به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین اول

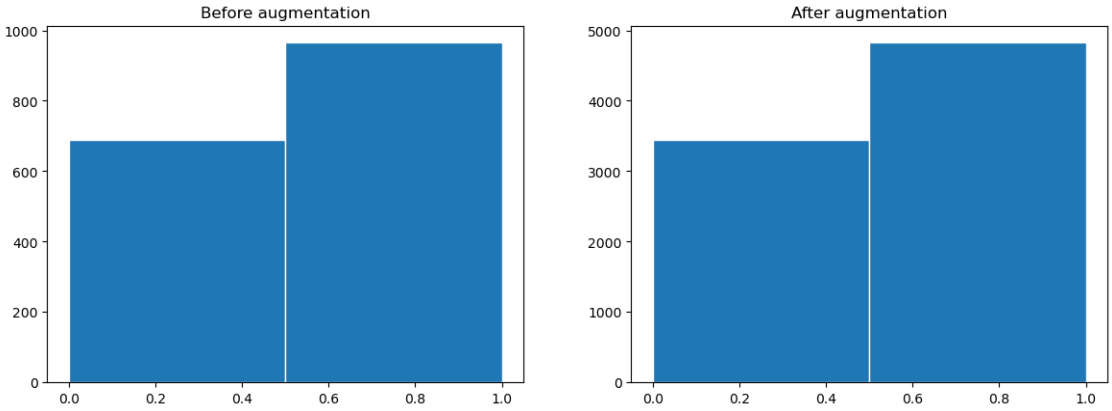
پرسش ۱	نام دانشجو	سیدمحسن ایزدی اونجی
	شماره دانشجویی	۸۱۰۱۰۱۳۱۷
پرسش ۲	نام دانشجو	سید محمد مهدی رضوی
	شماره دانشجویی	۸۱۰۱۰۲۱۵۵
	مهلت ارسال پاسخ	–

فهرست

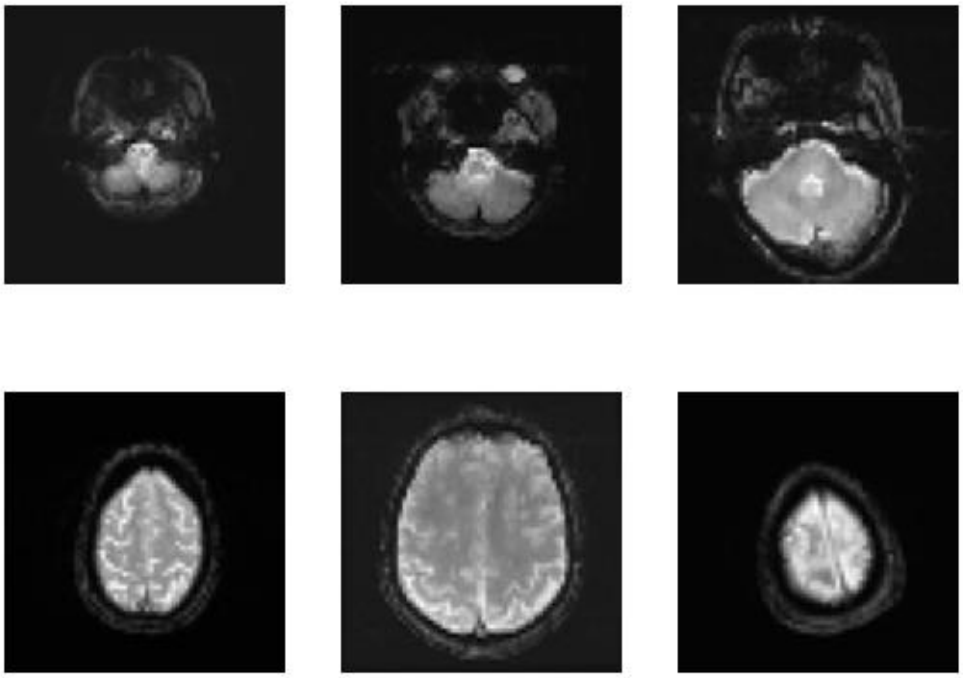
شکل‌ها

جدول‌ها

پرسش ۱. تشخیص آلزایمر با استفاده از تصویر برداری مغزی (ADNI)



شکل ۱ توزیع آماری دادگان قبل و بعد از داده افزایی



شکل ۲ نمونه‌هایی تصادفی از داده‌ها پس از داده افزایی

۱-۱. پیش پردازش تصاویر

در مرحله پیش پردازش طبق متن تمرین داده ها را به منظور تسریع فرآیند آموزش با روش Min-Max نرمال کرده‌ایم و همچنین طبق نیاز معماری‌های پیاده شده در مقاله اندازه تصاویر را به 64x64 تغییر می‌دهیم.

پس از اعمال تغییرات بیان شده از تعداد ۱۶۵۴ تصویر، پس از داده افزایی ۸۲۷۰ عکس تولید شده است. توزیع آماری تصاویر هر دسته در شکل ۱ مشاهده می شود. توجه شود که در این تمرین به نمونه‌های دارای بیماری آلزایمر برچسب "۱" و نمونه های دارای اختلال خفیف (MCI) برچسب "۰" زده شده است. همانطور که در تصویر مشاهده می‌شود توزیع آماری پس از داده افزایی مشابه قبل از آن خواهد بود. از مجموعه جدید دادگان تعداد ۶ تصویر نیز به طور تصادفی در شکل ۲ رسم شده است.

۲-۱. پیاده سازی

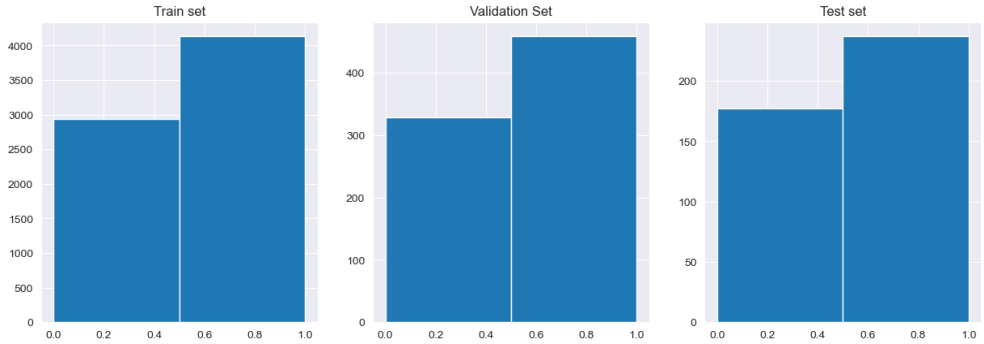
در این بخش سه معماری بیان شده در مقاله را پیاده‌سازی می‌کنیم. به عنوان تابع هزینه از crossentropy categorical و برای بهینه‌ساز از Adam استفاده می‌کنیم. همچنین از Glorot initialization به دلیل جلوگیری از شروع تابع فعال‌ساز نورون‌ها در حالت اشباع استفاده می‌شود که به طور چشمگیری سرعت همگرایی مدل را افزایش و دقت آنرا بهبود می‌دهد. در تمامی نتیجه‌گیری های انجام شده نرخ یادگیری مطابق مقاله بوده مگر در مواردی که ذکر شود.

جدول ۱ پارامترهای آموزش معماری‌های ارائه شده

	مقدار	توضیح
Bach Size	256	-
Num. of Epochs	20	-
Learning Rate	0.1	-
Test size	% 10	۱۰ درصد از مجموعه داده‌ها
Validation Size	% 10	۱۰ درصد از مجموعه تست
Initializer	Glorot	-
Optimizer	Adam	-
Loss Function	Cross Entropy Categorical	-

در ادامه نتایج آموزش سه معماری بیان شده با پارامترهای بیان شده در جدول ۱ را برای هر معماری رسم می‌کنیم.

نکته: طبق متن مقاله پس از هر لایه Activation یک لایه Bach Normalization قرار داده شده است.



شکل ۳ توزیع آماری مجموعه دادگان آموزش، آزمون و **Validation**

Layer (type)	Output Shape	Param #
conv2d_326 (Conv2D)	(None, 64, 64, 32)	320
batch_normalization_492 (BatchNormalization)	(None, 64, 64, 32)	128
dropout_231 (Dropout)	(None, 64, 64, 32)	0
conv2d_327 (Conv2D)	(None, 64, 64, 32)	9248
batch_normalization_493 (BatchNormalization)	(None, 64, 64, 32)	128
max_pooling2d_167 (MaxPooling2D)	(None, 32, 32, 32)	0
conv2d_328 (Conv2D)	(None, 32, 32, 32)	9248
batch_normalization_494 (BatchNormalization)	(None, 32, 32, 32)	128
dropout_232 (Dropout)	(None, 32, 32, 32)	0
conv2d_329 (Conv2D)	(None, 32, 32, 32)	9248
batch_normalization_495 (BatchNormalization)	(None, 32, 32, 32)	128
max_pooling2d_168 (MaxPooling2D)	(None, 16, 16, 32)	0
flatten_85 (Flatten)	(None, 8192)	0
dense_251 (Dense)	(None, 128)	1048704
batch_normalization_496 (BatchNormalization)	(None, 128)	512
dense_252 (Dense)	(None, 64)	8256
batch_normalization_497 (BatchNormalization)	(None, 64)	256
dropout_233 (Dropout)	(None, 64)	0
dense_253 (Dense)	(None, 2)	130

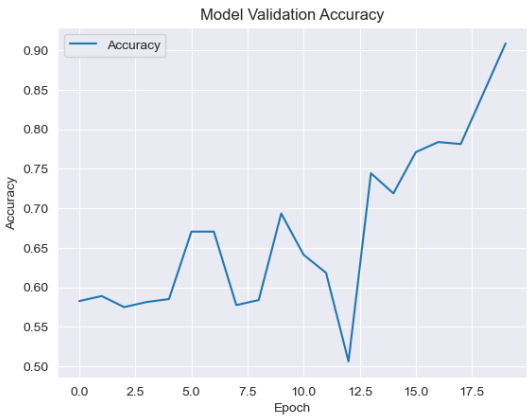
Total params: 1,086,434
Trainable params: 1,085,794
Non-trainable params: 640

شکل ۴ پیاده سازی معماری منتخب

۳-۱. معماری منتخب مقاله

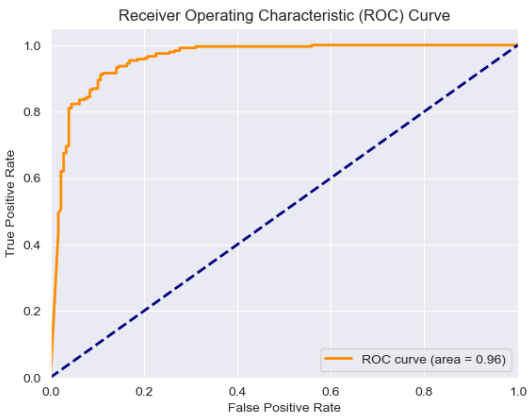
```
    Test Score  Test Accuracy
0     0.544429       0.898551

13/13 [==============================] - 0s 3ms/step
[68]: 0.8985507488250732
```

شکل ۵ نتایج دادگان تست روی معماری منتخب

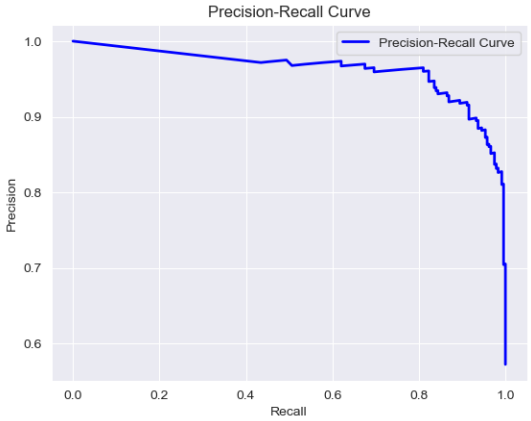


شکل ۶ دقت مجموعه **Validation** در معماری منتخب

شکل ۷ دقت مجموعه داده آموزش در معماری منتخب

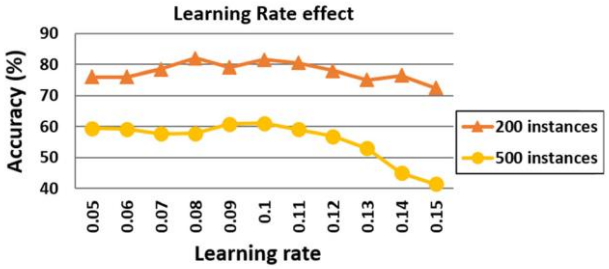


شکل ۸ نمودار **ROC** و **AUC** در معماری منتخب



شکل ۹ نمودار **Precision-Recall** معماری منتخب

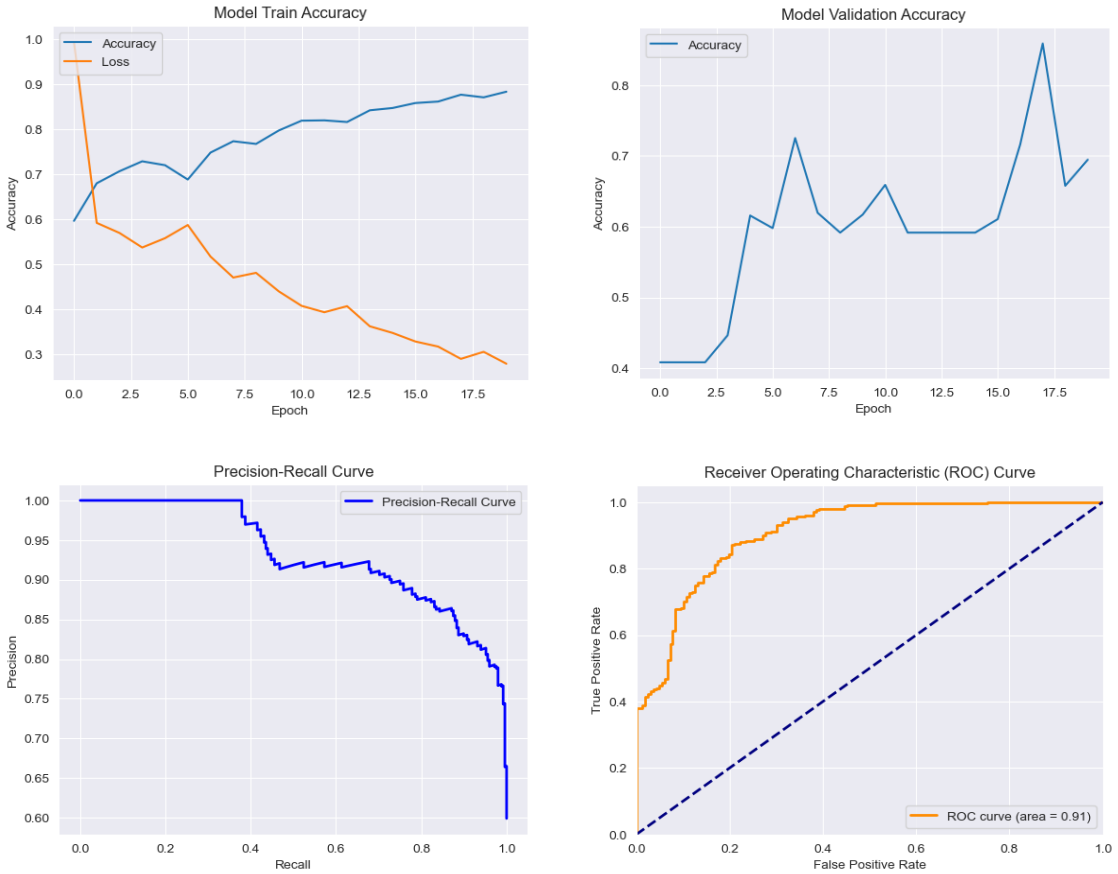
در این معماری مشاهده می‌شود که نمودار ROC متمایل به گوشه بالا سمت چپ است و از خط قطری (خط چین آبی) فاصله گرفته است همچنین شاخص AUC نیز نزدیک یک است که نتیجه مطلوبی است. اما دقت روی دادگان تست به نسبت آنچه در مقاله گزارش شده است کمتر است. دلیل این مورد با استناد به تصویر شماره ۹ مقاله که در شکل ۱۰ این گزارش هم آورده شده است می‌تواند کم بودن نمونه‌ها باشد.



شکل ۱۰ نمودار شکل ۱۰ مقاله مرجع

۴-۱. معماری آزمایشی ۱

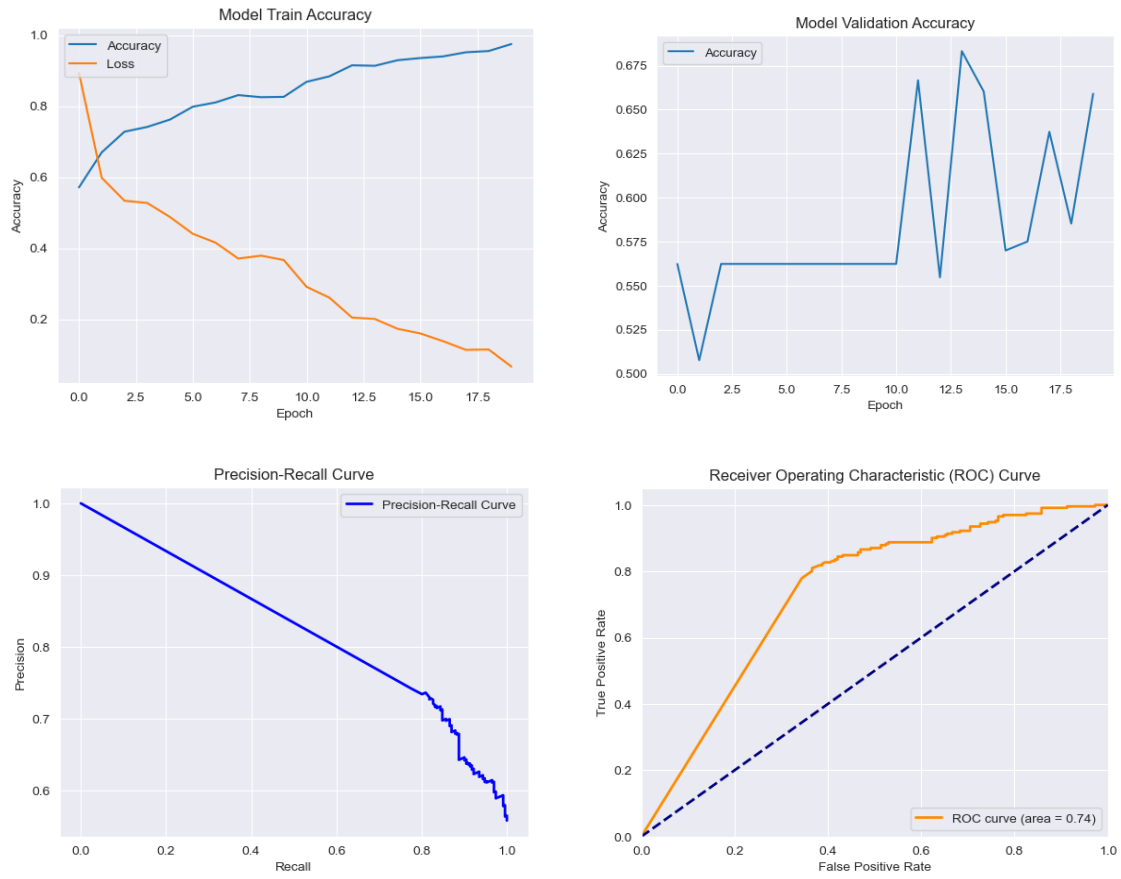
```
   Test Score  Test Accuracy
0    1.106329       0.647343

13/13 [==============================] - 0s 2ms/step
[87]: 0.6473429799079895
```



شکل ۱۱ نمودارهای معماری آزمایشی شماره ۱

۵-۱. معماری آزمایشی ۱

```
    Test Score  Test Accuracy
0    5.623737       0.644928

13/13 [==============================] - 0s 2ms/step
[92]: 0.6449275612831116
```



شکل ۱۲ نمودارهای معماری آزمایشی شماره ۱

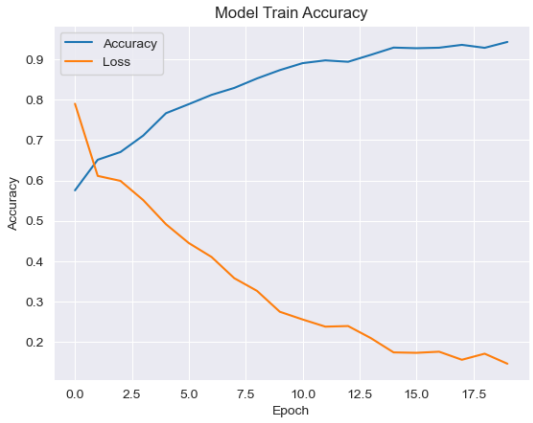
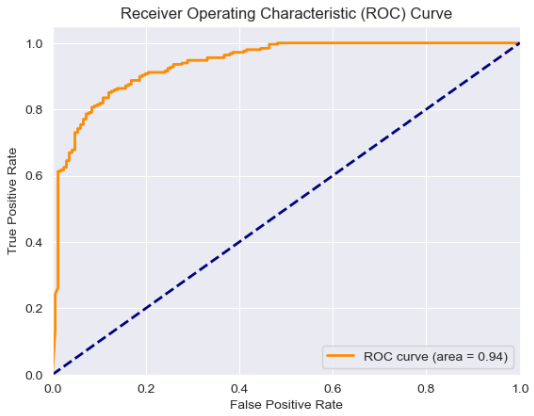
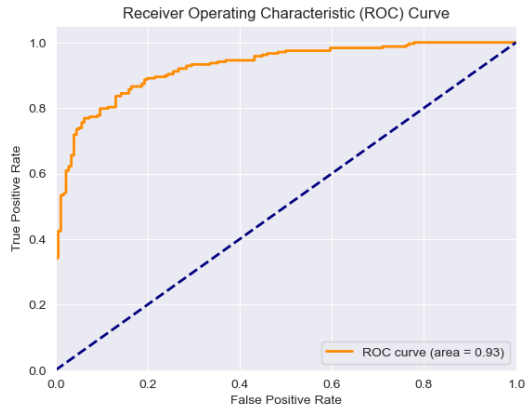
۶-۱. مقایسه معماری‌ها

با مقایسه نمودارهای حاصل شده واضح است که معماری منتخب چه از نظر دقت و چه از نظر شاخص‌های دیگر مانند نمودار ROC عملکرد بهتری داشته است. بین معماری های ۱ و ۲ هم معماری ۱ به نسبت از نظر نمودار ROC عملکرد بسیار بهتری داشته است.

همچنین از نظر تعداد پارامترها، مدل پیشنهادی مقاله نسبت به مدل آزمون ۲ تعداد بسیار کمتری پارامتر داشته و زمان آموزش آن نیز در عین دقت بهتر، کمتر است.

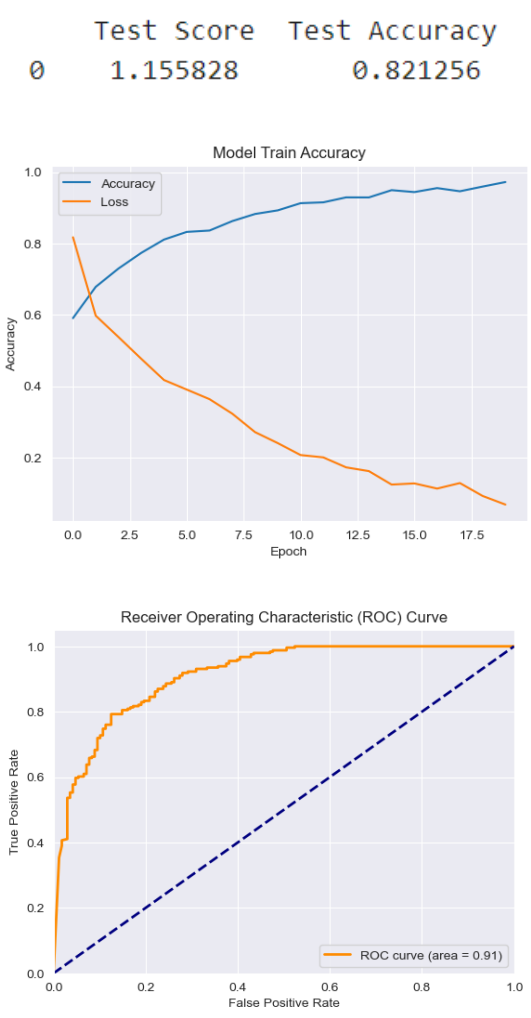
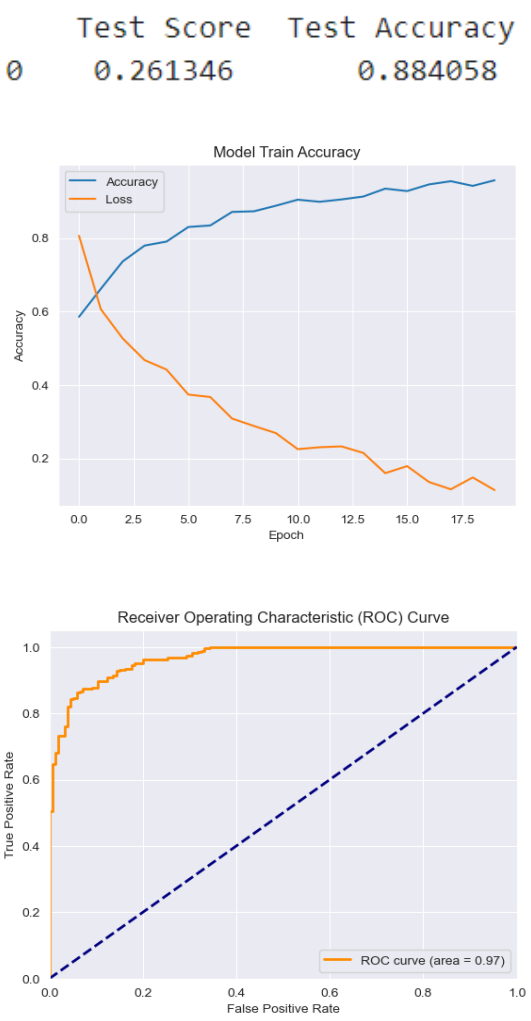
۷-۱. مقایسه معماری‌ها

در انتها اثر پارامترهای آموزش بر روی شبکه را بررسی می کنیم.

Test Size = 0.3	Test Size = 0.5
Test Score Test Accuracy 0 0.703548 0.826087	Test Score Test Accuracy 0 0.716494 0.838164
	
	

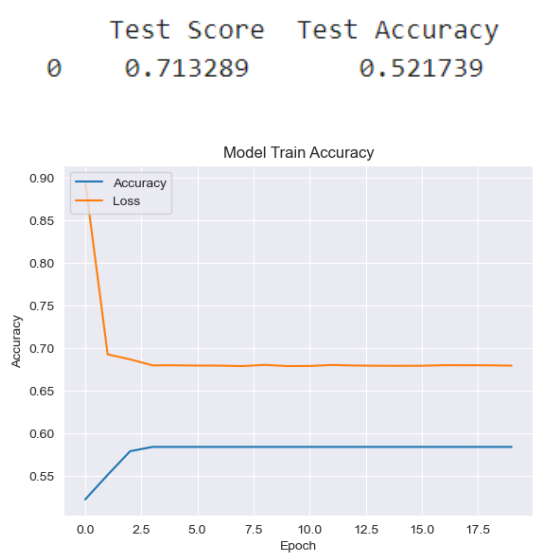
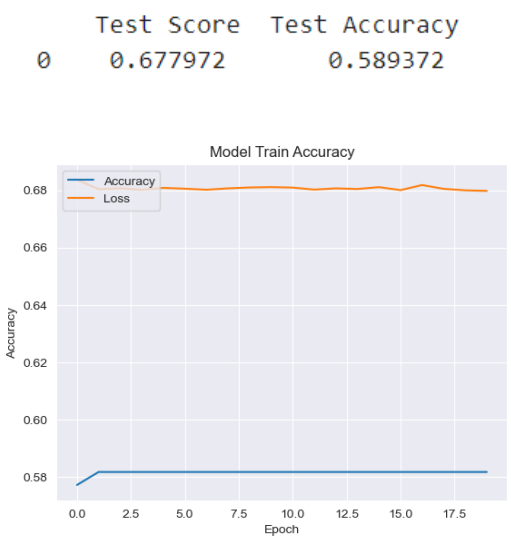
شکل ۱۳ نمودارهای مربوط به تقسیم بندی مجموعه داده

همانطور که مشاهده می‌شود دقت در تقسیم ۰.۵ نسبت به ۰.۳ از ۸۲.۶ به ۸۳.۸ درصد افزایش یافته ولی در نمودار ROC تغییر چشمگیری دیده نمی‌شود.

Without Dropout	With Dropout
	

شکل ۱۴ نمودارهای مربوط به اثر **Dropout**

اثر Dropout در نمودارهای رسم شده در شکل ۱۴ مشاهده می‌شود. با استفاده از Dropout دقت مدل و نمودار ROC و شاخص AUC به طور قابل توجهی افزایش پیدا کرده است.

Ones Initializer	Zeros Initializer
	

شکل ۱۵ نمودارهای مربوط به اثر **Initializer**

در این بخش برای مقایسه اثر Initializer ها نمودارهای مربوط به Zeros و Ones را در شکل ۱۵ رسم کردیم. همانطور که از مقایسه نمودارها با نمودار مربوط به Glorot مشاهده می‌شود استفاده از Glorot اثر چشمگیری در افزایش دقت مدل دارد.

پرسش ۲ــبررسی تاثیرافزایش داده برعملکردشبکه‌های کانولوشنی

۲-۱. پیش‌پردازش تصاویر

با استفاده از ۳ عملکرد ذکرشده در مقاله به فرآیند تولید عکس برای بهبود عملکرد مدل پرداخته‌ایم.

این عملیات شامل چرخش افقی کامل ، بزرگنمایی یا تغییر اندازه تصویر و چرخش با کمی زاویه دادن می‌باشد.

```
Mounted at /content/drive
Number of files in /content/drive/MyDrive/NNDL_HW2_After_Augment/Train/Cats/ After Augmentation : 1400
Number of files in /content/drive/MyDrive/NNDL_HW2_After_Augment/Train/Dogs/ Before Augmentation : 1408
--------------------------------------------
Number of files in /content/drive/MyDrive/NNDL_HW2_After_Augment/ After Augmentation : 2808
```

شکل۱۶ : تعداد تصاویر با برچسب سگ و گربه بعد از عمل اقزودن داده

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
Number of files in /content/drive/MyDrive/NNDL_HW2/HW2_Dataset/Train/Cats: 350
Number of files in /content/drive/MyDrive/NNDL_HW2/HW2_Dataset/Train/Dogs: 352
------------------------------------------------
```

شکل ۱۷ : تعداد تصاویر با برچسب سگ و گربه قبل از عمل افزودن داده

۲-۲. پیاده‌سازی

برای پیاده‌سازی مدل‌های زیر حقیقتا در ابتدای کار با مشکلات زیادی رودرو بودیم . عدم آشنایی کامل با اجرای کد بر روی واحد گرافیکی در گوگل کولب کمی آزاردهنده بود. ابتدا کدهایمان بر روی CPU قرارمی‌دادیم و تا صبح می‌گذاشتیم اجرا شود که معمولا حاصلی نداشت.

در نهایت از مدل کراس برای پیاده‌سازی هر دو مدل ذکرشده در مقاله استفاده کردیم. اما متاسفانه برای مدل رزنت به هیچ عنوان مدل به خوبی نمی‌توانست آموزش ببیند . طبیعتا من نتوانستم به خوبی شبکه رزنت را مدیریت کنم. کراس برای شبکه وی‌جی‌جی دقت‌های قابل‌قبولی را نمایش می‌داد.

بنابراین گزارش بنده برای مدل وی‌جی‌جی از کراس و مدل رزنت از پایتورچ می‌باشد.

برای پایتورچ به این صورت از فرامتغیرها استفاده کردم :

```
optimizer = optim.SGD(model.parameters(), lr=learning_rate
,momentum=momentum)
```

برای فریزکردن پارامترها در ابتدای آموزش از متغیر زیر کمک می‌گیریم :

```
for param in model.parameters():
        param.requires_grad = False
```

```
train_loader=DataLoader(train_dataset, batch_size=batch_size ,shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=batch_size, shuffle=False)
test_loader = DataLoader(test_dataset, batch_size=batch_size,shuffle=False)
```

برای مدل کراس به شرح زیر است :

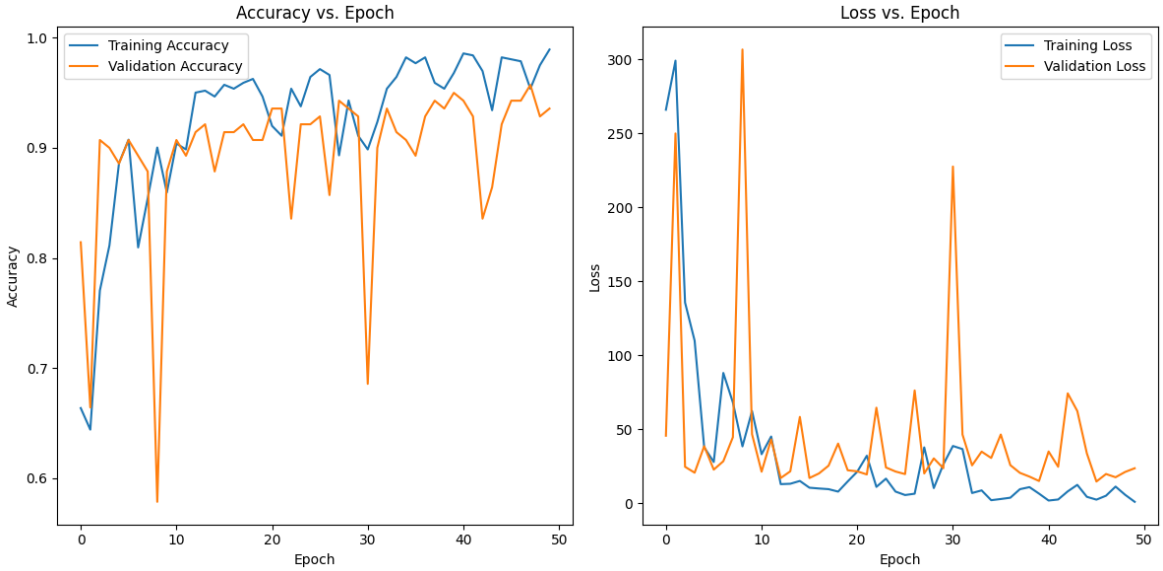
برای فریز کردن وزن‌ها در مدل کراس از متغیر زیر استفاده خواهیم کرد :

```
# Freeze all layers except the new FC layers
for layer in base_model.layers:
    layer.trainable = False
```

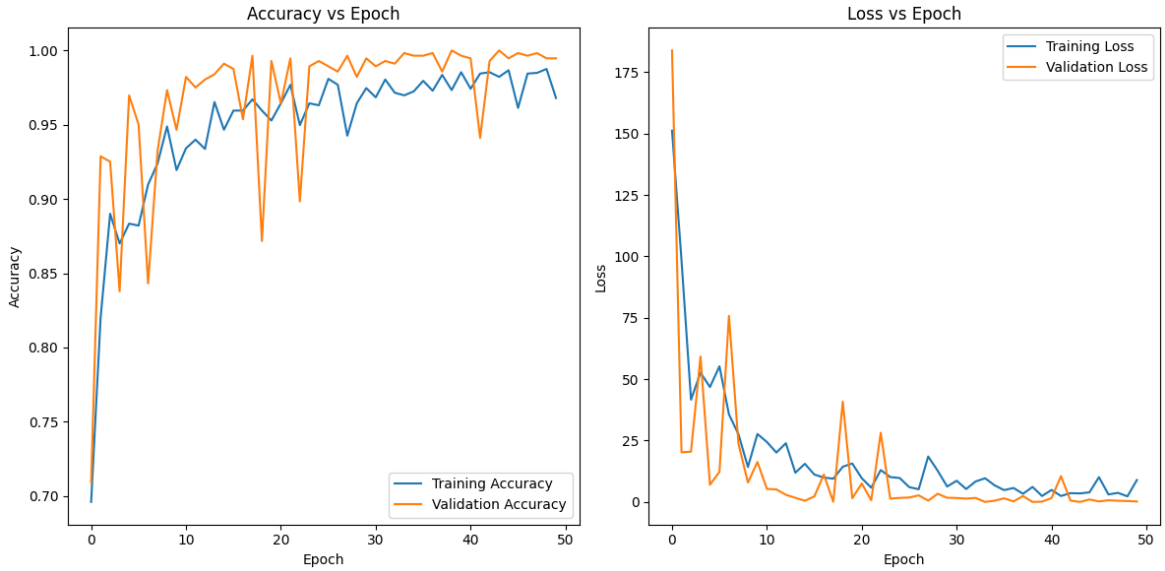
نرخ یادگیری و مومنتوم را به این صورت به مدل استوکستیک گرادیان دیسنت معرفی می‌کنیم:

```
# Compile the model
opt = SGD(learning_rate=initial_lr, momentum=momentum)
```

۳-۲: ResNet



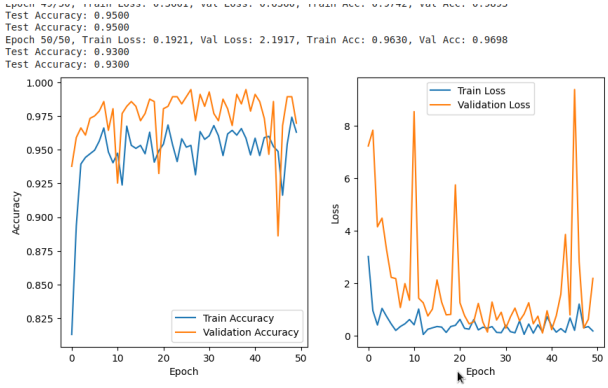
شکل ۱۸ : نمودار دقت و تابع ضرر قبل از افزودن داده – مدل رزنت



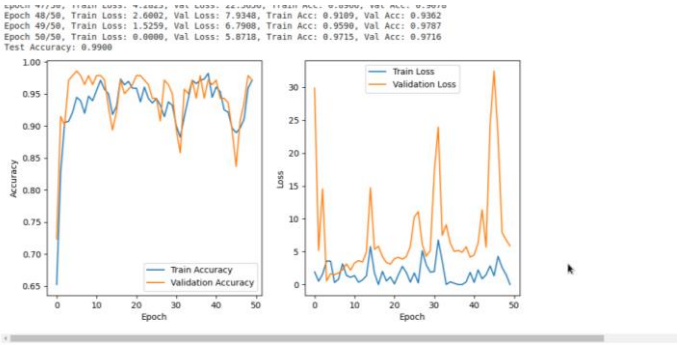
شکل ۱۹ : دقت و تابع ضرر بعد از افزودن داده – مدل رزنت

همانطور که از تصاویر نیز مشخص است ، تاثیر افزایش داده‌ها بر روی عملکرد شبکه‌عصبی کانولوشنی ما مثبت بوده است ، اگرچه هر دو مدل فوق (قبل و بعد افزونگی) از یک الگوی کاهش تابع ضرر پیروی می‌کنند ، اما به مراتب ضرر در مدل بعد از افزونگی کمتر است.(مقیاس نمودار عمودی در قبل از افزونگی بیشتر است.)

همچنین ضرر مجموعه ارزیابی با نوسان کمتری نسبت به حالت قبل از افزونگی تغییر می‌کند.

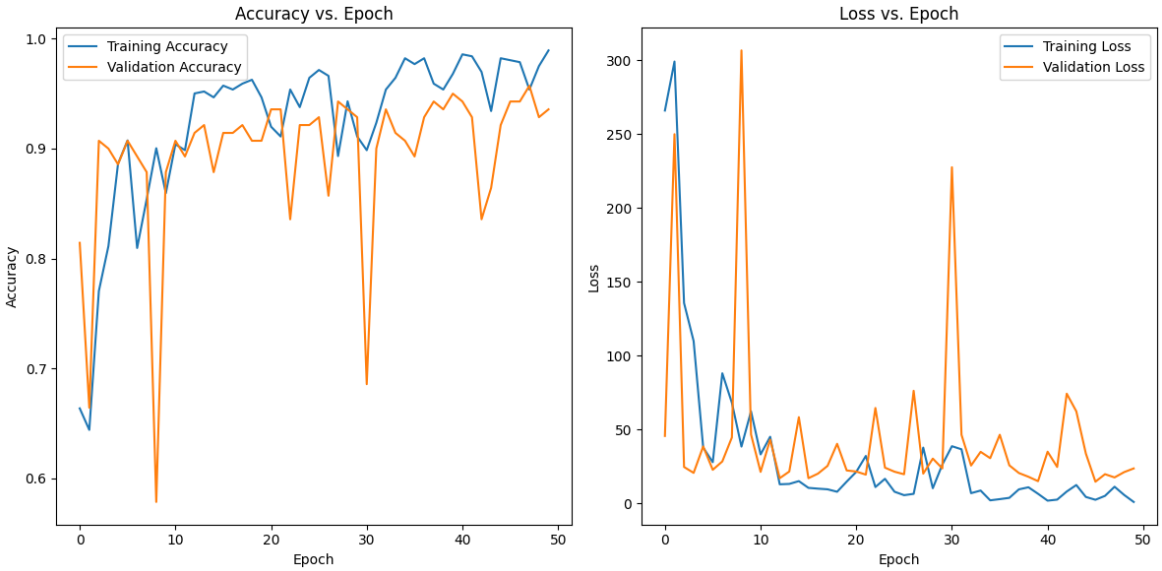


شکل ۲۰ : دقت مجموعه آموزشی ، ارزیابی و تست - قبل از افزودنگی داده - مدل رزنت

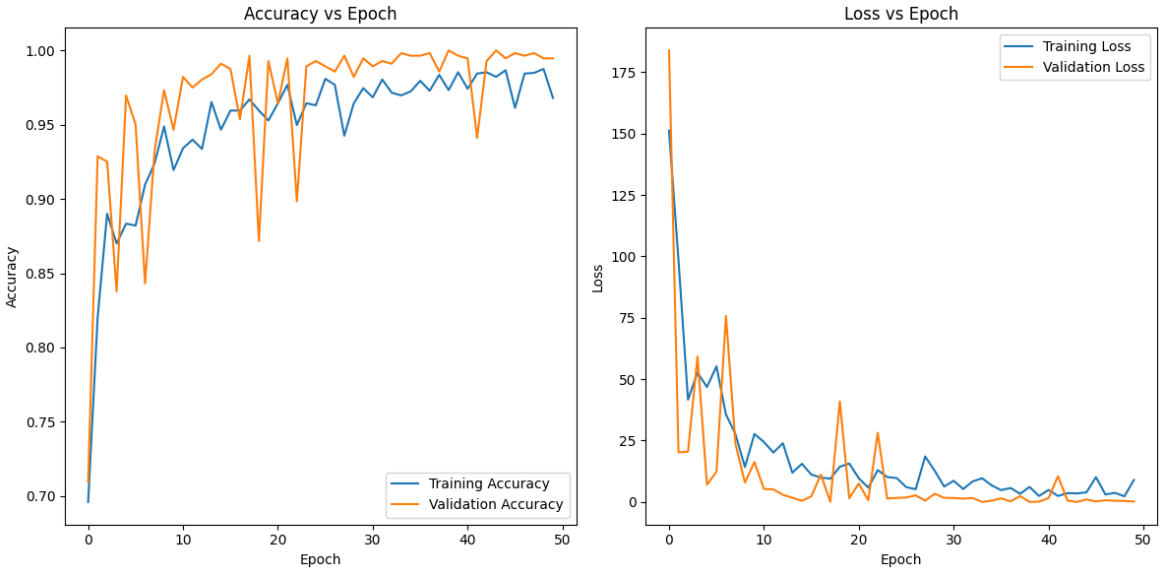


شکل ۲۱ : دقت مجموعه آموزشی ، ارزیابی و تست - بعد از افزونگی داده - مدل رزنت

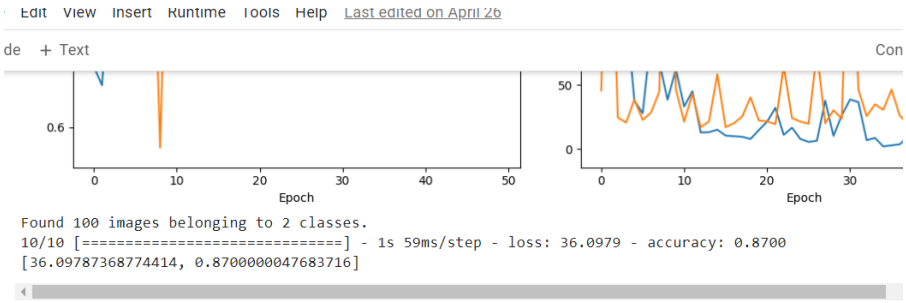
۴-۲: مدل VGG16



شکل ۲۲ : نمودار دقت و تابع ضرر - قبل از افزونگی داده - مدل وی‌جی‌جی



شکل ۲۳ : نمودار دقت و تابع ضرر - بعد از افزونگی داده - مدل وی‌جی‌جی



شکل ۲۴ : دقت تست قبل از افزونگی داده‌ها- مدل وی‌جی‌جی

```
225/225 [==============================] - 50s 221ms/step - loss: 5.5964 - accuracy: 0.9724 - val_loss: 10.2345 - val_accuracy: 0.9554
Epoch 40/50
225/225 [==============================] - 51s 226ms/step - loss: 2.3820 - accuracy: 0.9893 - val_loss: 0.1907 - val_accuracy: 0.9982
Epoch 41/50
225/225 [==============================] - 52s 229ms/step - loss: 2.4638 - accuracy: 0.9884 - val_loss: 1.1950 - val_accuracy: 0.9929
Epoch 42/50
225/225 [==============================] - 50s 222ms/step - loss: 2.4847 - accuracy: 0.9849 - val_loss: 0.6596 - val_accuracy: 0.9929
Epoch 43/50
225/225 [==============================] - 50s 223ms/step - loss: 5.4309 - accuracy: 0.9729 - val_loss: 4.0063e-06 - val_accuracy: 1.0000
Epoch 44/50
225/225 [==============================] - 48s 215ms/step - loss: 1.6343 - accuracy: 0.9884 - val_loss: 0.0082 - val_accuracy: 0.9982
Epoch 45/50
225/225 [==============================] - 52s 231ms/step - loss: 1.3668 - accuracy: 0.9924 - val_loss: 0.0787 - val_accuracy: 0.9982
Epoch 46/50
225/225 [==============================] - 51s 225ms/step - loss: 2.5954 - accuracy: 0.9875 - val_loss: 0.5639 - val_accuracy: 0.9964
Epoch 47/50
225/225 [==============================] - 50s 222ms/step - loss: 5.0558 - accuracy: 0.9826 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 48/50
225/225 [==============================] - 50s 222ms/step - loss: 3.7387 - accuracy: 0.9831 - val_loss: 0.7453 - val_accuracy: 0.9964
Epoch 49/50
225/225 [==============================] - 50s 222ms/step - loss: 0.8033 - accuracy: 0.9964 - val_loss: 0.2695 - val_accuracy: 0.9964
Epoch 50/50
225/225 [==============================] - 51s 225ms/step - loss: 8.1792 - accuracy: 0.9697 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Found 100 images belonging to 2 classes.
10/10 [==============================] - 70s 8s/step - loss: 53.0948 - accuracy: 0.9000
[53.094818115234375, 0.8999999761581421]
```

شکل ۲۵ : دقت مجموعه آموزشی ، ارزیابی و تست بعد از افزونگی داده‌ها - مدل وی‌جی‌جی

۲-۲. تحلیل و نتیجه‌گیری

جدول ۱ : دقت‌های مدل‌های ذکرشده

	VGG16 Model		ResNet Model	
	Without Augmentation	With Augmentation	Without Augmentation	With Augmentation
Train	98%	96%	96%	97%
Validation	93%	100%	96%	97%
Test	87%	90%	93%	99%

به نظرمی‌رسد که بهبود در میزان دقت مدل توسط مدل رزنت به مراتب بیشتر از مدل وی‌جی‌جی بوده‌است.

این مساله از آنجا می‌توان با اطمینان بالاتری به آن اشاره کرد که میزان تعداد دفعاتی که در آزمایش‌های تیم‌ما ، مدل وی‌جی‌جی در **قبل و بعد افزونگی داده هیچ تغییری در میزان دقت تست مشاهده نمی‌شود** ، بالا می‌باشد.

اما مدل رزنت تاثیر قابل توجهی در میزان بهبود دقت حالت تست در آزمایش‌های ما مشاهده‌می‌شود.

حال بهتر است که به تاثیر معماری این دو شبکه در میزان جواب‌های فوق بپردازیم :

مدل وی‌جی‌جی اساسا مدل ساده‌تری به نسبت مدل رزنت هست که این سادگی حتی منجر به ایجاد گرادیان نزدیک به صفر در این شبکه خواهد شد در بسیاری از موارد.

تفاوت در عملکرد هنگام اضافه کردن داده برای آموزش مدل‌های ResNet و VGG16:

ResNet با داشتن اتصالات باقیمانده، آموزش شبکه‌های عمیق را به صورت آسانتر ممکن می‌کند. اضافه کردن داده برای بهبود تعمیم و عملکرد مدل‌های ResNet، به خصوص هنگام آموزش بر روی مجموعه داده‌های بزرگ، مفید است.

رزنت به شبکه این قابلیت را اضافه خواهدکرد که با اضافه‌کردن اتصالات پرش که از یک یا چند لایه عبور می‌کند به گرادیان‌ها این اجازه را می‌دهد که راحت‌تر در شبکه جریان‌پیدا کنند . این راه‌حل کمک‌می‌کند که شبکه به مشکل ناپدیدشدن گرادیان‌ها دچار نشود و ساده‌تر با لایه‌های بیشتر آموزش ببیند.