دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)
دانشکده مهندسی کامپیوتر

مقدمه‌ای بر مباحث یادگیری ماشین و یادگیری عمیق

با رویکرد پیاده‌سازی سخت‌افزاری

نگارش، گردآوری و تنظیم:
فرزانه ارزاقی

ساخت تصاویر:
رضا آدینه‌پور

با نظارت:
دکتر مرتضی صاحب‌الزمانی

آذر ۱۴۰۲

فهرست مطالب

توضیحات:

- مطالب دارای ترتیب است. برای درک بهتر مفاهیم بهتر است از ابتدای گزارش راهنما خوانده شود.
- اگر مطلبی در مطالعه اولیه مورد نیازتان نیست و برای بخش پیاده‌سازی کاربرد دارد از آن گذر کنید.
- اگر برخی مطالب برای شما ساده است و یا بیش از نیاز شما از جزئیات صحبت کرده و پیچیده است از آن گذر کنید.
- در این گزارش راهنما سعی شده است یک جریان منظم برای مطالعه و آشنایی شما فراهم شود اما این به معنی این نیست که شما از جست‌وجوی شخصی بی‌نیاز می‌شوید بلکه با دید و اطلاعات طبقه‌بندی شده به جست‌وجوی مفیدتری خواهید رسید. لطفا از جست‌وجوهای شخصی خودتان دست برندارید.
- در کنار این گزارش راهنما فایلی قرار دارد که در آن فیلم‌های آموزشی زبان پایتون، چند نمونه از وزن‌ها و بایاس شبکه مفروض در این گزارش راهنما و یک سری مقاله مروری برای آشنایی با موضوعات پژوهشی این حوزه قرار داده شده است.
- کتابخانه‌های پرکاربرد این حوزه تنسورفلو، کراس و پای‌تورچ است. درباره اینکه این کتابخانه‌ها چه تفاوتی با یکدیگر دارند و چه بستری را مهیا کرده‌اند که برای پیاده‌سازی شبکه‌های عصبی از آن‌ها استفاده می‌کنند تحقیق کنید. در این گزارش آموزش پیشنهادی برای هر دو کتابخانه قرار داده شده است.
- تمام لینک‌های موجود در این گزارش راهنما پیشنهادی هستند و ممکن است شما برای درک یک مطلب مراجع بهتر و کاربردی‌تری داشته باشید. همچنین برای تمامی موضوعات می‌توانید با جست‌وجو در گوگل و یوتیوب مطالب و آموزش‌های بسیار زیادی بیابید.
- در این گزارش راهنما سعی شده است برای برخی مفاهیم ابتدایی مراجع فارسی قرار داده شود تا در ابتدا که مفاهیم و تعاریف گنگ هستند مخاطب تازه‌کار راحت‌تر مطلب را یاد بگیرد. چنانچه شما با خواندن توضیحات انگلیسی راحت‌تر هستید می‌توانید عنوان همان مطالب را به انگلیسی جست‌وجو کنید و راجع‌به آن مطالعه کنید.
- پس از مطالعه و آشنایی با بخش‌های تئوری این گزارش راهنما به راحتی قادر خواهید بود مقاله‌های کار شده در این حوزه را بخوانید و متوجه شوید پژوهشگران و نویسندگان مقالات چه فعالیت علمی انجام داده‌اند.

۱- یادگیری ماشین و یادگیری عمیق

امروزه به دلیل پیشرفت‌های سریع در حوزه هوش مصنوعی کمتر کسی است که با این حوزه آشنایی نداشته باشد. اگر بخواهیم نگاهی به دسته‌بندی‌های موجود در این حوزه بیندازیم، به مفاهیمی مانند یادگیری ماشین[۱]، یادگیری عمیق[۲] و شبکه‌های عصبی[۳] برخواهیم خورد. این مفاهیم به صورت تو در تو با یکدیگر در ارتباط هستند، به این صورت که یادگیری ماشین زیرشاخه‌ی هوش مصنوعی و یادگیری عمیق زیرشاخه‌ی یادگیری ماشین است. شبکه‌های عصبی نیز ستون فقرات الگوریتم‌های یادگیری عمیق را می‌سازند. به عبارتی دیگر معماری مدل‌های یادگیری عمیق از ساختاری با عنوان شبکه عصبی تشکیل شده است که با نام شبکه عصبی مصنوعی نیز شناخته می‌شوند. شکل ۱-۱ ارتباط بین سه مفهوم هوش مصنوعی، یادگیری ماشین و یادگیری عمیق را نمایش می‌دهد.

شکل۱-۱ ارتباط بین مفاهیم هوش مصنوعی، یادگیری ماشین و یادگیری عمیق[۱]

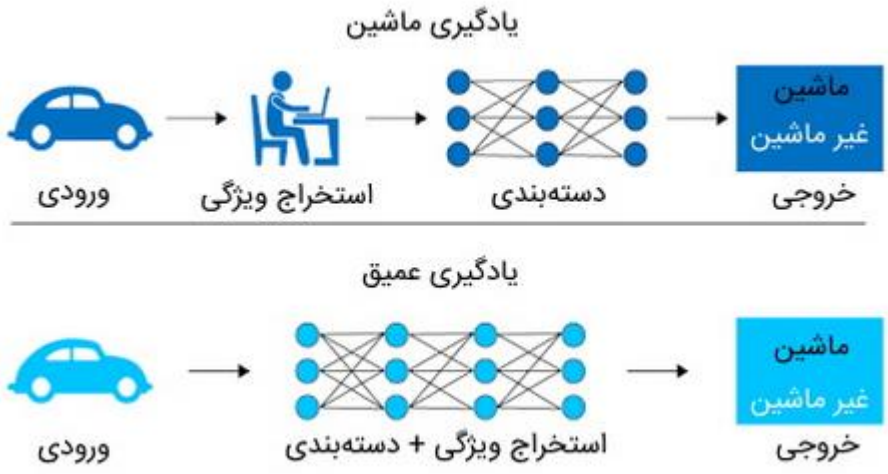
[1] Machine Learning

[2] Deep Learning

[3] Neural Network

روش‌های یادگیری ماشین سنتی نظیر درخت تصمیم[۴]، ماشین بردار پشتیبان[۵]، دسته‌بند بیز ساده[۶] و رگرسیون منطقی[۷] را نمی‌توان به‌طور مستقیم بر روی داده‌های خام نظیر فایل‌ها CSV، تصاویر و متون به منظور یادگیری داده‌ها اعمال کرد. به عبارت دیگر، باید با استفاده از مرحله پیش‌پردازش، از داده‌های خام، ویژگی‌هایی را به عنوان بازنمایی داده‌های خام استخراج کرد تا از این ویژگی‌ها به عنوان ورودی الگوریتم‌های یادگیری ماشین استفاده شود.

استخراج ویژگی روال پیچیده‌ای است و به دانشی عمیق پیرامون مسئله احتیاج دارد. به‌علاوه، باید از روال استخراج ویژگی چندین بار استفاده شود تا در نهایت بتوان بهترین ویژگی‌ها را برای مسئله تعریف شده انتخاب کرد. با ارائه روش‌های یادگیری عمیق، مشکل پیچیدگی فرآیند استخراج ویژگی و زمان استخراج آن‌ها حل شده است. به عبارت دیگر، مدل‌های یادگیری عمیق نیازی به گام مجزا برای استخراج ویژگی ندارند و لایه‌های شبکه‌های عصبی قادر هستند بازنمایی‌های ضمنی داده‌های خام را در روال آموزش شبکه یاد بگیرند. این موضوع در شکل ۱-۲ نشان داده شده است.



شکل۱-۲ تفاوت یادگیری ماشین و یادگیری عمیق[۱]

[4] Decision Tree

[5] Support Vector Machine

[6] Naive Bayes Classifiers

[7] Logistic Regression

به طور کلی در شبکه‌های عصبی دو فاز مختلف وجود دارد، ۱) فاز آموزش شبکه عصبی و ۲) فاز استنباط یا اینفرنس[۸]. منظور از اینفرنس همان وظیفه‌ای مانند طبقه‌بندی، تشخیص چهره و ... است که شبکه عصبی پس از آموزش باید انجام دهد. فاز آموزش شبکه عصبی به دو روش درون‌تراشه[۹] و برون‌تراشه[۱۰] صورت می‌گیرد. در روش درون‌تراشه، تمام مراحل آموزش شبکه عصبی بر روی تراشه صورت می‌گیرد. اما در روش برون‌تراشه، آموزش شبکه عصبی به صورت نرم‌افزاری انجام می‌شود. سپس برای اینفرنس، شبکه آموزش‌دیده بر روی تراشه پیاده‌سازی می‌شود. در کنار این دو روش موجود برای آموزش شبکه عصبی می‌توان به یک دسته‌بندی دیگر نیز اشاره کرد. شبکه‌های عصبی می‌توانند به صورت آنلاین یا آفلاین نیز آموزش ببینند. آموزش آفلاین در این دسته‌بندی به این معنی است که شبکه عصبی آموزش‌دیده در طول اینفرنس تغییری نمی‌کند و در طول این فاز، یادگیری جدیدی اتفاق نمی‌افتد و ساختار شبکه ثابت می‌ماند. در مقابل در آموزش آنلاین، شبکه عصبی در حین اینفرنس می‌تواند توسط داده‌های جدید آموزش ببیند و ساختارش تغییر کند. این نوع آموزش نام‌های دیگری مانند یادگیری افزایشی[۱۱] یادگیری مادام‌العمر[۱۲]، یادگیری سازنده[۱۳]و یادگیری تکاملی[۱۴] نیز دارد، که همگی نشان‌دهنده‌ی این است که فرآیند یادگیری توسط داده‌های جدید در زمان اینفرنس هم ادامه پیدا می‌کند.

برای مطالعه درباره شبکه عصبی انواع و جزئیات آن می‌توانید به لینک زیر مراجعه کنید. از جست‌وجوهای شخصی خودتان نیز دست برندارید.

شبکه عصبی، تعاریف و انواع آن

[8] Inference

[9] On_Chip

[10] Off _Chip

[11] Incremental Learning

[12] Lifelong Learning

[13] Constructive Learning

[14] Evolutionary Learning

در لینکی که پیش‌تر قرار داده شده است، درباره ساختار شبکه و تعاریف مربوط به آن همچنین انواع شبکه عصبی توضیحات کامل داده شده است.

انواع توابع فعال‌سازی به طور کامل در دو لینک زیر توضیح داده شده است. از جست‌وجوهای شخصی خودتان نیز دست برندارید.

توابع فعال‌سازی لینک ۱

توابع فعال‌سازی لینک ۲

در لینک‌هایی که برای مطالعه قرار داده شده درباره انواع شبکه عصبی توضیح داده شده است. برای آشنایی با جزئیات لایه‌های تشکیل‌دهنده هر یک از شبکه‌های نام‌برده می‌توانید درباره آن‌ها جست‌وجو کنید و مطالب مفیدی به دست آورید. به دلیل اینکه شبکه کانولوشنی شبکه پرکاربرد و پیچیده‌تری است در ادامه به بررسی جزئیات این شبکه پرداخته می‌شود.

۲- شبکه عصبی کانولوشنی

برای آشنایی با شبکه عصبی کانولوشنی به لینک‌های زیر مراجعه کنید. در تمامی لینک‌ها مطالب مشترک است اما در هر کدام از لینک‌ها بیان‌های مختلف با جزئیات متفاوت وجود دارد. شما می‌توانید هر توضیحی که برایتان راحت‌تر است را مطالعه کنید. از جست‌وجوهای شخصی خودتان نیز دست برندارید.

شبکه عصبی کانولوشنی توضیح فارسی لینک ۱

شبکه عصبی توضیح فارسی لینک ۲

شبکه عصبی کانولوشنی توضیح انگلیسی لینک ۳

در لینک زیر درباره تفاوت لایه کانولوشنی یک، دو و سه بعدی می‌توانید مطالعه کنید.

لایه کانولوشنی 1D، 2D و 3D و کاربرد هر یک

در لینک زیر درباره جزئیات بیشتر فیلتر و کرنل و تفاوت این دو در لایه‌های کانولوشنی می‌توانید مطالعه کنید. هنگام پیاده‌سازی نیاز دارید که با عملکرد دقیق فیلتر، کرنل و تفاوت این دو مفهوم آشنا باشید.

فیلتر، کرنل و تفاوت آن‌ها در لایه کانولوشنی

زمانی که با مطالعه لینک‌هایی که در بالا قرار داده شده و جست‌وجوهای خود مفهوم لایه‌های مختلف شبکه کانولوشنی را یاد گرفتید حتما به لینک زیر مراجعه کنید و جزئیات بیشتر درباره لایه‌های این شبکه را بخوانید. شما برای نوشتن یا تغییر کد لایه‌های مختلف باید ابعاد ورودی و خروجی هر لایه را بدانید با خواندن مطالب این لینک ابعاد ورودی خروجی‌ها و جزئیات بیشتر هر لایه مشخص می‌شود.

*** اگر نیاز به دانستن جزئیات بیشتر ندارید به عبارتی دیگر صرفا در حال آشنایی با شبکه‌های عصبی و کاربردهای آن‌ها هستید و در مرحله پیاده‌سازی نیستید، از مطالب لینک زیر گذر کنید.

جزئیات بیشتر لایه‌های مختلف شبکه کانولوشنی

۳- کاربردهای مختلف شبکه عصبی

هنگام مطالعه مقاله‌های مختلف، یادگیری و جست‌وجوی خود ممکن است با کاربردهای مختلف شبکه عصبی روبرو شوید. بهتر است اطلاعات کلی از کاربردهای مختلف شبکه عصبی داشته باشید که مطالبی که متناسب با کاربرد مد نظر شما هست را انتخاب و مطالعه کنید.

در لینک زیر کاربردهای مختلف شبکه عصبی به صورت خلاصه توضیح داده شده است. پیش‌تر هم این لینک برای توضیح شبکه عصبی و بخش‌های مختلف آن قرار داده شده است. به ادامه مطالب این لینک مراجعه کنید، کاربردهای مختلف شبکه عصبی را مطالعه کنید.

کاربردهای مختلف شبکه عصبی

۴- یادگیری در شبکه عصبی

در ابتدا باید به دو روش یادگیری با ناظر[۱۵] و یادگیری بدون ناظر[۱۶] در شبکه‌های عصبی اشاره شود. در روش یادگیری نظارت‌شده یا با ناظر برچسب‌های آماده شده برای داده‌ها به یادگیری مدل کمک می‌کنند. در مقابل در روش یادگیری بدون ناظر مدل بدون استفاده از هر گونه اطلاعات اضافی و فقط با در اختیار داشتن داده‌ها، به تشخیص انواع گروه‌های داده‌ها می‌پردازد. مسائل رگرسیون و دسته‌بندی‌داده‌ها در دسته‌ی یادگیری نظارت‌شده قرار می‌گیرند.

برای آموزش یک شبکه عصبی کل داده‌های اولیه را به دو مجموعه داده با نام مجموعه داده آموزشی[۱۷] و مجموعه داده آزمایشی[۱۸] تقسیم می‌کنند. سپس مدل با استفاده از مجموعه داده آموزشی و الگوریتم یادگیری مناسب آموزش می‌بیند و توسط مجموعه داده آزمایشی مورد ارزیابی قرار می‌گیرد. منظور از آموزش دیدن یک شبکه عصبی تنظیم پارامتر وزن و بایاس گره‌ها در آن شبکه است. پس از آموزش یک شبکه عصبی وزن‌ها و بایاس‌ها باید به شکلی تنظیم شده باشند که مدل با دقت بالایی خروجی مورد نظر را تولید کند.

۴-۱- الگوریتم پس‌انتشار

انتشار رو به عقب خطاها که به اختصار پس‌انتشار[۱۹] نامیده می‌شود، الگوریتمی برای یادگیری نظارت‌شده در شبکه عصبی است. این الگوریتم دارای دو مرحله کلی است. مرحله‌ی اول انتشار پیش‌خور[۲۰] و مرحله‌ی دوم پس‌انتشار می‌باشد.

---

[15] Supervised Learning

[16] Unsupervised Learning

[17] Train Dataset

[18] Test Dataset

[19] Backpropagation

[20] Feed forward

مرحله انتشار پیش‌خور: در این مرحله داده آموزش در شبکه منتشر می‌شود و از لایه‌های مختلف شبکه عبور می‌کند. در نهایت شبکه یک خروجی برای ورودی داده شده محاسبه می‌کند. این مرحله همان مرحله اینفرنس است که با آن آشنا هستید.

مرحله پس‌انتشار: هنگام شروع این مرحله باید اختلاف خروجی به دست‌آمده از مرحله انتشار پیش‌خور با مقدار مطلوب شبکه برای داده ورودی توسط یک تابع هزینه یا تابع زیان[۲۱] به عنوان خطای شبکه محاسبه شود. سپس گام به گام از آخرین لایه به سمت اولین لایه شبکه حرکت می‌کنیم تا به‌روزرسانی وزن‌ها و بایاس انجام شود. تابع بهینه‌سازی که در این قسمت انتخاب می‌شود مشخص می‌کند که به روزرسانی وزن‌ها و بایاس از طریق خطای محاسبه شده در گام قبلی به چه شکلی انجام و با چه فرمولی وزن و بایاس جدید محاسبه شود. با هر بار انجام این مراحل، پارامترها به مقدار مناسب خود نزدیک‌تر می‌شوند تا جایی که شبکه خطای بسیار کمی داشته باشد. به عبارتی دیگر الگوریتم پس‌انتشار، حرکت قدم به قدم به سمت کمترین مقدار خطا است.

به توابعی که هایلایت شده و توضیح عملکرد هر یک دقت کنید. در مطالعات خود به یک سری توابع برخورد خواهید کرد که تابع هزینه یا زیان هستند و نحوه محاسبه خطا را توسط یک فرمول مشخص تعیین می‌کنند. توابع دیگری تحت عنوان توابع بهینه‌سازی هستند که این موضوع را مشخص می‌کنند که در برگشت با استفاده از خطای محاسبه شده توسط تابع هزینه، با چه فرمولی هر وزن و بایاس به‌روزرسانی و مقدار جدید محاسبه شود.

نام این الگوریتم یادگیری پس‌انتشار است پس به خاطر داشته باشید حرکت و به‌روزرسانی وزن‌ها و بایاس از انتهای شبکه به سمت ابتدای شبکه انجام می‌شود.

نام چند نمونه از توابع هزینه مشهور در ادامه آمده است.

نام چند نمونه از توابع هزینه (Cost/Loss function)

- Mean squared
- Cross-entropy
- Hinge
- Softmax

[21] Loss Function

چند نمونه از توابع بهینه‌سازی مشهور در ادامه نام برده شده است.

- SGD
- SGD+Momentum
- RMSprop
- Adagrad
- Adadelta
- Adam

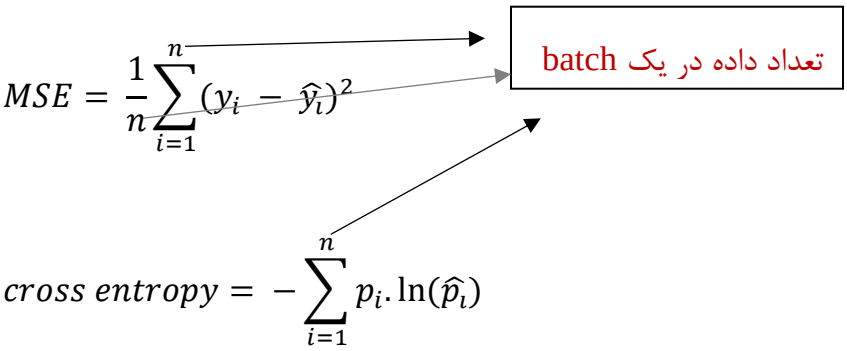
هر کدام از این توابع برای کاربرد خاصی از شبکه مورد استفاده قرار می‌گیرند به عنوان مثال از Mean Squared برای مسائل رگرسیون و از Cross_entropy برای مسائل طبقه‌بندی استفاده می‌شود. پس به خاطر داشته باشید که از هر تابع برای هر کاربرد دلخواهی نمی‌توان استفاده کرد زیرا فرمول هر کدام با توجه به کاربردی که در آن قرار است استفاده شوند نوشته شده است. برای مطالعه درباره فرمول‌های هر کدام از توابع و موارد استفاده از آن‌ها با توجه به نیاز خود باید جست‌وجو کنید.

۴-۲- مفاهیم دیگر هنگام آموزش شبکه عصبی

هنگام کار در حوزه آموزش شبکه عصبی و الگوریتم‌های مربوط به آن به مفاهیمی برخواهیم خورد که برای درک بهتر نحوه کار توابع همچنین برای پیاده‌سازی صحیح سخت‌افزاری باید با آن‌ها آشنا شویم.

Batch: یکی از پارامترهایی که هنگام نوشتن کد شبکه به صورت نرم‌افزاری باید تعیین گردد مفهوم batch یا دسته است. داده‌ها هنگام ورود به شبکه به صورت دسته دسته وارد می‌شوند و هنگام کدنویسی اندازه این دسته توسط طراح شبکه باید مشخص شود. هنگام آموزش شبکه عصبی به‌روزرسانی پارامترها به تعداد دسته‌ها خواهد بود و به تعداد نمونه‌های موجود در مجموعه داده‌ی آموزش نیست. برای مثال فرض کنید ۲۰۰ نمونه در مجموعه داده‌ی آموزش داشته باشیم و آن را به ۴۰ دسته ۵تایی تبدیل کنیم. اگر الگوریتم برگشت به عقب برای یادگیری استفاده شود، هنگام آموزش، پس از هر ۵ نمونه که فرآیند انتشار پیش‌خور را طی کرده‌اند یک بار خطا محاسبه می‌شود و فرآیند بازگشت به عقب و به‌روزرسانی پارامترها به ازای خطای محاسبه شده به دست می‌آید. به این ترتیب پس از هر ۱ دسته یک بار به‌روزرسانی پارامترها را خواهیم داشت. اگر به فرمول توابع هزینه نیز توجه شود مشاهده می‌شود که به ازای تعدادی از نمونه‌ها محاسبه می‌شوند.

در ادامه فرمول دو تابع Mean Squared و Cross_entropy آورده شده که این جمع شدن چند نتیجه و سپس محاسبه‌ی خطا را مشاهده کنید تا مفهوم آنچه گفته شده است را بهتر درک کنید.



Epoch: تعداد دفعاتی که می‌خواهیم آموزش بر روی شبکه انجام شود توسط پارامتر epoch یا دوره تنظیم می‌شود. همان‌طور که پیش‌تر توضیح داده شد با هر بار انجام الگوریتم پس‌انتشار قدم به قدم با تنظیم پارامترهای شبکه خطا کمتر می‌شود. حال اجرای یک بار این الگوریتم بر روی تمام داده‌های مجموعه آموزش برای رسیدن به خطای مورد نظر ما کافی نیست بلکه مثلا باید ۱۰۰ مرتبه این الگوریتم اجرا شود و با هر بار اجرا مقداری از اصلاح پارامترها صورت بگیرد تا در نهایت خروجی شبکه به خطای ناچیزی که که مطلوب ما است برسد. در این حالت با تنظیم پارامتر دوره بر روی عدد ۱۰۰ مشخص می‌شود چند بار الگوریتم آموزش اجرا شود. نکته‌ای که باید در نظر داشت این است که در هر دوره تمامی دسته‌ها وارد شبکه می‌شوند ولی به‌روزرسانی فقط به ازای هر دسته انجام می‌شود. در هر دوره، شبکه با تمامی داده‌های مجموعه آموزش، یادگیری را انجام می‌دهد.

*** دانستن نکته هایلایت شده هنگام پیاده‌سازی سخت‌افزاری مورد نیاز است. هنگام پیاده‌سازی با کد پایتون و به صورت نرم‌افزاری کتابخانه مورد استفاده این عملیات را انجام می‌دهد.

برای مطالعه بیشتر راجع‌به موارد گفته شده می‌توانید به لینک زیر مراجعه کنید.

difference-between-a-batch-and-an-epoch

۵- واژگان پرکاربرد

در این قسمت درباره مفاهیمی که هنگام مطالعه مقالات ممکن است با آن‌ها مواجه شوید توضیح داده شده است.

فراموشی: اگر شبکه عصبی با یادگیری اطلاعات جدید اطلاعاتی که پیش‌تر یاد گرفته بود را فراموش کند گفته می‌شود شبکه دچار فراموشی شده است. منظور از فراموشی اطلاعات قدیمی این است که با ورود داده‌های جدید و اجرای الگوریتم آموزش، پارامترهای شبکه طبق اطلاعات جدید طوری اصلاح شوند و تغییر کنند که شبکه بر روی داده‌های قدیمی و دسته‌بندی آن‌ها مانند گذشته خوب عمل نکند و خطا هنگام تشخیص مجموعه داده‌های آزمایش قدیمی خطا افزایش پیدا کند.

بیش‌برازش[۲۲]: بیش‌برازش هنگامی اتفاق می‌افتد که مدل ویژگی‌های داده‌های آموزشی را به‌جای یادگیری، حفظ کرده باشد. به عبارتی دیگر مدل بیش‌ازحد روی داده‌ها آموزش دیده باشد، درنتیجه این مدل فقط در مجموعه داده‌های آموزشی مفید خواهد بود و در مجموعه داده‌های دیگر که هنوز آن‌ها را ندیده است عملکرد مطلوبی ندارد. وقتی مدل بیش‌ازحد آموزش ببیند، نمی‌تواند به‌خوبی تعمیم یابد و به‌جز داده‌های آموزشی، نمی‌تواند پیش‌بینی‌های خوبی انجام دهد.

محوشدگی گرادیان[۲۳]: هرچقدر تعداد لایه‌های شبکه عصبی بیشتر باشد، احتمال اینکه مقدار گرادیان تابع هزینه به صفر نزدیک‌تر شود، بیشتر می‌شود. در پی این اتفاق، فرآیند یادگیری شبکه عصبی با دشواری روبه‌رو می‌شود زیرا مقادیر وزن و بایاس لایه‌های ابتدایی هنگام به‌روزرسانی تغییر چندانی نمی‌کنند و یادگیری به درستی انجام نمی‌شود. (این در مواردی صادق است که تابع بهینه‌سازی انتخاب شده از گرادیان در محاسبات خود استفاده می‌کند.)

انفجار گرادیان[۲۴]: مسئله انفجار گرادیان در پی رشد نمایی و ضرب مکرر مقادیر گرادیان‌های بزرگ رخ می‌دهد. در شبکه های عمیق ممکن است مقادیر گرادیان خطا روی هم انباشته شوند و مقادیر گرادیان خیلی بزرگ شوند. این مسئله باعث می‌شود مقدار به‌روزرسانی وزن‌ها بسیار بزرگ شود که همین امر منجر به ناپایداری شبکه خواهد

[22] Overfitting

[23] Vanishing Gradient

[24] Exploding Gradients

شد در این حالت یادگیری شبکه متوقف شده و وزن‌های شبکه به‌روزرسانی نمی‌شوند. (این در مواردی صادق است که تابع بهینه‌سازی انتخاب شده از گرادیان در محاسبات خود استفاده می‌کند.)

یادگیری انتقالی[۲۵]: تکنیک یادگیری انتقالی از دانش مدلی که از قبل برای وظیفه‌ای دیگر آموزش دیده شده است برای حل وظیفه‌ای دیگر استفاده می‌کند. قطعاً این دو وظیفه تا حد زیادی مشابه یکدیگر هستند. به عبارت ساده‌تر یک شبکه قبلا با یک دیتاست بزرگ آموزش دیده است و مقدار وزن و بایاس آن پس از آموزش مشخص شده‌اند. حال شما در صورتی که شبکه‌ای داشته باشید که ساختار لایه‌ای مشابه شبکه آموزش‌دیده داشته باشد و دیتاستی تقریبا مشابه دیتاستی که مدل اولیه بر روی آن آموزش دیده است داشته باشید، می‌توانید از مقادیر وزن و بایاسی که از شبکه آموزش‌دیده قبلی به دست آمده بر روی مدل خود استفاده کنید و نیاز به آموزش مجدد توسط دیتاست بزرگ نباشد. (طبیعتا دیتاست‌ها باید شبیه به یکدیگر باشند تا شبکه شما با استفاده از دانش شبکه آموزش‌دیده اولیه بتواند بر روی دیتاست جدید عملکرد خوبی داشته باشد.)

Batch Normalization: نرمال‌سازی دسته‌ای یک تکنیک است که روی ورودی لایه‌های شبکه عصبی اعمال می شود و از طریق تغییر مرکز توزیع داده‌ها یا تغییر دادن مقیاس آن‌ها موجب سریع‌تر و پایدارتر شدن شبکه عصبی می شود. رابطه‌های زیر نحوه محاسبه‌ی نرمال هر دسته را نمایش می‌دهد.

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$$

$$\hat{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2}}$$

$$y_i = \gamma\hat{x_i} + \beta$$

[25] Transfer Learning

همان‌طور که از نام تکنیک مشخص است این عملیات بر روی داده‌های هر دسته یا batch انجام می‌شود. m تعداد داده‌های هر دسته را نشان می‌دهد. با استفاده از رابطه‌ی اول میانگین یک دسته محاسبه می‌شود. با استفاده از رابطه‌ی دوم واریانس محاسبه می‌شود. با استفاده از رابطه‌ی سوم نرمال هر داده‌ی دسته به دست می‌آید. این رابطه فرمول پایه محاسبه‌ی نرمال است. در نهایت رابطه‌ی چهارم مقیاس و انتقال را انجام می‌دهد.

برای مطالعه بیشتر راجع به BN می‌توانید به لینک‌های زیر مراجعه کنید. به یاد داشته باشید که از جست‌وجوهای شخصی خودتان نیز دست برندارید.

لایه BN لینک ۱

لایه BN لینک ۲

Dropout: dropout یا حذف تصادفی تکنیکی است که هنگام آموزش، هربار از تعدادی از گره‌ها چشم‌پوشی می‌شود. گره‌های نادیده گرفته شده در آموزش در نظر گرفته نمی‌شوند، یعنی در انتشار پیش‌خور در محاسبات شرکت داده نمی‌شوند و در بازگشت، مقدار آن‌ها به‌روزرسانی نمی‌شود. به این ترتیب در هر بار آموزش با لایه متفاوتی محاسبات انجام می‌شود.

برای مطالعه راجع به انواع شبکه‌های کانولوشنی عمیق مانند Alexnet، VGG و ... می‌توانید به لینک زیر مراجعه کنید.

معرفی مدل‌های مختلف شبکه عصبی کانولوشنی عمیق

برای مطالعه راجع‌به تابع softmax می‌توانید به لینک زیر مراجعه کنید.

softmax

برای درک مفهوم تابع cross entropy می‌توانید از مطالب لینک زیر استفاده کنید.

cross entropy

در لینک زیر توضیحات کامل به همراه فرمول‌های ریاضی الگوریتم backpropagation زمانی که تابع هزینه cross entropy و تابع بهینه‌سازی SGD باشد آورده شده است. مشتق زنجیره‌ای در مطالب این لینک محاسبه شده است.

how-does-back-propagation-work-in-neural-networks-with-worked-example

هنگام مطالعه مقالات با نام دیتاست‌ها یا مجموعه داده‌های مختلف مواجه می‌شوید. در لینک زیر چند نمونه از از این مجموعه داده‌ها نام برده و توضیح داده شده است که برای آشنایی می‌توانید به آن مراجعه کنید.

چند نمونه از مجموعه داده‌های تصویری

۶- فیلم‌ها و لینک‌های آموزشی

برای جمع‌بندی مطالبی که پیش‌تر مطالعه کردید و یادگیری مفاهیمی مانند learning rate یا نرخ یادگیری به فیلم آموزشی که در لینک زیر قرار دارد مراجعه کنید. مدت این دوره ۵ ساعت است.

آموزش شبکه عصبی و شبکه کانولوشنی

پس از اینکه همه مفاهیم را یاد گرفتید برای آشنایی با نحوه استفاده از کتابخانه تنسورفلو و کراس برای پیاده‌سازی و آموزش یک شبکه عصبی می‌توانید به آموزش‌های زیر مراجعه کنید. اگر مطالب گفته شده را دنبال کرده باشید تمام آرگومان‌هایی که هنگام استفاده از کتابخانه‌ها در کدنویسی پایتون استفاده می‌کنید می‌شناسید و دلیل تعریف هر کدام را می‌دانید. برای این موضوع آموزش‌های زیادی در یوتیوب وجود دارد که می‌توانید از آن‌ها استفاده کنید.

آموزش کدنویسی شبکه عصبی با استفاده از کتابخانه تنسورفلو و کراس

کدنویسی شبکه عصبی با تنسورفلو و کراس لینک youtube

کدنویسی شبکه عصبی با تنسورفلو و کراس لینک youtube

اگر به مفاهیم زبان برنامه‌نویسی پایتون آشنا نیستید، می‌توانید ابتدا به فیلم آموزشی زبان پایتون که بر روی یکی از سیستم‌های آزمایشگاه موجود است مراجعه کنید.

برای آشنایی با نحوه کار با گوگل کلب می‌توانید به لینک‌های زیر مراجعه کنید.

گوگل کلب

گوگل کلب

گوگل کلب

برای آشنایی با نحوه کدنویسی شبکه عصبی با استفاده از کتابخانه پای‌تورچ و نوشتن کد چند شبکه کانولوشنی معروف می‌توانید به لینک زیر مراجعه کنید.

آموزش کدنویسی شبکه عصبی با استفاده از کتابخانه پای‌تورچ

برای تمام مفاهیم مربوط به یادگیری ماشین، شبکه عصبی و تمام الگوریتم‌های این حوزه در سایت medium که لینک آن در ادامه قرار داده شده است مطالب بسیار مختصر، واضح و کاربردی وجود دارد.

لینک سایت برای جست‌وجو مفاهیم یادگیری ماشین و شبکه عصبی

در لینک زیر کد الگوریتم backpropagation به زبان C وجود دارد.

backpropagation

در لینک زیر نحوه دسترسی به وزن و بایاس هر لایه برای ذخیره کردن (علاوه بر ذخیره به صورت HDF5 file) وجود دارد.

how-to-correctly-get-layer-weights-from-conv2d-in-keras

در لینک زیر کد پایتونی وجود دارد که در آن با استفاده از روشی که در لینک قبلی گفته شده است، وزن و بایاس بررسی و به صورت فایل txt ذخیره شده است.

Getting-layer-weights-and-bias

اگر علاقه‌مند به مطالعه عمیق و با جزئیات درباره شبکه‌های عصبی و یادگیری ماشین بودید می‌توانید به آموزش‌های لینک زیر مراجعه کنید.

آموزش‌های حوزه شبکه عصبی و یادگیری ماشین با جزئیات و عملیات‌های ریاضی

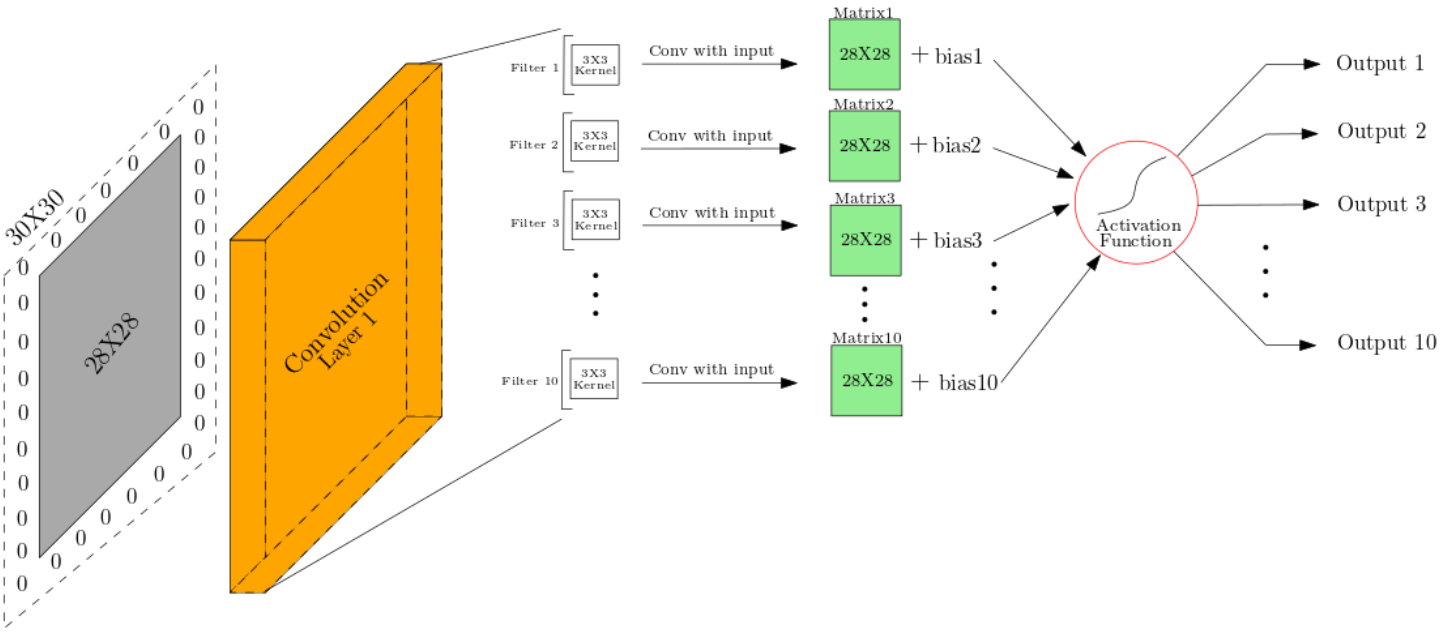
۷- پیاده‌سازی سخت‌افزاری

اگر وارد فاز پیاده‌سازی شده‌اید مطالب این قسمت را بخوانید. برای پیاده‌سازی نرم‌افزاری دانستن این مطالب مورد نیاز نیست.

۷-۱- عملکرد لایه کانولوشنی

اگر مطالب مربوط به فیلتر و کرنل را مطالعه کرده باشید، می‌دانید که هر فیلتر شامل چند کرنل است. همچنین می‌دانید که هر ورودی کرنل مخصوص به خودش را دارد. برای روشن شدن مطلبی که قرار است راجع‌به آن توضیح دهیم به این مثال دقت کنید. یک شبکه را در نظر بگیرید که از ۳ لایه کانولوشنی تشکیل و ۳ لایه مکس‌پولینگ تشکیل شده است. در ادامه این لایه‌ها نیز سه لایه تمام متصل قرار دارند. لایه اول و دوم تمام متصل دارای ۷۰ نورون و لایه آخر ۱۰ نورون دارد. لایه اول کانولوشنی دارای ۱۰ فیلتر، لایه دوم دارای ۱۵ و لایه سوم دارای ۲۰ فیلتر است. در کد نرم‌افزاری این شبکه کرنل‌ها ۳×۳ تعریف شده‌اند. ورودی این شبکه فرضی دارای ابعاد ۲۸×۲۸ می‌باشد و یک ماتریس ورودی داریم. به عبارتی دیگر ورودی ۱×۲۸×۲۸ است. ابعاد ماتریس با تعداد ماتریس دو موضوع متفاوت است. تعداد ماتریس ورودی را بعد سوم مشخص می‌کند. تفاوت ابعاد و تعداد را می‌توان هنگام گذر از لایه مکس‌پولینگ متوجه شد به این صورت که با گذر از هر لایه مکس‌پولینگ با stride دو، ابعاد ماتریس نصف می‌شود اما تعداد ماتریسی که وارد می‌شود برابر با تعداد ماتریس خروجی است.

همان طور که بیان شد یک ماتریس ورودی ۲۸×۲۸ که دارای padding یک است ورودی شبکه فرضی ما است. با در نظر گرفتن یک لایه padding می‌توان گفت ابعاد ماتریس ۳۰×۳۰ خواهد شد. این ماتریس وارد لایه اول کانولوشنی می‌شود که دارای ۱۰ فیلتر است. هر فیلتر با لغزیدن بر روی ماتریس ورودی یک ماتریس جداگانه با ابعاد ۲۸×۲۸ تولید می‌کند (اگر stride لایه کانولوشنی دو باشد ابعاد ماتریس ورودی نصف می‌شود، اما در این مثال stride یک است و فقط لایه padding اطراف ماتریس از بین می‌رود.). با توجه به این توضیحات خروجی لایه اول کانولوشنی ۱۰ ماتریس ۲۸×۲۸ است. هر کدام از این ۱۰ ماتریس پس از اینکه تولید شدند با بایاس آن فیلتر جمع می‌شوند و از تابع فعال‌ساز می‌گذرند سپس خروجی نهایی تولید می‌شود. این فرآیند در شکل ۷-۱ نمایش داده شده است.



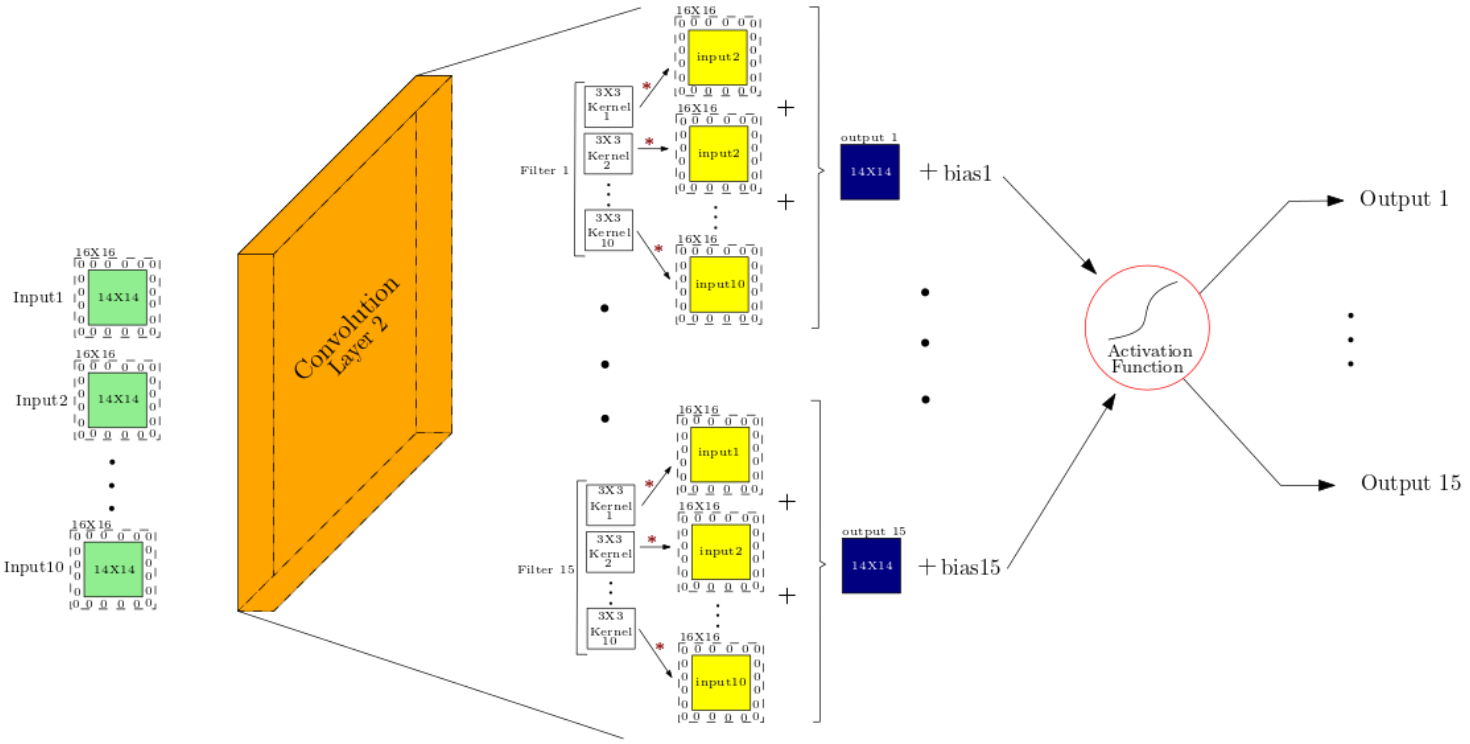
شکل۷-۱ لایه کانولوشنی اول دارای ۱۰ فیلتر

حال این ۱۰ ماتریس ۲۸×۲۸ وارد لایه مکس‌پولینگ می‌شوند. خروجی این لایه ۱۰ ماتریس ۱۴×۱۴ خواهد بود. این ۱۰ ماتریس پس از padding دارای ابعاد ۱۶×۱۶ خواهند بود.

این ۱۰ ماتریس ۱۶×۱۶، ورودی لایه کانولوشنی دوم خواهد بود. همان طور که در فرض ابتدای توضیحات بیان شد لایه کانولوشنی دوم ۱۵ فیلتر دارد. پیش‌تر بیان شد که هر ورودی، کرنل مخصوص به خود را دارد. پس هر فیلتر برای هر ورودی یک کرنل متفاوت دارد. به عبارتی دیگر هر فیلتر در این لایه ۱۰ کرنل متفاوت به ازای هر ورودی این لایه دارد. در مجموع در این لایه ۱۵۰ کرنل (۱۵ فیلتر هر کدام ۱۰ کرنل (متناسب با ورودی لایه)) یا ماتریس ۳×۳ برای لغزیدن بر روی ورودی‌ها وجود دارد. در این لایه نیز مانند گذشته هر فیلتر در نهایت یک خروجی خواهد داشت. به عبارتی دیگر تعداد ماتریس خروجی این لایه ۱۵ عدد خواهد بود. هر کدام از این ۱۵ ماتریس چگونه تولید می‌شود؟

بیایید نحوه تولید خروجی یک فیلتر را در نظر بگیریم. به طور مثال می‌خواهیم بدانیم فیلتر اول چگونه خروجی تولید می‌کند. طبق توضیحات بیان شده فیلتر ۱ دارای ۱۰ کرنل یا واضح‌تر بگوییم دارای ۱۰ ماتریس ۳×۳ است. می‌دانیم که لایه کانولوشنی دوم دارای ۱۰ ماتریس ورودی ۱۶×۱۶ است. هر کدام از ورودی‌ها در یکی از کرنل‌های فیلتر ۱ ضرب می‌شود و یک ماتریس خروجی تولید می‌شود. به این ترتیب ۱۰ ماتریس خروجی از ضرب هر کرنل

در ورودی تولید شده است. حال برای تولید خروجی فیلتر ۱ تمامی ۱۰ ماتریس با یکدیگر جمع می‌شوند و یک ماتریس را تشکیل می‌دهند. این روند برای فیلتر دوم، سوم تاپانزدهم تکرار می‌شود و هر فیلتر یک ماتریس خروجی ۱۴×۱۴ تولید می‌کند. به همین خاطر خروجی لایه دوم کانولوشنی ۱۵ ماتریس ۱۴×۱۴ خواهد بود. مانند لایه کانولوشنی قبلی هر کدام از این ۱۵ ماتریس پس از اینکه تولید شدند با بایاس آن فیلتر جمع می‌شوند و از تابع فعال‌ساز می‌گذرند سپس خروجی نهایی تولید می‌شود. هر لایه کانولوشنی به تعداد فیلترهایش مقدار بایاس متفاوت دارد. لایه کانولوشنی دوم و عملیات مربوط به آن در شکل ۷-۲ نمایش داده شده است.

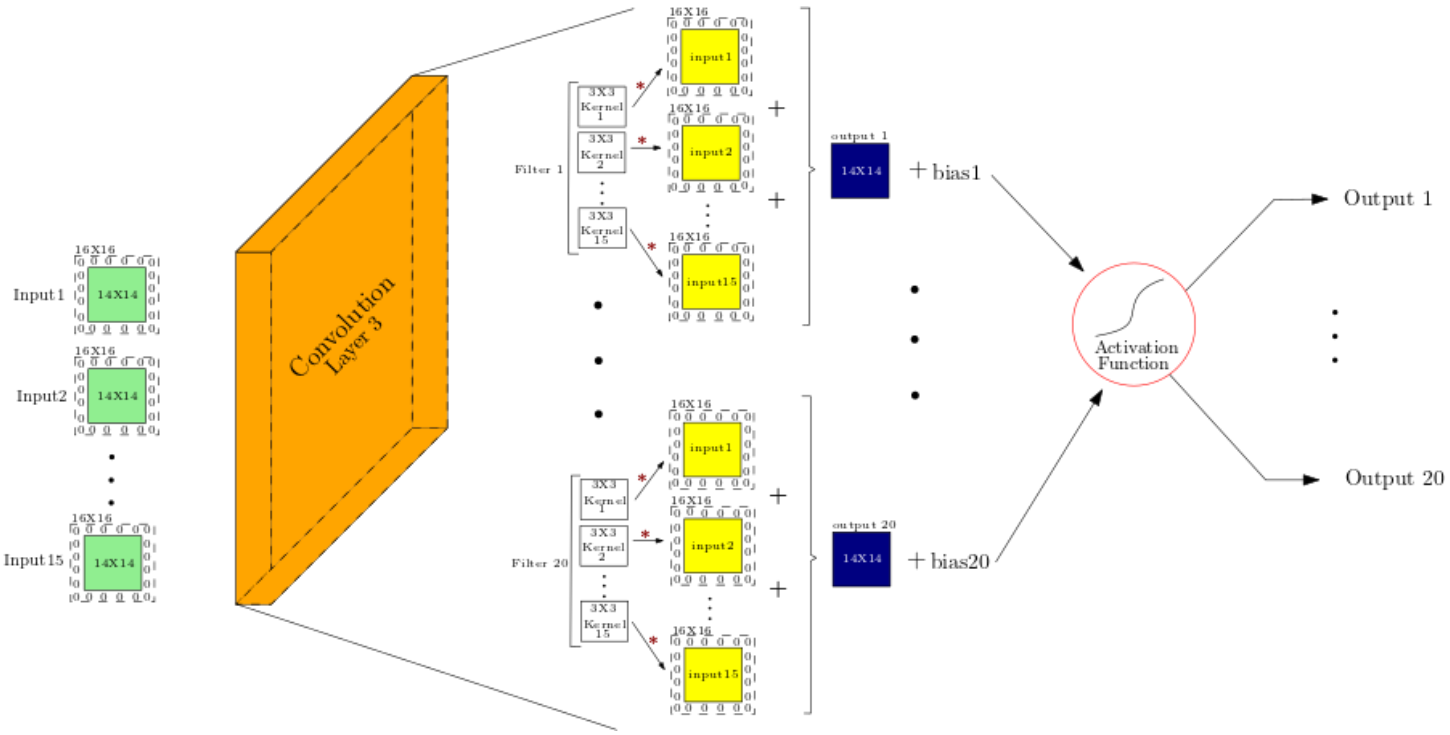


شکل۷-۲ لایه کانولوشنی دوم دارای ۱۵ فیلتر

این ۱۵ ماتریس نیز مانند گذشته پس از عبور از لایه مکس‌پولینگ و padding به ۱۵ ماتریس ۹×۹ تبدیل می‌شوند که ورودی لایه سوم کانولوشنی هستند.

با توجه به توضیحات بیان شده می‌توان حدس زد که تعداد کل کرنل‌های این لایه ۳۰۰ عدد است (۲۰ فیلتر هر کدام دارای ۱۵ کرنل ۳×۳). مانند لایه قبل در این لایه هم کرنل‌های هر فیلتر در ورودی مربوط به خود ضرب می‌شوند و سرانجام حاصل ضرب هر ۱۵ کرنل یک فیلتر در ورودی، با هم جمع می‌شوند و خروجی یک فیلتر به

دست می‌آید. در این لایه در نهایت پس از جمع با بایاس و گذر از تابع فعال‌سازی ۲۰ ماتریس ۷×۷ به عنوان خروجی تولید می‌شود. لایه کانولوشنی سوم و عملکرد این لایه در شکل ۷-۳ نمایش داده شده است.



شکل۷-۳ لایه کانولوشنی سوم دارای ۲۰ فیلتر

این ۲۰ ماتریس مجددا از لایه مکس‌پولینگ و padding گذر می‌کنند و در نهایت ۲۰ ماتریس ۳×۳ به دست می‌آید که وارد لایه تمام متصل می‌شوند.

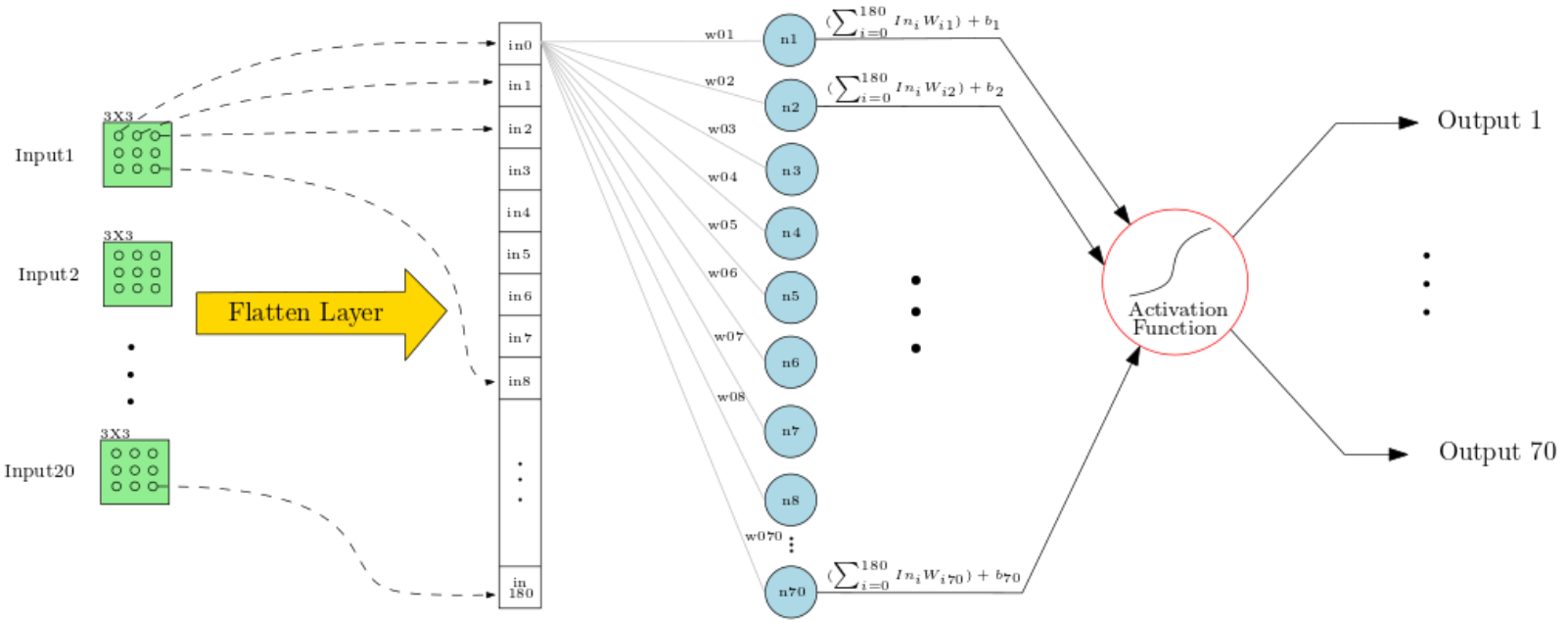
در ادامه شکل سه لایه کانولوشنی توضیح داده شده و عملیاتی که هر کدام انجام می‌دهند، قرار دارد.

۷-۲- عملکرد لایه تمام متصل

۲۰ ماتریس ۳×۳ توسط flatten به آرایه تبدیل می‌شود. به عبارتی دیگر یک آرایه ۱۸۰تایی (۲۰×۳×۳) به وجود می‌آید. به دلیل اینکه لایه تمام متصل اول دارای ۷۰ نورون است هر کدام از اعضای این آرایه به ازای هر نورون این لایه، وزن متفاوتی دارد که در آن وزن ضرب می‌شود و فرمول مربوط به لایه تمام متصل بر روی آن

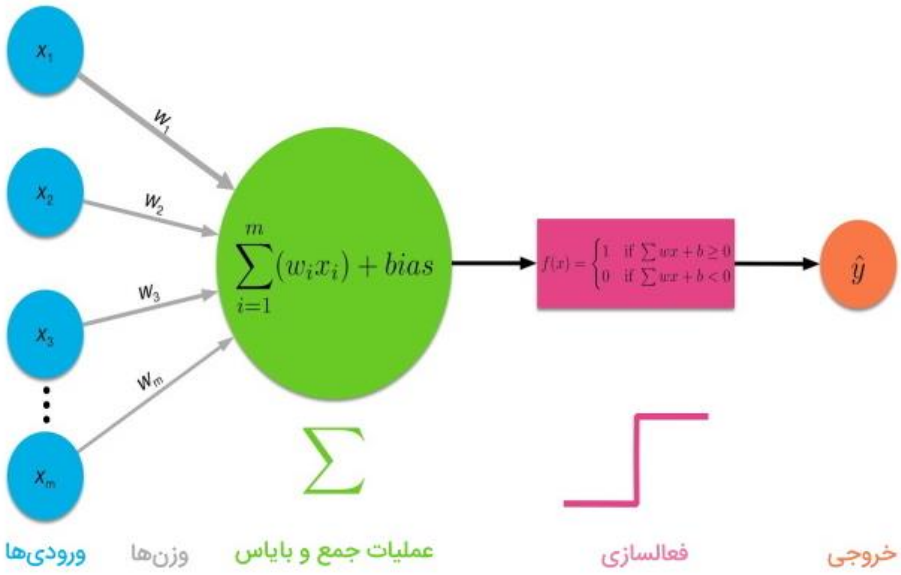
اجرا می‌شود. با این توضیح می‌توانید تصور کنید که ماتریس وزن این لایه یک ماتریس ۷۰×۱۸۰ است. تصور کنید در کنار هر سطر ماتریس یکی از اعضای آرایه ۱۸۰تایی ورودی قرار دارد و بالای هر ستون یکی از نورون‌های این لایه که مقدار وزن مخصوص آن نورون به ازای هر کدام از اعضای آرایه در آن ستون آورده شده است. به عبارت دیگر هر سطر شامل وزن‌های مخصوص یک ورودی به ازای هر کدام از نورون‌های لایه است.

در این لایه هر نورون یک خروجی تولید خواهد کرد. طبق فرمول عملیات جمع و بایاس که شکل آن در ادامه آورده شده است، مشاهده می‌کنید که هر نورون پس از اینکه وزن مخصوص هر ورودی را در آن ضرب کرد تمامی این مقادیر را با یکدیگر جمع می‌کند در نهایت با بایاس آن نورون جمع انجام می‌شود و خروجی از تابع فعال‌سازی می‌گذرد و تولید می‌شود. در لایه تمام متصل به تعداد نورون‌ها بایاس وجود دارد. در این لایه ۷۰ خروجی تولید می‌شود. عملیات لایه تمام متصل اول در شکل ۷-۴ نمایش داده شده است.



شکل۷-۴ لایه تمام متصل اول دارای ۷۰ نورون

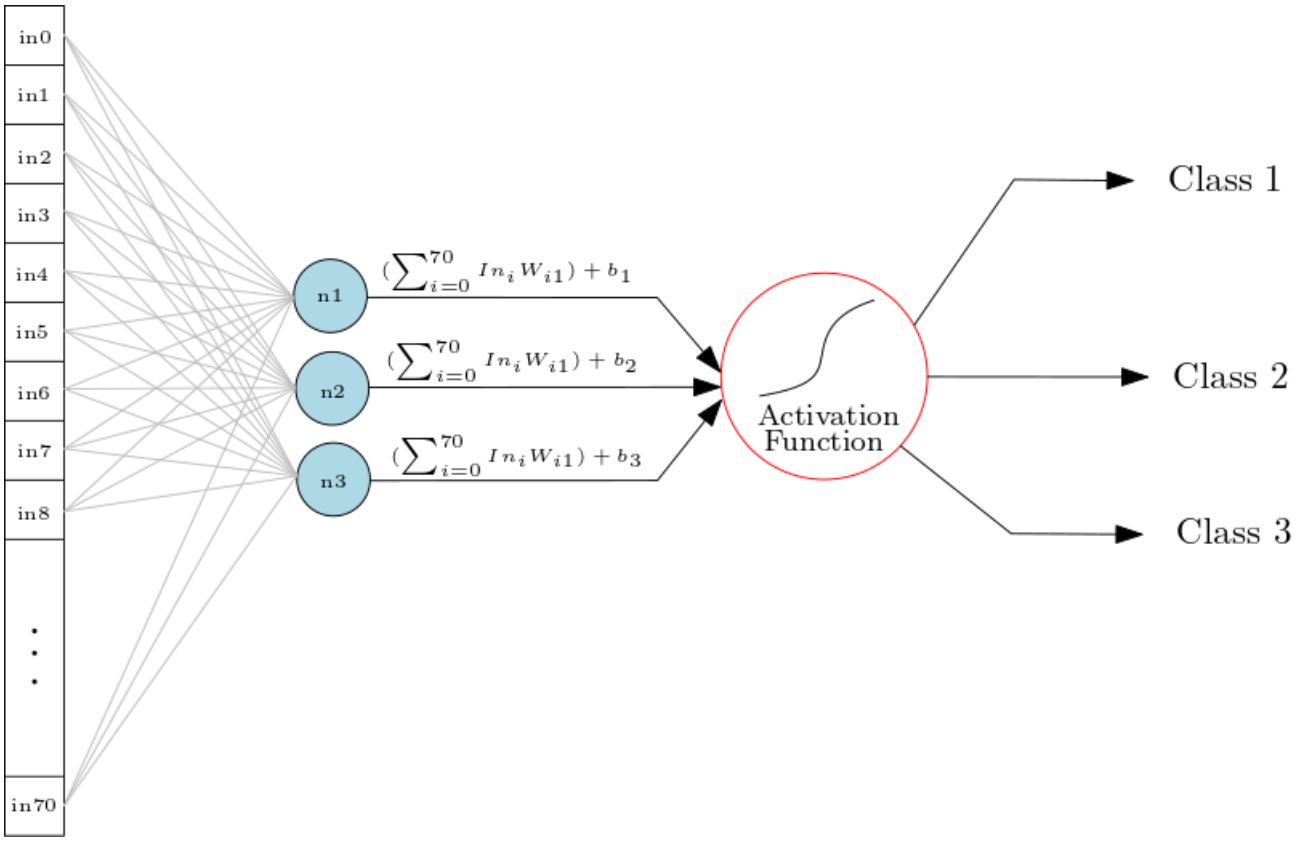
در شکل ۷-۵ نیز عملیات برای یک نورون نمایش داده شده است.



شکل۷-۵ عملیات مربوط به یک نورون در لایه تمام متصل [۱]

با توجه به اینکه لایه تمام متصل دوم ورودی ۷۰ تایی دارد و خودش دارای ۷۰ نورون است، ماتریس وزن‌های این لایه دارای ابعاد ۷۰×۷۰ خواهد بود. عملکرد این لایه مانند لایه تمام متصل اول خواهد بود و ۷۰ خروجی تولید می‌کند.

لایه تمام متصل سوم دارای ۳ نورون است و ورودی ۷۰ تایی را می‌گیرد و به این ترتیب دارای ماتریس وزن ۳×۷۰ خواهد بود. این لایه در نهایت ۳ خروجی تولید خواهد کرد که نشان‌دهنده ۳ کلاس متفاوت برای طبقه‌بندی است. عملیات این لایه در شکل ۷-۶ نمایش داده شده است.



شکل۷-۶ لایه تمام متصل سوم دارای ۳ نورون

توضیحاتی که در این بخش درباره جزئیات عملکرد لایه کانولوشنی، لایه‌های تمام متصل و سایر لایه‌های شبکه عصبی کانولوشنی داده شد را می‌توانید در لینک زیر بررسی کنید.

نحوه کار لایه‌های مختلف شبکه کانولوشنی

در لینک زیر نمونه‌ای از پیاده‌سازی لایه‌های کانولوشنی، تمام متصل، max pooling و بقیه لایه‌ها و تکنیک‌های مربوط به شبکه عصبی کانولوشنی به زبان VHDL وجود دارد.

شبکه عصبی کانولوشنی با VHDL

۷-۳- فیلم‌ها و لینک‌های آموزشی و کاربردی برای پیاده‌سازی بر روی FPGA

آموزش مربوط به Vitis AI

vitis AI

گیت هاب vitis AI

توضیحاتی راجع‌به یک روش پیاده‌سازی کرنل

CNN Kernel implementation in VHDL

توضیحات این ویدیو و کانال یوتیوب برای دید کلی پیدا کردن نسبت به نحوه پیاده‌سازی می‌تواند مفید باشد.

Machine Learning on FPGAs

نمونه کد مربوط به پیاده‌سازی CNN است اما درستی عملکرد آن بررسی نشده است. در صورت استفاده از آن باید نحوه پیاده‌سازی لایه‌ها نیز بررسی شود تا بتوانید طبق نیاز شبکه خود پارامترهای آن را تغییر دهید.

نمونه کد cnn در گیت هاب

این نمونه کد مربوط به پیاده سازی softmax است. در این پیاده سازی از LUT استفاده نشده است و عملیات اعشاری نوشته شده است. در صورت استفاده، صحت عملکرد کد و نحوه نوشتن کد باید بررسی شود.

نمونه کد softmax در گیت هاب

توضیحات محاسبات ریاضی در FPGA در لینک‌های زیر وجود دارد.

پیاده سازی تابع لگاریتم در FPGA

مبانی محاسبات ریاضی در FPGA (قسمت اول)

مبانی محاسبات ریاضی در FPGA (قسمت دوم)

اعداد اعشاری ممیز ثابت (بخش اول: مفاهیم کلی)

اعداد اعشاری ممیز ثابت (بخش دوم: محاسبات با دقت محدود)

اعداد اعشاری ممیز ثابت (بخش سوم: قوانین پایه محاسبات)

کتابخانه‌هایی برای تبدیل اعداد از فلوت به فیکس پوینت در لینک زیر قرار دارند.

کتابخانه تبدیل فلوت به فیکس لینک ۱

کتابخانه تبدیل فلوت به فیکس لینک ۲

برای استفاده از fixed point در کد VHDL هم می‌توانید از کتابخانه Fixed داخل VHDL 2008 یا از کتابخانه‌های جانبی استفاده کنید.

محاسبات اعشاری

کتابخانه محاسبات اعشاری لینک ۱

کتابخانه محاسبات اعشاری لینک ۲

در لینک زیر یک generator شبکه عصبی cnn قرار دارد که از کاربر اطلاعاتی مانند تعداد لایه، تعداد فیلتر، سایز کرنل، سایز ورودی و ... را می‌گیرد و کد RTL لایه‌های مختلف شبکه عصبی را تولید می‌کند. این generator یک سری محدودیت دارد که در لینک قرار داده شده محدودیت‌های آن را نام برده است. خود generator نیز در گیت‌هاب موجود است. می‌توانید از آن استفاده کنید و برای نحوه نوشتن لایه‌های خود از آن ایده بگیرید.

cnn-vhdl-generator

لینک زیر مربوط به پروژه eyeriss است که توسط دانشگاه MIT انجام می‌شود. در این پروژه سعی بر این است که پیاده‌سازی که از نظر انرژی بهینه است برای لایه‌های شبکه کانولوشنی ارائه شود.

eyeriss project

در ویدئو زیر نحوه پیاده‌سازی شبکه عصبی کانولوشنی توسط یک تیم که در رقابت‌های زایلینکس شرکت کرده‌اند توضیح داده می‌شود. دیدن نحوه پیاده‌سازی آن‌ها می‌تواند ایده‌های خوبی برای پیاده‌سازی لایه‌های شبکه به شما بدهد.

FPGA Based Sign Language Interpretation Using Convolutional Neural Networks

۸- زبان VHDL

اگر درباره زبان VHDL جستوجو کنید حتما به دو لینک زیر برخواهید خورد. برای یادگیری، یادآوری یا چک کردن برخی موارد از هر دو لینک می‌توانید استفاده کنید.

VHDL 1

VHDL 2

هدف ما در این قسمت آموزش زبان VHDL نیست. در این بخش فقط نکاتی که بهتر است هنگام آموزش به آن‌ها توجه کنید یادآوری شده است.

همان طور که می‌دانید در زبان VHDL کدها و دستورات برای اجرا شدن پس از کلمه کلیدی begin قرار می‌گیرند. جایی که کدها و دستورات در آن نوشته می‌شوند، Processها در آن تعریف می‌شوند، concurrent نام دارد. در این محیط هر چیزی نوشته شود به صورت همزمان اجرا می‌شود. در محیط concurrent با استفاده از processها می‌توان محیط sequential ساخت. تمام دستوراتی که ماهیت ترتیبی دارند مانند if، while، case و ... در processها قابل استفاده هستند. زمانی که شما چند process در concurrent تعریف می‌کنید همه آن‌ها به طور موازی با یکدیگر کار می‌کنند. با توجه به اینکه گفته شد که در محیط concurrent ترتیبی وجود ندارد و همه چیز همزمان انجام می شود پس جابه‌جایی خطوط و دستورات در آن تغییری ایجاد نمی‌کند و خطوط دستور نسبت به یکدیگر الویت ندارند.

نکته بعدی که بهتر است یادآوری شود این است که اگر انتساب ساده‌ای در محیط concurrent انجام شده باشد زمانی که سمت راست انتساب تغییری کند ارجاع انجام می‌شود. در process نیز زمانی که process به پایان برسد تمام انتسابات با هم انجام می شوند. به عبارتی دیگر شما تنها زمانی درون بدنه process با تغییر یک مقدار همان لحظه می‌توانید از مقدار تغییر یافته آن سیگنال استفاده کنید که آن را از نوع variable تعریف کرده باشید در غیر این صورت تا رسیدن به انتهای process سیگنال مقدار قبلی خود را دارد و هر عملیاتی را با آن مقدار قدیمی انجام می‌دهد نه آن مقداری که جدیدا به آن assign شده است. پس از رسیدن به انتهای process مقداردهی جدید انجام می‌شود.

منظور از انتساب ساده simple assignment است.

```
A <= B
```

نکته بعدی که حتما هنگام نو شتن کد پروژه‌های بزرگ با آن برخورد خواهید دا شت، طراحی ماژولار ا ست. در طراحی ماژولار هر قسمت یک طرح بزرگ به صورت یک ماژول نوشته می‌شود سپس این ماژول‌ها یا زیرماژول‌ها در کنار هم قرا می‌گیرند و ماژول اصلی یا top module را تشکیل می‌دهند. به عنوان مثال برای نو شتن کد یک شبکه ع صبی CNN لایه‌های مختلف آن (لایه کانولو شنی یا لایه تمام مت صل) به صورت یک ماژول نوشته می‌شوند سپس این لایه‌ها در کنار هم قرار می‌گیرند و به شکل صحیحی به هم متصل می‌شوند و شبکه CNN که ماژول اصلی ما است را تشکیل می‌دهند. هنگام طراحی ماژولار دانستن نحوه فراخوانی هر کدام از زیرماژول‌ها در ماژول اصلی و چگونگی اتصال آن‌ها به یکدیگر که اصطلاحا port map نامیده می‌شود، مهم است. برای یادگیری این موضوع می‌توانید "طراحی ماژولار در VHDL"، " اضافه کردن زیرماژول به ماژول اصلی"، "طراحی سلسه مراتبی در VHDL" و ... را جست‌وجو کنید. برای درک این موضوع مطالعه لینک زیر نیز می‌تواند مفید باشد.

نمونه کدنویسی ماژولار

در لینک زیر نیز توضیحاتی درباره کدنویسی با زبان VHDL به زبان فارسی قرار دارد. برای آشنایی کلی می‌توانید به آن مراجعه کنید.

آموزش VHDL

برای آشنایی با نرم‌افزار vivado و نحوه کار کردن با آن می‌توانید از لینک زیر استفاده کنید.

Vivado

برای دسترسی به این گزارش راهنما و به‌روزرسانی‌های بعدی آن می‌توانید به آدرس گیت‌هاب زیر مراجعه کنید.

https://github.com/FarzanehArzaghi

مراجع

[۱] https://faradars.org/