به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین اول

نام و نام خانوادگی	محمد ناصری – مریم عباس‌زاده
شماره دانشجویی	810100406 - 810100486
تاریخ ارسال گزارش	۱۴۰۱.۰۹.۱۵

فهرست

شکل‌ها

جدول‌ها

پاسخ ۱. آشنایی با یادگیری انتقالی Transfer Learning

۱-۱. کاربرد شبکه های عصبی کانولوشنال DenseNet برای پیش بینی COVID-19 با استفاده از تصاویر CT

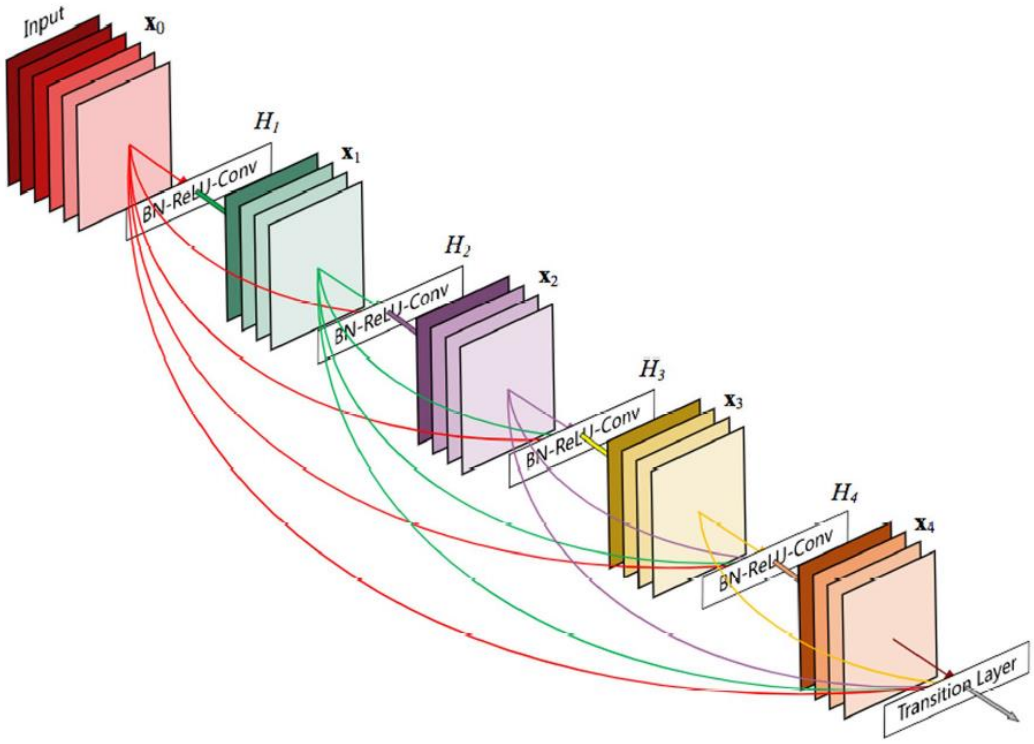
با توجه به در دسترس نبودن واکسن کووید-19 برای کنترل به جای درمان، تشخیص زودهنگام و دقیق ویروس می تواند یک تکنیک امیدوارکننده برای ردیابی و جلوگیری از گسترش عفونت باشد (به عنوان مثال، با جداسازی بیماران). تصویربرداری توموگرافی کامپیوتری (CT) یک تکنیک پرکاربرد به دلیل در دسترس بودن مورد است. تجزیه و تحلیل تصاویر به کمک هوش مصنوعی ممکن است یک جایگزین امیدوارکننده برای شناسایی COVID-19 باشد. این مقاله یک تکنیک برای پیش‌بینی بیماران COVID-19 از تصویر CT با استفاده از شبکه‌های عصبی کانولوشنال (CNN) ارائه می‌کند. رویکرد جدید بر اساس جدیدترین معماری CNN تعریف شده (DenseNet-121) و برای پیش بینی COVID-19 است. نتایج عملکرد قابل قبولی را برای پیش‌بینی COVID-19 نشان می‌دهد.

DenseNet معماری مدرن CNN برای تشخیص بصری اشیاء است که با پارامترهای کمتری پیشرفته‌تر شده است. با برخی تغییرات اصلی، DenseNet بسیار شبیه ResNet است. DenseNet، همراه با ویژگی های الحاقی (.) خود، خروجی لایه قبلی را با یک لایه آینده ترکیب می کند، در حالی که ResNet از یک ویژگی افزودنی (+) برای ادغام لایه قبلی با لایه های آینده استفاده می کند. هدف معماری DenseNet رفع این مشکل با اتصال متراکم تمام لایه‌ها است.

DenseNets واقعاً نقشه های عملکرد خروجی لایه را با ورودی ها خلاصه نمی کند، بلکه آنها را به هم متصل می کند. DenseNet یک مدل ارتباطی آسان برای بهبود جریان اطلاعات بین لایه ها ارائه می دهد: لایه lام ورودی ها را از ویژگی های تمام سطوح قبلی دریافت می کند

$$X_l = H_l[(X_0, X_1, \dots, X_{l-1})]$$

که در آن $(X_0, X_1, \dots, X_{l-1})$ یک Tensor منفرد است که از الحاق نقشه های خروجی لایه های قبلی تشکیل شده است. خارج از توابع، Hl(.) یک تابع تبدیل غیرخطی را نشان می دهد. این تابع از سه عملیات اصلی تشکیل شده است، نرمال سازی دسته ای (BN)، فعال سازی (ReLU) و ادغام و کانولوشن (CONV).



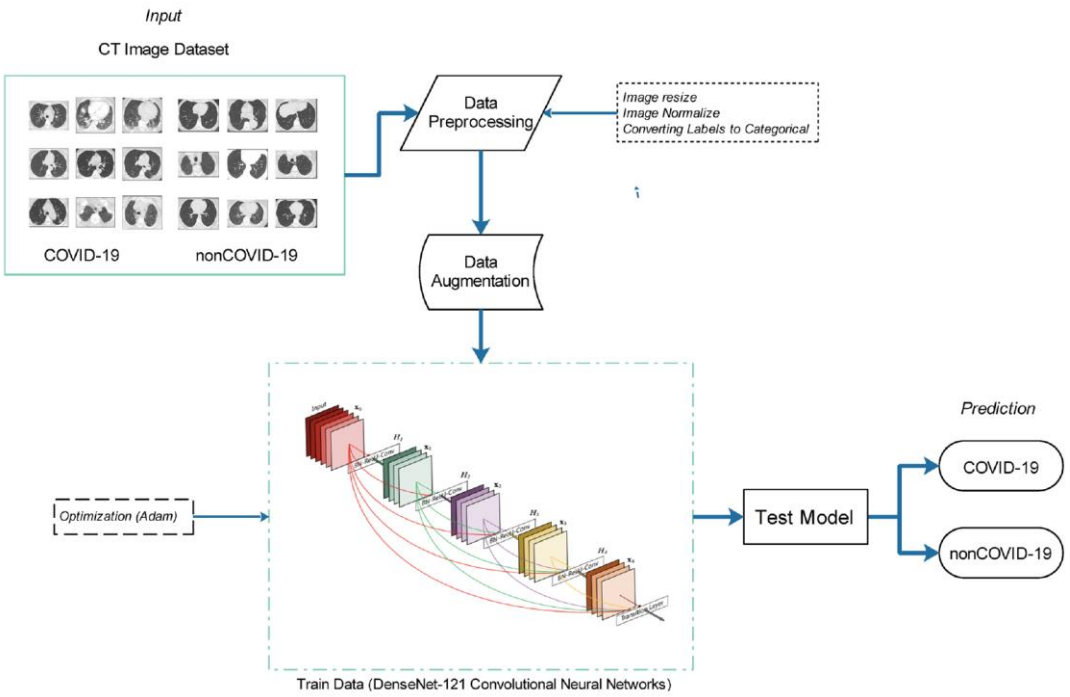
شکل 1 - معماری **Densenet**

پیش پردازش

هدف از مرحله پیش پردازش تصویر، خفه کردن پیچش های ناخواسته موجود در تصویر، تغییر اندازه و نرمالسازی تصویر برای پردازش بیشتر است. تغییر اندازه تصویر، نرمالسازی تصویر، و تبدیل Level به طبقه بندی معمولاً از تکنیک هایی هست که استفاده می شود. در این مطالعه، اندازه تصاویر با استفاده از بسته پایتون "Pillow 2.7+ +" به اندازه یکسان و پیکسل یکسان تغییر داده شد. این مطالعه مقادیر 64 × 64 پیکسل را برای تصاویر در نظر می گیرد. علاوه بر این، نرمالسازی تصویر نحوه تنظیم شدت پیکسل است تا تصویر به طور فزاینده ای طبیعی شود. به طور معمول، اکثر پیکسل های تصویر مقادیر بین 0 تا 255 را یکپارچه می کنند. اما به دلیل معماری شبکه، بهتر است تمام مقادیر بین 0 و 1 انجام شود که برای ساختمان مدل مناسب است. این تکنیک پیچیدگی محاسباتی را در طول آموزش مدل کاهش می دهد.

$$X_{norm} = \frac{X - X\_min}{X\_\max - X\_min}$$

پیاده‌سازی



شکل 2 - پیاده سازی الگوریتم

مدل مربوطه به شکل زیر پیاده سازی میشود:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 64, 64, 3)]       0

conv2d (Conv2D)              (None, 64, 64, 3)         84

densenet121 (Functional)     (None, None, None, 1024)  7037504

global_average_pooling2d (G  (None, 1024)              0
lobalAveragePooling2D)

batch_normalization (BatchN  (None, 1024)              4096
ormalization)

dropout (Dropout)            (None, 1024)              0

dense (Dense)                (None, 256)               262400

batch_normalization_1 (Batc  (None, 256)               1024
hNormalization)

dropout_1 (Dropout)          (None, 256)               0

root (Dense)                 (None, 2)                 514

=================================================================
Total params: 7,305,622
Trainable params: 7,219,414
Non-trainable params: 86,208
_________________________________________________________________
```

شکل 3 - مدل پیاده سازی شده

مزایا و معایب

مزایا:

- مشکل ناپدید شدن گرادیان را کاهش میدهد
- ارتقای ویژگی را هم در حالت رو به جلو و هم به سمت عقب بهبود میدهد
- به استفاده مجدد از ویژگی توجه شده
- کاهش تعداد پارامترها

از نظر ظاهری، DenseNets کاملا شبیه ResNets هستند. اما قالب ورودی در دو شبکه متفاوت است که منجر به رفتارهای متفاوتی می شود. جدول زیر تفاوت های بین دو شبکه را نشان می دهد.

جدول 1 - تفاوت‌های **ResNet** و **DenseNet**

Feature	Densenet	Resnet
Format of passing previous layer features to other layers	by concatening	by summation
No of inputs to l^{th} layer	l	1
Total no of connections in L-layer network	$L(L+1)/2$	L
The way to address vanishing-gradient problem	using dense connections	using stochastic depth
Performance Improvement	using power of deep architecture	using power of feature reuse

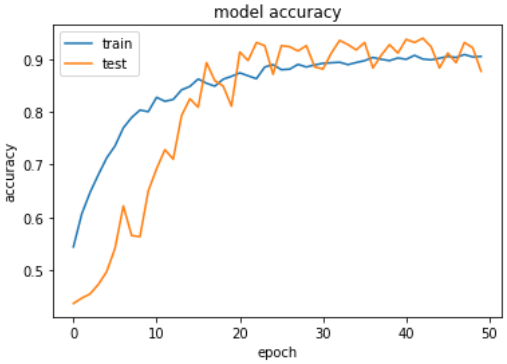
معایب

- در مقایسه با ResNet، DenseNet از حافظه بسیار بیشتری استفاده می کند، زیرا تنسورهای مختلف به یکدیگر متصل می شوند.

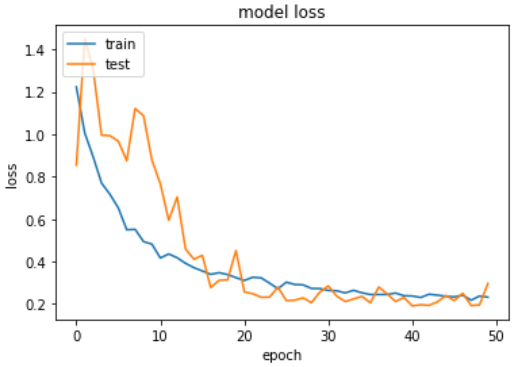
نتایج نهایی پیاده سازی

پس از پیاده‌سازی و کار با دیتاست نتایج نهایی به شکل زیر بود:

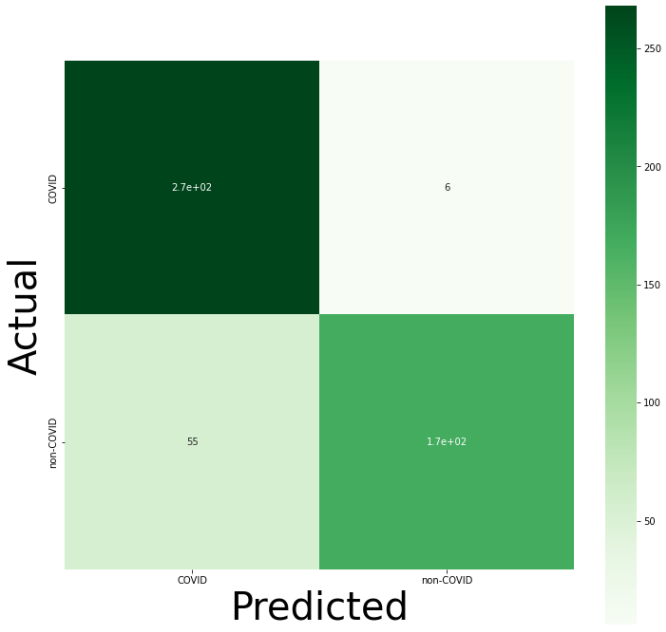
- LOSS: 0.2970520555973053,
- ACCURACY: 0.877263605594635,
- F1_SCORE: 0.8774126172065735,
- PRECISION: 0.8774126768112183,
- RECALL: 0.8774126 768112183



شکل **6** - نمودار دقت مدل



شکل **4** - نمودار **Loss** مدل



شکل **5** - ماتریس کانفیوژن مدل

پاسخ ۲ – آشنایی با تشخیص چهره مسدود شده

1)

آماده سازی دیتاست:

دیتاست مورد مطالعه از دو کلاس تشکیل شده است، پس زمینه و صورت، منطقه‌ی خاکستری یا مبهم.

صورت: قسمت پوست سر که شامل چشم‌ها، بینی، دهان می‌شود. ولی گوش را شامل نمی‌شود.

صورت (منطقه خاکستری): تتو، سایه، ریش و سبیل که با صورت همپوشانی دارند، همچنین پوست سر شخص طاس نیز جزئی از صورت محسوب می‌شود.

پس زمینه: منطقه غیر صورت، هر شیء ای مانند عینک، پیراهن، مو، میکروفون، دست ها به صورت فیزیکی پوست صورت را می‌پوشانند

پس زمینه (منطقه خاکستری): شفاف/ عینک شفاف

120 شیء معمولی در 20 دسته(مانند غذا، بطری، تلفن همراه و فنجان) از مجموعه داده‌های Microsoft Common Objects in Context (COCO) با COCOAPI استخراج شده است. با توجه به پایین بودن وضوح اصلی اشیاء COCO ، بنابراین وضوح فوق العاده (×4) تصاویر اصلی را با GLEAN انجام داده شده است. برخلاف سایر انسدادها، دستها رنگی شبیه به صورت دارند و تشخیص آنها به عنوان انسداد دشوارتر می شود. چندین نوع دیتاست برای دست وجود دارد. در ابتدا 200 دست از دیتاست EgoHands نمونه برداری شده است. با توجه به اینکه این تصاویر دست ها وضوح کمی دارند و تار هستند، پس اعمال تغییرات در آن ها لبه‌ها و جزئیات عقربه‌ها پس از تغییر اندازه حفظ نمی‌شوند و روی تصاویر CelebAMask-HQ قرار نمی‌گیرند، بنابراین 200 تصویر دست از دیتاست 11k Hands که وضوح بالاتر 1600×1200 دارد، نمونه‌برداری شده است. به دلیل عدم وجود ماسک های خوب، 200 تصویر دست به صورت دستی حاشیه نویسی شده اند. تنها ایراد این دیتاست دست ها عدم وجود حالت های مختلف است.

برای داده‌های ارزیابی از یک مجموعه Reallocc استفاده شده است که شامل 550 تصویر چهره‌ی انسداد شده با وضوح بالا (1024*1024) است که از وبسایت هایی مانند Pexels و Unsplash است که انسدادهای مختلف (مانند دست‌ها، ماسک‌ها، غذا و عینک آفتابی) را پوشش می‌دهد. برای ارزیابی استحکام مدل از دیتاست RealOcc-Wild استفاده شده است که متشکل از 270 چهره انسداد شده در طبیعت (بدون برش و تراز) هستند و توسط dlib شناسایی نشده اند.

تولید تصاویر با انسداد طبیعی(NatOcc)

از آنجایی که تعداد داده های با چهره مسدود شده در مقیاس بزرگتر در دنیای واقعی کم است، تکنیک های مختلفی برای تولید چنین تصاویری وجود دارد، در این مطالعه از الگوریتم NatOcc برای تولید تصاویر چهره مسدود شده طبیعی با کیفیت بالاتر از CelebAMask-HQ-WO استفاده شده است. (برای داده های آموزش)

روش کار الگوریتم:

در تصاویر واقعی، بیشتر دست‌هایی که روی صورت قرار می‌گیرند رنگ پوستی شبیه به رنگ پوست صورت دارند. برای شبیه سازی این سناریو با استفاده از color transfer via Sliced Optimal Transport (SOT) رنگ صورت را به دست ها انتقال می‌دهند. برای این کار لازم است اندازه تصویر مبدا(صورت) و مقصد(دست) برابر باشند. اگر تعداد پیکسل‌های سیاه در تصویر مبدا بیش تر از تصویر مقصد باشد، در برخی از بخش های تصویر دست سیاهی ظاهر می‌شود. همچنین اگر بخشی از تصویر صورت بسیار روشن باشد، دست‌ها غیر طبیعی به نظر می‌رسند. برای حل این مشکل، نسبتی پیکسل های سیاه موجود در تصویر صورت با میانگین هریک از کانال های RGB، پیکسل های تصویر صورت جایگزین می‌شوند.

افزایش داده:

تقویت افاین، فشرده سازی تصویر، روشنایی و کنتراست تصادفی به کمک albumentations بر روی هر دو تصویر صورت و انسداد کننده اعمال می‌شود. علاوه بر این تصویر انسداد کننده، به طور تصادفی بین 0.5 تا 1 برابر اندازه تصویر صورت تغییر می‌کند. لبه‌های تصویر انسداد کننده با دقت در نظر گرفته می‌شوند تا تصویر طبیعی تری ایجاد شود. این کار با اعمال بلور گاوسین بر روی ماسک انسداد کننده قبل از ترکیب آلفا انجام می‌شود (روشی برای پوشاندن یک تصویر پیش زمینه بر روی یک تصویر پس زمینه). انسداد کننده ها به طور تصادفی در اطراف صورت قرار می‌گیرند.

هماهنگ سازی تصویر:

برای بیشتر طبیعی جلوه دادن تصویر، RainNet را برای هماهنگ کردن پیش زمینه (انسداد کننده) برای مطابقت با پس زمینه (صورت)اعمال شده است. با این حال، این مورد برای مسدود کننده دست یکسان نبود زیرا رنگ دست پس از هماهنگی تصویر تغییر کرد و هدف از انتقال رنگ را شکست داد. بنابراین، هماهنگی تصویر برای انسداد دست اعمال نشد.

تولید تصاویر با انسداد تصادفی:

برای پوشاندن دست‌ها بر روی صورت، مطالعاتی صورت گرفته است که تصاویر صورت هدف منطبق را با حالتی مشابه با تصویر صورت مبدأ پیدا می‌کنند و به دنبال آن، دست‌های تصویر مبدأ را روی تصویر صورت هدف قرار می‌دهند. پوشاننده هایی مانند عینک آفتابی و ماسک صورت به ترتیب روی چشم ها و دهان قرار گرفتند .چنین روش تولید مجموعه داده مصنوعی را نمی توان برای اکثر برنامه ها اعمال کرد. یک روش انسداد عمومی تر Random Occlusion Generation (RandOcc) که داده‌های انسداد مصنوعی را با حداقل تلاش ایجاد می‌کند، معرفی شده است.

(2

طبق جدول زیر، مدل‌هایی که با دیتاست های NatOcc ترین شده اند C-WO-NatOcc-SOT, C-WO-NatOcc) عملکرد بهتری نسبت به مدل‌هایی دارند که با دیتاست real-world occluded face ترین شده اند.

مدل های CNN آموزش دیده با مجموعه داده های NatOcc می توانند چهره ها را با دقت بیشتری نسبت به مدل های آموزش دیده با C-CM تقسیم بندی کنند .افزودن C-CM به مجموعه داده NatOcc عملکرد مدل را بیشتر افزایش می‌دهد.

	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

(3

بیس کار انجام شده بر مبنای classification است، در واقع تصویر از دو قسمت صورت که شامل چشم‌ها، دهان و بینی می‌شود اما عینک، گوش و یا هر شیء دیگری که روی صورت قرار بگیرد در گروه بک گراند قرار می‌گیرد. دوواقع خروجی و نتیجه‌ی حاصل از شبکه‌ی طراحی شده تصویری است که در آن کلاس صورت از کلاس بک گراند تشخیص داده شده باشد.

4)

ResNet , VGG برای کلاس بندی تصویر گزینه‌ی مناسبی به شمار می‌آیند. ResNetبه دلیل ویژگی اتصال پرش خوب است در حالی که VGG به دلیل ساختار رمزگذار/رمزگشا خوب است. می توان هر دو عمل طبقه بندی و تقسیم بندی را با هم در چنین معماری VGG انجام داد.

به طور مشابه، Unetو VGGSegnet نقطه شروع خوبی برای کارهای تقسیم بندی هستند .معماری رمزگذار-رمزگشا به طور کلی خوب است، که به تقسیم بندی مناسب کمک می کند. با توجه به مطالعات انجام شده به نظر می‌رسد Unet نسبت به سه تای دیگر عملکرد بهتری در این گونه مسائل داشته باشد.

5)

با توجه به نتایج بدست آمده در جدول شماره 4 مقاله، نتایج بدست آمده برای هر دو مدل PSPNet و DeepLab تقریبا به یکدیگر نزدیک هستند و به نوعی می‌توان گفت که هر دو تقریبا کارایی یکسانی برای این مسئله دارند.

در دیتاست با انسداد کننده طبیعی (C- original) PSPNet عملکرد نسبتا بالایی دارد.

درRealOcc بیشترین مقدار PSPNet و DeepLab برای دیتاست C-CM + C-WO-NatOcc است که مقادیر خیلی نزدیک به هم دارند. این مقادیر به ترتیب برابر است با 97.28 و 97.33

در COFW نیز مشابها هر دو مدل مقادیر نسبتا یکسانی دارند که بیشترین مقدار PSPNet برای دیتاست C-WO + C-WO-NatOcc-SOT و برابر با 92.32 است که DeepLab معادل آن برابر با 91.74 می‌باشد. و بیشترین مقدار DeepLab برای دیتاست C-CM + C-WO-NatOcc-SOT و برابر با 92.91 است که PSPNet معادل آن برابر با 92.07 می‌باشد.

در RealOcc-Wild نیز مشابه با دو مجموعه فوق مقادیر PSPNet و DeepLab تفاوت چندان زیادی با هم ندارند. PSPNet بیشترین مقدار خود، یعنی 93.26 را در دیتاست C-WO + C-WO-NatOcc-SOT بدست آورده است که DeepLab معادل آن برابر با 92.69 است. DeepLab بیشترین مقدار خود، یعنی 93.81 را در دیتاست C-CM + C-WO-NatOcc بدست آورده است که PSPNet معادل آن برابر با 92.13 است.

DeepLab بهترین مقادیر خود را در هر سه حالت برای دیتاست های C-CM + C-WO-NatOcc-SOT و C-CM + C-WO-NatOcc بدست آورده است.

پاسخ ۳ – تشخیص بلادرنگ اشیا

۳-۱. شخصی سازی YOLOV6

برای آموزش آشکارساز ما مراحل زیر را انجام می دهیم:

در مرحله اول باید مجموعه داده را در قالب MT-YOLOv6 آماده کنیم. اینکار از قبل در مورد دیتاست در اختیار ما انجا گرفته است. خصوصیات این دیتاست باید به صورت زیر باشد:

فرمت دایرکتوری‌ها

```
# image directory
path/to/data/images/train/im0.jpg
path/to/data/images/val/im1.jpg
path/to/data/images/test/im2.jpg

# label directory
path/to/data/labels/train/im0.txt
path/to/data/labels/val/im1.txt
path/to/data/labels/test/im2.txt
```

فرمت داده text باید به صورت زیر تعریف شود:

```
# class_id      center_x        center_y        bbox_width      bbox_height
  1             0.408           0.302666        0.104           0.15733
```

همچنین باید در روت یک فایل YAML برای توصیف دیتاست وجود داشته باشد. فرمت فایل مورد نظر به شکل زیر خواهد بود:

```
train: ./images/train
val: ./images/valid
test: ./images/test

nc: 12
names: ['black-bishop', 'black-king', 'black-knight', 'black-pawn',
'black-queen', 'black-rook', 'white-bishop', 'white-king', 'white-
knight', 'white-pawn', 'white-queen', 'white-rook']
```

از آنجا که در انجام این سوال از کولب استفاده شده است برای آدرس‌های فایل YAML باید تغییرات زیر اعمال میشد:

```
train:
/content/drive/MyDrive/NNDL_Spring2022/CA3/YoloV6_Chess/images/train

val: /content/drive/MyDrive/NNDL_Spring2022/CA3/YoloV6_Chess/images/valid

test: /content/drive/MyDrive/NNDL_Spring2022/CA3/YoloV6_Chess/images/test
```

حال لازم است تا dependency های مربوط به MT-YOLOv6 را نصب میکنیم برای اینکار پس از fetch کردن ریپازیتوری اقدام به نصب میکنیم:

```
!git clone https://github.com/meituan/YOLOv6
%cd YOLOv6
!pip install -r requirements.txt
```

سپس مجموعه داده سفارشی سازی شده را در نوت‌بوک مورد نظر بارگذاری میکنیم. سپس آموزش MT-YOLOv6 را برای داده مورد نظر اجرا میکنیم. برای اینکار epoc=100 در نظر گرفته شده و از آنجا که سایز تصاویر ۴۱۶*۴۱۶ می‌باشد این مورد هم در دستور مربوطه قرار گرفته و آموزش را با به کارگیری یک GPU اجرا میکنیم:

```
!python tools/train.py --batch 32 --conf configs/yolov6s.py --epochs 100 --img-size 416 --data /content/drive/MyDrive/NNDL_Spring2022/CA3/YoloV6_Chess/data.yaml --device 0
```

پس از اجرای آموزش می توانیم عملکرد آموزش سفارشی خود را با استفاده از اسکریپت ارزیابی ارائه شده ارزیابی کنیم. مشابه آموزش، به عنوان نمونه تصاویر ۴۱۶*۴۱۶ را برای ارزیابی وارد می کنیم.

```
!python tools/eval.py --data /content/drive/MyDrive/NNDL_Spring2022/CA3/YoloV6_Chess/data.yaml --img-size 416 --weights runs/train/exp/weights/best_ckpt.pt --device 0
```

نتایج ارزیابی به شرح زیر است:

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.691
 Average Precision  (AP) @[ IoU=0.50       | area=   all | maxDets=100 ] = 0.972
 Average Precision  (AP) @[ IoU=0.75       | area=   all | maxDets=100 ] = 0.832
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.612
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.695
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.582
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.759
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.759
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.630
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.765
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = -1.000
```

شکل 7 - خروجی ارزیابی مدل **YOLO**

در اقدام بعد می‌توانیم inference را روی تصاویر مدل آموزش‌دیده سفارشی خود با استفاده از ابزار استنتاج ارائه شده اجرا کنیم. برای این کار باید فایل .yaml سفارشی خود را پاس کنیم تا نام برچسب درست باشد. همچنین دایرکتوری /test خود را برای اجرای استنتاج بر روی همه تصاویر در تقسیم آزمایشی خود پاس می دهیم.

در نهایت هم برخی خروجی ها را چاپ میکنیم: