به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

شبکه های عصبی و یادگیری عمیق

مینی پروژه اول

نام و نام خانوادگی	محمدحسین صرفی-مهرگان گیلانی نژاد
شماره دانشجویی	810696262-810696290
تاریخ ارسال گزارش	1399/08/30

فهرست گزارش سوالات

سوال 1 – CNN

در این قسمت به شبکه ی MLP طراحی شده در تمرین دوم لایه های convolution اضافه می شود. در تمرین دوم نتایج نهایی زیر به دست آمد. معماری شبکه، معیار های اندازه گیری شده(نرخ یادگیری، optimizer و ...) در ادامه ارائه شده اند.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 3072)              0
_________________________________________________________________
dense (Dense)                (None, 1000)              3073000
_________________________________________________________________
dropout (Dropout)            (None, 1000)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               512512
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5130
=================================================================
Total params: 3,590,642
Trainable params: 3,590,642
Non-trainable params: 0
_________________________________________________________________
```

Figure 1: Model Summary in Homework 2

Table 1: Characteristics of the Model in Homework 2

Activation Function	ReLu
Batch size	64
Number of epochs	50
Optimizer and learning rate	Adam(learning rate = 0.001)
Loss Function	Categorical Crossentropy

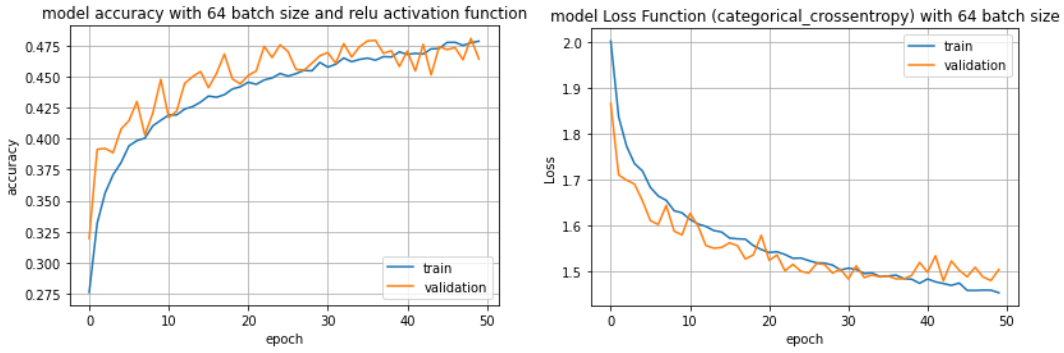


Figure 2: Model loss and accuracy in Homework 2

همان طور که در شکل بالا دیده می شود شبکه MLP توانست دقتی نزدیک به 47 درصد را به دست آورد. حل با اضافه کردن لایه های جدید به شبکه ی ذکر شده می توان دقت شبکه را بالا برد.

الف

به بهترین شبکه بدست آمده در سوال دوم تمرین دوم ، لایه های کانولوشنی را اضافه نمائید و شبکه را پیاده سازی نمائید. نتیجه بدست آمده را از نظر دقت و خطا با معماری MLP مقایسه نمائید.

در این مرحله با حذف لایه های Dropout از شبکه ی قبل و اضافه کردن 6 لایه Conv2D شبکه را آموزش میدهیم. معماری شبکه در ادامه ارائه شده است.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 32)        896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 64)        18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 32, 32, 64)        36928
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 32, 32, 128)       73856
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 32, 32, 128)       147584
_________________________________________________________________
flatten (Flatten)            (None, 131072)            0
_________________________________________________________________
dense (Dense)                (None, 1000)              131073000
_________________________________________________________________
dense_1 (Dense)              (None, 512)               512512
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5130
=================================================================
Total params: 131,877,650
Trainable params: 131,877,650
Non-trainable params: 0
_________________________________________________________________
```

Figure 3: Model summary

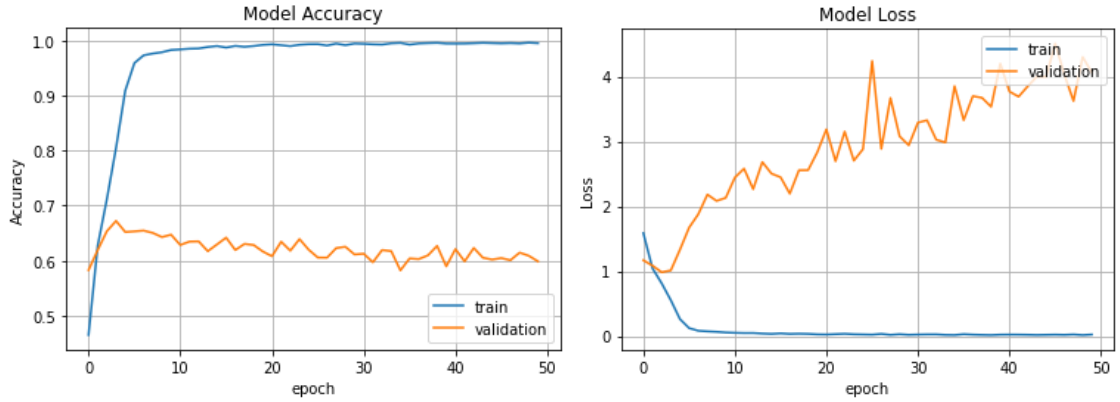


Figure 4: Model loss and accuracy

Test Loss	Test Accuracy
4.182	0.596

همان طور که دیده می شود شبکه به خوبی آموزش داده نشده است و دقت پایینی دارد که در قسمت های بعدی تصحیح خواهد شد.

ب

لایه های Batch normalization و Pooling را توضیح دهید و سپس این لایه ها را به توپولوژی شبکه اضافه نمائید و شبکه را پیاده سازی نمائید. نتیجه بدست آمده را از نظر دقت و خطا با معماری قسمت (الف) مقایسه نمائید.

با اضافه کردن لایه های MaxPooling و Batch normalization معماری شبکه و نتایج به صورت زیر در می آیند:

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_18 (Conv2D)           (None, 32, 32, 32)        896
_________________________________________________________________
batch_normalization (BatchNo (None, 32, 32, 32)        128
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 32, 32, 32)        9248
_________________________________________________________________
batch_normalization_1 (Batch (None, 32, 32, 32)        128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 16, 16, 64)        18496
_________________________________________________________________
batch_normalization_2 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 16, 16, 64)        36928
_________________________________________________________________
batch_normalization_3 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 8, 8, 128)         73856
_________________________________________________________________
batch_normalization_4 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 8, 8, 128)         147584
_________________________________________________________________
batch_normalization_5 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_15 (Dense)             (None, 1000)              2049000
_________________________________________________________________
batch_normalization_6 (Batch (None, 1000)              4000
_________________________________________________________________
dense_16 (Dense)             (None, 512)               512512
_________________________________________________________________
batch_normalization_7 (Batch (None, 512)               2048
_________________________________________________________________
dense_17 (Dense)             (None, 10)                5130
=================================================================
Total params: 2,861,490
Trainable params: 2,857,570
Non-trainable params: 3,920
```

Figure 5: Model summary

خطا و دقت شبکه در این قسمت به شرح زیر است. خطا و دقت داده های test نیز در ادامه ارائه شده است.

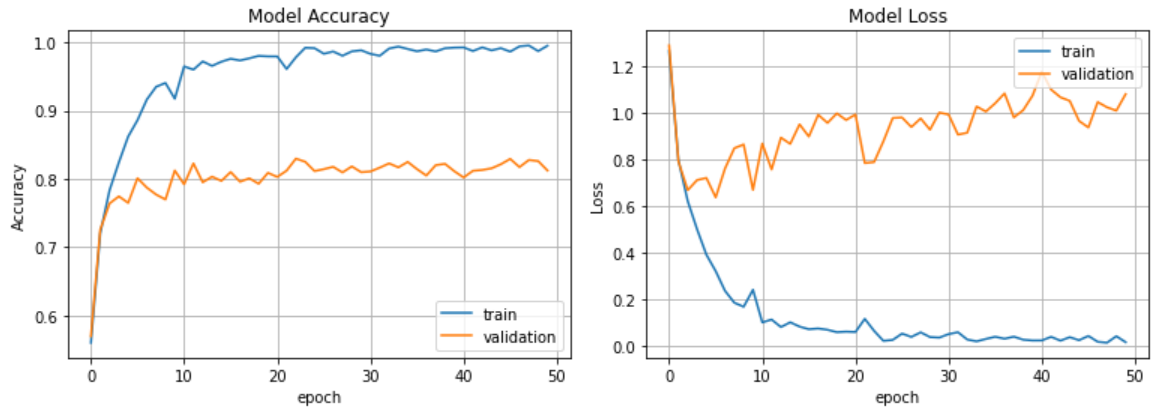


Figure 6: Model loss and accuracy

Table 2: Test data Loss and accuracy

Test Loss	Test Accuracy
1.21	0.811

همان طور که دیده می شود دقت داده های آموزش نزدیک 100 درصد است در صورتی که دقت داده های validation و test نزدیک 80 درصد است. این نشان دهنده ی overfit شدن شبکه است که در قسمت بعد تصحیح خواهد شد. در مقایسه با قسمت الف دیده می شود که دقت مدل دقت شبکه افزایش قابل توجهی داشته و خطا نیز کاهش یافته است.

ج

به معماری شبکه بدست آمده در قسمت (ب) اکنو ن Dropout را نیز اضافه نمائید و تاثیر آن را بررسی نمائید. چرا از Dropout در معماری شبکه عصبی استفاده می نمائیم ؟

معماری شبکه بعد از اضافه کردن لایه های Dropout و همچنین دقت و خطا داده ها در ادامه ارائه شده است.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_24 (Conv2D)           (None, 32, 32, 32)        896
_________________________________________________________________
batch_normalization_8 (Batch (None, 32, 32, 32)        128
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 32, 32, 32)        9248
_________________________________________________________________
batch_normalization_9 (Batch (None, 32, 32, 32)        128
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_26 (Conv2D)           (None, 16, 16, 64)        18496
_________________________________________________________________
batch_normalization_10 (Batc (None, 16, 16, 64)        256
_________________________________________________________________
conv2d_27 (Conv2D)           (None, 16, 16, 64)        36928
_________________________________________________________________
batch_normalization_11 (Batc (None, 16, 16, 64)        256
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_28 (Conv2D)           (None, 8, 8, 128)         73856
_________________________________________________________________
batch_normalization_12 (Batc (None, 8, 8, 128)         512
_________________________________________________________________
conv2d_29 (Conv2D)           (None, 8, 8, 128)         147584
_________________________________________________________________
batch_normalization_13 (Batc (None, 8, 8, 128)         512
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
flatten_4 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_18 (Dense)             (None, 1000)              2049000
_________________________________________________________________
batch_normalization_14 (Batc (None, 1000)              4000
_________________________________________________________________
dropout_3 (Dropout)          (None, 1000)              0
_________________________________________________________________
dense_19 (Dense)             (None, 512)               512512
_________________________________________________________________
batch_normalization_15 (Batc (None, 512)               2048
_________________________________________________________________
dropout_4 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_20 (Dense)             (None, 10)                5130
=================================================================
Total params: 2,861,490
Trainable params: 2,857,570
Non-trainable params: 3,920
_________________________________________________________________
```

Figure 7: Final Model Summary

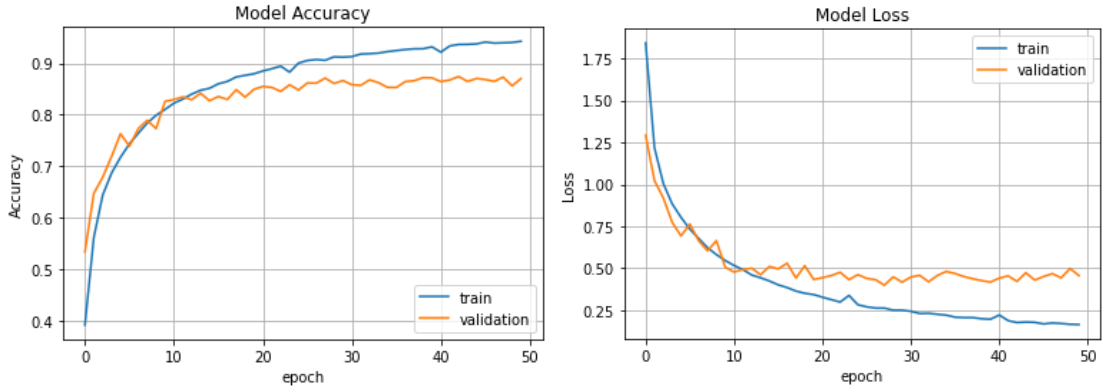


Figure 8: Model loss and accuracy

همان طور که دیده می شود با اضافه کردن لایه های Dropout مقدار overfit شبکه بسیار کاهش یافته است و دقت داده های validation به دقت داده های آموزش نزدیک شده است.

Test Loss	Test Accuracy
0.473	0.868

همان طور که دیده می شود دقت داده های تست نسبت به حالت (ب) افزایش یافته است همچنین مقدار خطای داده های تست کاهش یافته است که نشان دهنده ی بهبود شبکه است.

عملکرد لایه های Dropout: لایه ی Dropout به صورت رندم خروجی نورون ها را با فرکانس rate(که بین صفر و یک تعیین میشود) برابر صفر قرار می دهد و در واقع این نورون هارا از فرایند آموزش حذف می کند و به این صورت از overfitting مدل جلوگیری می شود. در این قسمت rate لایه های Dropout برابر با 0.2، 0.3، 0.4 و 0.5 قرار داده شده است.

*بهبود بیشتر overfitting با تعویض optimizer:

در سوال 2 تمرین 2 بین optimizerهای SGD و Adam مقایسه انجام شد و در نهایت Adam انتخاب شد اما در این سوال دیده می شود که این optimizer عملکرد بهینه ندارد. در نتیجه می توان با تنظیم کردن optimizer با شرایط موجود overfit از بین برد. بعد از تنظیم optimizer به SGD به نتایج زیر خواهیم رسید.

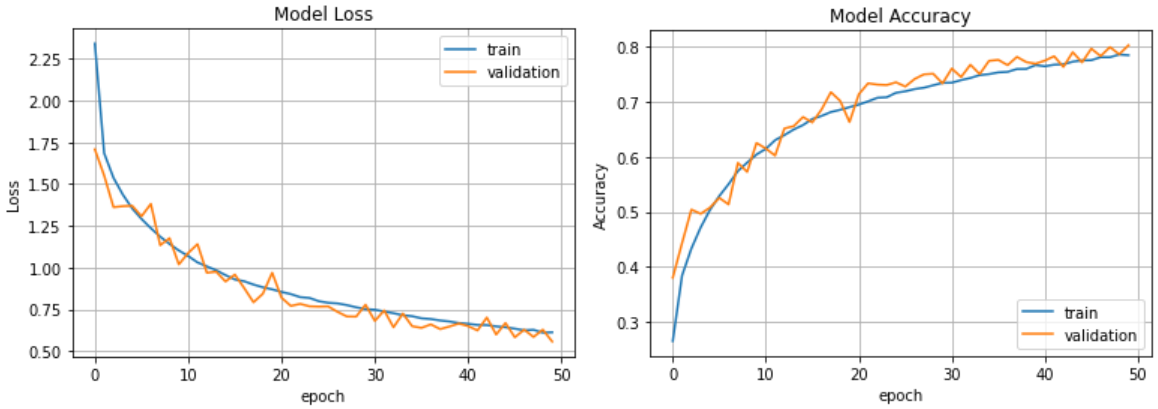


Figure 9: Model loss and accuracy with SGD optimizer

Test Loss	Test Accuracy
0.59	0.796

همان طور که دیده می شود وضعیت overfitting نسبت به حالت قبل بهبود یافته است.

د

توقف زود هنگام در شبکه های عصبی به چه معناست ؟ چه معیارهایی در این توقف زود هنگام استفاده می شوند؟ یک نمونه از آن را پیاده سازی نمائید.

در شبکه های عصبی اگر تعداد اپاک ها زیاد باشد ممکن است شبکه overfit شود و اگر تعداد اپاک ها بسیار کم باشد شبکه به خوبی آموزش دیده نمی شود. در keras دستور هایی تعیین شده تا شبکه را بر اساس معیار هایی که تعیین میشود نظارت کنند و اگر در این معیار ها بعد از چند اپاک بهبود داده نشود، آموزش شبکه را متوقف کنند. در اینجا از دستور زیر استفاده شده است:

```
EarlyStopping(monitor='val_loss', mode='min', verbose=1,patience=10)
```

بعد از اجرای کد پیام `Epoch 00035: early stopping` ظاهر می شود یعنی با 35 اپاک آموزش شبکه به نتایج زیر رسیده است که به نتایج نهایی قسمت ج که در 50 اپاک آموزش داده شده بود نزدیک است.

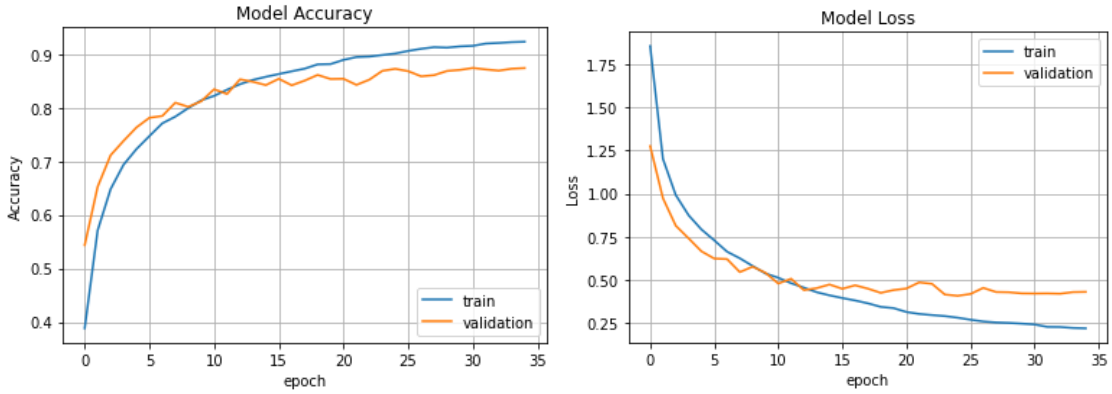


Figure 10: Model loss and accuracy after implementing early stopping

Test Loss	Test Accuracy
0.473	0.860

با مقایسه ی نتایج بالا با نتایج MLP دیده می شود که شبکه دقت بسیار بالاتری(تقریبا 2 برابر) و خطای پایین تری دارد.

سوال ۲ –Deep Learning

شماره دانشجویی اعضای گروه : 810696290 و 810696262 ، بنابراین در این سوال به شبکه inception پاسخ داده شده است.

الف

معماری شبکه

شبکه inception یکی از شبکه cnn برای دسته بندی تصاویر می باشد. به صورت کلی ساختار شبکه های CNN به صورت زیر می باشد

تعدادی لایه کانولوشنی به همراه لایه های Maxpoolin, batch normalization

تعدادی لایه fully connected در انتهای شبکه

چنین ساختاری در دیتاست های کوچک همانند cifar10 به خوبی جوابگو می باشد اما در دیتاست های بزرگ تر همانند imagenet چنین شبکه هایی دچار مشکل اورفیت می شوند.

یکی از شبکه های معروف جهت دسته بندی تصاویر و تشخیص اشیا شبکه R-CNN می باشد. در این شبکه object detection در دو مرحله صورت می گیرد

1. استفاده از ویژگی هایی همانند رنگ و ... برای تشخیص وجود شی در یک ناحیه
2. استفاده از شبکه های CNN برای تشخیص کلاس شی

سیتسم کاری شبکه inception برای تشخیص اشیا نیز بسیار شبیه به همین سیستم می باشد اما در هر 2 مرحله بهبود هایی صورت گرفته است که در ادامه با جزییات بررسی می شوند.یکی از راهکار هایی که برای بهبود شبکه های CNN پیشنهاد می شوند افزایش تعداد لایه ها و پارامتر ها می باشد. با این روش عمق شبکه بیشتر می شود اما احتمال overfit شدن شبکه بر روی داده های train وجود دارد. برای حل این مشکل دیتاست باید بزرگ تر شود، که خود کار سخت و زمانبری است . مشکل دیگر در این حالت هزینه سنگین محاسباتی می باشد.

در لایه های کانولوشنی باید اندازه kernel مشخص باشد. در صورتی که این اندازه بزرگ باشد، شبکه به دنبال اطلاعاتی می گردد که به صورت globalپخش شده اند. اگر سایز kernel کوچک باشد، اطلاعات با توزیع local مورد توجه شبکه قرار می گیرند. بنابراین در عکس هایی که شی در موقعیت های مختلف قرار گرفته است، انتخاب سایز kernel مهم می باشد. در شبکه inception برای حل این مشکل در هر

مرحله 3 فیلتر مختلف (با ابعاد 1*1 و 3*3 و 5*5) در نظر گرفته شده است تا حالت های مختلف پوشش داده شوند. در کنار این 3 فیلتر ، max pooling نیز قرار دارد. نهایتا نتایج به دست آمده از فیلتر ها و maxpooling در انتها با هم contact می شوند و نتیجه به دست آمده به عنوان ورودی به بلاک بعدی داده می شود.

بنابراین ایده اصلی شبکه های inception آن است تا یک ساختار بهینه local پیدا شود و سپس این ساختار تکرار شود. این block تکرار شونده در حالت ساده به صورت زیر می باشد.

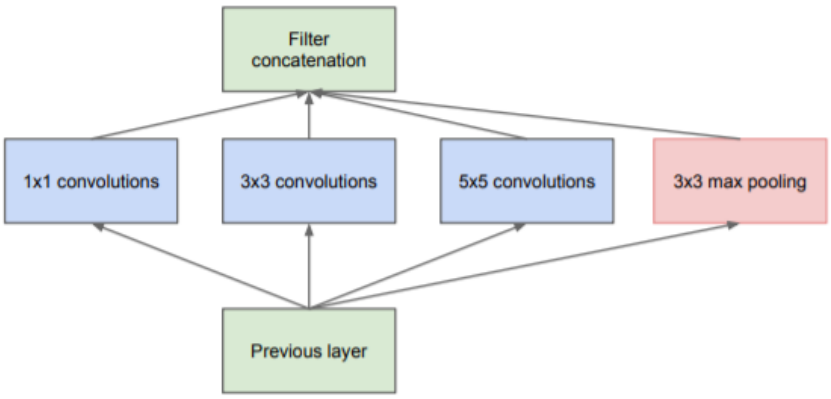


Figure 11: Simple inception module

اما مشکل اصلی این ماژول هزینه محاسباتی سنگین می باشد. فرض کنیم ورودی به یک ماژول inception دارای ابعاد 28*28*192 می باشد. با در نظر گرفتن 32 فیلتر که هر کدام ابعاد 5*5*192 دارند ، ابعاد خروجی این فیلتر ها برابر با 28*28*32 می شود (با در نظر گرفتن padding = same). بنابراین تعداد محاسبات مودر نیاز به صورت زیر می باشد:

$$28 * 28 * 32 * 5 * 5 * 192 = 120M$$

بنابراین، همانطور که مشاهده می شود نیاز به 120 میلیون ضرب نیاز است تا خروجی لایه 5*5 به دست بیاید. چنین محاسباتی در رابطه با فیلتر 3*3 نیز صادق می باشد. برای کاهش این هزینه محاسباتی ساختار زیر برای هر ماژول پیشنهاد شده است.

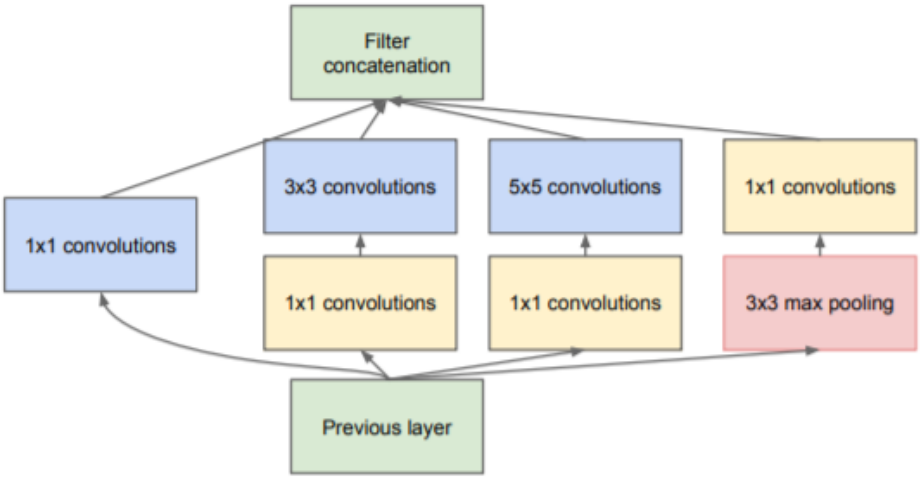


Figure 12 inception module with dimensionality reduction

همانطور که مشاهده می شود قبل از لایه های کانولشنی و بعد از max pooling یک لایه کانولوشنی 1*1 اضافه شده است. در ابتدا به نظر می رسد افزودن این لایه کانولوشنی، خود به پیچدگی های محاسباتی اضافه می کند اما در واقعیت میزان ضرب های مورد نیاز از 120 میلیون بار به 12 میلیون کاهش می یابد. ورودی ماژول هم چنان دارای ابعاد 192*28*28 می باشد. در مرحله اول 16 فیلتر با ابعاد 192*1*1 داریم. در نتیجه خروجی این مرحله دارای ابعاد 16*28*28 می باشد. بنابراین تعداد ضرب های مورد نیاز در مرحله اول برابر با تعداد زیر می باشد.

$$28 * 28 * 16 * 1 * 1 * 192 = 2.4\ M$$

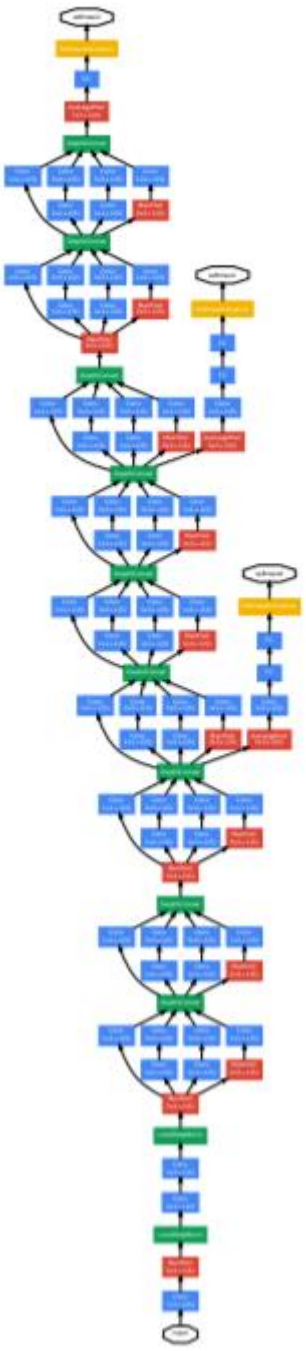
در مرحله دوم بلوک با ابعاد 16*28*28 تحت 32 فیلتر کانولوشنی با ابعاد 16*5*5 قرار می گیرد، بنابراین بلوک به دست آمده در نهایت ابعاد 32*28*28 خواهد داشت(همانند simple inception module).تعداد ضرب های صورت گرفته در مرحله دوم نیز به شرح زیر می باشد.

$$38 * 38 * 32 * 5 * 5 * 16 = 10\ M$$

بنابراین تعداد ضرب های صورت گرفته برابر با جمع تعداد ضرب های مرحله اول و مرحله دوم یا به عبارتی 12.4 میلیون ضرب می باشد. همانطور که مشاهده می شود در این حالت تعداد ضرب های مورد نیاز در مقایسه با حالت ساده ماژول inception ، 0.1 شده است.

حال ساختار کلی شبکه inception بر روی این بلوک های سازنده استوار است. 9 عدد از این ماژول ها پشت سر هم قرار می گیرند و این شبکه به وجود می آید. در واقع 22 لایه مخفی (یا 28 لایه مخفی با در نظر گرفتن max pooling ها) وجود دارد. ساختار کلی این شبکه در تصویر زیر مشاهده می شود.(تیم

توسعه دهنده این شبکه ، تیم googlent بوده است، به همین خاطر به این شبکه GoogLent نیز گفته می شود)



GoogLeNet network with all the bells and whistles-Figure 13

همانطور که در تصویر بالا مشخص است، علاوه بر ماژول های تکرار شونده inception، بلاک های کمکی نیز وجود دارند.در هر یک از این بلاک های کمکی inception تابع softmax اعمال می شود تا یک loss

کمکی در این مرحله نیز به دست بیاید. loss نهایی شبکه برابر با جمع وزن دار loss های به دست آمده کمکی و loss محاسبه شده در مرحله آخر می باشد.وزن پیشنهاد شده برای loss های کمکی در مقاله این شبکه ، 0.3 می باشد. این عمل سبب جلوگیری از overfitting می شود.

$$Total_{Loss} = Final_{Loss} + 0.3(aux_{loss1} + aux_{loss2})$$

از مزایای دیگر این روش افزایش تعداد unit در هر مرحله بدون افزودن به پیچیدگی می باشد. هم چنین داده ها در scale های مختلف بررسی می شوند و سپس نتایج حاصل در کنار یکدیگر به مرحله بعد داده می شود. در نتیجه مرحله بعدی اطلاعاتی از scale های مختلف را به صورت همزمان دارد

نهایتا ساختار کلی ارائه شده (بدون جزییات) در مقاله این شبکه، به صورت زیر می باشد.

Table 3: GoogLeNet incarnation of the Inception architecture

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

نسخه های مختلف

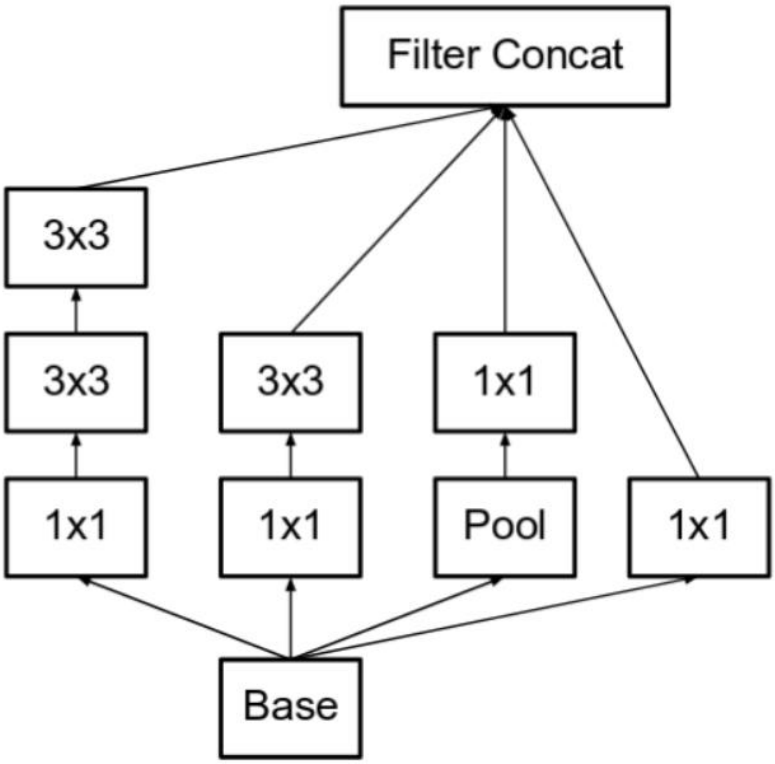
براساس مقالات منتشر شده ، نسخه های مختلف این شبکه عبارتند از :

- Inception-V1
- Inception-V2&V3 (هر دو در یک مقاله معرفی شدند)
- Inception-V4 & Inception-ResNet (هر دو در یک مقاله معرفی شدند)

در مرحله معماری شبکه، نسخه Inception –V1 که زیرساخت اصلی نسخه های مختلف Inception می باشد، بررسی گردید. حال به مطالعه ورژن های دیگر می پردازیمو

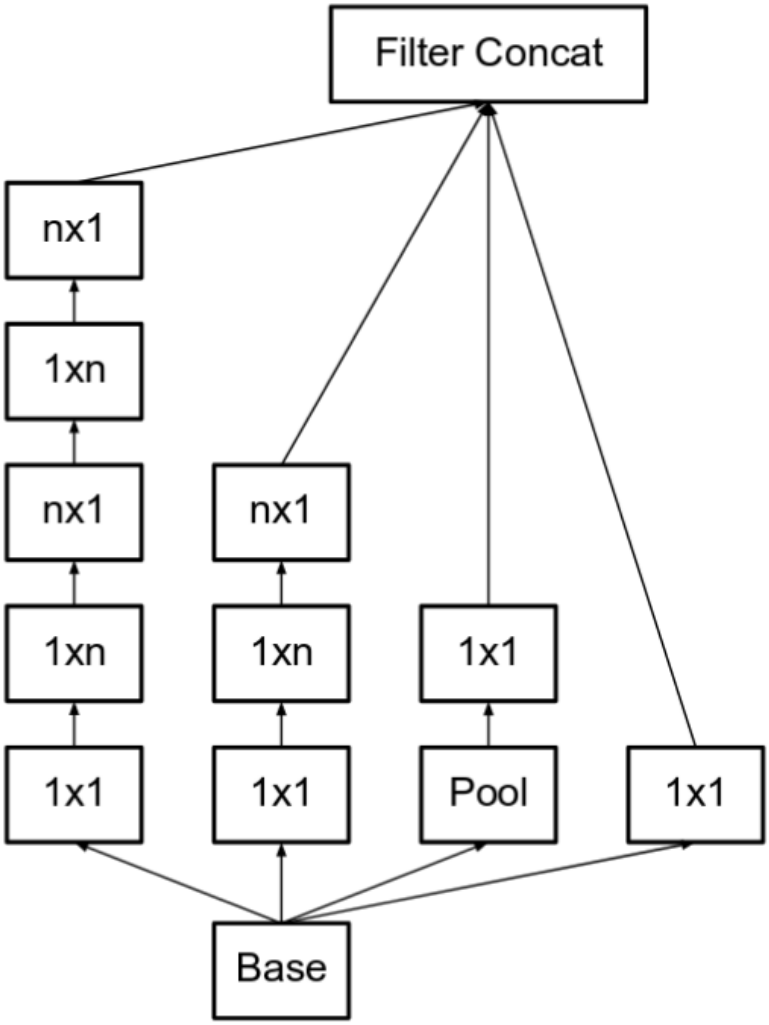
در ورژن های 2 و3 در کنار بالا رفتن دقت ، از پیچیدگی های محاسباتی کاسته شده است.

در ورژن 2 به جای لایه های کانولوشنی 5*5، دو عدد لایه کانولوشنی 3*3 پشت سرهم قرار گرفته اند(همانند شکل زیر). این مساله سبب افزایش سرعت محاسباتی می شود(دلیل افزایش سرعت محاسباتی همان دلیلی است که در قسمت مربوط به معماری شبکه یاد شد/کاهش تعداد ضرب)



Inception v2 modifications-14 **Figure**

هم چنین به جای فیلتر های n*n، فیلتر های n*1 و 1*n پشت سرهم قرار داده می شوند. به این حالت factorized convolution گفته می شود و محاسبات نسبت به حالت عادی سریع تر است.در این روش به جای آنکه به عمق شبکه اضافه شود، به پهنای شبکه اضافه می شود. در صورتی که شبکه عمیق شود، از اطلاعات مفید آن کاسته و در نتیجه feature ها به اندازه کافی استحراج نمی شوند. Factorized convolution به صورت زیر می باشد.



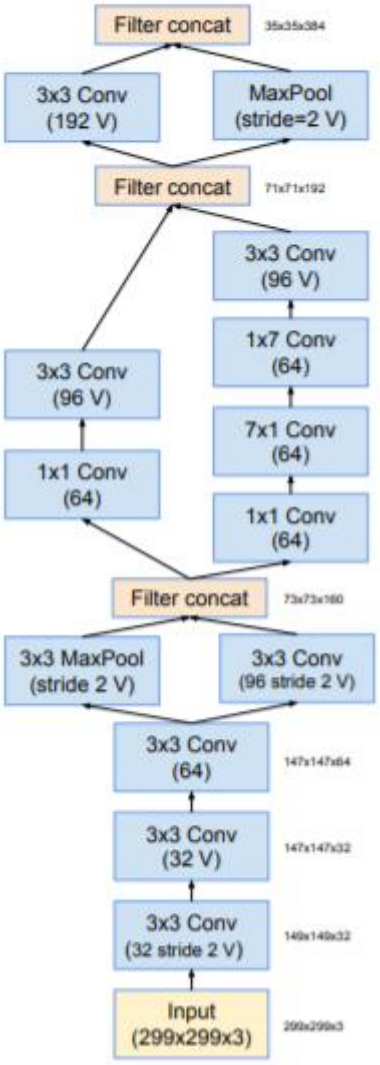
Factorized Convolution in inception v2 -15 Figure

Inception v3

تمامی آپدیت های موجود در ورژن 2 ، در این ورژن نیز وجود دارد. علاوه بر آن ها برخی تغییرات دیگر نیز رخ داده است که سبب افزایش سرعت و دقت شده است. به طور مثال در لایه های فرعی(کمکی)، batch normalization اضافه شده است یا نوع optimizer تغییر کرده است. از دیگر تغییرات اضافه شدن لایه های 7*7 factorized می باشد.

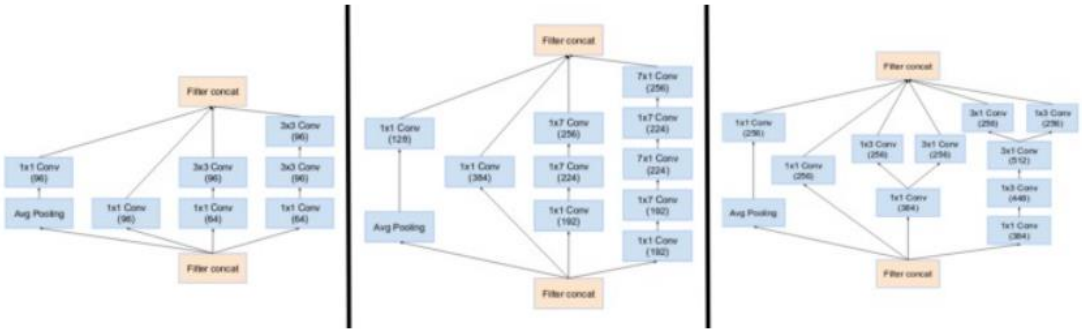
Inception v4

در این شبکه ، لایه های شبکه قبل از ورود به اولین ماژول Inception دچار تغییر شده است(به این لایه ها stem گفته می شود.). همانطور که مشاهده می شود، این لایه ها به صورت زیر می باشند. در حالی که در ورژن اولیه تنها تعدادی لایه کانولوشنی و pooling پشت سرهم قرار گرفته بودند.



stem of the pure Inception-v4-16 **Figure**

در این ورژن بلاک های inception بسیار شبیه به بلاک های inception ورژن های 2 و3 می باشد. از جمله تفاوت ها ، وجود average pooling در ورژن 4 می باشد. هم چنین لایه های کانولوشنی 7*7 به صورت لایه های 1*7 و 7*1 پشت سر هم قرار گرفته اند. چنین موردی در ورژن 2 وجود نداشت. بلاک های inception در ورژن 4 به صورت زیر می باشند.(حالت های A,B,C)



A,B,C Inception modules in version 4 -17 Figure

Inception-ResNet

این معماری، خود دارای 2 ورژن می باشد. بسیار شبیه به Inception v4 می باشدو در این شبمه تنها برخی از هایپرپارامتر ها تغییر کرده است.

مزایا و معایب نسبت به سایر مدل ها

برای بررسی مزایا و معایب این مدل نسبت به بررسی نتایج این معماری در چالش Imagenet Large Scale Visual Recognition می پردازیم.طبق جدول زیر ، این شبکه موفق به کسب مقام اول در سال 2014 از نظر کارآیی طبقه بندی شده بود.(بر روی داده های imagenet)

Table 4: Classification performance

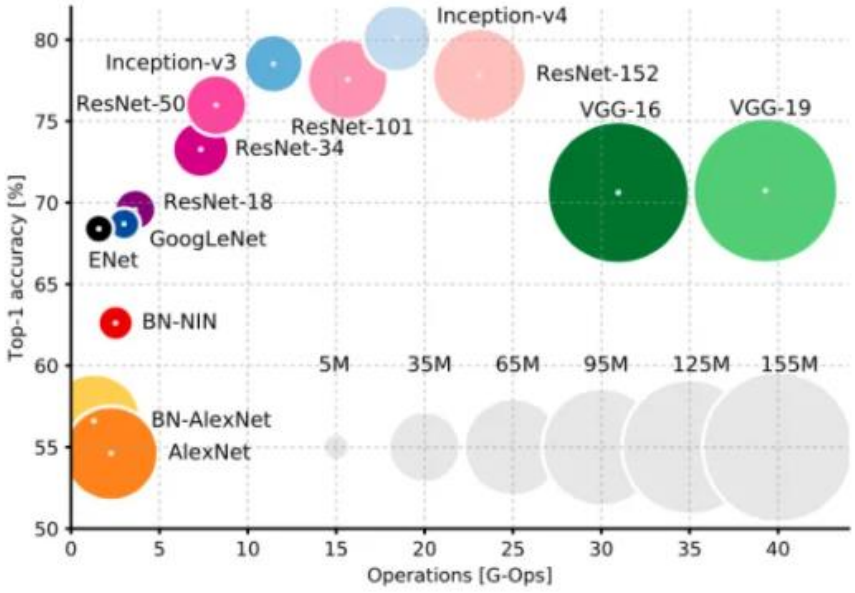
Team	Year	Place	Error (top-5)	Uses external data
SuperVision	2012	1st	16.4%	no
SuperVision	2012	1st	15.3%	Imagenet 22k
Clarifai	2013	1st	11.7%	no
Clarifai	2013	1st	11.2%	Imagenet 22k
MSRA	2014	3rd	7.35%	no
VGG	2014	2nd	7.32%	no
GoogLeNet	2014	1st	6.67%	no

هم چنین مشاهده می شود که این شبکه بالاترین map را در همین سال در همین چالش به دست آورده است.

Table 5: Classification performance

Team	Year	Place	mAP	external data	ensemble	approach
UvA-Euvision	2013	1st	22.6%	none	?	Fisher vectors
Deep Insight	2014	3rd	40.5%	ImageNet 1k	3	CNN
CUHK DeepID-Net	2014	2nd	40.7%	ImageNet 1k	?	CNN
GoogLeNet	2014	1st	43.9%	ImageNet 1k	6	CNN

نهایتا نمودار زیر نیز نمایانگر میزان دقت انواع ورژن های شبکه inception در مقایسه با سایر شبکه ها بر روی دیتاست imagenet می باشد.



imagenet winner models [1]-18 **Figure**

محمدودیت های سایز تصویر ورودی

سایز عکس ورودی باید 229*229 پیکسل باشد

پیش پردازش های اولیه برای تصویر ورودی

در مقاله مربوط به این شبکه در رابطه با پیش پردازش های لازم صحبتی نشده است. اما با توجه به کد مروبط به پیش پردازش های موجود در سایت keras نیاز است تا داده ها بین -2 تا 0 نرمالیزه بشوند

سایز خروجی مدل و معنای آن

خروجی در آخرین بلاک inception به صورت 7*7*1024 می باشد. سپس یک لایه Average pooling با ابعاد 7*7*1 وجود دارد. در نتیجه سایز به 1*1*1024 می رسد. به منظور جلوگیری از اورفیت شدن 40٪ دراپ اوت وجود دارد. نهایتا ابعاد به 1*1*1000 با عمق 0 می رسد. در لایه های آخر این شبکه به جای لایه های fully connected ، average pooling قرار داده شده است که سبب افزایش دقت تا 0.4 ٪ می شود.

ب/ج

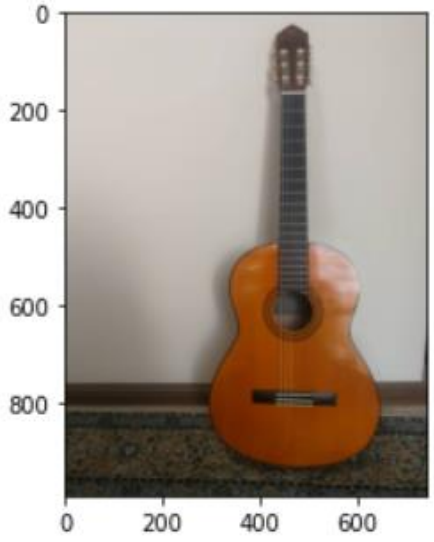
در این دو بخش از مدل اماده inception v3 موجود در کتابخانه keras استفاده شده است. این شبکه بر روی دیتا ست imagenet آموزش داده شده است و وزن های نهایی به صورت آماده در کتابخانه keras موجود می باشند . iamgenet دیتاست بسیار بزرگی است که 1000 دسته مختلف را درون خود جای

داده است. اکثر اشیای موجود در این دیتاست، اشیایی هستند که به صورت روزمره با آن ها سر و کار داریم.در این دیتاست 1.2 عکس برای آموزش، 50 هزار عکس برای validation و 100 هزار عکس برای تست وجود دارد.

پس از پیاده سازی این شبکه به روش transfer learning نوبت به بررسی عکس گرفته شده از یک گیتار می رسد.

ابتدا لازم است تا ابعاد این عکس به 229*229 پیکسل تغییر پیدا کند. سپس نیاز است تا shape ماتریس این عکس از (m,n,3) به (1,m,n,3) تغییر کند. سپس پیش پردازش های لازم به کمک دستور آماده کتابخانه keras صورت می گیرد. سپس سه دسته با بیشترین احتمال به ترتیب به صورت زیر می باشند.

```
1. acoustic_guitar: 95.80%
2. electric_guitar: 2.25%
3. banjo: 0.19%
<matplotlib.image.AxesImage at 0x7f51f79fdd68>
```



inception version 3 prediction on test image -19 **Figure**

با احتمال 95.8% حدس زده شده است که این تصویر گیتار اکوستیک می باشد. همانطور که مشخص است این حدس به درستی زده شده است.

د

اگر تصویر وردی به شبکه جزو اشیا قابل تشخیص توسط مدل نباشد، خروجی مدل قابل استناد نمی باشد. از همین رو قبل از اینکه عکس به شبکه اصلی داده شود لازم است تا بررسی شود که آیا جزو کلاس های قابل تشخیص توسط شبکه می باشد یا خیر. برای این کار راهایی متفاوتی پیشنهاد شده است، اما در این سوال از روش autoencoder استفاده می شود. از آنجایی که دیتاست imagenet دارای تعداد عکس بسیار فراوانی می باشد فرآیند یادگیری اتوانکودر بسیار زمان بر می شود. از همین رو از دیتاست cifar10 که دیتاست کوچک تری می باشد، استفاده شد تا صحت عملکرد شبکه بررسی شود.

در واقع شبکه های autoencoder در گام اول تصاویر را encode می کنند و به این ترتیب اطلاعات غیرمهم حذف شده و سایز عکس کوچکتر می شود. سپس در گام دوم این تصاویر انکود شده، decode می شوند. عکس به دست آمده با عکس اولیه همبستگی دارد. در صورتی که عکس داده شده به شبکه autoencoder از دسته داده های قابل تشخیص توسط مدل باشد، loss ای به دست می آید که در نزدیک به loss داده های test & validation, train می باشد. در غیر اینصورت تفاوت چشمگیری بین loss به دست آمده و loss سه دسته یاد شده وجود خواهد داشت. از همین اصل در حل این سوال استفاده شده است.

معماری اتو انکدور استفاده شده به شرح زیر می باشد.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_13 (InputLayer)        [(None, 3072)]            0
_________________________________________________________________
dense_20 (Dense)             (None, 1200)              3687600
_________________________________________________________________
dense_21 (Dense)             (None, 3072)              3689472
=================================================================
Total params: 7,377,072
Trainable params: 7,377,072
Non-trainable params: 0
```

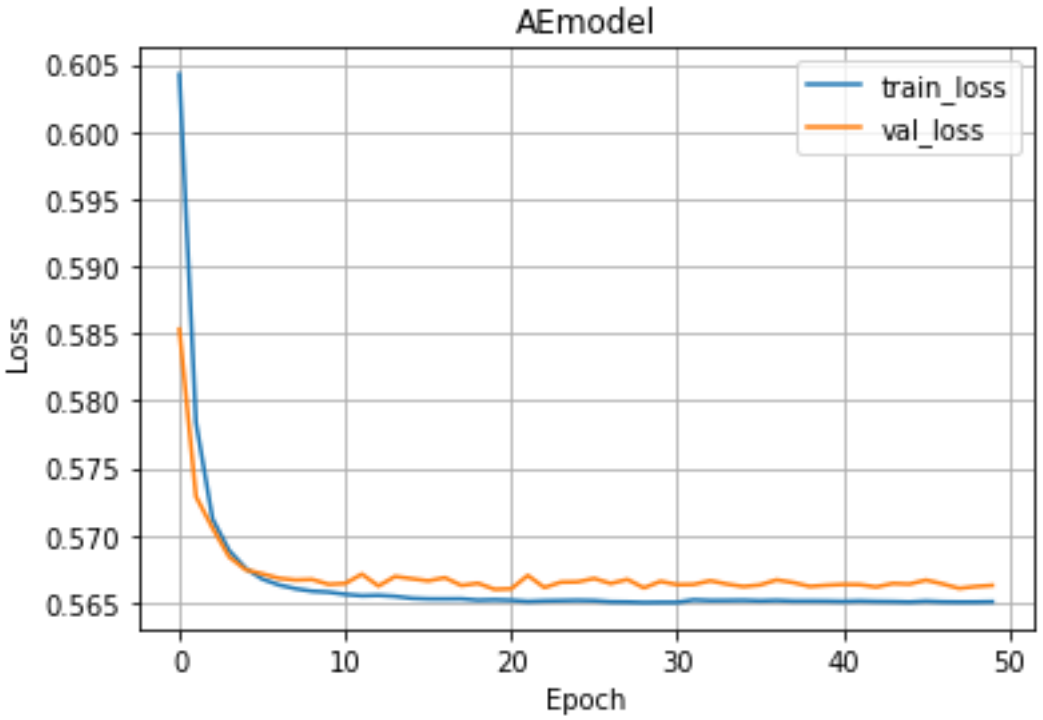
AEmodel -20 **Figure**

هایپرپارامترهای استفاده شده نیز به صورت زیر می باشند.

- Loss = binary_crossentropy
- Optimizer = adam

- Batch_size = 32
- Epochs = 50

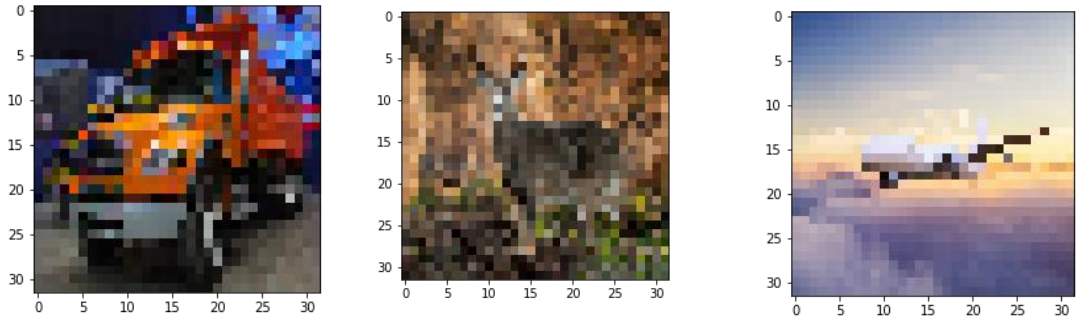
پس از آموزش شبکه بر روی دیتاست cifar10 مقدار loss_train برابر با 0.5651 و مقدار loss_valid برابر با 0.5663 به دست آمد. نمودار تغییرات loss نیز به صورت زیر می باشد.



AE model loss -21 **Figure**

سپس داده های تست مجموعه cifar10 بر روی شبکه مورد بررسی قرار گرفتند. در این حالت نیز میزان loss شبکه برابر با 0.5831 می باشد. هم چنین برای اطمینان کامل از صحت عملکرد شبکه عکس های زیر که خارج از دیتاست cifar10 می باشند ولی جزو اشیا قابل تشخیص این شبکه می باشند، به شبکه اتوانکودر داده شدند. Loss به دست آمده برای هر یک از عکس ها قابل مشاهده می باشد.(این عکس ها به ابعاد 32 در 32 تغییر پیدا کرده اند تا توسط شبکه قابل تشخیص باشند، تصاویر زیر با ابعاد 32 در 32 می باشند)

airplane Deer Truck

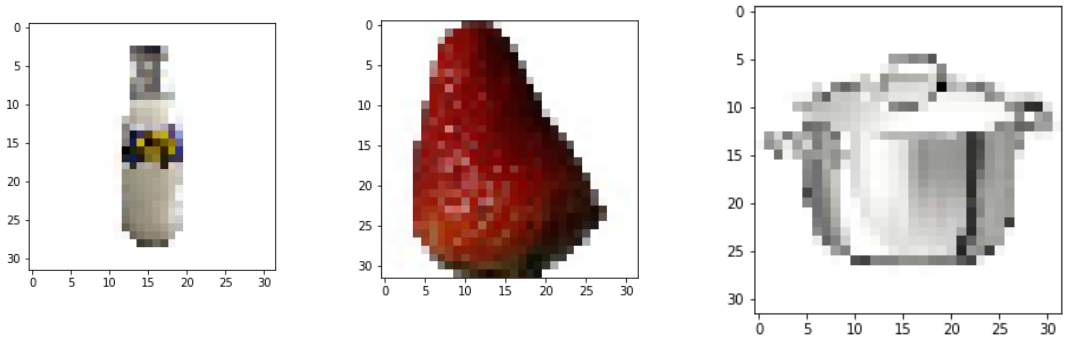


Loss = 0.5063 Loss = 0.59 Loss=0.58

AE model on known objects -22 **Figure**

حال تعدادی عکس خارج از اشیا موجود در دیتاست cifar10 به شبکه اتو انکودر می دهیم، نتیجه به شرح زیر می باشد.

بطری دوغ توت فرنگی قابلمه



Loss = 0.18 Loss = 0.288 Loss=0.2885

AE model on unknown objects-23 **Figure**

همانطور که مشاهده می شود ، loss بر روش اشیایی که در دیتاست موجود نیستند تفاوت چشمگیری با اشیا موجود در دیتاست دارد. در نتیجه می توان از این ویژگی استفاده کرد و برروی اشیا لیبل accept یا reject زد. با توجه به loss به دست آمده برای اشیا قابل تشخیص ، اگر loss شی ای بین 4.0 تا 0.6 باشد این شی توسط شبکه قابل تشخیص و در غیر اینصورت غیر قابل تشخیص می باشد. به طور مثال توت فرنگی دارای loss 0.288 می باشد ، بنابراین جزو اشیا قابل تشخیص توسط شبکه نیست. نتیجه به دست آمده کاملا صحیح می باشد چرا که در داده های cifar10 توت فرنگی وجود ندارد.

۵

یک لیست از اشیا قابل تشخیص توسط مدل بدست بیاورید ، یک مجموعه داده جمع آوری کنید که شامل دو کلاس باشد) حداقل هر کلاس شامل 200 تصویر باشد، قاعدتا می توانید مجموعه داده بیشتری جمع آوری کنید و محدودیتی وجود ندارد(و مدل را با استفاده از این مجموعه داده مجددا آموزش دهید.

در این سوال از شبکه ی InceptionV3 که با Imagenet آموزش داده شده است استفاده می شود. برای جمع آوری دیتاست جدید از داده های Fruit 360 استفاده شده و از دو کلاس Mango و Peach 400 تصویر برای داده های train و validation و 200 تصویر برای تست جدا شده است. در ابتدا نیاز است این تصاویر را با ابعاد ورودی (299,299) آپلود کرده و از دستور preprocessing موجود در کتابخانه ی InceptionV3 داده ها را برای استفاده ی شبکه آماده می کنیم. تصاویر زیر نمونه ای از دیتاهای داده شده به شبکه را نشان میدهد.

Figure 24: Mango and Peach used in training the network

بعد از آماده سازی دیتا وزن های شبکه را از کتابخانه ی InceptionV3 وارد کرده و یک لایه ی Pooling و دو لایه ی Dense به شبکه اضافه می کنیم که لایه ی آخر لایه ی Prediction بوده که دارای 2 نورون است(هر نورون نماینده ی یک کلاس جدید اضافه شده.). از آنجایی تعداد داده ها بسیار کم است تعداد 3 لایه برای آموزش شبکه کافی است. بعد از بازسازی لایه های آخر شبکه با استفاده از داده های جدید شبکه

را آموزش می دهیم. نتایج خطا و دقت شبکه و همچنین خطا و دقت داده های تست در ادامه ارائه شده است:

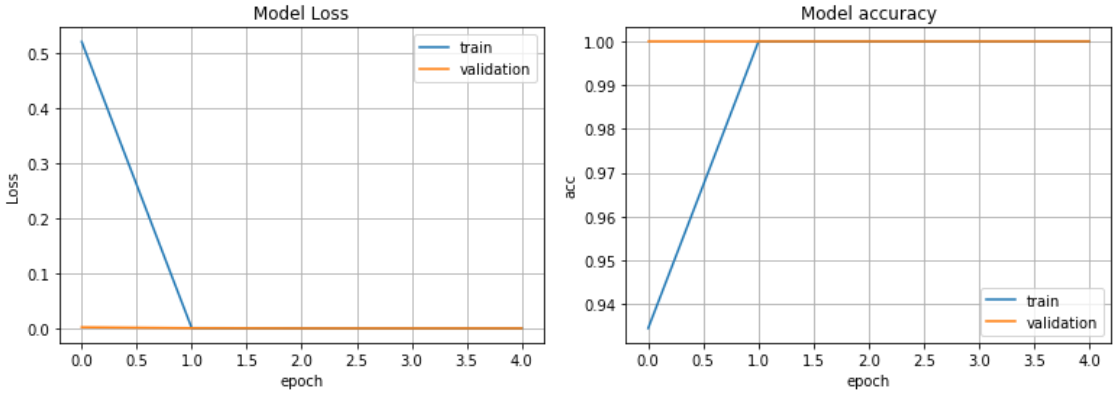


Figure 25: Model loss and accuracy after fine-tuning

Test loss	Test accuracy
5.7076e-05	1.0000

همان طور که دیده می شود شبکه هم در قسمت آموزش و هم در قسمت تست به دقت 1 و همچنین خظای بسیار پایین رسیده است.

سوال 3 – Object Detection

الف

در معماری شبکه yolo-v1 ، این شبکه ابتدا تصویر را به s*s قسمت مساوی تقسیم می کند.در این شبکه مقدار s معمولا برابر با 7 می باشد. هر بخش تنها تعداد محدودی boundary box را پیش بینی می کند. این تعداد(B) در شبکه YOLO-v1برابر با 2 می باشد. در این شبکه هر boundary box 5 ویژگی دارد که به شرح زیر می باشند.

- x : مختصات مرکز هر box
- y : مختصات مرکز هر box
- w : پهنای هر box
- h : ارتفاع هر box
- Box confidence score : این مقدار نشان دهنده احتمال وجود جسم در هر هر باکس و میزان دقت هر boundary box است.

(x,y,w,h) همگی نسبت به ابعاد تصویر نرمالیزه می شوند، پس مقداری بین 0 و 1 دارند.

نهایتا هر grid به هریک از کلاس های موجود (C Class) احتمالی را اختصاص می دهد. در این سوال C برابر با 300 می باشد. بنابراین ابعاد لایه اخر در شبکه YOLO-v1 به صورت زیر می باشد.

$$(S, S, B * 5 + C) = (7,7,310)$$

در معماری شبکه YOLO-v3 در لایه آخر، 3 feature map با ابعادهای مختلف(13*13,26*26,52*52) وجود دارد که برای تشخیص اشیا با اندازه های متفاوت قرار داده شده اند.در هر cell، 3 bounding box وجود دارد و هر یک از این bounding box ها همانند شبکه YOLO-v1 با 5 ویژگی مشخص می شوند. بنابراین ساختار این شبکه در لایه آخر برای حالتی که 300 کلاس داشته باشیم به صورت زیر می باشد.

$$N * N * \big(3 * (4 + 1 + 300)\big) = N * N * 915$$

N نیز برابر با یکی از 3 حالت اشاره شده می باشد (13و26 و 52 – هر یک از این اعداد به تفصیل در ادامه بررسی خواهند شد)

ب

همانطور که در بخش قبل ذکر شد در شبکه YOLO-V1، هر عکس به خانه های 7*7 تقسیم می شود و برای هر grid، 2 bounding box در نظر گرفته می شود. در نتیجه در صورتی که در هر grid بیش از دو شی قرار داشته باشد شبکه قادر به تشخیص آن ها نیست و تنها 2 جسم از بین تمامی اشیا را تشخیص می دهد.

حال در معماری YOLO-v2 هر عکس به به خانه های 13*13 تقسیم می شود. هم چنین در این شبکه هر grid دارای 5 bounding box می باشد(به جای 2 تا). در واقع 5 عدد bounding anchor از پیش تعیین شده به شبکه پیشنهاد داده می شود و شبکه با offset دادن به آنها بهترین bounding box برای هر جسم را تعیین می کند. بنابراین مشاهده می شود که 1-تعداد grid ها افزایش یافته است 2-تعداد bounding box نیز در هر grid افزایش یافته است. بنابراین می توان گفت مشکل اشیا با تداخل بالا در این روش نسبت به حالت اولیه بهبود یافته است.

ج

در شبکه YOLO-v1 بسیاری از ویژگی های مربوط به اشیا کوچک در لایه های اولیه شبکه به دست می آیند. از آنجا که در این شبکه مسیری از این الگو ها به انتهای شبکه وجود ندارد شبکه در تعیین اشیای کوچک دچار مشکل می شود. این مشکل در YOLO-V2 به کمک روش pass through تا مقداری حل شد. در واقع اگر feature ای را زودتر از شبکه خارج کنیم و بر روی آن پردازشی صورت نگیرد سبب می شود تا ویژگی ها را به طور کامل شناسایی نکنیم.در شبکه YOLO-V3 از معماری هرمی استفاده شده است که در تصویر زیر قابل مشاهده می باشد.

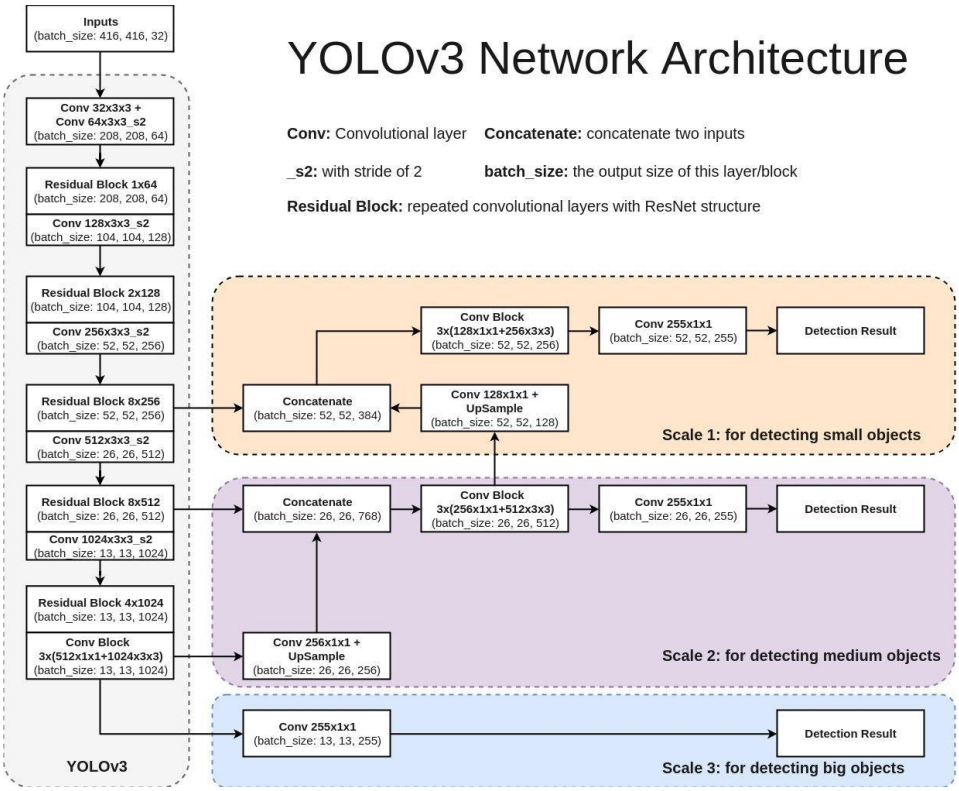


Figure 26-معماری شبکه **YOLO-v3**

همانطور که از تصویر بالا مشخص است، ابتدا تعدادی لایه کانولوشنی بر روی شبکه اعمال می شود و تعدادی ویژگی به دست می آید (feature map ای با ابعاد 52*52) . به صورت مستقیم از این feature map برای تشخیص اشیا با ابعاد کوچک استفاده نمی شود، بلکه از feature های به دست آمده در لایه های بعدی شبکه نیز استفاده می شود. در لایه آخر 13*13 feature map به دست می آید. ابتدا یک لایه کانولوشن بر روی آن صورت می گیرد و سپس می توان به این وسیله اشیا با ابعاد بزرگ را توسط شبکه شناسایی کرد. از طرف دیگر همین feature map با feature map به دست آمده در مراحل قبل با ابعاد 26*26 concat می شود. نتیجه به دست آمده از یک طرف برای شناسایی اشیا با ابعاد متوسط استفاده می شود و از طرف دیگر با feature map ابتدایی با ابعاد 56*56 concat شده و برای شناسایی اشیا با ابعاد کوچک مورد استفاده قرار می گیرد. همانطور که مشاهده شد این معمای قابلیت طبقه بندی اشیا در سه سطح زیر را دارد .

- Scale 1 : for detecting small objects
- Scale 2 :for detecting medium objects
- Scale 3 : for detecting big objects

نهایتا باید اشاره کرد که تعداد bounding box ها در YOLO-v3 10647 باکس می باشد. بنابراین دلایل کارایی شبکه در شناسایی اشیا باپراکندگی بالا در سایز ، بالا می باشد.

د

نهایتا به منظور بررسی توانایی شبکه yolov5 ، این شبکه بر روی داده های chess pieces آموزش دید و سپس تست شد. قبل از شروع آموزش شبکه، نیاز است تا تغییرات کوچکی در کد های موجود در گیت هاب yolov5 اعمال شود. با توجه به فایل data.yaml موجود در دیتاست مشخص می شود که 12 کلاس مختلف داریم. yolo-v5 خود دارای ورژن های مختلفی می باشد. (s,x,...). در حل این سوال از ورژن yolov5s استفاده شده است. در فایل yolov5s.yaml موجود در گیت هاب ، تعداد کلاس ها برابر با 80 نوشته شده است، اما در این مساله نیاز است که این عدد به 12 (تعداد کلاس های مهره های شطرنج) تغییر کند. پس از انجام این تغییر جزئی می توان به آموزش شبکه پرداخت. برای افزایش دقت شبکه بهتر است تا از وزن های آماده برای آموزش استفاده شود . هایپر پارامتر های انتخابی برای آموزش شبکه به شرح زیر می باشند:

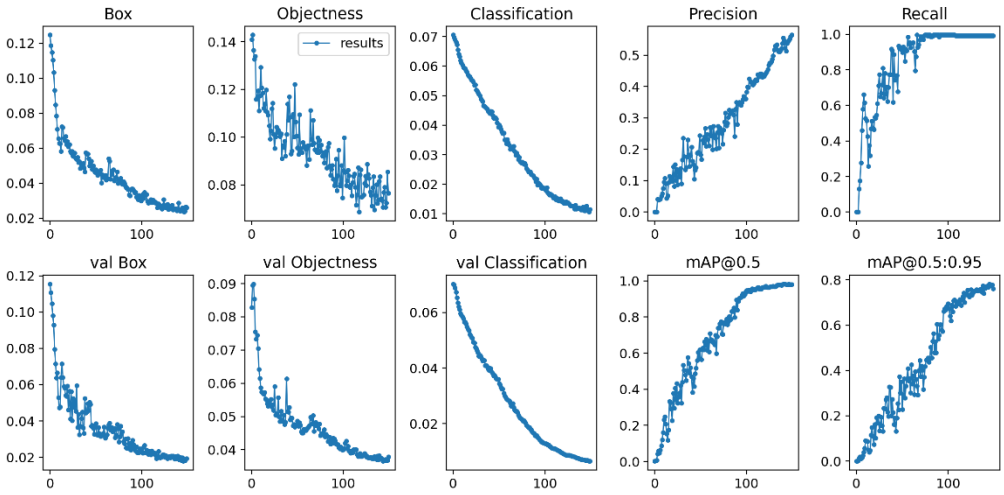
- Batch = 16
- Epochs = 150 (تعداد ایپاک نسبتا بالا در نظر گرفته شد تا map بهتری به دست بیاید)
- وزن های آماده در روند حل مساله

نتایج به دست آمده بر روی داده های train به صورت زیر می باشد.

map on train data :6 **Table**

map@0.5	map@0.5:0.95
0.981	**0.756**

نمودار تغییرات loss و معیارهای دیگر ارزیابی عملکرد شبکه به صورت زیر می باشند



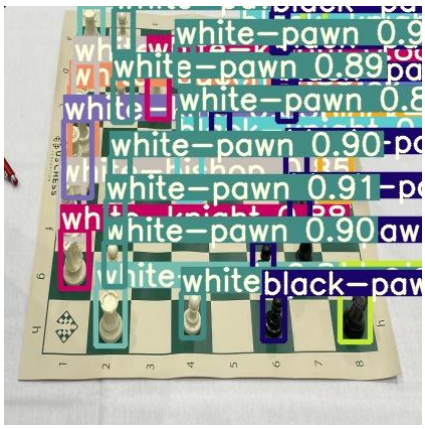
loss & other performance metrics -27 **Figure**

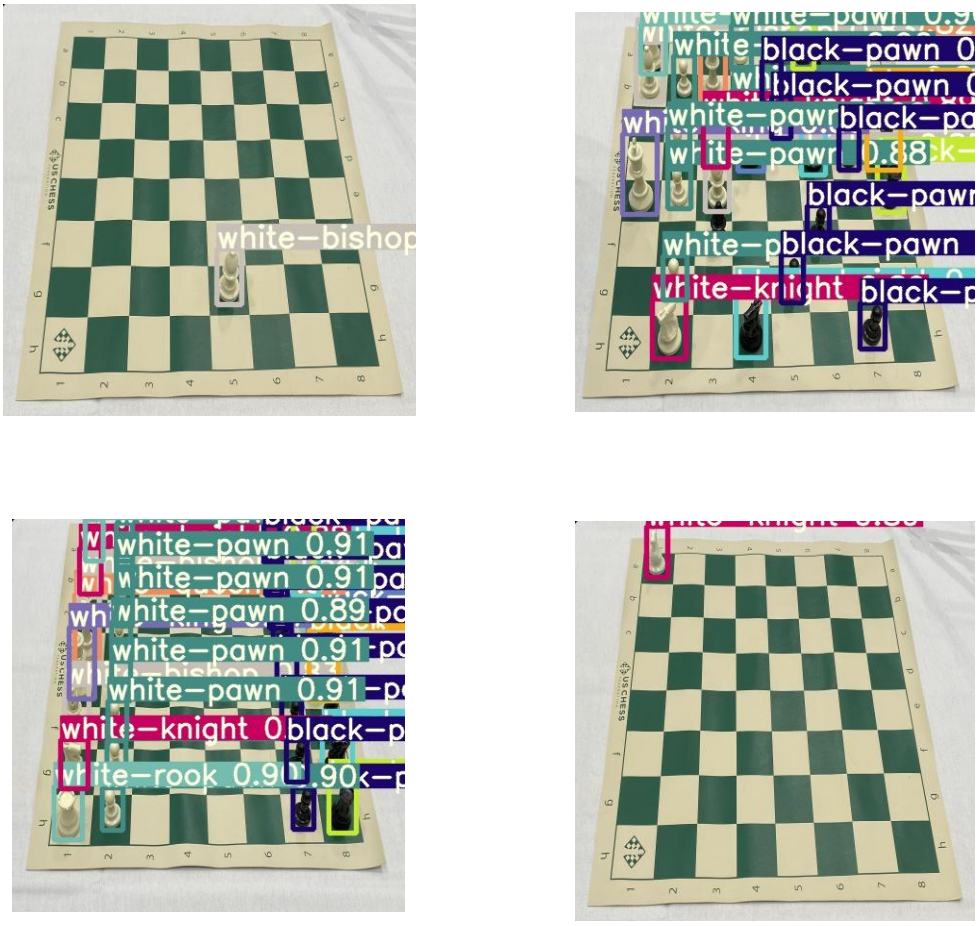
سپس بهترین وزن های به دست آمده در مرحله آموزش، بر روی داده های تست استفاده شدند. نتایج زیر نشان دهنده عملکرد شبکه بر روی داده های تست بر اساس معیار map می باشد.

map on test data :7 **Table**

map@0.5	map@0.5:0.95
0.982	**0.778**

تعدادی از تصاویر تشخیص داده شده توسط شبکه به صورت زیر می باشند .(برای ترسیم box به دور هر مهره شطرنج، conf=0.4 قرار داده شده است). همانطور که در تصویر مشخص است این شبکه به خوبی و با احتمال بالا هر یک از مهره های شطرنج را به درستی تشخیص داده است.





YOLO-V5 performance on test data -28 Figure

زمان استنتاج برای 5 داده اول نیز به شرح زیر می باشد.

Table 8: زمان استنتاج برای 5 داده اول

NO	Time(s)
1	0.010
2	0.011
3	0.009
4	0.010
5	0.009