به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین سوم

نام و نام خانوادگی	سارا رستمی – محمدامین شاهچراغی
شماره دانشجویی	۸۱۰۱۰۰۳۵۵ - ۸۱۰۱۹۹۱۹۶
تاریخ ارسال گزارش	۱۴۰۱،۰۹،۲۰

فهرست

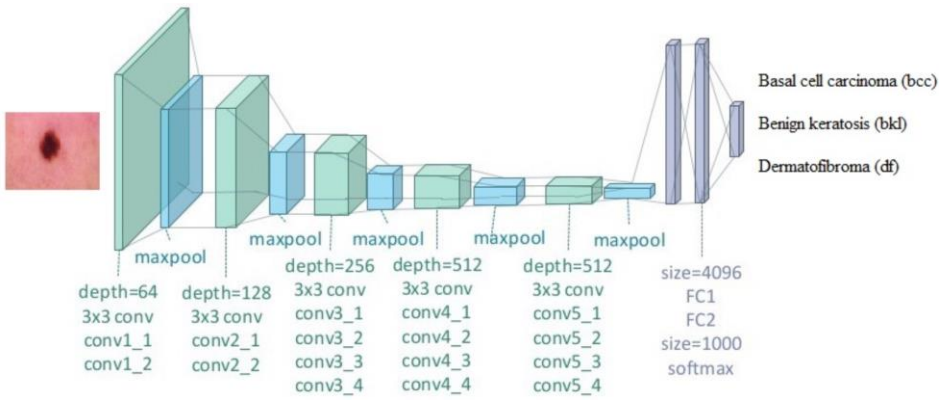
شکل‌ها

جدول‌ها

پاسخ ۱. آشنایی با یادگیری انتقالی Transfer Learning

۱. گزارش مقاله

مقاله انتخابی " مدل classification سرطان پوست بر اساس VGG19 و یادگیری انتقالی " می‌باشد. با استفاده از روش‌های پردازش تصویر، می‌توان ویژگی‌هایی برای تشخیص سرطان استخراج کرد. CNNها قابلیت استخراج ویژگی به طور خودکار دارند و می‌توانند به دقت بالایی در تشخیص سرطان پوست برسند. در این مقاله، داده‌های مربوط به دو نوع سرطان و یک کلاس بدون سرطان که از دیتاست "HAM10000" گرفته‌شده، طبقه‌بندی می‌شوند. این طبقه‌بندی با استفاده یک مدل CNN بر اساس تکنیک یادگیری انتقالی و VGG19 انجام می‌گیرد. معرفی دیتاست و پیش‌پردازش: دو نوع سرطان (DF و BCC) و یک نوع بدون سرطان (BKL) از دیتاست. به علت وجود فراوانی بیشتر کلاس BKL نسبت به دو کلاس دیگر، در دیتاست عدم توازن وجود دارد. عدم توازن در دیتاست یک نوع دیتا بایاس است که می‌تواند منجر به overfitting مدل شود. از این رو، برای افزایش کلاس‌های DF و BCC از augmentation به کار رفته است. تکنیک‌های augmentation شامل crop، scale، contrast، تنظیم روشنایی، horizontal flip، vertical flip و ترکیبی از این روش‌ها می‌شود. بعد از augmentation، هر یک از کلاس‌ها ۱۰۰۰ نمونه خواهند داشت. VGG19 یک CNN عمیق می‌باشد که چندین لایه‌ی کانوولوشنی و max pooling که در واقع feature extractor می‌باشند، را شامل می‌شود. بعد این لایه‌ها حداقل یک لایه‌ی fully connected قرار می‌گیرد که در واقع classifier می‌باشد. اندازه و تعداد لایه‌های کانوولوشن و fully connected را به طبق تصمیم طراح CNN می‌توان تعیین کرد. معماری کلی VGG19 را در شکل ۱ مشاهده می‌کنید.



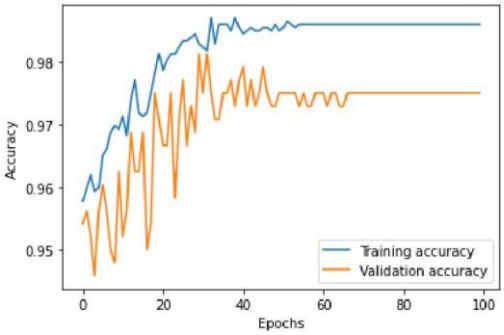
شکل ۱- معماری کلی **VGG19**

سایز لایه‌ی ورودی ۶۴x۶۴ می‌باشد و لایه‌ی خروجی با یک softmax function یکی از سه کلاس را نشان می‌دهد. به عبارتی با استفاده از یک مدل از پیش آموزش یافته‌ی VGG19 با پارامترهای

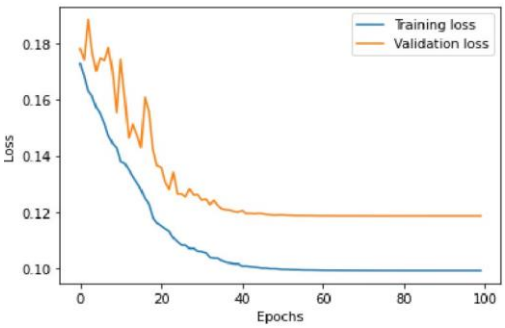
fine-tuned شده مدلی برای تشخیص سرطان پوست utilize شده‌است. ۸۰ درصد دیتاست به عنوان داده‌ی train (و ۲۰ درصد از این داده‌ی train به عنوان داده‌ی validation استفاده شد) و ۲۰ درصد دیتاست به عنوان داده‌ی test در نظر گرفته شد. مدل طی ۱۰۰ ایپاک با batch size برابر با ۵۰ آموزش یافت. Optimization function مورد استفاده برای آموزش این شبکه Adam بود. بعد از ۱۰۰ ایپاک، پارامترهای بهترین مدل انتخاب شدند و برای داده‌های تست به کار برده شدند. بعد از آموزش شبکه، شبکه با ۶۰۰ داده‌ی تست مورد آزمون قرار گرفت و عملکرد آن با استفاده از accuracy و loss کلی مورد بررسی قرار گرفت. نتایج نهایی آموزش و تست این مدل را در جدول ۱ و شکل‌های ۲ و ۳ مشاهده می‌کنید.

جدول ۱- خلاصه‌ی نتایج مدل‌ها

Epoch	Training		Validation	
	Accuracy	Loss	Accuracy	Loss
25	0.9823	0.1094	0.9708	0.1264
50	0.9849	0.0997	0.9750	0.1188
100	0.9859	0.0991	0.9750	0.1185

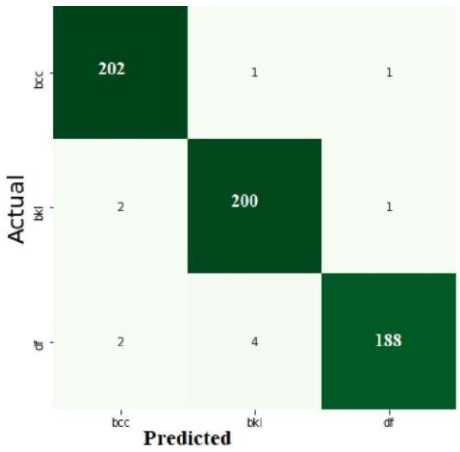


شکل ۲- **accuracy** داده‌های آموزش و **validation** در هر ایپاک



شکل ۳- میزان **loss** داده‌های آموزش و **validation** درهر ایپاک

جدول ۱ نتیجه‌ی test و train مدل را جمع‌بندی می‌کند. از روی جدول ۱ می‌توان دید که اختلاف زیادی بین نتیجه‌ی آموزش و تست وجود ندارد پس مدل overfit نمی‌کند. با توجه به شکل ۲ و ۳ می‌توان دید، میزان ارتعاشات نمودارهای loss و accuracy بین ایپاک ۶۰ و ۷۰ متوقف می‌یابد که به معنی stable شدن شبکه می‌باشد. برای بررسی بیشتر دقت شبکه در شکل ۴ ماتریس آشفتگی (confusion matrix) مدل را مشاهده می‌کنید. می‌توان دید که اکثر پیش‌بینی‌ها درست انجام شده و تعداد کمی از نمونه‌ها در کلاس اشتباه دسته‌بندی شدند.



شکل ۴ - ماتریس آشفتگی **VGG19**

نشان داده شد که VGG19-based CNN و یادگیری انتقالی ابزارهای قوی‌ای برای تشخیص سرطان با دقت بالا هستند. Accuracy و loss کلی شبکه رضایت‌بخش بوده و امید به بهبود آن وجود دارد.

۲. معماری شبکه و مزایا و معایب آن و پیش‌پردازش

پیش‌پردازش: سه کلاس BKL و BCC و DF را از دیتاست گرفتیم. به دلیلی نامتوزارن بودن تعداد دسته‌ها، از data augmentation استفاده کردیم تا تعداد داده‌های دو کلاس BCC و DF را افزایش دهیم که با تعداد داده‌های کلاس BKL تقریبا برابر شوند. سپس داده‌ها را به دو مجموعه test و train تقسیم کردیم.

معماری شبکه: از مدل VGG19 استفاده کردیم. VGG19 یک شبکه‌ی pre-trained می‌باشد که از آن برای آموزش شبکه خود به روش یادگیری انتقالی استفاده کردیم. از آنجایی که در مقاله ذکر شده که تصاویر دیتاست به اندازه‌ی 64x64 resize شده اند، و ورودی شبکه VGG19 تصاویر 224x224 می‌باشند، و تعداد کلاس‌های ما ۳ تا می‌باشد درحالیکه VGG19 برای ۱۰۰۰ کلاس طراحی شده‌است،

ما نمی‌توانیم از بخش fully-connected این شبکه استفاده کنیم. بنابراین نیاز بود که این بخش fully-connected شبکه را خودمان طراحی کنیم. شبکه fully-connected ما شامل دو لایه ۴۰۹۶ نودی با dropout = 0.5 می‌باشد که در نهایت با تابع softmax به ۳ کلاس تقسیم می‌شود.

۳. قابلیت تشخیص چه نوع عکس‌هایی

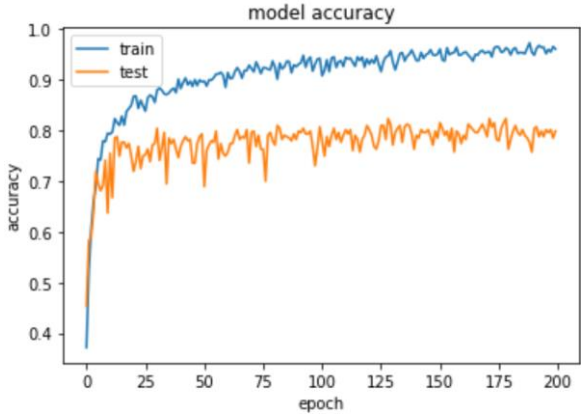
قابلیت تشخیص تصاویر مربوط به دو کلاس سرطانی DF و BCC و یک کلاس تصاویر بدون سرطان BKL را دارد. اگر عکسی در دسته‌های ذکر شده نباشد، شبکه‌ی ما آن تصویر را در دسته‌ای از ۳ کلاس بالا طبقه‌بندی می‌کند که بیشترین شباهت را به آن دسته دارد. برای حل این مشکل می‌توان دسته‌های دیگر سرطان پوست را هم به شبکه آموزش داد تا قابلیت دسته‌بندی آنها را هم داشته باشد.

۴. لود دیتاست

دیتاست را سایت Kaggle گرفتیم و در گوگل درایو خود ریختیم و با استفاده از کتابخانه‌ی کتابخانه‌ی PIL تصاویر دیتاست را لود کردیم (از تابع Image.open() استفاده کردیم).

۵. گزارش عملکرد شبکه روی داده‌های تست

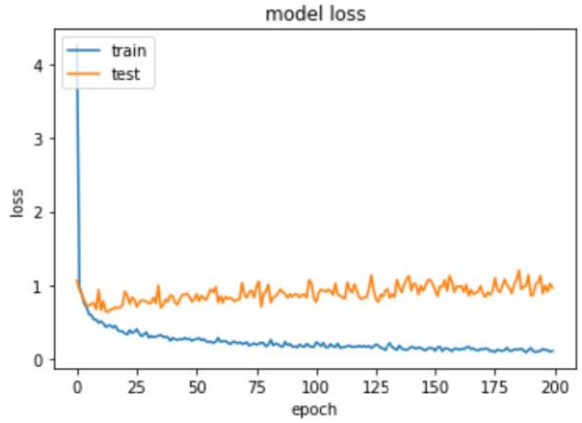
در شکل ۵ نمودار accuracy شبکه را روی داده‌های test و train می‌بینید. می‌بینیم که accuracy داده‌های train با گذشت ایپاک‌ها به طور صعودی (با ارتعاشات زیاد) در حال تغییر است. این در حالیست که accuracy شبکه روی داده‌های تست، از ایپاک ۵۰ به بعد حول و حوش ۰,۷۵-۰,۷۸ می‌ماند.



شکل ۵- نمودار accuracy مدل روی داده‌های train و test

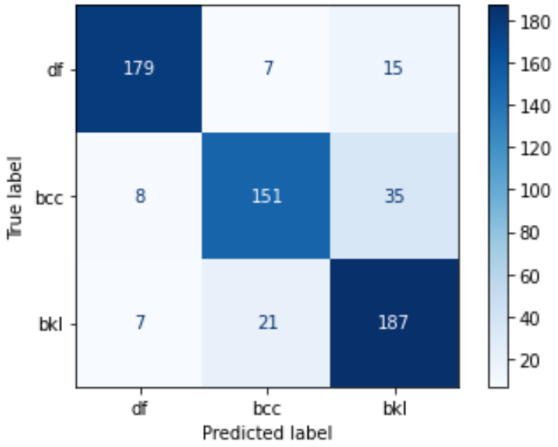
در شکل ۶ نمودار loss شبکه را روی داده‌های test و train مشاهده می‌کنید. میزان loss مدل روی داده‌های train همواره در حال کاهش می‌باشد و در ایپاک ۲۰۰م به حدود ۰ می‌رسد در حالیکه روی

داده‌های test روند متفاوتی را طی می‌کند. میزان loss روی داده‌های test ابتدا کمی کاهش می‌یابد و پس از آن با شیب کمی افزایش یافته و در حدود ۱ ارتعاش دارد.



شکل ۶- نمودار loss مدل روی داده‌های train و test

طبق ماتریس طبقه‌بندی مدل که در شکل ۷ مشاهده می‌کنید، مدل بیشترین قدرت را در تشخیص کلاس BKL و پس از آن کلاس DF داشته و پس از این دو کلاس ، کلاس BCC دارد.



شکل ۷ - ماتریس طبقه‌بندی مدل برای داده‌های test

نتیجه‌ی classification report شبکه روی داده‌های تست را در شکل ۸ مشاهده می‌کنید. مدل ما موفق به دستیابی f1_score برابر با 0.85، precision برابر با 0.85، accuracy برابر با 0.85 و recall برابر با 0.85 شد.

CL Report:

	precision	recall	f1-score	support
0	0.92	0.89	0.91	201
1	0.84	0.78	0.81	194
2	0.79	0.87	0.83	215
accuracy			0.85	610
macro avg	0.85	0.85	0.85	610
weighted avg	0.85	0.85	0.85	610

شکل ۸ - گزارش classification داده‌های تست توسط مدل

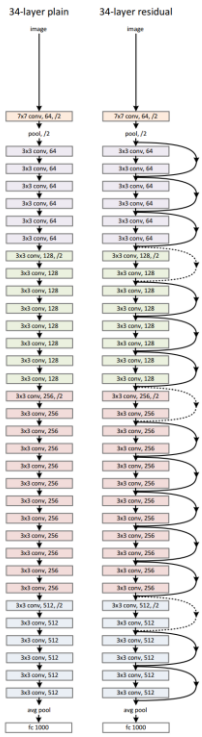
پاسخ ۲ – آشنایی با تشخیص چهره مسدود شده

۱. خلاصه‌ی ساختار شبکه

برای بررسی کارآمدی دیتاست‌های ارائه شده، آموزش با استفاده از ورژن‌های مختلف دیتاست روی دو CNN، PSPNet و DeepLabv3+ و همچنین یک Vision Transformer به نام SegFormer انجام شده‌است. Backbone دو CNN ذکر شده یک شبکه‌ی عمیق pre-trained به نام ResNet-101 می‌باشد. اکثر مدل‌های semantic segmentation شامل دو بخش هستند: Encoder و Decoder

منظور از backbone همان بخش Encoder یا در واقع feature extractor شبکه ما می‌باشد.

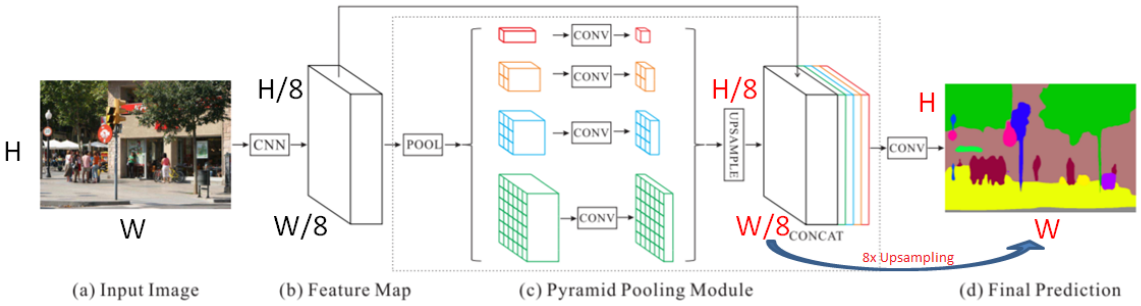
شبکه ResNet به همین منظور براس استخراج ویژگی به عنوان backbone دو شبکه‌ی مذکور استفاده می‌شود. برای این کار ResNet-101 از ۱۰۱تا block پشت سر هم استفاده می‌کند که همگی شامل تعدادی لایه‌ی کانوولوشن هستند. تفاوت Residual Network با یک شبکه ۱۰۱ لایه‌ی ساده در وجود shortcut هایی بین هر دو بلاک می‌باشد. در شکل ۵ معماری یک شبکه‌ی ساده با ۳۴ لایه‌ی پارامتری(سمت چپ) و یک شبکه‌ی Residual با ۳۴ لایه‌ی پارامتری را مشاهده می‌کنید. شبکه‌ی ResNet استفاده شده در این مقاله به عنوان backbone ۱۰۱ لایه دارد.



شکل ۹- شبکه‌ی ساده‌ی با ۳۴ لایه‌ی کانوولوشنی (سمت چپ) و شبکه‌ی residual با ۳۴ لایه‌ی کانوولوشنی (سمت راست)

پس خروجی این شبکه یک feature map می‌باشد.

Feature map تولید شده توسط ResNet-101 به ماژول Pyramid pooling شبکه ی PSPNet داده شده و در سایزهای متفاوت pool می‌شود. مهمترین بخش این شبکه همین ماژول می‌باشد. سپس هر یک از این ویژگی‌های pool شده با سایز مختلف، از یک Conv 1x1 عبور می‌کنند تا کاهش عمق یابند. در نهایت، همه‌ی آنها Upsample شده تا به اندازه‌ی feature map اولیه برسند و به انتهای آن افزوده شده و به decoder داده شوند. بیشترین decoder مورد استفاده برای PSPNet، یک لایه‌ی کانوولوشنی می‌باشد که پس از آن یک 8x upsampling decoder قرار می‌گیرد. معماری کلی PSPNet با دیکودر مذکور را در شکل ۶ مشاهده می‌کنید.

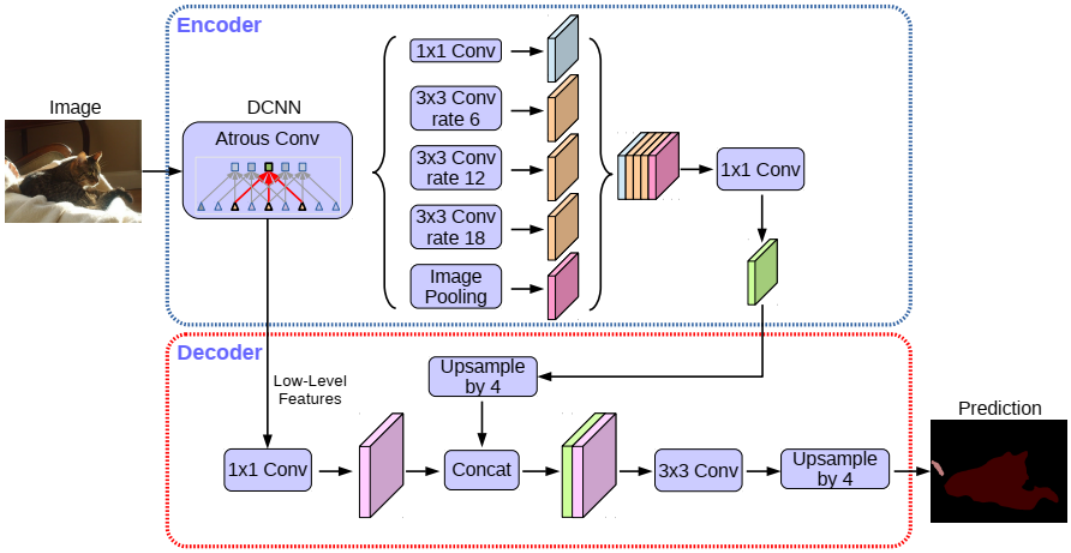


شکل ۱۰ - معماری **PSPNet** با دیکودر **8x upsampling**

معماری شبکه‌ی DeepLabv3+ به شرح زیر است:

- Feature map توسط شبکه‌ی backbone (در اینجا ResNet-101) تهیه می‌شود.
- برای کنترل سایز feature map، astrous convolution در بلاک‌های انتهایی شبکه backbone استفاده می‌شود.
- علاوه بر ویژگی‌های استخراج شده از backbone، یک شبکه‌ی astrous spatial pyramid pooling (ASSP) اضافه می‌شود تا هر پیکسل classify شود.
- خروجی ASSP از یک لایه‌ی کانوولوشنی 1x1 عبور می‌کند تا به اندازه‌ی تصویر اولیه برسد.
- ماژول decoder: ویژگی‌های encode شده ابتدا با فاکتور ۴ upsample می‌شوند تا با ویژگی‌های low-level ماژول انکودر هم بُعد شده و concate شوند. قبل از concatenation، ویژگی‌های low-level از کانوولوشن‌های 1x1 عبور می‌کنند تا تعداد کانال‌های کاهش یابد. بعد از concatenation، تعدادی کانوولوشن 3x3 اعمال شده و ویژگی‌ها با فاکتور ۴، upsample می‌شوند. اینکار باعث می‌شود سایز تصویر ورودی و خروجی برابر شود.

معماری کلی Deeplabv3+ را در شکل ۷ مشاهده می‌کنید.



شکل ۱۱ - معماری کلی **Deeplabv3+**

۲. تفاوت در دقت شبکه با Occlusionهای مختلف

مدل‌هایی که با دیتاست‌های NatOcc آموزش یافتند، هم سطح یا بهتر از دیتاست‌های Real-World occluded face dataset مثل C-CM عمل می‌کنند. طبق نتایج مدل‌ها در جدول ۴ مقاله، مدل‌های CNNی که با دیتاست‌های NatOcc آموزش یافتند، می‌توانند صورت را نسبت به CNNهایی که با دیتاست C-CM آموزش یافتند، بهتر segment کنند. پس افزودن occluderهای مختلف (همانند خوراکی، چنگال، دست، اشیاء مختلف و...) به دیتای train منجر به مدل‌های قوی‌تر با قابلیت generalizability بیشتری می‌شود. با وجود اینکه افزودن occluderهای رندوم (با شفافیت و اشکال مختلف) نیز منجر به بهبود قدرت شناسایی و generalizability مدل‌ها می‌شود اما به طور کلی مدل‌هایی که با ورژنی از دیتاست NatOcc آموزش یافته‌اند منجر به قابلیت generalizability و دقت بالاتری می‌شوند.

همچنین شبکه‌ی SegFormer آموزش یافته توسط داده‌های دیتاست C-WO-NatOcc که فاقد occluderهای شفاف (transparent/translucent) می‌باشد، نتوانست به خوبی شیشه‌های شفاف را به عنوان occluder در دیتای تست تشخیص دهد. حتی مدل آموزش یافته تنها با دیتاست C-WO-RandOcc (که شامل اشکال رندوم با texture و شفافیت رندوم می‌باشد) هم به خوبی نتوانست شیشه‌های شفاف را تشخص دهد. اما مدل آموزش یافته با دیتاست C-WO-Mix که ترکیبی از این دو دیتاست می‌باشد، می‌تواند شیشه‌های شفاف را به عنوان occlusion تشخیص دهد . این نشان می‌دهد

که ترکیبی از occluderهای رندوم (با شفافیت و textureهای مختلف) و اشیاء و دست، مکمل هم برای تشخیص شیشه‌های شفاف می‌باشند.

به طور کلی، بهترین عملکرد را شبکه‌های آموزش یافته با دیتاست C-CM + C-WO-NatOcc داشتند. C-CM همان دیتاست CelebAMask-HQ با maskهای اصلاح‌شده می‌باشد و دیتاست C-WO-NatOcc از دیتاست CelebAMask-HQ بدون occlusion تهیه شده که به تصاویر آن occlusionهای مختلف مثل دست و اشیاء طبیعی مختلف که ممکن است جلوی صورت قرار بگیرند افزوده شده‌است. دقت شود که این دیتاست فاقد occluderهایی مثل عینک آفتابی و ماسک صورت می‌باشد.

۳. آیا کلاس‌بندی داده‌ها لزومی دارد؟

بله، از آنجایی که هدف ما face segmentation تصاویر حاوی occlusion می‌باشد، نیاز است که پیکسل‌های هر تصویر لیبل‌گذاری شوند تا مشخص شود متعلق به چه کلاسی (face یا background) هستند. با دقت به عملکرد شبکه با آموزش توسط دیتاست‌های مختلف هم می‌توانیم به همین موضوع پی ببریم. به طور کلی، بهترین عملکرد را شبکه‌های آموزش یافته با دیتاست C-CM + C-WO-NatOcc داشتند. در دیتاست C-CM، maskهای دیتاست اولیه اصلاح شدند و همچنین افزودن دیتاست C-WO-NatOcc به داده‌ی آموزش منجر به عملکرد بهتری می‌شود.

۴. استفاده از شبکه‌های مطالعه شده در درس

در چنین شرایطی بهتر است از مدل‌های Encoder-Decoder-based استفاده کنیم. یک نمونه از این مدل ها که در کلاس یاد گرفتیم شبکه‌ی U-net می‌باشد. معماری U-net شامل دو بخش می‌باشد: یک مسیر فشرده‌سازی که محتوای کلی تصویر را capture کند و یک مسیر متقارن گسترده‌سازی که قابلیت localization دقیق را فراهم می‌کند. از آنجایی که در این شبکه، بخش down-sampling شبکه به بخش up-sampling شبکه کپی می‌شود، جلوی از دست رفتن اطلاعات الگوها گرفته می‌شود. معمولا shortcut connectionهایی از encoder به decoder وجود دارد که کمک می‌کند دکودر جزئیات اشکال را بهتر recover کند. با استفاده از این ساختار می‌توان مشکل تفاوت intensity را حل کرد.

۵. مقایسه‌ی کارآیی PSPNet و DeepLab

:DeepLab

- وجود astrous convolutionها در این شبکه، بدون نیاز به افزایش تعداد پارامترها باعث افزایش field of view می‌شود.
- امکان پردازش تصاویر با سایزهای متفاوت با استفاده از : دادن چندتا از هر تصویر resizeشده به branchهای موازی از CNN و یا استفاده از چند لایه موازی astrous convolutional با sampling rate های متفاوت
- پیش‌بینی با استفاده از fully- connected CRF انجام می‌شود. آموزش و tune کردن CRF به عنوان یه گام post processing جداگانه انجام می‌شود

:PSPNet

- استفاده از ماژول pyramid pooling که باعث capture کردن و تجمیع محتوای کلی تصویر می‌شود
- استفاده از auxiliary loss (یک loss اضافه بر loss شاخه‌ی اصلی تا به optimize کردن فرآیند یادگیری شبکه‌های عصبی کمک کند)

همچنین می‌توان عملکرد این دو شبکه را روی دیتاست‌های train و test ارائه شده توسط مقاله در جدول ۲ و ۳ مشاهده کرد.

جدول ۲ - performance کلی مدل‌ها

	Quantity	**RealOcc** (mIoU)			**COFW (Train)** (mIoU)			**RealOcc-Wild** (mIoU)		
		PSPNet	DeepLabv3+	SegFormer	PSPNet	DeepLabv3+	SegFormer	PSPNet	DeepLabv3+	SegFormer
C-Original	29,200	89.52	88.13	88.33	89.64	88.62	91.36	85.21	82.05	85.24
C-CM	29,200	96.15	96.13	97.42	91.82	92.77	**94.87**	91.33	91.01	95.16
C-WO	24,602	89.38	89.01	91.36	89.53	88.97	92.24	83.86	84.14	86.72
C-WO + C-WO-NatOcc	24,602 + 49,204	96.65	96.51	97.30	90.71	91.21	94.30	91.34	91.70	94.17
C-WO + C-WO-NatOcc-SOT	24,602 + 49,204	96.35	96.59	97.18	**92.32**	91.74	93.55	**93.26**	92.69	94.27
C-WO + C-WO-RandOcc	24,602 + 49,204	95.09	95.21	96.53	90.82	91.35	93.14	89.54	89.68	92.84
C-WO + C-WO-Mix	24,602 + 73,806	96.55	96.66	97.37	90.99	91.20	93.74	92.14	91.84	94.40
C-CM + C-WO-NatOcc	29,200 + 49,204	**97.28**	**97.33**	97.95	91.61	92.66	94.86	92.13	**93.81**	**95.43**
C-CM + C-WO-NatOcc-SOT	29,200 + 49,204	97.17	97.29	**98.02**	92.07	**92.91**	94.60	92.84	93.73	94.53

جدول ۳ - performance کلی مدل‌ها (ادامه)

	Quantity	**CelebAMask-HQ-WO (Test)** (mIoU)			**COFW (Train) (cropped and aligend)** (mIoU)		
		PSPNet	DeepLabv3+	SegFormer	PSPNet	DeepLabv3+	SegFormer
C-Original	29,200	97.71	97.23	97.18	93.21	92.37	92.87
C-CM	29,200	**97.78**	**97.79**	**97.88**	95.34	**95.32**	**95.62**
C-WO	24,602	97.66	97.70	97.84	92.88	92.74	93.54
C-WO + C-WO-NatOcc	24,602 + 49,204	97.77	97.76	97.86	94.45	94.46	94.87
C-WO + C-WO-NatOcc-SOT	24,602 + 49,204	97.71	97.77	97.87	94.61	94.47	94.63
C-WO + C-WO-RandOcc	24,602 + 49,204	97.68	97.76	97.83	93.97	93.83	94.19
C-WO + C-WO-Mix	24,602 + 49,204	97.70	97.76	97.76	93.92	94.55	94.67
C-CM + C-WO-NatOcc	29,200 + 49,204	97.74	**97.79**	97.87	**95.38**	**95.32**	95.53
C-CM + C-WO-NatOcc-SOT	29,200 + 49,204	**97.78**	97.76	97.85	95.26	95.23	95.46

این دو جدول می‌توان گفت:

به نظر می‌رسد که مدل های DeepLabv3+ و PSPNet نتایج تقریبا مشابهی به دست آورده اند.

PSPNet با آموزش روی دیتاست‌های C-WO+C-WO-NatOcc, C-Original, C-CM, C-WO و C-WO+C-WO-NatOcc-SOT به طور میانگین در ولیدیشن بهتر عمل کرده است، در حالی که DeepLabv3+ روی سایر دیتاست های Augment شده بهتر عمل کرده است.

پاسخ ۳ – تشخیص بلادرنگ اشیاء

۱. توضیح نحوه شخصی سازی یک مجموعه داده ی جدید روی YOLOv6

Yolov6 مخفف "You Only Look Once" به معنی شما فقط یکبار نگاه میکنید است این الگوریتم به جای دو مرحله شناسایی و دسته بندی در یک مرحله و بصورت بلادرنگ سعی میکند اشیا را تشخیص دهد و برچسب گذاری کند به این ترتیب محاسبات مورد نیاز کاهش و سرعت بالا میرود برای مواقع نیاز بلادرنگ مثل وب کم کاربرد دارد . این الگوریتم در اساس تصاویر train وزن های شبکه را بدست می آورد و سپس به کمک آن وزن های بدست آمده برای تصاویر جدید میتوانیم اشیا را در یک تصویر شناسایی کنیم .

در اینجا بازا هر تصویر دیتاست یک فایل برچسب (label) داریم که هر سطر آن نشان دهنده ی یک شی در تصویر است و در آن به فرمت YOLO TXT اطلاعات مربوط به آن شی درج شده است . در هر سطر 5 عدد داریم به اینصورت که عدد اول شماره کلاس آن شی است سپس 4 عدد که نشان دهنده ی محدوده ای است که شی مربوطه در آن قرار دارد به اینصورت که دوم و سوم به ترتیب مختصات x و y مرکز محدوده را نشان میدهند و اعداد چهارم و پنجم اندازه ی width و height محدوده را نشان میدهند .در زیر عناوین هر سطر آمده و مقادیر یک سطر از داده ها هم زیرش درج شده است.

class_id center_x center_y bbox_width bbox_height

9 0.13942307692307693 0.8173076923076923 0.078125 0.1346153846153846

برای هر دیتاست یک فایل به نام dataset.yaml میسازیم که در آن ابتدا آدرس پوشه ی تصاویر train سپس آدرس پوشه ی تصاویر valid بعد آدرس پوشه ی تصاویر test را داریم . و بعد از این 3 آدرس سطری داریم که در آن تعداد کلاس هایمان آمده (بازا هر شی یک کلاس) و سپس سطری که در آن شماره هر کلاس و نام آن کلاس آمده است . در این دیتاست برای هر رنگ و هر نوع از مهره یک کلاس داریم.

train: ./data/images/train

val: ./data/images/valid

test: ./data/images/test

nc: 13

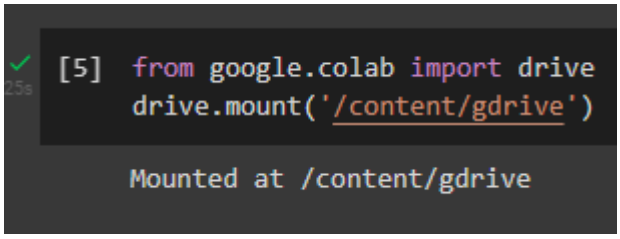
names: ['bishop', 'black-bishop', 'black-king', 'black-knight', 'black-pawn', 'black-queen', 'black-rook', 'white-bishop', 'white-king', 'white-knight', 'white-pawn', 'white-queen', 'white-rook']\

۲. فرآیند شخصی سازی دیتاست روی YOLOv6

از آدرس https://github.com/meituan/YOLOv6 مدل را دریافت میکنیم.

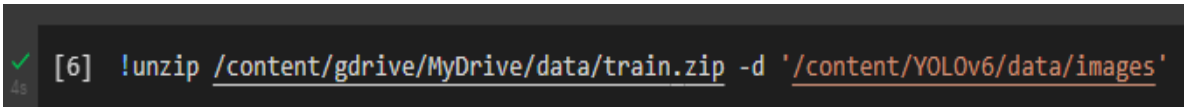
تمام کتابخانه های موجود در فایل requirements.txt را باید نصب کنیم که به کمک pip اینکار را انجام میدهیم.

سپس فایل های تصاویر و برچسب هایشان را در گوگل درایو ذخیره میکنیم تا از آنها استفاده کنیم .ابتدا پروژه کولب را به گوگل درایو متصل میکنیم تا به کمک کد زیر:



شکل ۱۲- قطعه کد برای اتصال به گوگل درایو

سپس فایل های مربوطه را از درایو خوانده و به کمک دستور زیر در پوشه های مربوطه در پروژه برای استفاده الگوریتم قرار میدهیم.



شکل ۱۳- قطعه کد برای خواندن فایل از درایو

سپس به کمک دستور زیر مدل را آموزش میدهیم .

```
!python tools/train.py --batch 16 --conf configs/yolov6s.py --data-path dataset.yaml --device 0 --epochs 100 --eval-interval 2
```

از فایل train.py برای آموزش استفاده میکند ، تعداد batch ها را نیز میتوانیم با این دستور تعیین کنیم آدرس تصاویر و فایل label های train و بقیه موارد مورد نیاز مثل کلاس ها را نیز از فایل dataset.yaml میخواند و در تعداد epoch تعیین شده مدل را آموزش میدهد .

ابتدا با تعداد 20 ایپاک آموزش دادیم که مدل نتوانست دقت لازم را بدست آورد این میزان ایپاک کافی نیست.

```
Results saved to runs/train/exp1
Epoch: 19 | mAP@0.5: 0.09609019176710559 | mAP@0.50:0.95: 0.04163158849534492

Training completed in 0.112 hours.
```

شکل ۱۴- نتایج آموزش در ۲۰ ایپاک

سپس با ۱۰۰ ایپاک ترین را انجام دادیم که توانستیم با این تعداد ایپاک مدل مناسب را بدست آوریم.

```
Epoch: 99 | mAP@0.5: 0.9802471289212636 | mAP@0.50:0.95: 0.7507316796403379

Training completed in 0.563 hours.
```

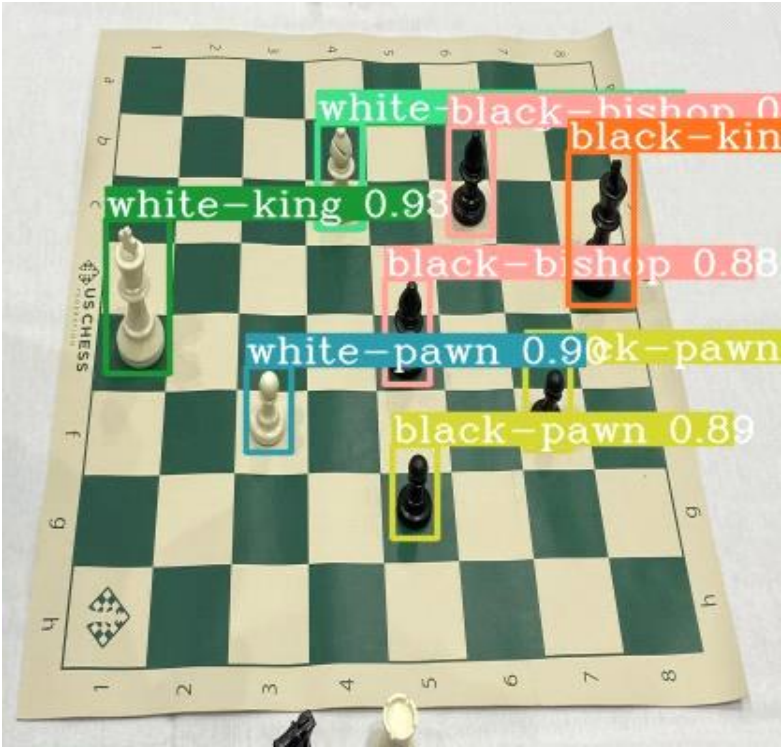
شکل ۱۵- نتایج آموزش در ۱۰۰ ایپاک

وقتی مدل ترین میشود وزن های شبکه را در پوشه ی weights با نام best_ckpt.pt ذخیره میکند سپس باید از دستور زیر کمک بگیریم تا با وزن های به دست آمده تشخیص شی را روی تصاویر تست انجام دهیم.

```
!python tools/infer.py --weights runs/train/exp/weights/best_ckpt.pt --source $img --yaml dataset.yaml --device 0
```

۳. segment شده مهره های شطرنج همراه با برچسب دقت بر روی تصویر

نتایج دسته بندی بدست آمده برای تمام تصاویر در پوشه Results قرار دارد برای نمونه ۳ تصویر دسته بندی شده را در زیر می آوریم:



شکل ۱۶- دسته بندی تصویر۱

شکل ۱۷- دسته بندی تصویر ۲

شکل ۱۸- دسته بندی تصویر ۳

در همه ی تصاویر عملکرد خوبی دارد و با دقت مناسب مهره های شطرنج را پیدا میکند و نوع آنها را برچسب گذاری میکند.