استخراج ویژگی و کلاسبندی تصاویر با شبکه های عصبی کانولوشن

پروژه کارشناسی

مهندسی کامپیوتر

ارائه شده به گروه مهندسی کامپیوتر دانشگاه بناب

استاد راهنما: دکتر مهدی حسین زاده

توسط: امیرمحّمد رمضانی

بهمن ۱۴۰۱

پیشگفتار

این پایان نامه برای توضیح کد و به صورت مکمل با کد تولید شده است و برای درک کامل این پایان نامه دسترسی به کد نیاز است. ابتدا در **چکیده** هدف، روش و نتیجه بیان شده است و پس از تعاریف اجمالی از کار های انجام شده در پروژه بدون زبان تکنیکی در فصل سوم یا همان **روش های یادگیری** از ابتدایی ترین روش ها و تفاوت بین انواع روش ها شروع شده و در نهایت به روش های یادگیری عمیق که در این پروژه و جهان واقعی استفاده می شوند میرسیم. در بخش چهارم که در مورد **تکنولوژی های مورد** استفاده در این پروژه و عموما در کل پروژه های انجام شده در پایتون مربوط به هوش مصنوعی اشاره می شود. در فصل پنجم به **ساختار پروژه** و روش های حل مسئله انتخاب شده آن میرسیم سعی میشود در این فصل روش انجام کار به صورت کامل و بدون چشم پوشی از هر نوع جزئیات توصیف شود. در بخش نتایج، داده های واقعی خروجی از این معماری شبکه های عصبی بین ۲ نوع الگوریتم کلاس بندی با ۴ معیار بررسی شده و بهترین آن ها نشان داده میشود. در پایان قبل از ارائه **لیست منابع** استفاده شده برای گردآوری این پایان نامه به نتیجه پروژه و روش های توسعه بیشتر این پروژه پرداخته می شود.

چکیده

این پایان نامه برای استفاده از تکنولوژی های مدرن مانند یادگیری عمیق و یادگیری ماشینی و قدرت روز افزون آن برای آماده سازی و منسجم کردن تصاویر ورودی و همچنین استخراج ویژگی انواع تصاویر داده شده به آن است.

محیط های انجام این پروژه گیت هاب، جوپیتر و سرویس کولاب گوگل بوده‌اند. برای این پروژه زبان پایتون به دلیل داشتن منابع زیاد آنلاین، کتابخانه های مورد نیاز و آشنا بودن شخصی برنامه نویس انتخاب شده است. این پروژه ممکن بود با زبان های دیگری مانند Java و JavaScript قابل انجام باشد اما احتمال انجام شدن در زبان های دیگر به سرعت و کیفیت پایتون کم است.

در پایتون از تعدادی کتابخانه استفاده شده است که بعضی از این ها عبارتند از: Numpy, Pandas, Tensorflow, Matplotlib. علاوه بر این کتابخانه ها از نرم افزار Adobe Photoshop برای بالا بردن کنتراست تصاویر و پیش آماده سازی آنها برای بخش اول استفاده شده است.

علاوه بر این تکنولوژی ها از ایده های شبکه های عصبی متراکم، کانولوشن، U-net و VGG-16 برای الگو گرفتن در طراحی شبکه های عصبی استفاده شده است.

نتایج گرفته از این پروژه نشان می دهند که استخراج ویژگی درست و اصولی خیلی مهم تر از الگوریتم کلاس بندی اهمیت دارد به صورتی که بیشتر شدن لایه های کانولوشن باعث بهتر شدن معیار هایی کارایی الگوریتم می شود. البته علاوه بر این نتایج گرفته شده برای کلاس بندی با الگوریتم های کرنل دار بسیار بهتر از الگوریتم های خطی هستند.

کلید واژه: هوش مصنوعی، پایتون، Tensorflow, Keras, Deep Learning, یادگیری عمیق، کلاس بندی، شبکه های عصبی کانولوشن، SVM، تصویر، VGG16، استخراج ویژگی، DCNN، یادگیری نظارت شده

فهرست مطالب

فصل ۱:

مقدمه

۱-۱- هوش مصنوعی

۱-۱-۱- تاریخچه هوش مصنوعی

تعریف های بی شماری از هوش مصنوعی[1] در دهه های اخیر به وجود آمده است. برای مثال جان مکارتی[2] در سال ۲۰۰۴ این تعریف را ارائه کرد «هوش مصنوعی در واقع علم و مهندسی ساختن ماشین های هوشمند است اما هوش مصنوعی نیازی ندارد که خودش را به چهارچوب محسوس زیستی محدود کند.»

اما دهه ها قبل تر از این تعریف تولد هوش مصنوعی توسط آلن تورینگ[3] در مقاله هوشمندی و ماشین های کامپیوتری ذکر شده بود که در سال ۱۹۵۰ منتشر شد. آلن تورینگ یا پدر علوم کامپیوتر این سوال را میپرسد «آیا ماشین های می توانند فکر کنند؟» از آن جا به سمت تست معروف تورینگ می رسد. در این تست سعی می شود که یک بازجو انسان بتواند تشخیص دهد چیزی که از آن بازجویی میشود انسان است یا ماشین؟

با انتشار کتاب رویکردی نوین به هوش مصنوعی استوارت راسل[4] و پیتر نورویگ[5] به هدف های ممکن و تعریف های مختلف هوش مصنوعی پرداختند. آن ها بین کامپیوتر ها با منطق عمل و رفتارشان تفاوت قائل می شوند. که به چهار گروه میرسند.

انسانی:

- ماشین هایی که شبیه به انسان فکر میکنند.
- ماشین هایی که شبیه به انسان عمل میکنند.

ایده‌آل:

- ماشین هایی که منطقی فکر میکنند.
- ماشین هایی که منطقی عمل میکنند.

[1] Artificial Intelligence

[2] John MacKarthy

[3] Alan Turing

[4] Stuart Russell

[5] Peter Norvig

تعریف آلن تورینگ در دسته ماشین هایی که مانند انسان عمل می کنند قرار می گیرد.

در حالت ساده هوش مصنوعی یک رشته است که برای حل کردن مسائل از قدرت داده و علوم کامپیوتر استفاده می کند. همچنین دارای زیر-رشته هایی مانند یادگیری ماشینی[6] و یادگیری عمیق[7] هست. این متد های برای ساختن سیستم های متخصص برای پیش بینی[8] یا کلاس بندی[9] داده های ورودی هست.

۱-۱-۲- کاربرد امروزه هوش مصنوعی

با وجود زمستان های هوش مصنوعی (دورانی که توسعه علم جدید برای چندین سال ممتد به صورت قابل توجه کاهش یافته است) پس از سال ۲۰۱۴ با منتشر شدن مقالات زیادی برای یادگیری عمیق امروزه این زیر-رشته از محبوبیت زیادی برخوردار است. شرکت های متعددی از آن برای کلاس بندی اشیا و تصاویر، تشخیص گفتار مانند دستیار های هوشمند و حرف زدن کامپیوتر با NLP استفاده میکنند. البته بسیار از دولت ها و اندیشمندان هوش مصنوعی برای قانون گذاری درست و جا انداختن اخلاقیات استفاده از آن تلاش میکنند.

۱-۲- زبان برنامه نویسی پایتون

پایتون[10] یک زبان برنامه نویسی قدرتمند و کامل است که ساختمان داده های سطح بالا را در خود دارد و همچنین رویکردی ساده اما موثر به برنامه نویسی شی گرا دارد. نحو مدرن و نوع داده های پویا به همراه طبیعت تفسیری پایتون، آن را زبانی ایده‌ال برای ساختن برنامه بر روی اکثر سیستم عامل ها میکند.

[6] Machine Learning
[7] Deep Learning
[8] Regression
[9] Classification
[10] Python

مفسر[11] پایتون و کتابخانه استاندارد آن در سایت پایتون به صورت کد منبع یا باینری برای تمامی سیستم عامل های اصلی موجود هستند. مفسر پایتون به راحتی میتواند با تابع های جدید و نوع داده های پیاده‌سازی شده در C یا C++ توسعه یابد.

گویدو وان روسم[12] در اواخر دهه ۸۰ میلادی بر روی پایتون به عنوان جانشین زبان ABC شروع به کار کرد و اویل نسخه آن را در سال ۱۹۹۱ به عنوان پایتون ۰.۹ منتشر کرد. در سال ۲۰۰۰ پایتون ورژن ۲.۰ و در سال ۲۰۰۸ پایتون نسخه ۳.۰ توسعه یافتند. نسخه سوم پایتون اولین نسخه است که به صورت کامل با نسخه های قبلی مطابق نیست.

۱-۲-۱- اصول پایتون

زبان پایتون توسط درخواست های بهبود پایتون (PEP) تغییر می یابد. یکی از مهم ترین آن ها اصول پایتون یا Zen of Python از ۱۹ اصل راهنما برای توسعه زبان توسط تیم پیترز در سال ۱۹۹۹ نوشته شده‌اند. برخی از اصول عبارتند از:

- زیبا بهتر از زشت است.
- مستقیم بهتر از غیرمستقیم است.
- ساده بهتر از مجتمع است.
- مجتمع بهتر از پیچیده است.
- مسطح بهتر از تو در تو است.
- خلوت بهتر از مزدحم است.
- خوانایی مهم است.
- ...

[11] Interpreter

[12] Guido Van Rosam

پایتون زبان برنامه نویسی چند-پارادایم[13] است. برنامه نویسی شی گرا[14] و برنامه نویسی ساختاری به صورت کامل پشتیبانی میشوند. همچنین بیشتر قابلیت های برنامه نویسی جهت گرایی (مانند برنامه نویسی متا و شی های متا) پشتیبانی میشوند. البته پارادایم های دیگر را نیز میتوان با افزونه های اضافه کرد.

۱-۲-۲- پیاده سازی پایتون

پایتون از نوع متغیر های پویا و ترکیبی از شمارش ارجاع[15] و زباله دان[16] تشخیص دور برای مدیریت حافظه استفاده میکند. از انقیاد دیررس[17] که در زمان اجرا دست به کار می شود استفاده میکند. همچنین به لطف توابع map, reduce و filter تا حدی میتواند از برنامه نویسی تابع گرا Lisp مانند پشتیبانی کند و ماژول های itertools و functools از زبان های Haskell و Standard ML آمده‌اند.

نوشتن کد با زبان پایتون به شدت ساده است به طوری که یک برنامه نویس ماهر میتواند در عرض چند ساعت کد به درد بخور در پایتون بنویسد. برای همین در محیط های آکادمیک و دوره های مبتدی از پایتون استفاده میشود.

۱-۳- استخراج ویژگی[18]

استخراج ویژگی نوعی کاهش ابعاد است که تصویر بزرگی متشکل از تعداد زیادی پیکسل به نوعی نشان داده میشوند که بخش های جالب تصویر به صورت موثر نشان داده شود. اگر دقیق تر بگوییم این عمل یعنی منتقل کردن داده های انواع مختلف مانند متن یا تصویر به ویژگی های عدد برای استفاده در یادگیری ماشینی است.

این عمل عموما هنگامی اتفاق می افتد که ورودی های یک الگوریتم خیلی بزرگ هستند و برنامه نویس مشکوک است که این مقادیر تکراری باشند که در پروسه یادگیری تاثیری ندارد و فقط هزینه پردازش دارند. استخراج

[13] Multi-Paradigm
[14] Object Oriented
[15] Reference Counting
[16] Garbage Collector
[17] Lazy Binding
[18] Feature Extraction

ویژگی در بسیاری از موارد میتواند از مشکل Over-fit جلوگیری کند چرا که نمیگذارد الگوریتم بر روی یک ورودی بیش از نیاز تمرکز کند و باعث کلی شدن مدل میشود.

۱-۳-۱- چند نمونه از الگوریتم های استخراج ویژگی

- آنالیز کامپوننت مستقل ,ICA
- Isomap
- Kernel PCA
- Latent semantic analysis
- Partial least squares
- PCA
- Autoencoder

۱-۳-۲- استخراج ویژگی در تصویر

در پردازش تصویر بعد از شناخت نقاط ویژگی پیکسل ها و یا تکه های اطراف آن می توانند استخراج شوند. با اینکه این عمل پردازش سنگینی است اما به نسبت پردازش کل تصویر بسیار سبک تر است. نتیجه در بردار[19] های ویژگی ذخیره می شود. تشخیص ویژگی می تواند در انواع سطح های مانند تشخیص گوشه، شکل، حرکت یا انحنا باشد.

۱-۳-۳- استخراج ویژگی با شبکه های عصبی و یادگیری عمیق

در پروژه به جای استفاده از تکنیک های استخراج ویژگی های متعارف که خروجی بسیار خوبی ندارند. از شبکه های عصبی کانولوشن برای استخراج ویژگی از تصاویر استفاده شده است. تکنیک ها در بخش های بعدی به صورت دقیق بیان می شوند. اما در کل از دو نوع ساختار شبکه عصبی استفاده کرده‌ایم که ابتدا از شبکه عصبی

[19] Vector

عمیق کانولوشن دستی با سه لایه کانولوشن و یک لایه نرمالیزه[20] کردن استفاده شده است و سپس از شبکه نیمه آماده[21] VGG16 که همانطور که از نامش معلوم است از ۱۶ لایه تشکیل شده است.

۱-۳-۴- پیاده‌سازی الگوریتم های استخراج ویژگی

برای تشخیص و استخراج ویژگی میتوان از پکیج های مختلفی که در MATLAB, SciLab, NumPy, scikit-learn و زبان R هستند استفاده کرد. اما در این پروژه این عمل با استفاده از شبکه های عصبی عمیق[22] انجام شده است.

۱-۴- توضیحات فصول

در فصل اول یا مقدمه هدف و انگیزه این پروژه با توضیح تکنولوژی و امکانات استفاده شده در این پروژه شروع میکنیم. این فصل سعی میکند خواننده را برای جزئیات تکنیکی در فصل های بعد آماده کند. در این فصل از دادن جزئیات مربوط به پیاده سازی پروژه پرهیز شده است.

فصل دوم با روش های مختلف یادگیری در کامپیوتر ها شروع میکند و به تفاوت بین یادگیری عمیق و یادگیری ماشینی میپردازد. سپس انواع روش های منتقل کردن داده و نظارت را توضیح می دهد و که برای تکمیل کردن پروژه دانستن نوع نظارت و داده بسیار مهم است. بعد از آن انواع متد ها را با توجه به نوع یادگیری توضیح داده میشوند. ابتدا از ساده ترین روش ها مانند رگرسیون که در نرم افزار های آماری نیز یافت میشوند شروع میکنیم و پس از توضیح روش های یادگیری ماشینی به یادگیری عمیق میرسیم که در انتها با کانولوشن[23] ها تمام میکنیم.

برای تمامی روش های یادگیری که دارای فرمول های سخت و غیر قابل محاسبه توسط انسان نیستند، فرمول یادگیری و معیار با توضیحات اضافی بیان شده است.

[20] Normalization
[21] Pre-Trained
[22] Deep Neural Networks
[23] Convolutions

پس از روش های یادگیری در فصل سوم به فریمورک ها یا همان کتابخانه های پایتون که مورد استفاده در صنعت هوش مصنوعی است میرسیم. واضح است که ممکن نیست کل کتابخانه های استفاده شده در این پروژه را مفصل مورد بررسی قرار داد و فقط کتابخانه های مهم و بزرگ یا متوسط نوشته شده‌اند.

توضیحات در مورد پروژه در فصل چهارم به صورت دقیق و با جزئیات کامل بیان شده است. این فصل به سه بخش ۱) آماده سازی داده ۲) استخراج ویژگی و در نهایت ۳) کلاس بندی تقسیم شده است. ابتدا از نوع داده های ورودی که در شکل ۱۵ کلاس و تصاویر با نسبت ۱۶ به ۹ هستند شروع میکنیم و پس از آماده کردن آنها به منتقل کردن به شبکه های عصبی عمیق می پردازیم و در پایان معماری و روش های متد های کلاس بندی کننده را بررسی میکنیم.

پس از توضیح روش انجام کار در فصل پنجم به نتایج گرفته شده میرسیم که شامل اطلاعاتی مانند دقت و سایر معیار های کارایی یک متد یادگیری برای کلاس بندی است.
در آخرین بخش یا همان فصل ششم از این پایان نامه سعی میکنیم قبل از بیان کردن مراجع و منابع نتایج گرفته شده را جمع بندی کنیم که بتوانیم در صورت امکان نتایج را بهبود دهیم.

فصل ۲:

روش های مختلف یادگیری

۲-۱ انواع داده برای یادگیری

۲-۱-۱- یادگیری نظارت شده و یادگیری نظارت نشده

در یادگیری نظارت شده[24] الگوریتم انتخاب شده با استفاده از مجموعه داده نامگذاری شده که نام ها به عنوان جواب برای اندازه‌گیری قدرت و بازدهی الگوریتم استفاده می شوند. در مدل نظارت نشده[25] برعکس این عمل اتفاق می افتد به صورتی که الگوریتم سعی میکند بدون نام ها با استفاده از تشخیص و استخراج ویژگی از داده ورودی معنی پیدا کند.

۲-۱-۲- یادگیری نیمه نظارت شده

البته روش دیگر هم به نام یادگیری نیمه نظارت[26] شده است که بخش کوچکی از داده ورودی به صورت نامگذاری شده است و الگوریتم سعی میکند ابتدا با استفاده از این داده ها به کل ورودی نام و لیبل اضافه کند.

۲-۱-۳- یادگیری تقویتی

یادگیری تقویتی[27] نیز با استفاده از سیستم جایزه دهی سعی میکند به الگوریتم یاد بدهد. برای این کار یک ایجنت داریم که با توجه به تصمیمات و عمل هایش فیدبک میگیرد.

۲-۲ تفاوت یادگیری عمیق و یادگیری ماشینی

در کل یادگیری ماشینی زیر مجموعه هوش مصنوعی است که یادگیری عمیق هم درون یادگیری ماشینی جای میگیرد. در یادگیری ماشینی سیستمی طراحی میشود که بدون دخالت برنامه نویس بتواند یاد بگیرد و خودش را با توجه به ورودی هایش تطبیق دهد.

[24] Supervised

[25] Unsupervised

[26] Semi-Supervised

[27] Reinforcement Learning

یادگیری عمیق نوعی تکنیک یادگیری ماشینی است که لایه هایی از نورون های مصنوعی دارد و سعی میکند مغز انسان را شبیه سازی کند. داده از این لایه ها به صورت غیر خطی عبور میکند شبیه به پروسه یادگیری ذهن انسان. این نوع یادگیری به صورت عمومی به داده بیشتری نیازمند است.

۳-۲ روش های یادگیری ماشینی

۲-۳-۱- رگرسیون خطی[28]

در این الگوریتم یک مدل خطی با محاسبه جمع وزن دار کلیه ویژگی های ورودی (متغیر های مستقل) و اضافه کردن بایس پیشبینی میکند.
معادله خطی مانند:

$$\widehat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

که در آن $\widehat{y}$ همان مقدار پیشبینی شده است، بایس با تتا صفر نشان داده شده است، n تعداد ویژگی ها و x ویژگی های ورودی هستند.

این معادله به صورت وکتوریزه به این حالت در می آید.

$$\widehat{y} = h_0(x) = \Theta . x$$

در این حالت هم تتا پارامتر های مدل یا همان وزن ها است که در ابتدا به آن مقدار های تصادفی می دهیم و تتا صفر یا همان بایس را هم شامل میشود. تابع h هم تابع پیشبینی است.

[28] Linear Regression

برای بهینه سازی[29] مدل از MSE یا کوچکترین ارور میانگین استفاده میکنیم. که جمع مربع فاصله بین عدد پیشبینی شده و خروجی واقعی است و بر تعداد ورودی m تقسیم میشود.

$$MSE(X,\ h_0)\ =\ \frac{1}{m_i}\sum_{i=1}^{m}(\theta^T x^{(i)} - y^{(i)})^2$$

سعی میکنیم که مقدار MSE به حداقل خود برسد. برای بهینه سازی از کاهش گرادیان[30] استفاده میکنیم. این کار نیازمند عوض کردن پارامتر ها (تتا ها) است به طوری که تابع هزینه به حداقل (مینیموم) خود برسد. برای این کار گرادیان تابع ارور (MSE) نسبت به تتا محاسبه می شود و در جهت مخالف آن حرکت میکنیم. هنگامی که مقدار ارور به صفر برسد به مقصد رسیده‌ایم.

یکی از مهم ترین چیز ها در کاهش گرادیان یافتن سرعت یادگیری (اِتا - η) است که اگر بزرگ باشد میتواند مینیموم را نادیده بگیرد و هرگز آن را پیدا نکند. اما اگر سرعت یادگیری[31] کم باشد میتواند با محاسبات خیلی زیادی به مینیموم برسد که باعث اتلاف وقت و قدرت پردازشی میشود.

یکی از مشکل های دیگر در کاهش گرادیان این است که شکل تابع هزینه شاید قوسی[32] کامل نباشد و دارای اکسترمم های محلی باشد و در این حالت الگوریتم می تواند در مینیمم محلی گیر کند و هیچگاه به صفر نرسد. برای جلوگیری از این موضوع باید داده های ورودی را نرمال کنیم که در بازه بین -۱ و ۱ باشند.

یکی از محبوب ترین روش های کاهش گرادیانی روش SGD یا Stochastic Gradient Descent است.
تفاوت این روش با کاهش ساده در این نکته نهفته است که بجای استفاده از کل داده برای یافتن گرادیان از یک داده استفاده میکنیم و در هر مرحله این داده را تغییر می دهیم. همچنین در این روش مشکل مینیما های محلی را نداریم.

[29] Optimization
[30] Gradients
[31] Learning Rate
[32] Convex

۲-۳-۲- درخت تصمیم گیری[33]

در این الگوریتم مانند درخت دودویی یک ساختمان داده درختی طراحی میکنیم که در هر مرحله با پاسخ دادن True یا False به مرحله عمیق تر منتقل میشویم. برای هر گره نیازمند محاسبه Gini هستیم که خالص بودن تتا را پیدا کنیم. هر چه Gini به صفر نزدیک تر است آن ویژگی خالص تر است. البته می توان به جای Gini از Entropy استفاده کرد.

فرمول Gini به صورت زیر است:

$$G_i = 1 - \sum_{k=1}^{n} P_i,\ k^2$$

در این فرمول $P_i,\ k^2$ نشانگر نسبت نمونه های کلاس k بین نمونه های i است.

الگوریتم های استفاده شده برای درخت تصمیم گیری CART و ID3 هستند که اولی فقط فرزند های دوتایی ایجاد میکند و الگوریتم دومی می تواند بیشتر از دو فرزند ایجاد کند. این نوع الگوریتم میتواند هم در کلاسبندی و هم در تخمین (رگرسیون) استفاده شود و پایه الگوریتم دیگری بنام جنگل تصادفی هستند.

۲-۳-۳- جنگل های تصادفی[34]

در یادگیری با جنگل های تصادفی ابتدا تعدادی درخت تصمیم گیری طراحی میکنیم که هر یک بر روی بخش مختلفی از داده یادگرفته‌اند، سپس جواب ورودی را از کل این درخت ها گرفته و پیشبینی میکنیم. البته این تکنیک را میتوان با استفاده از جواب گیری از الگوریتم های مختلف هم پیاده‌سازی کرد.

۲-۳-۴- ماشین های برداری یا SVM

الگوریتم SVM یا همان ماشین های برداری پشتیبان، الگوریتم های قدرتمندی و انعطاف پذیری هستند که امکان کلاسبندی خطی یا غیر خطی، رگرسیون، و حتی شناخت داده های پرت (Outlier) را دارند. یکی از ضروری و محبوب ترین الگوریتم ها در یادگیری ماشینی که برای داده های کوچک و متوسط مناسب هستند. SVM برعکس

[33] Decision Tree

[34] Random Forests

رگرسیون خطی میتواند بر روی ویژگی های درجه چند ریاضی هم عمل کلاس بندی را انجام دهد. که این عمل را با اضافه کردن درجه به مدل و تبدیل ورودی درجه چند به ورودی خطی انجام می دهد.

البته همیشه اضافه کردن درجه خوب نیست چون باعث بالا رفتن هزینه محاسبات میشود و بجای این در SVM میتوان از کرنل[35] ها استفاده کرد که همان نتیجه‌ی اضافه کردن درجه را می دهند یکی از محبوب ترین کرنل های SVM کرنل Gaussian یا گاووسی است. در کتابخانه scikit-learn این نوع مدل SVC نام دارد.

کلاس بندی در SVM با محاسبه کردن تابع تصمیم گیری است که اگر نتیجه گرفته شده از آن مثبت باشد مربوط به کلاس است و اگر منفی باشد از آن کلاس نیست. در SVM معیار یا metric از ضرب کردن ماتریس وزن[36] ها در خودش و نصف کردن آن به دست می آید.

تابع کلاس بندی SVM:

$$W^T X + b = w_1 x_1 + \ldots + w_n x_n + b$$

تابع معیار:

$$\frac{1}{2} W^T W$$

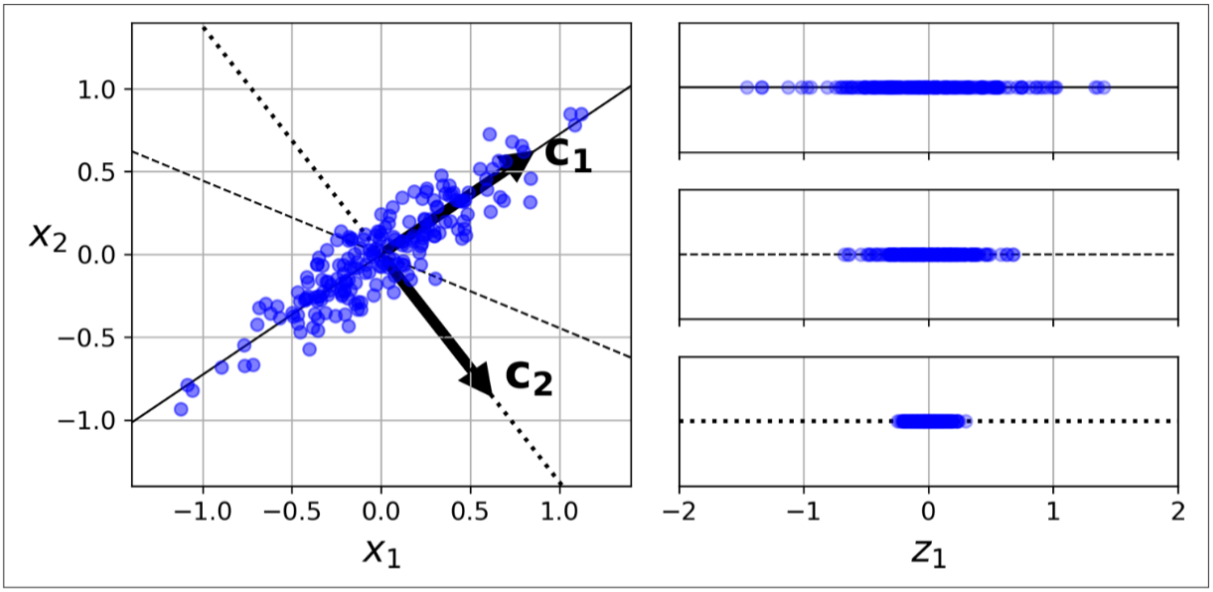
۲-۳-۵- کاهش ابعاد یا PCA

آنالیز کامپوننت اصلی مهم ترین الگوریتم در استخراج ویژگی و کاهش ابعاد است. این الگوریتم ابتدا با یافتن یک Hyperplane یا ابر صفحه که به داده ورودی نزدیک است و سپس نمودن داده بر روی آن این کار را انجام میدهد. برای انتخاب ابرصفحه باید دقت شود که ابرصفحه باید بتواند حداکثر واریانس[37] را حفظ کند. برای اینکه بعد از کاهش ابعاد حداکثر اطلاعات نگه داشته شود. ایده مهم دیگر برای PCA این است که فاصله مربعی میانگین بین شکل قبل کاهش و بعد کاهش به حداقل برسد.

[35] Kernel

[36] Weights

[37] Variance



در این شکل ما خط های C1 و C2 را انتخاب میکنیم که کلیه نقطه ها را بر روی آن منتقل کنیم.

ابتدا این الگوریتم خطی را انتخاب میکند که بیشترین مقدار واریانس را حفظ کند. همچنین محور[38] بعدی باید دقیقا قائم بر محور اول باشد، این محور برای نگهداری باقیمانده واریانس است. در واقع این محور ها principal component نام دارد که به معنی بخش اصلی داده هستند.

خوشبختانه روش عدد برای یافتن محور ها با فاکتور گیری از ماتریس ها داریم. این روش SVD نام دارد. در این روش ماتریس ورودی یا همان X را به سه ماتریس U Σ V^T استخراج میکنیم. به طوری که V شامل بردار تمامی محور های خواسته شده است. الگوریتم SVD را میتوان در کتابخانه numpy و در ماژول linalg یافت.

برای به دست آوردن ورودی ها بعد از کاهش ابعاد باید ورودی ها را در d ستون اول ماتریس وزن ها ضرب کنیم.

$$X_{d-proj} = XW_d$$

[38] Axis

از PCA می توان در کاهش حجم و زیپ کردن اطلاعات نیز استفاده کرد به طوری که اگر روی مجموعه داده MNIST این کار را انجام دهیم تعداد ورودی ها را از ۷۸۴ به ۱۵۰ می رسانیم در حالی که ۹۵ درصد از اطلاعات حفظ می شوند.

۲-۳-۶- میانگین چند نقطه

ابتدا به صورت تصادفی چند نقطه وسطی در میان داده قرار میدهیم. بعد از نامگذاری به تمامی داده های اطراف نقطه وسطی این نقاط را به اصلاح آپدیت میکنیم. این عمل تا زمانی ادامه پیدا میکند که نقاط وسط حرکت نکنند. این الگوریتم قطعا جدا سازی انجام میدهد اما در مورد درست یا غلط بودن آن قطعیت نداریم. برای بهتر کردن خروجی میتوان نقاط وسط را به صورت دستی مشخص کرد یا ممکن است این الگوریتم را به صورت تصادفی چند بار اجرا کنیم.

معیار اصلی هنگام استفاده از این الگوریتم inertia است که یعنی مجموع فاصله هر نقطه از نقطه مرکزی آن. اما این معیار مشکلاتی دارد برای مثال با افزایش نقاط امتیاز بالاتری نیز می دهد که این همیشه درست نیست. یکی از راه های دیگر یافتن تعداد نقاط مرکزی[39] استفاده از مقدار بازو است که نمودار افزایش inertia را بر اساس افزایش تعداد نقاط مرکزی میکشیم و سپس شکلی مانند بازو انسان ظاهر میشود و راس بازو مشخص کننده تعداد نقاط مورد نیاز است.

۲-۳-۷- خوشه بندی[40]

مانند کلاس بندی در این روش یادگیری نظارت نشده ورودی های مختلف به گروه های مختلف داده میشوند. این الگوریتم هنگامی مورد نیاز است که داده لیبل دار نداریم و نمی توانیم از روش های یادگیری مانند رگرسیون، SVM و جنگل تصادفی استفاده کنیم.

روش های یادگیری خوشه بندی کاربرد های گوناگونی دارند، که چند نمونه عبارتند از:

- گروه بندی مشتریان: بر اساس خرید و فعالیت مشتریان در وبسایت ممکن است کاربران را خوشه بندی یا گروه بندی کرد. این عمل برای پیشنهاد دهی به کاربران از اهمیت زیادی برخوردار است.

[39] Centroids

[40] Clustering

- آنالیز داده: ممکن است قبل آنالیز داده با استفاده از این الگوریتم داده را به خوشه های کوچکتری تبدیل کنیم و روی هر یک آنالیز مختلفی انجام دهیم.
- روش کاهش ابعاد: بعد از خوشه بندی می توان در هر خوشه مقدار affinity (معیار تطابق با خوشه انتخاب شده برای هر ورودی) را اندازه گرفت. سپس بجای نگهداری ویژگی های اضافی فقط مقادیر affinity را نگه داشت.
- برای شناخت داده غیر عادی[41]: مانند روش قبل با اندازه گرفتن مقدار affinity میتوان نقاط داده‌ای که امتیاز کمتری می گیرند را فیلتر کرد. این عمل ممکن است برای تشخیص رفتار غیر عادی کاربران یا سنسور ها انجام شود.
- یادگیری نیمه نظارت شده: اگر به تعداد کل داده ها لیبل نداشته باشیم میتوان با خوشه بندی و اضافه کردن لیبل به کل داده های تحت یک خوشه آن مجموعه داده را به دادی نامگذاری شده تبدیل کرد.
- سگمنت کردن تصویر (بخش بندی تصویر): با خوشه بندی بر مبنا رنگ پیکسل های همسایه و جایگزین کردن رنگ پیکسل های خوشه با میانگین مقدار رنگ، ممکن است که تعداد رنگ در یک تصویر را به شدت کاهش دهیم. این روش در پروژه های بسیاری مانند تشخیص اشیا و سیستم های تعقیب کننده و حتی در این پروژه استفاده شده است.

۴-۲ روش های یادگیری عمیق

۲-۴-۱- شبکه های عصبی

شبیه سازی شبکه های عصبی اولین بار توسط وارن مک کلوک[42] و والتر پیتز[43] در سال ۱۹۴۳ انجام شده است. در این پروژه سعی شد که مدل محاسباتی بر اساس نورون[44] های مغز حیوانات ساخته شود و بر اساس منطق گزاره‌ای کار کند. اما این پروژه نتوانست به صورت مطلوب به مسئله های جهان واقعی را حل کند و بعد از این شاهد دو زمستان هوش مصنوعی شده‌ایم.

[41] Anomaly Detection
[42] Mac Clock
[43] Walter Peters
[44] Neuron

امروزه پرسپترون[45] به عنوان ساده ترین مدل شبکه عصبی استفاده میشود که ممکن است با نام های TLU یا LTU شناخته شود. ورودی و خروجی های این واحد همگی عدد هستند و با مقادیر بول[46] کار نمیکنند. برای هر ورودی یک وزن تعیین شده است. هر واحد جمع وزندار ورودی هایش را محاسبه میکند و سپس تابع غیر خطی به آن اعمال میکند.

فرمول شبکه های عصبی:

$$h_w(x) = f(z)$$
$$z = x^T w$$

توابع بسیاری ممکن است برای تابع غیرخطی انتخاب شوند. مانند تابع تانژانت هیپربولیک،[47] ساین، ReLU و یا سیگموئید.

اما در بیشتر حالات یک پرسپترون کفایت نمیکند (در یک لایه حتی قادر به حل کردن XOR نیست) و برای درک روابط پیچیده توسط کامپیوتر لازم است که این واحد ها را به واحد های لایه قبلی وصل کنیم که به این لایه، لایه کاملا متصل[48] یا لایه متراکم[49] می گویند. اولین لایه از این واحد ها (نورون ها) لایه ورودی است که هیچ نوع تغییری روی مقدارهایش انجام نمیدهد. علاوه بر این در هر لایه یک ویژگی بایس هم با نام b اضافه میکنیم.

یادگیری شبکه تشکیل شده با قانون هب[50] است که در زیست شناسی استفاده گسترده دارد. دونالد هب پیشنهاد میدهد که وقتی یک نورون با نورون دیگر در مغز ارتباط زیادی دارد، این ارتباط برای هر دو نورون تقویت میشود که در شبکه های عصبی مصنوعی این کار را با ماتریس وزن ها انجام میدهیم. این تقویت ارتباط برای کاهش معیار خطا نیاز است.

[45] Perceptron
[46] Boolean
[47] Hyperbolic Tangent
[48] Fully Connected Layer
[49] Dense Layer
[50] Hebb's Rule

قانون یادگیری شبکه عصبی مصنوعی:

$$w_{i,j}^{(next\ step)} = w_{i,j} + \eta(y_j - \hat{y}_i)x_i$$

که در این معادله w همان ماتریس وزن ها هست و وزن بین دو نورون i و j را مشخص میکند. نماد اِتا برای سرعت یادگیری است، x همان مثال ورودی مورد استفاده است، $\hat{y}$ مقدار خروجی شبکه عصبی است و y مقدار واقعی خروجی این مثال است. به دلیلی خطی بودن حد تصمیم گیری، پرسپترون قادر به یادگیری الگو های پیچیده نیست. اما برای هر الگو خطی به نتیجه می رسد. به دلیل شباهت بسیار بین پرسپترون و کاهش گرادیان در بیشتر کتابخانه های هوش مصنوعی این دو را میتوان با الگوریتم SGD Classifier پیاده سازی کرد.

۲-۴-۲- یادگیری بازگشتی[51]

یک پرسپترون چند لایه از یک لایه ورودی، چند لایه نورون (لایه های مخفی) و یک لایه خروجی تشکیل شده است. لایه های نزدیک به ورودی لایه های پایینی و لایه های نزدیک به خروجی لایه های بالایی تلقی میشوند.

سال های طولانی تحقیقات در حوزه شبکه های عصبی به جایی نمی رسید اما این با منتشر کردن مقاله‌ای توسط جافری هینتون،[52] دیوید راملهارت[53] و رونالد ویلیامز[54] به پایان رسید. این دانشمندان در این مقاله ایده‌ی یادگیری بازگشتی را بیان کردند. در حالت کلی این ایده عبارت است از یکبار محاسبه رو به جلو که همان محاسبه شبکه های پرسپترون است و بار دیگر محاسبه در عکس جهت نورون ها و تنظیم کردن دوباره وزن ها است. برای تنظیم دوباره الگوریتم بازگشتی باید گرادیان ایراد شبکه را نسبت به هر پارامتر مدل محاسبه کند (همان auto differentiate در Tensorflow). هر عبور رو به جلو و عقب با هم را epoch می گویند.

[51] Backpropagation
[52] Geoffery Hinton
[53] David Ramlheart
[54] Ronald Williams

۲-۴-۳- اتوانکودر[55]

این نوع شبکه عصبی با کپی کردن ورودی ها به خروجی ها یاد می گیرد. شاید این نوع یادگیری ساده به نظر بیاید اما برنامه نویس سعی میکند این کپی کردن را تا حدی دشوار کند. برای مثال این کار با اضافه کردن نویز به تصویر یا محدود کردن تعداد نورون های قابل استفاده انجام می گردد. این سختی ها باعث می شوند که شبکه عصبی مجبور شود که راه های حفظ داده مورد نیاز را خود به خود پیدا کند.

اتوانکودر به صورت نظارت نشده یاد می گیرد و به اطلاعات یاد گرفته شده توسط این شبکه اطلاعات latent یا کدینگ ها گفته می شود. این خروجی ها معمولا نسبت به ورودی ها ابعاد کوچک تری دارند. یکی دیگر از کاربرد های اتوانکودر ها یادگیری یا استخراج ویژگی ها است. البته ممکن است به صورت تصادفی بتوانند داده تولید کنند که عموما این داده ها از وضوح خوبی برخوردار نیستند و استفاده کردن از GAN برای این کار بسیار عاقلانه تر است.

۲-۴-۴- شبکه های عصبی کانولوشن[56]

طرح اولیه این شبکه های عصبی اولین بار توسط دیوید هوبل[57] و تورستن ویزل[58] در سال ۱۹۵۸ هنگام آزمایش کردن بر روی گربه ها ارائه شد. بعد از این تحقیقات نشان داده شد که چندین نورون در چشم منطقه درک محلی دارند که سایز مشخصی ندارند یعنی فقط بر روی تحریک در یک منطقه واکنش نشان می دهند. البته این مناطق ممکن است با همدیگر نقاط مشترکی داشته باشند. علاوه بر این نویسندگان نشان دادند که بعضی از نورون ها فقط به خطوط افقی واکنش نشان می دهند در حالی که نوع دیگر به زوایای دیگر حساس هستند.

مهمترین بخش شبکه های عصبی کانولوشن لایه های کانولوشن هستند. این نوع لایه ها برعکس لایه های متراکم نیازی به متصل کردن هر نورون به نورون در لایه دیگر ندارند و فقط به نورون های منطقه درک محلی خود متصل هستند. این نوع معماری اجازه می دهد که شبکه در لایه های سطح پایین بر روی جرئیات تمرکز کند و در سطوح

[55] Autoencoder

[56] Convolutional Neural Networks

[57] David Hobel

[58] Trosten Wisel

بالا بر روی کلیات تمرکز کند. که این نوع معماری هم از جهان واقعی الهام گرفته شده است و برای همین شبکه های کانولوشن برای شناخت تصویر نتیجه خوبی می دهند.

فرمول پیکسل ها بعد از کانولوشن:

$$W_{out} = \frac{W - F + 2P}{S} + 1$$

در این فرمول W سایز تصویر ورودی است، F سایز منطقه درک محلی، S اندازه قدم ها یا Stride و P همان پدینگ یا حاشیه اضافه شده توسط الگوریتم به ورودی است.

فصل ۳:

فریمورک های پایتون

۳-۱- کتابخانه های تصویر

۳-۱-۱- کتابخانه PIL

کتابخانه تصویر پایتون (PIL) قابلیت های پردازش تصویر از فرمت های مختلف را به مفسر پایتون اضافه میکند. محاسبات این فریمورک[59] بسیار سریع و کم هزینه هستند و تقریبا تمامی قابلیت های پردازش تصویر را به اختیار برنامه نویس میگذارد. این فریمورک منبع باز است و نخستین بار تحت ورژن ۱.۱.۷ در سال ۲۰۰۹ عرضه شد و از سال ۲۰۱۱ به Pillow تبدیل شد. چند نمونه از قابلیت های این فریمورک عبارتند از:

- تغییر در سطح پیکسل
- پشتیبانی از ماسک کردن و سطح بندی لایه ها
- قابلیت های بلور و واضح کردن تصویر
- اضافه کردن متن به تصویر
- خروجی تصویر به آرایه عددی

۳-۲- کتابخانه های عددی و محاسباتی

۳-۲-۱- کتابخانه SciPy

فریمورک منبع باز برای محاسبات عدد و علمی دارای ماژول[60] هایی برای عملیات آماری، بهینه سازی خطی، جبر خطی، انتقال فوریه و پردازشگر تصویر و سیگنال. این فریمورک برای عملیات روی آرایه های نامپای طراحی شده است. برای تمامی سیستم های عامل قابل اجرا است و در حال حاضر با لایسنس BSD توزیع می شود و برنامه نویس های داوطلب آن را پشتیبانی میکنند. اولین بار در سال ۲۰۰۱ منتشر شد و با رشد IPython و Matplotlib رشد قابل توجه داشته است.

[59] Framework

[60] Modules

۳-۲-۲- کتابخانه Numpy

لیست ها و آرایه های پایتون بخاطر قابلیت نوع پویا سرعت محاسبات کمی دارند و با افزایش بعد و پیچیدگی هزینه‌ی محاسباتی زیادی خواهند داشت. کتابخانه نامپای قابلیت استفاده از آرایه و ماتریس های بزرگ چند بعدی به همراه تابع های سطح بالای ریاضی بر روی آن ها را به برنامه نویس میدهد. این کتابخانه اولین بار توسط نام Numeric به دست جیم هاگونین[61] ساخته شد که در سال ۲۰۰۵ ترویس اولفنت[62] نام آن را به نامپای تغییر داد و به صورت منبع باز منتشر کرد.

پایتون بخاطر مفسری بودن عموما نمی تواند کد ها را بهینه کند و سرعت کمتری از زبان کامپایل شده دارد. این کتابخانه به لطف مفسر CPython که از زبان C برای تفسیر کد به بایت کد استفاده میکند کار میکند. استفاده از نامپای در پایتون آن را شبیه به زبان مفسری دیگر یا همان MATLAB میکند. اما بخاطر ارتباط نزدیک بین نامپای و پایتون کار کردن با این کتابخانه نسبت به کار کردن با متلب[63] و سیمولینک[64] آسان تر است.

قدرت اصلی نامپای در ساختمان داده هایی به نام ndarray است که آرایه های n بعدی هستند. که همانطور که گفته شد این نوع آرایه ها برعکس لیست های پایتون یک نوع دارند و *نمی توان* نوع های مختلفی را در آن جای داد. در نامپای روشی به نام Vectorization باعث میشود که عملیات بدون نیاز به حلقه ها اجرا شوند و این در برخی موارد میتواند تا ۳۰ برابر از حلقه سریعتر شود. همچنین وجود تابع های Universal سرعت عملیات ریاضی را نسبت به پایتون به صورت قابل توجه افزایش میدهد.

۳-۲-۳- کتابخانه Pandas

یکی دیگر از مهمترین کتابخانه های آنالیز و تغییر داده پانداز است. پانداز که بر روی کتابخانه نامپای و سایپای پیاده سازی شده است، ساختمان داده و عملیات برای تغییر اعداد، تاریخ، و متن را به برنامه نویس میدهد. این

[61] Jim Hagonin
[62] Travis Olofent
[63] Matlab
[64] Symolink

کتابخانه نرم افزار آزادی است که تحت لیسانس BSD منتشر شده است. که در سال ۲۰۰۸ برای رفع نیاز برنامه نویسان اقتصادی توسط وس مکینی[65] ساخته شده بود.

این پکیج هم مانند نامپای برای سرعت بخشیدن به عملیات از روش وکتوریزه کردن به جای حلقه استفاده می کند. ساختمان داده های مورد استفاده در پانداز فریم های داده[66] هستند. اطلاعات این نوع ساختمان داده میتواند از فایل های CSV, SQL, Parquet, JSON و فایل های اکسل بیاید. که بعد ها می توان روی این داده ها عملیاتی از قبیل تغییر شکل، انتخاب (اندیس کردن)، چسباندن و تمیز کردن داده انجام داد. اکثر ایده های استفاده شده در این ساختمان داده از زبان برنامه نویسی R الهام گرفته شده‌اند.

نوع دیگر از ساختمان داده های این کتابخانه سری ها (Series) هستند. که بسیار شبیه به ndarray ها در نامپای هستند.

۳-۳- برنامه نویسی علمی

۳-۳-۱- کتابخانه IPython

به دلیل اینکه پایتون زبان مفسری است و زبان های مفسری را میتوان به صورت خط به خط اجرا کرد. ممکن است که کد های پایتون به صورت تعاملی[67] و جدا جدا اجرا شوند IPython به ما این قابلیت را میدهد.

علاوه بر اینها میتوان کد ها را به طور موازی یا توزیع شده اجرا کرد و برنامه نویس می تواند بیشتر از روش های دیگر در پروسه تولید و اجرا کد قرار بگیرد. البته بعد ها قابلیت های اجرا کردن موازی اختیاری شدند و در پکیج دیگری به نام ipyparallel توزیع شدند. همچنین IPython با کتابخانه های علمی پایتون یکپارچه است و

[65] Wes McKinney

[66] DataFrame

[67] Interactive

امکان استفاده راحت تر از آن ها را به برنامه نویس میدهد برای مثال رندر کردن تصاویر تولید شده در Matplotlib و Seaborn. این کتابخانه به صورت کامل از LaTeX پشتیبانی میکند.

۳-۳-۲- کتابخانه Jupyter

به دلیل اجرا شدن IPython در ترمینال در سال ۲۰۱۴ فرناندو پرز به پروژه جوپیتر استارت زد. ابتدا قابلیت های نوت بوک و مستقل بودن از زبان IPython به جوپیتر منتقل شدند و پشتیبانی از زبان هایی مانند R و Julia به علاوه پایتون به آن اضافه شد. برعکس IPython نوت بوک های جوپیتر بر پایه وب هستند و در مرورگر به صورت تعامل اجرا میشوند.

جوپیتر با استفاده از کرنل ها اجرا میشود که در ابتدا فقط کرنل IPython قابل استفاده بود اما با گذشت زمان تعداد آنها افزایش یافت. امروزه جوپیتر محبوب ترین روش برنامه نویسی علمی است که در اکثر موسسات استفاده میشود. این کتابخانه مانند دیگر کتابخانه ها توسط لایسنس BSD توزیع میشود و کد منبع آن به صورت آزاد در گیت هاب در دسترس است.

۳-۳-۳- سرویس Collab

سرویس آنلاین کولاب یا کولابروتاری به برنامه نویس این اجازه را میدهد که در مرورگر کد پایتون بنویسد و اجرا کند. این سرویس کار های اضافه مانند پیکربندی را به صورت خودکار انجام می دهد و برای مدل های بزرگ یادگیری عمیق که نیازمند کارت گرافیک هستند ایده‌آل است. همچنین دارای قابلیت های بیشتر برای دانشجو ها، محققین و دانشمندان داده است. ظاهری بسیار نزدیک به جوپیتر دارد اما کد ها بجای اجرا شدن محلی در ابر گوگل اجرا میشوند.

این نوع نوت بوک ها قابلیت پشتیبانی از متن،LaTeX، و کد HTML را دارند. برای اضافه کردن مجموعه داده میتوان آن را به درایو گوگل آپلود کرد و سپس به نوت بوک لینک کرد. یکی دیگر از امکانات اصلی این سرویس قابلیت اشتراک کد با دیگران یا حتی کار کردن بر روی پروژه با چندین نفر به صورت همزمان است.

۳-۴- کتابخانه های ترسیم نمودار

۳-۴-۱- کتابخانه Matplotlib

کتابخانه اصلی نمایش تصویر و آمار[68] یا به اصلاح پلات کردن با پایتون Matplotlib است. به برنامه نویس رابط های شی گرا برای اضافه کردن شکل ها به برنامه های گرافیکی میدهد. البته به دلیل یکپارچگی با نامپای و جوپیتر و دیگر کتابخانه ها میتوان بدون ساختن اپلیکیشن گرافیکی اشکال را فقط در نوت بوک مشاهده کرد.

این کتابخانه اولین بار توسط جان هانتر در سال ۲۰۰۳ منتشر شد و به خاطر داشتن لایسنس BSD از همان ابتدا توسط برنامه نویسان توسعه یافت. این کتابخانه میتواند انواع مختلفی از شکل ها مانند پلات های ۲ بعدی و ۳ بعدی، هیستوگرام[69]، نمودار های خطی، نمودار های نقطه‌ای و قطبی را نشان بدهد.

۳-۴-۲- کتابخانه Seaborn

نیاز برای این کتابخانه از آنجا آمد که Matplotlib برای ترسیم نمودار های پیچیده کارآمد نبود و برنامه نویسان می خواستند ارتباط بین مجموعه داده های پیچیده را درک کنند. این کتابخانه دارای API های سطح بالا است که ساختن آنها در Matplotlib بسیار دشوار است و برای راحت تر کردن کار برنامه نویس در کتابخانه‌ای که بر پایه Matplotlib است آماده شده‌اند. این کتابخانه علاوه بر رسم نمودار میتواند تخمین بخش های ناموجود در داده را هم انجام دهد.

[68] Plotting

[69] Histogram

۳-۵- کتابخانه های یادگیری عمیق

۳-۵-۱- کتابخانه Tensorflow

اصلی ترین پکیج یادگیری است که توسط گوگل به صورت منبع باز و رایگان در سال ۲۰۱۹ منتشر شد. بسیار از پیشرفت های امروزه در حوزه هوش مصنوعی مدیون منتشر شدن این کتابخانه است. کار بر روی این کتابخانه در سال ۲۰۱۵ توسط تیم مغز گوگل (Google Brain) که برای استفاده در تحقیقات و برنامه نویسی داخلی در گوگل طراحی شده بود شروع شد.

این کتابخانه زبان های برنامه نویسی مختلفی مانند Java, Python, Swift, C++ , JavaScript را پشتیبانی میکند. هسته محاسباتی این کتابخانه فریمورک Keras است. کد نوشته شده در Tensorflow توسط کامپایلر داخلی تبدیل به گراف محاسباتی میشود و توسط پردازشی به نام AutoDiff گرادیان آن محاسبه شده و عمل Backproagation روی آن انجام می شود که باعث بهینه شدن مدل می شود.

کلیه محاسبات به صورت گراف های جریانی حالت دار نشان داده میشوند و به اطلاعات چند بعدی تنسور گفته می شود که به ndarray های نامپای شباهت زیادی دارند.

۳-۵-۲- کتابخانه Keras

کِراس کتابخانه منبع باز پایتون است که به عنوان رابط Tensorflow عمل میکند. در واقع مدل سازی در Tensorflow توسط کراس انجام می شود. در ابتدا از backend های مختلفی پشتیبانی می کرد اما با گذشت زمان در نسخه ۲.۴ تنها Tensorflow به عنوان backend پشتیبانی میشود. سازنده اصلی این کتابخانه برنامه نویس گوگل، فرانسوا کوله[70] است و اولین بار در سال ۲۰۱۵ منتشر شد.

[70] Francios Collet

۳-۵-۳- کتابخانه PyTorch

کتابخانه رقیب Tensorflow که توسط تیم تحقیقاتی هوش مصنوعی Facebook یا همان (FAIR) طراحی شده است و بر پایه کتابخانه محاسباتی علمی Torch است که در زبان Lua که دارای طیف گسترده‌ای از الگوریتم های یادگیری عمیق است. امروزه کتابخانه Torch منسوخ شده است و فقط از PyTorch به عنوان جایگزین بهتر آن استفاده می شود. این کتابخانه حالا زیر چتر بنیاد لینوکس[71] است و مانند کتابخانه های دیگر از لایسنس BSD استفاده می کند.

این کتابخانه برای کاربرد هایی مانند بینایی کامپیوتری و پردازش زبان طبیعی (NLP) ساخته شده است. این کتابخانه به صورت کل بر پایه زبان پایتون نوشته شده است اما از رابط های زبان C و C++ نیز پشتیبانی میکند. سیستم خودران ماشین های تسلا و تکنولوژی های اوب[72]ر چند نمونه از استفاده جهان واقعی این کتابخانه هستند. یکی از اصلی ترین تفاوت های این کتابخانه با Tensorflow این است که به جای گراف های جهتی حالت دار این کتابخانه از شبکه های بر پایه نوار برای نگهداری اطلاعات بهینه سازی استفاده میکند.

[71] Linux Foundation

[72] Uber

فصل ۴:

پروژه استخراج ویژگی تصاویر

۴-۱- مهندسی داده

داده اصلی این پروژه ۷۵۰ تصویر کشمش در قالب کلاس جدا به صورت تصاویر RGB با ابعاد ۱۹۲۰ در ۱۰۸۰ افقی بودند. برای منتقل کردن این تصاویر به شبکه عصبی نیاز بود که حاشیه و جاهای اضافی از عکس کراپ شوند. ساده ترین راه برای انجام این کار استفاده از نرم افزار ویرایش تصویر مانند Gimp یا Adobe Photoshop بود. اما به دلیل حجم بالای تصاویر و همچنین یک شکل بودن آن ها از روش تشخیص بخش های مختلف عکس به نام Segmentation استفاده شد.

با اینکه ممکن بود این کار با شبکه عصبی کانولوشن خاصی به نام Unet که از طراحی شبیه به شکل U استفاده میکند و در نیمه اول اطلاعات را به صورت تصاعدی[73] کاهش می دهد و سپس سعی میکند اطلاعات را با با کانولوشن برعکس و ارتباط مستقیم به لایه های اولیه احیا کند و تصویر خروجی از این معماری تصویر سگمنت شده است. اما به دلیل پرهزینه بودن کار و آزمایش و خطا با این شبکه و همچنین گرفتن نتایج راضی کننده از کتابخانه های یادگیری ماشینی به جای یادگیری عمیق تصمیم بر عدم استفاده از این نوع معماری شد.

این کار در نوت بوک جوپیتر و به صورت تعاملی انجام شد. ابتدا نیاز بود که ورودی ها را ببینیم و از کتابخانه Matplotlib و تابع imshow که برای نشان دادن داده عدد به صورت تصویر است استفاده شد. بعد از دیدن شکل اولیه تصاویر آنها در لیست ساده پایتون قرار می گیرند و توسط متد seg.slic از کتابخانه scikit-image که نیاز به پارامتر های ورودی برای تصویر و تعداد سگمنت ها است بخش بندی یا همان سگمنتیشن می شوند. چون سگمنت ها به صورت ۱ و ۲ برای نمایندگی سفید و مشکی نگاشت می شوند باید برای ساده تر کردن داده آنها را یک واحد کوچکتر کنیم و داده به صورت مجموعه‌ای از صفر و یک شود.

پس مشخص شدن بخش های اضافی داده به به صورت نقطه هایی در راس مربع تبدیل می شوند که این کار با متد coordinate ممکن می شود. حالا فقط چهار نقطه داریم که دو عدد برای نشان دادن طول و دو عدد برای نشان دادن عرض است. با اندیس کردن[74] پیشرفته که توسط نامپای انجام می گردد می توان به صورت عمودی و افقی بخش های اضافی تصویر را که برای کامپیوتر فقط تعدادی عدد در ماتریس هستند را حذف کرد.

[73] Exponential

[74] Indexing

کتابخانه های استفاده شده در بخش اول: نامپای، سیستم عامل, Scikit-image, Matplotlib

۴-۲- شبکه عصبی کانولوشن

بخش بعدی مهمترین بخش این پروژه است و نتیجه از این بخش حاصل خواهد شد. به دلیل استفاده از سرویس colab گوگل نیاز بود که اطلاعات آماده شده تصاویر به جای ذخیره شدن محلی در ابر گوگل[75] ذخیره می شد خوشبختانه نیازی به نگرانی از طرف قوانین کپی رایت داده یا دزدیده شدن نیست چون ابر گوگل دسترسی را فقط به حساب کاربری آپلود کننده میدهد. اولین کار در نوت بوک colab وارد کردن تصاویر از گوگل به نوت بوک بود. تصاویر برای این مدل به دو بخش training و validation تقسیم شده‌اند. ۶۰۰ مورد در بخش اول و بقیه ۱۵۰ تصویر در مجموعه validation قرار می گیرند.

بعد از انتقال تصاویر شبکه عصبی طراحی میشود که متشکل از سه بار کانولوشن و Max Pooling است. ابتدا در صورت نیاز باید مقیاس بزرگی تصاویر تغییر کند و از حالت ورودی به یک آرایه ۳ بعدی ۱۰۰ در ۱۰۰ در ۳ میرسد که یعنی عکس ۱۰۰ در ۱۰۰ به صورت RGB داریم.

قبل از هر کار لایه ها را نرمالیزه می کنیم یعنی از اعداد بین ۰ تا ۲۵۵ به اعداد اعشاری بین ۰ تا ۱ تبدیل میکنیم و این کار باعث راحت تر شدن کار الگوریتم در حین یادگیری میشود.

بعد از تغییر مقیاس اولین لایه کانولوشن تصاویر ۱۰۰ در ۱۰۰ را به ۸ فیلتر و مربع های ۳ در ۳ با تابع فعال ساز ReLU میدهد که ابعاد از ۱۰۰، ۱۰۰، ۳ به ۹۸، ۹۸، ۸ تغییر میکنند. بعد از عملیات کانولوشن باید روی داده Max Pooling انجام دهیم و از مربع های ۲ در ۲ برای این کار استفاده میکنیم. این کار باعث انتخاب بزرگترین مقدار در هر مربع ۴ پیکسل میشود. روش دیگر انتخاب میانگین به جای بزرگترین است که Average Pooling نام دارد. بعد از Max Pooling ابعاد تصویر نصف شده و ۴۹، ۴۹، ۸ میشود. در اولین مرحله فقط یکبار کانولوشن انجام میدهیم و سپس از Max Pooling استفاده میکنیم ولی با توجه به معماری LeNet 5 در لایه بعدی از دو

[75] Google Drive

کانولوشن پشت سر هم بدون Max Pooling استفاده می کنیم این کار استخراج ویژگی به حد کافی را از تعداد ورودی های کم را تضمین میکند.

چون تصاویر خارج شده از این پروسه به الگوریتم SVM داده خواهند شد در پایان از لحاظ ریاضی نرمالیزه می کنیم و این کار با استفاده از لایه BatchNormalization از کراس انجام می شود. میتوان در این قدم تصویر را صاف کرد (همان لایه Flatten) اما به دلیل خوانایی کد این کار از طریق numpy انجام میشود. بعد از تبدیل ماتریس های سه بعدی با شکل ۲۲ در ۲۲ در ۳۲ به بردار های صاف ۱۵۴۸۸ تایی این تصاویر آماده کلاس بندی هستند. معیار خطا (Metric Function) در این مدل همان آنتروپی دودویی است که در ادامه فرمول آن نشان داده شده است و بهینه ساز همان SGD است که قبل تر از آن بحث شد.

فرمول محاسبه کراس آنتروپی:

$$Loss = -\frac{1}{m}\sum_{i=1}^{m} y_i \cdot log\,\widehat{y_i} + (1 - y_i) \cdot log(1 - \widehat{y_i})$$

در این فرمول وجود y اصلی یا همان مقدار واقعی باعث کار کردن فقط یک بخش از تابع در هر لحظه میشود و اگر مقدار پیشبینی شده مطابق با مقدار واقعی باشد وجود تابع لگاریتم باعث رشد نکردن Loss میشود. این تابع بخاطر مشتق پذیر بودن در الگوریتم های مختلف کلاس بندی استفاده میشود.

بجای استفاده از CNN دستی که ساخته‌ایم ما سعی میکنیم از یک روش استخراج ویژگی دیگر که با استفاده از شبکه VGG16 است نیز استفاده کنیم. این کار هزینه یادگیری زیادی ندارد چرا که VGG بر روی یک میلیون تصویر از هزار کلاس یادگیری انجام داده و ما فقط از آن برای استخراج ویژگی و مقایسه ویژگی ها با CNN استفاده میکنیم.

۴-۳- کلاس بندی با SVM

قبل از دادن داده به الگوریتم SVM از مقیاس استاندارد استفاده کرده‌ایم اما این کار نیاز نیست چون داده در دست بعد از CNN استاندارد شده‌اند هدف از این کار تطبیق با استاندارد مخصوص در Sci Kit Learn است.

همچنین لازم است با استفاده از متد reshape در نامپای شکل و ابعاد داده تغییر دهیم و داده را به اصلاح صاف کنیم. پس از صاف کردن تمامی ویژگی ها را که در خروجی CNN به همدیگر چسبانده بودیم به متد train_test_split می دهیم که داده ها را به بخش های train و test جدا کند. چون نمیخواهیم الگوریتم همه داده را ببیند و بخشی از آن را جدا میکنیم و برای تست کردن نگه می داریم. داده بعد از جدا شدن به دو بخش ۶۰۰ تایی و ۱۵۰ تایی تقسیم میشود. علاوه بر این اگر بخواهیم داده به سه بخش train, test و validation جدا شود میتوان آن کار را اینجا به سادگی انجام داد. البته برای این پروژه لازم نیست. چون بر روی داده train یادگیری انجام نخواهد شد.

بعد از استخراج ویژگی و آماده کردن داده، از دو نوع SVM یا همان ماشین پشتیبانی برداری استفاده میکنیم. چون کلاس های ما به صورت دودویی هستند میتوان از نوع ساده SVM یا همان Linear SVC (کلاس بند خطی) استفاده کرد. این نوع یادگیری به دلیلی نداشتن هایپر پارامتر[76] بسیار محبوب است. البته این کار به برنامه نویسی اجازه تغییر نوع یادگیری یا استفاده از ویژگی های درجه بالاتر را نمیدهد و باید بر روی داده مستقیم کار کند. واضح است که نتایج این نوع کلاس بندی نسبت به روش هایی که از کرنل استفاده میکنند یا تبدیل ویژگی دارند ضعیف تر است.

نوع پیشرفته SVM یا همان SVM با کرنل را نیز در این پروژه برای گرفتن نتیجه بهتر استفاده میکنیم. کرنل مورد استفاده برای این الگوریتم کرنل RBF است. این کرنل همان کرنل گاوسی است که در بخش ۲-۳ به آن اشاره شد. فقط کاهش ابعاد انجام می دهد و سپس ویژگی درجه بالا (مربعی) را می افزاید. برای یافتن و مطمئن تر شدن از هایپر پارامتر ها به دادن مقادیر آزمایشی بسنده نشده و از Grid Search که الگوریتم را با چندین نتیجه ضرب دکارتی هایپر پارامتر ها است آموزش می دهد استفاده کرده‌ایم. از اینجا مقادیر گاما و C که هر دو برای Overfit نشدن داده هستند را تست میکنیم.

کتابخانه های استفاده شده در بخش دوم و سوم: Tensorflow, SciKit Learn, Keras, Matplotlib

[76] Hyperparameter

فصل۵:

نتایج

بعد از استخراج ویژگی با استفاده از دو روش شبکه های عصبی کانولوشن و شبکه از پیش طراحی شده VGG باید ویژگی ها برای کلاسبندی به SVM ها داده شوند که این کار برای ۱۵ کلاس متفاوت انجام میشود.

از معیار های Accuracy, Precision, Recall و امتیاز F1 برای اندازه گیری معیار کارایی روش های مختلف کلاس بندی و استخراج ویژگی استفاده شده است که در ادامه تعریف ریاضی آن ها ارائه می شود. محدوده همه این معیار ها بین ۰ تا ۱۰۰ می باشد.

Accuracy:

$$\frac{TP + TN}{TP + FP + FN + TN}$$

Recall:

$$\frac{TP}{TP + FN}$$

Precision:

$$\frac{TP}{TP + FP}$$

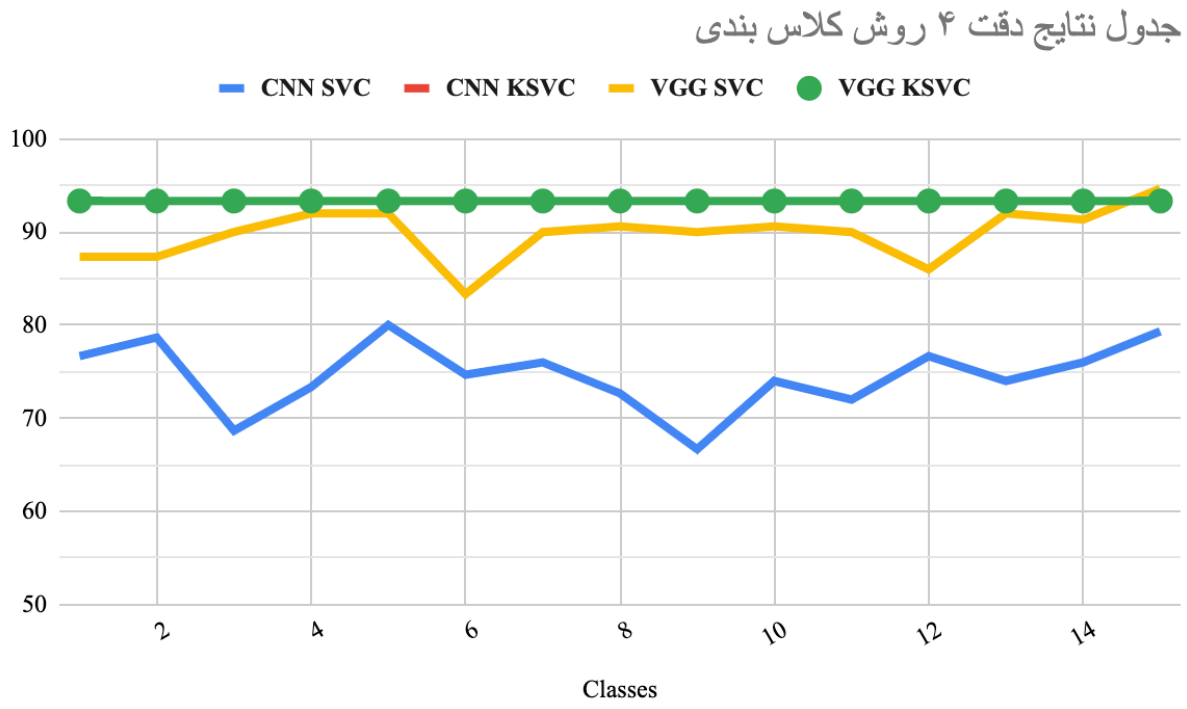
F1 Score:

$$2 * \frac{Precision * Recall}{Precision + Recall}$$

که در این معادلات ***TP, TN, FP, FN*** به ترتیب منفی کاذب، مثبت کاذب، منفی صادق و مثبت صادق هستند.

اولین و مهم ترین معیار در اکثر پروژه های کلاس بندی دقت یا همان Accuracy است که در این پروژه نیز از اهمیت زیادی برخوردار بود. اگر به نمودار ۴ روش مختلف استخراج ویژگی نگاه کنیم می بینیم که ماشین های برداری پشتیبانی که با کرنل هستند همواره دقت قابل قبول ۹۳.۳۳ میدهد که با تغییر دادن روش استخراج ویژگی تفاوتی نمی کند. بعد از نگاه کردن دقیق به ماتریس سردرگمی خواهیم دید که این الگوریتم کل جواب را به خاطر یک طرفه بودن ویژگی ها به صورت ۱ میدهد و این باعث می شود بدون تجزیه داده به جواب قابل قبول برسد.

اما برای ماشین های برداری پشتیبان بدون کرنل یا همان خطی این گونه نیست. واضح است که خروجی آنها با استخراج ویژگی بیشتر که با مدل VGG16 انجام می شود بهتر میشود و برای هر ویژگی به صورت جداگانه سعی میکنند ارتباطش را جواب یاد بگیرند.



شکل ۱ - بازدهی روش های مختلف

برای واضح تر شدن روش کار مدل ها باید به امتیاز های معیار های دیگر نیز توجه کنیم. البته این کار برای روش های کلاس بندی با کرنل نیاز نیست چون امتیاز کل معیار آن ها دقیقا یکسان می باشد.

Class	CNN → SVC				VGG16 → SVC			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
1	76.666	100	75	85.71	87.33	99.18	87.14	92.77
2	78.666	100	77.14	87.1	87.33	99.18	87.14	92.77
3	68.666	96	69.28	80.5	90	99.21	90	94.38
4	73.33	99	72.14	83.47	92	99.23	92.14	95.55
5	80	98.24	80	88.18	92	100	91.48	95.52
6	74.666	99.03	73.57	84.42	83.33	24.13	70	35.9
7	76	100	74.28	85.24	90	99.21	90	94.38
8	72.67	100	70.71	82.84	90.6	97.72	92.14	94.85
9	66.666	100	64.28	78.26	90	100	89.28	94.33
10	74	100	64.28	78.26	90.6	99.21	90.71	94.77
11	72	99	70.71	82.5	90	97.7	91.42	94.46
12	76.666	100	75	85.71	86	100	85	91.9
13	74	98	73.57	84.1	92	96.37	95	95.68
14	76	99.1	75	85.36	91.33	99.22	91.42	95.16
15	79.33	100	77.85	87.55	94.666	100	94.28	97.05

جدول ۱ - خروجی برای الگوریتم های بدون کرنل

جدول درستی برای الگوریتم های ماشین برداری پشتیبانی با کرنل

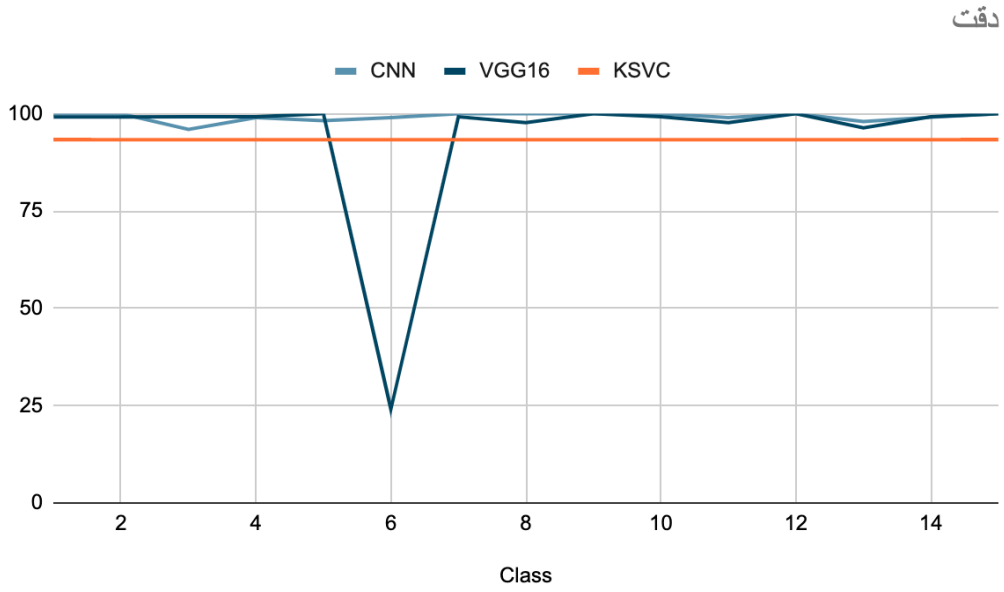
CNN → KSVC, VGG16 → KSVC			
Accuracy	Precision	Recall	F1
93.33	93.33	100	96.65

جدول ۲ - خروجی الگوریتم های با کرنل (به دلیل ثابت بودن نیازی برای ذکر اطلاعات کل ۱۵ کلاس نیست)

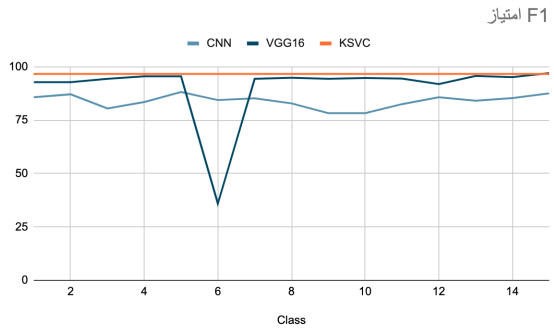
پس از مقایسه دقت و بازیافت روش های مختلف واضح است که الگوریتمی که از کرنل استفاده میکند بازیافت بالاتری نسبت به بقیه الگوریتم ها دارد در حالی که به صورت عمومی از لحاظ دقت ضعیف تر از بقیه الگوریتم ها است و الگوریتمی که ضعیف ترین نتیجه را میدهد (یعنی استفاده از شبکه کانولوشن دستی و الگوریتم کلاس بندی بدون کرنل) بیشترین دقت را دارد. اما داشتن کمترین بازیافت باعث عقب افتادن از لحاظ امتیاز F1 نسبت به بقیه است. پس میتوان گفت که این الگوریتم تمایل دارد که مثبت های کاذب کمتری داشته باشد اما تمامی مثبت های واقعی را بر نمی گرداند.

برعکس در الگوریتم های مبتنی بر کرنل تمامی مثبت های واقعی برگردانده می شود و علاوه بر این منفی های صادق را به عنوان مثبت کاذب می شناسد که این باعث می شود همواره جواب این الگوریتم ۱ باشد.

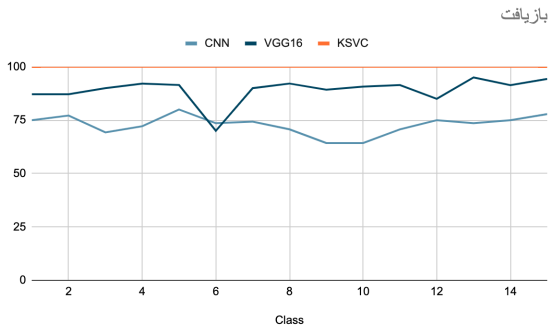
در نهایت الگوریتم بدون کرنل با VGG16 میتواند با الگوریتم های با کرنل رقابت کند که در واقع این نشانگر این است هر چه به تعداد لایه های استخراج ویژگی اضافه می کنیم به جواب بهتری می رسیم و در واضع الگوریتم نهایی از اهمیت کمتری نسبت به الگوریتم استخراج ویژگی خوب بهره مند است.



شکل ۲ - دقت برای الگوریتم های مختلف کلاس بندی



شکل ۴ - امتیاز F1 برای الگوریتم های مختلف کلاس



شکل ۳ - بازیافت برای الگوریتم های مختلف کلاس بندی

فصل ۶:

جمع بندی و روش های توسعه بیشتر

۶-۱- جمع بندی

هدف از این پروژه انجام پروژه جهان واقعی بود که با داده جهان واقعی و خروجی کاربردی باشد. بیشتر کتابخانه های استفاده شده در این پروژه به برنامه نویس آزادی عمل را می دهند و صرفا کمک به سرعت بخشی پروسه برنامه نویسی هستند بجای اینکه راه حل آماده را بدهند.

بخش اول این پروژه بیشتر عمل هایی بود که یک دانشمند داده بر روی مجموعه داده ورودی انجام می دهد که بعد ها خودش یا تیم دیگری بتواند با آن داده کار کند که برای این کار علاوه بر درستی داده تحویل داده شده، نیاز است که داده استاندارد و یک شکل باشند.

در بخش دوم و سوم که هسته این پروژه است بجای استفاده از تکنیک های سالخورده استخراج ویژگی مانند PCA و اتوانکودر ها سعی بر استفاده از روش های نوین مانند کانولوشن شد و مدل آماده VGG16شد که نتیجه بهتری را می دهد با اینکه نسبتا به قدرت پردازش زیادی نیازمند است.

الگوریتم های کلاس بندی در بخش سوم تفاوت چندانی ندارد حتی با استفاده از جستجو محیطی[77] نیز ماشین های برداری پشتیبانی با کرنل نتیجه خوبی می دهند که البته مشاهده شد با استفاده از الگوریتم استخراج ویژگی خوب تفاوت چندانی بین ماشین های با کرنل و خطی وجود ندارد.

[77] Grid Search

۶-۲- روش های توسعه

1. در ادامه برای بهتر کردن نتایج می توان از مدل های دیگر و عمیق تری مانند VGG19 که دارای ۱۹ لایه به جای ۱۶ لایه است یا Resnet 101 با ۱۰۱ لایه و استفاده کردن از لایه های پرنده بجای لایه های کانولوشن استفاده کرد. این معماری ها به دلیل نیاز بیش از حد به قدرت پردازشی قابل استفاده برای این پروژه نبودند.

2. ایده دیگر برای این توسعه این پروژه می تواند استفاده از مجموعه داده بزرگتر باشد که به نیازمند تهیه داده بیشتر در جهان واقعی است. این کار هزینه‌ی و وقت زیادی نیازمند است و به خاطر قابل قبول بودن بهره وری این مجموعه داده نیاز نبود.

3. دادن عکس ها با تعداد بیشتر پیکسل به دلیل بیشتر کردن اطلاعات قابل یادگیری نیز ایده قابل اجرایی است علی الخصوص وقتی تفاوت بین ۷ لایه و ۱۶ لایه را در شبکه های استخراج ویژگی مشاهده می کنیم. این کار نیازمند حافظه بیشتر در رم یا حتی واحد پردازنده گرافیکی که از CUDA پشتیبانی می کند است.

۷- منابع

1. Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." In Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18, pp. 234-241. Springer International Publishing, 2015.

2. Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

3. Harangi, Balazs. "Skin lesion classification with ensembles of deep convolutional neural networks." Journal of biomedical informatics 86 (2018): 25-32.

4. Keerthana, Duggani, Vipin Venugopal, Malaya Kumar Nath, and Madhusudhan Mishra. "Hybrid convolutional neural networks with SVM classifier for classification of skin cancer." Biomedical Engineering Advances 5 (2023): 100069.

5. Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

6. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. Nature 521, 436–444 (2015). https://doi.org/10.1038/nature14539

7. Géron, Aurélien. 2019. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. N.p.: O'Reilly.

8. Müller, Andreas C., and Sarah Guido. 2016. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. N.p.: O'Reilly Media, Incorporated

9. McKinney, Wes. 2022. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and Jupyter*. N.p.: O'Reilly Media, Incorporated.

10. Chollet, Francois. 2021. Deep Learning with Python, Second Edition. N.p.: Manning.

11. Google. n.d. “Tensorflow.” TensorFlow. Accessed February 18, 2023. https://www.tensorflow.org.

12. Sci Kit lean. n.d. “SK-Learn.” scikit-learn: machine learning in Python — scikit-learn 1.2.1 documentation. Accessed February 18, 2023. https://scikit-learn.org/stable/.

13. Google. n.d. “Google Colab.” Welcome To Colaboratory - Colaboratory. Accessed February 18, 2023. http://colab.research.google.com.

14. G, Rohini, and Shweta Gupta. 2021. “Everything you need to know about VGG16 | by Great Learning.” Medium. https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918.

15. Sobek, Tomas. 2019. “Auto-Encoder: What Is It? And What Is It Used For? (Part 1).” Towards Data Science. https://towardsdatascience.com/auto-encoder-what-is-it-and-what-is-it-used-for-part-1-3e5c6f017726.

16. “A Layman’s Guide to Deep Neural Networks | by Jojo John Moolayil.” 2019. Towards Data Science. https://towardsdatascience.com/a-laymans-guide-to-deep-neural-networks-ddcea24847fb.

17. Jupyter. n.d. “Jupyter Project.” Project Jupyter | Home. Accessed February 18, 2023. https://jupyter.org.

18. Gibiansky, Andrew. 2014. “Convolutional Neural Networks.” Andrew Gibiansky. https://andrew.gibiansky.com/blog/machine-learning/convolutional-neural-networks/.

Image Feature Extraction and Classification Using Deep Convolutional Neural Networks

A project report

Presented to:

Bonab University

Advisor:

Mehdi HosseinZadeh Aghdam

Conducted By:

AmeerMohammad Ramzani