به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین سوم

نام و نام خانوادگی	**آناهیتا هاشم زاده – پرهام بیچرانلو**
شماره دانشجویی	**۸۱۰۱۰۰۵۰۲ - ۸۱۰۱۰۰۳۰۳**
تاریخ ارسال گزارش	**۱۴۰۱.۰۹.۱۸**

فهرست

شکل‌ها

جدول‌ها

پاسخ ۱. آشنایی با یادگیری انتقالی Transfer Learning

با توجه به صورت سوال مقاله شماره ۱ «Multiple Classification of Flower Images Using Transfer Learning» با مدل AlexNet بررسی می شود.

۱-۱.

در سال های اخیر تکنولوژی deep learning در بسیاری از زمینه ها موفق بوده، که مسئله image classification یکی از این زمینه ها بوده است. در این مقاله به حل مسئله با استفاده از مدل های pretained پرداخته است. طبقه بندی تصاویر را با رویکرد transfer learning با استفاده از مدل های معروف از پیش آموزش داده شده مثل Alexnet, Googlenet, VGG16, DenseNet و ResNet انجام می دهد و نتایج قابل قبولی برای تمامی مدل های استفاده شده بدست می آورد.

ساختار مقاله به این شکل است که ابتدا بخش اول introduction و بخش دوم توصیف ساختار لایه های شبکه های عصبی کانولوشنال و بخش سوم روش پیشنهادی شامل داده ها و transfer learning و نتایج تجربی است.

این مطالعه تحت شرایط یکسان با استفاده از NVIDIA Geforce Gtx 950M Gpu با شبکه های DenseNet201, googleNet and ResNet,AlexNet, VGG16 اجرا شده است. دیتاست استفاده شده در این مدل Kaggle Flowers data با پنج کلاس tulip, rose, sunflower ,chamomile و dandelion می باشد. هرکلاس شامل ۸۰۰ عکس هست که رزلوشن عکس ها بالا نیست و ۳۲۰*۴۲۰ پیکسل می باشد.

یک تکنیک در ماشین لرنینگ transfer learning می باشد که یک مدل pretrained در یک کار مرتبط دوم دوباره تعریف می شود. در این مطالعه نیز از مدل های pretrained استفاده شده و با دیتاست گل ها re-trained می شوند. از آنجایی که این مدل ها با دیتاست Imagenet ترین شده اند و این دیتاست دارای ۱۰۰۰ کلاس می باشد پس لایه اخر این کلاس دارای خروجی ۱۰۰۰ است ولی دیتاست گل ها شامل ۵ کلاس می باشد، از این رو لایه fully connected که لایه آخر می باشد روی ۵ تنظیم شده است.

مدل های استفاده شده در این مقاله به شرح زیر می باشند:

مدل AlexNet: این شبکه دارای ۱۱ لایه می باشد. تعداد لایه های بالا بر روی استخراج ویژگی ها تاثیر مثبت دارد در حالی که تعداد بالا پارامترها تاثیر منفی بر روی سرعت مدل دارد. بعد از لایه کانولوشنی

در این شبکه max pooling و normalization قرار دارد و در لایه تابع فعالسازsoftmax پروسه طبقه بندی را انجام می دهد.

مدل VGG16: این شبکه جنبه های مهم طراحی معماری CNN را در بر میگیرد و همچنین دارای عمق خیلی زیاد است. در این ساختار جایی که متد افزایش داده استفاده شده بعد از هر لایه کانولوشنی تابع فعال ساز Relu استفاده شده است، و stack با استفاده از gradient descent آموزش داده شده و در تمامی لایه ها از filterهایی با سایز ۳*۳ استفاده شده است.

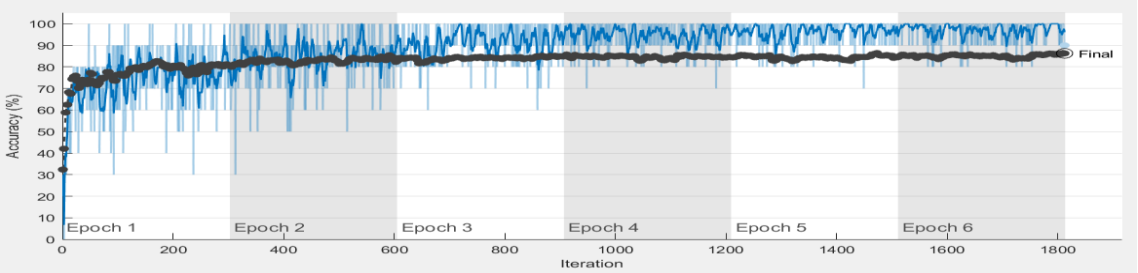
مدل ResNet: در سال ۲۰۱۵ این شبکه توسط Kaiming و همکارانش ارائه شد. چنین پیوندهایی به عنوان gated units, gate repeating units هستند و مشابه اخرین لایه موفق بر روی RNN اعمال می شوند. با استفاده از این تکنیک توانستند یک شبکه خیلی عمیق با پیچیدگی کمتر نسبت به VGGN را آموزش بدهند .

مدل GoogleNet: این شبکه توسط محققان گروه گوگل طراحی شده و عملکرد بسیار مشابه با عملکرد انسان دارد. آموزش این شبکه چند روز به طول انجامید و از یک CNN که از LeNET-5 الهام گرفته شده است. همه لایه های کانولوشنی از تابع فعال ساز خطی استفاده کرده و این ماژول برای کاهش تعداد پارامترها از کانولوشن های کوچک استفاده کرده است. معماری شبکه شامل ۲۲ لایه است که با کم کردن تعداد پارامترها دارای ۶۰-۴ میلیون پارامتر است.

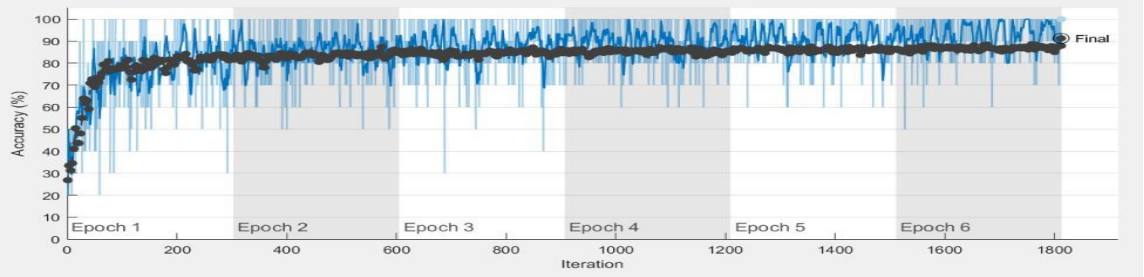
مدل DenseNet: ساختار شبکه به صورت پیوند هر لایه با همه لایه های دیگر به حالت feed forward است، به این صورت که اتصالات بین شبکه های متقاطع نزدیک به ورودی و نزدیک به خروجی کوتاه تر است، که میتواند عمیق تر و برای آموزش دقیق تر و مناسب تر باشد. این شبکه به پیشرفت های قابل توجهی در همه زمینه ها دست پیدا کرد و برای عملکرد بهتر به معماری و محاسبات کمتر نیاز دارد.

در این مطالعه هر مدل با ماکزیمم ۱۸۱۲ ایپاک و یک GPU اجرا شده است. طبق نتایج بدست آمده شبکه VGG16 دارای بالاترین دقت validation بوده است و اگر زمان اجرا را در نظر بگیریم DenseNet201 طولاترین زمان اجرا را داشته است.

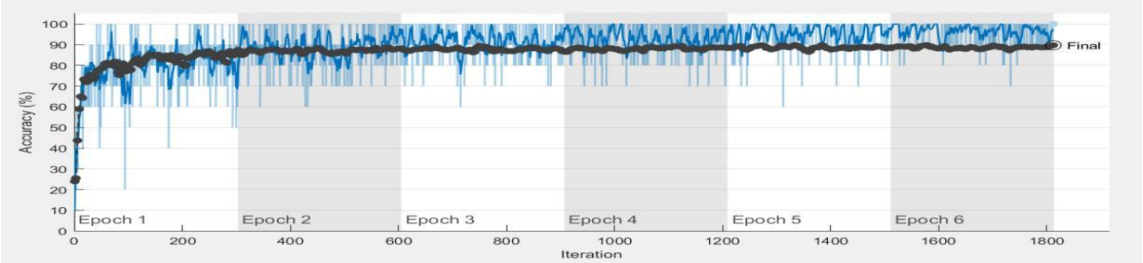
a) AlexNet



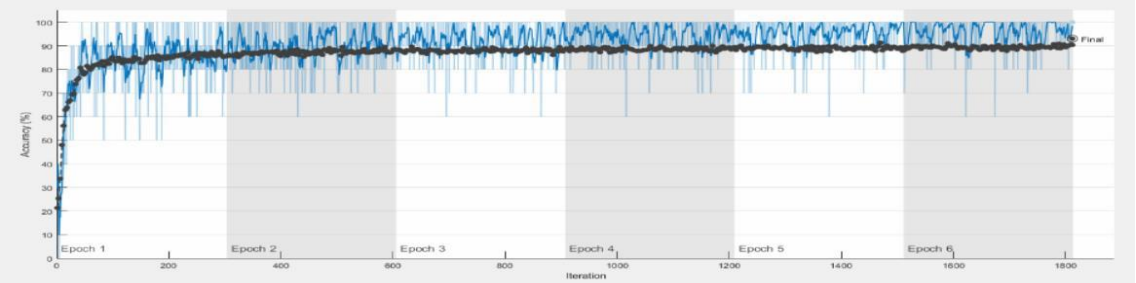
b) ResNet18



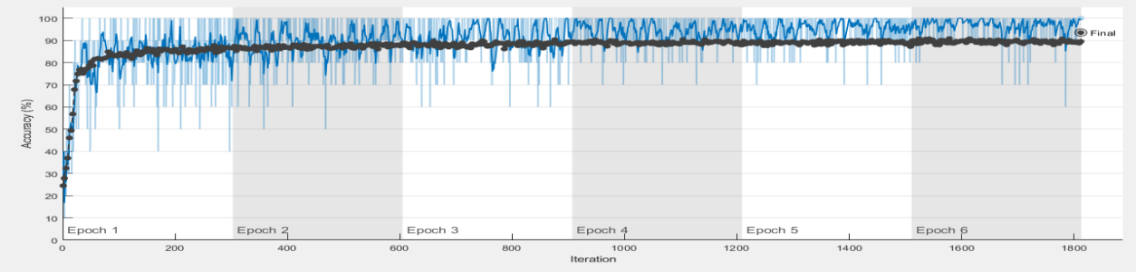
c) GoogleNet



d) DenseNet201

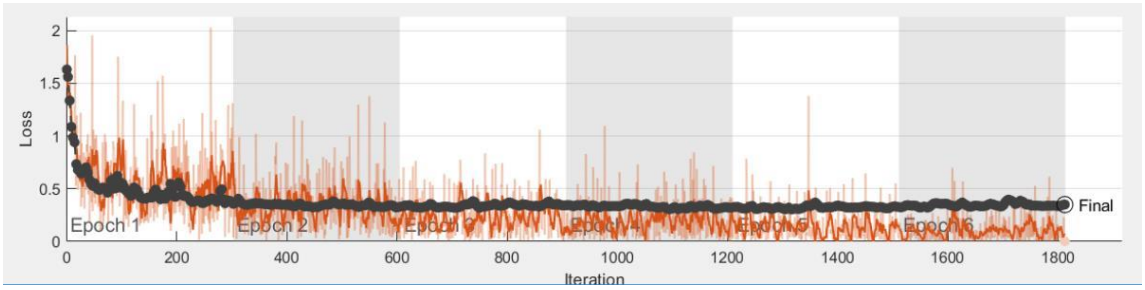


e) VGG16

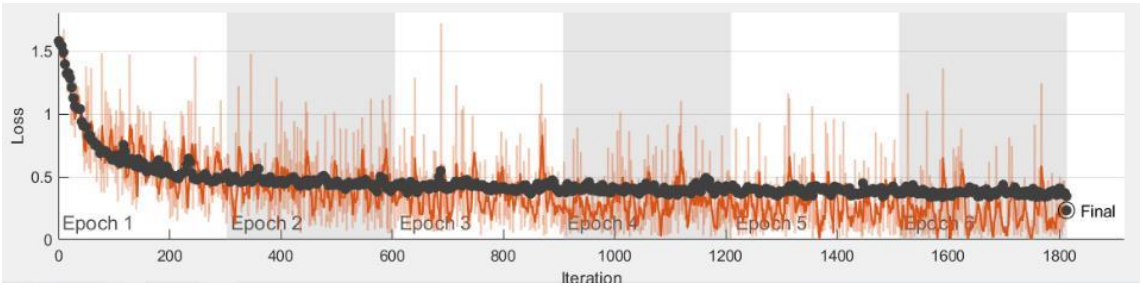


شکل ۱. نمودار دقت مدل ها

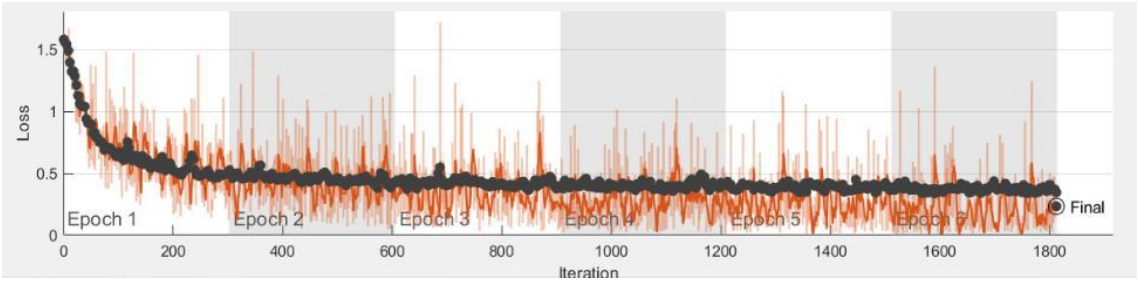
a) AlexNet



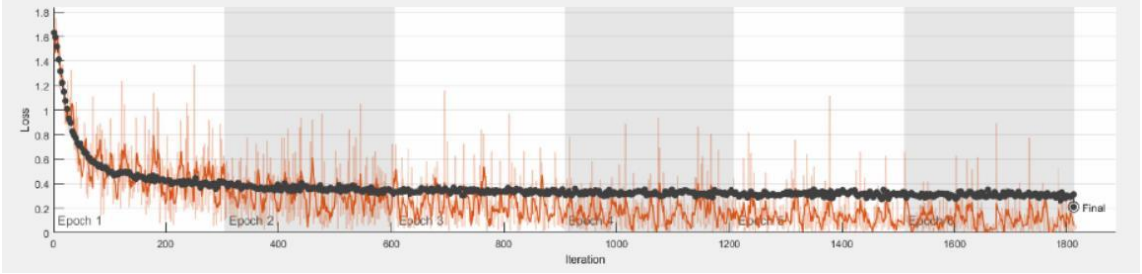
b) ResNet18



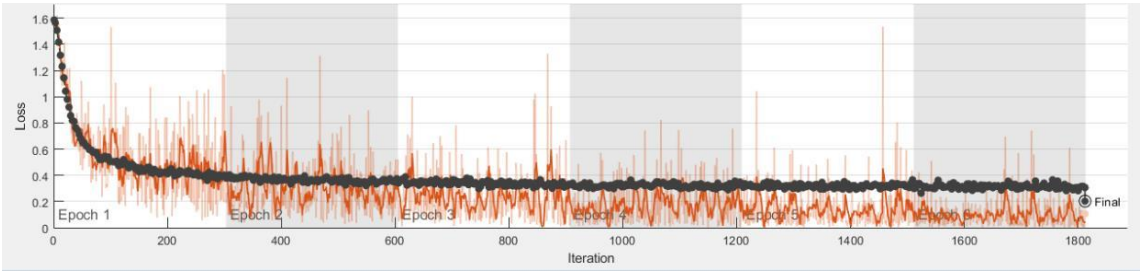
c) GoogleNet



d) DenseNet201



e) VGG16



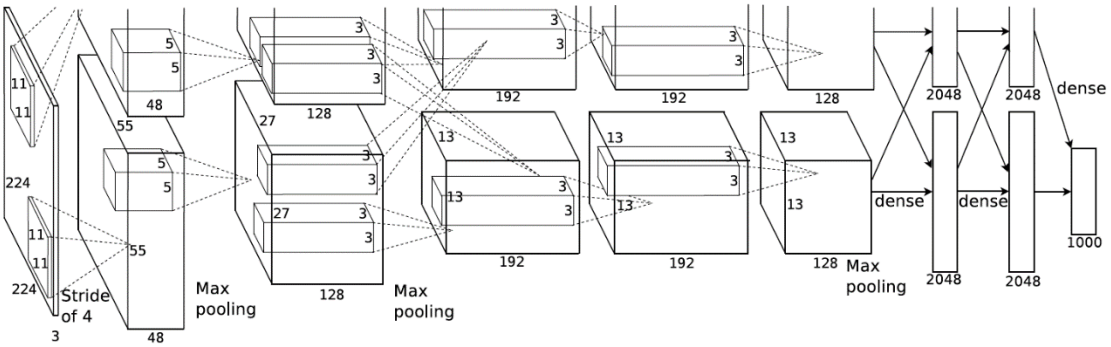
شکل۲. نمودار **loss** مدل ها

جدول ۱. نتایج تجربی مقاله

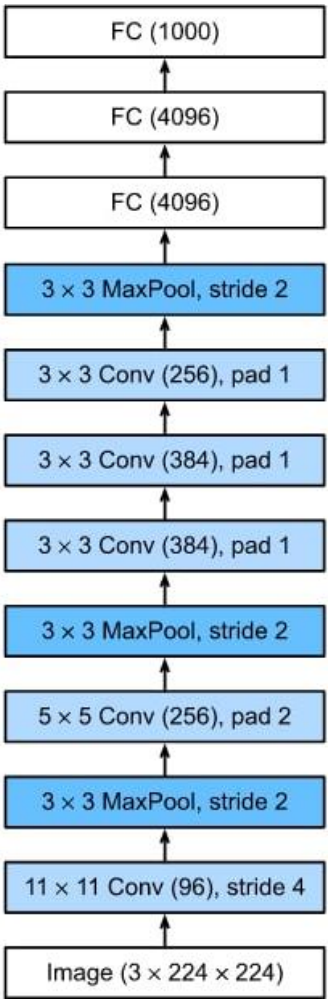
models	Validation Accuracy	Elepsad time	Hardware Resource	Max. Iterations
AlexNet	86.28	108 min 49 sec.	Single GPU	1812
Resnet18	91.29	186 min. 40 sec.	Single GPU	1812
GoogleNet	89.75	127 min. 22 sec.	Single GPU	1812
DenseNet201	93.06	479 min. 21 sec.	Single GPU	1812
VGG16	93.52	467 min. 37 sec.	Single GPU	1812

در ادامه همانطور که در صورت سوال خواسته شده فقط از مدل از پیش آموزش داده شده AlexNet برای retrain کردن استفاده میکنیم .

۱-۲.



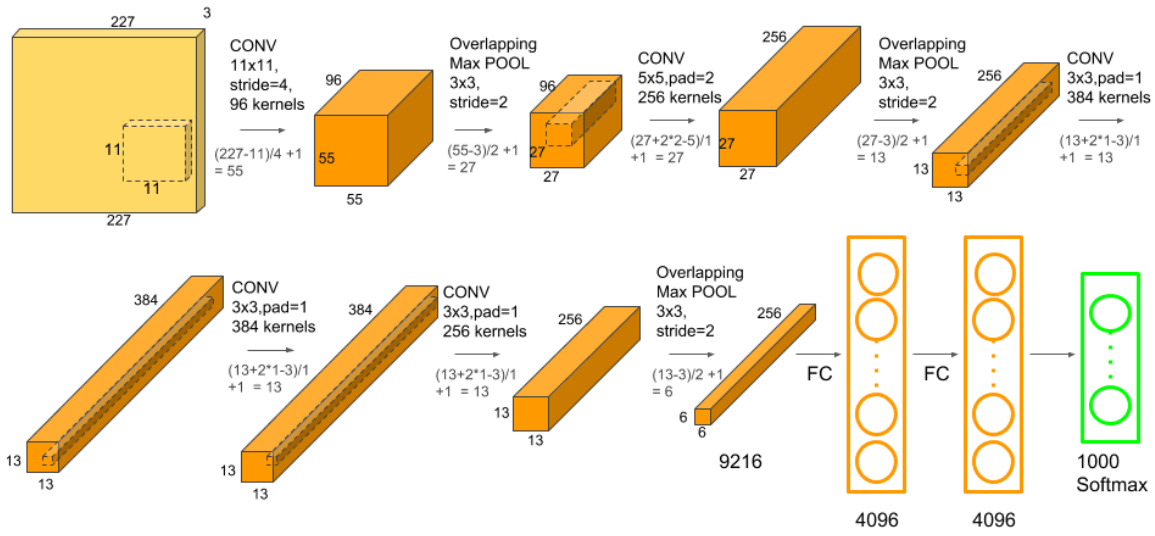
شکل ۳. معماری شبکه **AlexNet**



شکل۴. لایه های شبکه **AlexNet**

معماری AlexNet در سال ۲۰۱۲ در ImageNet competition معرفی شد. AlexNet اولین شبکه convolutional است که برای افزایش عملکرد از GPU استفاده کرد.

- معماری AlexNet شامل ۵ لایه کانولوشنی، ۳ لایه max-pooling، ۲ لایه نرمالیزیشن، ۲ لایه fully connected، و یک لایه softmax می باشد.
- هر لایه کانولوشنی شامل فیلترهای کانولوشنی و تابع فعالساز غیر خطی ReLU است.
- لایه های pooling برای اجرای max pooling مورد استفاده قرار می گیرند.
- سایز ورودی به علت وجود لایه های fully connected فیکس هست. و در بیشتر جاها اشاره شده است سایز برابر ۳*۲۲۴*۲۲۴ است و گاها به علت وجود برخی paddingها برابر ۳*۲۲۷*۲۲۷ است.
- به طور کلی این شبکه دارای ۶۰ میلیون پارامتر است.



شکل۵. معماری شبکه **AlexNet**

Size / Operation	Filter	Depth	Stride	Padding	Number of Parameters	Forward Computation
3 * 227 * 227						
Conv1 + Relu	11 * 11	96	4		(11*11*3 + 1) * 96=34944	(11*11*3 + 1) * 96 * 55 * 55=105705600
96 * 55 * 55						
Max Pooling	3 * 3		2			
96 * 27 * 27						
Norm						
Conv2 + Relu	5 * 5	256	1	2	(5 * 5 * 96 + 1) * 256=614656	(5 * 5 * 96 + 1) * 256 * 27 * 27=448084224
256 * 27 * 27						
Max Pooling	3 * 3		2			
256 * 13 * 13						
Norm						
Conv3 + Relu	3 * 3	384	1	1	(3 * 3 * 256 + 1) * 384=885120	(3 * 3 * 256 + 1) * 384 * 13 * 13=149585280
384 * 13 * 13						
Conv4 + Relu	3 * 3	384	1	1	(3 * 3 * 384 + 1) * 384=1327488	(3 * 3 * 384 + 1) * 384 * 13 * 13=224345472
384 * 13 * 13						
Conv5 + Relu	3 * 3	256	1	1	(3 * 3 * 384 + 1) * 256=884992	(3 * 3 * 384 + 1) * 256 * 13 * 13=149563648
256 * 13 * 13						
Max Pooling	3 * 3		2			
256 * 6 * 6						
Dropout (rate 0.5)						
FC6 + Relu					256 * 6 * 6 * 4096=37748736	256 * 6 * 6 * 4096=37748736
4096						
Dropout (rate 0.5)						
FC7 + Relu					4096 * 4096=16777216	4096 * 4096=16777216
4096						
FC8 + Relu					4096 * 1000=4096000	4096 * 1000=4096000
1000 classes						
Overall					62369152=62.3 million	1135906176=1.1 billion
Conv VS FC					Conv:3.7million (6%) , FC: 58.6 milion (94%)	Conv: 1.08 billion (95%) , FC: 58.6 milion (5%)

شکل۶. جزییات معماری شبکه **AlexNet**

جزییات مدل برنده مسابقه سال ۲۰۱۲ شامل:

- از تابع فعالساز ReLU استفاده شده بود.
- استفاده از لایه نرمالیزیشن که در حال حاضر رایج نیست
- سایز batch ۱۲۸ بوده است
- الگوریتم یادگیری SGD Momentum بوده است
- افزایش داده با استفاده از تکنیک هایی مثل flipping, cropping, color normalization و غیره صورت گرفته است.

مزایا:

- AlexNet اولین مدل CNN است که از GPU برای training استفاده کرده است و همین موضوع سبب رسیدن به مدل های سریعتر در آموزش شد.
- AlexNet یک شبکه عمیقتر با ۸ لایه است به این معنا که توانایی بهتری در استخراج ویژگی ها در مقایسه با مدل پیش از آن LeNet دارد، و همچنین برای زمان خودش با عکس های رنگی به خوبی کار میکرد.
- استفاده از تابع فعالساز Relu دو مزیت داشته است، آن باعث محدود شدن خروجی همانند سایز تابع فعالساز نمی شده است. بدین معنا که باعث از دست رفتن ویژگی های زیادی نمیشده است.
- خروجی های منفی جمع گرادیان ها (نه خود دیتاست) را نفی میکند، در واقع یعنی سرعت آموزش مدل را بهبود داده بدون آنکه همه perceptronها فعال شوند.

معایب:

- در مقایسه با سایر مدل ها این مدل دارای عمق خیلی کمتری است و در نتیجه برای یادگیری featureها از داده ها نیاز به تلاش بیشتری دارد.
- در مقایسه با مدل های جدیدتر، زمان بیشتری برای دستیابی به دقت بالاتر نیاز دارد.

AlexNet پیچیدگی لایه fully connected را با dropout کنترل میکند و برای افزایش بیشتر داده ها از image augmentation مثل flipping, clipping, color change استفاده میکند، که

باعث robust مدل و کاهش overfitting می شود. اگر عکس ورودی دارای سایز مناسب نباشد باید با resize, crop کردن ورودی آن را به این سایز برسانیم.

برای retrain کردن شبکه با استفاده از دیتاست جدید، سایز عکس ها را از طریق دستور resize, crop تنظیم کرده و سپس آنها را نرمالایز میکنیم.

۱-۳.

شبکه pretrained با استفاده از یک و چهارصد میلیون عکس از دیتابیس ImageNet آموزش دیده است و در نتیجه شبکه فیچرهای غنی از طیف وسیعی از عکس ها را یاد گرفته است، که می تواند عکس ها را در ۱۰۰۰ دسته طبقه بندی کند. اگر عکسی خارج از این دسته ها به شبکه داده شود، شبکه آن را به کلاسی اختصاص می دهد که به آن بیشترین شباهت را دارد به طور مثال اگر در شبکه کلاس گربه وجود نداشته ،ولی کلاس سگ وجود داشته باشد، با دادن عکس گربه به شبکه آن را به کلاس سگ اختصاص می دهد. برای راه حل آن می توان یک متریک جدید تعریف کرد که با ورود عکس جدید آن را تشخیص دهد. به این صورت که در فضای feature map نهایی بازنمایی به گونه ای باشد که داده هایی که شبیه به همدیگر هستند کنار هم قرار بگیرند و فاصله کمتری داشته باشند. در واقع هدف این است که فاصله تصاویر که از یک جنس هستند به هم نزدیک باشد و تصاویر که از یک جنس نیستند از هم دور شوند، با این کار اگر یک تصویر جدید وارد شود آن را تشخیص می دهد، چون فاصله آن از سایر تصاویر از یک threshold بیشتر است.

۱-۴.

در این سوال از ما خواسته شده تا از دیتاست Kaggle flower استفاده کنیم، ابتدا داده ها را از آدرس داده شده دانلود میکنیم و سپس با استفاده از دستور Imagfolder در کتابخانه پایتورچ آنها را لود میکنیم. این داده دارای ۵ کلاس می باشد. در ادامه با استفاده از دستور train_test_split ، ۸۰درصد داده ها را برای آموزش و ۲۰ درصد را برای تست جدا میکنیم.

شکل۷. چند نمونه از داده ها و لیبل آنها

۱-۵.

مطابق با مقاله برای پیاده سازی شبکه، ابتدا سایز ورودی را با استفاده از resize, crop تغییر داده سپس بعد از استفاده از ماژول ToTensor کتابخانه پایتورچ، نرمالایز میکنیم . شبکه AlexNet با وزن های آن را با دستور models.alexnet صدا میزنیم. در واقع با این کار ما مدل پایه را می سازیم که بر روی دیتاست ImageNet قبلا آموزش دیده است. برای retrain کردن آن، لایه های کانولوشنی مدل را فریز میکنیم و در نهایت یک طبقه بند اضافه میکنیم و آن را آموزش میدهیم. برای این کار می بایست با استفاده از دستور **requires_grad == False** از به روز رسانی وزن ها حین آموزش جلوگیری کنیم. در واقع ما وزن های تمامی لایه ها به جز لایه fully connected آخر را آپدیت نمیکنیم. ورودی لایه fully connected اخر تغییری نمیکند ولی خروجی آن برابر با کلاس دیتاست (پنج) خواهد بود.

جدول ۲. Net hyper parameters

Network Used Hyper Parameters	
Batch Size	10
Epochs	60
Input	224*224*3
Output	5
Normalization	Mean= [0.485, 0.456, 0.406] Std= [0.229, 0.224, 0.225]
Loss Function	Cross Entropy

Optimizer	SGD
Learning rate	0.001
momentum	0.9
Max Train Accuracy reached	97.4 %
Mx Val Accuracy reached	86.3 %

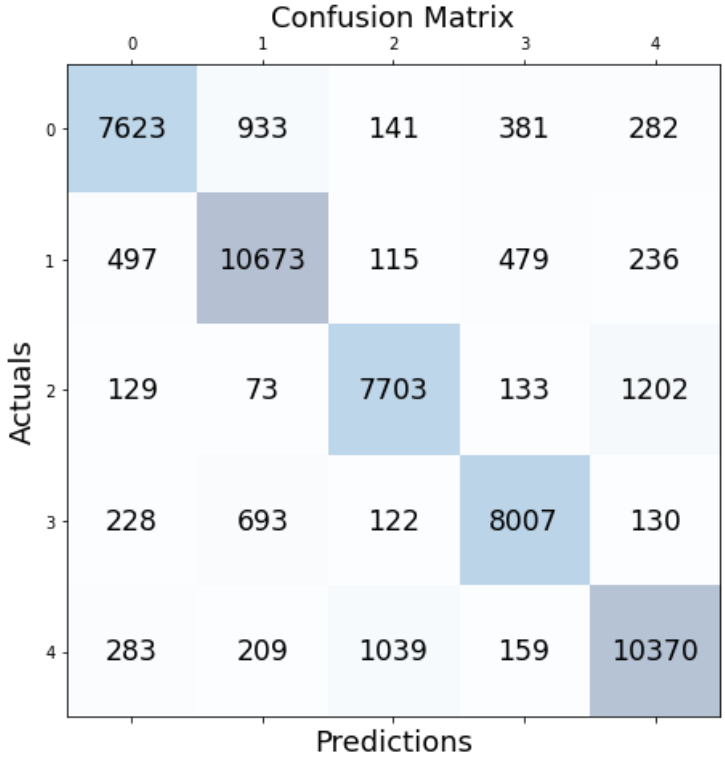
- نتایج بدست آمده از retrain کردن شبکه AlexNet در حالت feature extraction به شرح زیر است:

جدول ۳. نتایج شبکه برای داده های تست

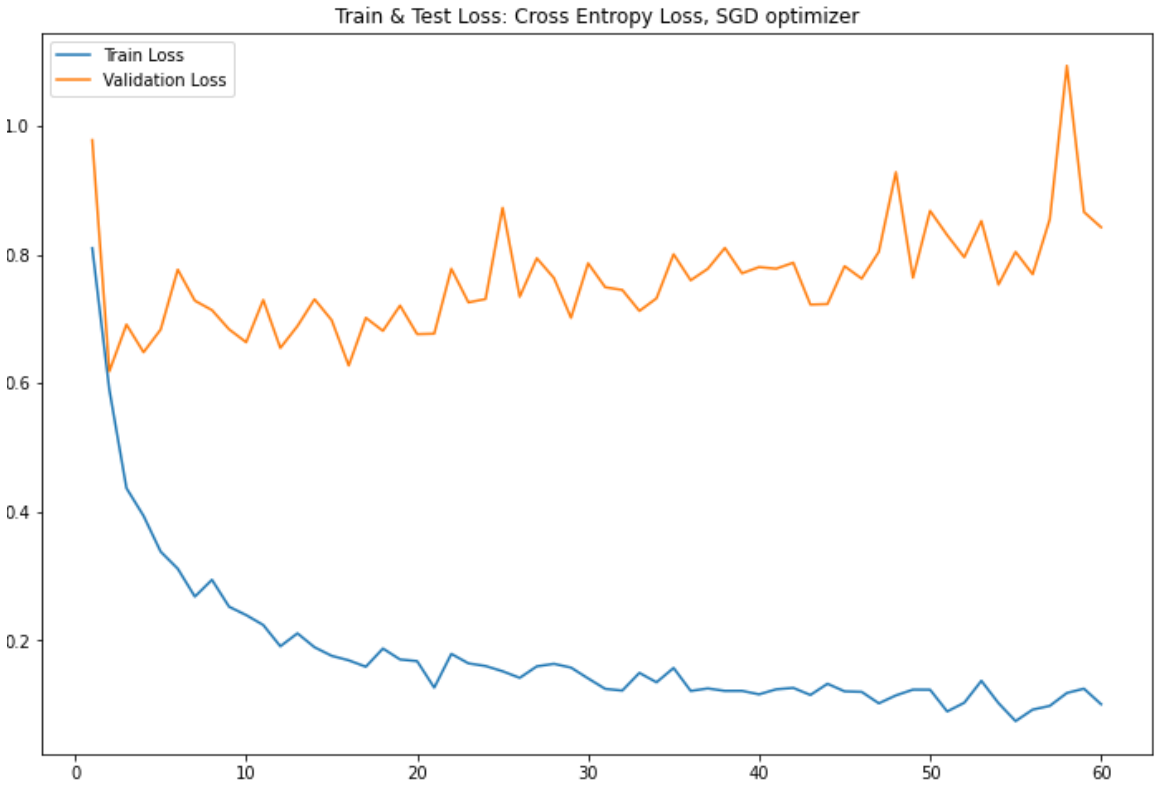
Test Accuracy	0.863
Test Loss	0.842
F1-score	0.862
Precision	0.865

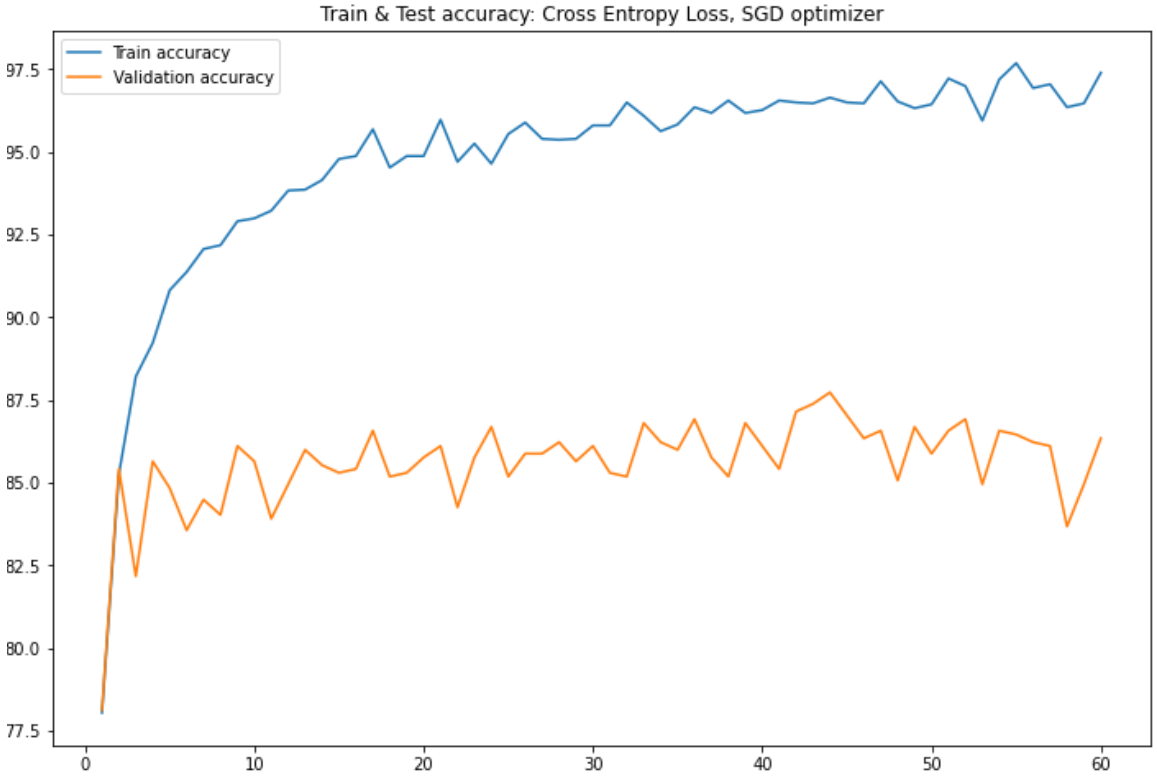
- دقت برای داده های تست به ازای هر کلاس برابر است با:

- Accuracy of daisy: 83.33 %
- Accuracy of dandelion: 89.50 %
- Accuracy of rose: 81.82 %
- Accuracy of sunflower: 86.93 %
- Accuracy of tulip: 88.56 %

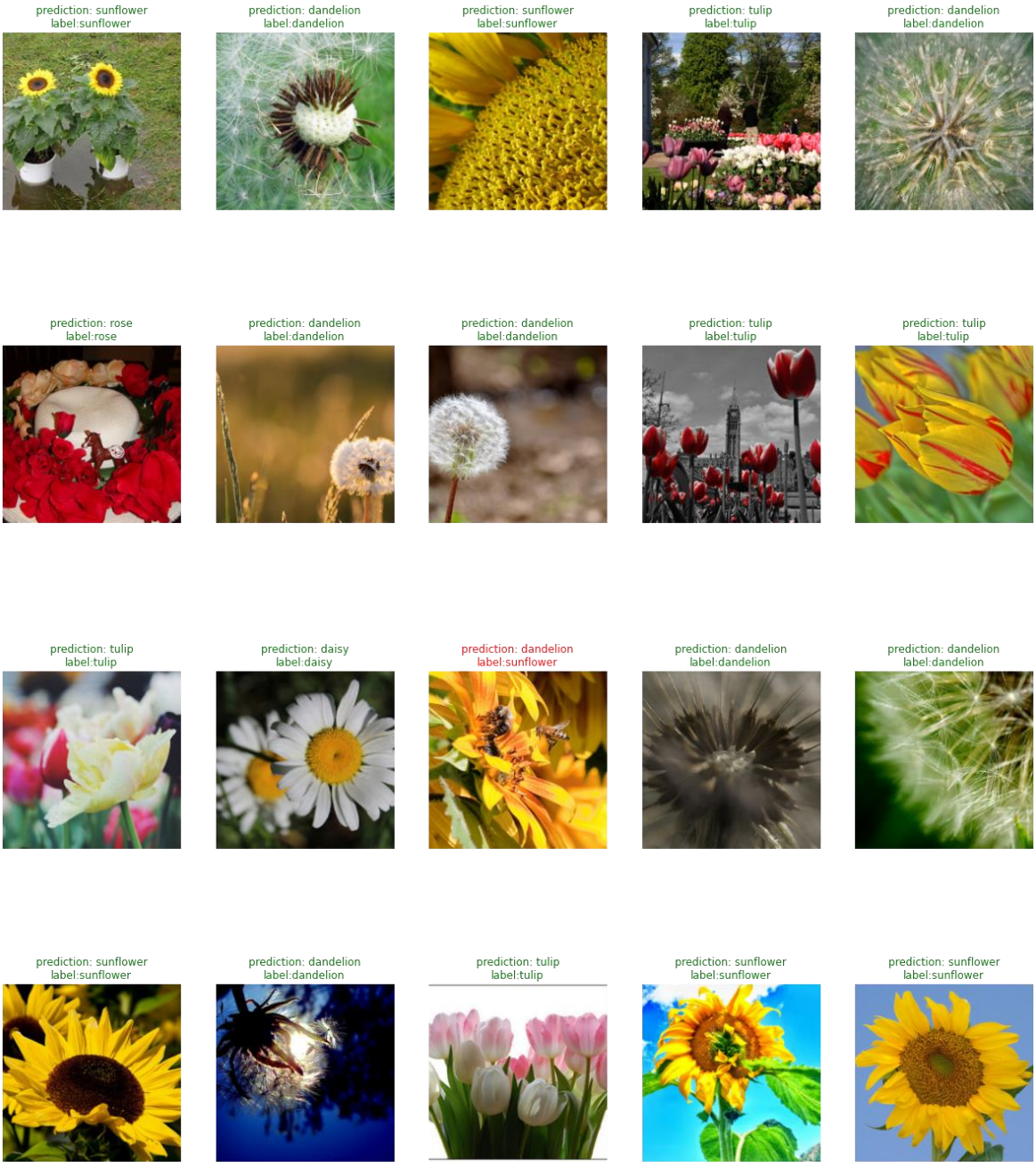


شکل۸. ماتریس طبقه بندی داده های تست





شکل۹. نمودار **accuracy** و **loss** مدل

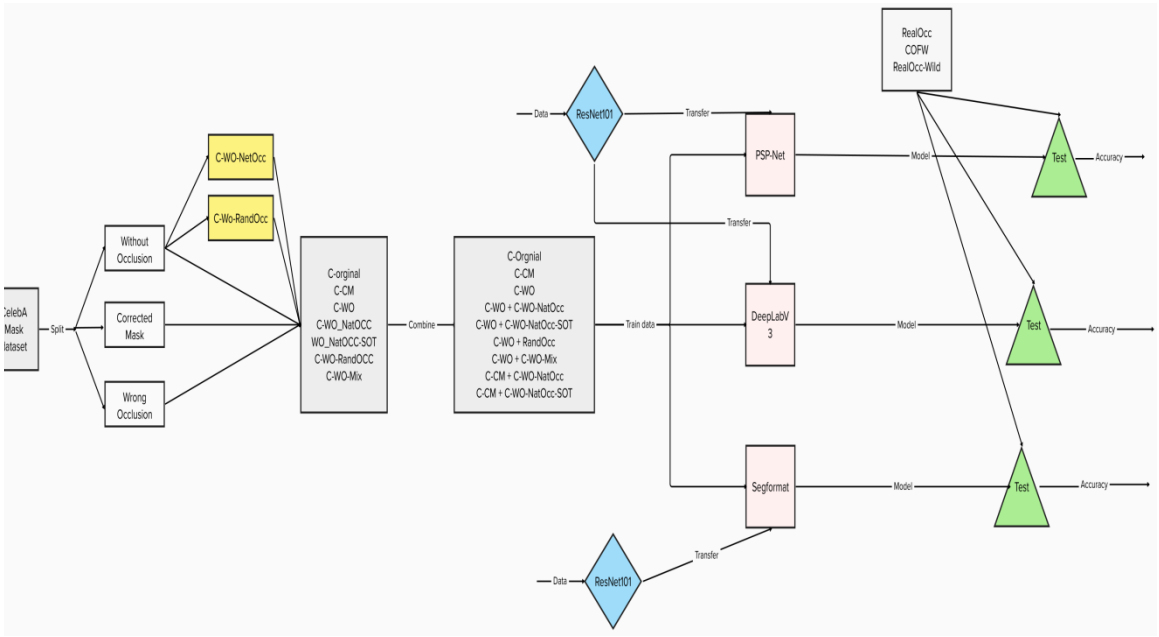


شکل ۱۰. چند نمونه از تصاویر طبقه بندی شده

پاسخ ۲ – آشنایی با تشخیص چهره مسدود شده

۲-۱. خلاصه ساختار شبکه

شکل کلی فرآیندی که در مقاله انجام شده کشیده شده است. بلاک‌های خاکستری داده‌ها هستند. بلاک‌های صورتی مدل‌های شبکه عصبی از جمله DeepLab هستند. لوزی آبی مدل‌هایی که برای pre-train استفاده شدند هستند. مثلث سبز برای تست کردن مدل‌ها با دیتاست‌های مختلف هست. مستطیل‌های زرد رنگ هم برای ایجاد Occlusionهای مصنوعی هستند.



شکل ۱۱. دیاگرام کلی شبکه

توضیح بیشتر: داده‌های train را از دیتای CelebAmask به اینصورت بدست می‌آوریم که ابتدا آن را به سه قسمت شامل داده‌های بدون مسدود سازی و با مسدود سازی درست لیبل زده شده و اشتباه لیبل زده شده تقسیم می‌کنیم. حالا روی داده بدون مسدود سازی شده با دو روش انسداد مصنوعی ایجاد می‌کنیم. اول طبیعی و دوم تصادفی.

که برای اسناد طبیعی از دیتاست‌های مختلف دیتا جمع کرده و روی دیتای اصلی اضافه می‌کنیم و برای طبیعی جلوه کردن عملیات Color Transfer, Augmentation, small random rotation, Image Harmonization را روی آن‌ها انجام می‌دهیم.

برای مسدودسازی تصادفی هم از دیتاست‌های مختلف دیتا را می‌گیریم و بعد از Occlusion Augmentation و Transparent Object Simulation را روی آن‌ها اعمال می‌کنیم.

بعد از این مرحله از چندین کلاس دیتا داریم. که با ترکیب آن‌ها با هم دیتاست‌های مختلف ایجاد می‌کنیم. که هر کدام از این دیتاست‌ها رو باید به مدل‌ها بدهیم و دقت را برای هر کدام بدست آوریم.

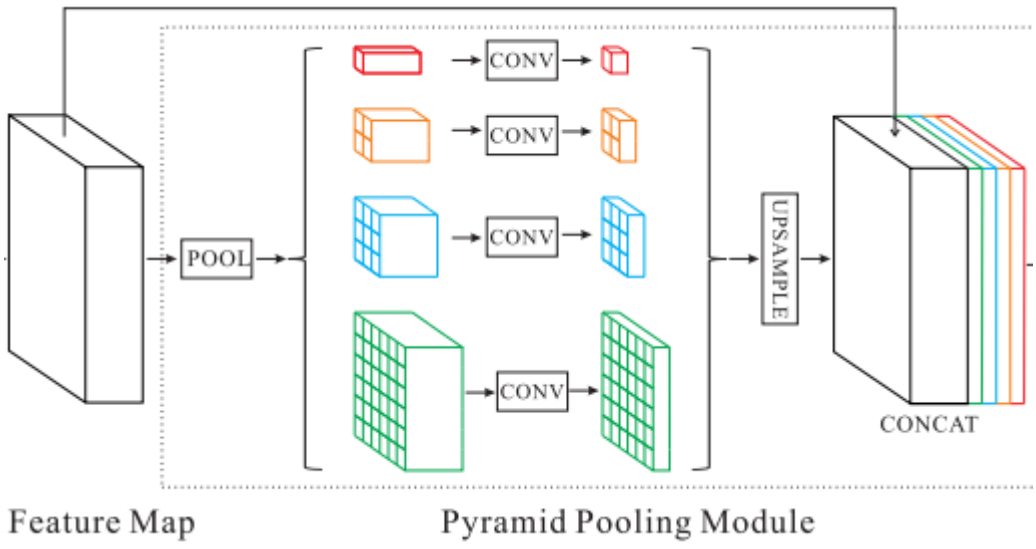
برای مدل‌های CNN محور یعنی PSPNet و DeepLabv3+ ابتدا از شبکه pre-train مدل ResNet-101 استفاده می‌کنیم سپس دیتای آموزش را برای یادگیری به مدل می‌دهیم. که از بهینه ساز SGD و خطای binary cross entropy استفاده کردیم.

برای مدل SegFormer از MIT-B5 برای Pre-train استفاده کرده است. که بهینه ساز آن هم Adam است.

در مرحله آخر هم دیتای تست را روی مدل‌ها اجرا می‌کنیم تا عملکرد هر کدوم را بدست آوریم.

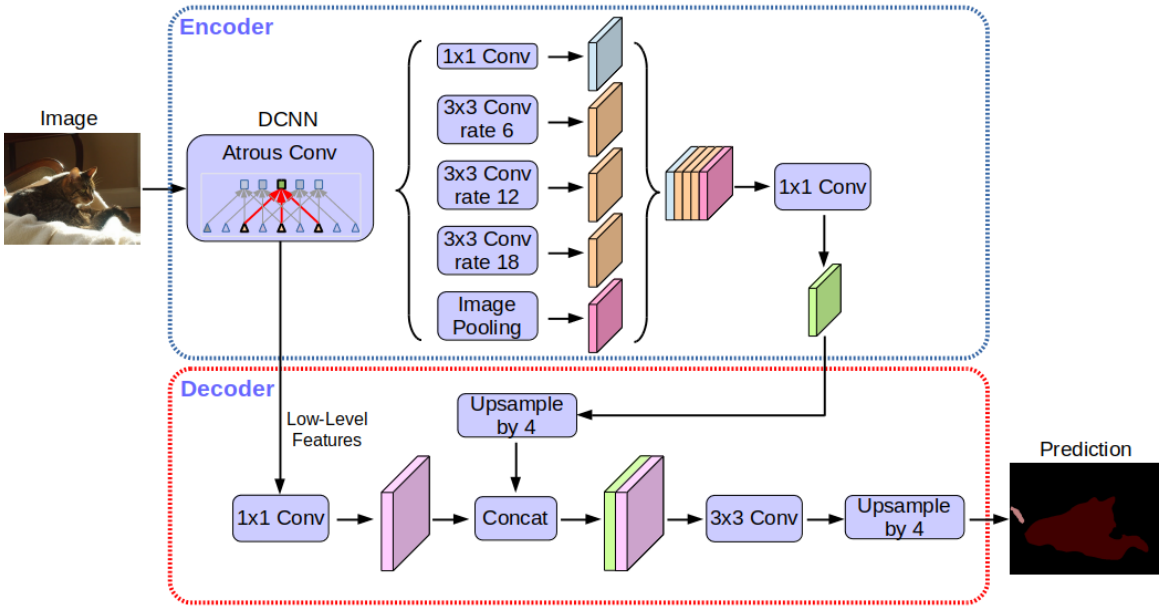
حال در مورد خود مدل‌ها کمی صحبت کنیم:

PSP-Net: ابتدا ورودی می‌دهیم سپس روی آن یک مدل CNN می‌زنیم بعد از اینکه یک Polling اعمال کردیم برای هر زیر ناحیه با سایز مختلف کانولوشن انجام می‌دهیم.مدل polling هرمی بخش اصلی این مدل است چون اجازه می‌دهد تا تصویر کلی تصویر را ثبت کند. و به آن کمک می‌کند پیکسل‌ها بر اساس اطلاعات کلی موجود در تصویر طبقه بندی کند.



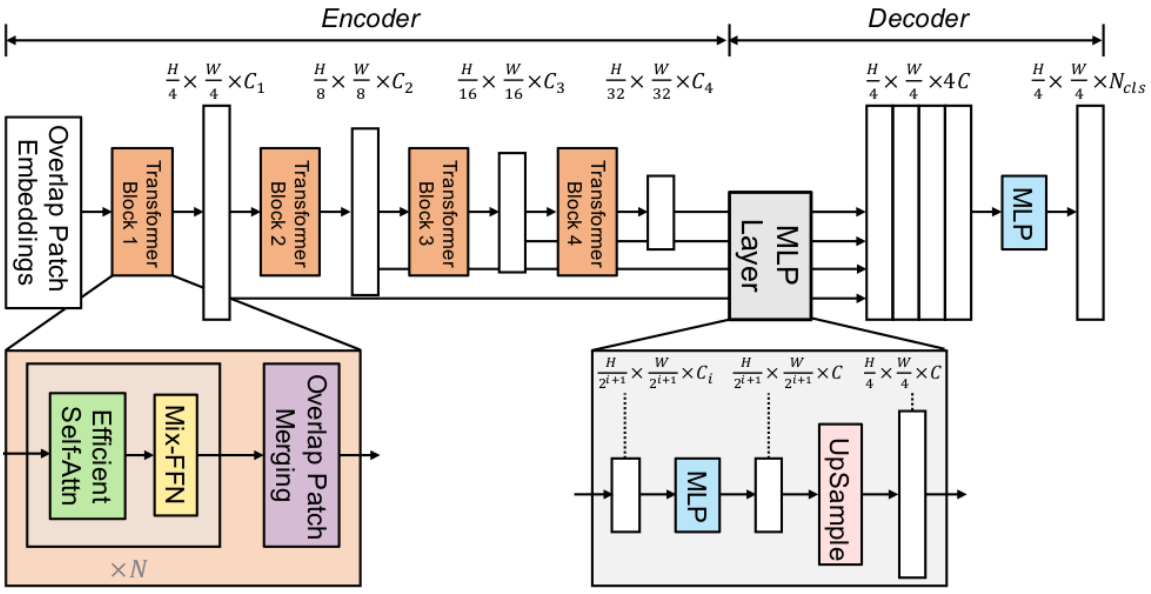
شگل ۱۲. معماری مدل PSPNet

مدل DeepLab: این معماری دارای سه نوآوری اصلی است: اول اینکه از کانولوشن با فیلتر های از نوع up sampled استفاده کرده که آنها را atrous convolution نامیده، این ویژگی برای کنترل رزولوشنی که ویژگی ها از آن تولید می شوند، طراحی شده است. دوم، طراحی atrous spatial pyramid pooling می باشد، تا بخش بندی پایدار تری در مقیاس های مختلف برای اشیا داشته باشد. سوم، ارتقا محلی سازی مرزبندی های اشیا با ترکیب روش های DCNN و مدل های گرافیکی احتمالی می باشد.



شکل ۱۳. معماری شبکه DeepLabv3+

مدل SegFormer: این معماری مبتی بر معماری Transformer هست. و ساختار encoder-decoder دارد.این باعث می‌شود که ویژگی‌های multiscale را در خروجی دهد. بنابراین از interpolation کدهای موضعی جلوگیری می‌کند که باعث کاهش عملکرد در زمانی که وضوح تست و آموزش متفاوت است، می‌شود. بر خلاف سایر decoder ها در این معماری از دیکودر ساده MLPاستفاده می‌کند و بنابراین توجه محلی و توجه سراسری را برای ارائه قدرتمندتر ترکیب می‌کند. معماری آن در شکل ۴ آمده است.



شکل ۱۴. معماری شبکه Segformer

۲-۲. تفاوت بین Occlusionها در دقت شبکه

در این مقاله از NatOcc و RandOcc برای مسدود کردن چهره استفاده می‌کنیم.

NatOcc : سعی می‌کنم مسدود کردن چهره طبیعی به نظر برسد.

RandOcc: یک روش کلی‌تر با پوشاندن چهره توسط اشکال تصادفی، شفافیت تصادفی.

روش سوم هم ترکیب این دو روش است.

عملکرد هر کدام در دقت:

توجه: از معیار MIoU برای ارزیابی عملکرد semantic segmentation استفاده می‌شود.

متد NatOcc دقتش از مجموعه داده بدون انسداد(C-WO) به طور قابل توجهی بیشتر است و نسبت به مجموعه داده مسدود شده در دنیای واقعی(C-CM) دقت بیشتر یا برابر دارد.

روش RandOcc هم از مجموعه داده بدون انسداد(C-WO) بیشتر است و دقتش نزدیک دقت مجموعه داده مسدود شده در دنیای واقعی(C-CM) است.

در اکثر موارد دقت روش NatOcc بیشتر از RandOcc است. به خصوص در تست با دیتاست RealOCC-Wild . دلیلشم این است که روش NatOcc به رنگ، بافت و لبه‌ها توجه می‌شود تا نزدیک دیتای واقعی باشد. برای همین برای دیتای واقعی بهتر عمل می‌کند.

گاهی اوقات Mix این دو روش باعث بهبود عملکرد نسبت به اعمال روش‌ها به صورت تکی می‌شود. در واقع میشه گفت که RandOcc مکمل روش NatOcc برای تشخیص اشیا است.

۲-۳. لزوم کلاس بندی داده‌ها

برداشت ۱: بله. اگر بخواهیم تاثیر Occlude های مختلف را ببینیم نیاز است ترکیب‌های مختلف دیتاست آموزشی را در نظر بگیریم و با هم مقایسه کنیم که آموزش روی کدام دیتاست عملکرد بهتری را در خروجی مدل به ما می‌دهد.

مثلا ترکیب C-CM + C-WO-NatOcc یکی از بهترین دیتا برای آموزش شبکه است چرا که برای همه‌ی داده‌های تست یا بهترین دقت یا نزدیک بهترین دقت را داشته است. از روی این متوجه می‌شویم که اولا نگه داشتن ماسک‌های درست از دیتاست CelebAMask-HQ-O معمولا بیشتر کمک می‌کند تا دور انداختن آن‌ها. دوما مسدود سازی کردن عکس‌ها با روش NatOcc و اضافه کردن آن به C-CM عملکرد بهتری از C-CM به تنهایی می‌دهد.

یا مثلا دیتای C-Orginal از همه تقریبا بدتر عمل می‌کند. که از خود این می‌توان مواردی را استنباط کرد. مثلا نگه داشتن ماسک اورجینال نسبت به اینکه ماسک‌های غلط را تصحیح کنیم عملکرد بدتری دارد. با مقایسه آن با دقت C-WO متوجه میشیم که حتی حذف دیتای ماسک زده شده دقت را آنچنان تغییر نمی‌دهد.

برداشت ۲: اگه منظور کلاس بندی جدول ۱ است یعنی Face, Background باز هم جواب بله است. واجب است که یک تعریف از کلاس‌ها داشته باشیم

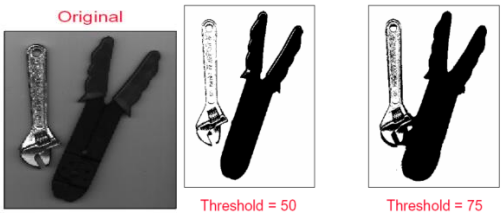
۲-۴. شبکه ساده‌تر در صورت تفاوت intensity چهره‌ها با Occlusion

زمانی که intensity بین ناحیه‌های مختلف متفاوت باشد ولی مقدار intensity درون ناحیه‌ای مشابه باشد آنگاه می‌توان از Threshold-Based Segmentation استفاده کرد که ساده‌ترین آن‌ها Binary Thresholding است. در این روش هر پیکسل با threshold مشخص شده مقایسه شده اگر کمتر بود مقدار 0 می‌گیرد و اگر بیشتر بود مقدار 1 می‌گیرد.

$$g(x,y) = \begin{cases} 0, & f(x,y) < Threshold \\ 1, & f(x,y) \geq Threshold \end{cases}$$

انتخاب مقدار threshold در عملکرد تشخیص اشیا اهمیت بالایی دارد. به شکل ۲ که انبردست تیره و آچار فرانسه را به عنوان اشیا تشخیص بده توجه کنید. در این شکل تفاوت مقدار intensity بین

اشیا زیاد است و درعین حال intensity داخل هر شی تقریبا یکنواخت هست. پس شرایط لازم را دارد. و اهمیت مقدار threshold در شکل وسط و سمت چپ مشخص می‌شود. در جایی که مقدار آستانه مناسب انتخاب شده یعنی ۵۰ دو شی درست تشخیص داده شدند ولی زمانی که مقدار آستانه بیشتر از حد مناسب انتخاب شده مرزهای دو شی کمی خطا دارد.



شکل ۱۵. سگمنت کردن با آستانه‌های مختلف با روش
binary-thresholding

۵-۲. مقایسه بین کارایی PSPNet و DeepLab

دقت هر دو تقریبا در تمام دیتاست‌ها تقریبا برابر است. البته اگر دقیق‌تر نگاه کنیم DeepLab در دیتاست‌هایی که دیتای مسدود شده به صورت مصنوعی وجود دارد در اکثر موارد کمی بهتر است. البته این تفاوت به اندازه کافی معنادار نیست.

اگر به صورت کلی(جدا از این مقاله) نگاه کنیم در مقاله مربوط به DeepLab که عملکرد آن را با روش‌های State-of-the-art مقایسه کرده از جمله PSPNet. که در دیتاست PASCAL VOC 2012 مقدار MIoU برای DeepLab عدد 86.9 ثبت شده و برای روش PSPNet مقدار 85.4 به ثبت رسیده است. پس مقاله ادعا می‌کند که DeepLabv3+ نسبت به PSPNet برتری دارد. اولین بار هم در چالش ILSVRC 2016 Scene Parsing این برتری دیده شد.

پاسخ ۳. تشخیص بلادرنگ اشیاء

YOLOv6 از بهترین و کارآمدترین ورژن های مدل YOLO می باشد که نتایج چشمگیر بدست آورده و همینطور در زمینه detection دارای سرعت و دقت بسیار خوب است. مدل YOLOv6 در سال ۲۰۲۲ منتشر شد و دقیق ترین مدل برای object detection شناخته شد. در واقع این مدل با هدف استفاده در کاربردهای صنعتی و حل مشکلات عملی به وجود آمده در حین برنامه های صنعتی منتشر شد.

این مدل یک single-stage چارچوب تشخیص اشیاء با طراحی سخت افزاری کارآمد و همچنین سرعت استنتاج کمتر و دقت بالاتر نسبت به ورژن های پیشین می باشد که بیشتر بر روی کاربرد صنعتی متمرکز است.

۳-۲،۱

مراحل شخصی سازی یک مجموعه جدید روی YOLOv6 شامل سه گام است:

۱- در ابتدا repository مربوط به مدل را clone می کنیم

۲- در ادامه ماژول های مورد نیاز برای اجرای مدل را نصب میکنیم

۳- در آخر وزن مدل آموزش دیده را دانلود میکنیم و بر روی داده های جدید اجرا میکنیم.

مراحل بالا به همراه جزییات و کد آنها در ادامه آورده شده است:

- آماده سازی دیتاست:

داده های train, validation و test باید به صورت جداگانه در دو فولدر images و labels ذخیره شوند. مطابق هر عکس یک فایل لیبل وجود دارد که فایل لیبل یک فایل تکست می باشد که شامل اطلاعاتی نظیر سایز و کادر اشیا موجود در تصویر (هر ردیف مربوط به یک تصویر است) می باشد. به طور مثال فرمت فایل لیبل به صورت زیر می باشد:

```
# class_id center_x center_y bbox_width bbox_height
0 0.300926 0.617063 0.601852 0.765873
1 0.575 0.319531 0.4 0.551562
```

- یک فایل به اسم dataset.yaml که حاوی ادرس داده های آموزش و validation و تست و همچنین اینکه از داده coco استفاده نمیکنیم و نام و تعداد کلاس ها می باشد.

کد مربوط به این فایل در ادامه آورده شده است:

```
with YOLOv6_DIR
train: /content/drive/MyDrive/data/chess_dataset/images/train # train
images
val: /content/drive/MyDrive/data/chess_dataset/images/val # val images
```

```
test: /content/drive/MyDrive/data/chess_dataset/images/test # test images
(optional)

# whether it is coco dataset, only coco dataset should be set to True.
is_coco: False
# Classes
nc: 13  # number of classes
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']  # class names
```

در دستور بالا آرگمان ها به شرح زیر است:

- train: مسیر ذخیره شدن داده های آموزش
- val: مسیر ذخیره شدن داده های validation
- test: مسیر ذخیره شدن داده های تست
- is_coco: آیا از دیتاست coco استفاده می شود یا خیر (از جایی که ما از دیتاست دیگری استفاده میکنیم می بایست آن را false قرار دهیم)
- nc: تعداد کلاس ها
- names: اسم کلاس ها

داده های ما شامل ۱۳ کلاس می باشد.

- یک فولدر به اسم weights ساخته و فایل مربوط به وزن های YOLOv6 را در آن اپلود میکنیم.
- دستور train را با بچ سایز ۱۶ و ۱۰۰ ایپاک اجرا میکنیم.

```
!python tools/train.py --batch 16 --conf configs/yolov6s_finetune.py --
data-path dataset.yaml --device 0 --epochs 100 --eval-interval 2
```

با اجرای دستور بالا نتایج آموزش مدل و وزن ها در فولدر exp با آدرس runs/train/exp ذخیره می شود.

- در ادامه یک لیست می سازیم و ادرس تمامی عکس های تست را در آن قرار می دهیم و سپس با استفاده از یک حلقه تمامی آنها را به مدل داده و پیش بینی آن را ذخیره میکنیم.

کد مربوط به پیش بینی مدل:

```
!list=ls /content/drive/MyDrive/data/chess_dataset/images/test/*.jpg
!echo $list
```

```
!for FILE in /content/drive/MyDrive/data/chess_dataset/images/test/*.jpg;
do python tools/infer.py --weights runs/train/exp2/weights/best_ckpt.pt --
source $FILE --yaml dataset.yaml --device 0;  done
```

در دستور بالا آرگمان ها به شرح زیر است:

- source--: آدرس تصاویر را نگه می دارد. این مسیر می تواند متعلق به یک فیلم یا مسیر یک دایرکتوری شامل چندین عکس و فیلم باشد.
- weights--: آدرس مسیر متعلق به فایل وزن ها را نگه می دارد.
- device--: در واقع computation device می باشد. می توانیم از cpu استفاده کنیم و یا یک عدد بین ۰ تا ۳ برای نشان دادن اینکه از کدام GPU استفاده میکنیم قرار دهیم.

در حقیقت با اجرای دستور بالا تصاویر تست به مدل داده می شود و مدل مهره های شطرنج موجود در عکس را پیش بینی میکند و فایل segment شده آنها را در فولدر exp به آدرس runs/inference/exp ذخیره میکند.

۳-۳.

همانطور که در تصویر زیر دیده می شود مدل با دقت بالایی (بالاتر از ۸۰ درصد) مهره های شطرنج را segment کرده است. در واقع همانطور که پیشتر گفته شد، از شبکه YOLOv6 انتظار میرفت که دارای زمان اجرای مناسب و دقت بالا بوده و در نتیجه در کاربرد صنعتی عمکرد خوبی داشته باشد.

شکل ۱۶. چند نمونه از تصاویر segment شده مهره های شطرنج با برچسب دقت