دانشگاه یزد

پردیس علوم

دانشکده ریاضی

پایان نامه

برای دریافت درجه کارشناسی ارشد

علوم کامپیوتر (محاسبات علمی)

عنوان:

شناسایی و ردیابی اشیاء متحرک با استفاده از الگوریتم‌های یادگیری چند هسته‌ای

استاد راهنما:

پروفسور محمدرضا هوشمنداصل

استاد مشاور:

دکتر الهام عباسی هرفته

پژوهش و نگارش:

امیر محمدی

بهمن ۱۳۹۸


بسم الله الرحمن الرحيم

تقدیم به:

پدر و مادر عزیز و خانواده ارزشمندم به‌خاطر تمامی زحمات بی‌دریغی که در دوران مختلف زندگی‌ام انجام داده‌اند و با مهربانی چگونه زیستن را به من آموخته‌اند. آن‌هایی که در سختی‌ها و دشواری‌های زندگی و همچنین در مسیر کسب علم و دانش همواره یاوری دلسوز و فداکار و پشتیبانی محکم و مطمئن برایم بوده‌اند.

سپاس‌گزاری

سپاس بی کران پروردگار یکتا را که هستی‌مان بخشید و به طریق علم و دانش رهنمون‌مان شد و به هم‌نشینی رهروان علم و دانش مفتخرمان نمود و خوشه‌چینی از علم و معرفت را روزیمان ساخت. پروردگارا از تو برای تمامی لطف‌های بی‌کرانت در لحظه لحظه از زندگی‌ام ممنون و سپاس‌گزارم.

بدون شک جایگاه و منزلت معلم، اجّل از آن است که در مقام قدردانی از زحمات بی شائبه او، با زبان قاصر و دست ناتوان، چیزی بنگاریم. اما از آنجایی که تجلیل از معلم، سپاس از انسانی است که هدف و غایت آفرینش را تامین می کند و سلامت امانت‌هایی را که به دستش سپرده‌اند، تضمین؛ بر حسب وظیفه و از باب مصداق «من لم یشکر المخلوق لم یشکر الخالق» بر خود واجب می‌دانم از تمامی اساتید بزرگوار که در طول سالیان گذشته مرا در تحصیل علم و معرفت و فضایل اخلاقي ياري نموده‌اند تقدیر و تشکر نمایم.

از استاد فرهیخته و گرامی‌ام جناب آقای دکتر محمدرضا هوشمنداصل که همواره با صبر و بردباری بسیار، چراغ راهم بودند و با راهنمایی‌های دلسوزانه و روشنی بخششان مرا در نگارش این پایان‌نامه همراهی و راهنمایی نمودند بسیار ممنون و سپاس‌گزارم.

همچنین از استاد فرزانه سرکار خانم دکتر الهام عباسی به عنوان استاد مشاور و به‌خاطر یاری‌ها و راهنمایی‌های بی‌چشم‌داشت ایشان که بسیاری از سختی‌ها را برایم آسان نمودند تشکر می‌کنم.

از استاد گرامی آقای دکتر جمال زارع‌پور و آقای دکتر محسن علمبردار که قبول زحمت فرمودند و داوری این پایان‌نامه را برعهده گرفتند نیز بسیار سپاس‌گزارم.

در آخر نیز از پدر و مادر عزیز، دلسوز و مهربانم که آرامش روحی و آسایش فکری فراهم نمودند تا با حمایت‌های همه جانبه در محیطی مطلوب، مراتب تحصیلی و نیز پایان‌نامه درسی را به نحو احسن به اتمام برسانم سپاس‌گزاری می‌نمایم.

چکیده

در این پایان نامه ابتدا مقدمه‌ای بر حوزه بینایی کامپیوتر و کاربردهای آن به عنوان یک زیر شاخه از یادگیری ماشین مطرح شده است. سپس دو نمونه از مسائل مهم و کاربردی در این زمینه تحت عنوان شناسایی و ردیابی اشیاء عنوان و تاریخچه‌ای از روش‌ها و الگوریتم‌های ارائه شده تاکنون برای این مسائل بیان شده است. در ادامه روش‌های مبتنی بر هسته برای حل مسائل یادگیری ماشین و به خصوص دسته‌بندی‌های غیرخطی مطرح شده و سپس حالت تعمیم یافته‌ای از این روش‌ها تحت عنوان یادگیری چند هسته‌ای معرفی می‌شود. در ادامه شبکه‌های تلفیقی، انواع لایه‌ها، معماری‌ها و همچنین موارد استفاده از آن‌ها برای مسائل مختلف تشریح شده است. سپس چند نمونه از الگوریتم‌های مبتنی بر شبکه‌های تلفیقی برای مسائل شناسایی و ردیابی اشیاء معرفی و با سایر الگوریتم‌ها مقایسه شده‌اند. در نهایت با استفاده از الگوریتم‌های مبتنی بر فیلتر هم‌بستگی هسته‌گذاری شده و استفاده از خصوصیات منحصر به فرد این روش‌ها و همچنین بهره‌گیری از یک مدل حافظه‌ای چند کاناله، یک الگوریتم پیشنهادی برای ردیابی اشیاء ارائه شده است که علاوه بر پیچیدگی محاسباتی بسیار کم، از پایداری و دقت مناسب در شرایطی مانند انسداد، تغییرات حالت و شلوغی پس‌زمینه برخوردار بوده و می‌توان از آن در ردیابی‌های بلادرنگ استفاده نمود.

فهرست مطالب

فهرست تصاویر

فصل ۱

بیان مفاهیم پایه‌ای و مروری بر پژوهش‌های انجام شده

۱.۱ مقدمه

ما انسان‌ها از چشمان و ذهن خود برای دیدن و درک کردن آنچه در پیرامون خود می‌بینیم، استفاده می‌کنیم. اما آیا کامپیوتر ها نیز قادر به انجام کاری مشابه با این هستند؟ پر واضح است که با افزایش روزانه تصاویر و فایل های ویدئویی ضبط شده، پردازش این حجم از داده‌ها توسط منابع انسانی امکان‌پذیر نیست. از این جهت نیاز به سیستم‌های کامپیوتری که مانند انسان بتوانند این تصاویر را مشاهده کرده و اطلاعات موجود در آن‌ها را تحلیل و بررسی کنند به شدت احساس می‌شود.

بینایی کامپیوتر یکی از زیر شاخه‌های زمینه‌های هوش مصنوعی و یادگیری ماشین است که هدف آن تجهیز کامپیوترها به توانایی دیدن و درک مفاهیم موجود در تصاویر دیجیتال است. مسائل موجود در این زمینه از دید افراد عادی بسیار ساده به نظر می‌رسند؛ زیرا توسط انسان‌ها به سادگی و بدون هیچ‌گونه پیچیدگی قابل حل هستند. این در حالی است که اگر از کامپیوترها برای این کار استفاده شود، موضوع کمی متفاوت خواهد بود تا جایی که برخی از این مسائل همچنان به طور کامل حل نشده‌اند. این مسائل کاربردهای بسیار وسیعی در زمینه‌های مختلف دارد و می‌تواند در تمامی حوزه‌ها استفاده‌های به سزایی داشته باشد.

شناسایی و ردیابی اشیاء در تصاویر و فایل‌های ویدئویی یکی از مهم‌ترین، پرکاربردترین و پر چالش‌ترین مسائل موجود در بینایی کامپیوتر است که امروزه با توجه به در دسترس بودن دوربین‌های مختلف و نسبتا کم هزینه اهمیت و موارد استفاده از آن‌ها به شدت در حال افزایش است.

به طور کلی مسئله ردیابی اشیاء متحرک را می‌توان به‌عنوان مسئله موقعیت‌یابی یک یا چند شئ در طی یک

توالی از تصاویر یا قاب‌های یک فایل ویدئویی در نظر گرفت که توسط یک یا چند دوربین ضبط شده‌اند. تا به حال روش‌ها و سازوکارهای مختلفی برای ردیابی اشیاء پیشنهاد و ارائه شده است به‌طوری که یکی از مهم‌ترین تفاوت‌های آن‌ها با یکدیگر معمولا بر اساس مواردی مانند نحوه نمایش اشیاء[1] و همچنین نحوه انتخاب و استخراج ویژگی از آن‌ها است. شایان ذکر است که منظور از نمایش اشیاء نحوه مدل کردن قالب و ظاهر آن‌ها است که بر اساس خصوصیات اشیاء مورد نظر، محیطی که ردیابی در آن انجام می‌شود و همچنین بسته به کاربرد مد نظر می‌توان روش مناسب را انتخاب و استفاده نمود. در ادامه هر یک از موارد مطرح شده و انواع حالت‌های موجود برای این منظور بیان و شرح داده خواهد شد.

۲.۱ روش‌های نمایش قالب اشیاء

اشیائی که قرار است شناسایی و ردیابی شوند ممکن است از دسته‌های مختلفی مانند انسان، خودرو، هواپیما و موجودات زنده باشند که هر کدام از آن‌ها می‌توانند با حالت‌های مختلفی نمایش داده شوند.

از جمله حالاتی که می‌توان اشیاء مورد نظر را با آن‌ها نمایش داد عبارت‌اند از:

۱. نقاط:

اشیاء توسط یک نقطه مرکزی [۱۴۸] و یا مجموعه‌ای از نقاط [۱۲۷] نمایش داده می‌شوند. به طور کلی این روش برای حالت‌هایی مناسب است که شئ درون تصویر، میزان کمی از محیط را اشغال کرده است یا به عبارت دیگر ابعاد کوچکی دارد.

۲. اشکال هندسی اولیه:

شمایل شئ توسط اشکال هندسی مانند مستطیل، بیضی نمایش داده می‌شود [۲۷]. این مدل نمایش برای ردیابی اجسام صلب مناسب است هرچند که برای اجسام غیر صلب هم می‌توان از آن‌ها استفاده کرد.

۳. سیاه نمایی و کانتور[2]:

نمایش کانتور، مرز اشیاء با محیط پس زمینه را تشکیل می‌دهد و ناحیه درون کانتور، ناحیه سیاه نمای شئ نامیده می‌شود [۱۵۷]. این مدل نمایش اشیاء برای اجسام غیر صلب و یا اجسامی که شمایل پیچیده‌تری دارند مناسب است.

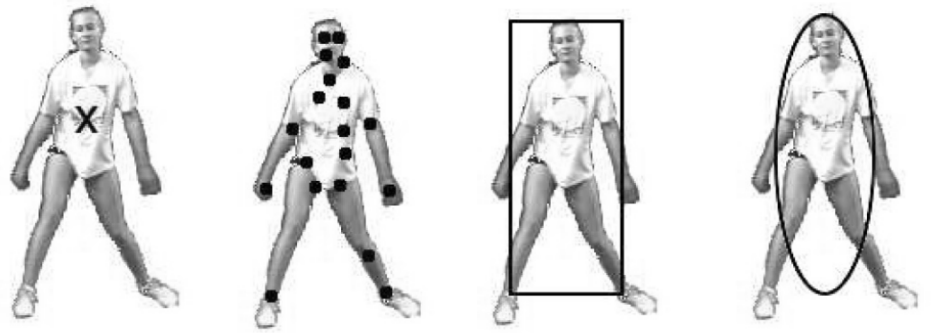
[1] Object Representation
[2] Contour

۴. شکل بند بند[1]:

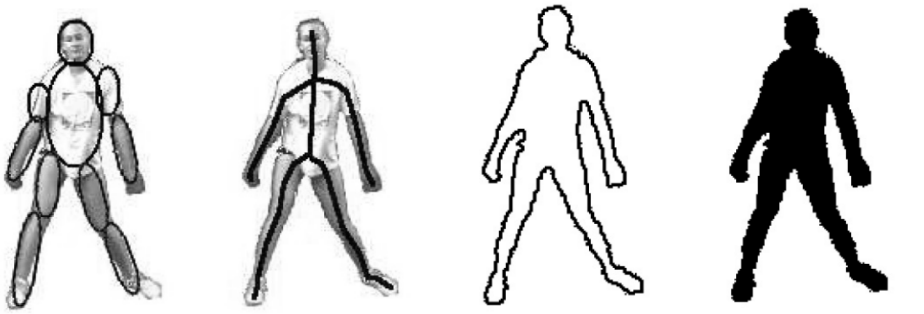
منظور از اشیاء بند بند اشیائی است که از قسمت‌های مختلفی تشکیل و ترکیب این قسمت‌ها با یکدیگر شئ مورد نظر را تشکیل می‌دهد. به‌عنوان مثال انسان یک شئ بند بند است که با به هم پیوستن اجزای مختلفی مانند دست‌ها، پاها سر و بقیه قسمت‌های بدن تشکیل شده است. ارتباط بین این اجزا توسط مدل‌های حرکتی مانند زوایای مشترک برقرار می‌شود و این اجزا را می‌توان با استفاده از بیضی‌ها نیز مدل کرد.

۵. مدل‌های اسکلتی:

اشیاء اسکلتی شکل را می‌توان با اعمال یک محور میانی تبدیل[2] بر روی مدل سیاه نمای شئ مدل نمود [۸]. این حالت از نمایش شئ می‌تواند برای اجسام بند بند و همچنین اجسام صلب مورد استفاده قرار گیرد.



(د) نمایش بیضوی (ج) نمایش مستطیلی (ب) نمایش چند نقطه‌ای (آ) نمایش نقطه‌ای



(ح) نمایش سیاه‌نمایی (ز) نمایش کانتوری (و) نمایش اسکلتی (ه) نمایش بندبند

شکل ۱.۱: روش‌های مختلف برای نمایش قالب اشیاء مورد نظر به منظور اعمال الگوریتم‌های ردیابی [۱۵۶].

مشابه با نمایش قالب اشیاء، روش‌های گوناگونی نیز برای نمایش ظاهر اشیاء وجود دارد. باید توجه داشت که

[1] Articulated shape
[2] Medial Axis Transform

رابطه زیادی بین نمایش ظاهر شئ و الگوریتم‌های ردیابی وجود دارد و همانطور که ذکر شد نمایش اشیاء معمولا با توجه به کاربرد مورد نظر در ردیابی مشخص می‌شود.

۳.۱ روش‌های نمایش ظاهر اشیاء

برخی از روش‌های رایج برای نمایش ظاهر[۱] اشیاء عبارت‌اند از:

۱. تراکم‌های احتمالاتی[۲]:

منظور از تراکم‌های احتمالاتی معمولا رنگ و بافت اشیاء مورد نظر است و می‌تواند با استفاده از ناحیه مشخص شده برای نمایش قالب شئ محاسبه گردد. تخمین این تراکم‌ها می‌تواند به صورت روش‌های پارامتری مانند گوسین [۱۶۴] و ترکیب گوسین‌ها [۱۰۳] و یا روش‌های ناپارامتری مانند هیستوگرام [۲۷] و پنجره‌های parzen [۳۴] انجام گیرد.

۲. الگوها[۳]:

الگوها با استفاده از اشکال هندسی ساده یا ناحیه سیاه نما تشکیل می‌شوند [۳۷]. مزیت استفاده از الگوها این است که هر دو دسته از اطلاعات ظاهری و مکانی را با خود به دنبال دارند اما در این روش ظاهر شئ تنها از یک نما و زاویه دید کد می‌شود. به همین دلیل استفاده از الگوها تنها برای ردیابی اشیائی مناسب هستند که تغییرات نمایشی آن‌ها در طول حرکت قابل توجه نیست.

۳. مدل های ظاهری چند نمایه[۴]:

این مدل‌ها نماهای متفاوتی از اشیاء را کد می‌کنند. یک نگرش رایج برای نمایش نماهای مختلف از اشیاء، ساخت یک زیر فضا از نماهای موجود است. برای این منظور می‌توان از روش‌هایی مانند تحلیل مولفه‌های اصلی[۵] و تحلیل مولفه‌های مستقل[۶] استفاده کرد که برای نمایش قالب و ظاهر اشیاء استفاده می‌شوند [۹۸].

[1] Appearance
[2] Probability Densities
[3] Templates
[4] Multiview Appearance
[5] Principal Component Analysis (PCA)
[6] Independent Component Analysis (ICA)

۴.۱ شناسایی اشیاء

بسیاری از رویکردهای اخیر در مورد ردیابی اشیاء از الگویی تحت عنوان ردیابی همراه با شناسایی[1] پیروی می‌کنند که دلیل محبوبیت این روش‌ها، به خاطر پیشرفت‌های مهم و چشم‌گیری بوده که در سال‌های اخیر در زمینه شناسایی اشیاء حاصل شده است. در روش‌های مبتنی بر این الگو، در گام ورودی ابتدا فرضیه‌های شناسایی[2] در تمامی یا برخی از قاب‌های ویدئو مشخص می‌شوند. سپس شناسایی‌های انجام شده که مربوط به یک شئ یکسان است، با یکدیگر پیوند داده می‌شوند تا مسیر حرکت شئ در ویدئو را بازسازی کنند و پس از آن گام بعدی که اتصال داده‌ها[3] نام دارد اجرا می‌شود. هر روش ردیابی نیازمند یک فرآیند شناسایی است که در اولین لحظه ورود شئ به تصویر و همچنین در تمام یا برخی از قاب‌های دیگر انجام شود.

اگر شئ شناسایی شود و ناحیه مورد نظر آن مشخص باشد، سپس این وظیفه ردیاب است تا شئ شناسایی شده را در بین قاب‌های متوالی ردیابی کرده و مسیر حرکت آن را مشخص کند.

به طور کلی روش‌های موجود در زمینه شناسایی اشیاء را می‌توان به دو دسته تقسیم‌بندی کرد. دسته اول روش‌هایی هستند که تا قبل از سال ۲۰۱۲ معرفی شدند و به روش‌های کلاسیک شناخته می‌شوند. دسته دوم نیز روش‌هایی هستند که پس از سال ۲۰۱۲ معرفی شدند و می‌توانند به‌عنوان روش‌های نوین در این زمینه در نظر گرفته شوند. در ادامه به توضیح مختصری در مورد روش‌های موجود در هر یک از دسته‌ها و پیشینه آن‌ها پرداخته خواهد شد.

۵.۱ روش‌های کلاسیک در شناسایی اشیاء

در سال ۲۰۰۱ ویولا و جونز یک الگوریتم شناسایی ارائه کردند که می‌توان آن را به‌عنوان یکی از موثرترین الگوریتم‌هایی شناخت که توجه دانشمندان علوم و بینایی کامپیوتر را به خود جلب کرد [۱۴۹]. پس از آن یک روش برای توصیف ویژگی‌های اشیاء توسط دالال و همکارانش تحت عنوان هیستوگرام گرادیان[4] معرفی گردید که برای شناسایی انسان‌ها از آن استفاده شد و آن را می‌توان به‌عنوان یکی از اولین الگوریتم‌های معرفی شده برای شناسایی اشیاء بر اساس ویژگی‌های آن‌ها شناخت [۳۱].

[1] Tracking-by-detection
[2] Detection hypothesis
[3] Data Association
[4] Histogram of Gradient

به طور کلی مسئله شناسایی اشیاء را می‌توان به‌عنوان یک مسئله دسته‌بندی در نظر گرفت به‌طوری که قصد دارد تا اشیاء مورد نظر را از محیط پس زمینه خود تفکیک کند. در روش‌های کلاسیک برای شناسایی اشیاء فرآیند کلی به این صورت است که ابتدا یک تصویر به‌عنوان ورودی انتخاب می‌شود. پس از انجام پیش پردازش‌های لازم توسط روش‌های توصیف ویژگی، ویژگی‌های اشیاء استخراج شده و برای تصمیم‌گیری به یک الگوریتم دسته‌بندی کننده داده می‌شوند. بنابراین می‌توان گام‌های لازم برای شناسایی اشیاء را به صورت زیر در نظر گرفت:

۱.۵.۱ پیش پردازش

در اغلب موارد قبل از استخراج ویژگی‌ها از تصاویر، ابتدا پیش پردازش‌هایی بر روی آن‌ها انجام می‌شود. به‌عنوان مثال می‌توان میزان روشنایی و شدت نور آن‌ها را نرمال‌سازی کرد که معمولا به صورت تفریق میانگین شدت نور تصویر و تقسیم بر انحراف معیار انجام می‌شود. البته این عملیات‌ها بسته به کاربرد مورد نظر دارد و نمی‌توان مشخص کرد که کدام یک از آن‌ها منجر به تولید نتایج بهتری خواهند شد و معمولا به صورت آزمون و خطا انجام می‌شوند. اما به‌عنوان یکی از اساسی‌ترین و رایج‌ترین عملیات‌های پیش پردازش می‌توان به برش و تغییر اندازه تصاویر به یک اندازه مشخص اشاره کرد؛ زیرا مرحله استخراج ویژگی‌ها باید بر روی تصاویر با ابعاد یکسان و ثابت انجام شود.

۲.۵.۱ انتخاب و استخراج ویژگی

انتخاب ویژگی‌های درست نقش مهم و اساسی در ردیابی ایفا می‌کند. مهم‌ترین خصوصیتی که یک ویژگی باید داشته باشد این است که یکتا باشد و بتواند شئ مورد نظر را به سادگی از محیطی که در آن قرار گرفته است متمایز کند. انتخاب ویژگی تا حد زیادی به نحوه نمایش شئ مربوط می‌شود. برای مثال زمانی که از هیستوگرام برای نمایش ظاهر شئ استفاده می‌شود، رنگ یک ویژگی خوب به شمار می‌آید و وقتی نمایش ظاهر شئ بر اساس کانتور باشد، یال‌های به دست آمده نقش یک ویژگی مناسب را ایفا می‌کنند. همچنین به کارگیری ترکیبی از این ویژگی‌ها مانند رنگ، یال، بافت و روشنایی تصویر امری رایج و پر استفاده است.

در روش‌های کلاسیک معمولا انتخاب ویژگی‌ها به طور دستی توسط کاربران انتخاب می‌شده اما پس از مدتی مسئله انتخاب ویژگی‌ها به صورت خودکار مطرح گردید که توجه محققین را به خود جلب کرد. روش‌های انتخاب ویژگی به صورت خودکار را می‌توان به دو دسته تقسیم‌بندی کرد. دسته اول تلاش می‌کنند تا ویژگی‌های موجود را بر اساس ضوابط کلی مانند عدم هم‌بستگی ویژگی‌ها انتخاب کنند. استفاده از تحلیل مولفه‌های اصلی (PCA) یکی

از تکنیک‌های موجود برای انتخاب ویژگی‌ها در این حالت است. دسته دوم بر اساس میزان اهمیت و مفید بودن ویژگی‌ها در کاربرد مورد نظر اقدام به انتخاب آن‌ها می‌کنند [۱۵۶].

به طور کلی می‌توان ویژگی‌های موجود در اشیاء را به دو دسته ویژگی‌های سطح پایین[۱] و ویژگی‌های سطح بالا[۲] تقسیم کرد. ویژگی‌های سطح پایین به ویژگی‌هایی اطلاق می‌شود که بدون هیچ‌گونه اطلاعات خاصی در مورد شئ مورد نظر می‌توان آن‌ها را استخراج کرد و از آن‌ها برای استخراج ویژگی‌های سطح بالا استفاده می‌شود. همچنین منظور از ویژگی‌های سطح بالا ویژگی‌هایی است که برای استخراج آن‌ها نیاز به داشتن اطلاعاتی در مورد اشیاء بوده و برای تعیین شکل (مکان، جهت و اندازه) اشیاء در تصاویر دیجیتال مورداستفاده قرار می‌گیرند[۱۰۱]. از جمله ویژگی‌های سطح بالا می‌توان به آستانه‌گذاری و تطبیق الگو[۳] اشاره کرد.

در هر صورت واضح است که میزان اطلاعات موجود در تصاویر بسیار زیاد است و تمامی آن‌ها برای عملیات دسته‌بندی لازم نیستند. بنابراین نیاز است تا با استخراج ویژگی‌های حیاتی و تاثیرگذار در تصمیم‌گیری، اطلاعات موجود در تصاویر کمی ساده‌تر شود. در استخراج شکل اشیاء ویژگی‌هایی حائز اهمیت هستند که نامتغیر[۴] بوده و در شرایط مختلف نتایج یکسانی داشته باشند. تغییرات روشنایی، تغییرات مکان، چرخش، زاویه دید و اندازه اشیاء از مهمترین ویژگی‌هایی هستند که انتظار می‌رود نامتغیر باشند و در نتیجه حاصل شده تاثیرگذار نباشند.

از جمله محبوب‌ترین و شناخته شده ترین الگوریتم‌های رایج برای آشکارسازی و استخراج ویژگی‌ها در شناسایی اشیاء با روش‌های کلاسیک عبارت‌اند از:

۱. SIFT[۵]:

روش SIFT در سال ۱۹۹۹ توسط دیوید لو مطرح شد [۹۲]. هدف کلی آن برطرف کردن مشکلات موجود در استخراج ویژگی‌های سطح پایین و استفاده از آن‌ها در تطابق‌دهی تصاویر بود به‌طوری که این ویژگی‌ها نسبت به موارد گفته شده نامتغیر باشند. در این روش با افزایش مقدار داده‌ها و همچنین ابعاد تصاویر، مشکلاتی مانند حجم محاسبات سنگین با زمان بالا پدید خواهند آمد [۱۳۷]. نسخه بهبود یافته این الگوریتم در سال ۲۰۰۴ ارائه شد که تا حدودی سعی داشت نقاط ضعف نسخه اولیه خود را برطرف نماید [۹۱].

۲. Haar-like:

[1] Low-Level Features
[2] High-Level Features
[3] Template Matching
[4] Invariance
[5] Scaled Invariante Feature Transform

این روش توسط ویولا و جونز در سال ۲۰۰۴ و برای شناسایی چهره معرفی شد [۱۵۰]. این الگوریتم با استفاده از تصاویری که در آن‌ها چهره وجود دارد (تصاویر مثبت) و تصاویری که در آن‌ها چهره وجود ندارد (تصاویر منفی) تابعی را آموزش می‌دهد تا بتواند مولفه‌هایی مانند فاصله بین چشم‌ها را پیدا کند. البته این الگوریتم منحصر به چهره نیست و می‌توان با آموزش دادن تابع مورد نظر برای هر شئ دلخواه آن را شناسایی کرد.

۳. SURF[1]:

الگوریتم SURF در سال ۲۰۰۴ توسط هربرت و همکارانش پیشنهاد شد [۱۰]. در این روش ابتدا یک ماتریس Hessian با استفاده از یک فیلتر ساخته شده و سپس با تخمین دترمینان آن، ویژگی‌های مورد نظر به دست خواهد آمد. این روش دارای پایداری و سرعت استخراج بیش‌تری بوده و هزینه محاسباتی آن نسبت به روش SIFT کم‌تر است [۸۵]. با این وجود این روش نیز برای کاربردهایی که باید به صورت بلادرنگ انجام شود مناسب نیست. به این جهت یانگ و همکارانش روشی جدید پیشنهاد دادند و مدعی شدند که این روش نسبت به روش‌های قبل پایداری بیشتری دارد و همچنین قابل استفاده برای کاربردهای بلادرنگ است [۸۴].

۴. HOG[2]:

روش هیستوگرام گرادیان در سال ۲۰۰۵ توسط دالال و همکارانش مطرح و باعث به وجود آمدن اولین الگوریتم شناسایی اشیاء شد که از آن برای شناسایی انسان‌ها استفاده کردند [۳۱]. ایده اصلی و کلی این روش به این صورت است که ویژگی‌های محلی اشیاء موجود در تصاویر را می توان با استفاده از توزیع چگالی گرادیان‌ها تفسیر کرد. برای این کار تصویر به قسمت‌هایی تحت عنوان سلول تقسیم‌بندی شده و به ازای هر پیکسل موجود در هر یک از سلول‌های به دست آمده جهت گرادیان آن محاسبه می‌شود. با ترکیب جهت‌های حاصل شده یک هیستوگرام برای هر سلول به دست می‌آید. به منظور مقاوم کردن هرچه بیشتر ویژگی‌ها در مقابل مواردی مانند تغییرات روشنایی، چرخش و تغییرات اندازه، این ویژگی‌ها باید نرمال شوند که این کار به واسطه محاسبه انرژی در ناحیه‌های بزرگ‌تری نسبت به سلول‌ها به نام بلوک انجام می‌گردد. پس از آن با توجه به انرژی به دست آمده مقدار هیستوگرام تمامی سلول‌های موجود در بلوک‌ها نرمال‌سازی می‌شوند.

[1] Speeded up robust features
[2] Histogram of Oriented Gradients

۳.۵.۱ دسته‌بندی و الگوریتم‌های یادگیری آن

با استخراج ویژگی‌ها، تصاویر به بردارهای ویژگی تبدیل می‌شوند. در گام بعدی الگوریتم‌های دسته‌بندی با توجه به این بردارها، برچسب شئ مورد نظر (شئ یا پس زمینه آن) را مشخص می‌کنند. فرآیند الگوریتم‌های مختلف در این زمینه متفاوت است اما در حالت کلی بردارهای ویژگی را به‌عنوان نقاطی در فضای با ابعاد بالا دریافت می‌کنند و سپس برای یافتن ابر صفحه متمایز کننده بین آن‌ها تلاش می‌کند. از نمونه الگوریتم‌های دسته‌بندی محبوب می‌توان به درخت تصمیم، روش‌های بیزین، Boosting، شبکه‌های عصبی و ماشین بردار پشتیبان[1] اشاره کرد که یکی از شناخته شده‌ترین و پر استفاده‌ترین الگوریتم‌های دسته‌بندی دودویی و به صورت با ناظر[2] است. ایده این الگوریتم برای اولین بار در سال ۱۹۶۳ معرفی شد و نمونه اصلاح شده آن که در حال حاضر از آن استفاده می‌شود توسط کورتس و واپنیک در سال ۱۹۹۵ پیشنهاد گردید [۲۸].

معمولا عملکرد الگوریتم‌های دسته‌بندی کننده برای داده‌هایی که قابلیت تفکیک‌پذیری به صورت خطی را دارند و همچنین ابعاد کمی داشته یا به عبارت دقیق‌تر ویژگی‌های کمتری را شامل می‌شوند، مناسب است. اما در عمل واضح است که همواره و حتی در اکثر موارد داده‌های موجود به صورت خطی تفکیک‌پذیر نیستند و در شرایط عادی به هیچ عنوان نمی‌توان معادله خطی و یا ابر صفحه جداکننده‌ای برای آن‌ها یافت. از طرف دیگر ویژگی‌های موجود در تصاویر دارای ابعاد بسیار بالایی هستند. به‌عنوان مثال بردارهای ویژگی حاصل از روش HOG دارای ابعادی با اندازه ۳۷۸۰ هستند. در چنین شرایطی نمی‌توان از دسته‌بندی کننده‌های موجود و با حالت عادی به هدف مورد نظر یعنی دسته‌بندی مناسب یا همان شناسایی صحیح دست یافت. در این مواقع روش‌های متعددی برای حل این مشکلات ارائه شده است. یکی از بهترین راهکارهای ارائه شده تا کنون، روش‌های مبتنی بر هسته[3] نام دارد که دارای عملکرد بسیار خوبی نیز بوده‌اند. در این روش‌ها ابتدا ویژگی‌های به دست آمده که به صورت خطی تفکیک‌پذیر نیستند توسط تابعی به فضایی با ابعاد معمولا بزرگتر نگاشت داده می‌شوند به‌طوری که در فضای جدید به آسانی و به صورت خطی تفکیک‌پذیر خواهند بود. از طرفی هزینه‌های محاسباتی تحمیل شده در این روش اندک بوده و به راحتی قابل انجام است. در ادامه به طور مفصل به معرفی این روش و نحوه استفاده از آن پرداخته خواهد شد.

[1] Support Vector Machine (SVM)
[2] Supervised
[3] Kernel-base Methods

۶.۱ روش‌های مبتنی بر هسته

به طور کلی در حوزه یادگیری ماشین، روش‌های مبتنی بر هسته نوعی از الگوریتم‌های آنالیز الگو[1] هستند که یکی از معروف‌ترین آن‌ها ماشین بردار پشتیبان است. هدف اصلی آنالیز الگو مطالعه و یافتن روابط کلی بین داده‌های مورد بررسی است. در بسیاری از الگوریتم‌های موجود برای مسائل مربوط به آنالیز الگو، داده‌های خام باید توسط نگاشت ویژگی[2] هایی که توسط کاربر مشخص می‌شوند، به صورت بردارهای ویژگی ارائه گردند. این در حالی است که روش‌های مبتنی بر هسته برای این کار تنها به یک هسته (تابع شباهت بین جفت داده‌های خام) نیاز دارند که باید توسط کاربر مشخص شود. روش‌های مبتنی بر هسته قابلیت فعالیت در فضای ویژگی ضمنی و با ابعاد بالا را به کاربر می‌دهند به‌طوری که برای این منظور نیازی به اطلاع از مختصات داده‌ها در فضای ویژگی جدید ندارند و تنها از ضرب داخلی بین همه جفت داده‌های نگاشت شده در فضای ویژگی اولیه استفاده می کنند. این نحوه عملکرد معمولا نسبت به محاسبه مختصات داده‌ها به صورت صریح، دارای پیچیدگی محاسباتی بسیار کمتری است.

برای معرفی هرچه بهتر مفهوم هسته از یک مثال ساده استفاده می‌کنیم. یک مسئله دسته‌بندی دو کلاسه را در نظر بگیرید که قرار است در آن نمونه‌ها بر اساس تنها یک معیار مشخص دسته‌بندی و یا به عبارت دیگر به مجموعه دو عضوی $\{-۱,+۱\}$ نگاشت شوند. فرض کنید که مجموعه‌داده $X = \{x_۱, \ldots, x_n\}$ موجود است و هریک از نمونه‌های آن به صورت $Y = \{y_۱, \ldots, y_n\}$ برچسب‌گذاری شده‌اند. اگر $x = (x^۱, \ldots, x^d)$ مجموعه بردارهای ویژگی برای هر نمونه در نظر گرفته شود، با توجه به داده‌های موجود نیازمند یادگیری یک تابع دسته‌بندی مانند F هستیم که به صورت زیر تعریف می‌شود:

$$F : \mathbb{R}^d \to \{-۱, +۱\}. \tag{۱-۱}$$

در حقیقت با استفاده از این تابع تصمیم‌گیری می‌شود که هر نمونه جدید متعلق به کدام یک از این دو کلاس است. معمولا روش‌های بسیار زیادی برای پارامترگذاری این تابع وجود دارد اما در روش مبتنی بر هسته، تابع F به‌عنوان یک تابع علامت از یک تابع خطی تعریف می‌شود که می‌توان آن را به صورت زیر نشان داد:

$$F(x) = sign\, f(x), \tag{۲-۱}$$

که در آن $f : \mathbb{R}^d \to \mathbb{R}$ یک تابع تصمیم‌گیر بوده و به صورت زیر پارامترگذاری می‌شود:

$$f(x) = a^۰ + a^۱x^۱ + a^۲x^۲ + \cdots + a^dx^d, \tag{۳-۱}$$

[1] Pattern Analysis
[2] Feature Map

که در آن $a^0, \ldots, a^n$ ضرایب ثابت هستند. با نوشتن $w = (a^0, \ldots, a^d) \in \mathbb{R}^{d+1}$ و با در نظر گرفتن مقدار یک برای ضریب ثابت خواهیم داشت:

$$\tilde{x} = (1, x^1, \ldots, x^d) \in \mathbb{R}^{d+1},$$

که در نتیجه می‌توان تابع f را به صورت زیر نوشت:

$$f(x) = \langle w, \tilde{x} \rangle, \tag{۴-۱}$$

که در آن $\langle \cdot, \cdot \rangle$ نشان دهنده یک ضرب داخلی است. همچنین معمولا برای ساده‌سازی نماد گذاری فرض می‌شود که عنصر اضافه شده به بردار x قبلا جزوی از این بردار بوده است بنابراین به جای $\tilde{x}$ از همان x و به جای $\mathbb{R}^{d+1}$ از نماد $\mathbb{R}^d$ استفاده می‌شود.

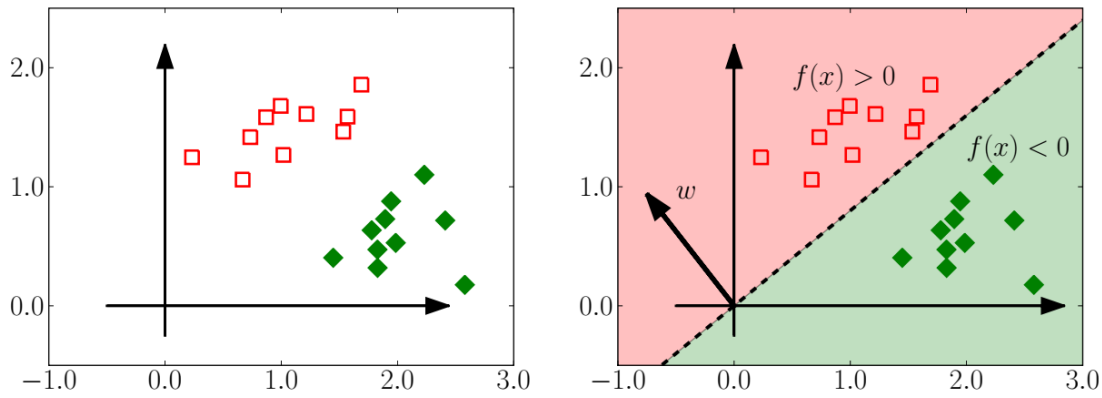
۱.۶.۱ دسته‌بندی خطی

به طور کلی سوال اصلی در مورد مسائل دسته‌بندی این است که چگونه می‌توان با استفاده از مجموعه‌داده موجود، تابع دسته‌بندی F را با دقت هرچه بیشتری به دست آورد؟ از آنجایی که تابع F کاملا توسط تابع تصمیم‌گیر (تابع f) معین می‌شود، لذا هدف اصلی مطالعات، تابع تصمیم‌گیر f خواهد بود که با نام دسته‌بندی کننده[1] نیز شناخته می‌شود. به طور مشابه از آنجایی که تابع f نیز توسط بردار وزن w معین می‌شود، مسئله یادگیری تابع f با مسئله یادگیری بردار وزن w معادل در نظر گرفته می‌شود. در تصویر (۲.۱) فرآیند دسته‌بندی و چگونگی تاثیرگذاری بردار وزن w بر روی علامت تابع f مشاهده می‌شود. در ادامه برای به دست آوردن الگوریتم‌های یادگیری مناسب، خصوصیاتی را بررسی می‌کنیم که انتظار می‌رود الگوریتم‌هایی که دارای این خصوصیات هستند، در مورد مسائل دسته‌بندی از عملکرد بالایی برخوردار باشند.

تعریف ۱.۶.۱ (صحت [۷۸]) *دسته‌بندی کننده f صحیح[2] نامیده می‌شود اگر برای تمامی نمونه‌های موجود در مجموعه داده، برچسب‌های پیش‌بینی شده توسط تابع f با برچسب های از پیش تعیین شده آن‌ها برابر باشد. به عبارت دیگر:*

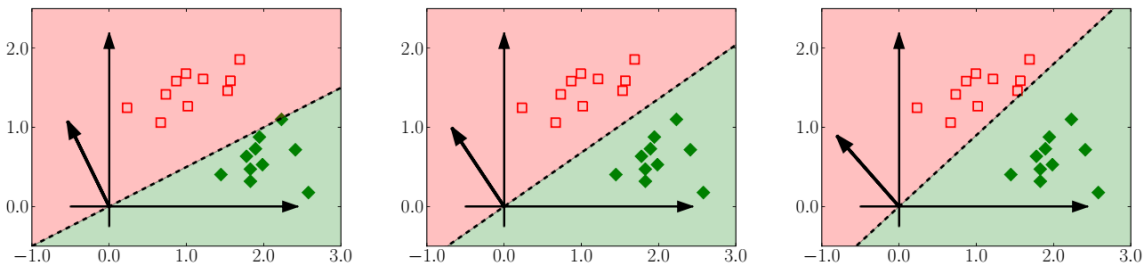
$$sign\, f(x_i) = y_i, \quad \forall i = 1, \ldots, n. \tag{۵-۱}$$

[1] Classifier
[2] Correct



شکل ۲.۱: در تصویر سمت چپ نمونه‌های دو کلاسه موجود در مجموعه داده مورد نظر مشاهده می‌شود. در تصویر سمت راست یک بردار وزن w با استفاده از یک تابع تصمیم‌گیر خطی $f(x) = \langle w, x \rangle$ تقسیم‌بندی فضای داده‌ها را نتیجه می‌دهد.

همانطور که واضح است صحت و درستی دسته‌بندی کننده یک ویژگی مطلوب است اما معمولا انتخاب‌های زیادی برای تعیین بردار وزن w وجود دارد که البته تعداد کمی از آن‌ها یک دسته‌بندی کننده خوب و قابل قبول را نتیجه می‌دهند. این حالت را می‌توان در شکل (۳.۱) نیز مشاهده کرد. در این تصویر همانطور که مشخص است هر سه دسته‌بندی کننده مشخص شده تمامی نمونه‌ها را به درستی دسته‌بندی کرده‌اند اما با این وجود اکثر افراد با دسته‌بندی کننده‌ای که در وسط قرار گرفته بیشتر موافق هستند؛ زیرا این مورد برای تصمیم‌گیری در مورد نمونه‌های بعدی از مقاومت بیشتری برخوردار خواهد بود.



شکل ۳.۱: در این تصویر هر سه دسته‌بندی کننده تمامی نمونه‌ها را به طور صحیح برچسب‌گذاری کرده‌اند اما میزان کارایی و عملکرد آن‌ها با یکدیگر متفاوت است و هر سه به‌عنوان یک دسته‌بندی خوب شناخته نمی‌شوند.

تعریف ۲.۶.۱ (پایداری [۷۸]) *میزان پایداری[1] یک دسته‌بندی کننده مانند f برابر است با بیشترین میزان ρ که می‌توان بردارهای نمونه[2] موجود در مجموعه‌داده را در جهت ابر صفحه تصمیم جابجا کرد بدون اینکه نتیجه پیش‌بینی*

[1] Robustness
[2] Sample Vectors

f *در مورد آن‌ها تغییر یابد.*

$$sign\, f(x_i + \epsilon) = sign\, f(x_i), \quad \forall i = ۱, \ldots, n, \tag{۶-۱}$$

برای تمامی جابجایی های $\epsilon \in \mathbb{R}^d$ *به‌طوری‌که* $||\epsilon|| < \rho$.

بر خلاف صحت، میزان پایداری یک دسته‌بندی کننده برابر با یک مقدار عددی است و لذا می‌توان میزان پایداری یک دسته‌بندی کننده با بردار وزن متناظرش را با سایر دسته‌بندی کننده‌های دیگر و بردارهای وزن متناظرشان مقایسه کرد.

قضیه ۳.۶.۱ (پایداری تابع دسته‌بندی کننده [۷۸]) *پایداری یک تابع دسته‌بندی کننده مانند* $f(x) = \langle w, x \rangle$ *برابر است با:*

$$\rho = \min_{i=۱,\ldots,n} |\langle \frac{w}{||w||}, x_i \rangle|. \tag{۷-۱}$$

اثبات: طبق رابطه $f(x_i) = \langle w, x_i \rangle$ خواهیم داشت:

$$f(x_i + \epsilon) = \langle w, x_i + \epsilon \rangle = \langle w, x_i \rangle + \langle w, \epsilon \rangle = f(x_i) + \langle w, \epsilon \rangle, \tag{۸-۱}$$

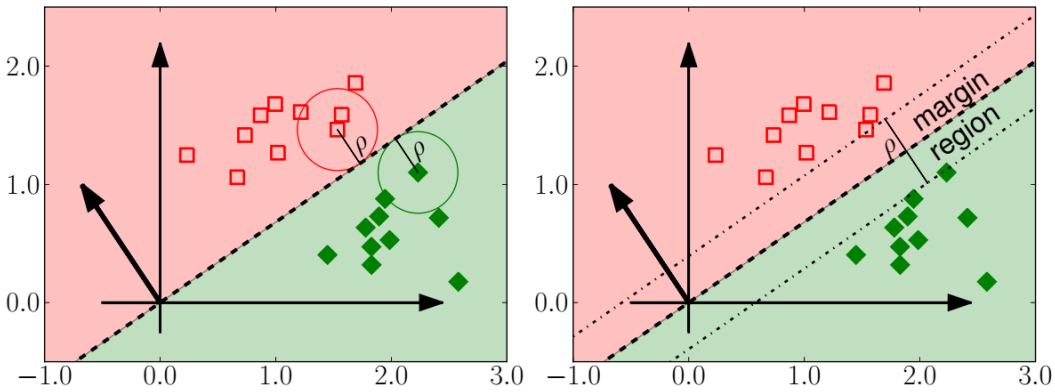
که با توجه به $\epsilon = \pm \frac{||\epsilon||}{||w||} w$ نتیجه خواهد شد:

$$f(x_i) - ||w||\,||\epsilon|| \quad \leq \quad f(x_i + \epsilon) \quad \leq \quad f(x_i) + ||w||\,||\epsilon||. \tag{۹-۱}$$

حال به منظور برقرار بودن معادله (۱-۶) برای تمامی داده‌های آموزشی باید داشته باشیم $|f(x_i)| \geq ||w||\,||\epsilon||$ که در آن $i = ۱ \ldots, n$. از آنجایی که این نامعادله برای تمامی داده‌ها برقرار است لذا برای داده با کمترین مقدار در رابطه (۱-۷) نیز برقرار خواهد بود. □

به طور هندسی $\frac{w}{||w||}$ یک بردار نرمال واحد برای ابر صفحه تصمیم است که منظور از این ابر صفحه تصمیم، صفحه‌ای است که به صورت $H_f = \{f(x) = ۰\}$ نشان داده می‌شود. همچنین $|\langle \frac{w}{||w||}, x_i \rangle|$ میزان فاصله نمونه x_i تا ابر صفحه تصمیم H_f را اندازه‌گیری می‌کند. همانطور که در شکل (۴.۱) نیز مشخص است پایداری تابع f برابر است با نصف عرض بزرگترین نوار موجود در طرفین ابر صفحه تصمیم که هیچ داده آموزشی در آن نوار قرار ندارد و معمولا حاشیه هندسی[1] نامیده می‌شود.

[1] Geometric Margin



شکل ۴.۱: پایداری یک دسته‌بندی کننده خطی یا به عبارت دیگر میزان فاصله‌ای که می‌توان داده‌های آموزشی را در جهت ابر صفحه یا مرز تصمیم حرکت داد بدون اینکه از آن عبور کنند.

تعریف ۴.۶.۱ (دسته‌بندی کننده با حاشیه حداکثری [۷۸]) *به دسته‌بندی کننده‌ای که ویژگی‌های صحت و حاشیه حداکثری را با یکدیگر و به طور همزمان داراست، یک دسته‌بندی کننده با حاشیه حداکثری[1] گویند.*

قضیه ۵.۶.۱ *فرض کنید $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ یک مجموعه‌داده آموزشی بوده و $Y = \{y_1, \ldots, y_n\}$ مجموعه برچسب‌های نمونه‌های آن باشد به‌طوری که به ازای هر i، $y_i \in \{-1, +1\}$. در این صورت بردار وزن w برای یک دسته‌بندی کننده با حاشیه حداکثری را می‌توان با حل مسئله بهینه‌سازی زیر به دست آورد.*

$$\min_{w \in \mathbb{R}^d} \ ||w||^2 \qquad (۱۰-۱)$$

$$y_i \langle w, x_i \rangle \geq 1, \quad \forall i = 1, \ldots, n. \qquad (۱۱-۱)$$

اثبات:

در حقیقت عبارت (۱-۱۰) در حال توصیف یک دسته‌بندی کننده با حاشیه حداکثری است زیرا ویژگی این نوع دسته‌بندی کننده می‌تواند به صورت زیر نیز بیان شود:

$$\max_{w \in \mathbb{R}^d} \ \rho \qquad (۱۲-۱)$$

$$\rho = \min_{i=1,\ldots,n} \left| \langle \frac{w}{||w||}, x_i \rangle \right| \qquad (۱۳-۱)$$

و همچنین

$$sign\langle w, x_i \rangle = y_i, \quad \forall i = 1, \ldots, n. \qquad (۱۴-۱)$$

[1] Maximum Margin Classifier

حال اگر با فرض اینکه ρ کمترین مقدار از تمامی مقادیر $\left|\langle \frac{w}{||w||}, x_i \rangle\right|$ به ازای $i = ۱, \ldots, n$ است، می‌توان ρ را محدودسازی کرده و مسئله را از حالت کمینه‌سازی خارج کرد. حال با اعمال بیشینه‌سازی بر روی آن، مسئله بهینه‌سازی (۱-۱۴)-(۱-۱۲) با حل مسئله زیر معادل خواهد بود:

$$\max_{w \in \mathbb{R}^d, \rho \in \mathbb{R}^+} \rho \tag{۱-۱۵}$$

$$\left|\langle \frac{w}{||w||}, x_i \rangle\right| \geq \rho, \quad \forall i = ۱, \ldots, n, \tag{۱-۱۶}$$

و همچنین

$$sign\langle w, x_i \rangle = y_i, \quad \forall i = ۱, \ldots, n. \tag{۱-۱۷}$$

با اعمال محدودیت‌های تساوی بر روی نامساوی‌ها و تقسیم بر ρ، معادل دیگری از آن به صورت زیر به دست خواهد آمد:

$$\max_{w \in \mathbb{R}^d, \rho \in \mathbb{R}^+} \rho \tag{۱-۱۸}$$

$$y_i \langle \frac{w}{\rho ||w||}, x_i \rangle \geq ۱, \quad \forall i = ۱, \ldots, n. \tag{۱-۱۹}$$

حال با در نظر گرفتن $\tilde{w} = \frac{w}{\rho ||w||}$ و استفاده از $||w|| = \frac{۱}{\rho}$ مسئله معادل دیگری به صورت زیر پدید می‌آید:

$$\max_{\tilde{w} \in \mathbb{R}^d} \frac{۱}{||\tilde{w}||} \tag{۱-۲۰}$$

$$y_i \langle \tilde{w}, x_i \rangle \geq ۱, \quad \forall i = ۱, \ldots, n. \tag{۱-۲۱}$$

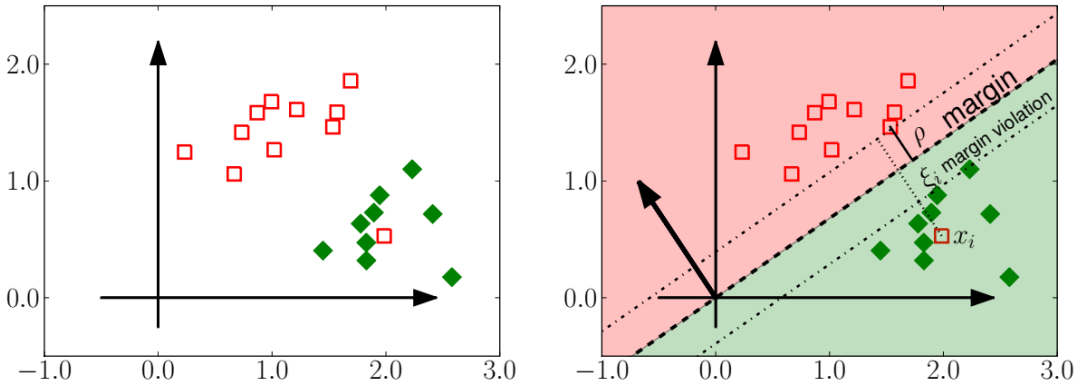
در نهایت این مسئله با فرمول ارائه شده در قضیه معادل است؛ زیرا بیشینه‌سازی بر روی $\frac{۱}{||\tilde{w}||}$ معادل کمینه‌سازی بر روی $||\tilde{w}||$ یا معادل آن یعنی $||\tilde{w}||^۲$ است [۷۸]. □

از دیدگاه بهینه‌سازی حل مسئله موجود در قضیه (۵.۶.۱) به طور عددی به خوبی قابل فهم و نسبتا آسان است؛ زیرا تابع هدف از درجه دوم بوده یا به عبارت دیگر محدب و قابل مشتق‌گیری به صورت جزئی است و تمامی محدودیت‌ها نیز خطی هستند. بنابراین می‌توان پاسخ بهینه سراسری را در $O(n^۲d + n^۳)$ به دست آورد [۱۸].

همچنین الگوریتم‌های متعدد دیگری نیز وجود دارند که می توانند پاسخ نزدیک به پاسخ بهینه را در مدت زمان کمتری بیابند [۱۰۶، ۶۶، ۲۲، ۱۲۹].

دسته‌بندی کننده با حاشیه نرم

تا به حال فرض بر این بود که دسته‌بندی کننده با حاشیه حداکثری الزاما وجود دارد اما بسته به توزیع نمونه‌های آموزشی ممکن است همواره چنین نباشد. همانطور که در تصویر سمت چپ از شکل (۵.۱) قابل مشاهده است، به خاطر وجود یک داده پرت □ در بین نمونه‌های ♦ هیچ دسته‌بندی کننده خطی صحیحی وجود ندارد تا داده‌های این دو کلاس را دسته‌بندی کند. از دیدگاه ریاضیاتی این بدین معناست که هیچ بردار وزن w وجود ندارد تا تمامی محدودیت‌های (۱-۱۱) را به طور همزمان برآورده کند و در نتیجه این امر باعث می‌شود تا حل مسئله (۱-۱۰) غیر ممکن باشد. با این وجود می‌توان محدودیت‌های سخت (محدودیت‌هایی که الزاما باید برآورده شوند) (۱-۱۱) را به محدودیت‌های آسان‌تری تبدیل کرد. به عبارت دیگر به داده‌ها اجازه داده می‌شود تا کمی نسبت به ابر صفحه تصمیم تجاوز کنند ولی به ازای هر خطا با یک مولفه هزینه مشخص جریمه خواهند شد. به تصویر سمت راست از شکل (۵.۱) توجه کنید.



شکل ۵.۱: در تصویر سمت چپ به دلیل وجود یک نمونه پرت □ در بین نمونه‌های ♦ و عدم وجود هیچ دسته‌بندی کننده خطی صحیح، نیاز به دسته‌بندی کننده با حاشیه نرم است. در تصویر سمت راست می‌توان از یک دسته‌بندی کننده خطی استفاده کرد اما تمامی نمونه‌ها به درستی دسته‌بندی نمی‌شوند. داده پرت x_i در طرف نادرستی از ناحیه حاشیه‌ای قرار گرفته است و مقدار ξ_i مقدار تجاوز از حاشیه برای آن نامیده می‌شود.

تعریف ۶.۶.۱ (دسته‌بندی کننده با حاشیه حداکثری نرم [۷۸]) *فرض کنید* $X = \{x_1, \dots, x_n\} \subset \mathbb{R}^d$ *یک مجموعه‌داده آموزشی بوده و* $Y = \{y_1, \dots, y_n\}$ *مجموعه برچسب‌های نمونه‌های آن باشد به‌طوری که به ازای هر* i، $y_i \in \{-1, +1\}$. *در این صورت بردار وزن* $w \in \mathbb{R}^d$ *برای یک دسته‌بندی کننده با حاشیه حداکثری نرم*[1] *و برای هر*

[1] Maximum Soft-Margin Classifier

ثابت تنظیم‌کننده[1] مانند $C > ۰$ با حل مسئله بهینه‌سازی زیر به دست خواهد آمد:

$$\min_{\substack{w\in\mathbb{R}^d \\ \xi_۱,\dots,\xi_n\in\mathbb{R}^+}} ||w||^۲ + \frac{C}{n}\sum_{i=۱}^{n}\xi_i \qquad (۲۲\text{-}۱)$$

$$y_i\langle w, x_i\rangle \geq ۱ - \xi_i, \quad \forall i = ۱,\dots,n. \qquad (۲۳\text{-}۱)$$

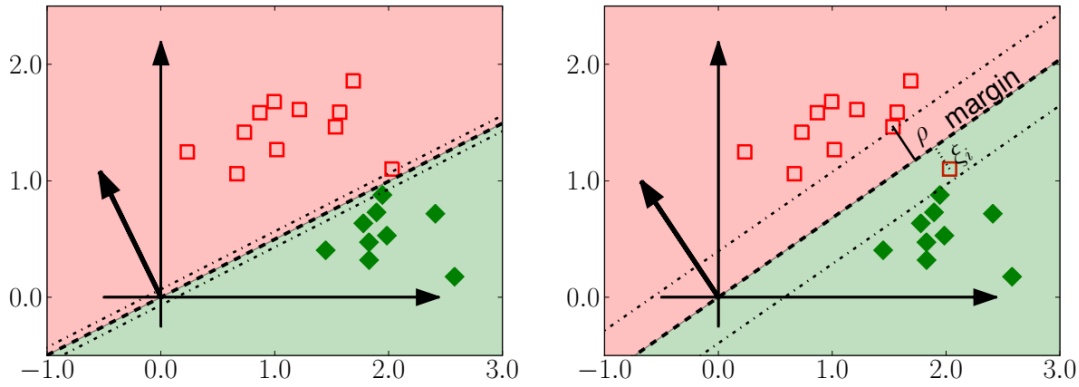
در اینجا متغیر جدید ξ_i یک متغیر کمکی[2] نامیده می‌شود. بر خلاف دسته‌بندی کننده نوع قبلی، دسته‌بندی کننده با حاشیه حداکثری نرم همیشه وجود دارد؛ زیرا همواره می‌توان محدودیت‌های (۱-۲۳) را با بزرگ در نظر گرفتن مقدار ξ_i به اندازه کافی برآورده نمود.

همانطور که واضح است $\xi_i > ۰$ به این معناست که نمونه x_i داخل سمت نادرستی از ابر صفحه تصمیم (برای $\xi_i > ۱$) و یا حداقل داخل ناحیه حاشیه‌ای (برای $۰ < \xi_i \leq ۱$) قرار گرفته است بنابراین تمایل بر این است تا ξ_i تا حد امکان کوچک باشد که این تمایل به واسطه اضافه کردن عبارت $\frac{C}{n}\sum_i \xi_i$ به مسئله بهینه‌سازی (۱-۲۲) برآورده خواهد شد.

مقدار ثابت C در تعریف (۶.۶.۱) که به‌عنوان ضریب جریمه نیز شناخته می‌شود، اجازه خواهد داد تا بین ویژگی‌های صحت و پایداری دسته‌بندی کننده موازنه و تعادلی برقرار شود. همانطور که در شکل (۶.۱) مشخص است، حتی در شرایطی که دسته‌بندی کننده صحیح موجود است یا به عبارت دیگر پاسخ مسئله (۱-۱۰) وجود دارد، ممکن است مایل باشیم یک یا تعداد محدودی از داده‌های پرت وجود داشته باشد و از دسته‌بندی کننده با حاشیه نرم استفاده شود. در این حالت این امید وجود دارد تا دسته‌بندی کننده پایدارتر و با نتیجه به مراتب بهتری نسبت به حالت قبلی حاصل شود و در نمونه‌های بعدی اشتباهات کمتری در امر دسته‌بندی رخ دهد.

در مورد ثابت C این نکته وجود دارد که اگر مقدار آن تا حد زیادی کوچک باشد، تجاوز از حاشیه موضوع مهمی نخواهد بود و مسئله بهینه‌سازی بر روی کمینه کردن $||w||^۲$ متمرکز می‌شود و در نتیجه منجر به حاصل شدن ناحیه حاشیه‌ای بزرگی خواهد شد. از طرف دیگر اگر C مقدار نسبتا بزرگی داشته باشد، هر مورد تجاوز از حاشیه جریمه سنگینی متحمل خواهد شد و به احتمال زیاد دسته‌بندی کننده با حاشیه نرم بسیار به دسته‌بندی کننده نوع قبلی نزدیک و یا حتی با آن یکسان خواهد بود. به طور کلی و در اکثر موارد نمی‌توان از قبل مشخص کرد که مقدار C باید چه میزان باشد. بسته به مجموعه داده‌ای که در اختیار است باید مقدار آن را به نحوی تعیین کرد تا بهترین

[1] Regularization Constant
[2] Slack Variable



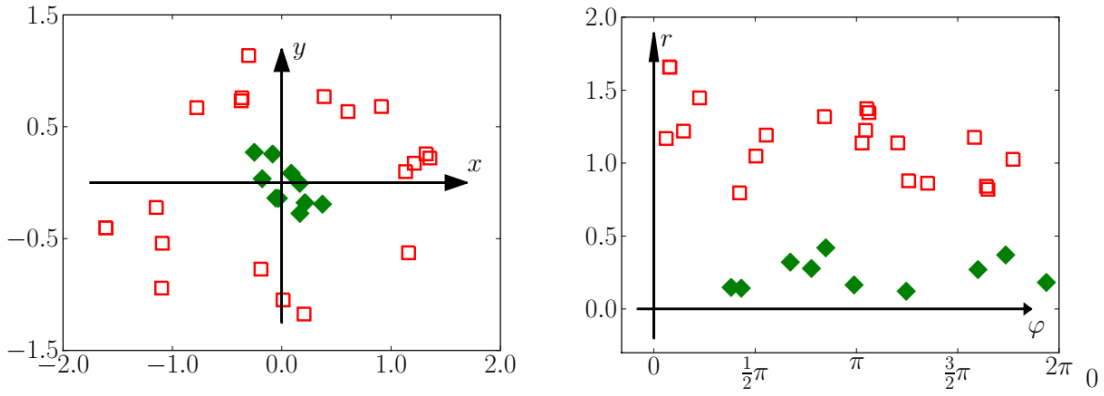
شکل ۶.۱: علیرغم اینکه در تصویر سمت چپ دسته‌بندی کننده صحیح وجود دارد اما معمولا ترجیح داده می‌شود که حتی با وجود چند خطا، دسته‌بندی کننده مقاوم‌تری انتخاب شود (تصویر سمت راست).

عملکرد ممکن را داشته باشد. بنابراین راهکار متداولی که برای تعیین مقدار آن وجود دارد این است که از یک مجموعه اعتبارسنجی و یا از روش اعتبارسنجی متقابل[1] استفاده شود.

۲.۶.۱ دسته‌بندی غیرخطی

در برخی مواقع هیچ امیدی برای یافتن یک دسته‌بندی کننده خطی حتی با امکان تجاوز از حاشیه و استفاده از دسته‌بندی کننده با حاشیه نرم وجود ندارد. در شکل (۷.۱) یک نمونه از این شرایط نشان داده شده است که در تصویر سمت چپ هیچ دسته‌بندی کننده خطی با عملکرد قابل قبولی به دست نخواهد آمد. به همین خاطر ایده‌های متعدد بسیاری در رابطه با توسعه دسته‌بندی کننده‌های غیرخطی مطرح شده که اکثر آن‌ها بر اساس ترکیب تعداد زیادی از دسته‌بندی کننده‌های خطی و تبدیل آن‌ها به یک دسته‌بندی کننده غیرخطی بوده است. برای مثال می‌توان به مواردی از معروف‌ترین آن‌ها مانند درخت تصمیم [۱۰۹، ۱۹]، پرسپترون چندلایه [۱۴] و یا روش افزایشی[2][۳۹] اشاره کرد. در روش‌های مبتنی بر هسته رویکرد دیگری اتخاذ می‌شود به این صورت که ابتدا یک نگاشت غیرخطی بر روی نمونه‌ها اعمال شده و پس از آن برای یافتن یک دسته‌بندی کننده خطی بر روی داده‌های نگاشت شده تلاش می‌گردد. با انتخاب یک مختصات انتقال مناسب، یک مسئله غیرخطی دشوار می‌تواند به یک مسئله خطی ساده تبدیل شود (تصویر سمت راست شکل ۷.۱). این رویکرد را می‌توان عامل تعمیم دسته‌بندی کننده‌های خطی به دسته‌بندی کننده‌های خطی تعمیم یافته[3] دانست.

[1] Cross-Validation
[2] Boosting
[3] Generalized Linear Classification



شکل ۷.۱: در تصویر سمت چپ به خاطر نوع توزیع نمونه‌ها هیچ دسته‌بندی کننده خطی با عملکرد مناسب وجود ندارد. در تصویر سمت راست اگر ابتدا نمونه‌ها به یک دستگاه مختصات دیگر (در اینجا از دستگاه مختصات دکارتی به دستگاه مختصات قطبی $(x, y) \mapsto (\theta, r)$) منتقل شوند، امکان استفاده از دسته‌بندی کننده خطی فراهم خواهد شد.

تعریف ۷.۶.۱ (دسته‌بندی کننده خطی تعمیم یافته با حاشیه حداکثری [۷۸]) *فرض کنید* $X = \{x_1, \dots, x_n\} \subset \mathbb{R}^d$ *یک مجموعه‌داده آموزشی بوده و* $Y = \{y_1, \dots, y_n\}$ *مجموعه برچسب‌های نمونه‌های آن باشد به‌طوری که به ازای هر* i، $y_i \in \{-1, +1\}$. *اگر* $\varphi : \mathbb{R}^d \to \mathbb{R}^m$ *یک نگاشت احتمالا غیرخطی در نظر گرفته شود آنگاه دسته‌بندی کننده خطی تعمیم یافته*[1] *برای هر ثابت تنظیم کننده* $C > 0$ *توسط* $F(x) = sign\, f(x)$ *به صورت زیر به دست می‌آید :*

$$f(x) = \langle w, \varphi(x) \rangle, \tag{۲۴-۱}$$

که در آن $w \in \mathbb{R}^m$ *جواب مسئله بهینه‌سازی زیر است:*

$$\min_{\substack{w \in \mathbb{R}^m \\ \xi_1, \dots, \xi_n \in \mathbb{R}^+}} ||w||^2 + \frac{C}{n} \sum_{i=1}^{n} \xi_i \tag{۲۵-۱}$$

$$y_i \langle w, \varphi(x_i) \rangle \geq 1 - \xi_i, \quad \forall i = 1, \dots, n. \tag{۲۶-۱}$$

در تعریف (۷.۶.۱) هیچ‌گونه محدودیتی برای انتخاب $\varphi : \mathbb{R}^d \to \mathbb{R}^m$ مطرح نشده اما پر واضح است که میزان عملکرد دسته‌بندی کننده تا حدود زیادی وابسته به انتخاب آن است. تعدادی از نمونه‌هایی که ثابت شده در اغلب موارد مفید واقع خواهند شد عبارت‌اند از:

- مختصات قطبی برای $(x, y) \in \mathbb{R}^2$:

$$\varphi : \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} \sqrt{x^2 + y^2} \\ arctan \frac{y}{x} \end{pmatrix}.$$

[1] Generalized Linear Max-Margin

- چند جمله‌ای‌های از درجه m برای $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$:

$$\varphi : x \mapsto \left(1, x_1, \ldots, x_d, x_1^2, \ldots, x_d^2, \ldots, x_1^m, \ldots, x_d^m\right).$$

- فاصله نوع اولیه برای $x \in \mathbb{R}^d$:

$$\varphi : x \mapsto \left(||x - p_1||, \ldots, ||x - p_N||\right),$$

که در آن $\{p_1, \ldots, p_N\} \subset \mathbb{R}^d$ یک مجموعه از تعداد N بردار اولیه است.

اگر به نگاشت چند جمله‌ای دقت شود می‌توان به این مسئله پی برد که بُعد ثانویه به دست آمده ممکن است بسیار بزرگ‌تر از بُعد اولیه داده‌ها باشد. همین مسئله باعث می‌شود بردار وزن به دست آمده برای دسته‌بندی کننده خطی تعمیم یافته در فضای $\mathbb{R}^m$ از انعطاف‌پذیری بیشتری نسبت به بردار وزن نمونه خطی آن در فضای $\mathbb{R}^d$ برخوردار باشد و در نهایت منجر به انجام دسته‌بندی با دقت بیشتری گردد. اگرچه انتظار می‌رود که با نگاشت داده‌ها به فضای جدید با بُعد بسیار بالا در مورد مصرف حافظه و زمان اجرای آن نگرانی‌هایی وجود داشته باشد اما به واسطه یکی از مهم‌ترین ویژگی‌های روش‌های مبتنی بر هسته که در ادامه مطرح می‌شود، این مسئله چندان جای نگرانی نخواهد داشت.

بردار وزن w برای یک دسته‌بندی کننده خطی تعمیم یافته با تابع نگاشت $\varphi : \mathbb{R}^d \to \mathbb{R}^m$ دارای m مولفه است که باید مشخص شوند. اما همانطور که در قضیه ذیل آورده شده است w نمی‌تواند به طور دلخواه معین شود اما در داخل زیر فضایی از $\mathbb{R}^m$ با بُعد n و یا کمتر از آن قرار خواهد گرفت که n در اینجا تعداد نمونه‌های مجموعه‌داده آموزشی است.

قضیه ۸.۶.۱ (قضیه بیان کننده (Representer Theorem) [۱۲۶]) *پاسخ مسئله کمینه‌سازی بردار وزن w برای مسئله بهینه‌سازی (۱-۲۵) را می‌توان به صورت زیر بیان نمود:*

$$w = \sum_{j=1}^{n} \alpha_j \varphi(x_j), \quad \forall \alpha_j \in \mathbb{R}, j = 1, \ldots, n. \tag{۱-۲۷}$$

اثبات: هرچند که این قضیه یکی از قضیه‌های اساسی در روش‌های مبتنی بر هسته است اما اثبات آن ساده و بدون هرگونه پیچیدگی است. فرض کنید V زیر فضایی از تمامی بردارهای حاصل از ترکیب خطی $\sum_j \alpha_j \varphi(x_j)$

بوده و همچنین $w \in \mathbb{R}^m$ پاسخ مسئله بهینه‌سازی (۱-۲۵) باشد. برای اثبات قضیه کافی است نشان داده شود که $w \in V$؛ زیرا V یک زیر فضای با بُعد متناهی است و هر بردار در $\mathbb{R}^m$ دارای یک تجزیه منحصر به فرد به یک مولفه از V و دیگری به مکمل متعامد آن یعنی $V^\perp = \{u \in \mathbb{R}^m : \langle u, v\rangle = ۰ \quad \forall v \in V\}$ است. فرض کنید $w = w_V + w_{V^\perp}$ یک نمونه از چنین تجزیه‌ای برای w باشد. بنابراین با توجه به اینکه $\varphi(x_i) \in V$ لذا برای $i = ۱, \ldots, n$ رابطه $\langle w_{V^\perp}, \varphi(x_i)\rangle = ۰$ برقرار است و در نتیجه داریم $\langle w, \varphi(x_i)\rangle = \langle w_V, \varphi(x_i)\rangle$. این عبارت نشان دهنده این است که w_V نامساوی (۱-۳۴) را با مقداری برابر با متغیر کمکی ξ_i و به‌عنوان بردار وزن w برآورده می‌کند. از طرف دیگر چون $\langle w_V, w_{V^\perp}\rangle = ۰$، $||w||^۲ = ||w_V||^۲ + ||w_{V^\perp}||^۲$ نیز بدون هیچ شرط متقابلی نتیجه خواهد شد.

سپس (برهان خلف) اگر فرض شود که $||w_{V^\perp}|| > ۰$ آنگاه این مفهوم را می‌رساند که $||w_V||^۲$ به طور اکید از $||w||^۲$ کوچک‌تر است و این در حالی است که در مسئله (۱-۲۵) فرض شد که w یک جواب کمینه است؛ لذا $w_{V^\perp} = ۰$ و به دنبال آن $w = w_v \in V$ به دست خواهد آمد [۱۲۶]. □

اولین نتیجه‌ای که از قضیه بیان کننده حاصل می‌شود این است که امکان بازنویسی مسئله یادگیری دسته‌بندی کننده خطی تعمیم یافته از یک مسئله بهینه‌سازی m بُعدی به یک مسئله بهینه‌سازی n بُعدی وجود دارد. برای این کار کافی است $w = \sum_{j=۱}^{n} \alpha_j \varphi(x_j)$ در عبارات (۱-۲۵) و (۱-۲۶) جایگذاری شود. بنابراین می‌توان به جای $w \in \mathbb{R}^m$ سعی در بهینه کردن مقادیر $(\alpha_۱, \ldots, \alpha_n) \in \mathbb{R}^n$ کرد. با استفاده از خاصیت خطی بودن ضرب داخلی خواهیم داشت:

$$\min_{\substack{\alpha_۱,\ldots,\alpha_n \in \mathbb{R} \\ \xi_۱,\ldots,\xi_n \in \mathbb{R}^+}} \quad \sum_{i,j=۱}^{n} \alpha_i \alpha_j \langle \varphi(x_i), \varphi(x_j)\rangle + \frac{C}{n} \sum_{i=۱}^{n} \xi_i \tag{۱-۲۸}$$

$$y_i \sum_{j=۱}^{n} \alpha_j \langle \varphi(x_j), \varphi(x_i)\rangle \geq ۱ - \xi_i, \quad i = ۱, \ldots, n, \tag{۱-۲۹}$$

و بنابراین تابع تصمیم خطی تعمیم یافته (۱-۲۴) به عبارت زیر تبدیل خواهد شد:

$$f(x) = \sum_{j=۱}^{n} \alpha_j \langle \varphi(x_j), \varphi(x)\rangle. \tag{۱-۳۰}$$

ویژگی خاصی که در مورد این مسئله وجود دارد این است که بردار ضرایب $(\alpha_j)_{j=۱,\ldots,n}$ تُنُک خواهد بود یا به عبارت دیگر تعداد زیادی از α_j ها در آن مقداری برابر با صفر خواهند داشت [۱۲۶]. همین امر باعث ساده‌سازی

و افزایش سرعت محاسبه تابع تصمیم (۱-۳۰) خواهد شد. با این وجود نمونه‌های آموزشی x_j که ضرایب α_j آن‌ها مخالف صفر باشد نیز وجود دارند که معمولا به آن‌ها بردارهای پشتیبان[1] گفته می‌شود.

۳.۶.۱ هسته‌گذاری

قضیه (۸.۶.۱) این امکان را می‌دهد تا مسئله بهینه‌سازی برای یافتن دسته‌بندی کننده تعمیم یافته با حاشیه حداکثری از نظر محاسباتی سبک‌تر شود. اما با این وجود همچنان محاسبه و ذخیره $\{\varphi(x_1), \ldots, \varphi(x_n)\}$ نیازمند حافظه زیادی است. اگر دقت شود مسئله (۱-۲۸) دارای ویژگی‌های نگاشت شده $\varphi(x_i)$ به صورت زوج‌های مرتب $\langle \varphi(x_i), \varphi(x_j) \rangle$ برای مقادیر $i, j = 1, \ldots, n$ است که به خاطر وجود تقارن در برخی از این زوج‌های مرتب، تعداد حقیقی و منحصر به فرد آن‌ها برابر با $\frac{n(n+1)}{2}$ است. در صورتی که این مقادیر از قبل محاسبه و ذخیره شده باشد، نیازی به ذخیره تعداد nm درایه از بردارهای $\varphi(x_i)$ نخواهد بود.

به طور کلی‌تر می‌توان تابع هسته $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ برای φ را به صورت زیر تعریف کرد:

$$k(x, x') = \langle \varphi(x), \varphi(x') \rangle. \tag{۱-۳۱}$$

اگر مسئله (۱-۲۸) را با توجه به این نمادگذاری بازنویسی کنیم یک مسئله بهینه‌سازی برای دسته‌بندی کننده خطی تعمیم یافته با حاشیه حداکثری و در قالب هسته‌گذاری شده به دست خواهد آمد که معمولا ماشین بردار پشتیبان (SVM) نامیده می‌شود.

قضیه ۹.۶.۱ (ماشین بردار پشتیبان [۲۸]) *فرض کنید $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ یک مجموعه‌داده آموزشی بوده و $Y = \{y_1, \ldots, y_n\}$ مجموعه برچسب‌های نمونه‌های آن باشد به‌طوری که به ازای هر i، $y_i \in \{-1, +1\}$. اگر $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ یک تابع هسته و $C > 0$ باشد آنگاه تابع تصمیم دسته‌بندی کننده هسته‌گذاری شده با حاشیه حداکثری برابر با عبارت زیر خواهد بود:*

$$f(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x), \tag{۱-۳۲}$$

که در آن ضرایب $\alpha_1, \ldots, \alpha_n$ با حل مسئله زیر به دست خواهد آمد:

$$\min_{\substack{\alpha_1, \ldots, \alpha_n \in \mathbb{R} \\ \xi_1, \ldots, \xi_n \in \mathbb{R}^+}} \quad \sum_{i,j=1}^{n} \alpha_i \alpha_j k(x_i, x_j) + \frac{C}{n} \sum_{i=1}^{n} \xi_i \tag{۱-۳۳}$$

[1] Support Vectors

$$y_i \sum_{j=1}^{n} \alpha_j k(x_j, x_i)\rangle \geq 1 - \xi_i, \quad i = 1, \ldots, n, \tag{۱-۳۴}$$

دلایل مناسب بودن هسته‌گذاری

با معرفی رابطه (۱-۳۱) ممکن است در نگاه اول $k(x, x')$ کاربرد و تاثیر خاصی نداشته باشد اما در واقع باید به جای $\langle \varphi(x_i), \varphi(x_j)\rangle$ برای مقادیر $i, j = 1, \ldots, n$ ،حاصل $k(x_i, x_j)$ محاسبه شود. اگرچه مقدار هر دو این عبارات دقیقا با یکدیگر برابر است اما دو دلیل مهم برای این کار وجود دارد که عبارت‌اند از:

- سرعت

در حالت کلی ممکن است عبارتی برای $k(x_i, x_j)$ یافت شود که سرعت محاسبه بیشتری نسبت به $\varphi(x_i)$ و در نتیجه $\langle \varphi(x_i), \varphi(x_j)\rangle$ داشته باشد. به‌عنوان یک مثال برای روشن‌تر شدن موضوع یک تابع نگاشت چند جمله‌ای از درجه دو مانند زیر را در نظر بگیرید:

$$\varphi : x \mapsto (1, \sqrt{2}x, x^2) \in \mathbb{R}^3 \tag{۱-۳۵}$$

که برای هر نمونه تعداد دو ضرب و برای کل نمونه‌های آموزشی تعداد $2n$ ضرب نیاز دارد. بنابراین هر ضرب داخلی تعداد پنج عملیات (سه ضرب و دو جمع) خواهد داشت که در مجموع برای تعداد $\frac{n(n+1)}{2}$ عنصر یکتای $\langle \varphi(x_i), \varphi(x_j)\rangle$ به ازای $i, j = 1, \ldots, n$ برابر با $\frac{5}{2}n(n+1)$ خواهد بود. بر این اساس قبل از شروع بهینه‌سازی تعداد $\frac{5}{2}n^2 + \frac{9}{2}n$ عملیات نیاز است. اما از طرف مقابل استفاده از تابع هسته این امکان را خواهد داد تا قبل از شروع محاسبات عبارات موجود تا حدودی ساده‌سازی شوند. به این ترتیب بر اساس نگاشت معرفی شده در رابطه (۱-۳۵) می‌توان نوشت:

$$k(x, x') = \langle \varphi(x), \varphi(x')\rangle = \langle (1, \sqrt{2}x, x^2)^t, (1, \sqrt{2}x', x'^2)^t\rangle \tag{۱-۳۶}$$

$$= 1 + 2xx' + x^2x'^2 \tag{۱-۳۷}$$

$$= (1 + xx')^2 \tag{۱-۳۸}$$

لذا در این حالت محاسبه هر کدام از $k(x_i, x_j)$ ها نیازمند تنها سه عملیات (دو ضرب و یک جمع) است. در نتیجه برای کل عناصر یکتای $k(x_i, x_j)$ به ازای $i, j = 1, \ldots, n$ تعداد $\frac{3}{2}n^2 + \frac{3}{2}n$ عملیات انجام خواهد شد و در نتیجه با هسته‌گذاری می‌توان تعداد عملیات‌های مورد نیاز را تا ۴۰٪ کاهش داد.

- انعطاف‌پذیری

در حقیقت به دلیل اینکه توابع $k(x, x')$ متناظر با ضرب‌های داخلی توابع نگاشت مانند φ هستند اما در مورد چگونگی محاسبه φ اطلاعی موجود نیست و همین موضوع باعث به وجود آمدن قضیه مهم دیگری در روش‌های مبتنی بر هسته شد.

قضیه ۱۰.۶.۱ (شرط مِرسِر (Mercer's Condition) [۹۶]) *فرض کنید χ مجموعه‌ای غیر تهی باشد. برای هر تابع هسته معین مثبت $k : \chi \times \chi \to \mathbb{R}$ یک فضای هیلبرت $\mathcal{H}$ و یک تابع نگاشت $\varphi : \chi \to \mathcal{H}$ وجود دارد به‌طوری که:*

$$k(x, x') = \langle \varphi(x), \varphi(x') \rangle_{\mathcal{H}} \tag{۳۹-۱}$$

که در آن $\langle ., . \rangle_{\mathcal{H}}$ یک ضرب داخلی در $\mathcal{H}$ است.

تعریف ۱۱.۶.۱ (تابع هسته معین مثبت [۷۸]) *فرض کنید χ مجموعه‌ای غیر تهی باشد. تابع هسته $k : \chi \times \chi \to \mathbb{R}$ معین مثبت نامیده می‌شود اگر دارای شرایط زیر باشد:*

- *k متقارن باشد یا به عبارت دیگر برای هر $x, x' \in \chi$ داشته باشیم $k(x, x') = k(x', x)$.*
- *برای هر مجموعه متناهی از نقاط مانند $x_1, \ldots, x_n \in \chi$ ماتریس هسته یا ماتریس گرام[1]*

$$K_{ij} = (k(x_i, x_j))_{ij} \tag{۴۰-۱}$$

نیمه معین مثبت باشد یا به عبارت دیگر برای تمامی بردارهای $t \in \mathbb{R}^n$ داشته باشیم:

$$\sum_{i,j=1}^{n} t_i K_{ij} t_j \geq 0. \tag{۴۱-۱}$$

به دلیل اینکه تمامی هسته‌هایی که در ادامه مورد بررسی قرار می‌گیرند در واقع معین مثبت هستند، به اختصار از هسته به جای تابع هسته معین مثبت استفاده می شود.

تعریف ۱۲.۶.۱ (فضای هیلبرت [۱۵۸]) *فضای برداری $\mathcal{H}$ یک فضای هیلبرت نامیده می‌شود اگر دارای شرایط زیر باشد:*

[1] Gram Matrix

- $\mathcal{H}$ *دارای ضرب داخلی به صورت زیر باشد:*

$$\langle .,. \rangle_{\mathcal{H}} : \mathcal{H} \times \mathcal{H} \to \mathbb{R}. \qquad (۴۲-۱)$$

- $\mathcal{H}$ *کاملا مطابق با نُرم* $||v||_{\mathcal{H}} = \sqrt{\langle v, v \rangle_{\mathcal{H}}}$ *باشد.*

به عبارت دیگر داریم:

i) $d(x, x') \geq ۰$ *برای تمامی* $x, x' \in \chi$.

ii) $d(x.x') = ۰$ *تنها برای* $x = x'$.

iii) $d(x, x') = d(x', x)$ *برای تمامی* $x, x' \in \chi$.

iv) $d(x, x') \leq d(x, x'') + d(x'', x)$ *برای تمامی* $x, x', x'' \in \chi$.

که البته در این مورد شرط دوم اهمیت به مراتب کم تری دارد اما شرط اول یعنی وجود ضرب داخلی شرطی حیاتی است؛ زیرا تابع هسته k باید به‌عنوان یک جایگزین برای محاسبه $\langle .,. \rangle_{\mathcal{H}}$ استفاده شود.

با توجه به قضیه (۱۰.۶.۱) این نتیجه استنباط می‌شود که هر تابع هسته $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ را تا زمانی که معین مثبت باشد می‌توان در زمینه دسته‌بندی کننده ماشین بردار پشتیبان (۱-۳۳) از آن استفاده کرد.

نتیجه این بهینه‌سازی بر اساس یک تابع نگاشت φ از فضای $\mathbb{R}$ به یک فضای ویژگی هیلبرت مناسب، همواره متناظر با یک دسته‌بندی کننده خطی تعمیم یافته با حاشیه حداکثری خواهد بود. توجه شود که φ و $\mathcal{H}$ هر دو به صورت ضمنی تعریف شده‌اند؛ به این معنا که وجود دارند اما ممکن است امکان محاسبه آن‌ها وجود نداشته باشد. در واقع $\mathcal{H}$ معمولا دارای ابعاد بالا و یا حتی نامتناهی است اما به دلیل اینکه تنها به واسطه ضرب داخلی آن یعنی $\langle .,. \rangle_{\mathcal{H}}$ با $\mathcal{H}$ کار خواهد شد، نگرانی چندانی در این مورد وجود ندارد.

علاوه بر آن به دلیل اینکه در شرایط قضیه (۱۰.۶.۱) به دامنه‌های ورودی $\mathbb{R}^d$ محدود نمی‌شوند، تا زمانی که بتوان یک هسته برای عناصر آن‌ها تعریف کرد می‌توان مفهوم دسته‌بندی کننده خطی تعمیم یافته را به مجموعه‌های ورودی دلخواه χ تعمیم داد.

ساخت توابع هسته

علی‌رغم ساده بودن تعریف (۱۲.۶.۱)، بررسی ضوابط تابع داده شده $k : \chi \times \chi \to \mathbb{R}$ در عمل دشوار است؛ زیرا شرط (۱-۴۱) در مورد مقادیر k بر روی زیر مجموعه‌های متناهی بزرگ و دلخواه از χ است. اما با این وجود ساخت توابع k که هسته‌های معین مثبت هستند از طریق زیر امری نسبتا آسان است.

۱) ساخت هسته‌ها از ابتدا

- برای هر $\varphi : \chi \to \mathbb{R}^m$ ، $k(x, x') = \langle \varphi(x), \varphi(x') \rangle$ یک هسته است.
- اگر $d : \chi \times \chi \to \mathbb{R}$ یک تابع فاصله باشد آنگاه $k(x, x') = exp(-d(x, x'))$ یک هسته است.

۲) ساخت هسته از روی سایر هسته‌ها

- اگر k یک هسته باشد و $\alpha \in \mathbb{R}^+$، آنگاه $k + a$ و αk نیز هسته هستند.
- اگر $k_۱$ و $k_۲$ هسته باشند، آنگاه $k_۱ + k_۲$ و $k_۱ \cdot k_۲$ نیز هسته هستند.

چند نمونه هسته

با استفاده از قانون ترکیب هسته‌ها می توان بررسی کرد که توابع زیر برای بردارهای موجود در $\mathbb{R}^d$ هسته‌های معین مثبت هستند.

- هر ترکیب خطی به صورت $\sum_j \alpha_j k_j$ که در آن k_j هسته بوده و $\alpha_j \geq ۰$.
- هر هسته چند جمله ای به صورت $k(x, x') = (۱ + \langle x, x' \rangle)^m$ که در آن $m \in \mathbb{N}$.
- هسته گوسی که به صورت $k(x, x') = exp\left(-\frac{||x-x'||^۲}{۲\sigma^۲}\right)$ تعریف می‌شود و در آن $\sigma > ۰$.

انتخاب تابع هسته و مولفه‌های آن

در قسمت قبل در مورد روش‌های ساخت هسته بحث شد که در واقع اغلب این روش‌ها از خانواده هسته‌های پارامترگذاری شده یا به عبارت دیگر دارای مولفه هستند. همچنین از وجود یک فضای هیلبرت $\mathcal{H}$ و تابع نگاشت φ نمی‌توان به سادگی نتیجه گرفت که پاسخ دسته‌بندی کننده خطی تعمیم یافته با دقت بالایی به دست خواهد آمد.

به طور کلی دو سوال مهم و اساسی برای استفاده کنندگان از روش‌های مبتنی بر هسته وجود دارد: الف) *از چه تابع هسته‌ای باید استفاده کرد؟* ب) *مجموعه پارامترهای هسته را چگونه باید مشخص نمود؟* متاسفانه هیچ پاسخ درست و عمومی برای آن‌ها وجود ندارد و معمولا بهترین رویکرد استفاده از روش‌های مختلف و سپس ارزیابی عملکرد دسته‌بندی کننده با استفاده از یک مجموعه ارزیابی و یا تکنیک ارزیابی متقابل است. در صورتی که این روش امکان‌پذیر نباشد می‌توان از یک روش اکتشافی[1] برای راهنمایی در مورد تصمیم‌گیری استفاده کرد. یکی از این روش‌ها به این صورت است که یک هسته خوب هنگامی که بر روی دو نمونه مشابه اعمال می شود باید دارای مقدار نسبتا بالایی باشد و بالعکس[2].

استدلال موجود در پشت این موضوع را می‌توان با در نظر گرفتن فرآیند یک دسته‌بندی دودویی با یک تابع نگاشت $\varphi : \chi \to \mathbb{R}$ و برچسب‌های $\{+1, -1\}$ مشاهده کرد. نتیجه تابع هسته $k(x, x') = \varphi(x)\varphi(x')$ به این صورت است که اگر x و x' متعلق به کلاس یکسانی باشند مقدار $+1$ و در غیر این صورت مقدار آن -1 خواهد بود.

۴.۶.۱ هسته‌ها در بینایی کامپیوتر

تصاویر و ویدئوها داده‌هایی هستند که خصوصیات ویژه زیادی دارند. از آنجایی که هر پیکسل از تصاویر یک مقدار را نشان می‌دهد نتیجه می شود که آن‌ها دارای ابعاد بسیار بالایی هستند. همچنین تصاویر را نمی‌توان به صورت یک نمایش برداری دلخواه (مثلا هم‌بستگی قوی بین پیکسل‌های مجاور) مشاهده کرد. به همین علت محققان بینایی ماشین توجهات ویژه‌ای بر روی یافتن روش‌های مناسب برای نمایش داده‌ها و الگوریتم‌ها داشته‌اند تا باعث شود مسائل موجود در این زمینه آسان‌تر حل شوند.

ثابت شده است که روش‌های مبتنی بر هسته در تمامی مسائل بینایی کامپیوتر به طور موفق عمل می‌کنند که دلیل عمده آن انعطاف‌پذیری و قابلیت تفسیر آن‌هاست. در این نوع روش‌ها برای ساخت یک تابع هسته می‌توان از دانش‌های موجود در مورد مسائلی که از قبل در اختیار بوده‌اند است استفاده نمود. از طرف دیگر هنگامی که یک تابع هسته مطلوب طراحی می‌شود، می‌توان از آن در هر الگوریتم دیگری که بر این اساس کار می‌کند استفاده کرد و تنها برای کاربرد اولیه خودش ساخته نخواهد شد. همین امر باعث شده تا محققان و افرادی که در این زمینه کار می‌کنند یک مجموعه بزرگ از توابع هسته را ایجاد کرده و در موارد بعدی توابع مورد نظر را از این مجموعه انتخاب

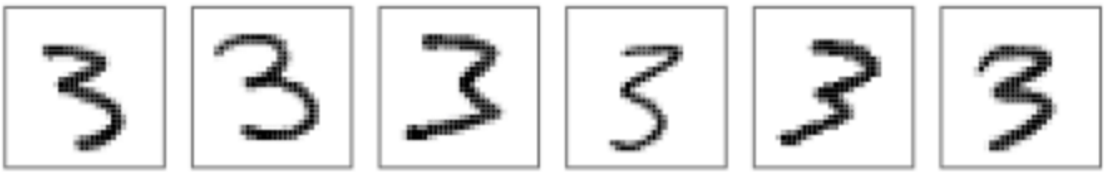
[1] Heuristic

[2] به خاطر همین خاصیت می‌توان هسته‌ها را به دید یک کلاس ویژه از معیارهای تشابه نیز در نظر گرفت.

کنند که این کار باعث افزایش شانس برای انتخاب یک تابع هسته مناسب با عملکرد خوب می‌شود. در ادامه برخی از این هسته‌ها همراه با فرضیاتی که بر اساس آن‌ها شکل گرفته‌اند و همچنین کاربردهای آن‌ها در مسائل بینایی کامپیوتر بررسی خواهد شد.

طراحی هسته برای تصاویر

ساده‌ترین روش برای رفتار با تصاویر، در نظر گرفتن آن‌ها به‌عنوان برداری از پیکسل‌ها است. تصاویر با ابعاد $N \times M$ با مقادیر بین $[۰, ۲۵۵]$ و یا $[۰, ۱]$ بردارهایی به صورت $x \in \mathbb{R}^{NM}$ خواهند بود و بنابراین می‌توان هسته‌های موجود برای داده‌های برداری مانند هسته‌های خطی و گوسی را بر روی آن‌ها اعمال کرد. در مورد تصاویر دودویی $x \in \{۰, ۱\}^{NM}$ ثابت شده است که معمولا هسته‌های چند جمله‌ای به طور موفق عمل می‌کنند. به خاطر اینکه این هسته‌ها تنها می‌توانند بردارهای با طول یکسان را با یکدیگر مقایسه کنند، پر واضح است که تصاویر در اختیار همگی بایستی دارای ابعاد یکسان باشند. متاسفانه در نظر گرفتن تصاویر به صورت یک مجموعه از بردارها، دارای برخی نتایج غیر شهودی است. به طور دقیق‌تر مقایسه یک جفت پیکسل به صورت داده‌های برداری می‌تواند کمی سخت‌گیرانه باشد. به‌عنوان مثال دو تصویری که توسط انسان بسیار شبیه به هم در نظر گرفته می‌شوند، اگر با معیارهای اختلاف بین یک جفت پیکسل مانند نُرم اقلیدسی مقایسه شوند، با یکدیگر بسیار متفاوت خواهند بود (شکل ۸.۱).



شکل ۸.۱: با وجود اینکه این تصاویر از نظر ظاهری متفاوت هستند و فاصله اقلیدسی نسبتا زیادی با یکدیگر دارند، اما انسان‌ها همه تصاویر را به‌عنوان عدد ۳ در نظر می‌گیرند.

میزان روشنایی، تغییرات کوچک در اندازه تصویر، چرخش‌های ناچیز، تاری اندک و از این قبیل موارد، عواملی هستند که معمولا در مورد مقایسه بین دو یا چند تصویر توسط انسان نادیده گرفته می‌شوند و دسته‌ای از دیگر عوامل نیز بسته به کاربرد مورد نظر می‌توانند مهم باشند و یا اهمیت چندانی نداشته باشند.

برای اینکه مقدار حاصل از تابع تصمیم تا اندازه قابل توجهی بزرگتر از صفر باشد باید تابع شباهت یا تابع هسته $k(x_i, x)$ برای تعداد نه چندان کمی از نمونه‌های آموزشی x_i به اندازه کافی بزرگ باشد. در غیر این صورت $f(x) \approx ۰$ بوده و به طور شماتیک در این حالت نمونه ارزیابی بسیار نزدیک به ابر صفحه تصمیم قرار می‌گیرد و

نمی‌توان چندان به تابع تصمیم مطمئن و متکی بود. به‌عنوان مثال برای هسته شناخته شده‌ای مانند هسته گوسی، $k(x, x_i)$ تنها در صورتی که نمونه‌های x_i در نُرم اقلیدسی به اندازه کافی به نمونه x نزدیک (شبیه) باشند، تابع تصمیم به اندازه قابل توجهی بزرگتر از صفر خواهد بود. اما همانطور که در قسمت قبل بحث شد معیار فاصله نُرم اقلیدسی بین جفت پیکسل‌ها نمی‌تواند عملکرد مناسبی خصوصا در مسایل بینایی کامپیوتر داشته باشد. به همین دلیل روش‌های متعددی برای برطرف کردن این مشکل توسعه داده شده است که می‌توان آن‌ها را به چهار رویکرد اصلی دسته‌بندی کرد:

۱. بسط دادن مجموعه آموزشی

۲. نرمال‌سازی تصویر

۳. توابع هسته تغییرناپذیر

۴. نمایش تصاویر به صورت تغییرناپذیر

بسط دادن مجموعه آموزشی

همانطور که مطرح شد برای دستیابی به یک دسته‌بندی کننده مناسب و با توجه به معیارهای اندازه‌گیری استفاده شده، ملزم به داشتن یک مجموعه آموزشی غنی هستیم که نمونه‌های آن تا حدود زیادی به داده‌های آزمون نزدیک باشد. یکی از ساده‌ترین روش‌ها برای افزایش شانس تطابق هر کدام از نمونه‌های آزمون با یک یا چند نمونه آموزشی، استفاده از تعداد بیشتری از نمونه‌های آموزشی است تا تقریبا کلیه حالت‌های احتمالی داده‌های آزمون پوشش داده شوند. در عمل تعداد نمونه‌های آموزشی معمولا محدود هستند و حتی ممکن است افزایش میزان آن‌ها ممکن نباشد اما می‌توان با تکنیک دیگری به این رویکرد دست یافت. برای این کار می توان از نمونه‌های مصنوعی یا مجازی استفاده کرد. به صورتی که به ازای هر نمونه اصلی یک یا چند نمونه کپی شده وجود دارد که کمی دستکاری یا تحریف شده‌اند. از جمله مهمترین این تحریفات انتقال تصویر به طرفین، چرخاندن آن‌ها واعمال مقداری نوفه هستند که انتظار داریم سیستم طراحی شده نسبت به این انحرافات تاثیر ناپذیر باشد واین موارد تاثیری در تصمیم‌گیری آن نداشته باشند.

نقاط قوت: در این روش سیستم طراحی شده نسبت به برخی تغییرات ظاهری در تصاویر تاثیر ناپذیر خواهد بود.

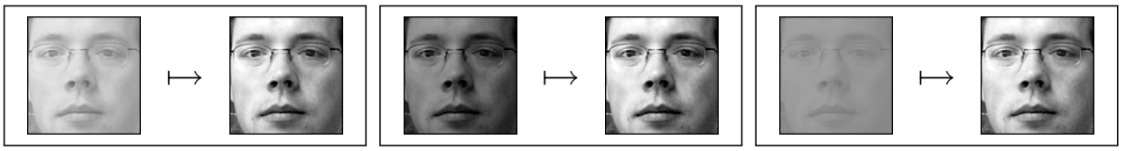
نقاط ضعف: برای این کار نیاز به سیستم تولید داده‌های مصنوعی است که باید به صورت دقیق عمل کند. همچنین افزایش تعداد نمونه‌های آموزشی منجر به افزایش زمان و حافظه مورد نیاز برای یادگیری است که می‌تواند نسبتا پرهزینه باشد.

نرمال‌سازی تصویر

بسیاری از خصوصیات تغییرپذیر موجود در تصاویر می‌توانند توسط نرمال‌سازی نمونه‌های آموزشی و همچنین نمونه‌های آزمون و قبل از اعمال هسته برطرف شوند. یک نمونه ساده از این حالت که قبلا نیز به آن اشاره شد، تغییر اندازه تمامی تصاویر به یک اندازه مشخص و ثابت است. نرمال‌سازی میزان روشنایی و شدت نور تصاویر نمونه‌های دیگری از این مورد هستند که می‌توانند به منظور کاهش تنوع داده‌ها و در نتیجه کاهش میزان اختلاف بین نمونه‌هایی که دارای برچسب‌های یکسان هستند استفاده شود. به‌عنوان نمونه با در نظر گرفتن رابطه

$$\hat{x} = \frac{x - x^{min}}{x^{max} - x^{min}}, \qquad (۴۳-۱)$$

که در آن x^{min} و x^{max} به ترتیب کمینه و بیشینه مقادیر خاکستری[1] در تصویر $x \in \mathbb{R}^{NM}$ هستند، مقادیر خاکستری تصویر $\hat{x}$ حاصل شده از این رابطه همواره در بازه $[۰, ۱]$ خواهد بود (شکل ۹.۱). همچنین در مورد تصاویر رنگی، نرمال‌سازی شدت نور می‌تواند برای هر کدام از کانال‌های رنگی (قرمز،سبز و آبی) به صورت مجزا انجام شود.



شکل ۹.۱: تصاویر سمت راست حاصل دگرگونی مقادیر خاکستری از تصاویر سمت چپ خود هستند. در تصاویر حاصل شده کمینه مقادیر خاکستری برابر ۰ و بیشینه آن‌ها برابر با ۱ است.

به منظور مقاوم کردن تصاویر کوچک و ناحیه‌های دیگر از تصاویر در مقابل تغییرپذیری‌های هندسی مانند انتقال و یا چرخش در تصاویر، می‌توان آن‌ها را به یک قالب یکپارچه مانند تطابق گشتاور[2] نرمال‌سازی کرد. برای این کار ابتدا باید گشتاور مرتبه اول و دوم برای تصویر x با پیکسل‌های $x[i, j]$ به صورت زیر محاسبه شوند:

$$m_x = \frac{۱}{m} \sum_{i,j} i x[i, j], \quad m_y = \frac{۱}{m} \sum_{i,j} j x[i, j], \qquad (۴۴-۱)$$

[1] Gray Value
[2] Moment Matching

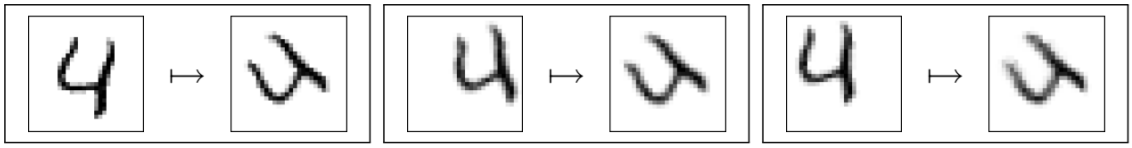
$$m_{xx} = \frac{1}{m}\sum_{i,j}(i - m_x)^2 x[i,j], \quad m_{yy} = \frac{1}{m}\sum_{i,j}(j - m_y)^2 x[i,j], \tag{۴۵-۱}$$

$$m_{xy} = \frac{1}{m}\sum_{i,j}(i - m_x)(j - m_y)x[i,j], \tag{۴۶-۱}$$

که در آن‌ها m برابر با $\sum_{i,j} x[i,j]$ است.

همانطور که واضح است انتقال تصویر بر روی مقادیر m_x و m_y تاثیرگذار خواهد بود و چرخش تصویر حول مرکز (m_x, m_y) مقادیر m_{xx}، m_{yy} و m_{xy} را تغییر خواهد داد.

شیفت دادن تصویر با در نظر گرفتن مقادیر $(-m_x, -m_y)$ و چرخش آن به صورت $\alpha = arctan(v_y/v_x)$ که در آن v اولین بردار ویژه ماتریس $M = [m_{xx}, m_{xy}; m_{xy}, m_{yy}]$ است، تصویر جدیدی ایجاد خواهد کرد که به صورت $m_x = m_y = m_{xy} = ۰$ و $m_{xx} = m_{yy} = ۱$ نرمال‌سازی شده است (شکل ۱۰.۱).



شکل ۱۰.۱: تصاویر سمت راست حاصل انتقال و چرخش تصاویر سمت چپ خود هستند به‌طوری که در آن‌ها $m_x = m_y = m_{xy} = ۰$ و $m_{xx} = m_{yy} = ۱$.

نقاط قوت: با نرمال‌سازی تصاویر، تغییرپذیری ناخواسته آن‌ها تا میزان زیادی کاهش می‌یابد. همچنین تشابه تصاویر مشابه هنگامی که به صورت برداری در نظر گرفته می‌شوند افزایش خواهد یافت.

نقاط ضعف: یافتن مولفه‌های فرآیند نرمال‌سازی برای هر تصویر ممکن است دشوار و غیر قابل اطمینان باشد. همچنین باید تکنیکی وجود داشته باشد تا نمونه‌هایی که به صورت اشتباه دستکاری شده‌اند را تصحیح، حذف و یا بی‌اثر کند.

توابع هسته تغییرناپذیر

به جای تخمین مولفه‌های نرمال‌سازی برای هر تصویر و سپس به دست آوردن مقادیر هسته، می‌توان نرمال‌سازی را به‌عنوان یک قسمت از خود تابع هسته در نظر گرفت. مزیت این روش این است که یک قاب مرجع و عمومی برای بقیه تصاویر نیاز نیست اما تمامی دگرگونی‌های مورد نیاز برای محاسبه فاصله بین جفت نمونه‌ها باید در یک زمان تخمین زده شود. به‌عنوان مثال به جای محاسبه هسته گوسی بر اساس فاصله اقلیدسی $||.||$، یک معیار اندازه‌گیری شباهت یا عدم شباهت بر اساس یافتن بهترین تطابق بین نمونه x و همه نسخه‌های تحریف شده x' تعریف می

شود:

$$d_{match}(x, x') = \min_{J \in \mathcal{J}} ||x - J(x')||^2, \quad (۴۷-۱)$$

به‌طوری که J شامل تمام عملگرهای تحریف مانند تمام چرخش‌ها و انتقال‌ها از مجموعه $\mathcal{J}$ می‌شود. $\mathcal{J}$ معمولا چند بُعدی است و بهینه‌سازی کاملا غیرخطی بر روی تمام عناصر در حالت عادی زمان زیادی را صرف خواهد کرد. لذا برای این حالت تقریب‌های خطی مانند فاصله تانژانت توسعه داده شدند:

$$d_{TD}(x, x') = \min_{\delta_1, \ldots, \delta_k \in \mathbb{R}} ||x - (x' + \delta_k t_k)||^2, \quad (۴۸-۱)$$

که در آن t_k اختلاف بین بردارهای بین x' و نسخه‌هایی از x' است که به‌طور بسیار جزئی تحریف شده‌اند. به‌عنوان مثال به اندازه یک پیکسل در راستای افقی یا عمودی جابه‌جا شده و یا در حد چند درجه چرخانده شده‌اند. با توجه به توابع فاصله d_{match} و d_{TD}، یکی از فرم‌های توابع هسته تغییرناپذیر به صورت زیر تعریف می‌شود:

$$k_{inv}(x, x') = e^{-\gamma[d(x,x')+d(x',x)]}. \quad (۴۹-۱)$$

بر این اساس یکی از روش‌هایی که می‌توان مقادیر هسته را نسبت به تحریفات تصاویر که از آن‌ها مطلع هستیم تغییرناپذیر کنیم این است که از میانگین‌گیری یا گرفتن انتگرال بر روی مجموعه تمام انتقالات ممکن استفاده کنیم. در این صورت می‌توان برای هر تصویر k تعریف کرد:

$$k_{int}(x, x') = \int_{J \in \mathcal{J}} \int_{J' \in \mathcal{J}} k\left(J(x), J'(x')\right). \quad (۵۰-۱)$$

در این صورت اگر مجموعه انتقالات $\mathcal{J}$ مناسب باشد، نتیجه توابع هسته k_{int} نسبت به هر تغییر احتمالی، تغییرناپذیر خواهد بود.

نقاط قوت: با نرمال‌سازی به صورت جفت جفت، می‌توان بدون داشتن یک قاب مرجع عمومی به تغییرناپذیری مطلوب دست یافت. همچنین با انتگرال‌گیری بر روی همه انتقالات، یک تغییرناپذیری کامل به دست خواهد آمد.

نقطه ضعف: هر ارزیابی‌کننده هسته به طور قابل ملاحظه‌ای کند خواهد شد و توابع فاصله‌ای که شامل فرآیندهای بهینه‌سازی هستند می‌توانند باعث پدید آمدن توابع هسته‌ای شوند که معین مثبت نیستند.

نمایش تصاویر به صورت تغییرناپذیر

همانطور که قبلا هم مطرح شد استفاده از تابع هسته $k : \chi \times \chi \to \mathbb{R}$ معادل با انتخاب یک تابع نگاشت $\varphi : \chi \to \mathcal{H}$ به صورت ضمنی است به‌طوری که χ مجموعه تمام تصاویر و $\mathcal{H}$ فضای هیلبرت معرفی شده توسط k است. برای

هسته‌های تغییرناپذیر، توابع نگاشت حاصل شده نسبت به تغییرناپذیر بودن تصاویر حساس نیست. در حقیقت در نرمال‌سازی تصاویر از یک گام پیش پردازش $\mathcal{N} : \chi \to \chi$ استفاده می‌شود که تاثیر تغییرات ناخواسته از تمامی تصاویر را قبل از اعمال φ حذف می‌کند. در این حالت می‌توان هر دو مفهوم را با استخراج ویژگی‌های تغییرناپذیر توسعه داد و سپس از هر هسته دلخواهی برای نمایش نتایج استفاده کرد. از نظر ریاضی هر کدام از این عملگرها صریحا متناظر با یک تابع نگاشت به صورت $\psi : \chi \to \tilde{\chi}$ هستند که از یک نگاشت ضمنی مانند $\varphi : \tilde{\chi} \to \mathcal{H}$ تبعیت می‌کنند که این نگاشت خود از تابع هسته $k : \tilde{\chi} \times \tilde{\chi} \to \mathbb{R}$ نتیجه شده است.

نقاط قوت: تابع نگاشت تغییرناپذیر این امکان را می‌دهد که بسیاری از تغییرات ناخواسته تصاویر به طور موثری حذف شوند.

نقاط ضعف: با اتخاذ نمایشی که نمونه‌ها در آن تا حدود زیادی یکپارچه و یک شکل هستند ممکن است منابع مهم تغییرات نیز به راحتی حذف شوند و از آنجایی که استخراج ویژگی همیشه در گام‌های نخستین انجام می‌شود، راهی برای بازگرداندن اطلاعات از دست رفته وجود ندارد.

۵.۶.۱ یادگیری هسته

تمامی روش‌هایی که تا به حال بررسی شدند تنها از یک تابع هسته استفاده می‌کردند. از طرف دیگر بهره‌گیری از توابع هسته گوناگون باعث استفاده از فضاهای ویژگی مختلفی خواهد شد به‌طوری که هر یک از آن‌ها از جنبه‌های مختلفی برای مسئله مورد نظر مناسب هستند. بر این اساس طبیعی است که سوال شود کدام یک از این‌ها بهترین تابع هسته است؟ در نهایت پاسخ این سوال وابسته به مسئله مورد نظر است؛ زیرا کیفیت یک هسته آموزش دیده بستگی به این دارد که عملکرد آن تا چه اندازه در عمل مناسب است. در برخی مواقع برای کشف این که کدامیک از هسته‌ها یا مولفه‌ها به صورت بهتری عمل می‌کنند، از یک رویکرد الگوریتمی برای این کار استفاده می‌شود. برای هسته‌هایی که از خانواده هسته‌های دارای مولفه هستند، (مانند هسته گوسی) با استفاده از قابل تنظیم بودن دامنه مولفه‌ها و معیار تراز هدف هسته، روشی برای تعیین مقادیر مناسب برای مولفه‌ها وجود خواهد داشت.

پس از آن اگر یک مجموعه متناهی از هسته‌ها مثلا توسط یک فرد خبره در نظر گرفته شده باشد و هدف انتخاب یک زیر مجموعه مناسب از آن و یک مجموعه وزن برای استفاده از یک ترکیب خطی از هسته‌ها باشد، یادگیری چند هسته‌ای پاسخ خوبی برای این سوال خواهد بود.

مسئله بعدی انتخاب ویژگی است که یک مورد ویژه در این خصوص محسوب می‌شود. در روش‌های مبتنی بر

هسته هر هسته پایه تنها به یکی از چند ویژگی موجود بستگی دارد. برای سادگی در بحث، مسائل مورد نظر را به مسئله دسته‌بندی محدود خواهیم کرد.

۶.۶.۱ تراز هدف هسته

ایده تراز هدف هسته[1] [۳۰] این است که مقادیر هسته مناسبی مانند $k : \chi \times \chi \to \mathbb{R}$ بایستی تا حدودی شبیه به هسته کامل فرضی $l : \chi \times \chi \to \mathbb{R}$ باشد که به برچسب‌های نمونه‌ها دسترسی دارد. به عبارت دیگر برای هر نمونه x و x' که دارای برچسب یکسانی هستند $l(x, x') = ۱$ و در غیر این صورت $l(x, x') = -۱$ باشد. برای اندازه‌گیری اینکه هسته k تا چه اندازه به هسته بهینه l نزدیک و شبیه است، معیار امتیاز تراز هسته به صورت زیر تعریف می‌شود:

$$A(Z; k, l) = \frac{trace(KL)}{n\sqrt{traceKK}}, \qquad (۵۱-۱)$$

که در آن $Z = \{(x_۱, y_۱), \ldots, (x_n, y_n)\}$ یک مجموعه آموزشی برچسب گذاری شده است. K ماتریس هسته k بر روی مجموعه Z بوده و L نیز ماتریس هسته متناظر با برچسب‌های موجود است. برای انتخاب یک هسته مناسب از بین سایر هسته‌های دیگر کافی است گزینه‌ای که باعث بیشینه شدن مقدار $A(Z; k, l)$ می‌شود را انتخاب کرد. از آنجایی که این فرآیند نیازی به آموزش و ارزیابی دسته‌بندی کننده ندارد، سریع‌تر از روش ارزیابی متقابل است. مزیت دیگر استفاده از رابطه (۱-۵۱)، مشتق‌پذیری آن با توجه به تابع هسته k است.

۷.۶.۱ یادگیری چند هسته‌ای

همانطور که عنوان شد مهمترین بخش در مسائل مبتنی بر هسته، انتخاب هسته مناسب است. اهمیت این موضوع به این دلیل است که انتخاب یک هسته مناسب می‌تواند نتایج فوق‌العاده‌ای را در بر داشته باشد در حالی که استفاده از یک هسته نامناسب ممکن است منجر به نتایج غیربهینه و غیرقابل قبولی شود. به طور کلی انتخاب هسته مناسب فرآیند نسبتا دشواری تلقی می‌شود؛ زیرا به اطلاعات خاصی از فضای ورودی نیازمند است و این در حالی است که معمولا دست‌یابی به این اطلاعات مقدور نیست.

در بینایی کامپیوتر این موضوع معمولا به نحوه نمایش و ویژگی‌های موجود مربوط می‌شود. بسته به کابردی که مدنظر است، رنگ، بافت، یال‌ها و دیگر ویژگی‌های شبیه به آن می‌تواند بیشترین اهمیت را داشته باشد. با این

[1] Kernel Target Alignment (KTA)

وجود اما در اکثر موارد جنبه‌های مختلف به طور هم‌زمان از اهمیت بالایی برخوردار هستند و لذا نیازمند آن هستیم تا چندین ویژگی مورد نظر را در عین واحد دخیل کنیم. یکی از رویکردهایی که تا کنون برای حل این مشکل مطرح شده و نتایج خوبی را نیز در پی داشته، استفاده از چند هسته است. در این روش به جای انتخاب یک هسته، از مجموعه‌ای از هسته‌ها استفاده می‌شود به‌طوری که نتایج حاصل از هر یک از آن‌ها با یکدیگر ترکیب می‌شوند. سپس با انجام یک فاز بهینه‌سازی برای این ترکیبات می‌توان ضرایب هسته‌های نامناسب را کوچک کرد که این کار معادل با حذف نسبی هسته‌های نامناسب است. به این ترتیب فرآیندی نیاز است که طی آن هسته‌های مناسب انتخاب و ترکیب شوند و سپس ترکیب به دست آمده بهینه گردد. این فرآیند معمولا با نام یادگیری چند هسته‌ای[1] شناخته می‌شود که مشابه با روش‌های مبتنی بر هسته، مهم‌ترین بخش از مسائل مبتنی بر چند هسته محسوب می‌شود.

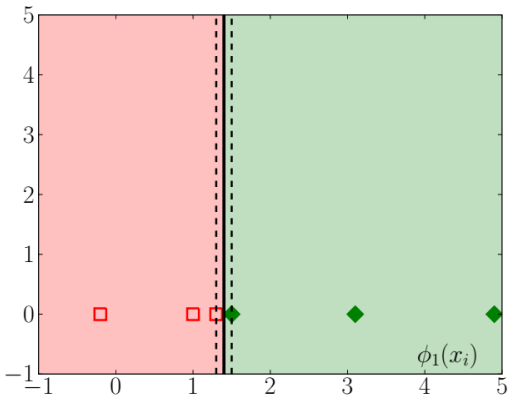
ایده استفاده از یادگیری چند هسته‌ای اولین بار در سال ۲۰۰۴ مطرح شد [۸۰]. از آن پس اغلب مطالعاتی که در این زمینه انجام می‌شود را می‌توان در دو دسته تقسیم‌بندی کرد. در دسته اول مطالعات تلاش می‌شود تا دقت طبقه‌بندی با یافتن فرمول‌بندی و روابط جدید برای هسته‌ها افزایش یابد و در مطالعات دسته دوم در مورد استفاده از روش‌های بهینه‌سازی برای مسئله یادگیری چند هسته‌ای تحقیق و پژوهش می‌شود. به طور کلی روش‌هایی که برای ترکیب هسته‌ها وجود دارد را می‌تواند به مواردی مانند قوانین ثابت، قوانین مکاشفه‌ای، روش‌های مبتنی بر بهینه‌سازی، روش‌های بیزی و روش‌های مبتنی بر Boosting تقسیم‌بندی کرد. در مورد بهبود دقت یادگیری در الگوریتم‌های چندهسته‌ای روش‌های گوناگونی ارائه شده است که اغلب بر اساس مسائل بهینه‌سازی درجه اول انجام می‌شوند [۱۱۸، ۵، ۷۷]. علاوه بر این در برخی مطالعات از روش تندترین شیب استفاده شده و ادعا می‌شود که این مورد روش مناسب‌تری است [۱۳۲].

۸.۶.۱ ترکیب خطی هسته‌ها

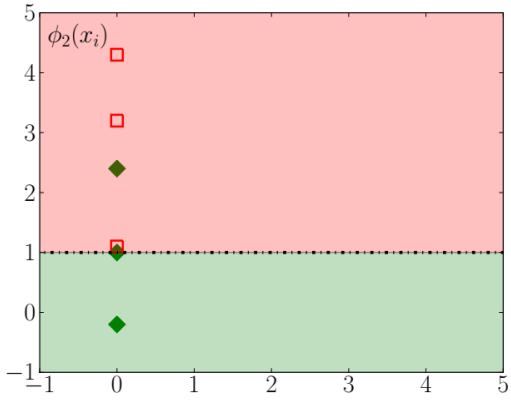
فرض کنید $k_1, \ldots, k_m$ توابع هسته باشند به‌طوری که $k_j : \chi \times \chi \to \mathbb{R}$ و $\mathcal{H}$ فضای هیلبرت متناظر بوده و توابع نگاشت $\varphi_j : \chi \to \mathcal{H}_j$ موجود باشند. در این صورت به دنبال یافتن وزن‌های ترکیب $\beta_j \geq 0$ که $j = 1, \ldots, m$ هستیم به‌طوری که این وزن‌ها منجر به تولید دسته‌بندی کننده SVM با یک هسته کلی به‌فرم زیر است:

$$k(x, x') = \sum_{j=1}^{m} \beta_j k_j(x, x'). \qquad (۵۲-۱)$$

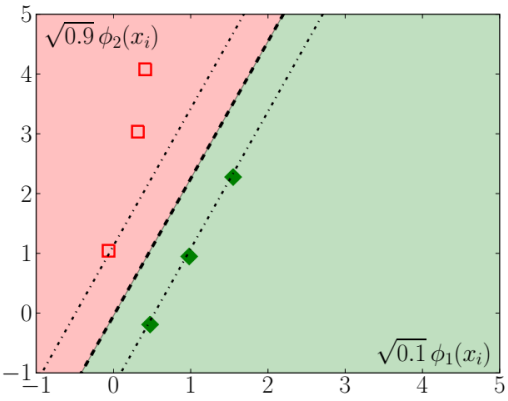
[1] Multi Kernel Learning (MKL)



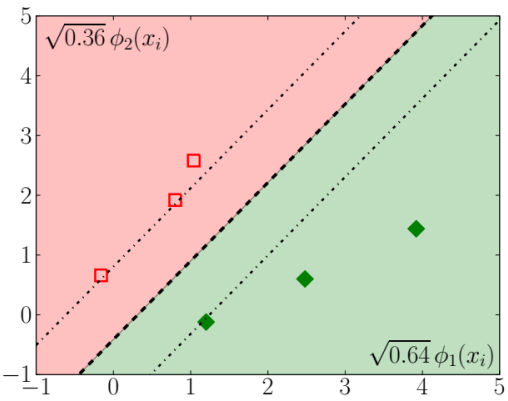
(آ) با استفاده از نمایش $\varphi_1(x_i)$ مسئله دسته‌بندی با یک حاشیه بسیار کوچک قابل حل است.



(ب) با استفاده از نمایش $\varphi_2(x_i)$ حتی تفکیک نمونه‌ها نیز به سختی قابل انجام است.



(ج) نمایش داده‌ها با استفاده از ترکیب $\varphi_1(x_i)$ و $\varphi_2(x_i)$ منجر به تفکیک پذیری آسان‌تر شده است.



(د) دخیل کردن ضریب $\beta = 0.64$ باعث به وجود آمدن حاشیه حداکثری تا حد ممکن شده است.

شکل ۱۱.۱: در شکل‌های (آ) و (ب) هسته‌های k_1 و k_2 با توابع نگاشت φ_1 و φ_2 هیچکدام برای حل مسئله دسته‌بندی مناسب نیستند. در شکل (ج) ترکیب خطی هسته $k(x,x') = \beta k_1(x,x') + (1-\beta)k_2(x,x')$ و تابع نگاشت $(\sqrt{\beta}\varphi_1, \sqrt{1-\beta}\varphi_2)$ نتیجه بهتری را می‌تواند در پی داشته باشد. در شکل (د) با حل مسئله بهینه‌سازی ۱-۵۴ می‌توان ضرایب بهینه ترکیب خطی را محاسبه کرد.

برای ضرایب ثابت $\beta_1, \ldots, \beta_m$ می‌توان k را با استفاده از توابع نگاشت $\varphi(x) = (\sqrt{\beta_1}\varphi_1(x), \ldots, \sqrt{\beta_m}\varphi_m(x))$ و به‌عنوان نتیجه ضرب فضای هیلبرت به صورت $\mathcal{H} = \mathcal{H}_1 \times \ldots \times \mathcal{H}_m$ در نظر گرفت؛ زیرا حاصل ضرب اسکالر به دست آمده از این سازوکار مقداری برابر با k دارد:

$$\langle \varphi(x), \varphi(x') \rangle_{\mathcal{H}} = \sum_{j=1}^{m} \beta_j \langle \varphi_j(x), \varphi_j(x') \rangle_{\mathcal{H}_j} = k(x,x'). \qquad (۱-۵۳)$$

حال با تجزیه و مولفه‌گذاری مجدد بردار وزن به صورت $(\frac{۱}{\sqrt{\beta_۱}}v_۱, \ldots, \frac{۱}{\sqrt{\beta_m}}v_m)$ ، تابع هدف (۱-۲۳) به صورت زیر تبدیل می‌شود:

$$\min_{\substack{(v_۱,\ldots,v_m)\in\mathcal{H}_۱\times\ldots\times\mathcal{H}_m \\ \xi_۱,\ldots,\xi_n\in\mathbb{R}^+}} \sum_{j=۱}^{m} \frac{۱}{\beta_j}||v_j||_{\mathcal{H}_j^۲} + \frac{C}{n}\sum_{i=۱}^{n}\xi_i \qquad (۱-۵۴)$$

به‌طوری که

$$y_i \sum_{j=۱}^{m} \langle v_j, \varphi(x_i)\rangle_{\mathcal{H}_j} \geq ۱ - \xi_i, \quad \forall i = ۱, \ldots, n. \qquad (۱-۵۵)$$

به منظور یافتن بهترین ضریب‌ها برای ترکیب خطی هسته‌ها، عبارت (۱-۵۴) را بر روی تمامی هسته‌های انتخابی $(\beta_۱, \ldots, \beta_m)$ بهینه می‌کنیم. تغییر β نه تنها در میزان حاشیه، بلکه از طریق تابع نگاشت φ بر روی توزیع داده‌ها در فضای ویژگی نیز تاثیرگذار است. ضرب k در یک عدد ثابت مقدار حاشیه را افزایش خواهد داد اما علاوه بر آن مختصات نقاط را نیز مقیاس‌بندی می‌کند که از لحاظ هندسی تاثیری نخواهد داشت. برای جلوگیری از این رفتار، نُرم β را به صورت $||\beta||_L^۱ = ۱$ یا $||\beta||_L^۲ = ۱$ محدود می‌کنیم. مشکل بهینه‌سازی به وجود آمده این است که بردارهای v_j و همچنین هسته‌های β_j به طور مشترک محدب هستند. بنابراین تنها یک کمینه سراسری منحصر به فرد وجود دارد که با استفاده از تکنیک‌های استاندارد بهینه‌سازی می‌توان آن را به دست آورد. نکته‌ای که باید به آن توجه شود این است که در مورد محدودیت $L^۱$ که در آن $\sum_{j=۱}^{m}\beta_j = ۱$، ضرایب β_j معمولا تُنُک خواهند بود یا به عبارت دیگر بسیاری از آن‌ها مقدار صفر خواهند داشت. همین موضوع باعث تمایز بین هسته‌های مناسب و نامناسب می‌گردد و همچنین باعث می‌شود از یادگیری چندهسته‌ای به‌عنوان یک تکنیک انتخاب ویژگی نیز تفسیر شود.

در چند سال اخیر مبحثی با عنوان *یادگیری بی‌نهایت هسته‌ای*[1] نیز مطرح شده است که در آن مزایای *تراز هدف* هسته و *یادگیری چند هسته‌ای* با یکدیگر ترکیب می‌شوند. این تکنیک این امکان را فراهم می‌آورد تا یادگیری یک ترکیب خطی از هسته‌ها و تنظیم مقادیر پیوسته در مولفه‌های هسته‌ها به طور هم‌زمان انجام پذیرد.

در موضوع دسته‌بندی تصاویر در حوزه بینایی ماشین، پس‌زمینه و همچنین حضور یا عدم حضور اشیاء کلاس‌های دیگر از اهمیت بالایی برخوردار هستند. لمپرت و همکارانش در سال ۲۰۰۸ یک روش مبتنی بر یادگیری چند هسته‌ای برای استفاده از این موضوع را ارائه دادند [۷۹]. آن‌ها برای اشیاء متعلق به هر کلاس و همچنین برای اشیاء پس‌زمینه یک هسته پایه تعریف کردند و یادگیری چند هسته‌ای را برای یادگیری یک ترکیب خطی از این هسته‌های پایه و

[1] Infinite Kernel Learning

برای هر کلاس به کار بردند. در این مطالعه نویسندگان نشان می‌دهند که ترکیب هسته‌هایی که از اطلاعات اشیاء سایر کلاس‌ها نیز استفاده می‌کنند، عملکرد بهتری نسبت به هسته‌هایی دارند که تنها از اطلاعات یک شئ و ناحیه مربوط به آن استفاده می‌کنند. علاوه بر آن ضرایبی که توسط یادگیری چند هسته‌ای به دست آمده‌اند نشان‌دهنده وابستگی ذاتی بین اشیائی هستند که با یکدیگر روابط معنایی دارند.

۷.۱ خوشه‌بندی

به طور کلی خوشه‌بندی[1] یکی از مسائل مهم در حوزه یادگیری ماشین محسوب می‌شود. این مسئله یک مسئله یادگیری بدون ناظر است که در آن داده‌های مورد نظر به مجموعه‌ای از گروه‌ها تقسیم‌بندی می‌شوند. مسائل خوشه‌بندی را می‌توان به دو دسته خوشه‌بندی سخت[2] و خوشه‌بندی نرم[3] تقسیم‌بندی نمود. در خوشه‌بندی سخت هر یک از داده‌ها تنها می‌تواند در یکی از خوشه‌ها قرار گیرد اما در خوشه‌بندی نرم که با نام خوشه‌بندی فازی[4] نیز شناخته می‌شود، به جای اینکه هر یک از داده‌ها تنها متعلق به یک خوشه باشد، هر داده می‌تواند با یک میزان احتمال مشخص به بیش از یک خوشه یا حتی تمامی خوشه‌ها تعلق داشته باشد.

۱.۷.۱ الگوریتم‌های خوشه‌بندی و انواع آن‌ها

تاکنون الگوریتم‌های متنوعی برای حل مسائل خوشه‌بندی ارائه شده است اما فرض مهم و کلیدی در تمامی این الگوریتم‌ها این است که داده‌هایی که در فضای ویژگی هم‌جوار هستند یا به عبارت دیگر شباهت بیش‌تری باهم دارند در یک خوشه قرار می‌گیرند. همانطور که واضح است می‌توان معیارهای متفاوتی برای تعیین میزان شباهت[5] داده‌ها به یکدیگر تعریف و استفاده نمود که بر همین اساس می‌توان الگوریتم‌های موجود برای خوشه‌بندی را به چند دسته تقسیم‌بندی نمود.

۱. مدل‌های اتصالی[6]:

همانطور که از نام این مدل مشخص است، در این مدل‌ها فرض می‌شود که داده‌های موجود در فضای داده‌ای، هرچقدر به یکدیگر نزدیک‌تر باشند یا به عبارت دیگر فاصله بین آن‌ها کم‌تر باشد، شباهت بیش‌تری

[1] Clustering
[2] Hard Clustering
[3] Soft Clustering
[4] Fuzzy Clustering (Fuzzy C-Mean)
[5] Similarity
[6] Connectivity Models

به یکدیگر دارند. در این مدل معمولا دو رویکرد اتخاذ می‌شود. در رویکرد اول ابتدا هر یک از داده‌ها به‌عنوان یک خوشه‌مجزا در نظر گرفته می‌شود و سپس بر اساس میزان فاصله بین آن‌ها و سایر داده‌ها، خوشه‌ها با یکدیگر ادغام می‌شوند و در نهایت تعداد خوشه‌ها به تعدادی مشخصی کاهش می‌یابد. در رویکرد دوم تمامی داده‌ها در یک خوشه قرار می‌گیرند و سپس مجددا بر اساس میزان فاصله بین آن‌ها و سایر داده‌ها، این خوشه به خوشه‌های دیگری تقسیم‌بندی می‌گردد. در این مدل تابع و معیار اندازه‌گیری فاصله[1] نیز می‌تواند بر اساس کاربرد و توسط کاربر مشخص شود اما از معیارهای محبوب و ساده در این گونه روش‌ها معیار فاصله اقلیدسی[2] و فاصله منهتن[3] است. این نوع مدل‌ها از نظر تفسیری بسیار ساده هستند و همچنین از پیچیدگی کمی برخوردار هستند اما برای مجموعه داده‌های بزرگ و حجیم مقیاس ناپذیر بوده و معمولا هزینه محاسباتی بالایی دارند. به‌عنوان یک الگوریتم معروف در این مدل می‌توان به الگوریتم خوشه‌بندی سلسله مراتبی[4] و انواع آن‌ها اشاره نمود.

۲. مدل‌های نقطه مرکزی[5]:

این مدل شامل الگوریتم‌های خوشه‌بندی تکراری است به‌طوری که به صورت تکراری اجرا می‌شوند تا به یک جواب بهینه محلی برسند و میزان شباهت داده‌ها در این مدل بر اساس نزدیکی آن‌ها به یک نقطه به‌عنوان مرکز خوشه‌ها در نظر گرفته می‌شود و الگوریتم شناخته شده k نزدیک‌ترین همسایه[6] در واقع نمونه‌ای از این الگوریتم‌ها است. در این مدل‌ها همچنین تعداد خوشه‌ها باید از قبل تعیین گردد که همین امر داشتن یک دانش قبلی در مورد داده‌ها را می‌طلبد و به‌عنوان یک پارامتر مهم شناخته می‌شود.

۳. مدل‌های توزیعی[7]:

این مدل‌های خوشه‌بندی مبتنی بر این مفهوم هستند که چقدر ممکن است که تمامی نقاط داده شده در خوشه به یک توزیع آماری مانند توزیع نرمال یا توزیع گوسین متعلق باشند. این مدل‌ها ممکن است با مواردی مانند بیش برازش سروکار داشته باشند و یکی از الگوریتم‌های شناخته‌شده در این مدل الگوریتم بیشینه‌سازی انتظار[8] است که از توزیع‌های نرمال چند متغیره استفاده می‌کند.

[1] Distance Measure
[2] Euclidean Distance
[3] Manhattan
[4] Hierarchial Clustering
[5] Centroid Models
[6] K-means Clustering
[7] Distribution Models
[8] Expection-maximization

۴. مدل‌های تراکمی[1]:

این مدل‌ها فضای داده‌ای را با هدف یافتن تراکم‌های متعدد از نقاط جستجو می‌کنند و نواحی یافته‌شده را جدا کرده و نقاط موجود در آن‌ها را به‌عنوان یک خوشه مشترک در نظر می‌گیرد. از الگوریتم‌های پر استفاده برای این مدل از خوشه‌بندی‌ها می‌توان به مواردی مانند Mean Shift، OPTICS[2] و DBSCAN[3] اشاره کرد.

۵. مدل طیفی[4]:

یکی دیگر از روش‌های خوشه‌بندی، خوشه‌بندی طیفی[5] نام دارد که در سال ۲۰۰۰ توسط جیانبو و همکارانش معرفی شد و به خاطر عملکرد مناسب و همچنین پیاده‌سازی ساده آن مورد توجه بسیاری از محققین قرار گرفت. این الگوریتم مبتنی بر تئوری گراف طراحی شده است و رویکرد آن شناخت اجتماع رئوس گراف بر اساس یال‌های اتصالی آن‌ها است. البته این نوع خوشه‌بندی از انعطاف‌پذیری بالایی برخوردار است و می‌توان برای سایر مسائل و مسائل مستقل از گراف نیز از آن استفاده کرد. برای درک بهتر نحوه عملکرد این روش، فرض کنید که A ماتریس مجاورت گراف داده شده و W ماتریس وزن حاصل از یال‌های آن باشد. اگر D یک ماتریس قطری باشد که درجه هر یک از رئوس گراف را می‌دهد و $L = D - W$ ماتریس لاپلاس این گراف باشد آنگاه گام بعدی در این الگوریتم محاسبه بردارهای ویژه ماتریس L که در واقع بردارهای ویژگی داده‌ها محسوب می‌شوند و در گام آخر اجرای الگوریتم k نزدیک‌ترین همسایه بر روی این بردارهای ویژگی است. یکی از کاربردهای این تکنیک استفاده از آن برای حل مسئله تقسیم‌بندی تصاویر است.

۲.۷.۱ خوشه‌بندی تصاویر

در بسیاری از موارد داده‌هایی که نیاز است بر روی آن‌ها پردازش انجام شود تنها به نقاط داده شده در فضا خلاصه نمی‌شوند. یک نمونه دیگر از این داده‌ها که امروزه به‌عنوان یک نوع داده مهم شناخته و اطلاعات زیادی را شامل می‌شوند، تصاویر و فایل‌های ویدئویی ضبط شده از انواع دوربین‌های در حال استفاده هستند. مشخص است که با رشد بسیار زیاد دوربین‌ها و به تبع آن تعداد تصاویر و فایل‌های ویدئویی، همواره نمی‌توان بسیاری از آن‌ها را برچسب‌گذاری کرد و نیاز است تا به شیوه بدون ناظر پردازش شوند. در این راستا خوشه‌بندی تصاویر اولین

[1] Density Models
[2] Ordering points To Identify the Clustering Structures
[3] Density-Based Spatial Clustering of Applications with Noise
[4] Spectral Model
[5] Spectral Clustering

راهکاری است که به ذهن می‌رسد و می‌توان از آن استفاده کرد و اطلاعات مهم و حائز اهمیتی را از تصاویر در اختیار کسب نمود. علاوه بر این با استفاده از تکنیک‌های مربوط به خوشه‌بندی تصاویر می‌توان در مسائل دیگری از زمینه بینایی ماشین مانند بازیابی تصاویر[1] و تقسیم‌بندی تصاویر[2] نیز استفاده کرد که خود کاربردهای بسیار متنوع و مهمی دارند.

استخراج ویژگی برای خوشه‌بندی تصاویر

تکنیک‌های خوشه‌بندی تصاویر به طور کلی دارای دو فرآیند اساسی هستند. در فرآیند اول با استفاده از ویژگی‌های استخراج شده از تصاویر، میزان شباهت بین آن‌ها بررسی شده و در فرآیند دوم الگوریتم‌های خوشه‌بندی بر روی آن‌ها اعمال می‌شود. دلیل اینکه معیار تشابه بر اساس ویژگی‌های استخراج شده از تصاویر به دست می‌آید این است که معمولا تصاویر ابعاد زیادی دارند و هزینه محاسباتی بسیار زیادی برای به دست آوردن معیار تشابه بر اساس آن‌ها تحمیل خواهد شد. علاوه بر آن اعمال معیارهایی همانند فاصله اقلیدسی یا فاصله منهتن بر روی تصاویر خام برای به دست آوردن میزان تشابه چندان منطقی نیست؛ زیرا با این معیارها تصاویر شبیه به هم نیز ممکن است اختلاف نسبتا زیادی با یکدیگر داشته باشند. همچنین استخراج ویژگی‌های مناسب برای به دست آوردن معیار تشابه بسیار حائز اهمیت است به‌طوری که بر اساس این ویژگی‌ها باید بتوان معیار تشابه‌های صحیحی را به دست آورد و با استفاده از آن‌ها عملیات خوشه‌بندی کارآمدتری را ارائه داد. همانطور که در قسمت‌های قبل مطرح شد روش‌های متنوعی برای استخراج ویژگی از تصاویر وجود دارد. از آنجایی که معیار تشابه یک رکن اساسی در خوشه‌بندی تلقی می‌شود، در خوشه‌بندی تصاویر ویژگی‌های مهم و تمایز کننده به‌عنوان یک نماینده از خود تصویر شناخته می‌شوند. همانند مباحث گفته شده برای شناسایی اشیاء در تصاویر، ویژگی‌های استخراج شده برای خوشه‌بندی تصاویر نیز می‌تواند شامل ویژگی‌های سطح پایین مانند رنگ، بافت، و شکل ظاهری آن‌ها باشد و یا شامل ویژگی‌های معنایی سطح بالا باشد. ویژگی رنگ در مسائل خوشه‌بندی به‌عنوان یک ویژگی پایدارتر و مقاوم‌تر نسبت به ویژگی‌هایی مانند بافت و شکل ظاهری اشیاء شناخته می‌شود. همچنین برای استخراج این ویژگی از روش بسیار رایج هیستوگرام رنگی استفاده می‌شود که نسبت به چرخش، تغییر اندازه و انواع تبدیلات تغییر ناپذیر است. البته این روش برای تصاویر مشابه که از نماهای مختلف تهیه شده‌اند و یا دچار تغییرات نور هستند، هیستوگرام‌های متفاوتی تولید می‌کند که البته می‌توان این مشکل را با استفاده از فضاهای رنگی[3] مختلف

[1] Image Retrieval
[2] Image Segmentation
[3] Color Space

تا حدودی برطرف ساخت. منظور از فضای رنگی در واقع یک مدل ریاضی است که با استفاده از آن رنگ‌ها توصیف و نشان داده می‌شوند که از معروف‌ترین آن‌ها مدل‌های RGB[1] و CMYK[2] است. علاوه بر رنگ، از ویژگی بافت نیز در بسیاری از موارد برای پردازش تصاویر دیجیتال توسط محققین استفاده می‌شود. اما از آنجا که بافت یک ویژگی مکانی است، هیستوگرام یک بُعدی قادر به توصیف آن نیست. از این رو یم ماتریس دو بُعدی برای تحلیل آن استفاده می‌شود. همچنین تکنیک‌های دیگری مانند تبدیل موجک [3] و فیلتر گابور [4] نیز استفاده می‌شود. تکنیک‌های مختلفی برای استخراج ویژگی بافت وجود دارد که به سه رویکرد آماری[5]، ساختاری[6] و طیفی[7] تقسیم‌بندی می‌شوند. رویکرد آماری ساده و مقاوم است و در مقایسه با سایر رویکردها از پیاده‌سازی آسان‌تری برخوردار است اما در مواجهه با تعداد داده‌های زیاد و حجیم توانایی کارایی مناسبی ندارد. رویکرد ساختاری برای تحلیل و استخراج بافت‌های مصنوعی مفید است؛ زیرا بر اساس مجموعه‌ای از عناصر بافت اولیه عمل می‌کند. همچنین این رویکرد نمایانگر ساختار سلسله مراتبی بافت‌ها یا به عبارت دیگر سلسله مراتب ترتیب‌های مکانی عناصر بافت اولیه است و یک توصیف نمادین خوبی از تصویر ارائه می‌دهد. در روش طیفی از بانک‌های فیلتر و یا هرم‌های تصویر برای تبدیل تصویر از دامنه فضایی به دامنه فرکانسی آن و بالعکس استفاده می‌شود.

[1] Red, Green, Blue
[2] Cyan, Magenta, Yellow, Key(Black)
[3] Wavelet Transform
[4] Gabor filter
[5] Statistical
[6] Structural
[7] Spectral

فصل ۲

شناسایی اشیاء به شیوه نوین

۱.۲ مقدمه

در روش‌های کلاسیک موجود برای شناسایی اشیاء دیدیم که ویژگی‌های تصاویر به چه صورت استخراج می‌شدند و سپس توسط الگوریتم‌های دسته‌بندی، شناسایی اشیاء انجام می‌گرفت. مسئله‌ای که وجود دارد این است که این روش‌ها در مواردی که تصاویر با چالش‌های متعددی مانند تغییرات روشنایی، تغییرات اندازه و مقیاس و همچنین انسداد قسمتی از اشیاء مواجه می‌شوند، قادر به ارائه عملکرد مناسبی از خود نیستند. علاوه بر آن هنگامی که در تصاویر مورد نظر چند دسته شئ مختلف وجود دارد، انتخاب و استخراج ویژگی‌هایی که در مورد همه آن‌ها عملکرد قابل قبولی داشته باشند به سادگی امکان‌پذیر نیست [۱۴۷]. به همین جهت محققان همواره به دنبال پیشنهاد روش‌های جدیدی بودند که بتوانند با شرایط ذکر شده به عملکرد بهتری نسبت به روش‌های موجود دست یابند.

با مطرح شدن مباحث یادگیری عمیق[۲] در سال ۲۰۰۶ و به واسطه موفقیت در مطالعاتی همچون بازشناسی گفتار[۳] بسیاری از محققان در تمامی حوزه‌ها و به خصوص دانشمندان بینایی کامپیوتر بر آن شدند تا مطالعات بیشتری در این زمینه انجام دهند [۵۷، ۵۶]. یادگیری عمیق در حقیقت یک شاخه از حوزه یادگیری ماشین[۴] و بر اساس مجموعه‌ای از الگوریتم‌ها است به‌طوری که در تلاش هستند تا مفاهیم انتزاعی سطح بالا را در داده‌های موجود مدل نمایند [۱۱]. با این کار ماشین‌ها درک بهتری از واقعیت وجودی داده‌ها پیدا کرده و می‌توانند الگوهای مختلف و مفید را بهتر شناسایی کنند.

اولین نمونه‌های استفاده از یادگیری عمیق در حوزه بینایی کامپیوتر را می‌توان شبکه‌ای تحت عنوان AlexNet

[2] Deep Learning
[3] Speech Recognition
[4] Machine Learning

دانست که در مسابقه ILSVRC[1] در سال ۲۰۱۲ توسط کریژِوسکی و هینتون معرفی شد و در مورد دسته‌بندی تصاویر دارای عملکرد بسیار خوبی نسبت به روش‌های قبل از خود بود [۷۵]. این روش را می‌توان به عنوان نقطه عطفی برای روش‌های نوین در بینایی کامپیوتر به شمار آورد؛ زیرا در سال ۲۰۱۲ AlexNet تنها روش ارائه شده در این مسابقه بود که بر اساس شبکه‌های عمیق عمل می‌کرد. اما این روش باعث ایجاد تمایل شدید محققان به استفاده از یادگیری عمیق شد تا جایی که در سال بعد تمامی روش‌های برنده شده در این مسابقه بر اساس یادگیری عمیق بودند.

به این ترتیب اولین شناسایی کننده اشیاء که مبتنی بر یادگیری عمیق بود توسط سرمننت و همکارانش در سال ۲۰۱۳ طراحی شد [۱۲۸]. در این روش از شبکه‌های تلفیقی[2] به همراه پنجره لغزان[3] استفاده می‌شد که در آن هر قسمت از تصویر که در ناحیه آن پنجره قرار می‌گرفت دسته‌بندی می‌شد. پس از آن شبکه‌های دیگری با معماری‌ها و عملکردهای مختلف برای شناسایی اشیاء ارائه شدند. در ادامه به معرفی شبکه‌های تلفیقی و بررسی چند نمونه از شناسایی کننده‌های اشیاء که بر اساس شبکه‌های تلفیقی عمل می‌کنند پرداخته خواهد شد.

۲.۲ شبکه‌های عصبی تلفیقی

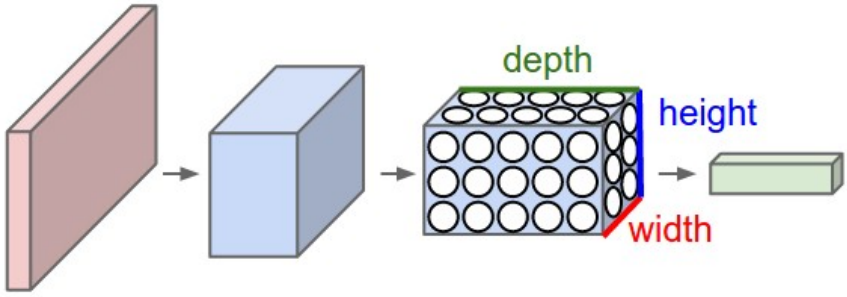
شبکه‌های عصبی تلفیقی که با نام‌های ConvNet یا به اختصار CNN نیز شناخته می‌شوند را می‌توان به عنوان نسخه توسعه یافته‌ای از شبکه‌های چندلایه پرسپترون[4] [۴۰] دانست که معمولا به صورت شبکه‌های تماما متصل نیز شناخته می‌شوند. شبکه‌های تلفیقی یکی از مهم‌ترین روش‌های یادگیری عمیق به شمار می‌آیند و از چند لایه برای استخراج ویژگی‌های سطح بالا از ورودی‌های خام و نحوه توزیع آن‌ها استفاده می‌کنند. در این شبکه‌ها فرض بر این است که داده‌های ورودی تصاویر هستند و همین مورد یکی از مهم‌ترین تفاوت‌های این نوع شبکه‌ها با شبکه‌های عصبی اولیه است.

این شبکه‌ها یک روش بسیار کارآمد و رایج در بسیاری از زمینه‌ها و به خصوص در حوزه بینایی کامپیوتر هستند که در سال‌های اخیر پیشرفت‌های چشم‌گیری را برای مسائل موجود در این حیطه رقم زده‌اند. از نمونه کاربردهای مهم آن‌ها در حوزه‌های دیگر می‌توان به سیستم‌های توصیه‌گر [۱۴۶] و پردازش زبان‌های طبیعی[5] [۲۵] اشاره کرد.

[1] ImageNet Large Scale Visual Recognition Challenge
[2] Convolutional Neural Networks (CNNs)
[3] Sliding Window
[4] Multilayer Perceptron (MLP)
[5] Natural Language Processing (NLP)

می‌دانیم که شبکه‌های عصبی از نرون‌های به هم متصل تشکیل شده‌اند که در سه لایه ورودی، لایه پنهان و لایه خروجی تقسیم‌بندی می‌شوند. در مواردی که ورودی شبکه یک تصویر است، شبکه‌های عصبی اولیه برای این نوع ورودی‌ها به خوبی مقیاس‌پذیر نیستند. به عنوان مثال یک تصویر با ابعاد $[۳ \times ۲۰۰ \times ۲۰۰]$ در نظر بگیرید که در آن عرض و ارتفاع تصویر دارای ۲۰۰ پیکسل بوده و همچنین دارای سه کانال رنگی قرمز، سبز و آبی است. بنابراین هر نرون در اولین لایه پنهان از شبکه عصبی دارای $۳ \times ۲۰۰ \times ۲۰۰ = ۱۲۰,۰۰۰$ تعداد وزن است. علاوه بر این در لایه‌های بعدی تعداد نرون‌های بیشتری وجود دارد که برای آن‌ها این تعداد بسیار بیش‌تر بوده و لذا سرعت رشد پارامترها برای تصویری با این مشخصات بسیار زیاد خواهد بود. مشخص است که در این حالت علاوه بر آن که احتمال بیش برازش شدن شبکه افزایش می‌یابد، همچنین ذخیره‌سازی و پردازش این تعداد از پارامترها نیازمند منابع سخت افزاری بسیار زیادی خواهد بود. در شبکه‌های تلفیقی برای جلوگیری از بیش برازش شبکه استفاده از تکنیک‌هایی مانند Dropout [۱۳۶] و Batch Normalization [۶۳] رایج است که از عملکرد خوبی نیز برخوردار هستند.

بر اساس این فرض که ورودی شبکه‌های تلفیقی تصاویر هستند، نرون‌های مورد نیاز برای این نوع شبکه‌ها در سه بُعد عرض، ارتفاع و عمق قرار می‌گیرند. همچنین نرون‌های هر لایه به جای اتصال با تمامی نرون‌های لایه قبل از خود تنها به ناحیه کوچکی از آن لایه متصل هستند. هر شبکه تلفیقی از چند نوع لایه تشکیل می‌شود که هرکدام از آن‌ها شیوه کار مربوط به خود را دارند. هر کدام از این لایه‌ها یک توده سه بُعدی را به عنوان ورودی دریافت کرده و سپس توسط مجموعه‌ای از عملیات‌ها به یک توده سه بُعدی دیگر به عنوان توده خروجی تبدیل می‌کنند.



شکل ۱.۲: در هر لایه نرون‌ها در سه بُعد مرتب می‌شوند و هر لایه از شبکه یک توده سه بُعدی را به عنوان ورودی دریافت کرده و به یک توده سه بُعدی دیگر تبدیل می‌کند. لایه اول که با رنگ قرمز مشخص شده در واقع تصویر اولیه است که به عنوان ورودی به شبکه داده می‌شود.

۱.۲.۲ انوع لایه‌ها در شبکه‌های تلفیقی

به‌طور کلی سه نوع لایه مختلف در شبکه‌های تلفیقی وجود دارد که عبارت‌اند از لایه تلفیقی[۱]، لایه pooling و لایه تماما متصل[۲] که دقیقا مشابه لایه‌هایی است که در شبکه‌های عصبی معمولی استفاده می‌شود. با ترکیب این لایه‌ها با یکدیگر می‌توان شبکه‌های تلفیقی مختلفی ساخت تا در مسائل مورد نظر استفاده شوند. در ادامه به بررسی و نحوه عملکرد هر یک از این لایه‌ها پرداخته خواهد شد.

لایه تلفیقی

همانطور که از اسم شبکه‌های تلفیقی مشخص است، لایه تلفیقی بخش اساسی تشکیل دهنده این شبکه‌ها است به‌طوری که ورودی و خروجی آن را می‌توان به صورت یک توده سه بُعدی از نرون‌ها در نظر گرفت. پارامترهای موجود در این لایه شامل مجموعه‌ای از فیلترها است که قابل یادگیری هستند. این فیلترها عرض و ارتفاع نسبتا کوچکی دارند اما به اندازه عمق داده ورودی امتداد می‌یابند. به عنوان مثال یک فیلتر ساده در لایه ابتدایی شبکه ممکن است دارای ابعادی به صورت $[۳ \times ۵ \times ۵]$ باشد. در حین آموزش شبکه این فیلتر در جهت عرض و ارتفاع ورودی شبکه حرکت می‌کند و ضرب نقطه‌ای بین درایه‌های فیلتر و درایه‌های توده ورودی که متناظر با موقعیت فیلتر هستند محاسبه می‌شود[۳]. با این کار یک نگاشت فعال‌سازی[۴] یا نگاشت ویژگی[۵] دو بُعدی تشکیل می‌شود.

اتصالات محلی

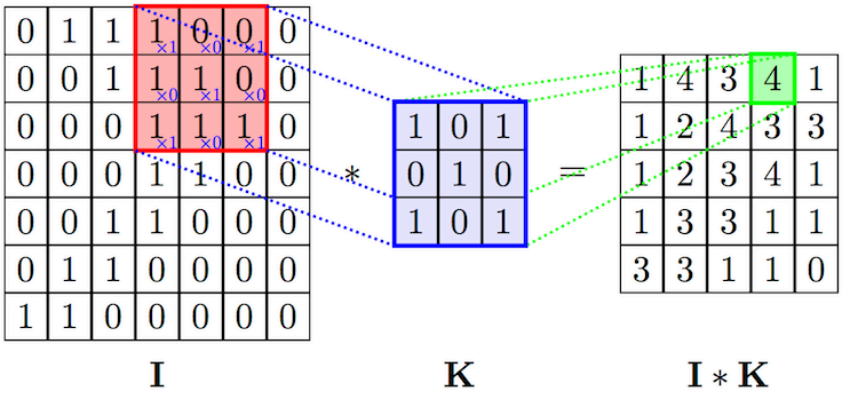
همانطور که مطرح شد در هنگامی که داده‌های ورودی دارای ابعاد زیادی هستند (مانند تصاویر) اتصال نرون‌ها به تمامی نرون‌های توده قبلی از لحاظ عملی امکان‌پذیر نیست و به جای این کار می‌توان هر نرون را تنها به یک ناحیه کوچکی از توده ورودی متصل کرد. وسعت مکانی یا همان ابعاد این ناحیه به عنوان یک ابر پارامتر[۶] با نام ناحیه ادراکی[۷] نرون‌ها شناخته می‌شود و اندازه آن برابر با اندازه فیلترهای استفاده شده در لایه تلفیقی است. همچنین واضح است که عمق ناحیه‌های ادراکی نیز باید به اندازه عمق توده ورودی امتداد یابد. به عنوان مثال فرض کنید یک تصویر با ابعاد $[۳ \times ۳۲ \times ۳۲]$ در اختیار است. اگر ابعاد فیلتر یا معادل آن یعنی ابعاد ناحیه ادراکی به صورت

[1] Convolutional Layer
[2] Fully Connected Layer (FC)

[۳] به این کار اصطلاحا Convolve کردن نیز می‌گویند.

[4] Activation Map
[5] Feature Map
[6] Hyperparameter
[7] Receptive Field

$[۳ \times ۳]$ باشد، آنگاه هر نرون در لایه تلفیقی به ناحیه‌ای با ابعاد $[۳ \times ۳ \times ۳]$ از توده ورودی متصل خواهد بود.



شکل ۲.۲: ناحیه ادراکی در توده ورودی با رنگ قرمز، فیلتر مورد نظر با رنگ آبی و نتیجه ضرب نقطه‌ای آن‌ها در توده خروجی و با رنگ سبز مشخص شده است. همچنین اتصالات محلی به صورت نقطه‌چین نمایش داده شده‌اند.

آرایش فضایی توده‌ها

تا به اینجا در مورد نحوه اتصال هریک از نرون‌ها در لایه تلفیقی به توده ورودی صحبت شد اما سوالی که می‌توان مطرح کرد این است که تعداد کل نرون‌های توده خروجی و نحوه قرارگیری آن‌ها به چه صورت است؟ در حقیقت ابعاد توده خروجی در شبکه‌های تلفیقی توسط سه ابرپارامتر عمق، گام[۱] و لایه‌گذاری با صفر[۲] مشخص می‌شود.

منظور از عمق توده خروجی تعداد نرون‌هایی است که در امتداد عمق توده خروجی قرارگرفته‌اند و همگی به یک ناحیه ادراکی یکسان از توده ورودی متصل هستند. تعداد این نرون‌ها برابر است با تعداد فیلترهایی که قرار است مورد استفاده قرار گیرد. به‌طور کلی استفاده از چند فیلتر به این علت است که هرکدام از آن‌ها می‌توانند جنبه‌های مختلفی از توده ورودی را یاد بگیرند و در نهایت منجر به افزایش عملکرد شبکه شوند.

منظور از پارامتر گام در واقع تعداد گام‌هایی است که فیلتر در جهت عرض و ارتفاع توده ورودی حرکت داده می‌شود. اگر تعداد گام‌های مشخص شده برابر با ۱ باشد به این معناست که در هر بار حرکت فیلتر تنها به اندازه یک پیکسل روی تصویر جابجا خواهد شد. مشخص است که هرچقدر تعداد گام‌های انتخاب شده بزرگ‌تر باشد ابعاد توده خروجی کوچک‌تر خواهد بود.

برخی مواقع لازم می‌شود که حاشیه‌های توده ورودی با مقادیر صفر پر شوند. هدف از این کار تعیین ابعاد توده خروجی با ابعاد دلخواه است که برای این کار کافی‌ست تعداد واحدهای حاشیه‌گذاری به عنوان یک ابرپارامتر مشخص شود. به‌طور کلی می‌توان ابعاد توده خروجی را به عنوان تابعی از ابعاد توده ورودی در نظر گرفت. فرض

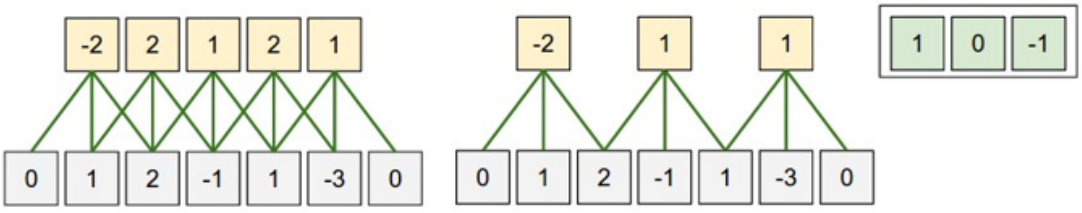
[1] Stride
[2] Zero-padding

کنید ابعاد توده ورودی و ناحیه ادراکی به ترتیب برابر با W و F باشد. اگر گام حرکت فیلترها برابر S و به اندازه P واحد حاشیه‌های توده ورودی با مقدار صفر حاشیه‌گذاری شود، آنگاه تعداد نرون‌های مورد نیاز در توده خروجی برابر است با $\frac{W-F+۲P}{S}+۱$. به عنوان مثال فرض کنید ابعاد توده ورودی به صورت [۷ × ۷]، اندازه فیلتر برابر [۳ × ۳] و ناحیه ادراکی بدون لایه‌گذاری صفر باشد. در این صورت اگر گام حرکت ۱ در نظر گرفته شود، ابعاد توده خروجی به صورت [۵ × ۵] و اگر گام حرکت برابر با ۲ باشد ابعاد آن به صورت [۳ × ۳] خواهد بود. نکته‌ای که باید به آن توجه نمود این است که حاصل رابطه معرفی شده باید یک عدد صحیح باشد و در غیر این صورت نمی‌توان آرایش صحیحی برای توده خروجی به دست آورد.

فرض کنید یک توده ورودی یک بُعدی با پنج مولفه ($W = ۵$) داده شده است ([۱ × ۱ × ۵]). اگر فیلتر مورد نظر به صورت $[۱, ۰, -۱]$ در نظر گرفته شود آنگاه:

الف) اگر پارامترهای ناحیه ادراکی، لایه‌گذاری صفر و گام حرکت فیلتر به ترتیب برابر ۳، ۱ و ۲ قرار داده شود یا به عبارت دیگر $F = ۳$، $P = ۱$ و $S = ۱$، آنگاه تعداد نرون‌های لایه خروجی برابر با $\frac{۵-۳+۲\times۱}{۱}+۱ = ۵$ خواهد بود.

ب) اگر گام حرکت فیلتر به صورت $S = ۲$ تغییر یابد، به‌طور مشابه تعداد نرون‌های توده خروجی برابر با $\frac{۵-۳+۲\times۱}{۲}+۱ = ۳$ است. توجه شود که با پارامترهای فعلی نمی‌توان گام حرکت فیلتر را برابر ۳ قرار داد؛ زیرا مقدار به دست آمده به عنوان تعداد نرون‌های توده خروجی دیگر یک عدد صحیح نخواهد بود. در صورتی که نیاز باشد S برابر با ۳ قرار گیرد می‌توان با تغییر پارامترهای دیگر (مثلا $P = ۲$)، تعداد نرون‌های توده خروجی را به یک عدد صحیح تبدیل کرد.



شکل ۳.۲: آرایش فضایی پس از عملیات تلفیقی بر روی یک توده ورودی.

به‌طور کلی استفاده از لایه‌گذاری با صفر در معماری شبکه‌های تلفیقی امری رایج و پر استفاده است اما نکته دیگری که در این مورد باید به آن توجه نمود این است که اگر مایل باشیم بُعد توده ورودی و خروجی با یکدیگر برابر باشند حتما نیازمند استفاده از لایه‌گذاری با صفر خواهیم بود. به عبارت دیگر هنگامی که $P = \frac{F-۱}{۲}$ و $S = ۲$ در

نظر گرفته شود، این تضمین وجود دارد که بُعد توده ورودی و خروجی با یکدیگر برابر خواهند بود.

لایه Pooling

در معماری‌های موجود برای شبکه‌های تلفیقی مرسوم است که در بین لایه‌های تلفیقی از یک لایه Pooling نیز استفاده شود. کاربرد این لایه کاهش تدریجی ابعاد توده در اختیار است به‌طوری که پارامترهای موجود و در نتیجه هزینه‌های محاسباتی نیز کاهش یابند. علاوه بر این کاهش ابعاد توده‌ها برای جلوگیری از بیش برازش شبکه نیز بسیار مفید است [۲۴]. لایه Pooling با استفاده از تابع Max، عمق هر یک از توده‌های ورودی را به‌طور مستقل کاهش می‌دهد. رایج‌ترین شکل لایه Pooling به صورتی است که یک فیلتر با ابعاد $[۲ \times ۲]$ بر روی عرض و ارتفاع هر برش عمقی از توده ورودی و با گام $S = ۲$ حرکت می‌کند. سپس عملگر Max بر روی هریک از نواحی ادراکی اعمال شده و بیشینه آن‌ها را انتخاب می‌کند (در صورتی که ابعاد فیلتر استفاده شده $[۲ \times ۲]$ باشد، از بین ۴ عدد موجود بیشینه آن‌ها را انتخاب می‌کند). واضح است که در این صورت عمق توده ورودی بدون تغییر باقی می‌ماند و تنها عرض و ارتفاع آن کاهش خواهد یافت. به‌طور خلاصه لایه‌های Pooling به شیوه زیر عمل می‌کنند:

۱. پذیرش توده ورودی با ابعاد $[W_۱ \times H_۱ \times D_۱]$.

۲. تعیین پارامترهای F و S به عنوان اندازه ناحیه ادراکی و گام حرکت فیلترها.

۳. تولید یک توده ورودی با ابعاد $[W_۲ \times H_۲ \times D_۲]$ به‌طوری که در آن

$$W_۲ = \frac{W_۱ - F}{S} + ۱,$$

$$H_۲ = \frac{H_۱ - F}{S} + ۱,$$

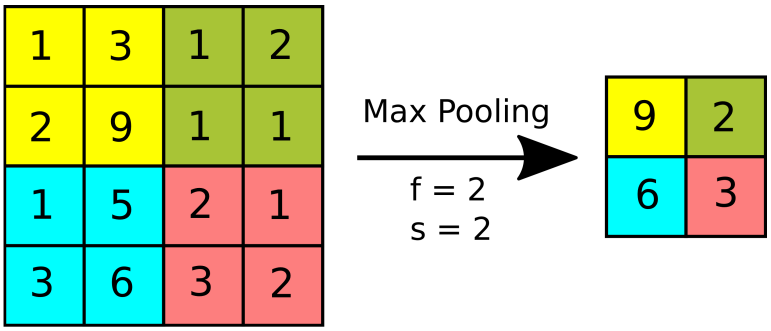
$$D_۲ = D_۱.$$

نکته ساده اما مهمی که باید به آن توجه کرد این است که هرچه میزان ناحیه ادراکی در لایه Pooling بیشتر باشد، مقدار بیشتری از توده ورودی از دست خواهد رفت. در هنگام استفاده از تابع Max در لایه Pooling از آن به عنوان Max pooling یاد می‌شود که علاوه بر این تابع می‌توان از توابع دیگری مانند میانگین (Average Pooling) و نُرم-۲[1] نیز استفاده کرد. لیکون و همکارانش یک تحلیل نظری دقیق از کارایی Max Pooling و Average Pooling ارائه

[1] L2-Norm

دادند [۱۷]. سپس مولر و همکارانش با انجام یک مقایسه بین این دو تابع نشان دادند که استفاده از Max Pooling منجر به همگرایی سریع‌تر و افزایش کارایی شبکه‌ها می‌شود [۱۲۵].

بسیاری از کاربران علاقه زیادی به لایه Pooling نداشته و معتقد هستند که می‌توان در معماری‌های شبکه از آن‌ها استفاده نکرد. به عنوان مثال توبیاس و همکارانش نوعی معماری از شبکه‌های تلفیقی را بررسی می‌کنند که در آن تنها از لایه‌های تلفیقی پی در پی استفاده شده است [۱۳۳]. در این مقاله آن‌ها برای کاهش ابعاد توده‌ها استفاده از لایه‌های تلفیقی با گام‌های بزرگتر برای فیلترها را پیشنهاد می‌کنند. علاوه بر این حذف لایه‌های Pooling در برخی از شبکه‌ها عملکرد بهتری را در پی داشته است و به نظر می‌رسد که در معماری‌های آینده تمایل کاربران به عدم استفاده از این لایه باشد.



شکل ۴.۲: نتیجه اعمال لایه Max Pooling بر روی یک توده ورودی.

تقریبا می‌توان گفت لایه Pooling لایه‌ای است که مطالعات بیشتری نسبت به سایر لایه‌های بر روی آن انجام شده است. چند نمونه پیشرفته‌تر از این لایه که محبوبیت زیادی به دست آورده اند عبارت‌اند از:

۱. Stochastic Pooling:

زیلر و همکارانش نشان دادند که Max Pooling در شبکه‌های تلفیقی نسبت به بیش برازش شبکه حساسیت بالایی دارد [۱۵۹]. سپس با معرفی Stochastic Pooling سعی کردند تا این مشکل را برطرف کنند [۱۶۰]. این روش با یک رویه تصادفی کار می‌کند به نحوی که از هر ناحیه مورد نظر با استفاده از یک توزیع چند جمله‌ای عملیات Pooling را انجام می‌دهد و تا حدودی از مشکل بیش برازش نیز جلوگیری می‌کند. البته این روش در بسیاری از کاربردها باعث کاهش دقت نتیجه خواهد شد.

۲. Spatial Pyramid Pooling (SPP):

همانطور که مطرح شد توده‌های ورودی شبکه‌های تلفیقی باید دارای ابعاد ثابتی باشند و نمی‌توان تصاویر با ابعاد دلخواهی را به شبکه تزریق کرد. این محدودیت ممکن است در برخی موارد برای بررسی تصاویر و

اشیاء موجود در آن‌ها که اندازه ثابتی دارند و در نواحی خاصی از تصویر هستند مشکل‌ساز باشد. در این روش با جایگزین کردن آخرین لایه pooling با یک لایه spatial pyramid pooling می‌توان به یک راه حل انعطاف‌پذیر برای مدیریت مقیاس‌های مختلف دست یافت [۵۱].

۳. Def-Pooling:

تغییر در شکل و ظاهر در مسائل بینایی کامپیوتر و به خصوص شناسایی و ردیابی اشیاء، همواره به عنوان یک چالش اساسی شناخته می‌شده است که Max pooling و Average Pooling چندان قادر به مدیریت آن نیستند. برای حل این مشکل اویانگ و همکارانش یک لایه Pooling دیگر با تغییر شکل محدود[1] معرفی کردند تا با یادگیری تغییر شکل الگوهای دیداری[2] دقت شبکه افزایش یابد [۱۰۲]. از این لایه می‌توان به جای Max Pooling و در هر جای از شبکه استفاده کرد.

لایه تماما متصل

همانند لایه‌های متصل در شبکه‌های عصبی، در شبکه‌های تلفیقی نیز نرون‌های موجود در لایه‌های تماما متصل با تمام نرون‌های موجود در لایه قبل از خود متصل هستند. اغلب پس از آخرین لایه Pooling لایه‌های تماما متصل قرار می‌گیرند که نگاشت ویژگی‌های دو بُعدی را به بردارهای ویژگی تبدیل می‌کنند. این بردارهای ویژگی این قابلیت را می‌دهند که نتیجه شبکه در قالب یک بردار با اندازه مشخص که برابر با تعداد کلاس‌های موجود است ارائه شود. اکثر پارامترهای موجود در شبکه‌های تلفیقی مربوط به این لایه‌هاست که باعث افزایش چشم‌گیر هزینه‌های محاسباتی می‌شوند. در برخی شبکه‌ها این لایه‌ها حذف و با لایه‌های تلفیقی جایگزین می‌شوند و یا حتی تعداد اتصالات آن‌ها با روش‌های خاصی مانند تعویض معماری تماما متصل با معماری به‌طور پراکنده متصل کاهش می یابد [۱۴۱]. از طرف دیگر یکی از روش‌های ارائه شده برای مدیریت هرچه بهتر لایه‌های تلفیقی، روش Network in Network (NIN) نام دارد که در آن لایه‌های تلفیقی با یک شبکه عصبی پرسپترون کوچک جایگزین می‌شوند [۸۷]. این شبکه‌های پرسپترون شامل چندین لایه تماما متصل و با توابع فعال‌سازی غیر خطی هستند. به این ترتیب فیلترهای خطی با شبکه‌های عصبی غیر خطی جایگزین شده و باعث دست یابی به عملکردهای خوبی در دسته‌بندی تصاویر می‌شود.

[1] Deformation Constrainded Pooling Layer
[2] Deformation of Visual Pattern

لایه ReLU

ReLU[1] در واقع یک تابع فعال‌سازی است که در شبکه‌های تلفیقی بسیار استفاده می‌شود اما معمولا به عنوان یک لایه از شبکه‌های تلفیقی شناخته می‌شود. نحوه عملکرد این تابع به این صورت است که ورودی x را به $(۰, x)$ Max نگاشت می‌کند [۴۸]. به عبارت دیگر در ورودی‌های مثبت تغییری ایجاد نکرده و به ازای ورودی‌های منفی ۰ را برمی‌گرداند و در مقایسه با توابع فعال‌سازی قبل می‌تواند مقادیر بزرگ‌تری را به عنوان خروجی داشته باشد. استفاده از این تابع به دلیل سادگی و هزینه محاسباتی کم، فرآیند یادگیری را تسریع می‌بخشد. به‌طور کلی انتخاب تابع فعال‌سازی وابسته به کاربرد مد نظر است اما تابع ReLU در طیف وسیعی از کاربردها عملکرد خوبی دارد [۴۳].

۲.۲.۲ الگوی استفاده از لایه‌ها

مرسوم‌ترین شکل استفاده از لایه‌های مختلف در شبکه‌های تلفیقی ترکیب چند لایه تلفیقی و ReLU (Conv-ReLU) است که بعد از آن‌ها لایه‌های Pooling قرار می‌گیرند. سپس این ترکیب تا زمانی که تصویر به اندازه دلخواه برسد تکرار می‌گردد. همچنین در برخی موارد و خصوصا در لایه‌های آخر استفاده از لایه‌های تماما متصل امری رایج است. آخرین لایه تماما متصل خروجی‌های حاصل یا به عبارت دیگر امتیاز کلاس‌های موجود را تولید می‌کند. به‌طور کلی قاعده استفاده از لایه‌های مختلف را می‌توان به صورت زیر نشان داد:

$$INPUT \rightarrow \left[[CONV \rightarrow ReLU]^{\star N} \rightarrow POOL?\right]^{\star M} \rightarrow [FC \rightarrow ReLU]^{\star K} \rightarrow FC$$

که در آن $\star$ نشان‌گر تکرار و $POOL?$ به معنی انواع مختلف لایه Pooling است. علاوه بر آن معمولا $۰ < N \leq ۳$، $M \geq ۰$ و $۰ \leq K < ۳$ است.

به‌طور خاص چند نوع از معماری‌های پر کاربرد شبکه‌های تلفیقی به صورت زیر است:

- $INPUT \rightarrow FC$

  در معماری یک دسته‌بندی کننده خطی پیاده‌سازی می‌شود به‌طوری که $N = M = K = ۰$.

- $INPUT \rightarrow CONV \rightarrow ReLU \rightarrow FC$

- $INPUT \rightarrow [CONV \rightarrow ReLU \rightarrow POOL]^{\star ۲} \rightarrow FC \rightarrow ReLU \rightarrow FC$

[1] Rectified Linear Unit

- $INPUT \to [CONV \to ReLU \to CONV \to ReLU \to POOL]^{*۳} \to [FC \to ReLU]^{*۲} \to FC$ در این معماری دو لایه تلفیقی قبل از هر لایه Pooling قرار گرفته‌اند. به‌طور کلی این شیوه برای شبکه‌های عمیق و بزرگ ایده مناسبی است؛ زیرا چندین لایه تلفیقی ترکیب شده می‌توانند قبل از اینکه توسط لایه Pooling از دست بروند، ویژگی‌های بیشتر و پیچیده‌تری از توده ورودی به دست بیاورند.

۳.۲.۲ مقادیر مناسب پارامترها در شبکه‌های تلفیقی

در مورد ابعاد تصاویر ورودی به شبکه محدودتی وجود ندارد اما توصیه می‌شود به نحوی باشند که چندین مرتبه بر عدد ۲ تقسیم شوند. به عنوان مثال ابعاد تصاویر موجود در مجموعه‌داده‌های شناخته شده در این زمینه مانند ImageNet [۳۳]، COCO [۸۸] و VOC [۳۶] برابراند با ۳۲، ۶۴، ۹۶، ۲۲۴، ۳۸۴ و ۵۱۲. در مدل‌ها و معماری‌های مختلف شبکه‌های تلفیقی سعی می‌شود تا از فیلترهای با ابعاد کوچک ($[۳ \times ۳]$ و یا حداکثر $[۵ \times ۵]$) به جای فیلترهای با ابعاد بزرگ استفاده شود؛ زیرا استفاده از ترکیب چند لایه تلفیقی با فیلترهای کوچک این امکان را فراهم می‌کند تا ویژگی‌های بیشتر و بهتری را با تعداد پارامترها و همچنین مصرف حافظه به مراتب کمتری به دست آوریم.

در مورد گام حرکت فیلترها توصیه می‌گردد که $S = ۱$ باشد و همچنین از لایه‌گذاری با صفر در توده ورودی استفاده گردد به‌طوری که لایه تلفیقی عرض و ارتفاع توده ورودی را تغییر ندهد. در این صورت وظیفه کاهش ابعاد توده‌ها به عهده لایه‌های Pooling قرار می‌گیرد و لایه‌های تلفیقی بهتر به وظیفه خود عمل می‌کنند.

۴.۲.۲ چند مورد از شبکه‌های تلفیقی معروف برای دسته‌بندی تصاویر

۱. LeNet:

این شبکه از نمونه‌های اولیه شبکه‌های تلفیقی است که توسط یان در سال ۱۹۹۰ توسعه داده شد و از جمله شبکه‌های موفق شناخته می‌شود [۸۳]. این شبکه برای خواندن کدهای پستی، ارقام و موارد مشابه با آن‌ها مورد استفاده قرار می‌گرفت.

۲. AlexNet:

اولین نمونه‌ای که توجه متخصصان بینایی کامپیوتر را به سمت شبکه‌های عصبی تلفیقی جلب کرد شبکه AlexNet بود [۷۵]. این شبکه توسط کریژِوسکی و همکارانش در سال ۲۰۱۲ و به منظور شرکت در رقابت

ILSVRC طراحی شد. معماری این شبکه بسیار شبیه به معماری شبکه LeNet بود با این تفاوت که دارای عمق بیشتری بود و به جای استفاده از لایه ReLU بعد از لایه تلفیقی، از ترکیب چندین لایه تلفیقی با یکدیگر استفاده می‌کرد.

۳. ZF:

این شبکه نیز به منظور شرکت در رقابت ILSVRC در سال ۲۰۱۳ طراحی و توسعه داده شده بود و پس از آن با نام ZF که مخفف نام توسعه‌دهندگان آن یعنی زیلر و فرگوس است شناخته می‌شود [۱۶۱]. معماری این شبکه نیز شبیه به معماری شبکه AlexNet است که به واسطه تغییر پارامترهای موجود و خصوصا افزایش اندازه لایه‌های تلفیقی میانی در آن باعث بهینه‌سازی آن نسبت به AlexNet شده بود.

۴. GoogLeNet:

برنده رقابت ILSVRC در سال ۲۰۱۴ این شبکه بود که توسط محققان شرکت گوگل طراحی و توسعه داده شده بود [۱۴۱]. ویژگی بارزی که در این معماری وجود داشت معرفی یک ماژول مفهومی[1] بود که تعداد پارامترهای موجود در شبکه را به شدت کاهش می‌داد. دستاورد دیگر GoogLeNet این بود که امکان افزایش عمق و عرض شبکه را بدون افزایش هزینه‌های محاسباتی فراهم می‌کرد.

۵. VGGNet:

جایگاه دوم برندگان رقابت ILSVRC در سال ۲۰۱۴ متعلق به این شبکه بود که توسط زیسرمن و همکارانش در گروه Visual Graphics Group در دانشگاه آکسفورد طراحی شده بود [۱۳۱]. خصوصیت جدیدی که این شبکه نشان داد این بود که عمق شبکه یک مولفه حیاتی برای کارایی مناسب است. نسخه نهایی این شبکه دارای ۱۶ لایه (VGG16) تلفیقی و تماما متصل است که فیلترهای استفاده شده در لایه‌های تلفیقی [۳ × ۳] و فیلترهای لایه‌های Pooling [۲ × ۲] هستند. پس از آن مشخص شد که علی‌رغم قدرت دسته‌بندی کمتر آن نسبت به GoogLeNet، این شبکه در وظایف یادگیری انتقالی چندتایی[2] بهتر عمل می‌کند. به همین دلیل در حال حاضر VGGNet محبوب‌ترین شبکه برای استخراج ویژگی از تصاویر است و مدل از پیش آموزش داده شده آن‌ها برای استفاده در کتابخانه Caffe وجود دارد. با این وجود اما تعداد پارامترهای نسبتا زیاد این شبکه یکی از نقاط ضعف آن محسوب می‌شود که سرعت آموزش شبکه را به شدت کاهش می‌دهد. اکثر

[1] Inception Module
[2] Multiple Transfer Learning Tasks

این وزن‌ها در اولین لایه تماما متصل است که البته می‌توان بدون کاسته شدن از میزان عملکرد، آن لایه‌های تماما متصل را حذف کرده و در نتیجه تعداد پارامترهای لازم را نیز کاهش داد. با این وجود همچنان شبکه VGGNet نسبت به GoogLeNet کندتر است و همچنین حجم مدل‌های آموزش دیده در آن حجم بالایی دارند.

۶. ResNet (Residual Networks):

این شبکه برنده رقابت ILSVRC در سال ۲۰۱۵ بود که توسط کایمینگ و همکارانش توسعه داده شده بود [۵۲]. در این شبکه از Batch Normalization تا حدود زیادی استفاده شده و همچنین لایه‌های تماما متصل در انتهای شبکه حذف شده‌اند. چون در شبکه‌های تلفیقی قبلی با افزایش بیش از حد تعداد لایه‌ها بر خلاف انتظار عملکرد آن‌ها نیز کاهش می‌یافت، در این معماری ساختاری معرفی شده که با استفاده از آن می‌توان تعداد لایه‌های بسیار زیادی را در نظر گرفت به‌طوری که عملکرد شبکه کاهش نیابد.

۳.۲ شبکه‌های تلفیقی در شناسایی اشیاء

مدل‌های مبتنی بر شبکه‌های تلفیقی و کاربردهای مختلف آن‌ها به مواردی مانند دسته‌بندی تصاویر محدود نمی‌شود. به این ترتیب می‌توان بر اساس این مدل‌ها چهارچوب‌های جدید دیگری به دست آورد که در برخی کاربردهای دیگر نیز قابل استفاده باشند که یکی از مهم‌ترین آن‌ها شناسایی اشیاء است. چهارچوب مشتق شده از این مدل‌ها برای شناسایی اشیاء R-CNN[1] نام دارد [۴۲]. ایده اصلی R-CNN ایجاد پیشنهادهای چندگانه[2] برای موقعیت اشیاء در تصاویر و سپس استخراج ویژگی از هریک از آن پیشنهادها توسط شبکه‌های تلفیقی است. به عبارت دیگر R-CNN ها از شبکه‌های تلفیقی به عنوان یک استخراج کننده ویژگی‌ها استفاده می‌کنند و معمولا تغییری در معماری آن‌ها نمی‌دهند. در گام بعدی هریک از پنجره‌های نامزد با یک ماشین بردار پشتیبان (SVM) خطی مخصوص برای هر دسته، دسته‌بندی می‌شوند. به این ترتیب الگوی تشخیص از طریق ناحیه‌ها[3] کارایی نسبتا خوبی برای شناسایی اشیاء به دست آورد و به یک معماری عمومی برای شناسایی اشیاء به شیوه نوین تبدیل شد [۴۱، ۱۱۵]. نکته‌ای که در مورد R-CNN ها باید دانست این است که آن‌ها تا حدود زیادی بر روی دقت مکان شئ تاکید دارند و همین امر ممکن است قدرت و کارایی آن‌ها را محدود سازد. از طرف دیگر در صورتی که تعداد پیشنهادها بسیار زیاد در نظر گرفته شود، آنگاه این کار باعث کاهش بهینگی آن‌ها خواهد شد. به همین دلیل اغلب مطالعات صورت گرفته

[1] Regions with CNN Features
[2] Multi object proposals
[3] Recognition Using Regions

در مورد آن‌ها بر روی همین دو مبحث اشاره دارند [۱۶۳، ۵۰].

۴.۲ مروری بر مطالعات پیشین

با توجه به مدل‌های ارائه شده مانند GoogLeNet و VGGNet و بررسی میزان عملکرد آن‌ها، این نتیجه به دست آمد که افزایش عمق شبکه‌های تلفیقی رابطه مستقیمی با افزایش کارایی آن‌ها دارد. ایده بعدی که برای افزایش کارایی شبکه‌ها مطرح شد، آموزش چندین شبکه به‌طور مشترک بود که توانست کارایی بهتری نسبت به شبکه‌های واحد از خود نشان دهد. همچنین برخی از محققان به طراحی ترکیب ساختارهای عمیق به صورت آبشاری روی آوردند به‌طوری که در آن‌ها خروجی یک شبکه می‌تواند به عنوان ورودی شبکه‌ای دیگر و به منظور ساخت یک شبکه بزرگ‌تر مورد استفاده قرار گیرد. به عنوان مثال وانگ و همکارانش دو شبکه را برای استخراج اشیاء به یکدیگر متصل کردند به‌طوری که اولین شبکه برای تعیین موقعیت اشیاء و خروجی آن مختصات متناظر با شئ مورد نظر بود [۱۵۱]. همچنین سان و همکارانش شبکه‌های تلفیقی سه مرحله‌ای برای شناسایی نقاط کلیدی صورت[1] ارائه دادند [۱۴۰]. این مدل به گونه‌ای عمل می‌کرد که در گام اول تخمین‌های اولیه نسبتا دقیقی از نقاط حاصل می‌شد و گام‌های بعدی افزایش دقت نقاط به دست آمده را در پی داشت. به همین ترتیب اویانگ از طرح یادگیری چند مرحله‌ای که توسط زِنگ و همکارانش [۱۶۲] پیشنهاد شده بود استفاده کرد [۱۰۲]. هدف از این مطالعه این بود که دسته‌بندی کننده‌های هر مرحله با دسته‌بندی کننده‌های مرحله قبل از خود ادغام شوند تا نمونه‌هایی که به صورت اشتباه دسته‌بندی شده بودند تصحیح گردند.

نگرش مشابه دیگری که در این زمینه وجود دارد این است که به جای طراحی یک معماری واحد، شبکه‌های مختلفی آموزش داده شوند به‌طوری که هر یک از آن‌ها قادر باشند وظیفه مستقلی انجام دهند و سپس نتایج حاصل از این شبکه‌ها با یکدیگر ترکیب شوند. در این راستا میکلوت و همکارانش نشان دادند که چگونه می‌توان نتایج نهایی را با توجه به نتایج به دست آمده از هر یک از شبکه‌ها به دست آورد [۹۷]. همچنین در طی آزمایشی به منظور ارزیابی کارایی تکنیک ترکیب مدل‌های مختلف با یکدیگر، ۱۰ مدل با تنظیمات گوناگون آموزش داده شدند و پس از میانگین‌گیری آن‌ها را با یکدیگر ترکیب کردند. نتایج حاصل شده نشان داد که مدل‌های موجود به صورت مکمل نسبت به یکدیگر عمل می‌کنند و باعث افزایش کارایی نهایی خواهند شد [۱۰۲].

از طرف دیگر برخی محققان بدون تغییر ساختار شبکه‌های تلفیقی تلاش کردند تا اطلاعات حاصل از آن‌ها

[1] Facial Keypoints

را با منابع دیگری مانند ساختارهای کم عمق یا اطلاعات متنی[1] ادغام کنند و نتیجه نهایی را به مدل ارائه دهند [۱۵۳]. به عبارت ساده‌تر این روش‌ها تنها ویژگی‌های لازم را با استفاده از شبکه‌های تلفیقی استخراج کرده و سپس آن‌ها را به ساختارهایی مانند SVM ارائه می‌کنند و همانطور که گفته شد R-CNN ها نیز از همین دسته‌اند و یکی از موفق‌ترین الگوریتم‌های شناسایی اشیاء محسوب می‌شوند. همچنین در برخی مواقع امکان اطلاعات متنی برای شناسایی اشیاء وجود دارند و می‌توان این اطلاعات را با جعبه محاطی[2] ادغام نمود.

۵.۲ الگوریتم‌های موجود برای شناسایی اشیاء به شیوه نوین

تا به اینجا مشاهده شد که ایده کلی و اولیه برای شناسایی اشیاء در روش‌های نوین این است که ابتدا تعدادی جعبه محاطی پیشنهادی مشخص شده و سپس با استفاده از شبکه‌های تلفیقی هر یک از این ناحیه‌های مشخص شده دسته‌بندی شوند. مشکل اصلی این رویکرد این است که اشیاء موجود در تصاویر ممکن است ابعاد و مقیاس‌های متفاوتی داشته باشند و نتوانند به خوبی در پنجره‌های مشخص شده قرار گیرند. به همین خاطر ممکن است که مجبور باشیم تا تعداد پنجره‌ها را افزایش دهیم که این کار باعث افزایش هزینه‌های محاسباتی خواهد شد و همچنان عملکرد قابل قبولی نخواهد داشت. از این روی الگوریتم‌های متعددی برای حل این مشکلات و افزایش سرعت شناسایی معرفی شدند که در ادامه به مهم‌ترین آن‌ها پرداخته خواهد شد.

قبل از توضیح این روش ابتدا نیاز است تا با چند اصطلاح رایج در الگوریتم‌های شناسایی اشیاء آشنا شویم که عبارت‌اند از:

۱. جعبه محاطی:

منظور از جعبه محاطی کادری است که به دور اشیاء شناسایی شده کشیده شده و به واسطه آن موقعیت اشیاء در تصویر مشخص می‌شود.

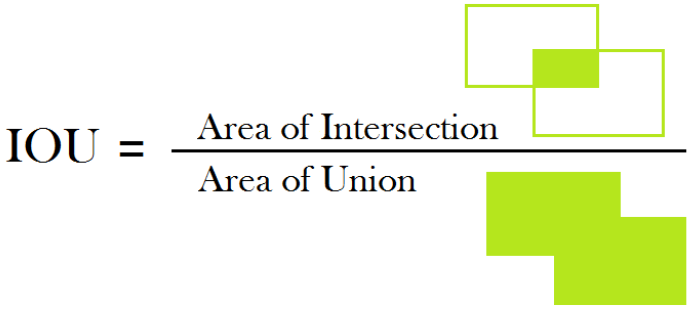
۲. جعبه محاطی حقیقی:

به جعبه محاطی واقعی و صحیح که در داده‌های آموزشی وجود دارد و برای تعیین مکان اشیاء در فرآیند آموزش استفاده می‌شود جعبه محاطی حقیقی گویند.

[1] Contextual Information
[2] Bounding Box

۳. IOU[1]:

IOU در واقع میزان هم‌پوشانی میان جعبه‌های محاطی پیش‌بینی شده و جعبه‌های محاطی حقیقی را اندازه‌گیری می‌کند. IOU ها همواره مقادیری بین ۰ و ۱ داشته و به صورت زیر تعریف می‌گردد:



هنگامی که جعبه محاطی برای اشیاء مختلف پیش‌بینی می‌شود نیاز است تا میزان IOU آن نسبت به جعبه محاطی حقیقی محاسبه گردد. هدف کلی این است تا مقدار IOU برای جعبه‌های محاطی به دست آمده تقریبا برابر با ۱ باشد.

شکل ۵.۲: در این تصویر جعبه‌های محاطی حقیقی با رنگ قرمز و جعبه‌های محاطی متناظر با رنگ زرد مشخص شده‌اند. در تصویر سمت چپ میزان IOU کم و در تصویر سمت راست مقدار آن زیاد و مورد قبول است.

۴. دقت[2]: معمولا دقت به صورت نرخ تشخیص‌هایی که است که به درستی به عنوان مثبت پیش‌بینی شده‌اند تقسیم بر کلیه پیش‌بینی‌های انجام شده (درست و غلط). این معیار را می‌توان به صورت زیر نیز تعریف کرد:

$$Precision = \frac{TP}{TP + FP}, \qquad (۱-۲)$$

[1] Intersect Over Union
[2] Precision

به‌طوری که در آن منظور از TP[1] نمونه‌هایی است که به درستی شناسایی شده‌اند و منظور از FP[2] نمونه‌هایی است که اشتباها به عنوان یک شئ شناسایی شده‌اند. به عنوان مثال فرض کنید ۵۰ تصویر در اختیار باشد و در مجموع تصاویر تعداد ۱۵۰ اتومبیل وجود داشته باشد. اگر در نهایت تعداد ۱۲۰ اتومبیل در بین ۵۰ تصویر شناسایی شود آنگاه برای محاسبه نرخ دقت باید ۱۲۰ اتومبیل شناسایی شده بررسی شوند. اگر از این تعداد ۲۰ تای آن‌ها نادرست شناسایی شده باشند یا به عبارت دیگر تنها ۱۰۰ اتومبیل به درستی شناسایی شده باشند پس نرخ دقت برابر خواهد بود با $\frac{۱۰۰}{۱۲۰} = ۰/۸۳$. در اینجا معیار شناسایی درست یا نادرست شبکه را می‌توان بر اساس مقدار IOU معین کرد به‌طوری که اگر IOU کمتر از یک حد آستانه (معمولا ۰/۵ یا ۰/۷۵) باشد شناسایی انجام گرفته غلط و در غیر این صورت صحیح خواهد بود.

۵. صحت[3]:

ملاحظه شد که در معیار دقت، تعداد کل اشیاء موجود (۱۵۰ اتومبیل) در نظر گرفته نمی‌شود. در این صورت اگر تعداد ۱۵۰۰ اتومبیل نیز در تصاویر موجود باشد طبق شناسایی صورت گرفته نرخ دقت مجددا برابر با ۰/۸۳ خواهد بود. به این ترتیب نتیجه می‌شود که معیار دقت نمی‌تواند معیار دقیقی در نظر گرفته شود. به همین علت معیار صحت به صورت زیر تعریف می‌گردد:

$$Recall = \frac{TP}{TP + FN}, \qquad (۲-۲)$$

به‌طوری که در آن FN[4] نمونه‌هایی است که اشتباها به عنوان یک شئ شناسایی نشده‌اند. به این ترتیب در مثال ذکر شده مقدار Recall برابر با $\frac{۱۰۰}{۱۵۰} = ۰/۶۶$ خواهد بود. به عبارت دیگر Recall به معنی این است که عملیات شناسایی تا چه اندازه به خوبی انجام شده است.

۶. دقت متوسط و میانگین دقت متوسط:

معیارهای متعددی برای ارزیابی مدل‌های شناسایی اشیاء معرفی شده است اما از رایج‌ترین آن‌ها دقت متوسط[5] و میانگین دقت متوسط[6] نام دارند. میانگین دقت متوسط در واقع متوسط بیشینه مقادیر دقت (Precision) به ازای مقادیر صحت (Recall) را اندازه‌گیری می‌کند. همچنین دقت متوسط را می‌توان برابر با

[1] True Positive
[2] False Positive
[3] Recall
[4] False Negative
[5] Average Precision (AP)
[6] mean Average Precision (mAP)

ناحیه زیر منحنی دقت و صحت در نظر گرفت و در واقع معیاری است که این دو معیار را با یکدیگر ترکیب می‌کند.

R-CNN ۱.۵.۲

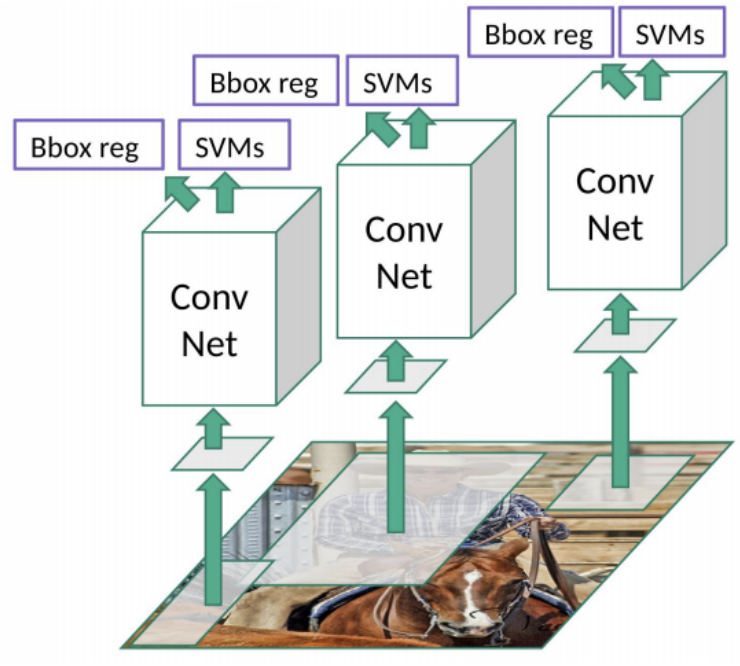
به منظور عدم استفاده از ناحیه‌های پیشنهادی زیاد برای شناسایی اشیاء، راس و همکارانش در سال ۲۰۱۴ روشی را ارائه دادند که با استفاده از جستجوی انتخابی[1] [۱۴۵] تنها ۲۰۰۰ ناحیه استخراج می‌گردد که به آن‌ها ناحیه‌های نامزد[2] گفته می‌شود و سپس با توجه به آن‌ها عملیات دسته‌بندی انجام می‌گیرد [۴۲]. سپس این ناحیه‌ها به ابعاد مربعی و یکسانی تبدیل شده و به یک شبکه تلفیقی تزریق می‌گردند که حاصل آن تولید یک بردار ویژگی با ابعاد ۴۰۹۶ است که پس از آن و به منظور دسته‌بندی به یک SVM ارائه می‌شوند (شکل ۶.۲). همچنین برای پیش‌بینی وجود یک شخص در تصویر،چهار مقدار دیگر نیز محاسبه می‌شود که معمولا مقادیر انحراف[3] نامیده می‌شوند و وظیفه آن‌ها افزایش دقت موقعیت پیش‌بینی شده اشیاء است. فرض کنید که یک شخص در درون یک ناحیه نامزد قرار گرفته است. در این صورت الگوریتم تمایل دارد تا آن را به عنوان یک شخص مشخص کند اما نیمی از صورت او در آن ناحیه وجود ندارد. در این صورت مقادیر انحراف کمک می‌کنند تا جعبه‌های محاطی به‌طور قابل قبولی تنظیم گردد.

نقطه ضعفی که این الگوریتم دارد این است که دسته‌بندی ۲۰۰۰ ناحیه به ازای هر تصویر همچنان زمان زیادی مصرف می‌کند (حدود ۴۷ ثانیه برای هر تصویر). به همین خاطر امکان استفاده از این روش در کاربردهای بلادرنگ به هیچ عنوان مناسب نیست. علاوه بر در طی اجرای الگوریتم جستجوی انتخابی هیچ عملیات یادگیری اتفاق نمی‌افتد و همین نکته ممکن است باعث انتخاب‌های نامناسبی گردد.

Fast R-CNN ۲.۵.۲

پس از روش R-CNN پیشنهاد دهندگان آن تلاش کردند تا برخی از نقاط ضعف الگوریتم خود را بهبود داده و سرعت اجرای آن را افزایش دهند و به همین جهت نام آن را به عنوان Fast R-CNN انتخاب کردند [۴۱]. این الگوریتم نیز تا حدودی مشابه الگوریتم قبلی است با این تفاوت که به جای تزریق ناحیه‌های نامزد به شبکه تلفیقی، کل تصویر ورودی به شبکه تزریق می‌شود و سپس یک نگاشت ویژگی تلفیقی تولید می‌گردد. پس از آن با توجه به

[1] Selective Search
[2] Region Proposals
[3] Offsets Values



شکل ۶.۲: نحوه عملکرد الگوریتم R-CNN.

نگاشت ویژگی تولید شده و با استفاده از الگوریم جستجوی انتخابی، ناحیه‌های نامزد مشخص می‌شوند و توسط لایه Pooling به ابعاد مربعی و ثابت تبدیل شده و به یک لایه تماما متصل تزریق می‌گردند. حال با توجه به بردارهای ویژگی هریک از ناحیه‌های مورد نظر[1]، از یک لایه Softmax استفاده می‌شود تا کلاس ناحیه‌های نامزد و همچنین مقادیر انحراف جعبه محاطی متناظر با آن‌ها پیش‌بینی شوند.

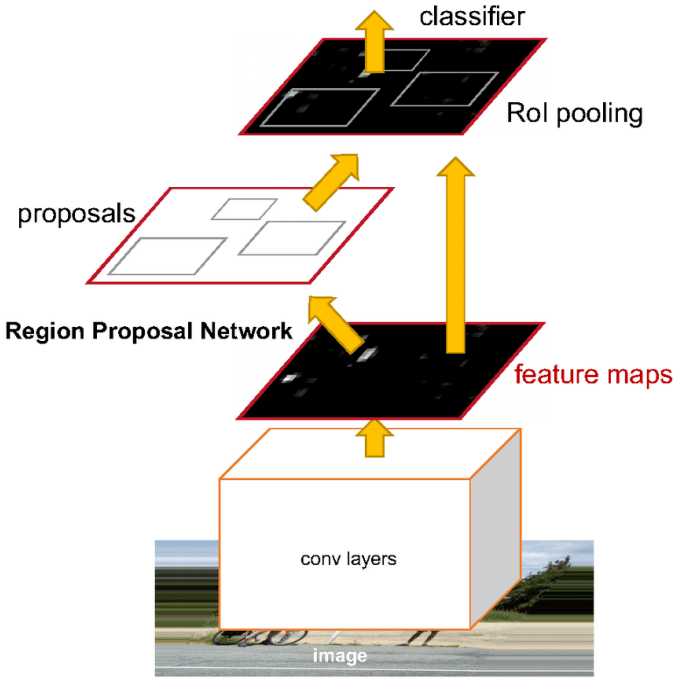
به‌طور خلاصه دلیل افزایش بسیار زیاد سرعت این الگوریتم نسبت به نمونه مشابه قبلی خود این است که در این نسخه نیازی نیست تا ۲۰۰۰ ناحیه نامزد در هر بار به شبکه‌های تلفیقی تزریق شوند و در عوض عملیات‌های تلفیقی به ازای هر تصویر تنها یک مرتبه اجرا شده و نگاشت ویژگی از آن‌ها حاصل می شود. سپس باید عملیات‌ها بر روی این نگاشت ویژگی‌های حاصل شده اجرا شود.

Faster R-CNN ۳.۵.۲

هر دو روش پیشنهاد شده از الگوریتم جستجوی انتخابی برای یافتن ناحیه‌های نامزد استفاده می‌کردند در حالی که این الگوریتم بسیار کند بوده و بر روی سرعت شناسایی به شدت تاثیرگذار است. به همین خاطر رِن و همکارانش در سال ۲۰۱۵ تلاش کردند تا این مرحله را از روش‌های موجود حذف کنند [۱۱۵]. روش کار این الگوریتم مشابه با روش Fast R-CNN است به‌طوری که ابتدا تصویر ورودی را به یک شبکه تلفیقی تزریق می‌کند تا نگاشت ویژگی

[1] Region of Interest (RoI)

متناظر با آن تولید شود. سپس به جای استفاده از الگوریتم جستجوی انتخابی بر روی نگاشت ویژگی حاصل شده برای یافتن نواحی نامزد، از یک شبکه مستقل دیگر برای پیش‌بینی آن‌ها استفاده می‌شود. پس از آن ابعاد نواحی نامزد پیش‌بینی شده با استفاده از لایه RoI Pooling مشخص می‌شود و بر این اساس جعبه‌های محاطی و مقادیر انحراف متناظرشان پیش‌بینی می‌گردد. با تغییر ایجاد شده سرعت الگوریتم بسیار بیش‌تر از نمونه اولیه خود شد تا جایی که حتی می‌توان از آن برای کاربردهای بلادرنگ نیز استفاده کرد.



شکل ۷.۲: نحوه عملکرد الگوریتم Faster R-CNN.

YOLO ۴.۵.۲

همه الگوریتم‌هایی که تا اینجا بررسی شدند بر اساس ناحیه‌های نامزد موقعیت اشیاء را مشخص می‌کردند. در این روش‌ها از تمام تصویر جز برای تولید نگاشت ویژگی و به منظور یافتن مکان‌هایی که احتمال وجود شئ در آن‌ها بیشتر است، استفاده نمی‌شد. YOLO[1] الگوریتمی است که در سال ۲۰۱۶ توسط ردمون و همکارانش پیشنهاد شد و نحوه عملکرد آن نسبت به روش‌های قبل از خود تا حدود زیادی متفاوت است [۱۱۲]. در این روش جعبه‌های محاطی و کلاس اشیاء موجود تنها با یک مرتبه تزریق به شبکه تلفیقی پیش‌بینی می‌شوند. نحوه عملکرد YOLO به این صورت است که ابتدا تصویر ورودی را به صورت $[S \times S]$ شبکه‌بندی می‌کند که در این الگوریتم معمولا $S = ۷$ در نظر گرفته می‌شود. سپس عملیات‌های دسته‌بندی و تعیین موقعیت به‌طور هم‌زمان بر روی این تعداد سلول

[1] You Only Look Once

ایجاد شده انجام می‌گیرد. از آن جایی که شبکه مربوط به دسته‌بندی و تعیین موقعیت در یک زمان تنها می‌توانند یک شئ را شناسایی کنند، بنابراین در هر سلول تنها یک شئ قابل شناسایی است.

شکل ۸.۲: تصویر به صورت [۳ × ۳] شبکه‌بندی شده و جعبه‌های محاطی حقیقی متناظر با اشیاء موجود در آن مشخص گردیده است.

حال اگر فرض شود که تعداد کلاس‌های اشیاء برابر n باشد آنگاه به ازای هر سلول به دست آمده برچسبی به صورت زیر می‌توان در نظر گرفت:

$$Y = [p_c, b_x, b_y, b_h, b_w, c_1, \ldots, c_n]^T, \qquad (۳-۲)$$

که در آن:

- p_c عبارت است از احتمال اینکه در سلول متناظر شئ‌ای وجود داشته باشد و استفاده از آن به خاطر جلوگیری از شناسایی پس‌زمینه تصاویر است. در مواقعی که شئ‌ای در سلول وجود نداشته باشد مقدار آن صفر است و در سایر موارد مایل هستیم تا مقدار آن برابر با مقدار IOU بین جعبه محاطی پیشنهادی و جعبه محاطی حقیقی باشد؛ زیرا از آن جایی که جعبه‌های محاطی به صورت دستی رسم می‌شوند و یقین وجود دارد که در آن‌ها شئ‌ای موجود است پس هرچقدر IOU نسبت به آن بیشتر باشد احتمال وجود شئ در آن جعبه محاطی پیشنهادی نیز بیشتر است.

- b_x، b_y، b_h و b_w در صورت وجود شئ در سلول مورد نظر، جعبه محاطی متناظر با آن را توصیف می‌کنند. به‌طوری که b_x و b_y مرکز شئ واقع شده در آن سلول یا جعبه محاطی متناظرش را مشخص می‌کنند و بر

اساس همان سلول مقدار آن‌ها بین صفر و یک نرمال می‌شود. همچنین مقادیر b_w و b_h به ترتیب میزان عرض و ارتفاع جعبه محاطی متناظر با آن را نشان می‌دهند که آن‌ها نیز بر اساس ابعاد تصویر در بازه $[۰, ۱]$ نرمال می‌گردند.

- مقادیر $c_۱, \ldots, c_n$ نیز احتمال تناظر اشیاء موجود در آن سلول را با هر یک از کلاس‌های مورد نظر مشخص می‌کنند.

با این کار می‌توان هر یک از سلول‌ها را با توجه به این برچسب‌ها به شبکه آموزش داد. به عنوان مثال فرض کنید سه کلاس از اشیاء مانند انسان ($c_۱$)، خودرو ($c_۲$) و موتورسیکلت ($c_۳$) ممکن است در تصاویر موجود باشند. هدف شناسایی این نوع اشیاء در تصاویر مورد نظر بوده و شکل (۸.۲) نیز یکی از تصاویر آموزشی باشد. حال به ازای هر شئ (در این تصویر خودرو) موجود در تصویر ابتدا نقطه‌ای به عنوان نقطه مرکزی شئ مورد نظر مشخص می‌شود. سپس سلولی که آن نقطه در آن واقع شده است به عنوان سلول مسئول برای شناسایی شئ مورد نظر انتخاب می‌گردد و برچسب متناظر با آن به صورت $Y = [۱, b_x, b_y, b_h, b_w, ۰, ۱, ۰]^T$ ایجاد خواهد شد. همانطور که واضح است چون در سلول‌ها شئ‌ای وجود دارد مقدار p_c آن‌ها برابر با ۱ و چون اشیاء متناظر با کلاس خودرو هستند مقدار $c_۲$ آن‌ها ۱ و بقیه برابر ۰ تعیین می‌گردد. همچنین مقادیر مربوط به جعبه محاطی آن‌ها به شیوه ذکر شده محاسبه می‌شود. به‌طور مشابه برای سایر سلول‌هایی که در آن‌ها شئ‌ای وجود دارد برچسب متناظر با آن‌ها تولید می‌شود. سلول‌هایی که فاقد هر گونه شئ‌ای هستند، برچسب متناظرشان به صورت $Y = [۰, \diamond, \diamond, \diamond, \diamond, \diamond, \diamond, \diamond]^T$ در نظر گرفته می‌شود. علامت $\diamond$ به معنی آن است که چون شئ‌ای در سلول وجود ندارد ($p_c = ۰$) لذا مقدار سایر مولفه‌ها اهمیتی نداشته و مقداری برای آن‌ها تعیین نمی‌شود. سپس هر یک از این سلول‌ها و برچسب‌های متناظرشان برای آموزش به شبکه تزریق می‌شوند. در اینجا توده ورودی شبکه به صورت $[۸ \times ۳ \times ۳]$ است؛ زیرا تعداد ۳×۳ سلول موجود است و برای هر سلول برچسبی با اندازه ۸ وجود دارد.

در الگوریتم YOLO به ازای هر سلول تعداد B تا جعبه محاطی وجود دارد که معمولا $B = ۲$ در نظر گرفته می‌شود. در این صورت به ازای هر جعبه محاطی تمامی متغیرهای موجود باید تکرار شود. در نهایت ابعاد توده ورودی را می‌توان به صورت $S \times S \times (B \times ۵ + n)$ در نظر گرفت به‌طوری که S تعداد سلول‌ها، B تعداد جعبه‌های محاطی متناظر با هر سلول و n تعداد کلاس‌ها است.

در نهایت توده حاصل شده به شبکه تلفیقی تزریق می‌شود تا شبکه آموزش ببیند به‌طوری که بتواند جعبه‌های محاطی اشیاء موجود و همچنین کلاس آن‌ها را به‌طور هم‌زمان مشخص کند. در این الگوریتم به‌طور پیش‌فرض از

چهارچوب Darknet [۱۱۱] و مجموعه داده ImageNet-1000 [۱۲۲] برای آموزش شبکه استفاده شده است. Darknet یک چهارچوب برای شبکه‌های عصبی بوده که از مدل معروف GoogLeNet [۱۴۱] الهام گرفته و به دلیل سرعت بالای آن برای کاربردهای بلادرنگ بسیار مفید است.

نکته‌ای که باید به آن توجه کرد این است که در برخی موارد اشیاء با مقیاس بزرگ در بیش از یک سلول قرار می‌گیرند و لذا ممکن است بیش از یک مرتبه شناسایی شوند. در این صورت سوالی که مطرح می‌شود این است که کدام یک از سلول‌های مورد نظر باید مسئول شناسایی آن شئ باشد و چگونه از شناسایی های مجدد جلوگیری شود؟ در این حالت به ازای کلاس‌هایی که در اختیار است باید به صورت زیر اقدام شود که به آن اصطلاحا non-maximum suppression گفته می‌شود.

۱. ابتدا تمامی جعبه‌های محاطی پیشنهادی که مقدار p_c آن‌ها از یک حد آستانه مانند ۰٫۵ کمتر است حذف می‌شوند.

۲. جعبه‌های محاطی باقی‌مانده بر اساس p_c ها به صورت نزولی مرتب می‌شوند.

۳. جعبه‌های محاطی پیشنهادی با بیشترین p_c به عنوان خروجی انتخاب می‌گردد.

۴. سپس سایر جعبه‌هایی که مقدار IOU آن‌ها با جعبه محاطی انتخاب شده از یک حد آستانه بیشتر باشد حذف می‌شوند.

۵. تا زمانی که تمامی box ها بررسی شوند از گام ۳ روند تکرار می‌شود.

همانطور که گفته شد هر سلول تنها می‌تواند یک شئ را شناسایی کند و از این رو بیشینه تعداد اشیاء قابل شناسایی در یک تصویر برابر با تعداد سلول‌های ایجاد شده است که ممکن است محدود باشد. همچنین اگر بیش‌تر از یک شئ در یک سلول قرار بگیرند یا به عبارت دیگر اشیاء دارای مقیاس کوچکی بوده و بسیار به یکدیگر نزدیک باشند، برای شناسایی آن‌ها نیز با مشکل مواجه خواهیم بود. علاوه بر این اگر در داده‌های آموزشی اشیاء موجود در تصاویر مقیاس کوچکی داشته باشند و شبکه بر اساس آن‌ها آموزش ببیند آنگاه در مورد شناسایی اشیاء یکسان که مقیاس بزرگ‌تری در تصاویر دارند عملکرد مناسبی نخواهد داشت. بر همین اساس همواره تلاش می‌شد تا دقت و همچنین سرعت این الگوریتم افزایش یابد که این تلاش‌ها منجر به معرفی دو نسخه دیگر از این الگوریتم شد. در ادامه به معرفی و تشریح تغییرات ایجاد شده در این نسخه‌ها پرداخته خواهد شد.

YOLOv2

الگوریتم اولیه YOLO دارای خطاهای موقعیت‌یابی نسبتا بالا و همچنین مقدار صحت پایینی بود و به همین دلیل در نسخه بعدی تلاش شد تا این نقاط ضعف برطرف شوند. نسخه بعدی از الگوریتم YOLO توسط ردمون و فرهادی در سال ۲۰۱۷ پیشنهاد شد که بهبودهای زیادی در سرعت و عملکرد خود داشت [۱۱۳]. به‌طور خلاصه تفاوت‌هایی که در این نسخه لحاظ شده است را می‌توان طبق گفته نویسندگان آن به صورت زیر دسته‌بندی و عنوان کرد:

- در این نسخه از الگوریتم در تمامی لایه‌های تلفیقی موجود در شبکه از Batch Normalizaion استفاده شده است که همین موضوع باعث همگرایی سریع‌تر شبکه شده و افزایش ۲ درصدی میانگین دقت متوسط (mAP) را به همراه داشته است. علاوه بر آن همانطور که در بخش‌های قبل نیز عنوان شد استفاده از Batch Normalizaion باعث جلوگیری از بیش برازش شبکه نیز می‌گردد.

- در تمامی الگوریتم‌های قبلی به منظور دسته‌بندی اشیاء در فرآیند شناسایی از مدل‌های از پیش آموزش داده شده بر روی مجموعه داده ImageNet-1000 [۱۲۲] استفاده شده است و اندازه تصاویر استفاده شده معمولا کمتر از [۲۵۶ × ۲۵۶] بوده است [۷۵]. در الگوریتم YOLO اولیه تصاویر استفاده شده برای آموزش دسته‌بندی کننده دارای ابعاد [۲۲۴ × ۲۲۴] بوده‌اند اما تصاویر ورودی شبکه برای انجام عملیات شناسایی ابعاد [۴۴۸ × ۴۴۸] داشته‌اند. این موضوع به این معناست که فیلترهای شبکه در زمان آموزش برای انجام عملیات شناسایی باید به‌طور هم‌زمان برای تصاویر با ابعاد بزرگ‌تر نیز تنظیم شود. اما در این نسخه پس از آموزش شبکه دسته‌بندی کننده با تصاویر [۲۲۴ × ۲۲۴] تایی، از تصاویر با ابعاد [۴۴۸ × ۴۴۸] و به اندازه ۱۰ مرحله (epoch) قبل از آموزش شناسایی کننده استفاده شده است. همین امر باعث شده تا فیلترهای شبکه برای انجام عملیات شناسایی بهتر عمل کنند و باعث افزایش ۴ درصدی mAP گردند.

- در YOLOv2 به منظور اضافه کردن قابلیت شناسایی اشیاء در تصاویر با ابعاد مختلف مانند $\{۳۲۰, ۳۵۲, ۳۸۴, \ldots, ۶۰۸\}$، از تصاویر با ابعاد مختلف نیز برای آموزش شبکه استفاده شده است. به‌طوری که پس از تعدادی تکرار در شبکه ابعاد تصاویر ورودی و به تبع آن متغیرهای لایه‌های تلفیقی و Pooling شبکه عوض شده و بر این اساس آموزش صورت می‌گیرد. با این کار شبکه حاصل شده قابلیت شناسایی اشیاء در تصاویر با ابعاد مختلف را خواهد داشت.

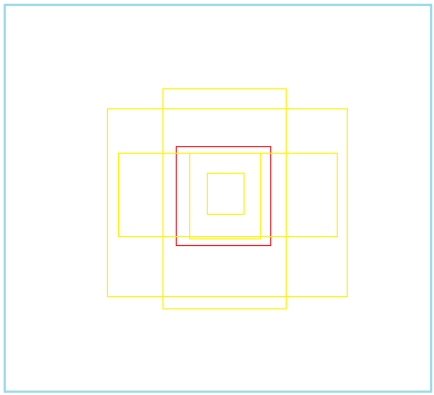
- در الگوریتم YOLO اولیه مشاهده شد که هر سلول تنها توانایی شناسایی یک شئ را دارد و در صورتی که

مرکز دو شئ مختلف در یک سلول یکسانی قرار بگیرد الگوریتم تنها یکی از آن‌ها را شناسایی می‌کند. در نسخه جدید از این الگوریتم تلاش شد تا با استفاده از K تا جعبه محاطی مختلف، امکان شناسایی چند شئ در یک سلول فراهم شود. به همین منظور از ایده‌ای با عنوان جعبه محوری[1] استفاده کردند. در نسخه قبلی محاسبه مختصات جعبه‌های محاطی مستقیما توسط لایه‌های تماما متصل در انتهای شبکه انجام می‌شد. از طرفی در روش‌های قبلی مانند Faster R-CNN پیش‌بینی جعبه‌های محاطی با استفاده از جعبه‌های محوری به دست می‌آمد که به صورت دستی مشخص می‌شدند. به‌طور کلی منظور از جعبه محوری مجموعه‌ای از جعبه‌ها یا کادرهایی هستند که از قبل مشخص می‌شوند و انتظار می‌رود اشیاء موجود در تصاویر در داخل آن‌ها قرار گیرند. با این کار به جای اینکه جعبه‌های محاطی نسبت به کل تصویر پیش‌بینی شوند، با توجه به جعبه‌های محوری از قبل در نظر گرفته شده مشخص می‌شوند و نشان داده شده که با استفاده از این ایده شبکه بهتر می‌تواند جعبه‌های محاطی را یاد گرفته و مشخص کند. همانطور که گفته شد این ایده قبلا در روش Faster R-CNN نیز به کار گرفته می‌شد به‌طوری که این جعبه‌های محوری به‌طور دستی معین و سپس با محاسبه مقادیر انحراف، جعبه‌های محاطی به اندازه مورد نیاز تنظیم می‌شدند. در این نسخه از الگوریتم YOLO محققان تصمیم به انجام کاری مشابه با آن گرفتند با این تفاوت که ابعاد آن‌ها به‌طور موثرتری تعیین شود. برای این کار آن‌ها الگوریتم k-نزدیک‌ترین همسایه (K-means) را بر روی مجموعه جعبه‌های محاطی حقیقی در داده‌های آموزشی و با مقادیر مختلف k اجرا کردند و میانگین IOU را با توجه به نقطه مرکزی آن‌ها رسم کردند. برای این کار به جای استفاده از معیار فاصله اقلیدسی از معیار IOU بین جعبه‌های محاطی و نقاط مرکزی به دست آمده استفاده کردند. با آزمایشات انجام شده مناسب‌ترین مقدار k برابر با ۵ در نظر گرفته شد و به این طریق جعبه‌های محوری بهینه محاسبه شدند که در اینجا نویسندگان آن‌ها را جعبه‌های اولیه[2] می‌نامند. پس از آن با محاسبه مقادیر انحراف می‌توان جعبه‌های محاطی مناسب‌تری را برای اشیاء موجود در هر سلول مشخص کرد.

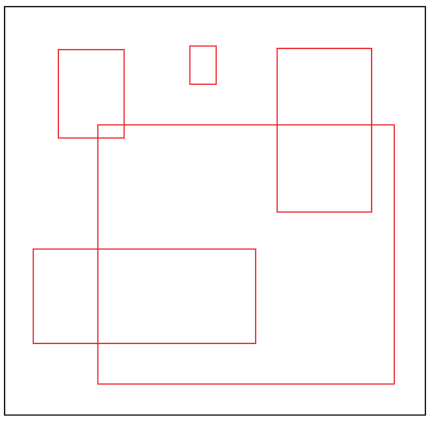
- در این مدل از شبکه Darknet-19 برای آموزش شبکه دسته‌بندی استفاده شده است. این شبکه دارای ۱۹ لایه تلفیقی و ۵ لایه Max Pooling است و از یک لایه Softmax استفاده می‌کند. هدف کلی از طراحی این شبکه کاهش پیچیدگی‌های محاسباتی و افزایش دقت شناسایی بوده است.

این تغییرات به وجود آمده که جوانب مختلفی را نیز در برداشته باعث بهبود قابل توجهی در عملکرد این روش

[1] Anchor Box
[2] Prior Boxes



(آ) سلول مورد نظر (ناحیه قرمز رنگ) و ۵ جعبه محوری متناظر با آن در ابعاد مختلف



(ب) میانگین ابعاد و موقعیت جعبه‌های محاطی موجود در تصاویر مجموعه داده VOC 2007

شکل ۹.۲: جعبه‌های محوری به دست آمده در الگوریتم YOLOv2 و نحوه استفاده از آن‌ها برای سلول‌ها.

شده است به‌طوری که تا حدودی تمامی مشکلات نسخه قبلی خود را پوشش داده است.

YOLO9000

در نسخه‌های قبلی الگوریتم YOLO تعداد کلاس اشیاء قابل شناسایی برابر با ۲۰ بود. مسلما این تعداد کلاس برای کاربردهای عمومی بسیار محدود است و تنها می‌تواند کلاس کلی اشیاء را شناسایی کند. به عنوان مثال در میان چند نوع سگ مختلف، الگوریتم تنها قادر است مشخص کند که شئ شناسایی شده متعلق به کلاس سگ است و نوع آن قابل تعیین نیست. دلیل این امر عدم غنی بودن مجموعه داده‌های آموزشی برای شناسایی است در حالی که مجموعه داده‌های مربوط به دسته‌بندی تصاویر بسیار غنی‌تر و کامل‌تر هستند. الگوریتم YOLO9000 در ادامه

الگوریتم YOLOv2 ارائه شده است. نویسندگان پس از معرفی YOLOv2 روشی را معرفی می‌کنند تا بتوان داده‌های آموزشی مربوط به دسته‌بندی و داده‌های آموزشی مربوط به شناسایی را با یکدیگر ادغام کرده و از این جهت بتوان یک شبکه شناسایی با قابلیت شناسایی اشیاء بیشتری را معرفی کرد. در نهایت با روش‌های گفته شده امکانی فراهم شده که شبکه قادر است بیش از ۹۰۰۰ دسته از اشیاء را شناسایی کند و لذا این شبکه با عنوان YOLO9000 نام‌گذاری می‌شود [۱۱۳].

YOLOv3

از آنجایی که رقابت مدل‌ها و روش‌های مختلف در شناسایی اشیاء بر اساس دقت و سرعت آن‌هاست، این نسخه نیز با هدف افزایش دقت و سرعت شناسایی اشیاء و با تغییرات دیگری توسط محققان نسخه قبلی در سال ۲۰۱۸ معرفی شد [۱۱۴]. در این نسخه از الگوریتم YOLO می‌توان تغییرات لحاظ شده را به صورت زیر عنوان کرد:

- در این نسخه از YOLO مقدار p_c برای هر یک از جعبه‌های محاطی با استفاده از رگرسیون لجستیک[1] محاسبه می‌شود و همچنین نحوه محاسبه تابع هزینه[2] نیز تغییر یافته است. بر این اساس زمانی که یک جعبه اولیه با جعبه محاطی حقیقی موجود بیش‌ترین IOU را داشته باشد آنگاه مقدار p_c برابر با ۱ خواهد بود و سایر جعبه‌های اولیه که IOU آن‌ها با جعبه محاطی حقیقی بیش‌تر از یک حد آستانه از پیش معین شده (به‌طور پیش فرض ۰٫۵) باشد، هزینه‌ای متحمل نمی‌شوند. علاوه بر این هر جعبه محاطی حقیقی تنها به یک جعبه اولیه منتسب می‌شود.

- در برخی موارد ممکن است برای شناسایی اشیاء به بیش از یک برچسب نیاز داشته باشیم. به عنوان مثال می‌خواهیم علاوه بر تشخیص انسان بودن یک فرد در تصویر، جنسیت آن نیز مشخص شود. الگوریتم‌های پیشین با استفاده از Softmax در لایه آخر شبکه تنها قادر به تعیین یک برچسب برای اشیاء شناسایی شده بودند. اما در YOLOv3 با استفاده از دسته‌بندی کننده‌های لجستیک مستقل[3] برای هر کلاس امکان تخصیص بیش از یک برچسب به اشیاء شناسایی شده و به‌طور هم‌زمان فراهم شد. همچنین به جای استفاده از تابع هزینه میانگین مربعات[4] در دسته‌بندی تصاویر، از تابع اختلاف آنتروپی دودویی[5] استفاده شده است که با این کار میزان پیچیدگی محاسباتی حاصل از Softmax نیز کاهش می‌یابد.

[1] Logistic Regression (LR)
[2] Cost Function
[3] Independent Logistic Classifiers (ILC)
[4] Mean Square Error
[5] Binary Cross-Entropy

- برخلاف YOLOv2 که خروجی‌ها را تنها در لایه آخر پیش‌بینی می‌کرد، در این نسخه جعبه‌ها در سه مقیاس مختلف پیش‌بینی می‌شوند. در هر مقیاس YOLOv3 از سه جعبه اولیه استفاده می‌کند و برای هر سلول نیز سه جعبه را پیش‌بینی می‌کند. در اینجا نیز همچنان هر شئ به یک سلول منتسب می‌شود.
- YOLOv3 نیز برای انتخاب جعبه‌های اولیه از الگوریتم خوشه‌بندی K-means استفاده می‌کند و تعداد ۹ جعبه اولیه را در نظر می‌گیرد. سپس آن‌ها بر اساس ابعادی که دارند به ۳ گروه متفاوت تقسیم می‌شوند و هر کدام از دسته‌ها برای شناسایی یکی از سه مقیاس ذکر شده در مورد قبلی استفاده می‌شود.
- در YOLOv3 شبکه Darknet-53 با ۵۳ لایه تلفیقی به عنوان استخراج کننده ویژگی جایگزین شبکه Darknet-19 شده است. Darknet-53 مانند شبکه Residual Network (ResNet) [۵۲] شامل فیلترهایی با ابعاد [۳×۳] و [۱ × ۱] است اما به دلیل استفاده از تعداد متغیرهای کمتر، از سرعت بیشتری نسبت به ResNet برخوردار است.

۵.۵.۲ SSD

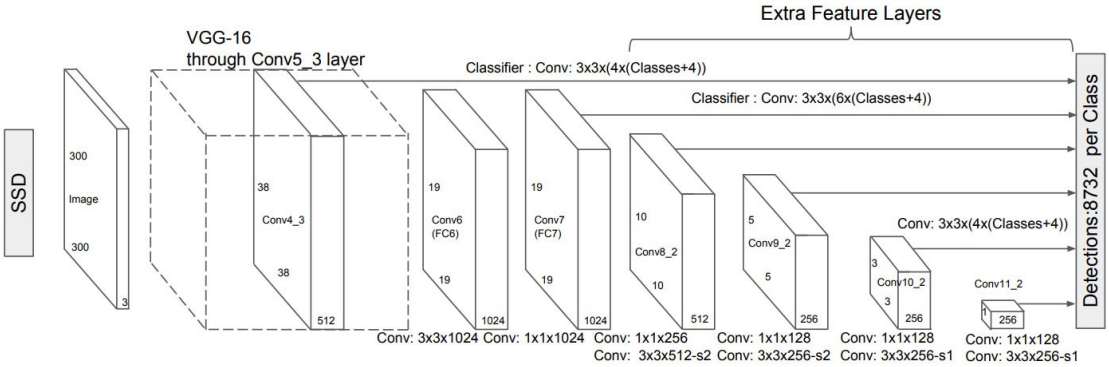
یکی دیگر از الگوریتم‌های شناخته شده در حوزه شناسایی اشیاء SSD[1] نام دارد که توسط لیو و همکارانش در سال ۲۰۱۶ ارائه شد [۹۰]. این الگوریتم نیز به شیوه‌ای عمل می‌کند که بر خلاف مدل‌های مبتنی بر R-CNN نیازی به استفاده از ناحیه‌های نامزد ندارد. به‌طور کلی عملیات شناسایی اشیاء در روش SSD را می‌توان به دو گام اساسی تقسیم‌بندی کرد. در گام اول نگاشت ویژگی‌ها استخراج شده و در گام دوم فیلترهای تلفیقی بر روی آن‌ها و به منظور شناسایی اشیاء اعمال می‌شود. در الگوریتم SSD برای استخراج نگاشت ویژگی‌ها از شبکه تلفیقی VGG16 [۱۳۱] که برای دسته‌بندی تصاویر طراحی شده بود استفاده می‌شود اما لایه‌های تماما متصل انتهایی آن حذف شده و با لایه‌های تلفیقی کمکی[2] جایگزین شده‌اند. به این ترتیب امکان استخراج ویژگی‌ها در مقیاس‌های مختلف و با کاهش تدریجی ابعاد توده‌های ورودی به لایه‌های بعدی فراهم می‌شود و سپس با استفاده از آن‌ها عملیات شناسایی صورت می‌گیرد.

در SSD برای عملیات شناسایی و محاسبه امتیاز کلاس‌ها از فیلترهای تلفیقی کوچک که معمولا دارای ابعاد [۳×۳] هستند، استفاده می‌شود. سپس این فیلترها بر روی هر یک از سلول‌های نگاشت ویژگی‌های استخراج شده اعمال می‌گردد (همانند فرآیند convolve در شبکه‌های تلفیقی). از این طریق امتیاز دسته‌های موجود و همچنین

[1] Single Shot Multibox Detector
[2] Auxiliary Convolutional Layer

مقادیر انحراف جعبه‌های از پیش تعیین شده محاسبه می‌شود. سپس به منظور افزایش دقت شناسایی، فرآیند ذکر شده بر روی نگاشت ویژگی‌ها با مقیاس‌های چندگانه[1] اجرا می‌شود و مجددا خروجی‌های متناظر با هر یک از سلول‌ها و در هر یک از نگاشت ویژگی‌های مورد نظر به دست می‌آید. SSD از نگاشت ویژگی‌های با ابعاد و وضوح کمتر برای شناسایی اشیاء بزرگ‌تر و از نگاشت ویژگی‌های با ابعاد و وضوح بیش‌تر برای شناسایی اشیاء کوچک‌تر استفاده می‌کند. به عبارت دیگر در طول شبکه و همزمان با کاهش تدریجی ابعاد نگاشت ویژگی‌ها ابتدا اشیاء با مقیاس کوچک‌تر و در مراحل بعدی اشیاء با مقیاس بزرگ‌تر شناسایی می‌شوند.

همانطور که گفته شد در انتهای شبکه VGG16 لایه‌های تلفیقی کمکی اضافه شده که تعداد آن‌ها برابر با ۶ است به‌طوری که ۵ تا از آن‌ها به منظور شناسایی استفاده می‌شود. در سه تا از این لایه‌ها به جای ۴ پیش‌بینی، ۶ پیش‌بینی انجام می‌شود که در نهایت تعداد کل پیش‌بینی‌های انجام گرفته به ازای هر کلاس برابر با ۸۷۳۲ است. در SSD

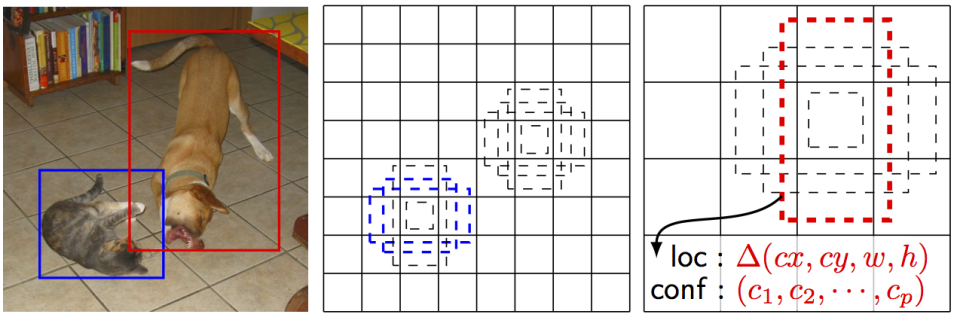


شکل ۱۰.۲: معماری شبکه طراحی شده برای الگوریتم SSD. در اینجا ۶ لایه تلفیقی به شبکه VGG16 اضافه شده و به واسطه آن‌ها شناسایی اشیاء صورت می‌گیرد. در نهایت نیز به ازای هر کلاس تعداد ۸۷۳۲ پیش‌بینی به دست می‌آید.

پیش‌بینی‌های صورت گرفته به دو دسته تطابق‌های مثبت و تطابق‌های منفی تقسیم می‌شوند به‌طوری که در این روش تنها از تطابق‌های مثبت برای محاسبه هزینه موقعیت‌یابی اشیاء استفاده می‌شود. در این شیوه در فرآیند آموزش اگر bounding box مشخص شده با ground truth موجود دارای IOU بیشتر از ۰٫۵ باشد، تطابق شکل گرفته مثبت و در غیر این صورت منفی در نظر گرفته خواهد شد. با استفاده از این استراتژی هر یک از پیش‌بینی کننده‌ها تشویق می‌شوند تا از بین bounding box های موجود گزینه‌ای که برای شئ مورد نظر مناسب‌تر است را انتخاب کنند.

نکته‌ای که باید به آن توجه شود این است که در این روش نمونه‌های منفی بسیار بیشتر از نمونه‌های مثبت

[1] Multi-Scale Features Map



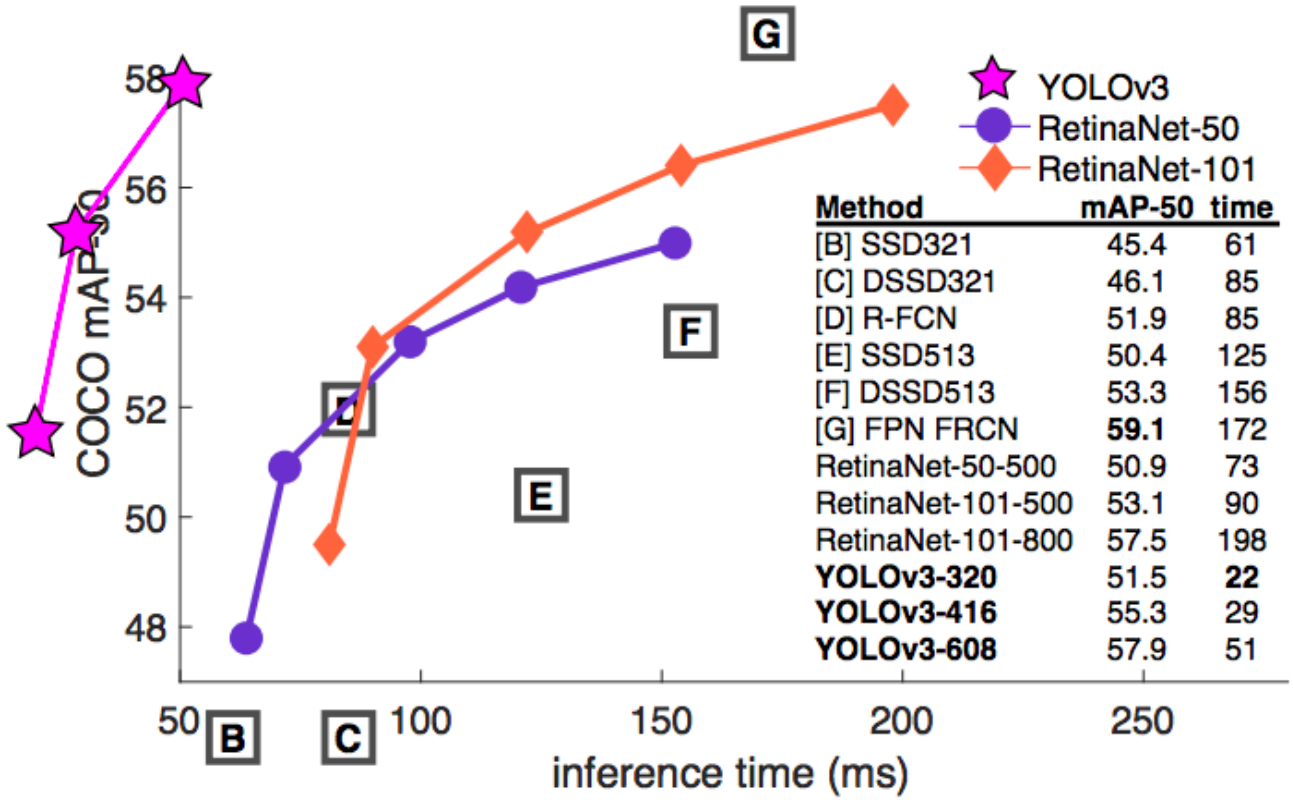
(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

شکل ۱۱.۲: در این تصویر دو نگاشت ویژگی با ابعاد [۸ × ۸] و [۴ × ۴] نشان داده شده است. جعبه‌های مورد نظر در هر یک از سلول‌های این نگاشت ویژگی‌ها حرکت کرده و به ازای هرکدام امتیاز کلاس‌ها و مقادیر انحراف نسبت به جعبه‌ها محاسبه می‌شود. در اینجا جعبه‌های مناسب که با رنگ‌های قرمز و آبی نشان داده شده‌اند به عنوان نمونه‌های مثبت و سایر آن‌ها به عنوان نمونه‌های منفی در نظر گرفته می‌شوند.

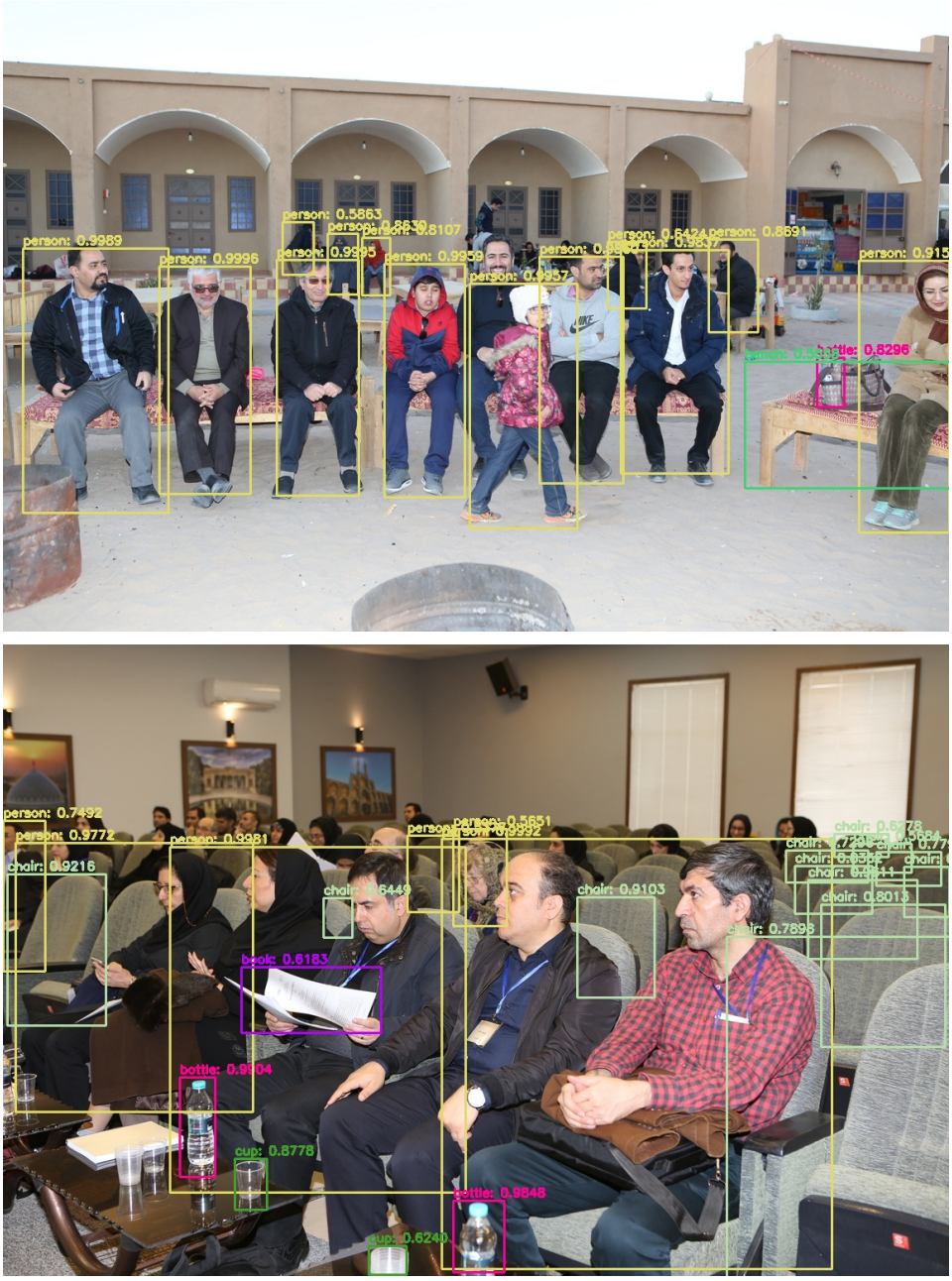
هستند که باعث می‌شود شبکه به سمت نمونه‌های منفی متمایل شود. به همین دلیل در این الگوریتم به جای استفاده از تمامی نمونه‌های منفی، ابتدا آن‌ها بر اساس ضریب خطای حاصل شده مرتب شده و نمونه‌های با ضریب خطای بالاتر انتخاب می‌شوند به‌طوری که نسبت نمونه‌های مثبت به نمونه‌های منفی حدود یک به سه باشد. این کار علاوه بر جلوگیری از متمایل شدن شبکه، باعث افزایش سرعت و پایداری آن نیز خواهد شد. در SSD نیز از تکنیک non-maximum suppresion استفاده می‌شود تا از بروز شناسایی کننده‌های تکراری برای یک شئ یکسان جلوگیری شود و در بین آن‌ها بهترین پیش‌بینی کننده را انتخاب کند.

یکی از نقاط ضعف این الگوریتم این است که همانطور که گفته شد اشیاء با مقیاس کوچک در لایه‌های ابتدایی که دارای ابعاد و وضوح بیشتری هستند شناسایی می‌شوند اما در این لایه‌ها ویژگی‌هایی که می‌توان به دست آورد ویژگی‌های سطح پایین مانند رنگ، یال‌ها و لبه‌ها هستند که اطلاعات کم اهمیتی برای دسته‌بندی محسوب می‌شوند و لذا در مورد شناسایی اشیاء با مقیاس‌های کوچک کمی با مشکل مواجه خواهد بود. در شکل (۱۲.۲) روش‌های مختلف برای شناسایی اشیاء از نظر دقت و سرعت آن‌ها با یکدیگر مقایسه شده‌اند.

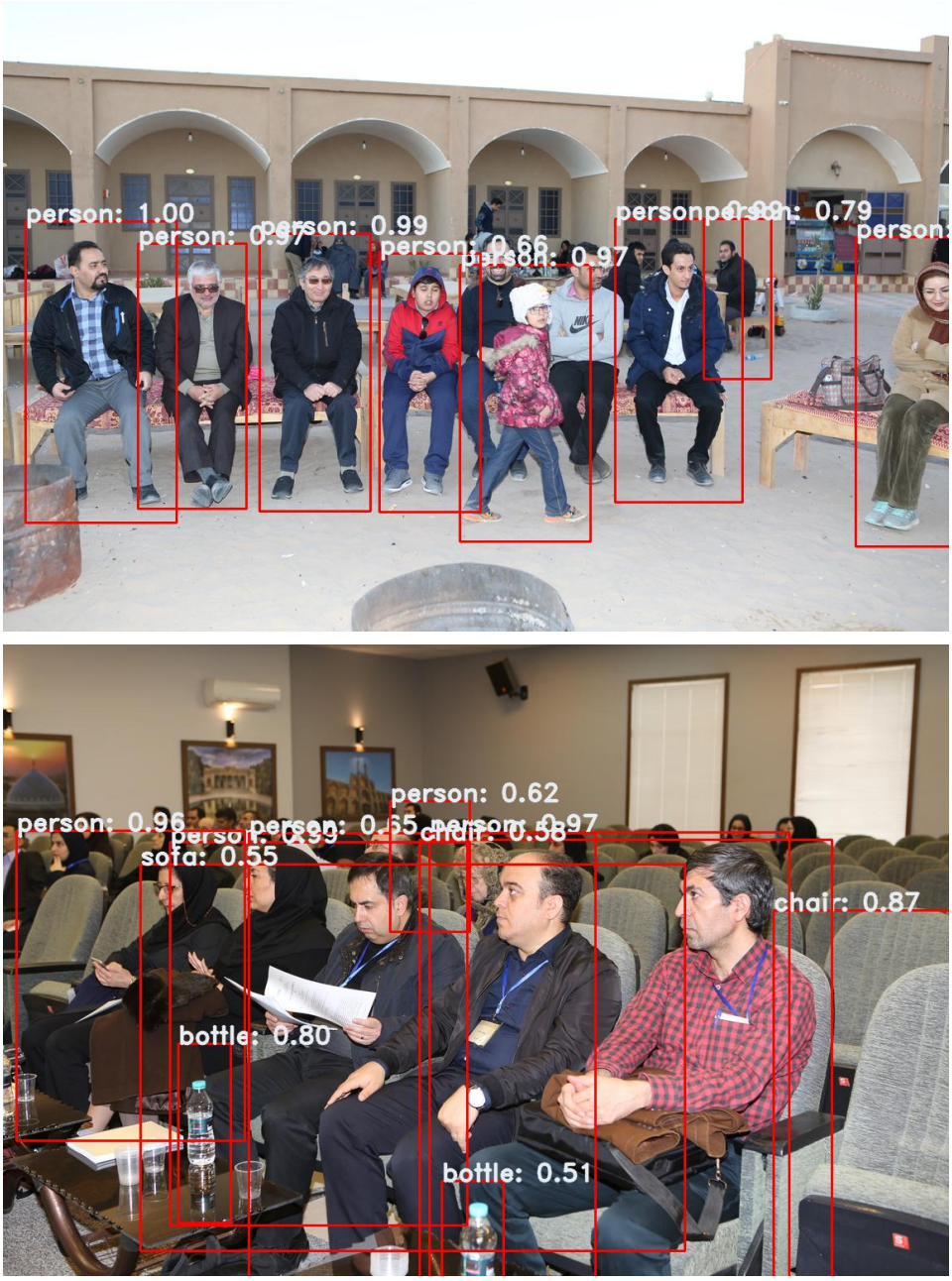
پیاده‌سازی و مقایسه روش‌های YOLO و SSD



شکل ۱۲.۲: مقایسه الگوریتم‌های مختلف در شناسایی اشیاء از نظر دقت و سرعت [۱۱۴].



شکل ۱۳.۲: نتایج پیاده‌سازی الگوریتم YOLOv3 برای شناسایی اشیاء موجود در تصاویر.



شکل ۱۴.۲: نتایج پیاده‌سازی الگوریتم SSD برای شناسایی اشیاء موجود در تصاویر.

فصل ۳
ردیابی اشیاء

۱.۳ مقدمه

مسئله ردیابی اشیاء در حال حاضر یکی از پر چالش‌ترین مسائل در حوزه بینایی ماشین است که توجهات زیادی را جلب کرده و مطالعات بسیاری در مورد آن صورت می‌گیرد. یکی از دلایل این موضوع کاربردهای بسیار و متنوعی است که با استفاده از آن حاصل می‌شود و می‌تواند دنیای پیرامون انسان‌ها را با تحولات زیادی روبه‌رو کند. به‌عنوان مثال ماشین‌های خودران یکی از مسائلی هستند که در آینده‌ای نه چندان دور جزو غیر قابل انکار از زندگی ما می‌شوند و موضوع ردیابی اشیاء قسمت مهمی از آن است. تحلیل و بررسی بازی‌های ورزشی با هدف به دست آوردن آمارهای مختلف برای تماشاگران و مربیان [۸۹، ۳۸]، ردیابی حرکات دست و صورت در واقعیت مجازی[۲] به منظور استفاده از ماشین‌ها در امور مختلف [۴۹، ۱۳۵، ۱۳۴، ۹۵]، به دست آوردن مسیر حرکت موجودات و پرندگان به منظور تحلیل رفتارهای آن‌ها توسط دانشمندان [۱۳۸] و استفاده‌های متعدد در علوم پزشکی [۹۴، ۳، ۴۴] از نمونه‌هایی هستند که باعث می‌شود اهمیت همه جانبه مسئله ردیابی در حوزه‌های مختلف بیش از پیش نمایان شود و این به معنی آن است که باید توجهات ویژه‌ای به آن شود. به طور معمول الگوریتم‌های ردیابی اشیاء بسیار سریع‌تر از الگوریتم‌های مربوط به شناسایی هستند؛ زیرا هنگامی که شئ‌ای در قاب‌های قبلی شناسایی می‌شود، اطلاعات زیادی مانند موقعیت قرارگیری آن، جهت حرکت و همچنین اطلاعاتی درباره ظاهر آن در دسترس قرار می‌گیرد. علاوه بر این در برخی مواقع مانند مسدود شدن شئ توسط اشیاء دیگر یا محیط پس‌زمینه، تغییرات زاویه دید، تغییرات مقیاس اشیاء، کاهش وضوح و مواردی از این قبیل، الگوریتم‌های شناسایی ممکن است موفق به شناسایی آن شئ نشوند اما یک ردیاب‌کننده قوی تا حدودی نسبت به این نوع مسائل مقاوم است.

به طور کلی مسئله ردیابی اشیاء شامل مراحل زیر می‌شود:

[2] Virtual Reality

۱. گرفتن یک مجموعه اولیه از اشیاء شناسایی شده (مانند یک مجموعه از مختصات جعبه‌های محاطی).

۲. تخصیص یک شناسه منحصر به فرد برای هر یک از اشیاء شناسایی شده (در صورت شناسایی و ردیابی چند شئ به طور هم‌زمان).

۳. ردیابی اشیاء در طی قاب های متوالی فایل ویدئویی و حفظ شناسه تخصیص داده شده برای هر یک از اشیاء در طول ردیابی.

نکته مهمی که در این قسمت باید مطرح شود و معمولا یک نقطه ابهام برای افراد تازه‌کار در این زمینه محسوب می‌شود این است که نمی‌توان با انجام عملیات شناسایی در تمامی قاب‌های فایل ویدئویی مسئله ردیابی را حل کرد؛ زیرا همانطور که گفته شد در ردیابی نیاز داریم تا یک شناسه منحصر به فرد به هر شئ تخصیص دهیم و آن شناسه تا زمانی که شئ در قاب‌ها وجود دارد حفظ گردد یا به عبارت دیگر اشیاء مختلف هر چند که از یک کلاس باشند باید از یکدیگر متمایز شوند. این در حالی است که در مسئله شناسایی، اشیاء در هر قاب به طور مستقل شناسایی می‌شوند و اطلاعات یک تصویر یا یک قاب هیچ‌گونه ارتباطی با سایر قاب‌ها ندارد. بنابراین تنها با اعمال الگوریتم‌های شناسایی به ازای تمامی قاب‌های فایل ویدئویی نمی‌توان به ردیابی مورد نظر دست یافت و این کار نیازمند الگوریتم‌های مخصوص به خود است.

یک الگوریتم ایده‌آل برای ردیابی بایستی شامل موارد زیر باشد:

- تنها یک مرتبه و هنگامی که شئ برای اولین بار در تصویر رؤیت می‌شود به فاز شناسایی نیاز داشته باشد.
- به طور قابل قبولی سریع باشد یا به عبارت دیگر نسبت به حالتی که از شناسایی کننده شئ استفاده می‌کنیم سریع‌تر عمل کند.
- قابلیت تشخیص خروج اشیاء از تصویر را داشته باشد و بتواند عملکرد مناسبی از خود نشان دهد.
- در مقابل مسائلی مانند انسداد[۱]، تغییرات زاویه دید، تغییرات روشنایی، تغییرات مقیاس و مواردی از این قبیل مقاوم باشد.
- زمانی که اشیاء برای مدتی در یک یا چند قاب آشکار نیستند، با آشکار شدن مجدد آن‌ها بتواند ردیابی سابق را ادامه دهد و اطلاعات به دست آمده از دست نرود.

[۱] منظور از انسداد حالتی است که تمام یا قسمتی از شئ هدف توسط سایر اشیاء پوشانده می‌شود.

معمولا اغلب روش‌های موجود برای ردیابی اشیاء بر اساس دو مدل مختلف در اشیاء آن‌ها را ردیابی می‌کنند. مدل اول مدل حرکت[1] نام دارد که بر اساس آن ردیاب می‌تواند موقعیت‌های احتمالی اشیاء در قاب‌های بعدی را پیش‌بینی کند و لذا فضای جستجوی خود را کاهش دهد [۷۷]. البته این مدل به تنهایی نمی‌تواند کافی باشد زیرا در مواقعی که شئ حرکات ناگهانی و بدون الگو داشته باشد یا حتی سرعت حرکت آن تغییر کند ممکن است ردیاب با شکست مواجه شود. مدل دوم مدل ظاهری[2] نام دارد که در آن باید شئ با محیط پس‌زمینه متمایز شود [۸۶]. اغلب روش‌هایی که عملکرد بهتری دارند و در آن‌ها تنها یک شئ ردیابی می‌شود از این مدل استفاده می‌کنند اما برای مواقعی که قرار است چند شئ به طور هم‌زمان ردیابی شوند این مدل به تنهایی کافی نیست.

۲.۳ دسته‌بندی الگوریتم‌های ردیابی

مسئله ردیابی اشیاء حالت‌های متفاوتی می‌تواند داشته باشد که بر اساس آن‌ها نیز الگوریتم‌های متفاوتی طراحی و توسعه داده می‌شود. بسته به کاربردی که مد نظر است و نوع ردیابی که باید انجام گیرد می‌توان الگوریتم مناسب را انتخاب و از آن استفاده کرد. به طور کلی انواع حالت‌ها و الگوریتم‌های ردیابی اشیاء را می‌توان بر اساس موارد زیر تقسیم‌بندی نمود:

۱. وجود یا عدم وجود فاز شناسایی

در الگوریتم‌های ردیابی که دارای فاز شناسایی هستند قاب‌های متوالی از فایل ویدئویی به یک شبکه از پیش آموزش دیده برای شناسایی اشیاء ارائه می‌شوند و موقعیت اشیاء معین می‌گردد. در این الگوریتم‌ها با ورود یک شئ جدید شناسایی اولیه به صورت خودکار انجام می‌شود و همچنین با خروج از قاب، ردیابی آن نیز پایان می‌پذیرد. به عبارت دیگر در این روش‌ها فاز شناسایی پس از هر n قاب یک مرتبه اجرا می‌شود و در سایر قاب‌ها این وظیفه الگوریتم‌های ردیابی است تا شئ را دنبال کنند. در حالتی که الگوریتم ردیابی دارای فاز شناسایی نباشد، شناسایی اولیه اشیاء در قاب ابتدایی باید به صورت دستی انجام شود و پس از آن الگوریتم ردیابی آن‌ها را دنبال می‌کند. واضح است که این الگوریتم‌ها نمی‌توانند اشیاء جدیدی که در قاب‌های بعدی وارد می‌شوند را ردیابی کنند.

۲. ردیابی یک شئ در مقابل چند شئ

[1] Motion Model
[2] Appearance Model

در برخی مواقع نیاز داریم تا تنها یک شئ خاص را دنبال کنیم. به‌عنوان مثال ردیابی یک خودرو در خیابان یا ردیابی یک شخص خاص در پیاده‌رو را در نظر بگیرید. در این مواقع می‌توان از الگوریتم‌های بدون فاز شناسایی استفاده کرد به طوری که در قاب اولیه دلخواه موقعیت آن شئ به طور دستی مشخص شود و سپس همواره شئ مورد نظر از محیط اطراف و پس‌زمینه خود متمایز گردد. علاوه بر این در برخی از کاربردها نیاز است تا کلیه اشیاء موجود در فایل ویدئویی و به طور هم‌زمان ردیابی شوند. در این حالت معمولا از الگوریتم‌هایی که دارای فاز شناسایی هستند استفاده می‌گردد تا بتواند ورود و خروج اشیاء به قاب‌های مختلف را مدیریت کند.

۳. ردیابی برخط در مقابل ردیابی برون‌خط

منظور از ردیابی برون‌خط شرایطی است که یک فایل ویدئویی از قبل ضبط شده و هدف ردیابی اشیاء موجود در آن باشد. به‌عنوان مثال ویدئویی از یک مسابقه فوتبال ضبط شده و هدف فعلی ما تحلیل و بررسی بازی با ردیابی بازیکنان آن است. در این حالت اگر یک قاب دلخواه را در نظر بگیریم، می‌توان به قاب یا قاب‌های قبل و بعد از آن نیز دسترسی داشت که همین امر باعث می‌شود اطلاعات بیشتری در مورد اشیاء در اختیار باشد و لذا ردیابی به طور دقیق‌تری صورت پذیرد. از طرف دیگر در مواقعی نیاز است تا اشیاء به صورت زنده و بلادرنگ ردیابی شوند و اطلاعاتی در مورد قاب‌های بعدی در دسترس نیست. مشخص است که در این حالت علاوه بر وجود اطلاعات کمتر، الگوریتم‌های طراحی شده نیز باید به میزان کافی سریع باشند تا بتوانند قاب‌های متوالی را به سرعت پردازش نمایند.

۴. یادگیری برخط در مقابل یادگیری برون‌خط

در الگوریتم‌های ردیابی که مبتنی بر یادگیری برخط هستند پس از مشخص کردن شئ مورد نظر در قاب اولیه، الگوریتم‌ها با استفاده از این اطلاعات و یا اطلاعات چند قاب بعد از آن یاد می‌گیرند تا شئ مورد نظر را دنبال کنند. این الگوریتم‌ها عمومی‌تر هستند؛ زیرا می‌توان ناحیه هر شئ دلخواهی را در قاب اولیه تعیین کرد و سپس ردیاب مورد نظر آن را دنبال می‌کند. این در حالی است که الگوریتم‌های ردیابی مبتنی بر یادگیری برون‌خط بر خلاف حالت قبلی در زمان ردیابی هیچ اطلاعاتی را یاد نمی‌گیرند و از اطلاعاتی که از قبل یاد گرفته‌اند استفاده می‌کنند. به‌عنوان مثال این ردیاب‌ها آموزش داده شده‌اند تا تنها افراد موجود در قاب‌ها را ردیابی کنند و لذا قابلیت ردیابی سایر اشیاء را نداند.

در ادامه به طور مختصر به برخی از این الگوریتم‌ها و نحوه عملکرد آن‌ها پرداخته خواهد شد.

۳.۳ الگوریتم‌های کلاسیک برای ردیابی اشیاء

در زمینه ردیابی اشیاء نیز تاکنون الگوریتم‌های متنوعی با عملکردها و کاربردهای مختلف ارائه شده‌اند که هر کدام نقاط قوت و ضعف مربوط به خود را دارند. به طور مشابه با مسئله شناسایی، الگوریتم‌های موجود در حوزه ردیابی را نیز می‌توان به دو دسته روش‌های کلاسیک و روش‌های نوین که مبتنی بر یادگیری عمیق هستند دسته‌بندی نمود. علاوه بر دسته‌بندی الگوریتم‌های مختلف برای ردیابی اشیاء، بسته به اینکه در الگوریتم‌های مختلف کدام یک از خصیصه‌های اشیاء مورد ردیابی قرار می‌گیرند، می‌توان آن‌ها را به انواع ردیابی نقطه‌ای[1]، ردیابی بر اساس هسته[2] (ردیابی ناحیه‌ای) و ردیابی سیاه‌نما[3] دسته‌بندی کرد [۱۵۶]. در جدول (۳-۱) الگوریتم‌های ارائه شده مربوط به هریک از این دسته‌ها معرفی شده‌اند.

دسته‌بندی	روش‌ها
Point Tracking	Modifying Greedy Exchange (MGE) tracker [۱۲۳] Greedy Optimal Assignment (GOA) tracker [۱۴۸] kalman filter [۲۰] Joint Probability Data Association Filter (JPDAF) [۹] Multiple Hypothesis Tracking (MHT) [۱۳۹]
Kernel Tracking	Mean-shift [۲۷] KLT [۱۳۰] Layering [۱۴۴] Eigentracking [۱۵] SVM tracker [۴]
Silhouette Tracking	State Space Models [۶۴] Variational methods [۱۲] Heuristic methods [۱۱۷] Hausdorff [۶۲] Hough Transform [۱۲۴] Histogram [۷۳]

جدول ۳-۱: دسته‌بندی روش‌های ردیابی کلاسیک و الگوریتم‌های ارائه شده برای هر کدام.

در ادامه به توضیح برخی از این الگوریتم‌ها به طور مختصر پرداخته می‌شود.

[1] Point Tracking
[2] Kernel Tracking
[3] Silhouette Tracking

۱.۳.۳ ردیابی نقطه مرکزی اشیاء

این الگوریتم یکی از ساده‌ترین و ابتدایی‌ترین الگوریتم‌های ردیابی به شمار می‌آید که به منظور درک بهتر مسئله ردیابی آن را مطرح می‌کنیم. این روش شامل چند مرحله بوده و مبتنی بر فاصله اقلیدسی بین مرکز فعلی شئ و مرکز جدید آن در قاب بعدی است.

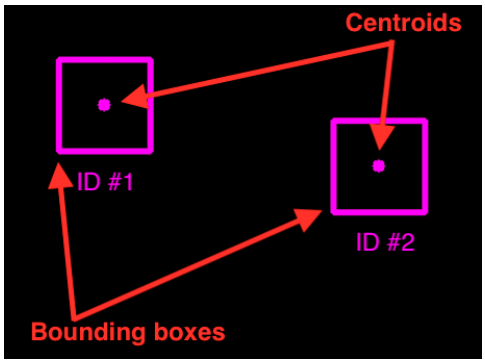
در این الگوریتم فرض می‌شود که در هریک از قاب‌های موجود، مختصات مرکز اشیاء شناسایی شده به صورت یک زوج مرتب (x,y) موجود است. در گام اول این مختصات با توجه به چهارچوب شکل گرفته به دور اشیاء محاسبه می‌شود که این چهارچوب در فاز شناسایی صورت می‌گیرد و می‌توان آن را با استفاده از الگوریتم‌های مختلف شناسایی اشیاء به دست آورد. همچنین در اولین حضور شئ در ویدئو و پس از شناسایی و محاسبه مرکز آن، باید یک شناسه یکتا به شئ جدید نسبت داده شود.

برای هر یک از قاب‌های موجود باید گام اول اجرا شود تا بتوان مرکز اشیاء شناسایی شده را به دست آورد. اما به جای این‌که یک شناسه جدید به آن‌ها نسبت داده شود، باید مشخص کرد که آیا می‌توان مرکز شئ شناسایی شده جدید را به مرکز یکی از اشیاء شناسایی شده در قاب(های) قبلی متصل کرد یا به نوعی آن‌ها را یک شئ یکسان در نظر گرفت یا خیر. برای این منظور فاصله اقلیدسی بین مرکز هر جفت شئ در قاب قبل و قاب فعلی محاسبه می‌شود و هر دو جفت شئ‌ای که فاصله بین مرکز آن‌ها کم‌ترین میزان را نسبت به بقیه جفت‌ها داشته باشد، به‌عنوان یک شئ یکسان در نظر گرفته می‌شود.

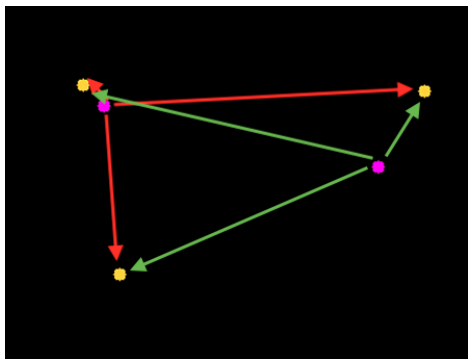
فرض مهم و اولیه در این الگوریتم ردیابی این است که فاصله اقلیدسی بین مرکز اشیاء در دو قاب متوالی F_t, F_{t+1} کوچک‌تر از فاصله بین تمامی جفت اشیاء در یک قاب باشد.

همان‌طور که گفته شد هر الگوریتم ردیابی باید بتواند در برابر ناپدید شدن اشیاء از ناحیه تحت پوشش دوربین عکس‌العمل مناسبی نشان دهد. در این الگوریتم هرگاه هرکدام از اشیاء موجود در قاب قبلی نتواند با یکی از اشیاء موجود در N قاب بعدی یکسان در نظر گرفته شود، نتیجه می‌شود که شئ از ناحیه خارج شده و لذا شناسه آن از بین می‌رود. در شکل (۱.۳) می‌توان مراحل این الگوریتم رابه‌طور خلاصه مشاهده کرد.

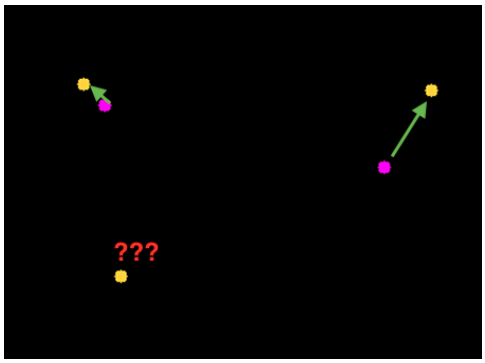
اولین نقطه ضعف این الگوریتم این است که در هر کدام از قاب‌ها باید فاز شناسایی اشیاء اجرا شود. البته زمانی که از برخی الگوریتم‌های شناسایی با پیچیدگی محاسباتی پایین استفاده می‌شود، چندان مشکل ساز نخواهد بود اما اگر از الگوریتم‌های مبتنی بر یادگیری عمیق استفاده شود که پیچیدگی محاسباتی بالایی دارند و همچنین اگر منابع سخت افزاری سیستم در اختیار محدود باشد، زمان زیادی برای هر قاب مصرف می‌شود و عملا استفاده از



(آ) شناسایی، محاسبه مرکز و تخصیص شناسه یکتا



(ب) محاسبه فاصله اقلیدسی بین هر جفت از مرکزهای موجود در قاب قبلی و فعلی



(ج) عدم توانایی اتصال مرکز یک یا چند مورد از اشیاء در قاب فعلی با مرکزهای موجود در قاب قبلی

(د) نتیجه‌گیری ورود شئ جدید، محاسبه مرکز و تخصیص شناسه یکتا به آن

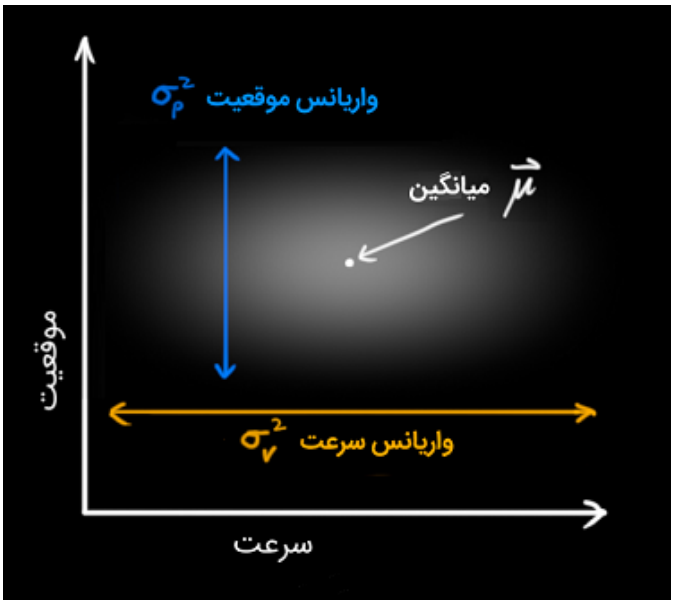
شکل ۱.۳: .روند ردیابی در روش مبتنی بر نقطه مرکزی اشیاء

این الگوریتم برای انجام ردیابی مناسب نخواهد بود.

نقطه ضعف دوم این روش مربوط می‌شود به شرط اولیه‌ای که در نظر گرفته می‌شود مبنی بر این‌که مرکزهای هر شئ در قاب‌های متوالی باید نزدیک به یکدیگر قرار بگیرد تا امکان ردیابی وجود داشته باشد. البته این شرط در حالت عادی رعایت می‌شود اما باید توجه داشت که در واقع اشیاء در فضای سه‌بُعدی وجود دارند اما در فضای دوبُعدی مدل می‌شوند و در این صورت اگر اشیاء با یکدیگر هم‌پوشانی داشته باشند یا تا حد زیادی به یکدیگر نزدیک شوند، ردیابی با مشکل مواجه خواهد شد به طوری که ممکن است شناسه‌های آن‌ها با یکدیگر تعویض شود. به دلیل مشابه می‌توان نتیجه گرفت که این روش در برابر انسدادهای جزئی نیز به هیچ عنوان مقاوم نیست.

۲.۳.۳ فیلتر کالمن

فیلتر کالمن[1] در واقع یک پیش‌بینی کننده است که از پیش‌بینی‌های حالات قبلی و مشاهدات فعلی برای پیش‌بینی حالت و تغییرات گام بعدی استفاده می‌کند و یک تکنیک قوی برای ترکیب اطلاعات غیر قطعی به شمار می‌آید [۲۰، ۷۰، ۷۱]. این فیلتر به طور کلی برای سیستم‌هایی که دائما در حال تغییر هستند بسیار مناسب بوده و در زمینه‌های مختلف به ویژه در مورد مسائل مربوط به ردیابی اشیاء بسیار استفاده شده است [۱۰۴، ۱۵۲]. به منظور درک بهتر از نحوه عملکرد این فیلتر، فرض کنید هر حالت از شئ مورد نظر توسط دو مولفه موقعیت و سرعت متناظر با آن بیان و به صورت $\hat{x}_k = \begin{bmatrix} p \\ v \end{bmatrix}$ نشان داده شود. در این شرایط معمولا اطلاعات دقیقی در مورد موقعیت و سرعت واقعی شئ در دسترس نیست و طیف وسیعی از فرض‌های اولیه در مورد آن‌ها موجود است اما مشخص است که برخی از آن‌ها به واقعیت نزدیک‌تر هستند. در فیلتر کالمن تصور می‌شود که دو متغیر (به‌عنوان مثال موقعیت و سرعت) تصادفی هستند و به صورت گوسی توزیع شده‌اند. هر یک از متغیرها دارای یک مقدار میانگین و یک مقدار واریانس است که به ترتیب با نمادهای μ و σ^2 نشان داده می‌شوند. مقدار میانگین در حقیقت بیان‌گر مرکز توزیع تصادفی (محتمل‌ترین حالت) و مقدار واریانس معرف دامنه توزیع تصادفی (اطلاعات غیر قطعی) است.



شکل ۲.۳: میانگین و واریانس متغیرهای موجود (موقعیت و سرعت) در فیلتر کالمن.

در برخی موارد متغیرهای موجود هیچ گونه هم‌بستگی با یکدیگر ندارند یا به عبارت دیگر اطلاعات مربوط به یکی از متغیرها هیچ اطلاعاتی در مورد متغیر دیگر نمی‌دهد. این در حالی است که دو متغیر مانند موقعیت و

[1] Kalman Filter (KF)

سرعت با یکدیگر هم‌بستگی دارند. هم‌بستگی بین متغیرها بسیار حائز اهمیت است و باعث می‌شود تا اندازه‌گیری یکی از آن‌ها اطلاعاتی را در مورد متغیر دیگر ارائه دهد. به‌عنوان مثال اگر قصد داشته باشیم موقعیت جدید شئ را بر اساس موقعیت قبلی آن به دست بیاوریم، اگر سرعت آن زیاد باشد احتمالا مسافت بیش‌تری را طی کرده است و لذا در موقعیت دورتری خواهد بود و بالعکس. در حالت کلی می‌توان هم‌بستگی بین داده‌ها را در قالب ماتریسی به نام ماتریس کوواریانس[1] نشان داد به طوری که هر درایه از این ماتریس درجه هم‌بستگی بین متغیر حالت $i-$اُم و متغیر حالت $j-$اُم را نشان می‌دهد. بر این اساس ماتریس کوواریانس یک ماتریس متقارن است و معمولا با Σ و درایه‌های آن با Σ_{ij} نشان داده می‌شود. به منظور تعریف مسئله در قالب ماتریسی آن، ابتدا دانش موجود درباره حالت مورد نظر به صورت یک حباب یا توده گوسی[2] مدل می‌شود. در این حالت با توجه به این که اطلاعات در قالب دو متغیر سرعت و موقعیت بیان می‌گردد، بهترین پیش‌بینی در زمان k به صورت زیر نوشته می‌شود:

$$\hat{x}_k = \begin{bmatrix} p \\ v \end{bmatrix},$$

$$P_k = \begin{bmatrix} \Sigma_{pp} & \Sigma_{pv} \\ \Sigma_{vp} & \Sigma_{vv} \end{bmatrix}.$$

اکنون باید حالت فعلی (زمان $k-۱$) و پیش‌بینی حالت بعدی (زمان k) مشخص شوند. مرحله پیش‌بینی توسط ماتریسی مانند F_k نشان داده می‌شود که در آن برای هر نقطه پیش‌بینی از نقطه اولیه انجام می‌شود و به یک محل پیش‌بینی شده جدید منتقل می‌کند. حال نحوه استفاده از این ماتریس برای پیش‌بینی موقعیت و سرعت در لحظه بعدی به صورت زیر تعریف می‌گردد:

$$\begin{aligned} p_k &= p_{k-۱} + v_{k-۱}\,\Delta t, \\ v_k &= v_{k-۱}, \end{aligned} \tag{۱-۳}$$

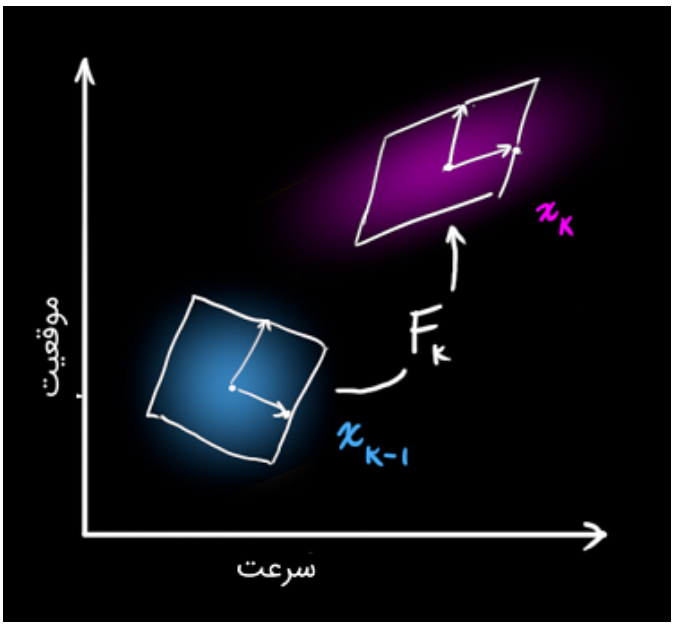
یا به عبارت دیگر:

$$\begin{aligned} P_k &= \begin{bmatrix} ۱ & \Delta t \\ ۰ & ۱ \end{bmatrix} \hat{x}_{k-۱}, \\ &= F_k\,\hat{x}_{k-۱}. \end{aligned} \tag{۲-۳}$$

به این ترتیب ماتریس پیش‌بینی F_k تشکیل شده و با استفاده از آن حالت بعدی پیش‌بینی می‌شود.

پس از شرایط جدید نیاز است تا ماتریس کوواریانس به روزرسانی شود. می‌دانیم که اگر هر نمونه یا نقطه‌ای

[1] Covariance Matrix
[2] Gaussian Blob



شکل ۳.۳: نحوه عملکرد ماتریس پیش‌بینی F_k برای پیش‌بینی حالت جدید با توجه به حالت پیش‌بینی شده قبلی.

مانند x در ماتریسی مانند A ضرب شود، ماتریس کوواریانس آن به صورت زیر تغییر خواهد کرد:

$$\begin{aligned} Cov(x) &= \Sigma \\ Cov(Ax) &= A\,\Sigma\,A^T \end{aligned} \tag{۳-۳}$$

حال با ترکیب روابط (۳-۲) و (۳-۳) خواهیم داشت:

$$\begin{aligned} \hat{x}_k &= F_k\,\hat{x}_{k-1}, \\ P_k &= F_k\,P_{k-1}\,F_k^T. \end{aligned} \tag{۴-۳}$$

اکنون در نظر بگیرید که حرکت شئ مورد نظر به هر دلیل خارجی روال معمولی حرکت خود را تغییر دهد. به‌عنوان مثال اگر جسمی مانع آن شود و یا به هر دلیلی سرعت حرکت خود را تغییر دهد. در این صورت بدیهی است که این حالات نیز باید در پیش‌بینی انجام شده در نظر گرفته شوند. این عوامل که معمولا به نام تاثیرات خارجی شناخته می‌شوند با استفاده از $\vec{u_k}$ نشان داده و به فرمول پیش‌بینی اضافه می‌شوند. به‌عنوان مثال اگر شئ با تغییر در سرعت خود به شتابی برابر با a برسد آنگاه برای روابط به دست آمده خواهیم داشت:

$$\begin{aligned} p_k &= p_{k-1} + v_{k-1}\,\Delta t + \tfrac{1}{2}a\Delta t^2, \\ v_k &= v_{k-1} + a\Delta t, \end{aligned} \tag{۵-۳}$$

به طوری که فرم ماتریسی آن برابر است با:

$$\begin{aligned} \hat{x}_k &= F_k\,\hat{x}_{k-1} + \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} a, \\ &= F_k\,\hat{x}_{k-1} + B_k\vec{u_k}, \end{aligned} \tag{۶-۳}$$

که در این رابطه B_k ماتریس کنترل[1] و $\vec{u_k}$ بردار کنترل[2] نامیده می‌شود که البته در سیستم‌های ساده می‌توان از آن‌ها چشم‌پوشی نمود. اما در صورتی که تاثیرات خارجی ناشناخته باشند و نتوان آن‌ها را محاسبه و در فرمول‌ها لحاظ کرد، چگونه می‌توان پیش‌بینی دقیقی از حالت بعدی شئ مورد نظر داشت؟ می‌دانیم هر یک از حالت‌های پیش‌بینی شده در مرحله قبل می‌تواند به محدوده‌ای از حالت‌ها حرکت کند. از آنجایی که یک حباب یا توده گوسی در نظر گرفته می‌شود، می‌توان گفت که هر نقطه در $\hat{x}_{k-1}$ به جایی در حباب گوسی با کوواریانس Q_t منتقل می‌شود. به بیان دیگر تاثیرات ردیابی نشده به‌عنوان نوفه‌ای با کوواریانس Q_t در نظر گرفته می‌شود. به این ترتیب یک حباب گوسی جدید به وجود می‌آید که کوواریانس متفاوتی نسبت به قبل دارد اما میانگین آن تغییری نمی‌کند. حال کوواریانس جدید که با نام کوواریانس توسعه یافته شناخته می‌شود را می‌توان با اضافه کردن Q_t به روابط قبل و به صورت زیر به دست آورد:

$$\begin{aligned} \hat{x}_k &= F_k\,\hat{x}_{k-1} + B_k\vec{u_k}, \\ P_k &= F_k\,P_{k-1}\,F_k^T + Q_k. \end{aligned} \qquad (۷-۳)$$

بر این اساس بهترین پیش‌بینی برای حالت جدید ($\hat{x}_k$) عبارت است از بهترین پیش‌بینی انجام شده در حالت قبلی ($\hat{x}_{k-1}$) به اضافه یک تصحیح برای تاثیرات خارجی معلوم ($\vec{u_k}$).

از آنجایی که فیلتر کالمن به حافظه کم و تنها برای نگهداری اطلاعات وضعیت‌های قبلی نیاز دارد و بسیار سریع است، تمایل به استفاده از آن را بیش از پیش افزایش می‌دهد. اما این در حالی است که اگر توزیع داده‌ها به صورت گوسی نباشد و حرکت اشیاء الگوی خطی نداشته باشد و همچنین در حالت‌هایی که شئ هدف مورد انسداد قرار می‌گیرد، این فیلتر نمی‌تواند عملکرد چندان مناسبی به ارائه دهد.

به منظور رفع التزام توزیع گوسی داده‌ها و همچنین الگوی حرکت خطی اشیاء در فیلتر کالمن، روش فیلتر جزئی[3] معرفی شد [۳۲]. این فیلتر را می‌توان حالت تعمیم یافته‌ای از فیلتر کالمن دانست که می‌تواند در مورد ردیابی‌هایی که داده‌ها در آن توزیع گوسین ندارند و همچنین حرکت آن‌ها از هیچ الگوی خطی پیروی نمی‌کند، اعمال شود و به همین دلیل بیشتر از فیلتر کالمن مورد استفاده قرار می‌گیرد [۶۱]. با این وجود نکته‌ای که این روش از آن رنج می‌برد، هزینه محاسباتی نسبتا سنگین آن است که در مورد ردیابی‌های بلادرنگ نمی‌تواند عملکرد مناسبی داشته باشد.

[1] Control Matrix
[2] Control Vector
[3] Particle Filter (PF)

روش‌ها	سرعت	انسداد	تغییر مقیاس	تغییر حالت
Active models-based KF [۱۲۳]	بالا	×	✓	✓
Structural KF [۱۴۸]	متوسط	✓	✓	✓
Adaptive KF [۲۰]	بالا	✓	✓	✓
Feature-based KF [۹]	متوسط	✓	✓	✓
MCMC-based PF [۱۳۹]	پایین	✓	✓	✓
Swarm intelligence-based PF [۲۷]	متوسط	×	✓	✓
Occlusion-aware PF [۱۳۰]	پایین	✓	✓	✓
Interactive PF [۱۴۴]	پایین	✓	✓	✓

جدول ۳-۲: روش‌های ارائه شده برای ردیابی اشیاء بر اساس فیلتر کالمن (KF) و فیلتر جزئی (PF) [۶۱]. (علامت ✓ به معنی آن است که روش مورد نظر در برابر موارد ذکر شده مقاوم است.)

Mean-shift ۳.۳.۳

Mean-shift یا جستجوگر مُد[1] یک الگوریتم محبوب است که به طور کلی برای مسائل خوشه‌بندی و سایر مسائل بدون ناظر استفاده می‌شود و هدف اصلی آن یافتن مُد (بیش‌ترین تجمع) بر روی توزیع داده‌هاست [۲۳]. نحوه عملکرد آن مانند الگوریتم K-means است اما به جای تکنیکی که در K-means برای یافتن نقاط مرکزی استفاده می‌شود، از یک میانگین وزن دار استفاده می‌کند به طوری که به نقاطی که به میانگین نزدیک‌تر هستند اهمیت بیشتری می‌دهد. برای ردیابی اشیاء Mean-shift به واسطه خصوصیات ظاهری اشیاء مانند رنگ، بافت و هیستوگرام این کار را انجام می‌دهد. برای این منظور ابتدا شئ مورد نظر باید شناسایی و ویژگی‌های آن استخراج شود و سپس الگوریتم Mean-shift بر روی آن‌ها اعمال گردد. از این طریق یک ایده کلی در مورد اینکه مُد توزیع ویژگی‌های استخراج شده در حال حاضر در کدام موقعیت از قاب وجود دارد به دست خواهد آمد. سپس در قاب بعدی که به واسطه حرکت شئ این توزیع تغییر یافته است الگوریتم Mean-shift مُد جدید را با توجه به ویژگی‌های به دست آمده جستجو می‌کند و به این ترتیب شئ مورد نظر ردیابی می‌شود [۲۶].

Boosting Tracker ۴.۳.۳

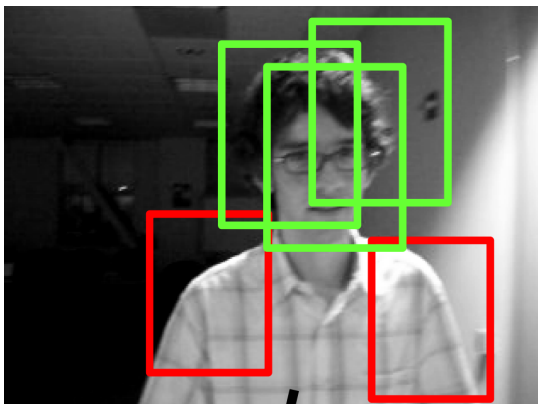
این ردیاب بر اساس یک مدل برخط از AdaBoost است و نیازمند آن است تا با مثال‌های مثبت و منفی آموزش داده شود و همچنین چهارچوب اولیه اطراف شئ هم می‌تواند توسط کاربر و هم توسط یکی از الگوریتم‌های شناسایی مشخص شود [۴۶]. سپس شئ درون چهارچوب به‌عنوان یک مثال مثبت برای این ردیاب، و قسمت‌های خارج از آن به‌عنوان پس‌زمینه آن در نظر گرفته می‌شوند. با نمایش قاب جدید، دسته‌بندی کننده مورد نظر بر روی تمامی

[1] Mode

پیکسل‌های موجود در همسایگی موقعیت قبلی شئ اعمال می‌شود و امتیاز دسته‌بندی کننده ذخیره می‌گردد. مکان جدید شئ مورد نظر جایی است که دسته‌بندی کننده در آن‌جا امتیاز بیشتری را کسب کرده است که یک مثال مثبت دیگر برای دسته بندی کننده محسوب می‌شود. همان‌طور که قاب‌های بیشتری وارد می‌شوند، دسته‌بندی کننده با این اطلاعات جدید خودش را به روزرسانی می‌کند. افزون بر یک دهه از معرفی این ردیاب می‌گذرد و به دلیل عملکرد نه چندان خوب آن و سرعت پایینی که دارد تقریبا تنها برای مقایسه بین الگوریتم‌های دیگر استفاده می‌شود.

MIL ۵.۳.۳

ایده کلی روش MIL[1] مشابه روش Boosting Tracker است. تنها تفاوت عمده این دو روش این است که به جای در نظر گرفتن مکان فعلی شئ به‌عنوان یک نمونه مثبت، محدوده کوچکی از اطراف موقعیت فعلی شئ نیز در نظر گرفته می‌شود و چند نمونه مثبت دیگر نیز تولید می‌گردد به طوری که در آن نمونه‌ها موقعیت شئ شناسایی شده در وسط آن ناحیه نیست و ممکن است حتی بخشی از آن شئ نیز در تصاویر موجود نباشد [۵]. احتمالا این تصور به وجود آید که به دلیل اینکه در اکثر این مثال‌های مثبت، شئ مورد نظر در مرکز تصویر واقع نشده است پس این ایده مناسب به نظر نمی‌رسد اما با معرفی یادگیری چند نمونه‌ای این سوال بر طرف خواهد شد. در یادگیری چند نمونه‌ای به جای نمونه‌های مثبت و منفی، از مجموعه‌ای از نمونه‌های مثبت و منفی استفاده می‌شود. این روش نسبت به روش قبلی دقت و عملکرد بهتری دارد و همچنین در مقابل انسداد های جزئی تا حدودی می‌تواند مقاوم باشد اما همچنان تا عملکرد قابل قبول فاصله دارد.



شکل ۴.۳: نمونه‌های مثبت (سبز رنگ) و نمونه‌های منفی (قرمز رنگ) در روش یادگیری چند نمونه‌ای

[1] Multi Instanse Learning (MIL) Tracker

MedianFlow Tracker ۶.۳.۳

این ردیاب اشیاء را در هر دو جهت جلو و عقب به طور هم‌زمان ردیابی می‌کند و سپس اختلاف بین این دو مسیر را به دست می‌آورد [۶۸]. کمینه‌سازی این خطاهای جلو-عقب این امکان را فراهم می‌سازد تا شکست های ردیابی به راحتی شناسایی شده و مسیرهای قابل اطمینان در توالی قاب‌ها انتخاب شود. به طور کلی این روش در حالت‌هایی که حرکات اشیاء قابل پیش‌بینی و کوتاه هستند و انسدادی وجود ندارد، عملکرد خوبی دارد و برخلاف سایر روش‌ها که در هنگام خراب شدن ردیابی باز هم آن را ادامه می‌دادند، این روش هنگامی که ردیابی با خطا مواجه شود مطلع می‌گردد. اما همان‌طور که ذکر شد اگر اشیاء دارای پرش باشند یا به‌طور سریع حرکت کنند، این روش با مشکل مواجه خواهد شد.

TLD ۷.۳.۳

روش TLD[1] وظیفه ردیابی طولانی مدت را به سه قسمت کوتاه ردیابی، یادگیری و شناسایی تجزیه می‌کند [۶۹]. فاز شناسایی ابتدا تمامی اشیاء ظاهر شده را شناسایی می‌کند و هنگامی که ردیاب با خطا مواجه شود در صورت لزوم آن را اصلاح می‌کند. در فاز یادگیری، خطا موجود در شناسایی کننده تخمین زده می‌شود و به روزرسانی می‌گردد تا در مراحل بعدی از وقوع آن‌ها جلوگیری شود. این روش دقت نسبتا بالایی دارد و در برابر تغییرات حالت‌های زیاد و حتی انسداد در طی چند قاب متوالی تقریبا مقاوم است اما مشکلاتی از قبیل وجود پرش در ردیابی و همچنین تعداد بالا در شناسایی‌های اشتباه، این روش را تقریبا غیر کارآمد کرده است. (منظور از پرش در ردیابی این است که اگر در حال ردیابی یک انسان باشیم و انسان دیگری در فاصله نه چندان دور از آن وجود داشته باشد، ممکن است این ردیاب برای مدتی انسان دیگر را ردیابی کند.)

MOSSE ۸.۳.۳

روش MOSSE[2] از هم‌بستگی انطباقی[3] برای ردیابی اشیاء استفاده می‌کند به طوری که وقتی با استفاده از یک قاب مقداردهی می‌شود، یک فیلتر هم‌بستگی پایدار[4] تولید می‌کند [۱۶]. این روش در مقابل تغییرات روشنایی، حالت و اندازه مقاوم است و همچنین انسدادها را بر اساس نرخ Peak-to-Sidelobe که توانایی توقف و ادامه ردیابی را در

[1] Tracking, Learning and Detection Tracker
[2] Minimum Output Sum of Squared Error Tracker
[3] Adaptive Correlation
[4] Stable Correlation Filter (SCF)

هنگام ناپدید شدن و ظهور مجدد شئ به ردیاب می‌دهد، شناسایی می‌کند. این روش از نظر پیاده‌سازی ساده است و دقت نسبتا خوبی دارد اما در برخی موارد نیز حتی در حرکات ساده اشیاء را گم می‌کند و همچنین در مقیاس محاسباتی هم در حدود روش‌های مبتنی بر یادگیری عمیق است.

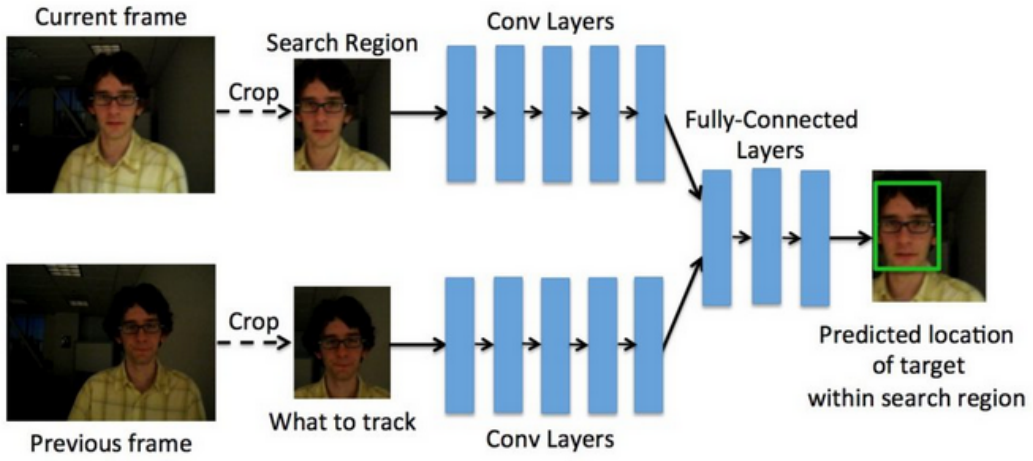
۴.۳ شیوه‌های نوین در ردیابی اشیاء

مجددا با فراگیر شدن مباحث یادگیری عمیق علاوه بر الگوریتم‌های شناسایی، الگوریتم‌های مربوط به ردیابی نیز به این سمت روی آوردند. یادگیری عمیق همانطور که در سایر حوزه‌ها پیشرفت‌های چشم‌گیری را با خود به همراه داشت، در این زمینه نیز عملکرد بسیار خوبی از خود به نمایش گذاشت و به همین دلیل امروزه اغلب از روش‌های مبتنی بر یادگیری عمیق برای ردیابی اشیاء استفاده می‌شود. این نوع الگوریتم‌ها نیز می‌توانند به چند دسته تقسیم شوند که در ادامه دسته‌های مختلف و الگوریتم‌های متناظر با آن‌ها شرح داده خواهد شد.

GOTURN ۱.۴.۳

GOTURN[1] یکی از نمونه‌های اولیه ردیاب‌های مبتنی بر شبکه‌های تلفیقی است که از قدرت این نوع شبکه‌ها در امر ردیابی استفاده می‌کند. در این الگوریتم ابتدا ردیاب با استفاده از جفت قاب‌های متوالی از هزاران ویدئو مربوط به ردیابی اشیاء آموزش داده می‌شود و سپس می‌توان با استفاده از آن و بدون نیاز به هیچ فاز یادگیری دیگری اشیاء مختلف در سایر ویدئوها را ردیابی نمود. به طور دقیق‌تر در این روش در قاب ابتدایی که معمولا با قاب پیشین یاد می‌شود، موقعیت شئ مشخص است و قاب مربوطه به اندازه دو برابر ناحیه ایجاد شده در اطراف شئ شناسایی شده برش داده می‌شود در حالی که شئ هدف در این قاب همواره در مرکز تصویر قرار دارد. حال باید موقعیت شئ در قاب دوم که معمولا به‌عنوان قاب فعلی یاد می‌شود پیش‌بینی شود. چهارچوبی که در قاب اولیه به منظور برش استفاده شده، مجددا برای برش قاب بعدی نیز استفاده می‌شود. از آنجایی که شئ ممکن است حرکت کرده باشد، به احتمال زیاد شئ در قاب دوم در مرکز تصویر نخواهد بود. سپس یک شبکه تلفیقی به نحوی آموزش داده می‌شود تا بتواند موقعیت شئ در قاب دوم را مشخص کند. لایه‌های استفاده شده در این شبکه تلفیقی از ۵ لایه ابتدایی شبکه CaffeNet [۶۵] برداشته شده است که خروجی آن‌ها به یک بردار با طول ۴۰۹۶ متصل می‌شود. سپس این بردار به سه لایه تماما متصل دیگر به‌عنوان ورودی داده می‌شود و پس از آن‌ها لایه خروجی قرار دارد.

[1] Generic Object Tracking Using Regression Networks (GOTURN)



شکل ۵.۳: نحوه ردیابی اشیاء در روش GOTURN و با استفاده از شبکه‌های تلفیقی و جفت قاب‌های متوالی.

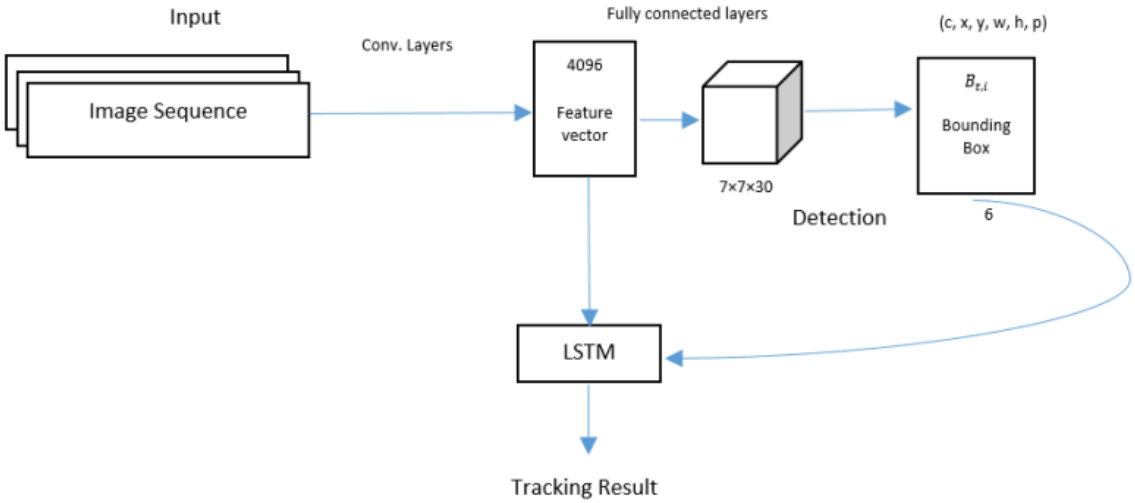
خروجی حاصل شامل چهار قسمت است که در واقع مختصات چهارگوشه bounding box به دست آمده را نشان می‌دهد. در مقایسه با دیگر الگوریتم‌های ردیابی مبتنی بر یادگیری عمیق، GOTURN از سرعت بیشتری برخوردار است. هرچند که این مدل یک مدل عمومی محسوب می‌شود اما می‌توان با خصوصی کردن مجموعه‌های آموزشی برای یک دسته از اشیاء مثلا انسان‌ها، عملکرد آن را بهبود بخشید.

ROLO ۲.۴.۳

ROLO[1] نوع دیگری از روش‌های ردیابی اشیاء به شمار می‌آید که بر اساس شبکه‌های تلفیقی و یادگیری برخط طراحی شده است. همانطور که از نام آن مشخص است، ROLO مبتنی بر روش YOLO طراحی شده که یکی از موفق‌ترین روش‌های ارائه شده تاکنون برای شناسایی اشیاء است و در قسمت قبل به آن پرداخته شد. علاوه بر آن این الگوریتم از شبکه‌های حافظه طولانی کوتاه مدت[2] استفاده می‌کند که عملکرد آن را در مورد یافتن مسیر حرکت اشیاء تا حدود زیادی بهبود می‌بخشد. شبکه‌های LSTM نوع خاصی از شبکه‌های عصبی بازگشتی[3] [۱۲۱] هستند که توانایی یادگیری وابستگی‌های بلند مدت را دارند [۵۸]. هدف از طراحی این شبکه‌ها حل مشکل وابستگی بلند مدت بود و به گونه‌ای طراحی شدند که به یاد سپاری اطلاعات برای بازه‌های زمانی بلند مدت رفتار پیش فرض و عادی آن‌هاست. از طرف دیگر این شبکه‌ها هزینه محاسباتی سنگینی ندارند و لذا استفاده از آن‌ها برای کاربردهای ردیابی اشیاء به صورت بلادرنگ بسیار مناسب است. روش کار ROLO به این صورت است که ابتدا قاب‌های مورد

[1] Recurrent YOLO
[2] Long Short Term Memory (LSTM)
[3] Recurrent Neural Network (RNN)

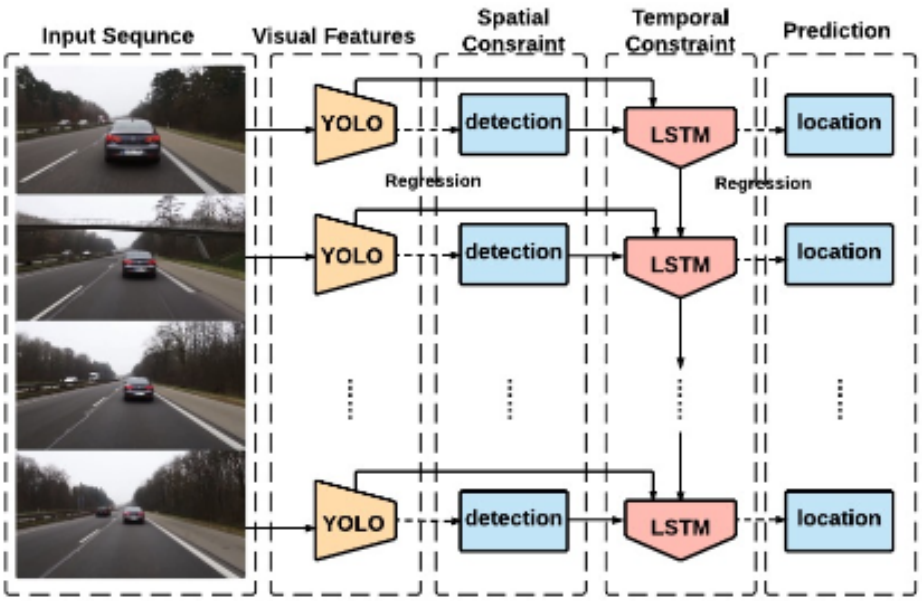
نظر به‌عنوان ورودی به شبکه YOLO داده می‌شود که خروجی آن بردارهای ویژگی قاب‌های ورودی و مختصات جعبه‌های محاطی محاسبه شده است. پس از انجام شناسایی، ویژگی‌های به دست آمده و مختصات جعبه‌های محاطی به شبکه LSTM وارد شده و خروجی آن مسیر حرکت شئ یا مختصات جعبه‌های محاطی پیش‌بینی شده آن در طی ردیابی است.



شکل ۶.۳: دیاگرام ردیابی اشیاء در روش ROLO با استفاده از الگوریتم شناسایی YOLO و شبکه LSTM.

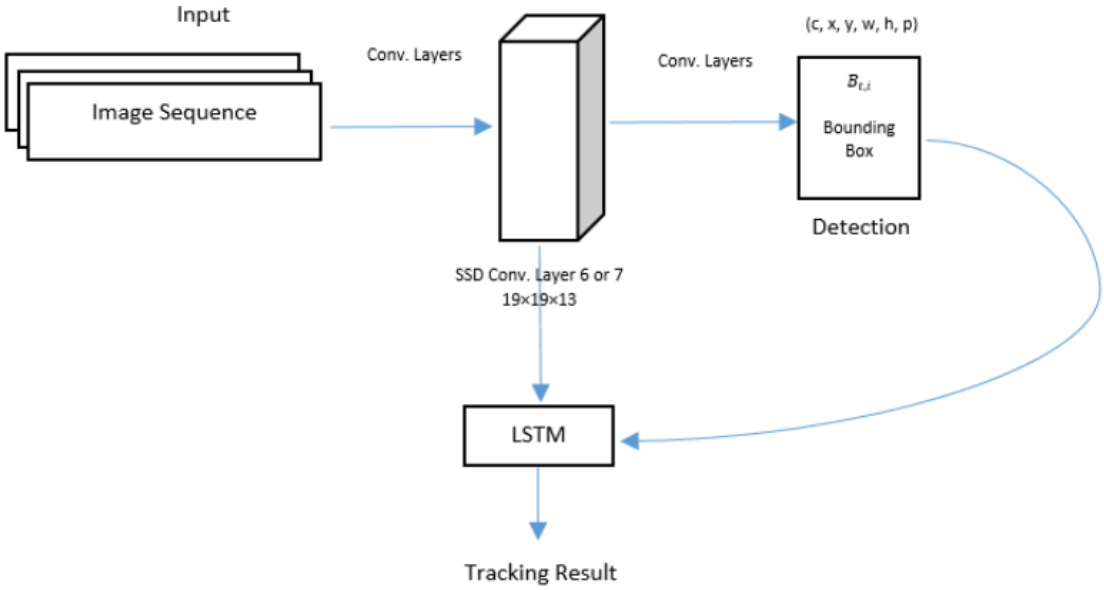
در واقع موقعیت اولیه‌ای که توسط YOLO به LSTM داده می‌شود برای این منظور است تا LSTM به طور دقیق‌تر و مشخص‌تری به مولفه‌های شئ مورد نظر توجه کند. سپس ROLO تاریخچه زمانی و مکانی موقعیت آن شئ را بررسی کرده و از ویژگی‌های بصری آن برای انجام عملیات ردیابی استفاده می‌کند. نکته جالب‌تر این است که حتی در مواقعی که YOLO نمی‌تواند به خوبی عملیات شناسایی را انجام دهد، ROLO به طور پایدارتری نسبت به آن عمل می‌کند و به همین دلیل این گونه ردیاب‌ها در مواقعی که شئ مورد نظر با انسداد روبه‌رو می‌شود، احتمال شکست به مراتب کمتری دارند.

البته می‌توان برای استخراج ویژگی‌ها و تعیین جعبه‌های محاطی از روش‌های دیگری مانند SSD استفاده کرد که در قسمت شناسایی اشیاء تشریح شد. یکی از تفاوت‌های YOLO و روش SSD در این بود که SSD در معماری خود از لایه‌های تماما متصل استفاده نمی‌کند و لذا نگاشت ویژگی‌هایی که به صورت توده‌ای هستند باید به یک بردار ویژگی نگاشت شده و به شبکه LSTM تزریق شوند. از آنجایی که در SSD نگاشت ویژگی‌ها در مقیاس‌های مختلفی به دست می‌آیند، نمی‌توان تمامی آن‌ها را مد نظر قرار داد. از طرف دیگر ممکن است اطلاعات برخی از آن‌ها برای LSTM مفید نباشد اما طبق آزمایش‌های انجام شده ثابت شده است که اطلاعات موجود در نگاشت ویژگی‌های لایه ۶ یا ۷ دارای اطلاعات مناسب‌تری نسبت به سایر لایه‌ها هستند [۱۰۷]. بنابراین حالت کلی ردیابی



شکل ۷.۳: نحوه عملکرد الگوریتم ROLO برای ردیابی اشیاء.

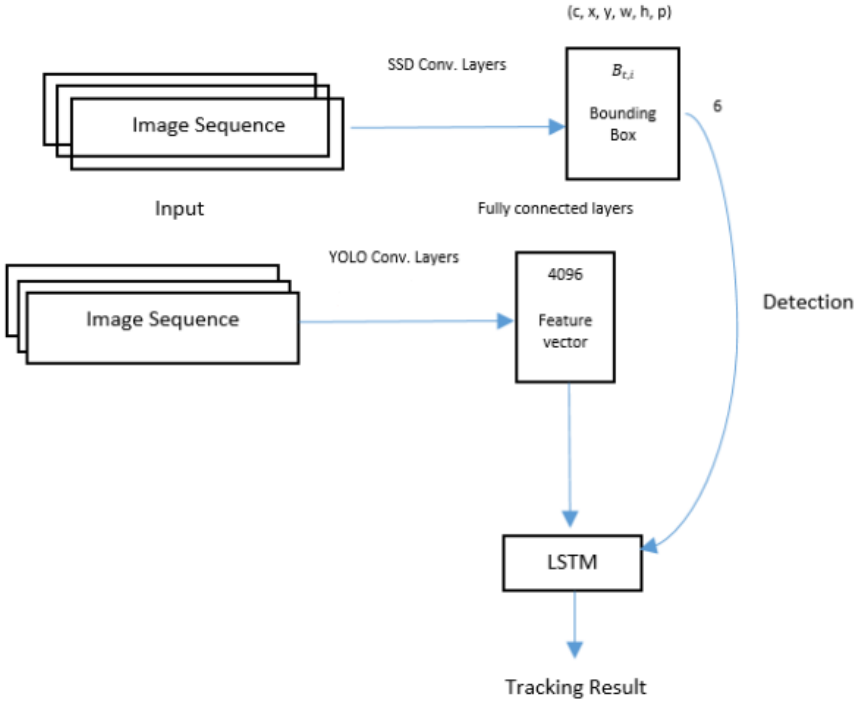
در این روش زمانی که از الگوریتم SSD استفاده می‌شود به صورت شکل (۸.۳) خواهد بود.



شکل ۸.۳: دیاگرام ردیابی اشیاء در روش ROLO با استفاده از الگوریتم شناسایی SSD و شبکه LSTM.

از آنجایی که ویژگی‌های استخراج شده توسط YOLO مقاوم‌تر بوده و دارای اطلاعات بیشتری نسبت به روش SSD دارند، می‌توان هر دو روش را با یکدیگر ترکیب کرد [۱۰۷]. به طوری که عملیات استخراج ویژگی‌ها توسط شبکه YOLO و وظیفه تعیین مختصات جعبه‌های محاطی بر عهده روش SSD باشد و سپس ترکیب اطلاعات حاصل شده از آن‌ها به شبکه LSTM وارد شود. در این صورت نمای کلی از نحوه عملکرد این الگو به صورت شکل

(۹.۳) خواهد بود.



شکل ۹.۳: دیاگرام ردیابی اشیاء در روش ROLO با استفاده از الگوریتم شناسایی SSD و YOLO.

Deep SORT ۳.۴.۳

همانطور که در مورد فیلتر کالمن توضیح داده شد، این الگوریتم بر اساس موقعیت و سرعت حرکت شئ موقعیت آن را پیش‌بینی و محاسبه می‌کند اما از اطلاعاتی مانند ویژگی‌های ظاهری اشیاء که در جعبه‌های محاطی به دست آمده موجود است، استفاده مفیدی نمی‌کند. الگوریتم SORT[1] برای اولین بار در سال ۲۰۱۶ توسط فابیوس و همکارانش معرفی شد [۱۳]. ایده اولیه آن‌ها بر اساس استفاده از فیلتر کالمن و الگوریتم Hungarian [۷۶] برای اتصال داده‌های حاصل از ردیاب‌ها به هر یک از اشیاء بود. این الگوریتم تنها برای شناسایی و ردیابی انسان‌ها اجرا می‌شد و همچنان با مشکلاتی مانند جابجایی شناسه‌ها با یکدیگر مواجه بود.

پس از آن در سال ۲۰۱۷ نویسندگان آن با به کار گیری مبحث یادگیری عمیق و شبکه‌های تلفیقی الگوریتم دیگری تحت عنوان Deep SORT ارائه دادند. در این الگوریتم ویژگی‌های ظاهری هر یک از جعبه‌های محاطی توسط شبکه‌های عمیق استخراج و از آن‌ها در کنار اطلاعاتی مانند موقعیت و سرعت در فیلتر کالمن استفاده می‌شد. سپس بر اساس این ویژگی‌ها و اطلاعات مربوط به حرکت، یک ماتریس هزینه[2] ساخته می‌شود که با استفاده از

[1] Simple Online and Realtime Tracker
[2] Cost Matrix

آن از شکست ردیابی جلوگیری شود. در این مطالعه با استفاده از مدلی که بر روی میلیون‌ها تصویر از انسان‌ها آموزش داده شده، به ازای هر جعبه محاطی یک بردار ۱۲۸ تایی از ویژگی‌های کلیدی به دست آمده است. همانطور که واضح است استخراج ویژگی‌ها به این روش بسیار دقیق‌تر بوده و همین امر باعث می‌شود عملیات ردیابی در حالت‌هایی که اشیاء نزدیک به یکدیگر قرار می‌گیرند و یا حتی تداخلی بین آن‌ها صورت می‌گیرد، بسیار مقاوم‌تر عمل کند. از آنجایی که شناسایی اشیاء در این الگوریتم با استفاده از مدل‌های از پیش آموزش دیده اتفاق می‌افتد، در مواقعی که نمای دوربین کمی دور و یا از بالا باشد و همچنین جمعیت موجود در صحنه زیاد باشند، شناسایی اشیاء در برخی مواقع به طور صحیح انجام نمی‌شود.

فصل ۴
روش پیشنهادی و نتایج

۱.۴ مقدمه

همانطور که در فصل قبل نیز مطرح شد، ردیابی اشیاء همواره از اهمیت بالایی برخوردار بوده و به ویژه در سال‌های اخیر توجه افراد زیادی را در حیطه‌های مختلف به خود جلب کرده است. به طور کلی می‌توان الگوریتم‌های مربوط به ردیابی اشیاء را به دو دسته الگوریتم‌های مولد[۲] [۱۱۹] و الگوریتم‌های متمایزکننده[۳] [۶] تقسیم نمود. در بین این دو دسته، الگوریتم‌های متمایزکننده به خاطر عملکرد مناسب‌تر، هزینه محاسباتی کمتر و در نتیجه سرعت بالا، محبوبیت بیشتری کسب کرده‌اند. به طور خلاصه این دسته از الگوریتم‌ها تلاش می‌کنند تا با کمک یک دسته‌بندی کننده مناسب، بهترین ناحیه را به عنوان موقعیت شئ هدف شناسایی و آن را از محیط پس‌زمینه متمایز کنند.

ردیابی اشیاء موجود در یک فایل ویدئویی دارای سه مرحله استخراج نمونه، دسته‌بندی و تصمیم‌گیری بر اساس دسته‌بندی‌های انجام گرفته است. برای داشتن یک ردیابی با عملکرد مناسب، تمامی این مراحل و به خصوص فاز دسته‌بندی بایستی با دقت فراوان انجام شود. در برخی از ردیابی‌هایی که از قبل مشخص نیست برای کدام دسته از اشیاء مورد استفاده قرار می‌گیرند یا به عبارتی بدون مدل[۴] هستند، نیاز است تا یادگیری و تطابق ظاهر اشیاء به صورت برخط انجام شود. اما یکی از نقاط ضعف این الگوریتم‌ها محدود بودن تعداد نمونه‌های آموزشی است و به همین خاطر تابع دسته‌بندی کننده آن‌ها نمی‌تواند از عملکرد بالایی برخوردار باشد. برای حل این مشکل راه‌حل‌های متنوعی پیشنهاد شده است. به عنوان نمونه می‌توان در هر قاب برخی از ناحیه‌های مجاور شئ هدف را به عنوان ناحیه‌های مثبت و منفی در نظر گرفت و بر اساس آن‌ها تابع دسته‌بندی را آموزش داد. مشکل این روش این است که این نواحی هم‌پوشانی‌های زیادی با یکدیگر دارند و اطلاعات غیرمفید زیادی باید ذخیره شود. از طرف دیگر در

[2] Generative
[3] Discriminative
[4] Model-less

این روش بازهم تعداد نمونه‌های آموزشی کم است و بهبود چشم‌گیری در عملکرد تابع دسته‌بندی کننده ایجاد نمی‌کند و در مواردی که شئ هدف توسط شئ دیگری پوشانده می‌شود و یا تغییر حالت می‌دهد، ممکن است ردیاب مربوطه با شکست مواجه شود.

از طرف دیگر و با توجه به محدود بودن نمونه‌های آموزشی در این نوع از ردیابی‌ها، در صورتی که تنها از یک ویژگی برای آموزش دسته‌بندی کننده استفاده شود، بازهم باعث تضعیف عملکرد آن خواهد شد؛ زیرا در این موارد انتخاب تنها یک ویژگی که بتواند در تمامی حالت‌ها شئ مورد نظر را به درستی از محیط پس زمینه آن متمایز کند بسیار دشوار و در اغلب موارد غیر ممکن است [۱۴۷]. در این شرایط راه حل اولیه‌ای که مطرح می‌شود استفاده از مجموعه‌ای از ویژگی‌ها است به طوری که در شرایط و برای اشیاء مختلف بتوانند دقت تمایزدهی بین شئ هدف و محیط پس زمینه را افزایش دهند.

در چنین حالتی و با هدف ادغام ویژگی‌های مختلف برای انجام عملیات ردیابی شئ هدف، استفاده از یک هسته می‌تواند ویژگی‌های متمایزکننده نسبتا خوب را نیز با آشفتگی مواجه کند و به طور واضح یک انتخاب مناسبی نخواهد بود. در این صورت استفاده از مجموعه‌ای از ویژگی‌ها و همچنین یک مجموعه از هسته‌های مختلف به منظور ترکیب ویژگی‌های مختلف به دلایلی که در فصل سوم به طور مفصل به آن‌ها پرداخته شد، یک گزینه مناسب به نظر می‌رسد. سپس با ترکیب خطی از این هسته-ویژگی‌ها و سعی برای بهینه کردن ضرایب آن‌ها که با نام یادگیری چند هسته‌ای شناخته می‌شود، می‌تواند تا حدود زیادی عملکرد تابع دسته‌بندی کننده به دسته آمده را بهبود بخشد. برای این منظور کانو و همکارانش یک الگوریتم موثر با نام SimpleMKL پیشنهاد دادند که از فرم اولیه گرادیان کاهشی برای حل این مسئله استفاده می‌کند [۱۱۰]. وارما و همکارانش نیز با معرفی یک قید اضافه در ضرایب ترکیب خطی، ساختار مسئله MKL را توسعه دادند و از آن برای مسئله دسته‌بندی تصاویر استفاده کردند [۱۴۷].

۲.۴ فیلتر هم‌بستگی هسته‌گذاری شده (KCF)

الگوریتم فیلتر هم‌بستگی هسته‌گذاری شده[1] یک روش ردیابی متمایزکننده به حساب می‌آید که توسط هنریکیو و همکارانش در سال ۲۰۱۲ ارائه شد [۵۳]. به‌طور کلی هدف از طراحی این الگوریتم رفع نقطه‌ضعف کمبود تعداد نمونه‌های آموزشی در یادگیری تابع دسته‌بندی کننده بود به‌طوری که هزینه محاسباتی حاصل از این افزایش تعداد

[1] Kernelized Correlation Filter (KCF)

نمونه‌ها نیز تا حد ممکن کم باشد. در این الگوریتم ابتدا با استفاده از یک ساختار چرخشی[1] از یک نمونه آموزشی اولیه، یک مجموعه داده غنی به شکل یک ماتریس چرخشی[2] ایجاد می‌شود. سپس با بهره‌مندی از تبدیلات فوریه[3] به جای استفاده از عملگرهای پرهزینه ماتریسی، هزینه‌های محاسباتی تا حدود زیادی کاهش یافته و در نتیجه سرعت ردیابی نیز افزایش می‌یابد. به طور دقیق‌تر فرض کنید $\mathbf{x}$ یک نمونه آموزشی اولیه یا به عبارت دیگر قطعه تصویر شامل شئ هدف باشد که می‌توان آن را به صورت $\mathbf{x} \in \mathbb{R}^{l-1}$ نشان داد. پس از آن با استفاده از یک ماتریس جایگشت مانند $\mathbf{P}$ بردار نمونه $\mathbf{x}$ به صورت چرخشی شیفت پیدا کرده و در نهایت یک ماتریس چرخشی به صورت $\mathbf{X} = \left[\mathbf{x}, \mathbf{P}\mathbf{x}, \ldots, \mathbf{P}^{l-1}\mathbf{x}\right]^T$ تشکیل می‌شود. به عبارت دیگر اگر $\mathbf{x} = [x_0, x_1, \ldots, x_{l-1}]$ باشد، آنگاه ماتریس چرخشی $\mathbf{C}(\mathbf{x})$ حاصل شده از آن به صورت زیر خواهد بود:

$$\mathbf{C}(\mathbf{x}) = \begin{bmatrix} x_0 & x_1 & x_2 & \ldots & x_{l-1} \\ x_{l-1} & x_0 & x_1 & \ldots & x_{l-2} \\ x_{l-2} & x_{l-1} & x_0 & \ldots & x_{l-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1 & x_2 & x_3 & \ldots & x_0 \end{bmatrix}. \qquad (۱-۴)$$

به این ترتیب هر یک از سطرهای ماتریس $\mathbf{C}(\mathbf{x})$ به عنوان یک نمونه آموزشی محسوب می‌شوند و اصطلاحا به این کار نمونه‌گیری متراکم[4] گویند. سپس با توجه به این مجموعه آموزشی تلاش می‌شود تا تابع دسته‌بندی کننده مربوطه به‌طور موثر و با کمترین پیچیدگی محاسباتی به دست آید به‌طوری که نتیجه آن یک ردیاب با سرعت بالا و دقت مناسب خواهد بود. برای اطلاع از نحوه آموزش و محاسبه تابع دسته‌بندی کننده با استفاده از ساختارهای گفته شده به [۵۳] مراجعه شود.

با اینکه عملکرد این الگوریتم تا حدودی مناسب است اما این روش نیز مشکلات مختلفی دارد. یکی از مشکلات این الگوریتم این است که از یک هسته برای به دست آوردن شباهت بین نمونه‌ها و یادگیری تابع دسته‌بندی کننده استفاده می‌کند و همانطور که گفته شد این رویکرد نمی‌تواند دسته‌بندی دقیقی را نتیجه دهد. علاوه بر این برای دسته‌بندی و موقعیت‌یابی شئ هدف در قاب بعدی، روش KCF تنها از اطلاعات به دست آمده از قاب قبلی استفاده می‌کند و تابع دسته‌بندی کننده بر اساس نمونه‌های آموزشی حاصل از آن محاسبه می‌شود. در چنین حالتی واضح است که مجددا دسته‌بندی کننده به دست آمده در مواردی که ظاهر و حالت شئ هدف تشابه زیادی با ظاهر و حالت آن در قاب قبلی نداشته باشد، با شکست مواجه می‌شود. بنابراین استفاده از رویکردی که با استفاده از آن بتوان

[1] Circulant Structure
[2] Circulant Matrix
[3] Fourier Transforms
[4] Dense Sampling

از اطلاعات سایر قاب‌های قبل نیز استفاده کرد می‌تواند دقت دسته‌بندی را تا حد قابل توجهی افزایش دهد و در نتیجه عملکرد ردیاب‌ها را در موارد مختلف بهبود دهد.

۳.۴ مدل حافظه‌ای چند کاناله

مکانیزم حافظه یک فرآیند کلیدی برای ذهن انسان محسوب می‌شود و در مواردی مانند شناسایی هدف، هویت‌یابی و ردیابی اشیاء توسط انسان‌ها نقشی مهم و اساسی ایفا می‌کند. با استفاده از این فرآیند در هنگام رویارویی با یک شئ جدید، اطلاعات مربوط به آن (در صورتی که آن شئ قبلا مشاهده شده باشد) استخراج و بازیابی می‌شود و باعث می‌شود تا عملیات مورد نظر سریع‌تر و دقیق‌تر انجام گیرد. به این ترتیب می‌توان از یک رویکرد مشابه برای افزایش دقت و عملکرد ردیاب کننده‌های خودکار استفاده کرد. در همین راستا آمر یک روش بلادرنگ برای تقسیم‌بندی هدف بر اساس مکانیزم حافظه ارائه داده است که در ویدئوهای نظارتی و برای تصاویر با حالت‌های مختلف از پایداری بالایی برخوردار است [۱]. علاوه بر این کانگ و همکارانش یک حافظه کوتاه مدت برای فرآیند به روزرسانی مدل شئ هدف معرفی کردند که تاثیرات حاصل از تغییر حالت هدف را کاهش می‌داد [۷۲]. همچنین با هدف رفع مشکلات حاصل از انسداد و تغییرات زاویه دید، وانگ و همکارانش یک مدل پردازش اطلاعات بصری بر اساس مکانیزم حافظه ارائه دادند [۱۰۸]. این مدل سپس به چهارچوب‌هایی مانند فیلتر جزئی و Mean-shift داده می‌شد که نتایج قابل قبولی را برای عملیات ردیابی ارائه می‌داد. پس از آن گونگ و همکارانش نیز در سال ۲۰۱۸ الگوریتمی بر همین اساس ارائه دادند [۴۵]. در این الگوریتم از یک مدل حافظه چند کاناله استفاده می‌شود که شامل یک کانال کنترل و دو کانال اجرایی است. سپس این مدل برای به روزرسانی مدل شئ هدف به تابع دسته‌بندی کننده در الگوریتم فیلتر هم‌بستگی هسته‌گذاری شده ارائه می‌شود به‌طوری که با ارائه نتایج عملی نیز نشان داده شده که ردیاب به دست آمده از دقت بالایی برخوردار است. بنابراین استفاده از مکانیزم‌های حافظه در کنار الگوریتم‌های ردیابی متمایز کننده مانند فیلتر هم‌بستگی که پیچیدگی محاسباتی کمی دارند و به خصوص استفاده از روش‌های مبتنی بر چند هسته برای یادگیری تابع دسته‌بندی کننده می‌تواند بهبودهای چشم‌گیری را در مورد مسئله ردیابی و در شرایطی مانند انسداد و تغییرات حالت به ارمغان آورد.

۴.۴ الگوریتم پیشنهادی

در این قسمت با توجه به موارد گفته شده، یک الگوریتم پیشنهادی برای ردیابی اشیاء ارائه می‌شود که قابلیت استفاده از مجموعه‌ای از ویژگی‌ها را به واسطه استفاده از الگوریتم‌های یادگیری چندهسته‌ای فراهم می‌کند. علاوه بر این با بهره‌گیری از یک مدل حافظه‌ای چند کاناله ردیاب معرفی شده در برابر مسائلی از قبیل انسداد، تغییرات حالت و شلوغی پس‌زمینه از پایداری قابل توجهی برخوردار است. در ادامه ابتدا به محاسبه فرمول‌های مورد نظر برای شناسایی و ردیابی اشیاء بر اساس استفاده از چند هسته و سپس به توضیح در مورد پیاده‌سازی مدل حافظه‌ای چند کاناله پرداخته می‌شود.

۱.۴.۴ یادگیری تابع دسته‌بندی کننده

پس از تعیین موقعیت شئ در قاب اولیه، برای شناسایی موقعیت و تفکیک آن از محیط پس‌زمینه در قاب‌های بعدی نیاز است تا یک تابع دسته‌بندی کننده محاسبه شود. هنریکیو و همکارانش برای الگوریتم فیلتر هم‌بستگی مبتنی بر یک هسته و با استفاده از تبدیلات سریع فوریه روش محاسبه آن را ارائه دادند [۵۳]. در این مطالعه هدف از یادگیری تابع رگرسیون ریج[1] [۱۱۶] یافتن تابعی مانند $f(x)$ است به‌طوری که مجموع مربع خطاهای فرآیند یادگیری را کمینه کند یا به عبارت دیگر داشته باشیم:

$$\min_f \frac{1}{2} \sum_{i=0}^{l-1} (f(\mathbf{x}_i) - y_i)^2 + \lambda ||f||_k^2, \qquad (۲-۴)$$

که در آن l تعداد نمونه‌های آموزشی و f تابع دسته‌بندی کننده است و در یک زیر مجموعه محدب از فضای هیلبرت و توسط یک تابع معین مثبت با نام تابع هسته[2] مانند $k(\mathbf{x}, \mathbf{x}')$ تعریف شده است. y_i برچسب نمونه‌های آموزشی و $\lambda \geq 0$ نیز یک پارامتر تنظیم کننده است. با توجه به قضیه بیان کننده[3] [۱۲۶] تابع f^* می‌تواند به صورت زیر بیان شود:

$$f^*(\mathbf{x}) = \sum_{i=0}^{l-1} \alpha_i \, k(\mathbf{x}, \mathbf{x}_i). \qquad (۳-۴)$$

به این ترتیب $||f||_k^2 = \alpha^T \mathbf{K} \alpha$ که در آن $\alpha = (\alpha_0, \ldots, \alpha_{l-1})^T$ و $\mathbf{K} = (k_{ij})$ یک ماتریس نیمه معین مثبت است به‌طوری که $k_{ij} = k(x_i, x_j)$. بنابراین مسئله (۴-۲) را می‌توان به یافتن بردار α محدود کرد یا به عبارت دیگر

[1] Ridge Regression
[2] Kernel Function
[3] Representer Theorem

داریم:

$$\min_{\alpha\in\mathbb{R}^l} \frac{1}{2} ||\mathbf{y} - \mathbf{K}\alpha||_2^2 + \frac{\lambda}{2}\alpha^T\mathbf{K}\alpha, \tag{۴-۴}$$

که در آن $\mathbf{y} = (y_0, y_1, \ldots, y_{l-1})^T$.

این در حالی است که در الگوریتم پیشنهادی روش ارائه شده مبتنی بر چند هسته بوده و لذا فرمول‌بندی‌های مربوطه با حالت یک هسته‌ای کمی متفاوت خواهد بود. برای این منظور هسته‌های مختلف مانند k_m در نظر گرفته می‌شود که در آن $m = 1, 2, \ldots, M$. همانطور که در قسمت‌های قبل گفته شد می‌توان هسته‌های داده شده را به صورت یک ترکیب خطی مانند $k(x_i, x_j) = \sum_{m=1}^{M} d_m k_m(x_i, x_j)$ در نظر گرفت که در آن $\sum_{m=1}^{M} d_m = 1$ و $d_m \geq 0$. حال اگر به ازای هر یک از هسته‌ها ماتریس هسته[1] متناظر k_m را به صورت $\mathbf{K}_m$ در نظر بگیریم، فرم ماتریسی $\mathbf{K}$ به صورت زیر خواهد بود:

$$\mathbf{K} = \sum_{m=1}^{M} d_m \mathbf{K}_m, \tag{۵-۴}$$

که در آن درایه‌های $\mathbf{K}_m$ به صورت $k_{ij}^m = k_m(x_i, x_j)$ است. حال با جایگزینی ماتریس هسته $\mathbf{K}$ موجود در رابطه (۴-۵) با ماتریس هسته رابطه (۴-۴) و اضافه کردن قید ذکر شده برای جمع d_m ها، تابع هدفی به صورت $F(\alpha, \mathbf{d})$ برای مسئله رگرسیون ریج به دست می‌آید که به صورت زیر خواهد بود:

$$\min_{\alpha,d} F(\alpha, d) = \frac{1}{2} \left|\left| \mathbf{y} - \sum_{m=1}^{M} d_m^2 \mathbf{K}_m \alpha \right|\right|_2^2 + \frac{\lambda}{2}\alpha^T \sum_{m=1}^{M} d_m^2 \mathbf{K}_m \alpha + \frac{v}{2}\left(\sum_{m=1}^{M} d_m^2 - 1\right)^2, \tag{۶-۴}$$

که در آن پارامتر های λ و v می‌توانند به ترتیب مقادیری برابر با 10^{-3} و 10^{-2} داشته باشند [۱۴۳]. به منظور اطمینان از اینکه تمامی ضرایب d_m مثبت خواهند بود، از عبارت d_m^2 به جای آن استفاده می‌کنیم. حال می‌دانیم که $F(\alpha, \mathbf{d})$ یک تابع مشتق‌پذیر است و لذا با توجه به بردارهای α و $\mathbf{d}$ و استفاده از گرادیان‌ها می‌توان کمینه آن را به دست آورد. اکنون برای به دست آوردن بردار α و با در نظر گرفتن $\nabla_\alpha F(\alpha, \mathbf{d}) = 0$ خواهیم داشت:

$$\alpha = \left(\sum_{m=1}^{M} d_m^2 \mathbf{K}_m + \lambda \mathbf{I}\right)^{-1} \mathbf{y}, \tag{۷-۴}$$

که در آن $\mathbf{I}$ ماتریس همانی با ابعاد $l \times l$ است.

[1] Kernel Matrix

۲.۴.۴ یافتن ضرایب هسته‌ها در ترکیبات خطی

در مورد تعیین بردار ضرایب $\mathbf{d}^{۲}$ مینگ و همکارانش دو روش با رویکردهای تکراری و تحلیلی ارائه می‌دهند [۱۴۳]. در روش اول یا همان رویکرد تکراری آن‌ها از روش گرادیان کاهشی برای تعیین بردار ضرایب استفاده می‌کنند و فرمولی به صورت

$$\begin{aligned} \mathbf{d}_{t+۱}^{۲} &= \mathbf{d}_{t}^{۲} + \gamma_{\mathbf{d},t} \nabla_{\mathbf{d}^{۲}} F(\alpha, \mathbf{d}_t) \\ &= \mathbf{d}_{t}^{۲} + \frac{\gamma_{\mathbf{d},t}}{۲}(\mathbf{d}^{۲}\mathbf{B} - \mathbf{c}), \end{aligned} \tag{۸-۴}$$

ارائه می‌کنند که در آن $\mathbf{B}$ یک ماتریس با ابعاد $M \times M$ است و درایه‌های آن به صورت $b_{mn} = \alpha^T \mathbf{K}_{mn} \alpha + ۲v$ هستند به‌طوری که $\mathbf{K}_{mn} = \mathbf{K}_m \mathbf{K}_n + \mathbf{K}_n \mathbf{K}_m$ و $\mathbf{c}$ یک بردار M تایی با درایه‌هایی به صورت $c_n = \alpha^T \mathbf{K}_n (۲\mathbf{y} - \lambda\alpha) + ۲v$ است. همچنین $\gamma_{\mathbf{d},t} \geq ۰$ طول گام بهینه برای تکرار t-ام است.

در رویکرد تحلیلی با حل دستگاه معادلات $\nabla_{\mathbf{d}} F(\alpha, \mathbf{d}) = ۰$ و با مشتق‌گیری از آن بردار ضرایب $\mathbf{d}^{۲}$ به صورت زیر به دست می‌آید:

$$d_n \left[\alpha^T \left(\sum_{m=۱}^{M} d_m^{۲} \mathbf{K}_{mn} \right) \alpha + ۲v \sum_{m=۱}^{M} d_m^{۲} \right] = d_n c_n, \tag{۹-۴}$$

که در آن d_n در واقع درایه‌های بردار $\mathbf{d}$ بوده و بنابراین

$$\alpha^T \left(\sum_{m=۱}^{M} d_m^{۲} \mathbf{K}_{mn} \right) \alpha + ۲v \sum_{m=۱}^{M} d_m^{۲} = c_n. \tag{۱۰-۴}$$

این معادله در واقع یک معادله خطی است و با توجه به ضرایب $d_m^{۲}$ ها، می‌توان آن را به صورت زیر بازنویسی کرد:

$$\mathbf{d}_p^{۲} = \mathbf{c}_p \mathbf{B}_p^{-۱}, \tag{۱۱-۴}$$

که در آن $\mathbf{B}_p$ یک ماتریس مربعی $(M - N)$ تایی با درایه‌هایی به صورت $b_{mn} = \alpha^T \mathbf{K}_{mn} \alpha + ۲v$ است. همچنین $\mathbf{d}_p^{۲}$ و $\mathbf{c}_p$ دو بردار با درایه‌هایی به صورت $d_m^{۲}$ و c_n هستند. بر این اساس می‌توان ضرایب هسته‌ها را به‌طور بهینه و موثر به دست آورد.

۳.۴.۴ افزایش سرعت و کاهش هزینه‌های محاسباتی در فرآیند یادگیری دسته‌بندی کننده

همانطور که گفته شد ماتریس نمونه‌های آموزشی یک ماتریس چرخشی است و لذا ماتریس‌های هسته $\mathbf{K}_m$ متناظر با آن‌ها و همچنین ماتریس‌های حاصل از جمع، ضرب و وارون آن‌ها و در نتیجه $\mathbf{K}_{mn}$ نیز چرخشی و دارای

خصوصیات ویژه‌ای هستند. به این ترتیب می‌توان از این ویژگی‌ها استفاده نمود و عملگرهای پرهزینه ماتریسی مانند محاسبه وارون عبارت داده شده در معادله (۴-۷) را به طور ساده‌تری به دست آورد. بنابراین می‌توان بهینه‌سازی تابع هدف $F(\alpha, \mathbf{d})$ را با توجه به تبدیلات فوریه انجام داد. فرض کنید $\mathbf{k}_{mn}$ اولین سطر از ماتریس $\mathbf{K}_{mn}$ بوده و $\mathbf{k'}_m$ نیز اولین ستون از ماتریس $\mathbf{K}_m$ باشد. از آنجاکه ماتریس‌های هسته $\mathbf{K}_m$ متقارن هستند، واضح است که $\mathbf{k'}_m = \mathbf{k'}_m^T$. بنابراین می‌توان بردار α حاصل از معادله (۴-۷) را به منظور افزایش سرعت محاسبات به صورت زیر به دست آورد:

$$\alpha = \mathcal{F}_1^{-1}\left(\frac{\mathcal{F}(\mathbf{y})}{\mathcal{F}_1\left(\sum_{m=1}^{M} d_m^2 \mathbf{K}_m\right) + \lambda}\right). \qquad (۴-۱۲)$$

به این ترتیب معادلات به دست آمده در (۴-۱۱) را نیز می‌توان به صورت زیر نوشت:

$$\begin{aligned} \mathbf{k}_{mn} &= \mathcal{F}_1^{-1}(\mathcal{F}_1^*(\mathbf{k'_n}) \odot \mathcal{F}_1(\mathbf{k'_m})) + \mathcal{F}_1^{-1}(\mathcal{F}_1^*(\mathbf{k'_m}) \odot \mathcal{F}_1(\mathbf{k'_n})), \\ b_{mn} &= \alpha^T \mathcal{F}_1^{-1}(\mathcal{F}_1^*(\mathbf{k}_{mn} \odot \mathcal{F}_1(\alpha)) + 2v, \\ c_m &= (\lambda\alpha - \mathbf{y})^T \mathcal{F}_1^{-1}(\mathcal{F}_1^*(\mathbf{k}_m) \odot \mathcal{F}_1(\alpha)) - 1. \end{aligned} \qquad (۴-۱۳)$$

مجددا واضح است که ترکیب خطی حاصل از ماتریس‌های چرخشی خود یک ماتریس چرخشی است و لذا $\sum_{m=1}^{M} d_m^2 \mathbf{K}_m$ نیز چرخشی بوده و سطر اول به صورت $\sum_{m=1}^{M} d_m^2 \mathbf{k}_m$ است. به این ترتیب تابع هدف $F(\alpha, \mathbf{d})$ را می‌توان به فرم زیر بازنویسی نمود:

$$F_2(\alpha, \mathbf{d}) = \frac{1}{2}\left\|\mathbf{y} - \mathcal{F}_1^{-1}\left(\mathcal{F}_1^*\left(\sum_{m=1}^{M} d_m^2 \mathbf{k}_m\right) \odot \mathcal{F}_1(\alpha)\right)\right\|_2^2 + \frac{\lambda}{2}\alpha^T \mathcal{F}_1^{-1}\left(\mathcal{F}_1^*\left(\sum_{m=1}^{M} d_m^2 k_m\right) \odot \mathcal{F}_1(\alpha)\right) \qquad (۴-۱۴)$$

۴.۴.۴ شناسایی و تعیین موقعیت مرکزی شئ هدف

با توجه به معادلات به دست آمده، پاسخ فیلتر هم‌بستگی چند هسته‌ای به ازای هر نمونه آزمون مانند $\mathbf{z}_j = \mathbf{P}^j \mathbf{z}$ به ازای $j = 0, 1, \ldots, l-1$ و در قاب $\mathcal{I}$ به صورت زیر تعریف می‌شود:

$$y^j(\mathbf{z}) = \sum_{m=1}^{M} d_m^2 \sum_{i=0}^{l-1} \alpha_i k_m(\mathbf{z}_j, \mathbf{x}_i), \qquad (۴-۱۵)$$

که در آن $\mathbf{z}$ یک نمونه آزمون اولیه بوده و $\mathbf{P}$ نیز ماتریس جایگشت مربوطه برای تشکیل ماتریس چرخشی است. همچنین $\mathbf{x}_i = \mathbf{P}^i \mathbf{x}$ به ازای $i = 0, \ldots, l-1$ نیز در واقع نمونه آموزشی قاب قبلی است. بنابراین عبارت $\mathbf{y}(\mathbf{z})$ را می‌توان به صورت زیر نوشت:

$$\mathbf{y}(\mathbf{z}) \equiv \left(y^0(\mathbf{z}), y^1(\mathbf{z}), \ldots, y^{l-1}(\mathbf{z})\right)^T = \sum_{m=1}^{M} d_m^2 \mathbf{C}(\mathbf{k_m})\alpha, \qquad (۴-۱۶)$$

که در آن $\mathbf{k}_m = (k_{m,0}, \ldots, k_{m,l-1})$ و $k_{m,i} = k_m(\mathbf{z}, \mathbf{P}^i\mathbf{x})$ است. همچنین $\mathbf{C}(\mathbf{k}_m)$ یک ماتریس چرخشی است که $\mathbf{k}_m$ سطر اول آن است. بنابراین خواهیم داشت:

$$\mathbf{y}(\mathbf{z}) = \sum_{m=1}^{M} d_m^2 \mathcal{F}_1^{-1}\left(\mathcal{F}_1^*(\mathbf{k}_m) \odot \mathcal{F}_1(\alpha)\right), \qquad (۴-۱۷)$$

به‌طوری که در واقع عناصر $\mathbf{y}(\mathbf{z})$ موقعیت بهینه شئ در قاب $\mathcal{I}$ را مشخص می‌کند.

۵.۴.۴ به روزرسانی دسته‌بندی کننده بر اساس مدل حافظه‌ای چند کاناله

سیستم حافظه انسان را می‌توان به سه نوع حافظه لحظه‌ای[۱]، حافظه کوتاه مدت[۲] و حافظه بلند مدت[۳] تقسیم کرد و همین تقسیم‌بندی نیز در الگوریتم ارائه شده توسط گانگ و همکارانش لحاظ می‌گردد [۴۵]. در این مدل ابتدا سه فضای حافظه شامل یک کانال کنترل[۴] $\mathcal{C}$ و دو کانال اجرایی[۵] با نام‌های $\mathcal{R}_1$ و $\mathcal{R}_2$ ایجاد می‌شود. کانال کنترل در واقع برای ذخیره قالب‌های مربوط به شئ هدف طراحی شده و کانال‌های اجرایی $\mathcal{R}_1$ و $\mathcal{R}_2$ به ترتیب برای ذخیره سازی پارامترها و ویژگی‌های مربوط به دسته‌بندی کننده طراحی شده‌اند. هر یک از این کانال‌ها دارای سه نوع حافظه لحظه‌ای[۶]، کوتاه مدت[۷] و بلند مدت[۸] هستند. در هر کدام از این کانال‌ها، حافظه‌های IMS تنها یک قالب از شئ هدف و حافظه‌های STMS و LTMS هر کدام $\mathcal{N}$ قالب را ذخیره می‌کنند.

در این الگوریتم با توجه به قالب تخمینی که از نتیجه ردیابی به دست می‌آید (جعبه محاطی که دور شئ هدف کشیده می‌شود)، در فضای حافظه کانال کنترل، هیستوگرام خاکستری[۹] (q_t) مربوط به قالب تخمینی شئ هدف، در فضای حافظه کانال اجرایی $\mathcal{R}_1$ پارامترهای مربوط به دسته‌بندی کننده (α_t) و در فضای حافظه کانال اجرایی $\mathcal{R}_2$ ویژگی‌های هیستوگرام گرادیان (h_t) ذخیره می‌شوند. پس از به دست آوردن یا مشخص کردن ناحیه شئ هدف (در قاب اولیه)، q_t به عنوان یک قالب تخمینی در فضای حافظه IMS از کانال کنترل ذخیره می‌شود. سپس پارامترهای دسته‌بندی α_t و h_t نیز در فضای حافظه IMS مربوط به خودشان (به ترتیب $\mathcal{R}_1$ و $\mathcal{R}_2$) ذخیره می‌شوند. پس از آن مقادیر موجود در IMS هر یک از کانال‌ها به عنوان اطلاعات مربوط به قالب فعلی در فضای حافظه STMS متناظر با خودشان ذخیره می‌شوند. سپس نیاز است که حالت فعلی با حالت‌هایی که در فضاهای IMS، STMS

[1] Instantaneous Memory
[2] Short-term Memory
[3] Long-term Memory
[4] Control Channel
[5] Executive Channel
[6] Instantaneous Memory Space (IMS)
[7] Short Term Memory Space (STMS)
[8] Lont Term Memory Space (LTMS)
[9] Grayscale Histogram

و LTMS ذخیره شده‌اند مقایسه گردد و در صورت داشتن تشابه به میزان کافی با هریک از آن‌ها، از پارامترها و اطلاعات مربوط به آن برای دسته‌بندی و تعیین موقعیت هدف استفاده کند. در ادامه نحوه بررسی قالب‌ها در فضاهای مختلف و نحوه عملکرد آن‌ها پرداخته می‌شود.

تطبیق با فضای حافظه IMS و STMS

در فرآیند تطبیق فضای حافظه STMS دو حد آستانه تطبیقی $\mathcal{T}_{dc}$ و $\mathcal{T}_{ds}$ در نظر گرفته می‌شود. $\mathcal{T}_{dc}$ مربوط به قالب فعلی و $\mathcal{T}_{ds}$ مربوط به قالب‌های موجود در فضای حافظه STMS است. در فرآیند تطابق قالب تخمینی و قالب موجود در حافظه IMS، ابتدا معیار شباهت ρ بین آن‌ها تعریف و محاسبه می‌شود. به منظور بررسی اینکه آیا قالب تخمینی با قالب ذخیره شده تطابق دارد یا خیر، لازم است تا معیارهای ρ و $\mathcal{T}_{dc}$ با یکدیگر مقایسه شوند. ابتدا قالب تخمینی q_t با قالب فعلی موجود در IMS مقایسه می‌شود و مقدار ρ به دست می‌آید. اگر $\rho \geq \mathcal{T}_{dc}$ آنگاه تطبیق موفقیت آمیز بوده است (قالب تخمینی تا اندازه کافی با قالب ذخیره شده از قبل تطابق داشته است) و در غیر این صورت خیر. اگر این تطبیق با شکست مواجه شود، عملیات تطبیق با قالب‌های موجود در حافظه STMS ادامه می‌یابد. در این مرحله اگر به ازای یکی از قالب‌های موجود در STMS مانند قالب j-ام $\rho \geq \mathcal{T}_{ds}$ آنگاه عملیات تطابق با موفقیت انجام می‌شود و شمارنده تطبیق متناظر با آن قالب که با نماد E_j نشان داده می‌شود، یک واحد افزوده خواهد شد. همچنین قالب شئ هدف در کانال کنترل به صورت زیر به روزرسانی می‌شود:

$$q_t = (1-\epsilon)q_{t-1} + q' \qquad (۱۸-۴)$$

که در آن q_t و q_{t-1} ویژگی‌های خاکستری قاب‌های فعلی و قبلی هستند و ϵ نرخ به روزرسانی ویژگی خاکستری و q' قالب تطبیق شده است که همان j-امین قالب در STMS است. سپس ویژگی‌های HOG و همچنین پارامترهای مربوط به دسته‌بندی در کانال‌های اجرایی $\mathcal{R}_1$ و $\mathcal{R}_2$ نیز به ترتیب و به صورت زیر به روزرسانی می‌شوند:

$$\alpha_t = (1-\beta)\alpha_{t-1} + \alpha_{x'} \qquad (۱۹-۴)$$

$$h_t = (1-\beta)h_{t-1} + h' \qquad (۲۰-۴)$$

که در آن‌ها α_t و α_{t-1} پارامترهای دسته‌بندی کننده در قاب فعلی و قبلی هستند و $\alpha_{x'}$ نیز دسته‌بندی کننده تطبیق داده شده است (دسته‌بندی کننده متناظر با قالب j-ام در حافظه STMS کانال کنترل) همچنین h_t و h_{t-1} ویژگی‌های

HOG مربوط به قاب فعلی و قبلی هستند و همچنین h' نیز ویژگی HOG مربوط به قالب تطبیق داده شده است. علاوه بر آن β نیز نرخ به روزرسانی دسته‌بندی به شمار می‌آید.

تطبیق با فضای حافظه LTMS

مشابه با روند تطبیق در STMS، یک حدآستانه $\mathcal{T}_{dl}$ نیز برای این بخش در نظر گرفته می‌شود که برای تعیین موفق بودن عملیات تطابق در فضای LTMS استفاده می‌گردد. در این مرحله ابتدا قاب فعلی موجود در IMS با قاب‌های موجود در LTMS مقایسه می‌شود و معیار شباهت ρ' به دست می‌آید. اگر $\rho' \geq \mathcal{T}_{dl}$ آنگاه عملیات تطابق با موفقیت انجام شده است و در غیر این صورت خیر. اگر تطابق با موفقیت انجام شده باشد، متغیر شمارنده تطابق E_j متناظر یک واحد اضافه می‌گردد و اطلاعات موجود در کانال‌های اجرایی $\mathcal{R}_1$ و $\mathcal{R}_2$ طبق روابط ذکر شده به روزرسانی می‌شود. سپس قالب تطابق داده شده در فضای حافظه LTMS (قالب j-ام) به جایگاه اول در فضای STMS و به عنوان قالب فعلی انتقال می‌یابد و سایر قالب‌ها در STMS یک واحد به عقب منتقل می‌شوند.

همچنین می‌توان یک حد آستانه دیگر مانند $\mathcal{T}_E$ برای متغیر شمارنده تطابق‌ها در نظر گرفت. به طوری که اگر در تازه‌ترین توزیع مانند p_n از قالب‌های موجود در STMS به ازای هرقالب j اگر $E_j < \mathcal{T}_E$ آنگاه این بدان معناست که این قالب مفید نیست و لذا آن قالب حذف می‌شود. اما اگر $E_j \geq \mathcal{T}_E$ آنگاه آن قالب به فضای حافظه LTMS منتقل می‌گردد.

نکته‌ای که باید به آن توجه شود این است که در واقع فرآیند تطابق بین قالب‌ها در کانال کنترل انجام می‌شود و سپس عملیات به روزرسانی مقادیر موجود در کانال‌های اجرایی $\mathcal{R}_1$ و $\mathcal{R}_2$ با توجه به آن صورت می‌گیرد.

۵.۴ نتیجه‌گیری

در قسمت قبل الگوریتم پیشنهادی به‌طور مفصل تشریح و ارائه شد. تاکنون الگوریتم‌های متمایز کننده زیادی ارائه شده‌اند و بر اساس روش‌های مختلف عمل می‌کنند اما در این الگوریتم از روش فیلتر هم‌بستگی استفاده شده است که دلایل آن را می‌توان به سه دسته تقسیم‌بندی کرد.

۱. در روش فیلتر هم‌بستگی از تکنیک نمونه‌گیری متراکم استفاده شده است که همین تکنیک منجر به تشکیل مجموعه آموزش غنی‌تری نسبت به مجموعه آموزشی سایر روش‌ها می‌شود و همین امر باعث به دست آوردن یک تابع دسته‌بندی کننده مقاوم‌تر و بهتری می‌گردد.

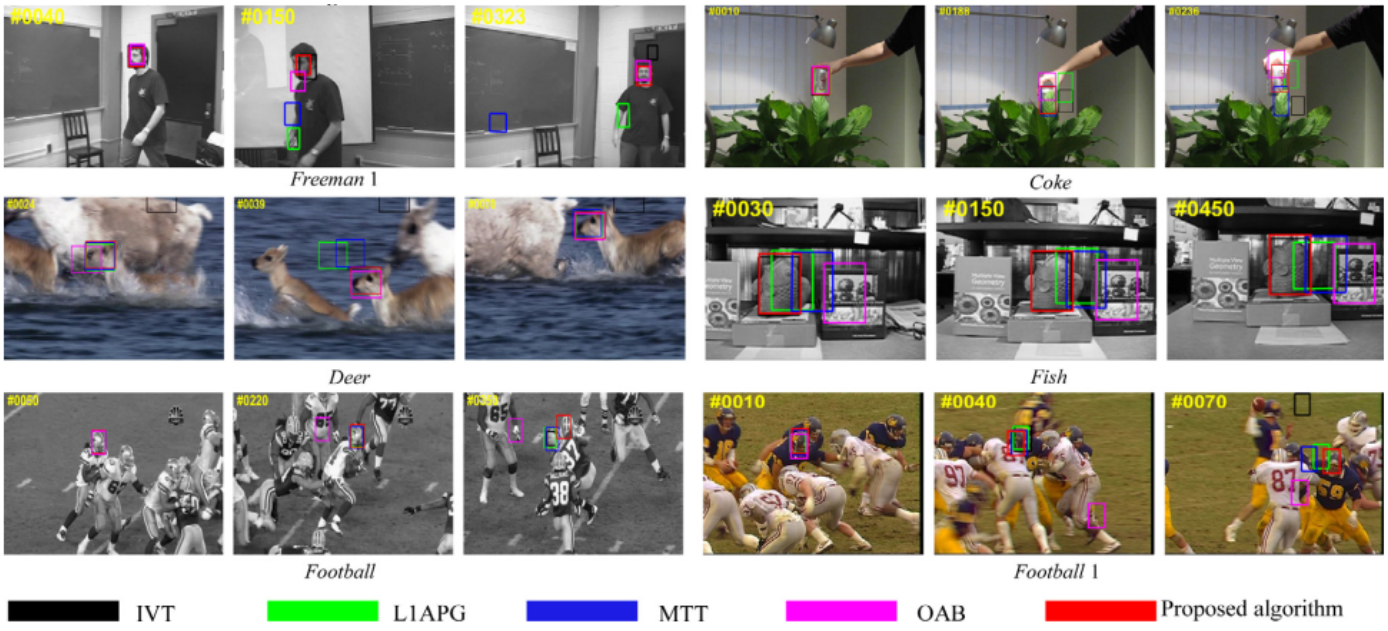
۲. در این الگوریتم با تشکیل یک ساختار چرخشی و بهره‌مندی از ویژگی‌های مفید آن و سپس استفاده از تبدیلات فوریه، از انجام عملیات‌های ماتریسی پرهزینه مانند محاسبه وارون ماتریس جلوگیری می‌شود و مقادیر مورد نیاز با کم‌ترین هزینه محاسباتی به دست می‌آیند.

۳. به واسطه کم بودن پیچیدگی‌ها و هزینه‌های محاسباتی، این الگوریتم قابلیت استفاده در ردیابی‌های بلادرنگ را فراهم می‌کند. قابل ذکر است که این افزایش سرعت و ردیابی بلادرنگ در حالتی اتفاق می‌افتد که الگوریتم به صورت برخط در حال یادگیری است و از هیچ مدل از پیش آموزش داده شده‌ای استفاده نمی‌شود.

اما در مورد الگوریتم پیشنهادی نیز این اطمینان وجود دارد که عملکرد آن به مراتب بهتر از روش‌های ارائه شده است؛ زیرا همانطور که گفته شد این الگوریتم از دو جهت بهبود یافته است. در گام اول و در مورد استفاده از الگوریتم فیلتر هم‌بستگی، به جای استفاده از یک هسته به منظور یافتن شباهت بین نمونه‌ها، از مجموعه‌ای از هسته‌ها استفاده می‌شود که ثابت شده در حالت کلی بهره‌مندی از چند هسته در الگوریتم‌های ردیابی متمایز کننده می‌تواند عملکرد ردیابی را تا حد زیادی بهبود ببخشد و عملیات تمایز بین شئ و محیط پس‌زمینه را با دقت بیشتری انجام دهد [۱۴۷، ۸۱].

از طرف دیگر واضح است که با دخیل کردن یک مدل مبتنی بر حافظه و به واسطه آن در نظر گرفتن تعداد قاب‌های بیشتر یا به عبارت دیگر تعداد حالت‌های بیشتری از شئ هدف در قاب‌های قبل، ردیاب طراحی شده قادر است در مواردی مانند انسداد که شئ به‌طور عادی قابل نمایش نیست بتواند آن را تشخیص داده و ردیابی کند.

در نهایت با ادغام این تکنیک‌ها و ویژگی‌ها می‌توان یک ردیاب پایدار، مقاوم و بلادرنگ را طراحی نمود. همچنین با توجه به اینکه این ردیاب برای هیچ دسته خاصی از اشیاء طراحی نشده است، می‌توان برای هر نمونه از اشیاء مورد نظر از آن استفاده نمود و محدودیتی از این جهت وجود ندارد. در تصویر زیر با استفاده از مجموعه داده‌های استاندارد مانند VOT و MOT [۱۵۴، ۸۲] برای محک ردیاب‌های مختلف، الگوریتم ارائه شده مورد بررسی قرار گرفته است که بهبود آن را می‌توان نسبت به سایر روش‌ها ملاحظه نمود.



شکل ۱.۴: نتایج حاصل از مقایسه عملکرد روش پیشنهادی و چند نمونه از روش‌های ردیابی مشابه دیگر.

ضمیمه ۱

پردازش تصویر و بینایی کامپیوتر

کلیت ماهیت پردازش تصویر را می‌توان به دو قسمت پردازش تصویر سطح پایین[۱] و پردازش تصویر سطح بالا[۲] تقسیم نمود. در پردازش تصویر سطح پایین یک تصویر ورودی به سیستم پردازشی داده می‌شود و خروجی آن سیستم نیز همچنان یک تصویر است که البته ممکن است با تصویر ورودی کمی متفاوت باشد. به عنوان مثال می‌توان از پردازش تصویر سطح پایین برای حذف نوفه‌ها، کاهش حجم، تغییرات در میزان روشنایی و چنین عملیات‌هایی بر روی تصاویر استفاده کرد. آنچه که واضح است این است که در این نوع پردازش تصویر هیچ‌گونه اطلاعات اضافه در مورد تصویر ورودی و اشیاء موجود در آن به دست نخواهد آمد. در مقابل، پردازش سطح بالا و اصطلاحاتی دیگر مانند بینایی ماشین [۳] و درک تصویر[۴] معمولا به جای بینایی کامپیوتر استفاده می‌شوند و اهداف یکسانی را دنبال می‌کنند. در همه این موارد پردازش بر روی یک تصویر ورودی انجام می‌شود اما خروجی حاصل شده لزوما یک تصویر نخواهد بود بلکه می‌تواند تفسیری از تصویر باشد.

از منظری دیگر اگر به پردازش تصویر از دیدگاه یک تعریف مفرد بنگریم، می‌توان آن را به عنوان یک گام پیش‌پردازش برای بینایی کامپیوتر در نظر گرفت و هدف از آن را استخراج ویژگی‌های اولیه تصویر مانند لبه‌ها، یال‌ها، فیلترهاو گوشه‌ها عنوان کرد. در این حالت تفاوت بین پردازش تصویر و بینایی کامپیوتر این خواهد بود که پردازش تصویر به طور کلی بر روی تصاویر خام تمرکز می‌کند و هیچ دانش افزون‌تری را ارائه نمی‌دهد در حالی که بینایی کامپیوتر قادر است توصیفات معنایی و تفاسیر مختلفی از تصاویر را ارائه دهد.

---

[1] Low-Level Image Processing
[2] High-Level Image Processing
[3] Machine Vision
[4] Image Understanding

Anaconda

Anaconda یک توزیع رایگان و متن باز از زبان های برنامه‌نویسی پایتون و R به شمار می‌آید که برای انجام محاسبات علمی و پردازش داده‌های حجیم مورد استفاده قرار می‌گیرد و هدف آن سهولت در مدیریت پکیج‌های مورد استفاده و مرتبط است. توزیع Anaconda هم اکنون توسط بیش از ۱۵ میلیون کاربر در حال استفاده است و شامل بیش از ۱۵۰۰ پکیج کاربردی در اکثر سیستم‌عامل‌های موجود است. اگر قبل از نصب Anaconda روی سیستم خود پایتون نصب شده باشد، با نصب آن یک پوشه به طور جداگانه ایجاد می‌شود که به پایتون اصلی آسیبی نمی زند و به طور جداگانه از هر کدام از آنها می‌توان استفاده کرد.

conda چیست؟

conda یک سیستم متن‌باز برای مدیریت پکیج‌هاست که قابل استفاده از Anaconda prompt و بر روی سیستم‌عامل‌های و بر ویندوز، لینوکس و مکینتاش است و به کاربر این امکان را می‌دهد تا پکیج‌های مورد نظر خود را جستجو، نصب، اجرا و به روزرسانی کند. این سیستم در ابتدا برای زبان پایتون ایجاد شده بود اما در حال حاضر می‌توان برای سایر زبان‌های برنامه نویسی مانند پایتون، جاوا، روبی، سی و فورترن نیز استفاده کرد. برای مطالعه اطلاعات کامل به این صفحه مراجعه شود.

conda همچنین دارای یک محیط مجزا تحت عنوان conda environment است که در واقع مسیری است که شامل پکیج‌های مختلف conda می‌شود. استفاده از محیط‌های مجزای conda در مواقعی مطرح می‌شود که به عنوان مثال نیاز داریم تا برای کاربردهای مختلف، از نسخه‌های مختلف یک کتابخانه استفاده کنیم. بدیهی است که بدون ایجاد محیط‌های مجزا این امکان وجود ندارد تا نسخه‌های مختلف از یک کتابخانه را همزمان نصب داشته باشیم. در این حالت می‌توان یک محیط conda مجزا ایجاد کرد و نسخه مورد نظر از کتابخانه مورد نیاز را نصب کرد. با این کار نسخه‌های مختلف از آن کتابخانه هیچ تداخلی با یکدیگر ندارند و به طور مجزا از یکدیگر قابل استفاده‌اند. برای اطلاعات بیشتر در مورد ایجاد و کار با محیط‌های conda به اینجا مراجعه شود.

(دستورات مهم و کاربردی برای کار با conda در اینجا به طور خلاصه آمده است.)

pip چیست؟

pip یک سیستم استاندارد مدیریت پکیج برای زبان برنامه‌نویسی پایتون است که این امکان را به کاربران می‌دهد تا بسته‌هایی را که به طور پیش‌فرض در کتابخانه‌های پایتون موجود نیستند را همراه با ملزومات آن‌ها نصب کنند. pip به قدری حائز اهمیت است که از نسخه ۴.۳ و بالاتر برای پایتون ۳ و از نسخه ۷.۲ و بالاتر برای پایتون ۲ به همراه فایل نصبی پایتون نصب خواهد شد. همچنین برای اطمینان از نصب آن بر روی سیستم می‌توان از دستور pip –version استفاده کرد. برای مطالعه اطلاعات دقیق‌تر و آگاهی از دستورات و نحوه کارکرد آن به این صفحه مراجعه شود.

تفاوت pip و conda

تفاوت بین conda و pip در نحوه مدیریت پکیج‌ها و ملزومات آن‌هاست. در واقع pip تمامی پیکج‌های مورد نظر و متعلقات آن‌ها را بدون در نظر گرفتن اینکه آیا تداخلی با دیگر پکیج‌هایی که قبلا نصب شده‌اند دارند یا خیر، نصب می‌کند. به عنوان مثال اگر تنسورفلو قبلا نصب شده باشد و با دستور pip یک پکیج دیگر(که نیاز به نسخه دیگری از کتابخانه‌ای مانند numpy دارد) نصب شود، ممکن است در حین اجرا تنسورفلو متوقف شود. به عبارت دیگر دستور pip تنها پکیج مورد نظر و ملزومات آن را بدون در نظر گرفتن سازگاری یا ناسازگاری آن با سایر پکیج‌ها نصب می‌کند. این در حالی است که conda محیط فعلی و تمامی پکیج‌های نصب شده را بررسی می‌کند و نسخه‌های سازگار با آن‌ها را نصب می‌کند یا به کاربر اطلاع می‌دهد که عملیات مد نظر شما قابل انجام نیست.

کتابخانه‌های مورد استفاده در بینایی ماشین

Numpy

numpy یک کتابخانه مهم و کاربردی برای زبان برنامه نویسی پایتون است. با استفاده از این کتابخانه امکان استفاده از آرایه‌ها و ماتریس‌های چند بعدی و با ابعاد بزرگ فراهم می‌شود.

آرایه‌های numpy مانند لیست‌های پایتون هستند با این تفاوت که عملکردی به مراتب بهتر از آن دارند. دستکاری یک آرایه numpy ساده‌تر از دستکاری لیست پایتون است و می‌توان از یک آرایه numpy به جای چند لیست پایتون استفاده کرد. همچنین آرایه‌های numpy محاسباتی سریع‌تر از لیست‌ها دارند و برای اجرای عملیات ریاضیاتی و

منطقی بسیار کارآمدتر هستند. به طور خاص فرض کنید تصویری با ابعاد مثلا ۴۵۲ × ۵۸۹ در فضای رنگی RGB در اختیار داریم. بنابراین هرکدام از پیکسل‌های تصویر مورد نظر شامل سه مقدار قرمز، سبز و آبی هستند و مقدار هر کدام از آن‌ها در بازه صفر و ۲۵۵ می‌تواند قرار گیرد. به این ترتیب در واقع یک ماتریس با ابعاد ۴۵۲ × ۵۸۹ در اختیار داریم که هرکدام از درایه‌های آن یک پیکسل از تصویر هستند. با استفاده از کتابخانه numpy می‌توان این تصویر را در قالب یک آرایه سه بعدی با ابعاد ۳ × ۴۵۲ × ۵۸۹ ذخیره کرد. این نحوه ذخیره‌سازی نه تنها باعث انجام محاسبات آسان‌تر و سریع‌تر بر روی تصویر خواهد شد، بلکه بسیاری از کتابخانه‌های دیگر مانند

openCV، scikit-image و h5py که در زمینه‌های پردازش تصویر و یادگیری ماشین وجود دارند، آرایه‌های ساخته شده توسط این کتابخانه را به عنوان ورودی دریافت کرده و عملیات‌های لازم را بر روی آن انجام می‌دهند. برای اطلاع از نحوه استفاده از این کتابخانه و متدهای آن به اینجا مراجعه شود. لینک مفید

Scipy

Scipy یکی دیگر از کتابخانه‌های پایه‌ای پایتون است که به صورت آزاد و متن باز ارائه شده است و برای محاسبات علمی و مهندسی از آن استفاده می‌شود. این کتابخانه ماژول‌های مختلفی برای بهینه‌سازی، جبر خطی، انتگرال، درون‌یابی، معادلات دیفرانسیل و پردازش تصویر و سیگنال را در بر می‌گیرد و دارای تعداد زیادی از کلاس‌ها و الگوریتم‌های مختلف برای تغییر داده‌ها و مصورسازی آن‌هاست. علاوه بر آن شامل توابع مختلفی برای کار با اعداد و حل معادلات دیفرانسیل و کارهایی از این قبیل است. برای آشنایی با نحوه استفاده از این کتابخانه و قابلیت‌های آن به اینجا مراجعه شود.

Matplotlib

این کتابخانه در سال ۲۰۰۳ ایجاد شد و دستورات آن از دستورات نرم افزار MATLAB الهام گرفته شده است. همانطور که از اسم آن مشخص است، این کتابخانه برای رسم نمودارهای مختلف مورد استفاده قرار می‌گیرد. علوه بر آنبه منظور درج اطلاعات اضافه در تصاویر نیز می‌توان از آن استفاده کرد. به عنوان مثال هنگامی که قرار است یک تصویر دسته بندی شود، برای درج میزان دقت دسته بندی در تصویر و همچنین نمایش آن از این کتابخانه استفاده می‌شود.

در صورت آشنا بودن با دستورات مربوط به رسم نمودارها در نرم افزار MATLAB، می‌توان از ماژول pyplot استفاده کرد زیرا دستورات هردو مشابه یکدیگر است و همچنین از این طریق امکان کنترل بیشتر بر روی جزئیات

نمودارها فراهم خواهد شد.

Matplotlib یک کتابخانه پایه‌ای است و چند کتابخانه دیگر مانند Seaborn ، Bokeh ، Basemap و NetworkX بر اساس آن و برای رسم نقشه ها و نمودارهای پیشرفته‌تر طراحی و ارائه شده‌اند. برای اطلاع از نحوه نصب و کار با این کتابخانه به اینجا مراجعه شود. لینک مفید

PIL یا Pillow

کتابخانه Python Imaging Library (یا به اختصار PIL) یک کتابخانه آزاد برای زبان برنامه نویسی پایتون است و برای پشتیبانی از باز کردن، ایجاد تغییرات و ذخیره تصاویر در قالب‌های مختلف طراحی شده است. این کتابخانه در نسخه‌های آخرین خود با اضافه کردن قابلیت‌های جدید و تحت عنوان Pillow معرفی گردید و در در توزیع‌های سیستم عامل لینوکس مانند Debian و Ubuntu جایگزین PIL شد.

کتابخانه Pillow از فرمت‌های مختلف مانند Gif، PNG و JPG پشتیبانی می‌کند و دارای چند سازوکار استاندارد برای ایجاد تغییرات در تصاویر است که از جمله آن‌ها می‌توان به موار زیر اشاره کرد:

- تغییرات در پیکسل‌های تصاویر
- فیلتر کردن تصویر، مانند تار کردن، شکل دادن، صاف کردن، یا یافتن لبه
- کنترل شفافیت، تنظیم روشنایی و تناسب رنگ
- تغییرات اندازه تصاویر و چرخش آن‌ها
- اضافه کردن متن به تصاویر و کارهایی از این قبیل

البته شایان ذکر است که برای انجام کارهای جدی‌تر و اساسی‌تر در حوزه کار با تصاویر بهتر است از کتابخانه‌هایی مانند OpenCV یا SimpleCV استفاده شود. برای اطلاع از نحوه نصب و استفاده از این کتابخانه به اینجا مراجعه شود.

OpenCV

OpenCV یا به طور کامل‌تر Open Source Computer Vision یک کتابخانه متن باز با بیش از۲۵۰۰ الگوریتم بهینه‌سازی و یک مجموعه کامل از الگوریتم‌های کلاسیک و نوین در زمینه بینایی کامپیوتر و یادگیری ماشین است

که می‌توان از آن‌ها برای تشخیص و شناسایی چهره، تشخیص اشیاء، تشخیص رفتارها در فایل‌های ویدئویی، ردیابی اشیاء متحرک، استخراج مدل‌های سه بعدی از اشیاء، یافتن تصاویر مشابه با یک تصویر از دیتاست مورد نظر، حذف قرمزی چشم حاصل از فلش دوربین و از این قبیل کارها استفاده کرد.

این کتابخانه که یکی از مهم‌ترین و پر استفاده‌ترین کتابخانه‌ها در حوزه پردازش تصویر است، توسط یک اجتماع ۴۷ هزار نفری و با استفاده از زبان C/C++ توسعه داده شده است اما در زبان‌های برنامه‌نویسی پایتون و جاوا و همچنین نرم‌افزار متلب نیز قابل استفاده است اما یادگیری و تسلط کافی بر این کتابخانه به دلیل وسعت آن کمی دشوار و زمان‌بر خواهد بود.

تا کنون بسیاری از استارتاپ‌های موفق و حتی شرکت‌های بزرگ فناوری مانند گوگل، یاهو، مایکروسافت، اینتل، هوندا و تیوتا نیز در پروژه‌های مختلف خود از این کتابخانه استفاده کرده‌اند که از بین این پروژه‌ها به مواردی مانند هدایت ربات‌ها برای برداشتن اشیاء، نظارت بر تجهیزات معدنی، بهبود عملکرد غریق نجات‌ها در استخرهای پر ازدحام، بررسی باند و درگاه ورود فرودگاه‌ها و بازرسی برچسب گذاری‌ها بر روی محصولات کارخانه‌ها می‌توان اشاره کرد. به منظور کسب اطلاعات کامل‌تر در را بطه با این کتابخانه و مطالعه قابلیت‌های بیشتر آن به اینجا مراجعه شود.

SimpleCV

SimpleCV را می‌توان به عنوان یک چهارچوب (framework) شناخت که از تعدادی از کتابخانه‌های دیگر از جمله OpenCV در خود استفاده می‌کند و عملکردی شبیه به آن دارد. هدف اصلی طراحی آن یادگیری سریع‌تر و استفاده آسان‌تر نسبت به OpenCV برای کاربردهای بینایی کامپیوتر بوده است که برای این منظور مجموعه‌ای از ساده‌سازی‌ها بر روی آن صورت گرفته است. بدیهی‌ست که این ساده سازی‌ها در نهایت منجر به کاهش تعدادی از توابع و قابلیت‌های پیشرفته‌تر نسبت به OpenCV شده است اما با همه این وجود SimpleCV برای کاربردهای اولیه و پیاده‌سازی‌های ساده گزینه بسیار مناسبی است. همچنین این کتابخانه با زبان پایتون نوشته شده است و برخلاف OpenCV، تنها در زبان برنامه‌نویسی پایتون نیز قابل استفاده است. برای آشنایی با نحوه استفاده از این کتابخانه به اینجا مراجعه شود.

Scikit-learn

Scikit-learn یک کتابخانه پایتون بوده و مربوط به یادگیری ماشین است که مبتنی بر سایر کتابخانه‌های اساسی

مانند numpy و matplotlib طراحی شده است. از این کتابخانه می‌توان در مواردی مانند دسته‌بندی شامل KNN)، رگرسیون خطی و لجستیک، پیش پردازش‌ها مثل Min-Max Normalization و الگوریتم‌های خوشه‌بندی مثل SVM و K-means از آن استفاده کرد. برای آشنایی با نحوه استفاده از این کتابخانه به اینجا مراجعه شود.

Scikit-image

ک کتابخانه متن باز برای پردازش تصویر در زبان برنامه‌نویسی پایتون است که شامل الگوریتم‌های قسمت‌بندی(segmentation)، تبدیلات هندسی، تغییرات رنگ، فیلترگذاری و کاربردهایی از این قبیل می‌شود. این کتابخانه بر پایه آرایه‌های ساخته شده توسط کتابخانه numpy عمل می‌کند و از کتابخانه‌ها و ماژول‌های مختلف و مهم دیگر مانند matplotlib و scipy استفاده می‌کند. برای آشنایی با نحوه استفاده از این کتابخانه به اینجا مراجعه شود.

h5py

واضح است که به دلیل کمبود فضای حافظه نمی‌توان کل اطلاعات یک دیتاست را در آن بارگزاری کرده و از آن استفاده نمود. بنابراین نیازمندی به یک کتابخانه برای مدیریت این وظیفه به شدت احساس می‌شود. h5py کتابخانه‌ای استاندارد برای ذخیره داده‌های عددی حجیم است و از آرایه‌های ساخته شده توسط کتابخانه numpy پشتیبانی می‌کند. به عنوان مثال اگر چندین ترابایت اطلاعات به صورت آرایه‌های numpy بر روی دیسک ذخیره شده باشند، می‌توان آن‌ها را به شیوه دلخواه برش زده و از آن‌ها استفاده کرد. علاوه بر آن می‌توان چندین دیتاست را در یک فایل ذخیره نمود و آن‌ها را بر حسب نیاز دسته بندی و برچسب گذاری کرد. همچنین می‌توان مجموعه ویژگی‌های استخراج شده را در یک دیتاست h5py ذخیره کرده و سپس سایر کتابخانه‌های دیگر را بر حسب نیاز بر روی آن اعمال کرد. برای آشنایی با نحوه استفاده از این کتابخانه به اینجا مراجعه شود.

Pandas

Pandas کتابخانه‌ای متن باز برای پایتون است که امکان ارائه ساختارهای داده‌ای با کارایی بالا و برای تجزیه و تحلیل داده‌ها در زمینه‌های مختلف مانند علوم اجتماعی و مهندسی، اقتصاد و آمار را فراهم می‌کند. در این کتابخانه سه نوع ساختار داده‌ای مهم و کاربردی وجود دارد که عبارت‌اند از Series یا همان آرایه یک بعدی، DateFrame یا همان آرایه دو بعدی که از یک سری سطرها و ستون‌ها تشکیل شده است و می‌توان آن را همانند جدول‌های دیتاست در نظر گرفت که شامل نمونه‌ها و ویژگی‌های آن‌هاست و مورد سوم Panel ها نام دارند که در واقع آرایه‌های

سه بعدی یا بالاتر هستند.

در این کتابخانه ابزارهای متنوعی وجود دارد که قابلیت‌های زیادی برای کاربران آن فراهم می‌کند. از جمله این قابلیت‌ها می‌توان به مواردی مانند پشتیبانی از فرمت‌های مختلف فایل برای خواندن و یا نوشتن داده‌ها بر روی آن ها، مدیریت داده‌های گم شده در جداول نمونه‌ها، حذف و درج ستون‌ها بنابر نیاز کاربر، توانایی برش و ادغام جداول مختلف و مواردی از این قبیل اشاره کرد. برای آشنایی با نحوه استفاده از این کتابخانه به اینجا مراجعه شود.

IPython

IPython که مخفف Interactive Python است، در واقع یک پوسته فرمان تعاملی (interactive command shell) برای زبان برنامه‌نویسی پایتون است که در سال ۲۰۰۱ ارائه شد. یکی از روش‌های مرسوم برای اجرای برنامه‌های پایتون، نوشتن یک فایل با پسوند py. است که در آن دستورات نوشته شده از ابتدا تا انتها یکی پس از دیگری اجرا شده و در نهایت یک یا چند خروجی نمایش داده می‌شود. اما با استفاده از پوسته فرمان IPython می‌توان بلافاصله بعد از نوشتن یک دستور خروجی آن را مشاهده کرد. این قابلیت در مواقع تحلیل و بررسی یک سری داده و یا کار با مدل‌های محاسباتی بسیار حائز اهمیت و کاربردی است.

Jupyter Notebook

در سال ۲۰۱۱ IPython یک ابزار با نام Notebook معرفی کرد که از نرم‌افزارهای دیگر نظیر Mathenatica الهام گرفته شده بود. Notebook در واقع یک رابط قدرتمند مبتنی بر وب برای پایتون ارائه می‌داد.

در مقایسه با خط فرمان (terminal) اصلی IPython، Notebook امکان استفاده از یک ویرایش‌گر متن مناسب‌تر را فراهم کرده است که می‌توان انواع متون و فرمول‌های ریاضی را نوشت و همچنین از قابلیت‌های گرافیکی بیشتری برخوردار بود. در سال ۲۰۱۵ توسعه‌دهندگان سازماندهی‌های بیشتری بر این پروژه اعمال کرده و نام آن را به Jupyter Notebook تغییر دادند. از آن پس این رابط علاوه بر زبان پایتون از حدود ۴۰ زبان برنامه‌نویسی دیگر نیز پشتیبانی می‌کند.

هم اکنون با استفاده از این ابزار کاربران و برنامه‌نویسان می‌توانند در کنار برنامه‌های خود علاوه بر کدهای موجود، یادداشت‌ها و توضیحات مختلف، اشکال و نمودارها و حتی فرمول‌های ریاضی را اضافه کنند و به عنوان

یک کتابچه در قالب‌های pdf و html ذخیره کرده و با افراد دیگر به اشتراک بگذارند. همچنین می‌توانند در صورت نیاز ورودی‌ها را تغییر داده و خروجی‌های متناظر را بلافاصله مشاهده کنند. لازم به ذکر است که پسوند فایل‌هایی که با استفاده از Jupyter Notebook ایجاد می‌شوند، ipynb. خواهد بود. برای مشاهده امکانات و نحوه کار با این کتابچه‌ها به صورت آنلاین می‌توان از نسخه آزمایشی وب‌سایت Jupyter استفاده کرد. در اینجا هر کتابچه از واحدهایی به نام سلول تشکیل می‌شود که هر سلول می‌تواند شامل کدهای برنامه یا متن و توضیحات (در قالب Markdown) باشد.

به منظور راهنمایی برای نصب IPython که هسته اصلی و پیش فرض برای Jupyter Notebook است می‌توان از اینجا استفاده کرد. همچنین برای سهولت در راه‌اندازی آن پیشنهاد می‌شود تا از توزیع Anaconda استفاده شود تا تمامی ملزومات و کتابخانه‌های مربوطه به طور خودکار نصب گردد. برای آشنایی کامل‌تر به وب‌سایت IPython مراجعه شود. لینک مفید

Compute Unified Device Architecture (CUDA)

تکنولوژی CUDA در سال ۲۰۰۷ برای اولین بار توسط شرکت Nvidia که تولید کننده کارت‌های گرافیک یا همان VGA ها بود مطرح شد که بر اساس آن کارت‌های گرافیکی این شرکت مجهز به تکنولوژی CUDA شدند. در این حالت شرکت Nvidia بر روی VGA های تولیدی خود یک یا چند CPU قرار می‌داد تا این کارت‌های گرافیکی بتوانند قابلیت پردازش اطلاعات را به صورت مجزا از پردازنده مرکزی داشته باشند. سپس به دلیل عدم ایجاد تداخل با نام CPU، به CPU هایی که بر روی کارت‌های گرافیکی قرار می‌گرفتند، Graphics Processing Unit(GPU) گفته می‌شد که البته توان محاسباتی GPU به خاطر ساختار و معماری متفاوت خود و همچنین تعداد هسته‌های بسیار زیاد از توان محاسباتی CPU بالاتر است. به طور ساده‌تر CUDA یک پلتفرم نرم‌افزاری است باعث می‌شود تا صدها تراشه گرافیکی مجزای Nvidia در کنار هم به پردازش موازی جهت پردازش حجیم ترین حجم اطلاعات با کارایی بالا بپردازند. برای استفاده از CUDA لازم است که سه مولفه را در اختیار داشته باشیم.

۱. کارت گرافیک شرکت Nvidia و مجهز به پردازنده و با قابلیت پشتیبانی از CUDA.

۲. CUDA Toolkit که در واقع محیط توسعه‌ای است شامل کامپایلر، ابزارهای رفع خطا (دیباگ) و ابزارهای ارزیابی عملکرد.

۳. CUDA Software Development Kit (SDK) که شامل کتابخانه‌های پایه‌ای و تعدادی نمونه کدهای مختلف است.

لینک مفید

لینک مفید

cuDNN

cuDNN کتابخانه‌ای مبتنی بر CUDA و شامل مجموعه‌ای از بهینه‌سازی‌های سطح پایین است. در این کتابخانه برخی فرآیندهای استاندارد موجود در حوزه شبکه‌های عصبی عمیق (Deep Neural Network) مانند لایه‌های فعال‌سازی، نرمال‌سازی، فازهای forward و backward پیاده‌سازی شده‌اند و هدف اصلی آن این است که با استفاده از کارت‌های گرافیکی که از cuDNN پشتیبانی می‌کنند، سرعت پردازش‌های موجود در این حوزه را افزایش دهد. در حال حاضر نیز بسیاری ازتوسعه‌دهندگان و محققان در زمینه یادگیری ماشین به منظور افزایش و بهبود عملکرد GPU کارت‌های گرافیک خود متکی به این کتابخانه هستند زیرا این کتابخانه این امکان را فراهم می‌کند تا آن‌ها به جای صرف زمان بر روی GPU های سطح پایین و با عملکرد نه چندان مناسب، بر روی آموزش و توسعه شبکه‌های عصبی خود تمرکز کنند. cuDNN به طور گسترده در چهارچوب‌های موجود در زمینه یادگیری عمیق مانند Caffe ، Tensorflow ، Keras و PyTorch استفاده می‌شود. شرکت Nvidia نشان داده است که کاربران در هنگام استفاده از Caffe به همراه cuDNN از افزایش سرعت ۱۰ برابری برخوردار خواهند شد.

لینک مفید

Tensorflow

در حال حاضر Tensorflow محبوب‌ترین کتابخانه در حوزه یادگیری ماشین است که توسط تیم Google Brain شرکت گوگل طراحی شده است. Tensorflow در سال ۲۰۱۵ به عنوان یک کتابخانه متن‌باز معرفی شد و همین کار باعث گردید تا بسیاری از کاربران برای استفاده از آن تمایل نشان دهند و شهرت و محبوبیت بسیاری کسب کند.

در حقیقت Tensorflow این قابلیت را به برنامه‌نویسان و دانشمندان مهندسی داده می‌دهد تا با بهره‌گیری از API های موجود، به طور ساده‌تر و بدون نیاز به درگیر شدن با جزئیات مربوطه در حوزه یادگیری ماشین فعالیت کنند. به عنوان مثال این کتابخانه باعث کاهش چشم‌گیر پیچیدگی‌ها برای آموزش سیستم‌های تشخیص گفتار، بینایی ماشین

و پردازش زبان‌های طبیعی شده است. به طور خاص‌تر با استفاده از Tensorflow می‌توان شبکه‌های عصبی عمیق برای دسته‌بندی اعداد با دست‌خط‌های مختلف، تشخیص تصاویر، پیاده‌سازی شبکه‌های عصبی بازگشتی(RNN)، شبکه‌های عصبی پیچشی(CNN) و کاربردهایی از این قبیل را آموزش داده و اجرا نمود.

Tensorflow این امکان را می‌دهد تا توسعه‌دهندگان گراف‌های جریان داده‌ای (dataflow graphs) ایجاد کنند. این مدل در واقع ساختاری است که نحوه جریان داده‌ها در طول این گراف‌ها را توصیف می‌کند. هر رأس گراف بیان‌گر یک عملگر ریاضی است و هر یال بین رئوس را می‌توان یک آرایه چند بعدی و یا یک تنسور در نظر گرفت. همه این مدل‌ها ساختاری انتزاعی دارند که توسط زبان برنامه‌نویسی پایتون می‌توان آن‌هارا پیاده‌سازی کرد. رئوس و تنسورها در Tensorflow در واقع اشیاء زبان پایتون هستند. این کتابخانه به این علت Tensorflow نامیده می‌شود که ورودی آن آرایه‌های جند بعدی هستند که معمولا با نام Tensor نامیده می‌شوند. همچنین می‌توان ترتیبی از فلوچارت‌های عملیاتی را ساخت که بر روی داده‌ها اعمال می‌شوند. به طوری که داده‌ها از یک طرف وارد می‌شوند و به طور جریان‌گونه در بین این عملگرها حرکت می‌کنند و از طرف دیگر به عنوان خروجی خارج می‌شوند.

Tensorflow تنها کتابخانه یادگیری عمیق موجود نیست اما در حال حاضر محبوب‌ترین و پر استفاده‌ترین آن‌هاست. از دیگر کتابخانه‌های مشابه موجود می‌توان مواردی مانند Torch که توسط محققان سوئیسی ساخته شد، Caffe که توسط تعدادی از دانشجویان دوره دکتری دانشگاه کالیفرنیا توسعه داده شد و در ادامه توسط فیسبوک و تحت عنوان Caffe2 عرضه شد را نام برد. علاوه بر آن در نوامبر ۲۰۱۷ گوگل نسخه سبک‌تر این کتابخانه با نام TensorFlow Lite را معرفی کرد که هدف اصلی این کار ارائه نسخه‌ای با قابلیت اجرا بر روی دستگاه‌هایی با سخت‌افزار ضعیف‌تر مانند تلفن‌های همراه بود.

لینک مفید لینک مفید لینک مفید لینک مفید

Keras

Keras در واقع کتابخانه‌ای front end برای کار در حوزه یادگیری عمیق است که API سطح بالاتر و ساده‌تری ارائه می‌دهد. دستورات و توابع موجود در این کتابخانه در سطوح پایین‌تر با استفاده از یک back end نوشته شده‌اند و بر اساس آن‌ها عمل می‌کنند. در ابتدا فریم‌ورک دیگری به نام Theano به عنوان یک back end برای Keras در نظر گرفته شده بود و بر اساس آن استفاده می‌شد اما پس از مدتی قابلیت استفاده از Tensorflow به عنوان یک back end دیگر نیز به آن اضافه شد. البته می‌توان بدون استفاده از Keras نیز از Tensorflow استفاده کرد اما استفاده از

Keras به عنوان یک API سطح بالاتر می‌تواند پیچیدگی‌های موجود را کاهش دهد و راحتی بیشتری فراهم می‌کند.
لینک مفید لینک مفید لینک مفید لینک مفید

Convolutional Architecture for Fast Feature Embedding (CAFFE)

Caffe یک فریم‌ورک برای یادگیری عمیق است که توسط دانشجویان دوره دکتری دانشگاه برکلی کالیفرنیا طراحی شده است. این فریم‌ورک توسط زبان C++ توسعه داده شده و دارای رابط (Interface) پایتون نیز هست. Caffe از بسیاری از معماری‌های یادگیری عمیق و کاربردهایی مانند دسته‌بندی تصاویر (image classification) و تقسیم بندی تصاویر (image segmentation) و همچنین شبکه‌های عصبی CNN و RCNN پشتیبانی می‌کند.

در آپریل ۲۰۱۷ شرکت فیسبوک با اضافه کردن ویژگی‌های دیگری همانند شبکه‌های عصبی بازگشتی (RNN) نسخه دیگری از Caffe با نام Caffe2 را ارائه داد و در اواخر مارس ۲۰۱۸ Caffe2 با فریم‌ورک Pytorch ادغام گردید.
لینک مفید

مقایسه Tensorflow و Caffe

در مورد مقایسه این دو فریم‌ورک می‌توان موارد ذیل را ذکر کرد:

- Tensorflow نسبت به Caffe نگرش برنامه‌نویسی برجسته‌تری در طراحی شبکه‌های مختلف دارد. این یعنی برای افرادی که دارای پیش‌زمینه برنامه‌نویسی هستند یا استفاده از دیدگاه برنامه‌نویسی برای ساخت شبکه‌های عصبی را ترجیح می‌دهند، Tensorflow یک گزینه مناسب‌تر است. این در حالی است که در Caffe کاربران باید لایه‌های موجود در شبکه‌های مختلف را به صورت پارامتری و کدهایی شبیه به XML تعریف کنند که معمولا کمی دشوارتر است.

- Tensorflow با هدف مصارف تحقیقاتی و سرورها طراحی شده اما Caffe برای دستگاه‌های با سخت‌افزار محدود مانند تلفن‌های همراه ایجاد شده است.

- شرکت‌هایی مانند فیسبوک ادعا می‌کنند که طبق معیارهای داخلی آن‌ها Caffe عملکری ۲.۱ تا ۵ برابری بیش‌تری نسبت به Tensorflow دارد و لذا از Caffe برای کاربردهای خود استفاده می‌کنند.

- مطابق نظر بسیاری از کاربران Caffe برای کاربردهای یادگیری عمیق بر روی تصاویر عملکرد بسیار خوبی دارد اما در مورد مدل‌های متوالی و شبکه‌های عصبی بازگشتی (RNN) عملکرد ضعیف‌تری دارد. از طرف دیگر Tensorflow هم بر روی مدل‌های متوالی و هم بر روی تصاویر عملکرد بسیار خوبی دارد اما درک مدل‌های مبتنی بر گراف که Tensorflow استفاده می‌شود معمولا برای مبتدیان کمی دشوار است.
- سرعت توسعه Tensorflow بیش‌تر از سرعت رشد و توسعه Caffe است و این احتمال وجود دارد که پس از مدتی توسعه Caffe به طور کامل متوقف شود.
- به دلیل محبوبیت و استفاده بیش‌تر از Tensorflow، منابع و راهنمایی‌های بیش‌تری از آن نسبت به Caffe وجود دارد.
- Tensorflow را با سیستم مدیریت پکیج pip در پایتون به راحتی می‌توان نصب و راه‌اندازی کرد در حالی که روش آسانی برای نصب Caffe وجود ندارد و تمامی کدهای مربوط به آن را باید کامپایل کرد.
- Tensorflow به گونه‌ای طراحی شده است که API سطح بالاتری را با استفاده از زبان پایتون ارائه می‌دهد (خصوصا اگر از Keras استفاده شود) اما Caffe دارای API به مراتب سطح پایین‌تری است.

لینک مفید لینک مفید لینک مفید

مراجع

[1] Amer, A. Memory-based spatio-temporal real-time object segmentation for video surveillance. In *Real-Time Imaging VII* (2003), vol. 5012, International Society for Optics and Photonics, pp. 10–21.

[2] Amit, Y., and Geman, D. A computational model for visual selection. *Neural computation 11*, 7 (1999), 1691–1715.

[3] Applegate, K. T., Besson, S., Matov, A., Bagonis, M. H., Jaqaman, K., and Danuser, G. plustip-tracker: Quantitative image analysis software for the measurement of microtubule dynamics. *Journal of structural biology 176*, 2 (2011), 168–184.

[4] Avidan, S. Support vector tracking. *IEEE transactions on pattern analysis and machine intelligence 26*, 8 (2004), 1064–1072.

[5] Babenko, B., Yang, M.-H., and Belongie, S. Visual tracking with online multiple instance learning. In *2009 IEEE Conference on computer vision and Pattern Recognition* (2009), IEEE, pp. 983–990.

[6] Babenko, B., Yang, M.-H., and Belongie, S. Robust object tracking with online multiple instance learning. *IEEE transactions on pattern analysis and machine intelligence 33*, 8 (2010), 1619–1632.

[7] Bach, F. R., Lanckriet, G. R., and Jordan, M. I. Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the twenty-first international conference on Machine learning* (2004), ACM, p. 6.

[8] Ballard, D. H., and Brown, C. M. Computer vision. englewood cliffs. *J: Prentice Hall* (1982).

[9] Bar-Shalom, Y., Willett, P. K., and Tian, X. *Tracking and data fusion*. YBS publishing Storrs, CT, USA:, 2011.

[10] Bay, H., Tuytelaars, T., and Van Gool, L. Surf: Speeded up robust features. In *European conference on computer vision* (2006), Springer, pp. 404–417.

[11] Bengio, Y., et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning 2*, 1 (2009), 1–127.

[12] Bertalmio, M., Sapiro, G., and Randall, G. Morphing active contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 7 (2000), 733–737.

[13] Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)* (2016), IEEE, pp. 3464–3468.

[14] Bishop, C. M., et al. *Neural networks for pattern recognition*. Oxford university press, 1995.

[15] Black, M. J., and Jepson, A. D. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision 26*, 1 (1998), 63–84.

[16] Bolme, D. S., Beveridge, J. R., Draper, B. A., and Lui, Y. M. Visual object tracking using adaptive correlation filters. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2010), IEEE, pp. 2544–2550.

[17] Boureau, Y.-L., Ponce, J., and LeCun, Y. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)* (2010), pp. 111–118.

[18] Boyd, S., and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

[19] Breiman, L. *Classification and regression trees*. Routledge, 2017.

[20] Broida, T. J., and Chellappa, R. Estimation of object motion parameters from noisy images. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 1 (1986), 90–99.

[21] Brown, M., and Lowe, D. G. Invariant features from interest point groups. In *BMVC* (2002), vol. 4.

[22] Chapelle, O. Training a support vector machine in the primal. *Neural computation 19*, 5 (2007), 1155–1178.

[23] Cheng, Y. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence 17*, 8 (1995), 790–799.

[24] Ciresan, D. C., Meier, U., Masci, J., Gambardella, L. M., and Schmidhuber, J. Flexible, high performance convolutional neural networks for image classification. In *Twenty-Second International Joint Conference on Artificial Intelligence* (2011).

[25] Collobert, R., and Weston, J. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning* (2008), ACM, pp. 160–167.

[26] Comaniciu, D., Ramesh, V., and Meer, P. Real-time tracking of non-rigid objects using mean shift. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)* (2000), vol. 2, IEEE, pp. 142–149.

[27] Comaniciu, D., Ramesh, V., and Meer, P. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 5 (2003), 564–575.

[28] Cortes, C., and Vapnik, V. Support-vector networks. *Machine learning 20*, 3 (1995), 273–297.

[29] Crammer, K., Keshet, J., and Singer, Y. Kernel design using boosting. In *Advances in neural information processing systems* (2003), pp. 553–560.

[30] Cristianini, N., Shawe-Taylor, J., Elisseeff, A., and Kandola, J. S. On kernel-target alignment. In *Advances in neural information processing systems* (2002), pp. 367–373.

[31] Dalal, N., and Triggs, B. Histograms of oriented gradients for human detection.

[32] Del Moral, P. Non-linear filtering: interacting particle resolution. *Markov processes and related fields 2*, 4 (1996), 555–581.

[33] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09* (2009).

[34] Elgammal, A., Duraiswami, R., Harwood, D., and Davis, L. S. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. *Proceedings of the IEEE 90*, 7 (2002), 1151–1163.

[35] Everingham, M., Gool, L., Williams, C., and Zisserman, A. Pascal visual object classes challenge results. *Available from www. pascal-network. org 1*, 6 (2005), 7.

[36] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. The pascal visual object classes (voc) challenge. *International journal of computer vision 88*, 2 (2010), 303–338.

[37] Fieguth, P., and Terzopoulos, D. Color-based tracking of heads and other mobile objects at video frame rates. In *Proceedings of IEEE computer society conference on computer vision and pattern recognition* (1997), IEEE, pp. 21–27.

[38] Figueroa, P. J., Leite, N. J., and Barros, R. M. Tracking soccer players aiming their kinematical motion analysis. *Computer Vision and Image Understanding 101*, 2 (2006), 122–135.

[39] Freund, Y., and Schapire, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences 55*, 1 (1997), 119–139.

[40] Gardner, M. W., and Dorling, S. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment 32*, 14-15 (1998), 2627–2636.

[41] Girshick, R. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 1440–1448.

[42] Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2014), pp. 580–587.

[43] Glorot, X., Bordes, A., and Bengio, Y. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (2011), pp. 315–323.

[44] Godinez, W. J., Lampe, M., Wörz, S., Müller, B., Eils, R., and Rohr, K. Deterministic and probabilistic approaches for tracking virus particles in time-lapse fluorescence microscopy image sequences. *Medical image analysis 13*, 2 (2009), 325–342.

[45] Gong, L., Mo, Z., Zhao, S., and Song, Y. An improved kernelized correlation filter tracking algorithm based on multi-channel memory model. *Signal Processing: Image Communication* (2019).

[46] Grabner, H., Grabner, M., and Bischof, H. Real-time tracking via on-line boosting. In *Bmvc* (2006), vol. 1, p. 6.

[47] Ha, S.-W., and Moon, Y.-H. Multiple object tracking using sift features and location matching. *International Journal of Smart Home 5*, 4 (2011), 17–26.

[48] Hahnloser, R. H., Sarpeshkar, R., Mahowald, M. A., Douglas, R. J., and Seung, H. S. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature 405*, 6789 (2000), 947.

[49] Hamer, H., Schindler, K., Koller-Meier, E., and Van Gool, L. Tracking a hand manipulating an object. In *2009 IEEE 12th International Conference on Computer Vision* (2009), IEEE, pp. 1475–1482.

[50] Hariharan, B., Arbeláez, P., Girshick, R., and Malik, J. Simultaneous detection and segmentation. In *European Conference on Computer Vision* (2014), Springer, pp. 297–312.

[51] He, K., Zhang, X., Ren, S., and Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence 37*, 9 (2015), 1904–1916.

[52] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[53] Henriques, J. F., Caseiro, R., Martins, P., and Batista, J. Exploiting the circulant structure of tracking-by-detection with kernels. In *European conference on computer vision* (2012), Springer, pp. 702–715.

[54] Henriques, J. F., Caseiro, R., Martins, P., and Batista, J. High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence 37*, 3 (2014), 583–596.

[55] Hino, H., Reyhani, N., and Murata, N. Multiple kernel learning by conditional entropy minimization. In *2010 Ninth International Conference on Machine Learning and Applications* (2010), IEEE, pp. 223–228.

[56] Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Kingsbury, B., et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine 29* (2012).

[57] Hinton, G. E., and Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *science 313*, 5786 (2006), 504–507.

[58] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

[59] Hoi, S. C., Jin, R., Zhao, P., and Yang, T. Online multiple kernel classification. *Machine Learning 90*, 2 (2013), 289–316.

[60] Horn, B. K., and Schunck, B. G. Determining optical flow. *Artificial intelligence 17*, 1-3 (1981), 185–203.

[61] Hou, L., Wan, W., Hwang, J.-N., Muhammad, R., Yang, M., and Han, K. Human tracking over camera networks: a review. *EURASIP Journal on Advances in Signal Processing 2017*, 1 (2017), 43.

[62] Huttenlocher, D. P., Noh, J. J., and Rucklidge, W. J. Tracking non-rigid objects in complex scenes. In *1993 (4th) International Conference on Computer Vision* (1993), IEEE, pp. 93–101.

[63] Ioffe, S., and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[64] Isard, M., and Blake, A. Condensation—conditional density propagation for visual tracking. *International journal of computer vision 29*, 1 (1998), 5–28.

[65] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia* (2014), ACM, pp. 675–678.

[66] Joachims, T. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (2006), ACM, pp. 217–226.

[67] Juan, L., and Gwon, L. A comparison of sift, pca-sift and surf. *International Journal of Signal Processing, Image Processing and Pattern Recognition 8*, 3 (2007), 169–176.

[68] Kalal, Z., Mikolajczyk, K., and Matas, J. Forward-backward error: Automatic detection of tracking failures. In *2010 20th International Conference on Pattern Recognition* (2010), IEEE, pp. 2756–2759.

[69] Kalal, Z., Mikolajczyk, K., and Matas, J. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence 34*, 7 (2011), 1409–1422.

[70] Kalman, R. E. A new approach to linear filtering and prediction problems. *Journal of basic Engineering 82*, 1 (1960), 35–45.

[71] Kalman, R. E., and Bucy, R. S. New results in linear filtering and prediction theory. *Journal of basic engineering 83*, 1 (1961), 95–108.

[72] Kang, H.-B., and Cho, S.-H. Short-term memory-based object tracking. In *International Conference Image Analysis and Recognition* (2004), Springer, pp. 597–605.

[73] Kang, J., Cohen, I., and Medioni, G. Object reacquisition using invariant appearance model. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* (2004), vol. 4, IEEE, pp. 759–762.

[74] Khan, G., Tariq, Z., and Khan, M. U. G. Multi-person tracking based on faster r-cnn and deep appearance features. In *Visual Object Tracking in the Deep Neural Networks Era*. IntechOpen, 2019.

[75] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (2012), pp. 1097–1105.

[76] Kuhn, H. W. The hungarian method for the assignment problem. *Naval research logistics quarterly 2*, 1-2 (1955), 83–97.

[77] Kwon, J., and Lee, K. M. Visual tracking decomposition. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2010), IEEE, pp. 1269–1276.

[78] Lampert, C. H. *Kernel Methods in Computer Vision*. Now Publishers Inc., Hanover, MA, USA, 2009.

[79] Lampert, C. H., and Blaschko, M. B. A multiple kernel learning approach to joint multi-class object detection. In *Joint Pattern Recognition Symposium* (2008), Springer, pp. 31–40.

[80] Lanckriet, G. R., Cristianini, N., Bartlett, P., Ghaoui, L. E., and Jordan, M. I. Learning the kernel matrix with semidefinite programming. *Journal of Machine learning research 5*, Jan (2004), 27–72.

[81] Lanckriet, G. R., De Bie, T., Cristianini, N., Jordan, M. I., and Noble, W. S. A statistical framework for genomic data fusion. *Bioinformatics 20*, 16 (2004), 2626–2635.

[82] Leal-Taixé, L., Milan, A., Reid, I., Roth, S., and Schindler, K. MOTChallenge 2015: Towards a benchmark for multi-target tracking. *arXiv:1504.01942 [cs]* (Apr. 2015). arXiv: 1504.01942.

[83] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, 11 (1998), 2278–2324.

[84] Lee, Y.-H., Ahn, H., Cho, H.-J., and Lee, J.-H. Object recognition and tracking based on object feature extracting. *J. Internet Serv. Inf. Secur. 5*, 3 (2015), 48–57.

[85] Lee, Y.-H., Park, J.-H., and Kim, Y. Comparative analysis of the performance of sift and surf. *Journal of the Semiconductor & Display Technology 12*, 3 (2013), 59–64.

[86] Li, X., Hu, W., Shen, C., Zhang, Z., Dick, A., and Hengel, A. V. D. A survey of appearance models in visual object tracking. *ACM transactions on Intelligent Systems and Technology (TIST) 4*, 4 (2013), 58.

[87] Lin, M., Chen, Q., and Yan, S. Network in network. *arXiv preprint arXiv:1312.4400* (2013).

[88] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In *European conference on computer vision* (2014), Springer, pp. 740–755.

[89] Liu, J., Tong, X., Li, W., Wang, T., Zhang, Y., and Wang, H. Automatic player detection, labeling and tracking in broadcast soccer video. *Pattern Recognition Letters 30*, 2 (2009), 103–113.

[90] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. Ssd: Single shot multibox detector. In *European conference on computer vision* (2016), Springer, pp. 21–37.

[91] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International journal of computer vision 60*, 2 (2004), 91–110.

[92] Lowe, D. G., et al. Object recognition from local scale-invariant features. In *iccv* (1999), vol. 99, pp. 1150–1157.

[93] Lucas, B. D., Kanade, T., et al. An iterative image registration technique with an application to stereo vision.

[94] Matov, A., Applegate, K., Kumar, P., Thoma, C., Krek, W., Danuser, G., and Wittmann, T. Analysis of microtubule dynamic instability using a plus-end growth marker. *Nature methods 7*, 9 (2010), 761.

[95] Melax, S., Keselman, L., and Orsten, S. Dynamics based 3d skeletal hand tracking. In *Proceedings of Graphics Interface 2013* (2013), Canadian Information Processing Society, pp. 63–70.

[96] Mercer, J. Xvi. functions of positive and negative type, and their connection the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character 209*, 441-458 (1909), 415–446.

[97] Miclut, B. Committees of deep feedforward networks trained with few data. In *German Conference on Pattern Recognition* (2014), Springer, pp. 736–742.

[98] Moghaddam, B., and Pentland, A. Probabilistic visual learning for object representation. *IEEE Transactions on pattern analysis and machine intelligence 19*, 7 (1997), 696–710.

[99] Mohan, A., Papageorgiou, C., and Poggio, T. Example-based object detection in images by components. *IEEE transactions on pattern analysis and machine intelligence 23*, 4 (2001), 349–361.

[100] Nam, H., and Han, B. Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 4293–4302.

[101] Nixon, M., and Aguado, A. S. *Feature extraction and image processing for computer vision*. Academic Press, 2012.

[102] Ouyang, W., Luo, P., Zeng, X., Qiu, S., Tian, Y., Li, H., Yang, S., Wang, Z., Xiong, Y., Qian, C., et al. Deepid-net: multi-stage and deformable deep convolutional neural networks for object detection. *arXiv preprint arXiv:1409.3505* (2014).

[103] Paragios, N., and Deriche, R. Geodesic active regions and level set methods for supervised texture segmentation. *International Journal of Computer Vision 46*, 3 (2002), 223–247.

[104] Patel, H. A., and Thakore, D. G. Moving object tracking using kalman filter. *International Journal of Computer Science and Mobile Computing 2*, 4 (2013), 326–332.

[105] Pitts, W., and McCulloch, W. S. How we know universals the perception of auditory and visual forms. *The Bulletin of mathematical biophysics 9*, 3 (1947), 127–147.

[106] Platt, J. C. Using analytic qp and sparseness to speed training of support vector machines. In *Advances in neural information processing systems* (1999), pp. 557–563.

[107] Poormehdi Ghaemmaghami, M. Tracking of humans in video stream using lstm recurrent neural network, 2017.

[108] Qi, Y.-J., and Wang, Y.-J. Robust object tracking algorithm by particle filter based on human memory model. *Pattern Recognit. Artif. Intell 25* (2012), 810–816.

[109] Quinlan, J. R. Induction of decision trees. *Machine learning 1*, 1 (1986), 81–106.

[110] Rakotomamonjy, A., Bach, F. R., Canu, S., and Grandvalet, Y. Simplemkl. *Journal of Machine Learning Research 9*, Nov (2008), 2491–2521.

[111] Redmon, J. Darknet: Open source neural networks in c. `http://pjreddie.com/darknet/`, 2013–2016.

[112] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 779–788.

[113] Redmon, J., and Farhadi, A. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 7263–7271.

[114] Redmon, J., and Farhadi, A. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).

[115] Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (2015), pp. 91–99.

[116] Rifkin, R., Yeo, G., Poggio, T., et al. Regularized least-squares classification. *Nato Science Series Sub Series III Computer and Systems Sciences 190* (2003), 131–154.

[117] Ronfard, R. Region-based strategies for active contour models. *International journal of computer vision 13*, 2 (1994), 229–251.

[118] Ross, D. A., Lim, J., Lin, R.-S., and Yang, M.-H. Incremental learning for robust visual tracking. *International journal of computer vision 77*, 1-3 (2008), 125–141.

[119] Ross, D. A., Lim, J., Lin, R.-S., and Yang, M.-H. Incremental learning for robust visual tracking. *International journal of computer vision 77*, 1-3 (2008), 125–141.

[120] Rowley, H. A., Baluja, S., and Kanade, T. Neural network-based face detection. *IEEE Transactions on pattern analysis and machine intelligence 20*, 1 (1998), 23–38.

[121] Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. Learning representations by back-propagating errors. *Cognitive modeling 5*, 3 (1988), 1.

[122] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International journal of computer vision 115*, 3 (2015), 211–252.

[123] Salari, V., and Sethi, I. K. Feature point correspondence in the presence of occlusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence 12*, 1 (1990), 87–91.

[124] Sato, K., and Aggarwal, J. K. Temporal spatio-velocity transform and its application to tracking and interaction. *Computer Vision and Image Understanding 96*, 2 (2004), 100–128.

[125] Scherer, D., Müller, A., and Behnke, S. Evaluation of pooling operations in convolutional architectures for object recognition. In *International conference on artificial neural networks* (2010), Springer, pp. 92–101.

[126] Scholkopf, B., and Smola, A. J. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[127] Serby, D., Meier, E., and Van Gool, L. Probabilistic object tracking using multiple features. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* (2004), vol. 2, IEEE, pp. 184–187.

[128] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229* (2013).

[129] Shalev-Shwartz, S., Singer, Y., Srebro, N., and Cotter, A. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming 127*, 1 (2011), 3–30.

[130] Shi, J., et al. Good features to track. In *1994 Proceedings of IEEE conference on computer vision and pattern recognition* (1994), IEEE, pp. 593–600.

[131] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[132] Sonnenburg, S., Rätsch, G., and Schäfer, C. A general and efficient multiple kernel learning algorithm. In *Advances in neural information processing systems* (2006), pp. 1273–1280.

[133] Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806* (2014).

[134] Sridhar, S., Mueller, F., Oulasvirta, A., and Theobalt, C. Fast and robust hand tracking using detection-guided optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 3213–3221.

[135] Sridhar, S., Mueller, F., Zollhöfer, M., Casas, D., Oulasvirta, A., and Theobalt, C. Real-time joint tracking of a hand manipulating an object from rgb-d input. In *European Conference on Computer Vision* (2016), Springer, pp. 294–310.

[136] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research 15*, 1 (2014), 1929–1958.

[137] Stommel, M., and Herzog, O. Binarising sift-descriptors to reduce the curse of dimensionality in histogram-based object recognition. In *International Conference on Signal Processing, Image Processing, and Pattern Recognition* (2009), Springer, pp. 320–327.

[138] Straw, A. D., Branson, K., Neumann, T. R., and Dickinson, M. H. Multi-camera real-time three-dimensional tracking of multiple flying animals. *Journal of The Royal Society Interface 8*, 56 (2010), 395–409.

[139] Streit, R. L., and Luginbuhl, T. E. Maximum likelihood method for probabilistic multihypothesis tracking. In *Signal and Data Processing of Small Targets 1994* (1994), vol. 2235, International Society for Optics and Photonics, pp. 394–405.

[140] Sun, Y., Wang, X., and Tang, X. Deep convolutional network cascade for facial point detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2013), pp. 3476–3483.

[141] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 1–9.

[142] Szegedy, C., Reed, S., Erhan, D., Anguelov, D., and Ioffe, S. Scalable, high-quality object detection. *arXiv preprint arXiv:1412.1441* (2014).

[143] Tang, M., and Feng, J. Multi-kernel correlation filter for visual tracking. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 3038–3046.

[144] Tao, H., Sawhney, H. S., and Kumar, R. Object tracking with bayesian estimation of dynamic layer representations. *IEEE transactions on pattern analysis and machine intelligence 24*, 1 (2002), 75–89.

[145] Uijlings, J. R., Van De Sande, K. E., Gevers, T., and Smeulders, A. W. Selective search for object recognition. *International journal of computer vision 104*, 2 (2013), 154–171.

[146] Van den Oord, A., Dieleman, S., and Schrauwen, B. Deep content-based music recommendation. In *Advances in neural information processing systems* (2013), pp. 2643–2651.

[147] Varma, M., and Ray, D. Learning the discriminative power-invariance trade-off. In *2007 IEEE 11th International Conference on Computer Vision* (2007), IEEE, pp. 1–8.

[148] Veenman, C. J., Reinders, M. J., and Backer, E. Resolving motion correspondence for densely moving points. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23*, 1 (2001), 54–72.

[149] Viola, P., Jones, M., et al. Rapid object detection using a boosted cascade of simple features. *CVPR (1) 1*, 511-518 (2001), 3.

[150] Viola, P., and Jones, M. J. Robust real-time face detection. *International journal of computer vision 57*, 2 (2004), 137–154.

[151] Wang, X., Zhang, L., Lin, L., Liang, Z., and Zuo, W. Deep joint task learning for generic object extraction. In *Advances in neural information processing systems* (2014), pp. 523–531.

[152] Weng, S.-K., Kuo, C.-M., and Tu, S.-K. Video object tracking using adaptive kalman filter. *Journal of Visual Communication and Image Representation 17*, 6 (2006), 1190–1208.

[153] Weston, J., Ratle, F., Mobahi, H., and Collobert, R. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 639–655.

[154] Wu, Y., Lim, J., and Yang, M.-H. Online object tracking: A benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2013).

[155] Yang, T., Mahdavi, M., Jin, R., Yi, J., and Hoi, S. C. Online kernel selection: Algorithms and evaluations. In *Twenty-Sixth AAAI Conference on Artificial Intelligence* (2012).

[156] Yilmaz, A., Javed, O., and Shah, M. Object tracking: A survey. *Acm computing surveys (CSUR) 38*, 4 (2006), 13.

[157] Yilmaz, A., Li, X., and Shah, M. Contour-based object tracking with occlusion handling in video acquired using mobile cameras. *IEEE Transactions on pattern analysis and machine intelligence 26*, 11 (2004), 1531–1536.

[158] Young, N. *An introduction to Hilbert space*. Cambridge university press, 1988.

[159] Zeiler, M. D. *Hierarchical convolutional deep learning in computer vision*. PhD thesis, New York University, 2013.

[160] Zeiler, M. D., and Fergus, R. Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557* (2013).

[161] Zeiler, M. D., and Fergus, R. Visualizing and understanding convolutional networks. In *European conference on computer vision* (2014), Springer, pp. 818–833.

[162] Zeng, X., Ouyang, W., and Wang, X. Multi-stage contextual deep learning for pedestrian detection. In *Proceedings of the IEEE International Conference on Computer Vision* (2013), pp. 121–128.

[163] Zhang, Y., Sohn, K., Villegas, R., Pan, G., and Lee, H. Improving object detection with deep convolutional networks via bayesian optimization and structured prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 249–258.

[164] Zhu, S. C., and Yuille, A. Region competition: Unifying snakes, region growing, and bayes/mdl for multiband image segmentation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 9 (1996), 884–900.

واژه‌نامه انگلیسی به فارسی

هم‌بستگی انطباقی ..... Adaptive Correlation
کادر محوری ..... Anchor Box
اشیاء متحرک ..... Animated Objects
مدل ظاهری ..... Appearance Model
شلوغی پس‌زمینه ..... Background Clutter
روشنایی ..... Brightness
ماتریس چرخشی (مدوّر) ..... Circulant Matrix
ساختار چرخشی (مدوّر) ..... Circulant Structure
دسته‌بندی ..... Classification
دسته‌بندی کننده ..... Classifier
خوشه‌بندی ..... Clustering
بینایی کامپیوتر ..... Computer Vision
کانتور ..... Contour
تلفیقی لایه ..... Convolution Layer
شبکه‌های صبی تلفیقی ..... Convolutional Neural Networks
صحت ..... Correctness
ارزیابی متقابل ..... Cross-Validation
یادگیری عمیق ..... Deep Learning
نمونه‌گیری متراکم ..... Dense Sampling
متمایز کننده ..... Discriminative
فاصله اقلیدسی ..... Euclidean Distance
استخراج ویژگی ..... feature extraction
نگاشت ویژگی ..... Feature Map
انتخاب ویژگی ..... feature selection
تبدیلات فوریه ..... Fourier Transforms
قاب ..... frame
مولد ..... Generative
حاشیه هندسی ..... Geometric Margin
جعبه محاطی حقیقی ..... Ground Truth
ویژگی‌های سطح بالا ..... High-level Features
فضای هیلبرت ..... Hilbert Space
هیستوگرام گرادیان ..... Histogram of Gradient

ابر پارامتر ..... Hyperparameter
دسته‌بندی تصاویر ..... Image Classification
پردازش تصویر ..... Image Processing
فیلتر کالمن ..... Kalman Filter
هسته ..... Kernel
فیلتر هم‌بستگی ..... Correlation Filter
رگرسیون لجستیک ..... Logistic Regression
ویژگی‌های سطح پایین ..... Low-level Features
یادگیری ماشین ..... Machine Learning
شبکه چند دامنه‌ای ..... Multi-Domain Network
چند کاناله ..... Multi-channel
یادگیری چند هسته‌ای ..... Multi-kernel Learning
شبکه‌های عصبی ..... Neural Networks
نوفه ..... Noise
نرمال‌سازی ..... Normalization
شناسایی اشیاء ..... Object Detection
نمایش اشیاء ..... Object Representation
ردیابی اشیاء ..... Object Tracking
انسداد ..... Occlusion
برون‌خط ..... Offline
مقادیر تنظیم (انحراف) ..... Offsets Value
برخط ..... Online
فیلتر جزئی ..... Partial Filter
آنالیز الگو ..... Pattern Analysis
معین مثبت ..... Positive Definition
دقت ..... Precision
پیش پردازش ..... preprocessing
تجزیه و تحلیل مولفه‌های اساسی ..... Principal Component Analysis
صحت ..... Recall
ناحیه ادراکی ..... Receptive Field
ناحیه نامزد ..... Region Proposal
رگرسیون ..... Regression
قضیه بیان کننده ..... Representer Theorem
رگرسیون ریج ..... Ridge Regression

پایداری ..... Robustness
سیاه‌نمایی ..... Silhouette
تشابه ..... Similarity
پنجره لغزان ..... Sliding Window
دسته‌بندی نرم ..... Soft Classification
تشخیص گفتار ..... Speech Recognition
گام ..... Stride
بردار ماشین پشتیبان ..... Support Vector Machine
تطابق قالب ..... Template Matching

Abstract

In this thesis, an introduction to the field of computer vision and its applications as a subfield of machine learning is discussed first. Then two instances of important and applicable issues in this field, Object detection and tracking and a history of their corresponding methods and algorithms are presented. In the following, kernel-based methods for solving machine learning problems and especially non-linear classifications are presented and then a generalized state of these methods is introduced as multi-kernel learning (MKL). after that, convolutional networks, types of their layers and architectures as well as their different applications are described. In addition, some examples of algorithms based on these networks for object detection and tracking problems are introduced and compared with other algorithms. Finally, using the multi-kernel correlation filter (MKCF) algorithm and applying the unique properties of these methods as well as utilizing a multi-channel memory model, a proposed algorithm for object tracking is presented which in addition to very low computational complexity, it has good accuracy in conditions such as occlusion, object deformation, and background clutter. Also, it can be used in real-time object tracking.

Yazd University

Faculty of Mathematics

Department of Computer Science

Title:

Detection and tracking of animated objects using multi-kernel learning algorithms

Supervisor:

Prof. Mohammad Reza Hooshmandasl

Advisor:

Dr. Elham Abbasi

By:

Amir Mohammadi

January 2020