایت‌های برگزاری مسابقات یادگیری ماشین و داده‌کاوی است که جوایز برخی از چالش‌های آن به چند ده هزار دلار هم می‌رسد.

[33] Will you be the next Monet? https://www.kaggle.com/c/gan-getting-started/ [34] Generative Adversarial Network

[35] Claude Monet [36] https://en.wikipedia.org/wiki/Impressionism [37] Web Crawler [38] Accuracy

شکل ۸: خروجی روی ده تصویر آزمون. تصویر اول از سطر دوم، متعلق به سبک مینیاتور بوده است که به اشتباه متعلق به سبک امپرسیونیسم شمرده شده است.

جدول ۲: مشخصات سیستم گوگل کولب، مورد استفاده در آزمایش‌ها.

CPU	Intel(R) Xeon(R) CPU @ 2.20GHz
GPU	Tesla P100-PCIE-16GB 3584 CUDA cores , 16GB vRAM
RAM	12.6 GB

که به درستی طبقه‌بندی شده‌اند به کل داده‌های مورد بررسی است و به صورت عددی در بازه‌ی صفر تا یک و یا برحسب درصد بیان می‌شود. میزان صحت مدل بدست آمده، روی داده‌های آزمون ۹۲ درصد بوده است که به منزله‌ی شناسایی درست ۵۹ تصویر از ۶۴ تصویر تست (آزمون) بوده است. مدت زمان کل آموزش مجدد مدل فوق با پانزده تکرار روی ۶۲۶ تصویر آموزشی یک دقیقه و شش ثانیه (۶۶ ثانیه) بوده است. برنامه روی سرورهای گوگل و مجهز به کارت گرافیک تسلا اجرا شده است. مشخصات سیستم مورد استفاده در جدول ۲ آمده است.

زمان اجرای مورد نیاز این برنامه روی یک دستگاه معمولی (Core i5) و بدون بهره‌بردن از توان کارت گرافیک، حدوداً پنجاه برابر زمان اجرا روی سیستم فوق است. پس از آموزش اولیه، هرس شبکه با روش بسط تیلور انجام شد. همان‌گونه که در بخش ۱.۳ اشاره شد، پس از حذف اتصالات، شبکه باید مورد آموزش مجدد قرار گیرد تا اثر ناشی از هرس اتصالات، ترمیم شده، سایر اتصالات بتوانند خودشان را با این ساختار جدید وفق دهند. روش آموزش، مبتنی بر همان روال آموزشی است که در بخش ۲.۲ توضیح داده شد. روال تکراری حذف تعدادی از اتصالات و آموزش مجدد شبکه ۱۳ دقیقه و ۴۵ ثانیه به طول انجامید (۸۲۵ ثانیه). حدود ۸۳ درصد وزن‌های شبکه هرس شدند. میزان صحت مدل نهایی بدست آمده بر روی داده‌های آزمون ۹۶ درصد بوده است. شکل ۸، نتیجه‌ی اجرای برنامه‌ی پایتون نوشته شده روی ده تصویر آزمون را نشان می‌دهد. سطر اول، نمونه نقاشی‌های با سبک امپرسیونیسم هستند که همگی به درستی تشخیص داده شده‌اند. سطر دوم نمونه‌های متعلق به مینیاتور فرشچیان هستند که اولی به اشتباه «امپرسیونیسم» برچسب

جدول ۳: خلاصه مدل و پارامترهای مدل هرس شده با استفاده از بسط تیلور. هرس وزن‌ها فقط برای لایه‌های بالای نقطه‌چین در این جدول انجام شده است. اتصالات این لایه‌ها نسبت به لایه‌های متناظر در جدول ۱ نزدیک به ۹۵ درصد کاهش پیدا کرده‌اند. اگر تعداد کل پارامترهای مدل مدنظر باشد، نسبت تعداد پارامترهای این مدل به مدل اولیه 0.17 است که به منزله‌ی کاهش ۸۳ درصدی می‌باشد. به این ترتیب مدل بسیار کوچکتری حاصل شده است که کارایی کمتری از مدل اولیه ندارد.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [22, 224, 224]             616
            Conv2d-3          [29, 224, 224]           5,771
            Conv2d-6          [48, 112, 112]          12,576
            Conv2d-8          [39, 112, 112]          16,887
           Conv2d-11            [66, 56, 56]          23,232
           Conv2d-13            [62, 56, 56]          36,890
           Conv2d-15            [61, 56, 56]          34,099
           Conv2d-18            [64, 28, 28]          35,200
           Conv2d-20            [53, 28, 28]          30,581
           Conv2d-22            [61, 28, 28]          29,158
           Conv2d-25            [59, 14, 14]          32,450
           Conv2d-27            [46, 14, 14]          24,472
           Conv2d-29            [30, 14, 14]          12,450
           Linear-33                 [4096]       6,025,216
................................................................
           Linear-36                 [4096]      16,781,312
           Linear-38                    [2]           8,194
================================================================
                                Total params: 23,109,104
----------------------------------------------------------------
```

خورده است. این تشخیص نادرست توسط ماشین، نشان‌گر خطاهایی است که انسان هم ممکن است در تشخیص داشته باشد. یک تفاوت اصلی شبکه‌های عصبی پیچشی در یادگیری عمیق با سایر طبقه‌بندهای معمول مثل شبکه‌های عصبی چند لایه آن است که در شبکه‌های پیچشی، استخراج ویژگی توسط خود شبکه انجام می‌شود. برای آشنایی با بصری‌سازی لایه‌های میانی شبکه که عمل استخراج ویژگی‌ها را انجام می‌دهند، به مرجع [۱۷] مراجعه شود.

جدول ۳، پارامترهای مدل VGG16 را بعد از اعمال کاهش تعداد وزن‌ها نشان می‌دهد. در بالای نقطه‌چین جدول ۱، به تعداد $134,268,738 - 16,789,506 = 117,479,232$ پارامتر داریم؛ این تعداد در جدول ۳، به $23,109,104 - 16,789,506 = 6,319,598$ رسیده است که بیانگر نسبت $6,319,598/117,479,232 = 0.05$ است. به عبارت دیگر، روش مورد بررسی، کاهش حدود ۹۵ درصدی تعداد اتصالات لایه‌ی پیچشی در مدل جدید نسبت به مدل اولیه را باعث شده است. البته اگر تمام اتصالات مدنظر قرار گیرد نسبت کاهش وزن‌ها ۸۳ درصد بوده است. داده‌های این جدول و جدول ۱ حاصل خروجی دستور `summary` در پای‌تورچ بر روی مدل هستند.

۵. جمع‌بندی

نمایش کاربردهای ملموس نظریه‌های ریاضی می‌تواند برانگیزاننده‌ی اشتیاق دانش‌آموز یا دانشجو برای مطالعه و تعمق در آن حوزه باشد. در نوشتار حاضر، کاربردی از بسط تیلور برای کوچک‌سازی یک مدل بزرگ شبکه‌های عصبی پیچشی

به تفصیل شرح داده شد. مبحث شبکه‌های عصبی پیچشی با شروع از مدل ساده پرسپترون و بر پایه‌ی دانش ریاضیات عمومی مرور گردید. تابع هزینه‌ی شبکه‌های عصبی، میزان اختلاف خروجی شبکه با مقدار واقعی آن بر روی داده‌های آموزشی است. روال آموزش، شامل مشتق‌گیری از تابع هدف و روش گرادیان کاهشی برای رسیدن به نقطه‌ی بهینه است. در بسط تیلور، توابع از هر مرتبه مشتق‌پذیر را می‌توان به صورت چندجمله‌ای حول یک نقطه نمایش داد. با بسط تیلور تابع هدف در نقطه‌ای که فقط یکی از وزن‌ها صفر است، و سنجش اختلاف مقدار این تابع با وضعیتی که همه وزن‌ها لحاظ شوند، رابطه‌ای برای برآورد اهمیت حضور هر اتصال شبکه بدست می‌آید. بر اساس این رابطه که حاصل‌ضرب وزن اتصال مورد هرس در مشتق تابع در آن نقطه است، اتصالاتی که تأثیر کمتری در کارایی شبکه دارند، شناسایی و حذف شدند. کارایی شیوه‌ی مبتنی بر بسط تیلور در هرس اتصالات شبکه و کوچک کردن مدل بر روی کاربردی جدید (تفکیک تابلوهای با دو سبک نقاشی برداشت‌گرایی و مینیاتور) نشان داده شد. در این کاربرد خاص، تعداد پارامترهای مدل ۸۳ درصد کم شد و مدل جدیدی حاصل شد که گرچه فقط ۱۷ درصد تعداد اتصالات مدل اولیه را دارد، اما همانند مدل اولیه دقت بالای ۹۲ درصد روی داده‌های تست را حفظ کرده است.

در این نوشتار فقط از تقریب مرتبه‌ی اول بسط تیلور استفاده شد. استفاده از تقریب‌های با مرتبه‌ی بالاتر می‌تواند از جمله کارهای آتی باشد. همچنین در این نوشتار، اهمیت همه‌ی اتصالات یکسان درنظر گرفته شده و ملاک انتخاب اتصالات برای هرس، تنها حاصلضرب مقدار وزن اتصال در مشتق تابع هدف بود که بر اساس بسط تیلور بدست آمده بود. اما ممکن است شرایط دیگری هم مدنظر باشد؛ به عنوان مثال، ممکن است کاربر ترجیح دهد به جای هرس شدن مختصر دو لایه‌ی پیچشی، کل یک لایه حذف شود. تعیین حداکثر میزان فشردگی مدل، به شرط اینکه دقت کلی از میزان معینی کمتر نشود نیز می‌تواند از جمله موضوعات تحقیقاتی بعدی باشد.

سپاس‌گزاری

از داوران گرامی که با مطالعه‌ی دقیق و نظرات سازنده‌ی خود موجبات بهبود نوشتار حاضر را فراهم کردند، سپاس‌گزاری می‌کنم.

مراجع

[1] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu, "A survey of deep learning-based object detection," *IEEE Access*, vol.7, pp.128837–128868, 2019.

[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp.580–587, 2014.

[3] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.42, no.2, pp.386–397, 2020.

[4] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," *CVPR*, 2017.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp.1097–1105, Curran Associates, Inc., 2012.

[8] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol.abs/1207.0580, 2012.

[9] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proceedings of the 30th International Conference on Machine Learning* (S. Dasgupta and D. McAllester, eds.), vol.28 of *Proceedings of Machine Learning Research*, (Atlanta, Georgia, USA), pp.1058–1066, PMLR, 17–19 Jun 2013.

[10] H. Wu and X. Gu, "Towards dropout training for convolutional neural networks," *Neural Networks*, vol.71, pp.1 – 10, 2015.

[11] B. Liu, M. Wang, H. Foroosh, M. F. Tappen, and M. Pensky, "Sparse convolutional neural networks.," in *CVPR*, pp.806–814, IEEE Computer Society, 2015.

[12] X. Chen, "Escort: Efficient sparse convolutional neural networks on gpus," *CoRR*, vol.abs/1802.10280, 2018.

[13] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Advances in Neural Information Processing Systems 29* (D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds.), pp.2074–2082, Curran Associates, Inc., 2016.

[14] K. Mitsuno, J. Miyao, and T. Kurita, "Hierarchical group sparse regularization for deep convolutional neural networks," 2020.

[15] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[16] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Neurocomputing: Foundations of research," chap. Learning Representations by Back-propagating Errors, pp.696–699, Cambridge, MA, USA: MIT Press, 1988.

[17] F. Chollet. *Deep Learning with Python*. Manning, Nov. 2017.

محمود امین‌طوسی

سبزوار، دانشگاه حکیم سبزواری، دانشکده ریاضی و علوم کامپیوتر

m.amintoosi@hsu.ac.ir ،http://mamintoosi.ir

محمود امین‌طوسی، استادیار دانشگاه حکیم سبزواری است. وی دوره‌های کارشناسی و کارشناسی ارشد خود را در رشته‌های ریاضی (گرایش کاربرد در کامپیوتر) و مهندسی کامپیوتر (گرایش نرم‌افزار) در دانشگاه فردوسی به اتمام رسانده و دوره دکترای خود را در رشته مهندسی کامپیوتر (گرایش هوش مصنوعی) در دانشگاه علم و صنعت ایران گذرانده است. علائق پژوهشی وی یادگیری ماشین، بینایی ماشین و بهینه‌سازی ترکیبیاتی می‌باشد. چاپ مقالات متعدد در کنفرانس‌ها و مجلات و انجام چند طرح تحقیقاتی از جمله کارهای پژوهشی وی می‌باشد.