دانشکده مهندسی برق

عنوان پروژه :

رویکرد یادگیری ویژگی تبعیض آمیز

برای تشخیص چهره

نگارش :

مهدی اردستانی

استاد :

دکتر محمد باقر منهاج

مرداد 1400


بسم الله الرحمن الرحيم

چکیده :

شبکه‌های عصبی کانولوشنی به‌طور گسترده در بحث‌های بینایی ماشین استفاده می‌شود. در اکثر این شبکه‌ها تابع هزینه softmax به عنوان سیگنال loss برای آموزش مدل استفاده می‌شود.

به‌منظور بهبود عملکرد جدا پذیری یا تبعیض بیشتر در بین ویژگی‌های یادگرفته شده در مدل، در این پروژه یک سیگنال loss جدید معرفی می‌شود که آن را center loss می‌نامیم و برای کارهای تشخیص چهره از آن استفاده می‌کنیم. Center loss به‌طور همزمان مرکز هر کلاس را بدست می‌آورد و فاصله بین ویژگی‌های موجود در کلاس و مرکز متناظر با آنها را محاسبه کرده و این فاصله را کم می‌کند تا واریانس در کلاس کم شود و ویژگی‌ها به یکدیگر نزدیک شده و به سمت مرکز کلاس همگرا شوند و بدین ترتیب تبعیض بین ویژگی ها را برای طبقه بندی بهینه تر بهبود می‌دهد.

واژه‌های کلیدی :

شبکه‌های عصبی کانولوشنی، بینایی ماشین، تشخیص چهره، center loss

فهرست عناوین صفحه

فهرست اشکال صفحه

فناوری تشخیص چهره روشی مبنی بر فناوری‌های روز دنیا برای شناسایی و راستی آزمایی هویت افراد بر اساس چهره‌ی آن‌ها است. سیستم‌های تشخیص چهره می‌توانند در تصاویر، فیلم‌ها و هم‌چنین به‌صورت زنده و بی‌درنگ[1] استفاده شوند. یک سیستم تشخیص چهره از نشان‌گرهای زیستی برای مدل‌سازی ویژگی‌های چهره در یک عکس یا فیلم بهره می‌برد. این سیستم، اطلاعات پایگاه داده‌ی چهره‌های شناخته شده را با سایر چهره‌ها مقایسه می‌کند تا اینکه به جواب درست برسد. تشخیص چهره می‌تواند در شناسایی هویت افراد کمک کند و از طرف دیگر چالش‌های حریم خصوصی را بالا می‌برد. در ادامه به این سوال به طور کامل پاسخ می‌دهیم که فناوری تشخیص چهره چیست؟ و هم‌چنین با مهم‌ترین روش‌های تشخیص چهره آشنا می‌شویم.

تاریخچه :

تشخیص خودکار چهره در دهه ۱۹۶۰ آغاز شد. وودی بلدسو، هلن چان ولف و چارلز بیسون روی رایانه کار کردند تا چهره انسان را تشخیص دهند. پروژه تشخیص چهره اولیه آنها «انسان-ماشین» لقب گرفت زیرا مختصات ویژگیهای صورت در یک عکس قبل از استفاده توسط رایانه برای شناسایی توسط یک انسان باید تعیین می‌شد. بر روی یک تبلت گرافیکی یک انسان باید مختصات ویژگی‌های صورت مانند مراکز مردمک چشم، گوشه داخلی و خارجی چشم‌ها و خط موها مشخص کند. از مختصات برای محاسبه ۲۰ فاصله از جمله عرض دهان و چشم استفاده شد. یک انسان می‌تواند در هر ساعت حدود ۴۰ تصویر را به این روش پردازش کند و بنابراین یک پایگاه داده از فواصل محاسبه شده ایجاد کند. سپس یک رایانه به‌طور خودکار فواصل مربوط به هر عکس را مقایسه می‌کند، اختلاف فواصل را محاسبه می‌کند و سوابق بسته را به عنوان یک تطابق احتمالی بازمی‌گرداند.

در سال ۱۹۷۰ تاکئو کاناده به‌طور علنی یک سیستم تطبیق چهره را نشان داد که ویژگی‌های آناتومیکی مانند چانه را در آن قرار داشت و نسبت فاصله بین ویژگی‌های صورت را بدون دخالت انسان محاسبه کرد. آزمایش‌های بعدی نشان داد که این سیستم همیشه نمی‌تواند ویژگی‌های صورت را به‌طور قابل اعتماد شناسایی کند. اما علاقه به این موضوع افزایش یافت و در سال ۱۹۷۷ تاکئو کاناده اولین کتاب تفصیلی درباره فناوری تشخیص چهره را منتشر کرد.

در سال ۱۹۹۳، آژانس تحقیقات پیشرفته دفاع (DARPA) و آزمایشگاه تحقیقات ارتش (ARL) برنامه فناوری تشخیص چهره FERET را برای توسعه قابلیت‌های شناسایی خودکار چهره که می‌تواند در یک محیط تولیدکننده زندگی واقعی استفاده شود ایجاد کردند تا به امنیت، اطلاعات کمک کند.

تا دهه ۱۹۹۰ سیستم‌های تشخیص چهره در درجه اول با استفاده از پرتره‌های عکاسی از چهره انسان ساخته می‌شدند. تحقیق در مورد شناسایی چهره برای یافتن قابل اطمینان صورت در تصویری که شامل سایر اشیا است که در اوایل دهه ۱۹۹۰ با تجزیه و تحلیل مولفه اصلی [2] مورد توجه قرار گرفتند. روش PCA در تشخیص چهره با نام Eigenface نیز شناخته می‌شود و توسط متیو تورک و الکس پنتلند ساخته شده‌است. Eigenfaces براساس ویژگیهای جهانی و متعامد در چهره انسان تعیین می‌شود. صورت انسان به عنوان ترکیبی وزنی از تعدادی از Eigenfaces محاسبه می‌شود. روش تشخیص چهره PCA ترک و پنتلند مقدار داده‌هایی را که باید برای تشخیص چهره پردازش شوند، بسیار کاهش می‌دهد.

[1] Real Time
[2] Principal Component Analysis

پنتلند در سال ۱۹۹۴ ویژگی‌های Eigenface را تعریف کرد، از جمله چشمهای اختصاصی، دهانهای اختصاصی و بینیهای خاص، برای پیشبرد استفاده از PCA در تشخیص چهره. در سال ۱۹۹۷، روش PCA Eigenface در تشخیص چهره با استفاده از تجزیه و تحلیل تشخیصی خطی (LDA) برای تولید Fisherfaces بهبود یافت. LDA Fisherfaces به‌طور غالب در شناسایی چهره مبتنی بر ویژگی PCA مورد استفاده قرار گرفت. در حالی که از Eigenfaces برای بازسازی صورت نیز استفاده شد. در این رویکردها هیچ ساختار جهانی صورت محاسبه نمی‌شود که ویژگی‌های صورت یا قسمت‌های آن را به هم پیوند دهد.

رویکردهای کاملا مشخص مبتنی بر ویژگی برای شناسایی چهره در اواخر دهه ۱۹۹۰ توسط سیستم بوخوم، که از فیلتر گابور برای ضبط ویژگی‌های چهره و محاسبه شبکه ای از ساختار صورت برای پیوند دادن ویژگی‌ها، پیشی گرفت. کریستوف فون در مالسبورگ و تیم تحقیقاتی وی در دانشگاه بوخوم در اواسط دهه ۱۹۹۰ میلادی تطبیق نمودار الاستیک را برای استخراج چهره از تصویر با استفاده از تقسیم‌بندی پوست ایجاد کردند. در سال ۱۹۹۷، روش تشخیص چهره که توسط مالسبورگ توسعه یافته بود، عملکرد بهتری نسبت به سایر سیستم‌های تشخیص چهره در بازار داشت. به اصطلاح «سیستم بوخوم» برای شناسایی چهره به صورت ZN-Face در بازار به اپراتورهای فرودگاه‌ها و دیگر مکان‌های شلوغ به صورت تجاری در بازار فروخته شد. این نرم‌افزار به اندازه کافی قوی بود تا بتواند از نمای کمتری از چهره استفاده کند. همچنین از طریق موانعی برای شناسایی سبیل، ریش، مدل موهای تغییر یافته و عینک - حتی عینک آفتابی از طریق موانع مختلف نیز قابل مشاهده است.

تشخیص چهره به صورت واقعی در فیلم‌های ویدئویی در سال ۲۰۰۱ با چارچوب تشخیص شی Viola – Jones برای چهره‌ها امکان‌پذیر شد. پاول ویولا و مایکل جونز روش تشخیص چهره خود را با رویکرد ویژگی Haar-like برای تشخیص اشیا در تصاویر دیجیتال ترکیب کردند تا AdaBoost، اولین ردیاب چهره در زمان واقعی از جلو را راه اندازی کنند. تا سال ۲۰۱۵ الگوریتم Viola-Jones با استفاده از ردیاب‌های کم قدرت کم در دستگاه‌های دستی و سیستم‌های جاسازی شده پیاده‌سازی شده بود؛ بنابراین، الگوریتم Viola-Jones نه تنها کاربرد عملی سیستم‌های تشخیص چهره را گسترش داده بلکه برای پشتیبانی از ویژگی‌های جدید در رابط‌های کاربری و کنفرانس تلفنی نیز مورد استفاده قرار گرفته‌است

نحوه کارکرد فناوری تشخیص چهره چیست؟

شاید شما استعداد ذاتی در شناسایی تصاویر داشته باشید. احتمالا فکر می‌کنید شناسایی چهره‌ی یک عضو خانواده، یک دوست و یا یک فرد آشنا کار بسیار ساده‌ای است. شما با ویژگی‌های چهره‌ی آن‌ها آشنایی دارید مانند چشمان، بینی و دهان آن‌ها و این‌که چگونه این ویژگی‌ها در کنار هم قرار می‌گیرند.

این همان نحوه‌ی کارکرد یک سیستم تشخیص چهره است؛ اما در یک مقیاس بزرگ و الگوریتمی. وقتی شما یک تصویر را می‌بینید، فناوری تشخیصی در اصل فقط داده می‌بیند. این داده می‌تواند ذخیره شود و در دسترس قرار گیرد..

مراحل پایه‌ای فناوری تشخیص چهره چیست؟

مرحله اول :

عکسی از چهره‌ی شما از یک فیلم یا عکس گرفته می‌شود. چهره‌تان ممکن است به تنهایی و یا در گروه نمایان باشد. هم‌چنین ممکن است تصویر شما صاف و یا متمایل باشد.

مرحله دوم :

نرم‌افزار تشخیص چهره هندسه‌ی چهره‌ی شما را می‌خواند. عوامل اصلی فاصله‌ی میان چشمان و هم‌چنین فاصله‌ی میان پیشانی و چانه‌ی شما هستند. این نرم‌افزار نقاط برجسته‌ی تصویرشما که کلید اصلی تمایز چهره‌تان است را شناسایی می‌کند. از ویژگی‌های چره شما به‌عنوان امضای تصویر یاد می‌شود.

مرحله سوم:

امضای تصویری شما، یک فرمول ریاضی، با یک پایگاه داده از چهره‌های شناخته شده مقایسه می‌شود .

مرحله چهارم:

در این مرحله تشخیص صورت می‌گیرد؛ به این معنی که اثرچهره‌ی شما با یکی از داده‌های پایگاه داده منطبق می‌شود.

در واقع مراحل گفته شده بدین صورت عمل می‌کند که بعد از اینکه عکس شما دیتکت شد، حاشیه‌های اضافی از تصویر حذف می‌شود و سپس فاصله بین ویژگی‌های موجود در چهره محاسبه شده طبق فرمولهایی و در ادامه در پایگاه داده این فواصل چک می‌شود و چهره فرد مطابقت داده می‌شود.

روش‌های تشخیص چهره :

در حالی که انسان بدون تلاش زیاد می‌تواند چهره را تشخیص دهد، تشخیص چهره یک مشکل تشخیص الگو در محاسبات است. سیستم‌های تشخیص چهره بر اساس تصویر دو بعدی آن سعی در شناسایی چهره انسان دارند که سه بعدی است و با نور و حالت چهره تغییر شکل می‌دهد. برای انجام این کار محاسباتی، سیستم‌های تشخیص چهره چهار مرحله را انجام می‌دهند. اولین تشخیص چهره برای تقسیم چهره از پس زمینه تصویر استفاده می‌شود. در مرحله دوم، تصویر چهره تقسیم شده با توجه به حالت چهره، اندازه تصویر و خصوصیات عکاسی، مانند نور و مقیاس خاکستری تراز می‌شود. هدف از فرایند تراز، امکان محلی سازی دقیق ویژگی‌های صورت در مرحله سوم، استخراج ویژگی‌های صورت است. ویژگی‌هایی مانند چشم، بینی و دهان در تصویر مشخص شده و اندازه‌گیری می‌شوند تا چهره را نشان دهند. بردار مشخصه چهره در مرحله چهارم با پایگاه داده‌ای از چهره‌ها مطابقت دارد

روش‌های سنتی :

برخی از الگوریتم‌های تشخیص چهره با استخراج نشانه‌ها یا ویژگی‌هایی از تصویر صورت، ویژگی‌های صورت را شناسایی می‌کنند. به عنوان مثال، یک الگوریتم ممکن است موقعیت نسبی، اندازه و یا شکل چشم‌ها، بینی، استخوان‌های گونه و فک را تجزیه و تحلیل کند. سپس از این ویژگی‌ها برای جستجوی تصاویر دیگر با ویژگی‌های منطبق استفاده می‌شود.

الگوریتم‌های دیگر گالری تصاویر چهره را عادی می‌کنند و سپس داده‌های صورت را فشرده می‌کنند، فقط داده‌های موجود در تصویر را که برای تشخیص چهره مفید است ذخیره می‌کنند. سپس یک تصویر کاوشگر با داده‌های چهره مقایسه می‌شود. یکی از اولین سیستم‌های موفق مبتنی بر تکنیک‌های تطبیق الگو است که روی مجموعه ای از ویژگی‌های برجسته صورت اعمال می‌شود، و نوعی نمایش چهره فشرده را ارائه می‌دهد.

الگوریتم‌های تشخیص را می‌توان به دو رویکرد اصلی تقسیم کرد: هندسی، ویژگی‌های متمایز را بررسی می‌کند، یا عکس متریک، که یک روش آماری است که یک تصویر را به مقادیر تقطیر می‌کند و مقادیر را با الگوها مقایسه می‌کند تا واریانس‌ها را از بین ببرد. برخی این الگوریتم‌ها را به دو دسته گسترده طبقه‌بندی می‌کنند: مدل‌های جامع و مبتنی بر ویژگی. اولی تلاش می‌کند تا صورت را به‌طور کامل تشخیص دهد در حالی که ویژگی مبتنی بر ویژگی‌ها به اجزایی تقسیم می‌شود مانند ویژگی‌ها و تجزیه و تحلیل هر یک از آنها و همچنین مکان مکانی آن با توجه به سایر ویژگی‌ها.

روش تشخیص سه‌بعدی :

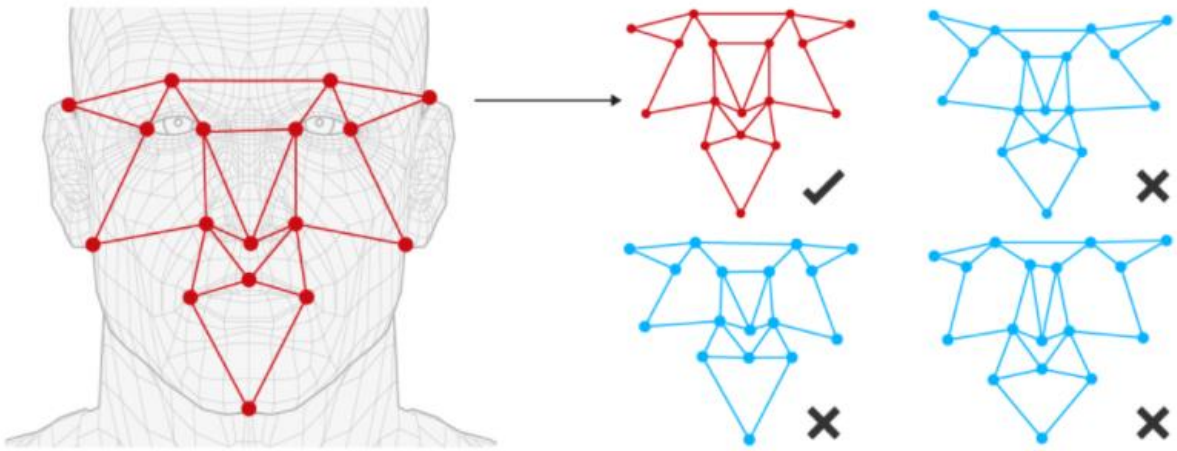
در تکنیک تشخیص چهره سه بعدی از حسگرهای سه بعدی برای گرفتن اطلاعات در مورد شکل صورت استفاده می‌شود. سپس از این اطلاعات برای شناسایی ویژگی‌های متمایز سطح صورت مانند کانتور حفره‌های چشم، بینی و چانه استفاده می‌شود. یکی از مزایای تشخیص چهره سه بعدی این است که مانند سایر تکنیک‌ها تحت تأثیر تغییرات نور قرار نمی‌گیرد. همچنین می‌تواند چهره را از طیف وسیعی از زاویه دید، از جمله نمای نمایه شناسایی کند. نقاط داده سه بعدی از یک چهره دقت تشخیص چهره را بسیار بهبود می‌بخشد. تحقیقات تشخیص چهره سه بعدی با تولید حسگرهای پیچیده‌ای که نور ساختاری را بر روی صورت انجام می‌دهند، امکان‌پذیر است. تکنیک تطبیق سه بعدی به عبارات حساس است، بنابراین محققان از ابزارهای هندسه متریک برای درمان عبارات به عنوان ایزومتری استفاده کردند. یک روش جدید برای گرفتن تصاویر سه بعدی از چهره‌ها از سه دوربین ردیابی استفاده می‌کند که از زوایای مختلف نشان داده می‌شوند. یک دوربین در جلو شخص قرار دارد، دوربین دوم به طرف و دوربین سوم به صورت زاویه دار. همه این دوربین‌ها با هم کار می‌کنند بنابراین می‌تواند صورت شخص را در زمان واقعی ردیابی کند و قادر به تشخیص و تشخیص چهره باشد.

در ادامه با ارائه یک شکل نحوه تشخیص چهره و چهار مرحله ذکر شده را بیان می‌کنیم :

اول از همه باید به سیستم آموزش داده شود که یک چهره چیست تا بتواند آن را از سایر موجودیت های اطرافش تشخیص دهد. این کار با استفاده از یک سری تصاویر که به یک الگوریتم، که معمولا شبکه عصبی عمیق است، داده می شود، پس از آن هربار که به این الگوریتم تصویر جدیدی ارائه می‌شود، با ارزیابی آن تخمین می‌زند که چهره مورد نظر در کجای تصویر قرار گرفته است. در ابتدا ممکن است شبکه دچار اشتباه شود و تخمین‌های اشتباهی تحویل دهد، اما با تکرار این کار طی دفعات متوالی، الگوریتم ارتقا یافته و از طریق آموزش های قبلی می تواند عملیات تشخیص چهره را به درستی انجام دهد. این اولین قدم است.

در قدم بعدی باید به سیستم توانایی تشخیص داده شود. برای انجام این کار روش‌های متعددی وجود دارد، اما معمولا از یک شبکه عصبی دومی استفاده می شود. در این روش، با استفاده از تصاویر به سیستم آموزش داده می شود که چگونه بتواند یک فرد را از دیگری تشخیص دهد. برخی از الگوریتم‌ها نیز از صورت یک فرد به نوعی نقشه برداری نموده و به عنوان مثال برای انجام این کار فاصله بین چشم‌ها تا ابروها، فاصله بین دو چشم، فاصله بین بینی و دهان و به طور کلی تمام فواصل بین اجزای یک صورت را اندازه گیری می‌نماید. برخی الگوریتم های دیگر با استفاده از ویژگی های انتزاعی تر، نقشه یک چهره را ترسیم می کنند. در این روش، شبکه برای هر چهره یک وکتور ایجاد کرده که با استفاده از سلسله‌ای عدد، می تواند یک فرد را در میان افراد دیگر شناسایی کند.

در تصویربرداری زنده، سیستم این قابلیت را دارد که فیلم را به بخش های کوتاهی تقسیم کرده و آن‌ها را در زمان واقعی پردازش کند. در فیلم‌هایی که در نقاط شلوغی مانند ورودی یک استادیوم ورزشی گرفته می‌شود، سیستم در ابتدا چهره هر فرد را در یک فریم از فیلم شناسایی کرده و برای هر کدام یک وکتور جدا ایجاد می‌کند. سپس از این وکتورهای ایجاد شده برای شناسایی فردی که از قبل به سیستم داده شده است استفاده کرده و آن را با هر کدام از وکتورهای بدست آمده تطابق می‌دهد. هر کدام از چهره‌هایی که با ویژگی‌های چهره‌ی داده شده شباهت داشت، جمع آوری شده و بر اساس میزان شباهت رتبه بندی می‌شود.

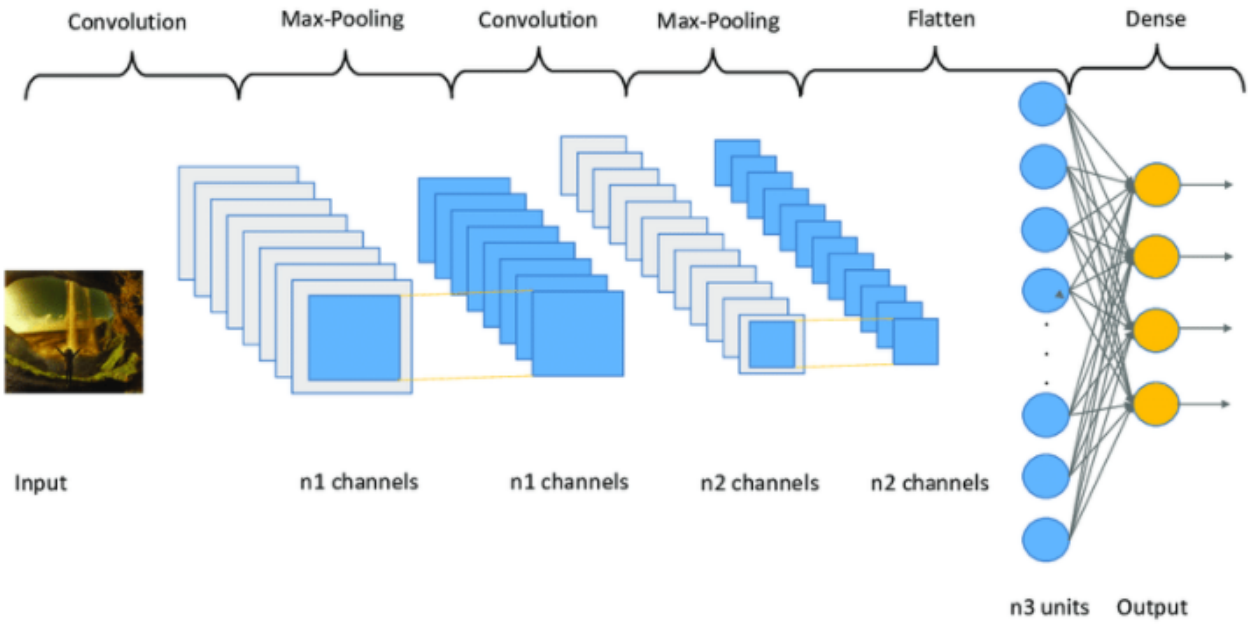


شکل 1 .تشخیص چهره

فناوری تشخیص چهره در ابتدا با استفاده از هندسه یک صورت، یک "اثر صورت" (faceprint) برای هر فرد ایجاد می کند که همانند اثر انگشت منحصر به فرد است.

از این faceprint برای مقایسه با چهره افرادی که از قبل به سیستم داده شده است استفاده می شود و سیستم بر اساس بیشترین شباهت تصاویر را رتبه بندی می‌کند. صحت گزینه‌هایی که رتبه بندی شده‌اند در نهایت با تایید یک اپراتور انسانی مشخص می‌شوند.

شبکه عصبی کانولوشن نوع خاصی از شبکه عصبی با چندین لایه است. داده‌هایی را که دارای آرایش شبکه‌ای هستند پردازش می‌کند و سپس ویژگی‌های مهم را استخراج می‌کند. یک مزیت بزرگ استفاده از CNN این است که نیازی به انجام بسیاری از مراحل پردازش روی تصاویر ندارید



شکل 2 .شبکه عصبی کانولوشن

با وجود بیشتر الگوریتم‌هایی که پردازش تصویر را کنترل می‌کنند، فیلترها معمولاً توسط یک مهندس بر اساس ابتکارات ایجاد می‌شوند. CNN می‌تواند بیاموزد مهم‌ترین ویژگی در فیلترها چیست. این باعث صرفه‌جویی زیادی در وقت و آزمایش و خطا می‌شود زیرا ما به پارامترهای زیادی احتیاج نداریم.

یک تفاوت بزرگ بین CNN و یک شبکه عصبی دیگر این است که CNN ها از کانولوشن برای مدیریت ریاضیات پشت‌صحنه استفاده می‌کنند. حداقل در یک لایه CNN به‌جای ضرب ماتریس از کانولوشن استفاده می‌شود. کانولوشن‌ها در دو تابع محاسبه می‌شوند و یک تابع را برمی‌گرداند.

در اصل CNN با استفاده از فیلترها روی داده‌های ورودی شما کار می‌کند. آنچه آن‌ها را بسیار خاص می‌کند این است که CNN ها می‌توانند فیلترها را در صورت آموزش تنظیم کنند. به‌این‌ترتیب نتایج در زمان واقعی تنظیم می‌شوند، حتی وقتی مجموعه داده‌های عظیمی مانند تصاویر دارید.

ازآنجاکه می‌توان فیلترها را برای آموزش بهتر CNN به‌روز کرد، این امر نیاز به فیلترهای دستی را از بین می‌برد. این به ما امکان انعطاف‌پذیری بیشتری در تعداد فیلترهایی که می‌توانیم برای مجموعه داده‌ها و ارتباط آن فیلترها اعمال کنیم، می‌دهد. با استفاده از این الگوریتم می‌توانیم روی مشکلات پیچیده‌تری مانند تشخیص چهره کار کنیم.

یکی از مواردی که از بسیاری از مشکلات استفاده از CNN جلوگیری می‌کند، کمبود داده است. درحالی‌که می‌توان شبکه‌ها را با نقاط داده نسبتاً کمی آموزش داد اما هرچه اطلاعات بیشتری در دسترس باشد، CNN بهتر تنظیم می‌شود.

شبکه‌های عصبی کانولوشن چگونه کار می‌کنند؟

شبکه‌های عصبی کانولوشنال بر اساس یافته‌های علوم اعصاب است. آن‌ها از لایه‌های نورون مصنوعی به نام گره ساخته شده‌اند. این گره‌ها توابعی هستند که مجموع وزنی ورودی‌ها را محاسبه می‌کنند و یک نقشه فعال‌سازی را برمی‌گردانند. این قسمت تجمع شبکه عصبی است.

هر گره در یک لایه با مقادیر وزنی آن تعریف می‌شود. وقتی به لایه، برخی از داده‌ها را می‌دهید، مانند تصویر، مقادیر پیکسل را می‌گیرد و برخی از ویژگی‌های بصری را انتخاب می‌کند.

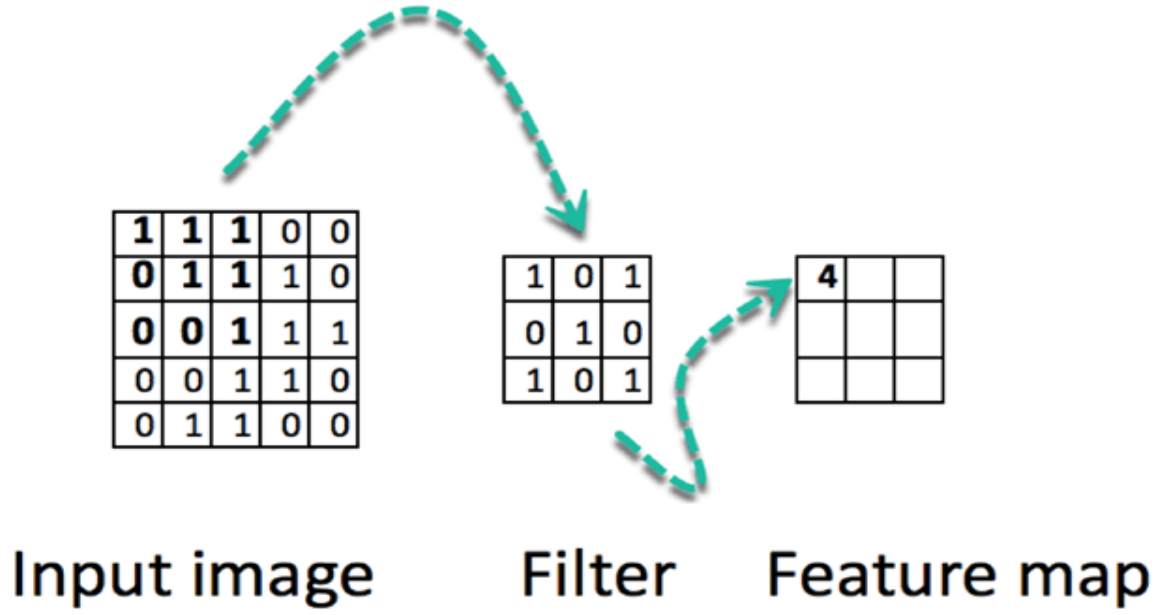
هنگامی‌که با داده‌های CNN کار می‌کنید، هر لایه نقشه فعال‌سازی را برمی‌گرداند. این نقشه‌ها به ویژگی‌های مهم مجموعه داده اشاره دارند. اگر به CNN تصویری داده باشید، به ویژگی‌های مبتنی بر مقادیر پیکسل مانند رنگ‌ها اشاره می‌کند و عملکرد فعال‌سازی را به شما می‌دهد.

معمولاً با تصاویر، CNNدر ابتدا لبه‌های تصویر را پیدا می‌کند. سپس این تعریف جزئی از تصویر به لایه بعدی منتقل می‌کند. سپس آن لایه شروع به شناسایی مواردی مانند گوشه‌ها و گروه‌های رنگی می‌کند. سپس این تعریف تصویر به لایه بعدی منتقل می‌شود و چرخه تا پیش‌بینی ادامه می‌یابد.

هنگامی‌که بیشتر لایه‌ها تعریف می‌شوند، به این حداکثر تجمع می‌گویند. این فقط مرتبط‌ترین ویژگی‌ها را از لایه موجود در نقشه فعال‌سازی برمی‌گرداند. این همان چیزی است که به هر لایه پی‌درپی منتقل می‌شود تا زمانی که لایه نهایی را به دست آورید.

در شکل زیر نحوه کانوال کردن یک فیلتر یا به اصطلاح کرنل را در ماتریس اصلی می‌بینیم. وقتی به عنوان مثال کرنل سه در سه را شروع می‌کنیم در ماتریس اصلی ضرب کردن و به پیش بردن آن این ضرب باعث می‌شود ما از ماتریس ورودی تصویر خودمان یک ماتریس به اندازه کرنل داشته باشیم.

همانطور که در شکل زیر قابل مشاهده است.



شکل 3 .نحوه کانوال کردن تصویر و فیلتر

آخرین لایه CNN لایه طبقه‌بندی است که مقدار پیش‌بینی‌شده را بر اساس نقشه فعال‌سازی تعیین می‌کند. اگر یک نمونه دست خط را به CNN منتقل کنید، لایه طبقه‌بندی به شما می‌گوید که حرف در تصویر چیست. این همان چیزی است که وسایل نقلیه خودکار برای تعیین اینکه یک شی اتومبیل دیگری است، یا یک شخص یا یک مانع دیگر است، استفاده می‌کنند.

آموزش CNN مشابه آموزش بسیاری از الگوریتم‌های یادگیری ماشین است. شما با برخی از داده‌های آموزشی که جدا از داده‌های آزمون شما است شروع خواهید کرد و وزن خود را بر اساس دقت مقادیر پیش‌بینی‌شده تنظیم خواهید کرد.

از گزینه‌های مختلف برای یک شبکه عصبی کانولوشن استفاده می‌شود.

انواع مختلفی از CNN وجود دارد که می‌توانید بسته به مشکل خود از آن‌ها استفاده کنید.

مدل CNN 1D: با این‌ها هسته CNN در یک‌جهت حرکت می‌کند. CNN های D1 معمولاً روی داده‌های سری زمانی استفاده می‌شوند.

مدل CNN 2D: این نوع هسته‌های CNN در دو جهت حرکت می‌کنند. این موارد را با برچسب‌گذاری و پردازش تصویر مشاهده خواهید کرد.

مدل CNN 3D: این نوع CNN دارای هسته‌ای است که در سه جهت حرکت می‌کند. با استفاده از این نوع CNN، محققان از آن‌ها در تصاویر سه‌بعدی مانند سی‌تی‌اسکن و MRI استفاده می‌کنند.

در بیشتر موارد، CNN های دوبعدی را مشاهده خواهید کرد زیرا معمولا با داده‌های تصویر مرتبط هستند. در اینجا برخی از برنامه‌هایی که ممکن است CNN مورد استفاده را مشاهده کنید، آورده شده است.

تشخیص تصاویر با پیش‌پردازش کم

تشخیص دست نوشته‌های مختلف

برنامه‌های دید رایانه‌ای

استفاده در بانکداری برای خواندن رقم چک

استفاده در سرویس‌های پستی برای خواندن کد پستی روی پاکت نامه

عملکرد کانولوشن:

در عملگر کانولوشن چهار مولفه اصلی داریم:

ماتریس یا تصویر ورودی(Input)

فیلتر یا کرنل کانولوشنی(Convolution Filter)

عملگر کانولوشن(*)

ویژگی خروجی کانولوشن(Output)

خیلی ساده بخواهم عملکرد کانولوشن را توضیح دهم، باید بگویم که عملگر کانولوشن (*)، کرنل یا فیلتر کانولوشنی را برمی‌دارد و روی تصویر یا ماتریس ورودی می‌لغزاند. به‌عبارتی دیگر، کرنل یا فیلتر روی تصویر حرکت می‌کند یا تصویر ورودی را اسکن می‌کند.

اعدادی که در ماتریس خروجی ذخیره می‌شوند، تابعی از ورودی و فیلتر هستند. چه زمانی به ازای ضرب بین فیلتر روی یک محل از تصویر، خروجی بزرگ یا کوچک می‌شود؟ ساده است، هروقت فیلتر با یک پنجره از تصویر خیلی شبیه هم باشند (از لحاظ عددی)، خروجی عدد بزرگی می‌شود. اگرهم شبیه هم نباشند، خروجی عدد کوچکی می‌شود. یعنی چه؟ یعنی اینکه، فیلتر به دنبال پیدا کردن نواحی مشابه خود در تصویر است و هرجایی ناحیه مشابهِ خود را پیدا کرد عدد بزرگ تولید می‌شود.

پس کانولوشن منجر به یافتن الگوهای خاص در تصویر باتوجه به فیلتر می‌شود. اعداد موجود در فیلتر بسیار مهم هستند.

ارتباط دادن عمل کانولوشن به عملکرد یک نورون. یک نورون یک جمع وزن‌دار (ضرب بین پارامترها و ورودی و نهایتا جمع) بود. اینجا هم همین است! فیلتر که شامل یک سری اعداد است به ورودی‌ها وزن می‌دهد (ضرب درایه به درایه بین فیلتر و پنجره‌های ماتریس ورودی) و نهایتا اعداد وزن‌دهی‌شده ورودی را باهم جمع می‌کند.

در نورون پارامترها متغیر بودند و از طریق فرآیند آموزش بدست می‌آمدند. اینجا هم اعداد موجود در فیلتر از طریق فرآیند آموزش به‌دست می‌آیند. البته، قبل از پیدایش شبکه عصبی کانولوشن، از کانولوشن در پردازش تصویر و سیگنال بسیار زیاد استفاده می‌شد. اما اعداد فیلترهای کانولوشنی ثابت بودند و توسط یک متخصص این اعداد طراحی می‌شوند .

هسته اصلی شبکه CNN لایه کانولوشنی است که درصد اعظم محاسبات شبکه عصبی کانولوشن را به خود اختصاص داده است. هر لایه کانولوشن در شبکه عصبی کانولوشن شامل مجموعه‌ای فیلتر است و از کانولوشن بین فیلترها و لایه ورودی است که خروجی ساخته می‌شود. به خروجی لایه کانولوشنی، فیچرمپ (Feature Map) گفته می‌شود.

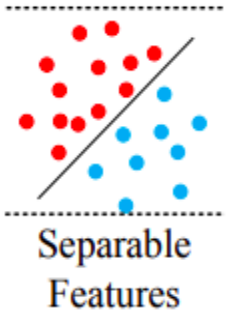
همان‌طور که بخش قبلی گفتم، یک فیلتر می‌تواند شامل یک الگویی خاص باشد و در تصویر به دنبال آن الگو باشد. اتفاقا در فرآیند آموزش شبکه، به دنبال این هستیم که این فیلترها الگوهای معناداری از هر تصویر استخراج کنند. مثلا، فیلتری داریم که شامل الگوی گوش گربه هست و می‌تواند حضور یک گربه در تصویر ورودی را تشخیص دهد. اما فقط به یک الگو بسنده کنیم؟

آیا گربه، فقط شامل یک الگوی خاص است؟ مثلا، دهان، گوش‌ها، چشم‌ها و دم هرکدام الگوی خاصی از گربه نیستند؟ با یک فیلتر می‌شود همه این الگوها را شکار کرد؟ بهتر نیست شواهد بیشتری جمع کنیم و بعد تصمیم بگیریم؟این همان کاری است که در لایه کانولوشنی انجام می‌شود. جستجو در تصویر برای یافتن تنها یک الگو منجر به نتایج خوبی نمی‌شود و باعث می‌شود شبکه از لحاظ کارایی محدود باشد. برای حل این مشکل، نیاز است که لایه کانولوشنی چندین فیلتر داشته باشد. هریک از فیلترها به تنهایی یک الگوی خاص داشته باشند و خروجی لایه کانولوشنی مجموعه‌ای از الگوهای مختلف باشد.

این رویکرد به معنای یادگیری مسافت در فضای کم ابعاد است که با مفهوم شباهت معنایی مطابقت دارد. این به معنی یادگیری فاصله در یک فضای کم بعدی (فضای غیر ورودی) است به طوری که تصاویر مشابه در فضای ورودی منجر به نمایش مشابه (فاصله کم) و تصاویر غیر مشابه منجر به نمایش متفاوت (فاصله زیاد) می‌شود. یادگیری متریک به مشکل راه اندازی باز در یادگیری ماشین می‌پردازد ، یعنی در زمان آزمایش به نمونه‌های جدید تعمیم می‌دهیم. این توسط یک استخراج کننده ویژگی و به دنبال آن شبکه طبقه بندی لایه کاملاً متصل امکان پذیر نیست. دلیل آن به شرح زیر است:

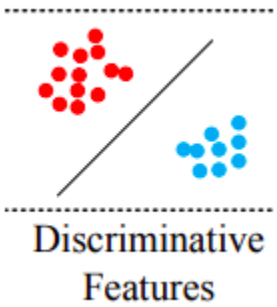
طبقه بندی کننده ویژگی‌های خاص کلاس را یاد می‌گیرد و لزوماً ویژگی‌های عمومی را یاد نمی‌گیرد.

طبقه بندی کننده با از دست دادن استاندارد آنتروپی متقابل فاصله بین کلاس‌ها را به حداکثر می‌رساند به طوری که ویژگی‌های قبل از لایه فولی کانکتت بصورت خطی قابل تفکیک هستند.



شکل 4 .ویژگی‌های استخراج شده توسط شبکه طبقه بندی

و به دنبال به حداقل رساندن فاصله درون طبقه‌ای نباشید که منجر به ویژگی‌های مطلوب تبعیض آمیز می‌شود.



شکل 5 .ویژگی‌ها با low intra-class distance and high inter-class distance

قبل از اینکه به متداول ترین روشهای مورد استفاده در یادگیری متریک بپردازیم ، اجازه دهید کاربردهای آن را بررسی کنیم تا بیان مسئله بیشتر مشخص شود و اینکه چرا یک روش طبقه بندی استاندارد ممکن است مناسب نباشد.

موارد استفاده از یادگیری متریک به شرح زیر است:

Image retrieval

Near duplicate detection

Few/zero shot learning

حال روشهای مورد استفاده در یادگیری متریک را مشاهده کنیم:

Siamese network with contrastive loss(pairs)

Triple networks with triplet loss(triplets)

Classification based methods

هدف از این فرایند در واقع میشه گفت :

برای ایجاد جاسازی‌هایی که در فضای اقلیدسی (برای فرض) برای تصاویر مشابه نزدیک و برای تصاویر غیرمتعارف که از هم فاصله دارند.

Siamese network with Contrastive loss:

به ترتیب زیر عمل می‌کنیم:

مجموعه های مشابه و متفاوت برای هر تصویر در مجموعه داده ایجاد کنید.

دو تصویر (از مجموعه مشابه یا غیر مشابه) به یک شبکه عصبی منتقل کرده و جاسازی یا نمایش ابعاد کم را استخراج کنید.

فاصله اقلیدسی بین هر دو جاسازی شده را محاسبه کنید

ضرر را به حداقل برسانید.

4 مرحله بالا را برای تعداد زیادی از جفت‌ها تکرار کنید تا مدل همگرا شود.

بنابراین در حال حاضر ما به یک تابع ضرر (شامل فاصله اقلیدسی) نیاز داریم که برای جفت‌های مشابه صفر و برای جفت‌های متفاوت یک است.

در ادامه این تابع ضرر را تعریف می‌کنیم.

به همین علت تابع ضرر Contrastive Loss تعریف می‌شود:

فرض کنیم جفت ورودی عکسمان باشد. G_w نگاشتی باشد که یک نمایش low dimensional را ایجاد می‌کند.

w نمایش دهنده پارامترها باشد و Y برچسب حقیقی و D_w نشان دهنده فاصله اقلیدسی باشد.

$$L(W, Y, \vec{X}_1, \vec{X}_2) = (1-Y)\frac{1}{2}(Dw)^2 + (Y)\frac{1}{2}\{\max(0, m - Dw)\}^2$$

$$D_w(\vec{X}_1, \vec{X}_2) = \| G_w(\vec{X}_1) - G_w(\vec{X}_2) \|_2$$

این تابع هزینه مقداری شبیه به cross entropy loss است. اولین ترم در تابع هزینه طوری است که برای جفتهای مشابه Y=0 متریک فاصله به سمت صفر میرود و دومین ترم بدین صورت است که برای جفتهای غیر مشابه Y=1 متریک فاصله حداقل m میشود.

Triplet network, Triplet Loss:

مقاله FaceNet یکپارچه سازی برای تشخیص چهره و خوشه بندی رویکرد مشابهی با contrastive loss دارد - مگر اینکه در هر مرحله با جفت ها برخورد کند ، یک triplet از تصاویر را در نظر می‌گیرد.

سه گانه شامل یک تصویر اصلی ، مثبت (شبیه به اصلی) و منفی (متفاوت با اصلی) است.

مراحل زیر را انجام می‌دهیم:

شکل سه گانه (شامل یک جفت مشابه و غیر مشابه با یک تصویر اصلی مشترک).

یک سه گانه را از طریق همان شبکه عصبی عبور دهید و جاسازی های کم بعدی را استخراج کنید.

فاصله اقلیدسی را محاسبه کرده و ضرر را به حداقل برسانید.

سه مرحله بالا را انجام میدهیم تا triplet همگرا شود.

تابع هزینه triplet به صورت زیر تعریف می‌شود:

فرض کنیم f یک نگاشت باشد برای نمایش low dimensional و x_a تصویر اصلی باشد و x_p نصویر مثبت باشد و همچنین x_n تصویر منفی باشد و a حاشیه. در نتیجه فرمول زیر را خواهیم داشت:

$$\sum_i^N [\| f(x_i^a) - f(x_i^p) \|_2^2 - \| f(x_i^a) - f(x_i^n) \|_2^2 + a]$$

از triplet loss تضمین می‌کند که نمایش یک تصویر به همه تصاویر مشابه آن نزدیک تر از هر تصویر منفی دیگر است.

در نتیجه معماری که در آن از یک شبکه (یعنی اشتراک مجموعه‌ای از پارامترها) برای استخراج نمایش‌های کم بعدی برای همه تصاویر در یک سه گانه استفاده می‌شود ، معماری شبکه سه گانه نامیده می‌شود.

مشکلات روش‌های Contrastive loss و triplet loss عبارتند از:

با افزایش تعداد نمونه‌های آموزشی ، تعداد جفت‌های تصویر و تریپلت به شدت افزایش می‌یابد و آموزش بر روی همه جفت‌ها یا ترپلت‌های ممکن را دشوار می کند.

انتخاب ضعیف جفت‌های آموزشی و تریپلت یعنی نمونه‌های آسان می‌تواند منجر به یادگیری ناکارآمد ویژگی‌های تبعیض آمیز شود.

این مشکلات را می‌توان از طریق راه زیر حل کرد:

با انتخاب دقیق جفت های تصویری و سه تایی آموزشی - آفلاین یا آنلاین و با استفاده از اندازه دسته ای بزرگتر.

Classification Based: Center Loss

مقاله یک روش یادگیری ویژگی تبعیض آمیز برای تشخیص چهره عمیق با استفاده از طبقه بندی شبکه عصبی با معرفی یک تابع ضرر جدید به نام center loss، علاوه بر cross entropy loss، یک سیگنال نظارت مشترک با ادغام این دو تابع ، به هدف مطلوب می‌رسد.

این مقاله مشکل ذکر شده در ابتدای مقاله را حل می کند. مشکل این بود که طبقه بندی کننده‌ها تنها فاصله بین کلاس‌ها را افزایش می‌دادند و فاصله درونی کلاس‌ها را کم نمی‌کردند که منجر به ویژگی‌های خطی قابل تفکیک می شود اما ویژگی های تبعیض آمیز نیست.

Center loss، فاصله بین هر مرکز کلاس و نمایش نمونه‌های کلاس را به حداقل می رساند این امر اطمینان می‌دهد که نمایش نمونه‌ها در یک کلاس علاوه بر حفظ فاصله بین کلاس مشابه می ماند.

فرض کنیم x سمپل ورودی شبکه باشد و c مرکز کلاس متناظر باشد. درنتیجه خواهیم داشت:

$$L_c = \frac{1}{2}\sum_{i=1}^{m} \| x_i - c_{y_i} \|_2^2$$

فاصله بین مرکز کلاس (جاسازی ها) و نمونه های جاسازی شده برای هر تکرار محاسبه شده و وزن ها به روز می شوند.

برای تعریف cross entropy به صورت زیر پیش می‌رویم:

آنتروپی متقاطع بین دو توزیع احتمال p ,q روی یک مجموعه داده شده به صورت زیر تعریف می‌شود:

$$H(p,q) = E_p[-\log q] = H(p) + D_{KL}(p \,\|\, q)$$

جایی که H(p) آنتروپی p و $D_{KL}(p \,\|\, q)$ دیورژانس کولبک – لیبر از p به q است.

برای p و q گسسته داریم:

$$H(p,q) = -\sum_x p(x) \log q(x)$$

که مشابه توزیعهای پیوسته است باید فرض کنیم که p و q با توجه به اندازهگیری مرجع r کاملا پیوسته هستند.

اگر P وQ توابع چگالی احتمال p,q باتوجه به r باشند، بنابراین:

$$\int_x P(x) \log Q(x) dr(x) = E_p[-\log Q]$$

باید توجه داشت که نوتیشن H(p,q) برای مفهوم دیگری برای p,q به نام آنتروپی توام نیز مورد استفاده قرار می‌گیرد.

در تئوری اطلاعات قضیه کرافت-مک میلن هر کدام از روشهای کد قابل قبول را برای کدگذاری یک پیام شناسایی x_i از مجموعهای از احتمالات X ایجاد میکند.این امر میتواند به صورت یک توزیع احتمال ضمنی $q(x_i) = 2^{-l_i}$ نشان داده میشود، جایی که l_i، طول کد x_i در حالت بیتی است. بنابراین آنتروپی متقاطع میتواند به عنوان طول پیام مورد انتظار در هر پایگاه داده تفسیر شود زمانی که توزیع نادرست Q در حالی فرض میشود که دادهها توزیع P را دنبال میکنند به همین دلیل است که توزیع احتمال مورد انتظار P است.

$$H(p,q) = E_p[l_i] = E_p[\log \frac{1}{q(x_i)}]$$

$$H(p,q) = \sum_x p(x_i) \log \frac{1}{q(x_i)}$$

$$H(p,q) = -\sum_x p(x) \log q(x)$$

آنتروپی متقاطع میتواند برای تعریف تابع زیان در یادگیری و بهینه سازی ماشین استفاده شود.احتمال درست p_i یک برچسب واقعی است و توزیع داده شده q_i ارزش پیش بینی شده از مدل فعلی است.به طور خاص اگر رگرسیون لجستیک را در نظر بگیریم که در فرم اصلی آن با طبقه بندی یک مجموعه داده ای از داده ها به دو دسته ممکن میپردازد که با صفر و یک برچسب گذاری شده است. بنابراین مدل رگرسیون لجستیک یک خروجی $y \in \{0,1\}$ را با توجه به ورودی x پیش بینی می کند.احتمال با تابع لجستیک مدل میشود.

$$g(z) = \frac{1}{1+e^{-z}}$$

یعنی احتمال یافتن خروجی y=1 به صورت زیر است:

$$q_{y=1} = \hat{y} \equiv g(w.x) = \frac{1}{1+e^{-w.x}}$$

جایی که بردار وزن w از طریق برخی الگوریتم های مناسب مثل گرادیان کاهشی بهینه سازی شده است. به طور مشابه احتمال یافتن خروجی y=0 به صورت زیر داده شده است

$$q_{y=0} = 1-\hat{y}$$

احتمال واقعی را میتوان به صورت مشابه $p_{y=1} = y, p_{y=0} = 1-y$ بیان کرد با قرار دادن نماد های $p \in \{y, 1-y\}, q \in \{\hat{y}, 1-\hat{y}\}$ ما میتوانیم آنتروپی متقاطع را برای اندازه گیری عدم هماهنگی بین p,q استفاده کنیم:

$$H(p,q) = -\sum_i p_i \log q_i = -y \log \hat{y} - (1-y)\log(1-\hat{y})$$

تابع هزینه معمولی که از آن در رگرسیون لجستیک استفاده میشود با در نظر گرفتن میانگین تمام آنتروپی های متقابل در نمونه محاسبه شده است به عنوان مثال فرض کنید که ما N نمونه داریم برای هر نمونه نشان داده شده n=1,.....,N تابع زیان به صورت زیر است:

$$J(w) = \frac{1}{N}\sum_{n=1}^{N} H(p_n, q_n) = -\frac{1}{N}\sum_{n=1}^{N}[y_n \log \hat{y}_n + (1-y_n)\log(1-\hat{y}_n)]$$

در جایی که $\hat{y}_n \equiv g(w.x_n) = \frac{1}{1+e^{-w.x_n}}$ که با g(z) مثل قبل یک تابع لجستیک است.

تابع بیشینه هموار Softmax Function که به عنوان softargmax یا تابع نمایی نرمال شده normalized exponential function، شناخته می‌شود. از طرفی این تابع را تعمیم‌یافته تابع لجستیک در حالت چند بعدی محسوب می‌کنند. همچنین از آن در رگرسیون لجستیک چند جمله‌ایMultinomial Logistic Regression استفاده می‌شود و غالبا به عنوان آخرین تابع فعال‌سازی برای یک شبکه عصبی برای نرمال سازی خروجی شبکه و تبدیل آن به توزیع احتمال بهره می‌برند. نرمال سازی در این حالت نسبت به کلاس‌های خروجی پیش بینی شده، صورت می‌گیرد.

تابع بیشینه هموار به عنوان یک ورودی یک بردار z از k عدد حقیقی را می‌گیرد و آن را به یک توزیع احتمال تبدیل می‌کند که متشکل از k احتمال متناسب با نمای اعداد ورودی هستند. این امر به این معنی است که قبل از استفاده از تابع softmax بعضی از اجرای بردار z ممکن است منفی یا بیشتر از یک باشند. اما بعد از استفاده از تابع بیشینه هموار هر مولفه در بازه 0تا1 قرار می‌گیرد. بطوری که مجموع آنها برابر یک باشد. بنابراین می‌توان آنها را به عنوان مقدار احتمال تفسیر کرد. علاوه بر این ورودی‌های با مقادر بزرگتر دارای احتمال بیشتری هستند.

تابع استاندارد بیشینه هموار که گاهی به آن تابع بیشینه هموار واحد (Unit softmax function) نیز گفته می‌شود به صورت زیر تعریف میشود:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_i}} \quad for\ i = 1,....,k\ and\ z = (z_1,...,z_k) \in R^k$$

برای هر عنصر z_i از بردار ورودی z تابع نمایی استاندارد اعمال شده است و تقسیم کردن هر مقدار بر مجموع تمامی آنها باعث نرمال شدن شده و اطمینان می‌دهد که مجموع اجزای بردار خروجی یک خواهد بود.

$$\sigma(z) = 1$$

البته به جای استفاده از عدد نپر به عنوان پایه تابع نمایی میتوان از یک پایه متفاوت مثل $\beta > 0$ استفاده کرد. انتخاب مقدار بزرگ برای b باعث می‌شود که توزیع احتمالی چگالی مقادیر خروجی حول مقادیر بزرگتر بیشتر می‌شود به این ترتیب برای مقدار حقیقی β به صورت $b = e^{\beta}, b = e^{-\beta}$خواهیم داشت:

$$\sigma(z_i) = \frac{e^{\beta z_i}}{\sum_{j=1}^{k} e^{\beta z_i}}$$

$$\sigma(z_i) = \frac{e^{-\beta z_i}}{\sum_{j=1}^{k} e^{-\beta z_i}}$$

ار تابع softmax در مکانیک آماری در توزیع بولتزمن که در مقاله بنیادی او در سال 1868 منتشر شد استفاده شده است. همچنین در کتاب معروف و تاثیر گذار گیبس(Gibbs) در سال 1908 نیز این تابع به کار رفته است و برای حل مسائل مرتبط با سیستم‌های مبتنی بر مکانیک آماری و ترمودینامیک آماری امری ضروری تلقی شده‌اند.

از این تابع در نظریه تصمیم نیز استافده شده است. کارهایی که Duncan Luce در سال 1959 انجام داد باعث استفاده از شرط استقلال در نظریه انتخاب منطقی شد. او در این زمینه از تابع بیشینه هموار برای ترجیح نسبی استفاده کرد.

در سال John Bridle 1990 برای الگوریتم‌های یادگیری ماشین از این تابع بهره گرفت و نشان داد که برای شبکه‌های عصبی غیرخطی یا پرسپترونهای چند لایه استفاده از این تابع می‌تواند نتیجه بسیار بهتری ارائه کند و این موضوع را به صورت زیر بیان کرد.

ما در حوزه شبکه‌های غیرخطی پیشخور مثل پرسپترون چند لایه یا همان MLP با چندین خروجی مواجه هستیم و می‌خواهیم خروجی‌های شبکه را به عنوان احتمال گزینه‌ها به عنوان مثال کلاس‌های الگو به شکل تابعی از ورودی‌ها در نظر بگیریم. همچنین به دنبال پیدا کردن تابعی غیر خطی به عنوان خروجی مناسب هستیم تا معیارهای مناسب برای انطباق با پارامترهای شبکه مثلا ضرایب وزنی برای گره‌ها را دارا باشد. این کار بوسیله دو تغییر اجرا می‌شود.

امتیازدهی احتمالیprobability scoring که جایگزینی برای کمینه سازی مربع خطا squared error minimization است و دیگری نرمال سازی نمایی بیشینه هموارsoftmax روی ورودی‌های چند متغیر از نوع لجستیک غیر خطی.

خروجیهای حاصل برای هر ورودی مثبت بوده و مجموع آنها باید برابر یک باشد. با توجه به مجموعه مقادیر مثل V_j که بدون هیچ محدودیتی در نظر گرفته می‌شوند. میتوان این دو تغییر را با استفاده از یک تابع نرمال شده به صورت زیر ایجاد کرد :

$$Q_j(x) = \frac{e^{V_j(x)}}{\sum_k e^{V_k(x)}}$$

این تغیرات را می‌تواند یک تعمیم از لجستیک با ورودی‌های چند متغیره در نظر گرفت که روی لایه خروجی اعمال می‌شود. به این ترتیب رتبه مقادیر ورودی حفظ شده و یک تعمیم جدید برای پیدا کردن مقدار حداکثر ارائه می‌دهد به همین دلیل از واژه softmax یا همان بیشینه هموار برای آن استفاده کردیم.

از تابع softmax اغلب در لایه نهایی تکنیک طبقه بندی مبتنی بر شبکه‌های عصبی استفاده می‌شود. چنین شبکه‌هایی معمولا تحت توابع cross Entropy آموزش می‌یابند و یک نوع رابطه غیرخطی از رگرسیون لجستیک چند جمله‌ای ارائه می‌کنند.

از آنجا که این تابع یک بردار را و یک اندیس خاص مثل i را به یک مقدار از اعداد حقیقی نگاشت می‌کند باید هنگام مشتق گیری به اندیس i نیز توجه شود در نتیجه خواهیم داشت:

$$\frac{\partial}{\partial q_k}\sigma(q,i) = \sigma(q,i)(\delta_{ik} - \sigma(q,k))$$

این عبارت نسبت به اندیس های i و k متقارن است. در نتیجه می‌توان به شکل ساده‌تری آن را نمایش داد.

$$\frac{\partial}{\partial q_k}\sigma(q,i) = \sigma(q,i)(\delta_{ik} - \sigma(q,i))$$

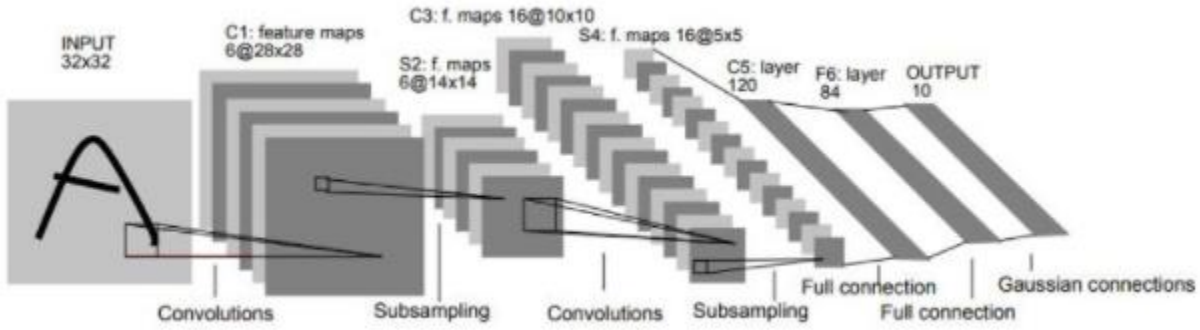
اما چیزی که به نظرم نیاز هست بگم این است که softmax در لایه آخر استفاده می‌شود که خروجی لایه قبل را می‌گیرد و تبدیلش می‌کند به یک توزیع احتمال روی کلاسها که این فرایند در تست و آموزش هیچ فرقی ندارد. اما در آموزش یا همان train وقتی می‌خواهیم خطای شبکه را بدست بیاریم باید loss تعریف بشود که این loss بسته به مساله متفاوته. برای softmax معمولا cross Entropy استفاده می‌کنیم که بهش softmaxloss می‌گویند.

LeNet یک ساختار شبکه عصبی کانولوشن است که توسط Yann LeCun و همکاران در سال ۱۹۸۹ پیشنهاد شده‌است. یک شبکه عصبی کانولوشن ساده است. شبکه‌های عصبی کانولوشن نوعی شبکه عصبی پیش خور هستند که در ان نورون‌های مصنوعی می‌توانند به بخشی از سلول‌های اطراف در محدوده پوشش پاسخ دهند و در پردازش تصویر در مقیاس بزرگ عملکرد خوبی داشته باشند.

LeNet یکی از اولین شبکه‌های عصبی کانولوشن بود و توسعه یادگیری عمیق را ارتقا داد. از سال ۱۹۸۸ پس از سالها تحقیق و تکرارهای موفقیت‌آمیز بسیاری، کار پیشگامان LeNet5 نامگذاری شده‌است. در سال ۱۹۸۹، یان لکون و همکاران در آزمایشگاه‌های بل ابتدا الگوریتم انتشار مجدد را در کاربردهای عملی اعمال کرد و معتقد بود که با یادگیری محدودیت‌های دامنه کار توانایی یادگیری تعمیم شبکه تا حد زیادی افزایش می‌یابد. او یک شبکه عصبی کانولوشن آموزش دیده را ترکیب کرد که توسط الگوریتم‌های انتشار مجدد برای خواندن شماره‌های دست‌نویس آموزش داده شده و با موفقیت آن را در شناسایی شماره‌های کد پستی دست‌نویس ارائه شده توسط سرویس پستی ایالات متحده اعمال کرد. این نمونه اولیه چیزی بود که بعدا LeNet نام گرفت. در همان سال، لکون در مقاله دیگری یک مسئله کوچک شناسایی رقمی دست‌نویس را توصیف کرد و نشان داد که حتی اگر این مسئله به صورت خطی قابل تفکیک باشد، شبکه‌های تک لایه قابلیت‌های ضعیف تعمیم را به نمایش می‌گذارند. هنگام استفاده از آشکارسازهای ویژگی تغییرناپذیر در یک شبکه چند لایه و محدود، مدل می‌تواند عملکرد بسیار خوبی داشته باشد. وی معتقد بود که این نتایج ثابت می‌کند که به حداقل رساندن تعداد پارامترهای آزاد در شبکه عصبی می‌تواند توانایی تعمیم شبکه عصبی را افزایش دهد. در سال ۱۹۹۰، مقاله آنها کاربرد شبکه‌های انتشار مجدد در شناسایی رقمی دست‌نویس را دوباره توصیف کرد. آنها فقط پیش پردازش حداقل داده‌ها را انجام دادند و مدل با دقت برای این کار طراحی شده و بسیار محدود بود. داده‌های ورودی شامل تصاویر، هر یک حاوی یک عدد بود، و نتایج آزمون داده‌های دیجیتال کد پستی ارائه شده توسط سرویس پستی ایالات متحده نشان میداد که مدل فقط دارای نرخ خطا ۱درصد و میزان رد در حدود ۹درصد بود. تحقیقات آنها برای هشت سال آینده ادامه داشت و در سال ۱۹۹۸، یان لکون، لئون بوتو، یوشوا بنگیو و پاتریک هافنر روش‌های مختلف شناسایی شخصیت دست‌نویس را در کاغذ بررسی کردند و از ارقام دست‌نویس استاندارد برای شناسایی وظایف معیار استفاده کردند. این مدل‌ها مقایسه شده و نتایج آنها نشان داد که این شبکه از همه مدل‌های دیگر بهتر عمل کرده‌است. آنها همچنین نمونه‌هایی از کاربردهای عملی شبکه‌های عصبی، مانند دو سیستم برای شناسایی شخصیت‌های دست‌نویس به‌صورت آنلاین و مدل‌هایی که می‌توانند میلیون‌ها چک در روز را بخوانند، ارائه کردند. این تحقیق موفقیت زیادی کسب کرده و علاقه محققان را به مطالعه شبکه‌های عصبی برانگیخته است. با این وجود که امروزه معماری شبکه‌های عصبی با بهترین عملکرد باز هم همانند شبکه LeNet نیست، این شبکه نقطه شروع تعداد زیادی از معماری‌های شبکه عصبی بود و همچنین باعث ایجاد انگیزه در این زمینه شد.

در ادامه سعی می‌کنیم این شبکه را بررسی کنیم.

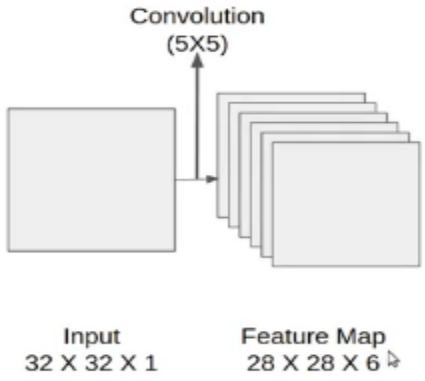
شکل زیر تصویر اصلی از شبکه می‌باشد که در سال 1998 منتشر شده است.



شکل 6 .تصویر اصلی شبکه LeNet

این شبکه شامل دو مجموعه کانولوشنی و لایه میانگین و پولینگ است که با یک لایه کانولوشنی که فلت شده است شروع می‌شود و سپس دولایه به هم وصل شده و در نهایت از یک طبقه بند softmax استفاده می‌کنیم.

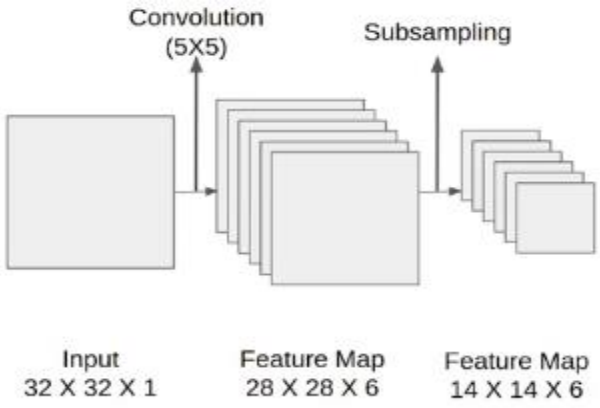
به فرض مثال یک تصویر 32 در 32 خاکستری در واقع یک کاناله از یک لایه کانولوشنی رد می‌شود که به عنوان مثال 6 فیلتر یا به اصطلاح feature map دارد که سایز هر فیلتر 5 در 5 می باشد که در شکل زیر قابل مشاهده است.



شکل 7 .لایه اول شبکه LeNet

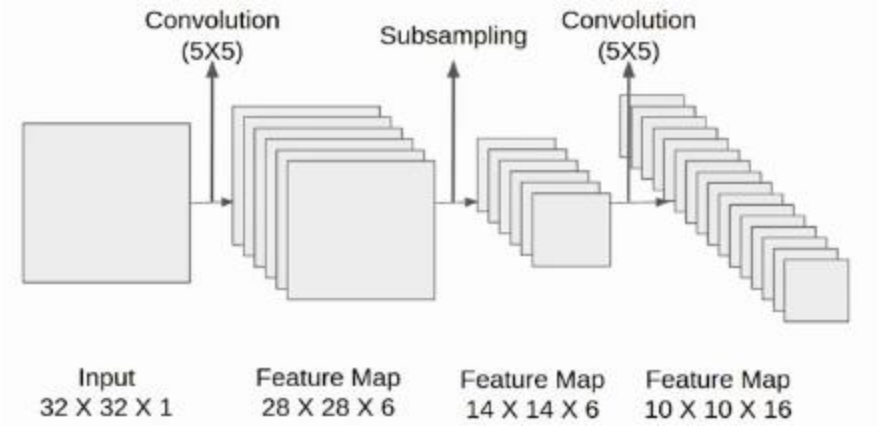
$$Output\ shape = ((32-5+1)\times 32-5+1)\times 6) = (28\times 28\times 6)$$

سپس شبکه average pooling یا sub-sampling را با یک فیلتر 2 در 2 و stride با عدد 2 را به خروجی‌های قبلی اعمال می‌کند. سپس نتیجه زیر را خواهیم داشت:



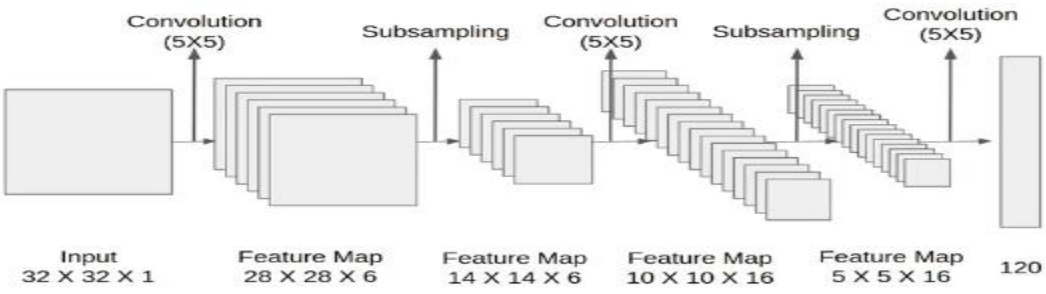
شکل 8 .لایه دوم شبکه LeNet

بعد از اعمال عملیات pooling ما سپس average pooling را به سیستم اعمال می‌کنیم که منجر به نتیجه زیر می‌شود:



شکل 9 .اعمال average pooling بر شبکه LeNet

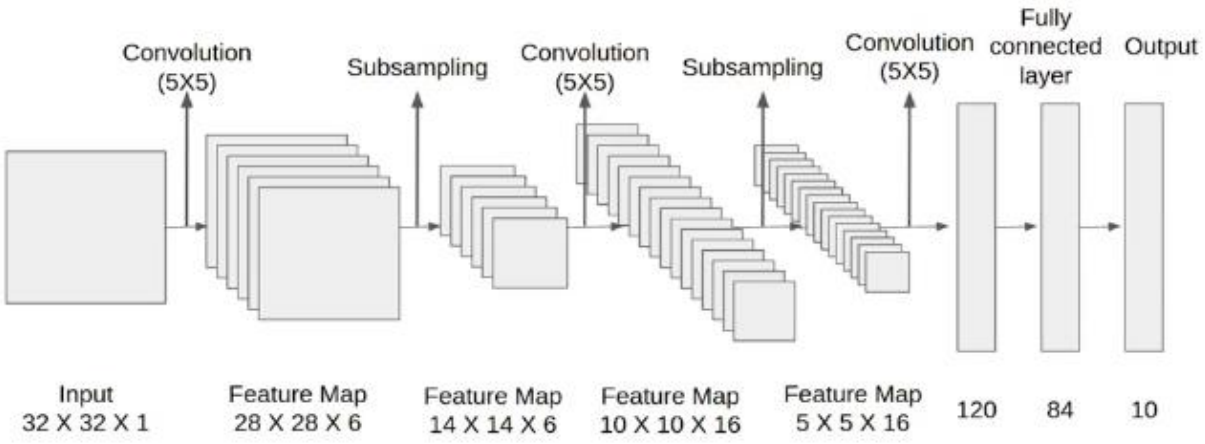
سپس ما یک لایه کانولوشنی با تعداد 16 فیلتر 5 در 5 خواهیم داشت. و بعد از اعمال این فرایند ما دوباره average pooling را به شبکه اعمال کرده تا اندازه ویژگی‌ها را کاهش دهیم



شکل 10 .اعمال لایه آخر کانولوشنی و فلت کردن داده در LeNet

سپس ما لایه آخر کانولوشن را خواهیم داشت با اندازه 5 در 5. با تعداد 120 فیلتر. و سپس داده‌ها را همان‌طور که در شکل بالا می‌بینید به یک وکتور با استفاده از Flatten تبدیل می‌کنیم.

در نهایت شبکه به صورت زیر می‌شود :



شکل 11. شبکه نهایی LeNet

Layer	# filters / neurons	Filter size	Stride	Size of feature map	Activation function
Input	-	-	-	32 X 32 X 1	
Conv 1	6	5 * 5	1	28 X 28 X 6	tanh
Avg. pooling 1		2 * 2	2	14 X 14 X 6	
Conv 2	16	5 * 5	1	10 X 10 X 16	tanh
Avg. pooling 2		2 * 2	2	5 X 5 X 16	
Conv 3	120	5 * 5	1	120	tanh
Fully Connected 1	-	-	-	84	tanh
Fully Connected 2	-	-	-	10	Softmax

شکل 12. جزییات شبکه LeNet

برای شبیه سازی این مقاله من از دیتاست MNIST استفاده کردم که در ابتدا به توضیح این دیتاست پرداخته و سپس نحوه شبیه سازی را توضیح خواهم داد.

MNIST یک پایگاه داده است که مخفف کلمه‌ی Modified National Institute of Standards and Technology می‌باشد. پایگاه داده MNIST شامل ارقام دست نویس (0 تا 9) است و می‌تواند یک پایه برای آزمایش سیستم‌های پردازش تصویر فراهم کند. MNIST به دو مجموعه داده تقسیم می شود: مجموعه آموزش دارای 60000 نمونه اعداد دست نوشته شده و مجموعه آزمون 10000 است. MNIST زیرمجموعه یک مجموعه داده بزرگتر است که در انستیتوی ملی استاندارد و فناوری موجود است. تمام تصاویر آن یک اندازه هستند و درون آنها ، ارقام در مرکز قرار می گیرند و اندازه نرمال می شوند.

شبیه سازی در در گوگل کولب انجام شده است به علت سرعت بهتر اجرا و همچنین به علت اینکه در کلاس تدریس یاری از آن استفاده می‌شد.

در ابتدا ماژول‌های مورد نیاز خود را وارد کرده و به اصلاح آنها را import کردم. که نحوه کد نویسی آن را در شکل زیر میبینیم.

Import Modules

```
[ ]  import numpy as np
     import tensorflow as tf
     from tensorflow.keras.datasets import mnist
     from tensorflow.keras import layers
     from tensorflow.keras.models import Model
     from tensorflow.keras.utils import to_categorical     #One-Hot Encoding
```

از to_categorical برای تبدیل اعداد که در واقع عکس های موجود در دیتاست منظورم هست به وکتوری از اعداد 0 و 1 استفاده کردم.

سپس با استفاده از دستور mnist.load_data() دیتاست خود را به برنامه اضافه کردم و آنها را در ماتریس‌های train و test ذخیره سازی کرده و در ادامه آنها را برای این که بین 0 تا 1 نرمالیزه سازی کنم بر 255 تقسیم کرده که این کار باعث می‌شود شبکه بهتر train شود و همچنین از پیچیدگی محاسباتی هم جلوگیری می‌کنیم.

در شکل زیر نحوه کد نویسی مورد نیاز برای امور گفته شده آورده شده است.

Read Dataset and Normalization

```
[ ]  #Read Dataset
     (X_train, y_train), (X_test, y_test) = mnist.load_data()

     #Normalization
     X_train = X_train.astype("float32")/255.0         #shape = (60000, 28, 28)
     X_test = X_test.astype("float32")/255.0           #shape = (10000, 28, 28)

     #Batch size
     X_train = np.expand_dims(X_train, axis = -1)     #shape = (60000, 28, 28, 1)
     X_test = np.expand_dims(X_test, axis = -1)       #shape = (10000, 28, 28, 1)


     #One-Hot Encoding

     # Befor t0_categorical: y_train equals to [5 0 4 ..... 5 6 8]
     #After to_categorical: y_train = [1 0 0 0 ... 0],.....

     y_train_one_hot = to_categorical(y_train)
     y_test_one_hot = to_categorical(y_test)
```

سپس به معماری شبکه CNN می‌پردازیم.در ابتدا به علت اینکه تصاویر موجود در دیتاست (28x28x1) هستند که این اعداد به ترتیب بیانگر طول و عرض و عمق تصویر هستند که به علت اینکه تصاویر خاکستری یا به اصطلاح gray هستند بر خلاف تصاویر RGB که سه کاناله هستند عدد سوم را یک می‌گذاریم و این تنسور ورودی‌ما را تشکیل می‌دهد. سپس با استفاده از مقاله تعداد لایه‌ها و فیلترهای مورد نیاز برای پیاده سازی معماری CNN را تشکیل می‌دهیم.

	Stage 1		Stage 2		Stage 3		Stage 4
Layer	Conv	Pool	Conv	Pool	Conv	Pool	FC
LeNets	$(5,20)_{/1,0}$	$2_{/2,0}$	$(5,50)_{/1,0}$	$2_{/2,0}$			500
LeNets++	$(5,32)_{/1,2} \times 2$	$2_{/2,0}$	$(5,64)_{/1,2} \times 2$	$2_{/2,0}$	$(5,128)_{/1,2} \times 2$	$2_{/2,0}$	**2**

برای این پروژه من از معماری LeNets++ استفاده کردم.

سپس داده‌ها را با استفاده از BatchNormalization نرمالیزه می‌کنیم. این تابع داده‌های خروجی اش را با توجه به میانگین و و انحراف معیار Batch ورودی به آن نرمالیزه می‌کند و سپس با استفاده از تابع PReLU که تابع فعال ساز ما هست داده‌های خروجی برای این لایه را بدست می‌آوریم و در ادامه بقیه لایه‌ها همین کار را انجام می‌دهند با تفاوت در کرنل آنها.

در نهایت داده را برای ورودی به دو نرون خود مسطح یا به اصطلاح فلت می‌کنم و یک لایه با دو نرون قرار می‌دهیم و دلیل استفاده از دو نرون این است که بتوانم در این ابعاد آنها را نمایش دهم و توزیع احتمال ویژگی‌ها را ببنیم.

در ادامه توضیحاتی که داده شد را میتوان نحوه کد نویسی آن را در شکل زیر دید.

Main part of network - CNN

```
[ ]  #Start defining the input tensor
     #we define input_1 beacause keras ducument say:"When using Conv2D as the first layer in a model, provide the keyword argument input_shape".
     input_1 = layers.Input(shape=(28, 28, 1))

     #Define layers

     x = layers.Conv2D(32, (5, 5))(input_1)    #number of filter = 32 , dimension of filter = (5, 5)
     x = layers.BatchNormalization()(x)
     x = layers.PReLU()(x)

     x = layers.Conv2D(32, (5, 5))(x)          #number of filter = 32 , dimension of filter = (5, 5)
     x = layers.BatchNormalization()(x)
     x = layers.PReLU()(x)

     x = layers.Conv2D(64, (5, 5))(x)          #number of filter = 64 , dimension of filter = (5, 5)
     x = layers.BatchNormalization()(x)
     x = layers.PReLU()(x)

     x = layers.Conv2D(64, (5, 5))(x)          #number of filter = 64 , dimension of filter = (5, 5)
     x = layers.BatchNormalization()(x)
     x = layers.PReLU()(x)

     x = layers.Conv2D(128, (5, 5))(x)         #number of filter = 128 , dimension of filter = (5, 5)
     x = layers.BatchNormalization()(x)
     x = layers.PReLU()(x)

     x = layers.Conv2D(128, (5, 5))(x)        #number of filter = 128 , dimension of filter = (5, 5)
     x = layers.BatchNormalization()(x)
     x = layers.PReLU()(x)

     #Make data to Flatten
     x = layers.Flatten()(x)

     #We choose 2 for visualizing in x-y (2D plane)
     x = layers.Dense(2)(x)

     out_1 = layers.PReLU(name= "out_1")(x)    #Output of 2 neurons we define
```

سپس با توجه به فرمولی که در مقاله برای center loss تعریف شده بود که آن فرمول به صورت زیر تعریف می‌شود:

$$L_C = \frac{1}{2}\sum_{i=1}^{m} \| x_i - c_{yi} \|_2^2$$

این فرمول میزان اختلاف هر ویژگی از کلاس را با مرکز متناظر با آن کلاس بدست آورده و آن را بتوان 2 می‌رساند.

این فرایند گفته شده را با تعریف تابع custom_layer انجام می‌دهیم و میزان این اختلاف را به عنوان خروجی تابع برمی‌گردانیم. این توضیحات به صورت زیر پیاده سازی می‌شود :

Center Loss network

```
#Lambda_c is a constant for center loss function
lambda_c = 1
input_2 = layers.Input(shape=(1,))          # This is input number[0, 1, ....,9]

#Turns positive integers (indexes) into dense vectors of fixed size with Embedding

centers = layers.Embedding(10, 2)(input_2)    #Embeddding convert number to vector

# 10 is type of input for this layer[0, 1, 2, ... ,9] : input_dim
# 2 is the number of output for compare with 2 output in main network : output_dim

#Define center-loss function
def custom_layer(x):
  out_1 = x[0]
  centers = x[1]

  return tf.sqrt(tf.reduce_sum(tf.square(out_1 - centers[:, 0]), axis = 1, keepdims = True))
print(out_1.shape)
print(centers.shape)
intra_loss = layers.Lambda(custom_layer)([out_1, centers])
```

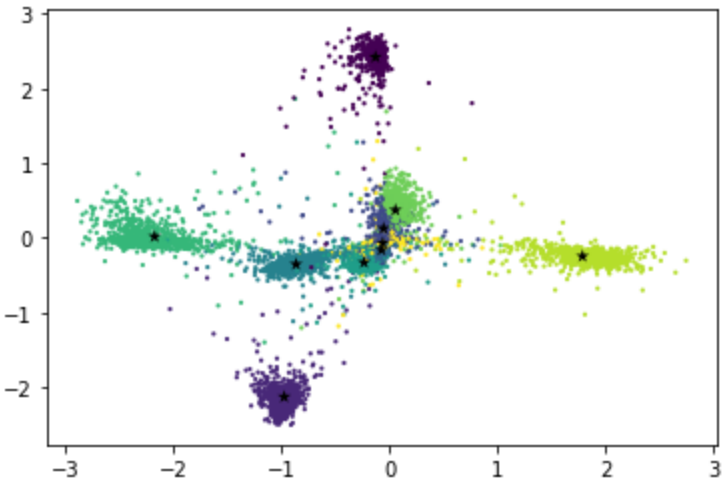
سپس شبکه را به صورت زیر train میکنیم فقط توجه کنیم چون ما دو تابع loss داریم به همین علت باید در قسمت compile برای شبکه که میخواهیم loss تعریف کنیم دوتا تابع به کار ببریم که یکی categorical_crossentropy است که برای تابع softmax استفاده میشود و دیگری custom_loss که این تابع را خودمان تعریف کرده ایم که همان خروجی تابع custum_layer را برمیگرداند.

نحوه train کردن کل شبکه به صورت زیر می‌باشد:

Complete Model and Compile

```
[24] model_center_loss = Model([input_1, input_2], [out_2, intra_loss])

    #Compile Model
    def custom_loss(y_true, y_pred):
      # y_pred equals to loss because we calculate loss in output of custom_layer
      return y_pred


    model_center_loss.compile(optimizer= "sgd",
                              loss = ["categorical_crossentropy", custom_loss],  #An objective function is any callable with the signature loss = fn(y_true,y_pred)
                              loss_weights = [1, lambda_c/2],                    # scalar coefficients
                              metrics=  ["accuracy"])


    dummy_matrix1 = np.zeros((X_train.shape[0], 1))     #X_train.shape[0] = 60000
    dummy_matrix2 = np.zeros((X_test.shape[0], 1))      #X_test.shape[0] = 10000

    model_center_loss.fit(x = [X_train, y_train], y = [y_train_one_hot, dummy_matrix1],
                          batch_size = 32, epochs = 5, verbose = 1,
                          validation_data=([X_test, y_test], [y_test_one_hot, dummy_matrix2]))
```

در نهایت خروجی زیر که نشان دهنده این است که 10 کلاس که متعلق به اعداد 0 تا 9 هستند از هم جداسازی شده‌اند و مرکز کلاس‌ها هم با یک نقطه نشان داده می‌شود.



- A Discriminative Feature Learning Approach for Deep Face Recognition
  Yandong Wen, Kaipeng Zhang, Zhifeng Li and Yu Qiao.
  Shenzhen Key Lab of Computer Vision and Pattern Recognition.
  European Conference on Computer Vision
  ECCV 2016: Computer Vision – ECCV 2016