به نام خدا

تمرین سری ششم - درس مبانی بینایی کامپیوتر

سید محمد علی نخاری - شماره دانشجویی : 99521496

سوال سوم)

الف) در ابتدا دیتاست مربوط را لود کرده و درون متغیر های تعریف شده میریزیم. اگر از داده های مورد نظر shape گرفته و در خروجی چاپ کنیم به صورت زیر خواهد بود:

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 13s 0us/step
Training: (50000, 32, 32, 3) (50000, 1)
Validation: (10000, 32, 32, 3) (10000, 1)
```

همانطور که مشخص است در این دیتاست، 50000 داده آموزشی با اندازه 3*32*32 و 10000 داده که به منظور ارزیابی عملکرد مدل آموزش یافته به کار میرود با همان اندازه داده های آموزشی، وجود دارد.

در گام بعدی، چند نمونه از داده های آموزشی را به همراه برچسب هایشان نمایش میدهیم.

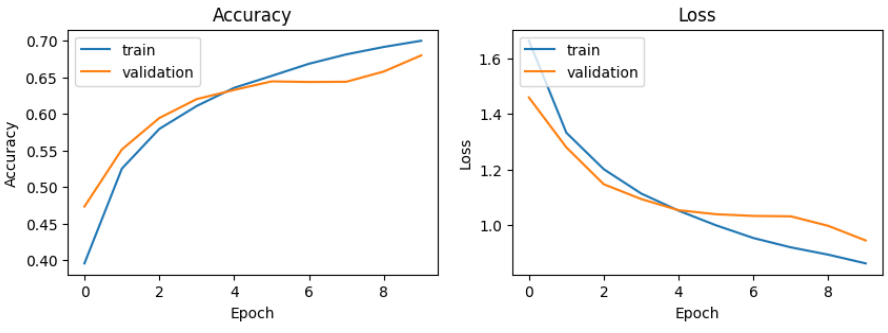
در گام بعد داده های آموزشی و validation را با تقسیم کردن بر 255 و تبدیل مقادیر پیکسل ها به بازه 0 تا 1 آنها را نرمالایز و برچسب های داده های آموزشی را به فرمت one-hot تبدیل میکنیم. اگر خروجی برای چند عکس از داده های آموزشی را نمایش دهیم به صورت زیر خواهد بود:

پس از انجام چند گام پیش پردازشی، شبکه عصبی ساده ای برای دسته بندی تصاویر میسازیم. برای اینکار ابتدا یک مدل از نوع sequential ساخته و لایه ورودی با اندازه تصاویر دیتاست ایجاد میکنیم. لایه بعدی یک لایه کانولوشنی با 32 فیلتر و اندازه 3*3 و تابع فعالسازی relu است که باعث میشود مقادیر خطی ایجاد شده به مقادیر غیر خطی تبدیل شوند و پس از آن با استفاده از یک لایه maxpooling و گام دو، ابعاد خروجی را از لحاظ مکانی کوچک میکنیم. همین مراحل را تکرار کرده و در آخر با استفاده از یک لایه flatten که آخرین لایه برای مدل ساخته شده است خروجی ها را دسته بندی میکنیم. (تابع فعالسازی برای اینکه لایه تابع softmax است که احتمال تعلق هر تصویر به ده کلاس را به دست آوریم)

خلاصه ای از مدل ساخته شده به صورت زیر است:

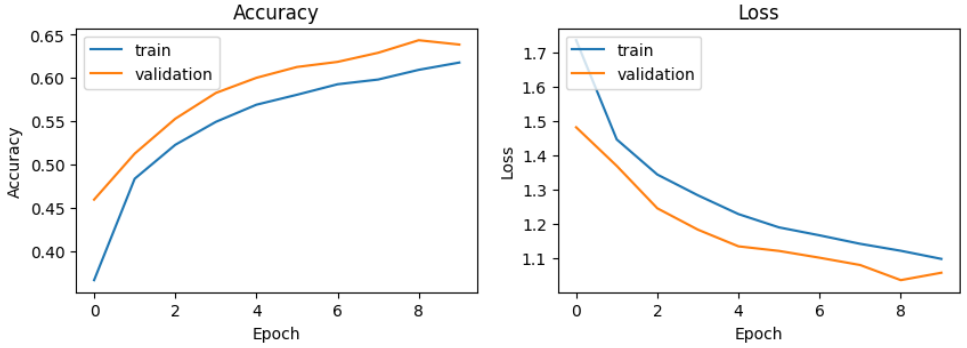
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896

 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        51264

 max_pooling2d_1 (MaxPooling  (None, 3, 3, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 1, 1, 64)          36928

 flatten_1 (Flatten)         (None, 64)                0

 dense_2 (Dense)             (None, 10)                650

=================================================================
Total params: 89,738
Trainable params: 89,738
Non-trainable params: 0
```

پس از ساختن مدل، آن را کامپایل کرده و آموزش مدل را با صدا زدن تابع fit و دادن داده های آموزشی و داده های تست آغاز میکنیم. نتیجه نهایی به صورت زیر خواهد بود:



همانطور که در نمودار های بالا قابل مشاهد است، داده های آموزشی نسبت به داده های تست عملکرد بهتری از خود نشان داده اند و این یعنی شبکه آموزش داده شده داده های آموزشی را حفظ کرده و با این کار توانسته نتایج بهتری به دست بیاورد.

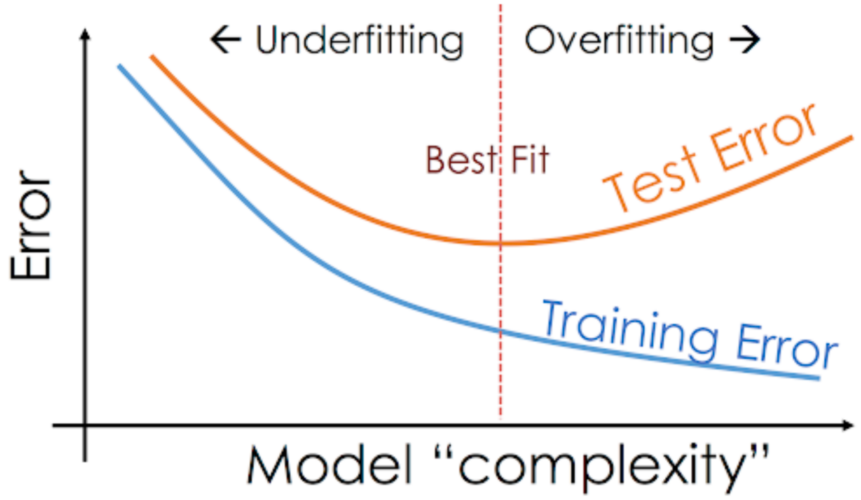
ب) همانطور که در قسمت قبل دیدیم، شبکه با حفظ کردن داده های آموزشی توانست نتیجه بهتری نسبت به داده های تست بگیرد که اصلا خوب نیست و شبکه با دیدن تصویر جدید نمیتواند تشخیص خوبی داشته باشد. یکی از راه حل ها برای این مشکل استفاده از تکنیک داده افزایی است. در این روش به دیتاست موجود یک سری تصویر که از لحاظ محتوایی مشابه با تصاویر موجود است اما از نظر ظاهری تفاوت دارند(چرخش یافته تصاویر، زوم شده تصاویر، قرینه شدن و ...) با این کار شبکه نمیتواند به راحتی داده های آموزشی را حفظ کند.
چند مورد از ویژگی هایی که در این سوال استفاده شده است از جمله horizontal_flip, rotation و zoom میباشد. نتیجه به دست آمده در این قسمت به صورت زیر میباشد:



همانطور که در نمودار به دست آمده واضح است در این قسمت شبکه بر روی داده های آموزشی بهتر عمل کرده و صرفا نتایج را حفظ نکرده و توانسته با دیدن تصاویر جدید تشخیص و پیشبینی بهتری داشته باشد.

ج) با توجه به نمودار قسمت الف همانطور که گفته شد شبکه در حین آموزش توانست داده های آموزشی را تا حد خوبی به درستی تشخیص دهد اما در تشخیص داده ها تستی که تا به حال با آنها مواجه نشده بود، عملکرد خوبی نداشت، یا به عبارتی شبکه با over-fitting مواجه شده بود. اما در قسمت "ب" با پیچیده کردن مدل سعی در این داشتیم که شبکه نتواند داده های آموزشی را حفظ

کند و نتیجه را تشخیص دهد بلکه با سخت تر کردن مدل، نتایج داده های آموزشی به نسبت قسمت قبل عملکرد معقول تری داشته باشد و با دیدن داده جدید بتواند تشخیص درستی بدهد. و با توجه به نمودار نتیجه دیدیم که تا حدودی این کار جواب داد و شبکه توانست عملکرد بهتری را در مواجه با داده ها جدید داشته باشد و از over-fitting تا حد امکان، جلوگیری کردیم.



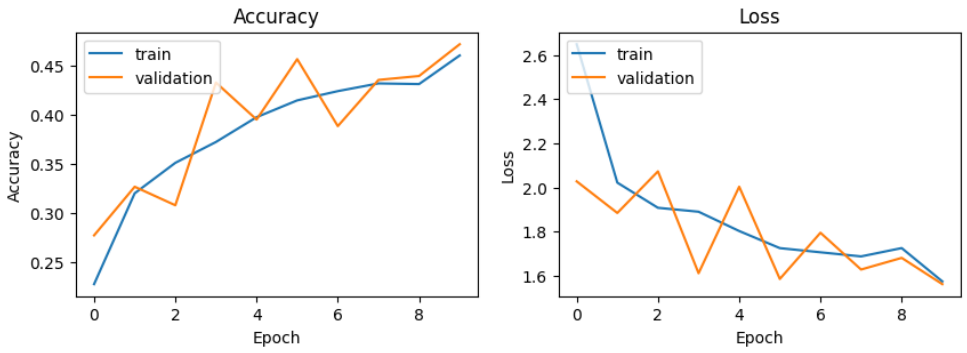
منظور از under-fitting این است که مدل طراحی شده آنقدر بد است که شبکه هم بر روی داده های آزمایشی و هم داده های تستی عملکرد بدی دارد که خوشبختانه در این سوال با این مشکل مواجه نشدیم.

د) در این قسمت ابتدا نیاز داشتیم که مدل از پیش آموزش دیده شده ResNet50 را با استفاده از keras دانلود کرده و درون متغیر resnet50 بریزیم. پس از این کار به دلیل آنکه که گفته شده باید تنها از همین مدل برای آموزش تصاویر دیتاست cifar10 استفاده کنیم، ابتدا لایه های قابل آموزش در این مدل را با پیمایش بر روی آنها به حالت freeze بردیم. در ادامه یک مدل sequential ساخته و لایه ای ایجاد میکنیم که وظیفه آن تبدیل تصویر ورودی به ابعاد مورد نظر در مدل ResNet50 میبریم.(224*224*3) در ادامه همان مدل resnet50 را به مدل ساخته شده اضافه کرده و با استفاده از یک لایه کاملا متصل عملیات دسته بندی را بر روی داده های ورودی انجام میدهیم.

نتیجه به دست آمده از این مدل ساخته شده به صورت زیر است:

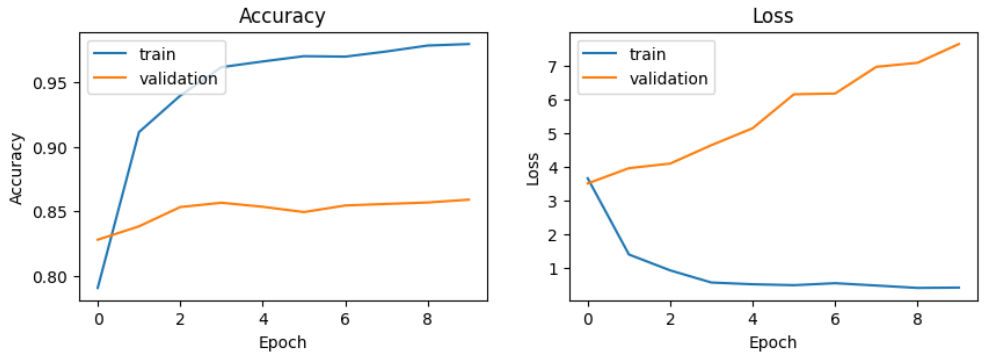
```
Epoch 1/10
500/500 [==============================] - 172s 336ms/step - loss: 2.4111 - accuracy: 0.2039 - val_loss: 2.0526 - val_accuracy: 0.2092
Epoch 2/10
500/500 [==============================] - 166s 333ms/step - loss: 1.9065 - accuracy: 0.3046 - val_loss: 1.9761 - val_accuracy: 0.2865
Epoch 3/10
500/500 [==============================] - 166s 333ms/step - loss: 1.7972 - accuracy: 0.3454 - val_loss: 1.7142 - val_accuracy: 0.3848
Epoch 4/10
500/500 [==============================] - 180s 361ms/step - loss: 1.7199 - accuracy: 0.3729 - val_loss: 1.6925 - val_accuracy: 0.3855
Epoch 5/10
500/500 [==============================] - 180s 361ms/step - loss: 1.6693 - accuracy: 0.3951 - val_loss: 1.6748 - val_accuracy: 0.3779
Epoch 6/10
500/500 [==============================] - 167s 333ms/step - loss: 1.6266 - accuracy: 0.4127 - val_loss: 1.6497 - val_accuracy: 0.3932
Epoch 7/10
500/500 [==============================] - 166s 333ms/step - loss: 1.5945 - accuracy: 0.4229 - val_loss: 1.5376 - val_accuracy: 0.4421
Epoch 8/10
500/500 [==============================] - 167s 333ms/step - loss: 1.5612 - accuracy: 0.4365 - val_loss: 1.5494 - val_accuracy: 0.4314
Epoch 9/10
500/500 [==============================] - 166s 333ms/step - loss: 1.5502 - accuracy: 0.4369 - val_loss: 1.5415 - val_accuracy: 0.4396
Epoch 10/10
500/500 [==============================] - 180s 360ms/step - loss: 1.5257 - accuracy: 0.4499 - val_loss: 1.4691 - val_accuracy: 0.4675
```

همانطور که مشاهده میشود با توجه به پیچیده بودن شکبه مدل ResNet50 به نسبت شبکه های ساده ساخته شده، زمان بیشتری را صرف آموزش کرده و در نهایت به نتیجه مطلوبی هم نرسیده است. از جمله دلایل بهبود نیافتن نتیجه به نسبت حالت های قبل میتواند تغییر اندازه تصویر های دیتاست cifar10 باشد چرا که این تصاویر 32*32*3 هستند و وقتی آنها را به 224*224*3 تغییر میدهیم، ممکن است اطلاعاتی از تصویر از بین برود و به همین دلیل مدل نتواند عملکرد خوبی داشته باشد. یکی از دلایل دیگر میتوان به اختلاف در تعداد دسته بندی های انجام شده در دو دیتاست است که در دیتاست ResNet50 با 1000 کلاس اما در دیتاست cifar10 با 10 کلاس مواجه هستیم. نمودار این قسمت هم به صورت زیر میباشد:

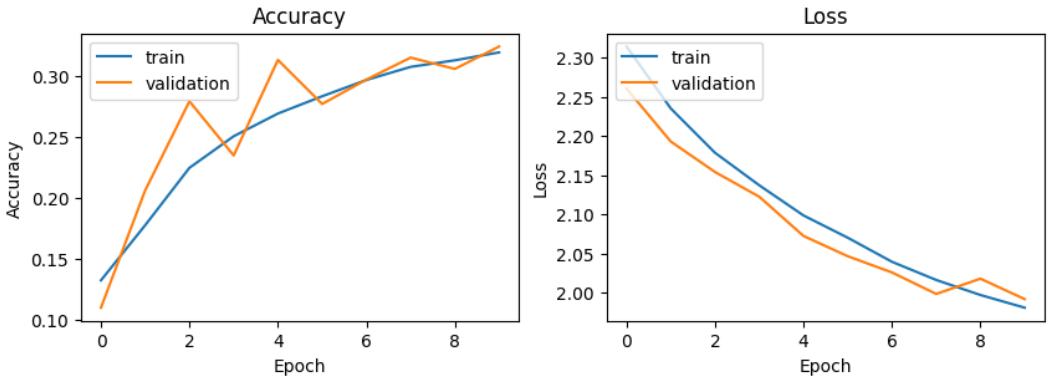


همانطور که مشخص است مدل در این حالت اصلا عملکرد خوبی ندارد و مقادیر مربوط به داده های تست دائما در حال تغییر است.

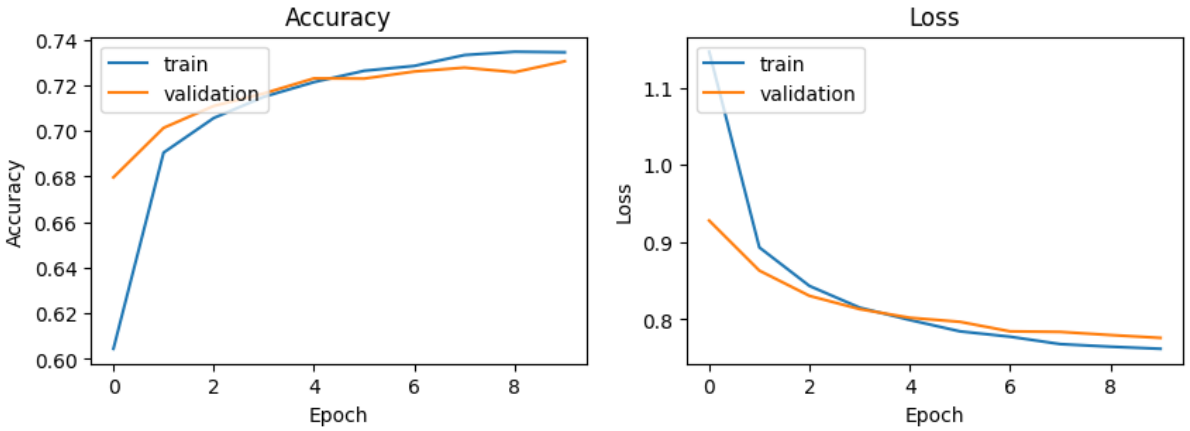
مورد دیگری که با آن برخورد داشتم این بود که زمانیکه از داده های نرمالایز شده برای آموزش استفاده میکردم نتیجه مطلوب نبوده و عمدتا به صورت بالا خروجی میداد اما زمانی که داده های دیتاست را بدون نرمالایز کردن به شبکه میدادم نتیجه برای داده های تست مطلوب تر(صرفا مقدار accuracy بهتر میشد) و به صورت زیر در می آمد اما همچنان با مشکل over-fitting در این قسمت مواجه هستیم و داده های آموزشی عملکرد خیلی بهتری از داده های تست از خود نشان میدهند :



ه) در این قسمت هم ابتدا داده های نرمالایز شده را به مدل ساخته شده که تنها تا لایه سوم از مدل resnet50 را شامل میشود داده و نتایج را محاسبه میکنیم. همانطور که در نمودار پایین قابل مشاهده است مدل عملکرد خوبی را از خود به جای نمیگذارد:



مشاهده میشود که هم داده های آموزش و هم داده های تست اصلا به خوبی در شبکه ساخته شده، عمل نکرده اند. در قسمت بعد داده هایی که نرمالایز نشده اند را مستقیماً به مدل مورد نظر داده و با استفاده از یک لایه batch normalization داده ها را نرمالایز کرده و آموزش میدهیم. نتیجه برای epoch 10 به صورت زیر در آمد:



در این حالت همانطور که مشخص است نتیجه هم برای داده های آموزشی و هم داده های تست عملکرد نزدیک به هم و خوبی داشته اند و از over-fitting تا حد خوبی جلوگیری شده است.