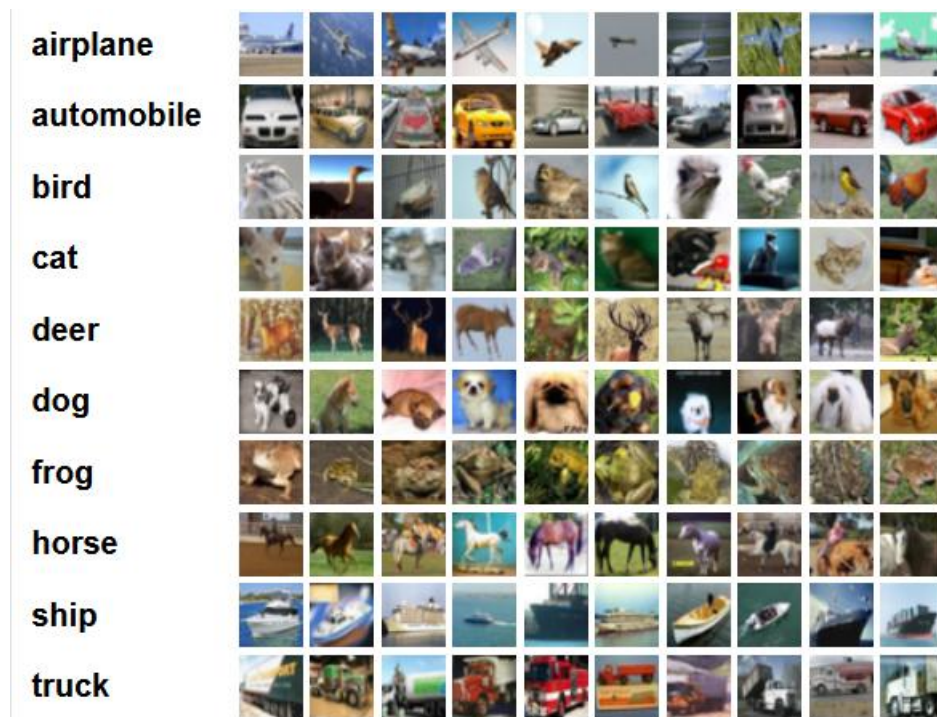


## به نام خداوند بخشنده و مهربان

### گزارش پروژه پایانی در بینایی ماشین

ابتدا نیاز است درمورد دیتاست مورد استفاده CIFAR10 صحبت بشه، این دیتاست مجموعه کاملی شامل ۶۰۰۰۰ تصویر رنگی ۳۲x۳۲ است. ما دیتا رو به ۵۰۰۰۰ تصویر آموزش (train) و ۱۰۰۰۰ تصویر تست (test) تقسیم کردیم. کلاس های مختلف با ۱۰ برچسب وجود دارند.



هر کلاس دقیقا ۶۰۰۰ تصویر دارد ۵۰۰۰ آموزش و ۱۰۰۰ تست

هدف استخراج ویژگی های سلسله مراتبی از تصاویر است.

سه لایه کانکلوشنی داریم ۱۶ لایه سه در سه ، ۳۲ لایه سه در سه و ۶۴ لایه سه در سه

از کرنل (kernel) های سه در سه استفاده شده چون برای تشخیص ویژگی های محلی عملکرد بهتری دارد. سعی شده افزایش تعداد فیلتر ها تدریجی باشد تا به یادگیری ویژگی های پیچیده تر کمک کند.

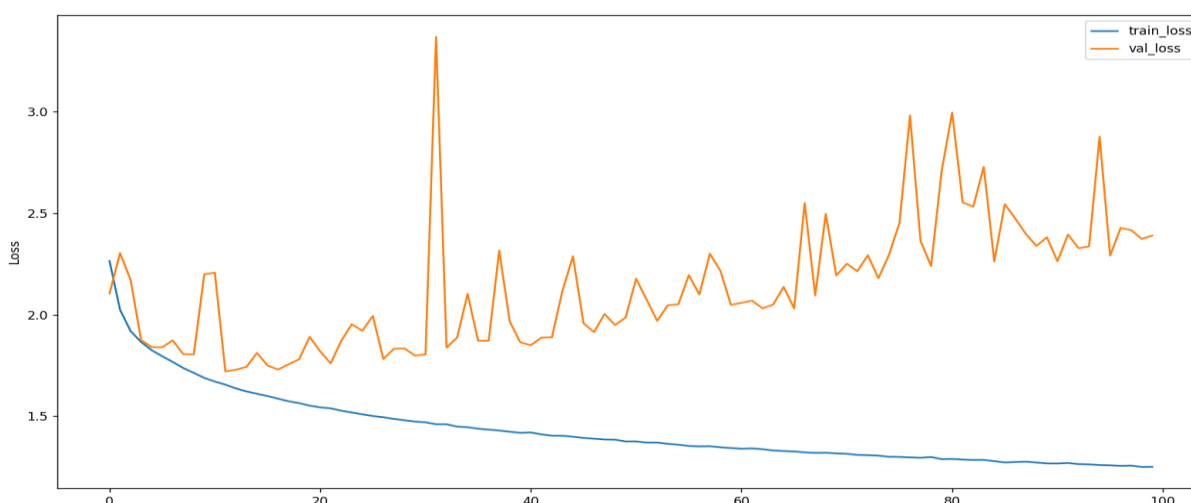
## لایه pooling

از MaxPooling برای کاهش ابعاد و حفظ ویژگی ها استفاده شده است. در لایه آخر هم برای یکپارچگی ویژگی ها از averagePooling استفاده شده است.

در لایه fully connected هم تبدیل ویژگی های استخراج شده به برچسب اتفاق افتاده است.

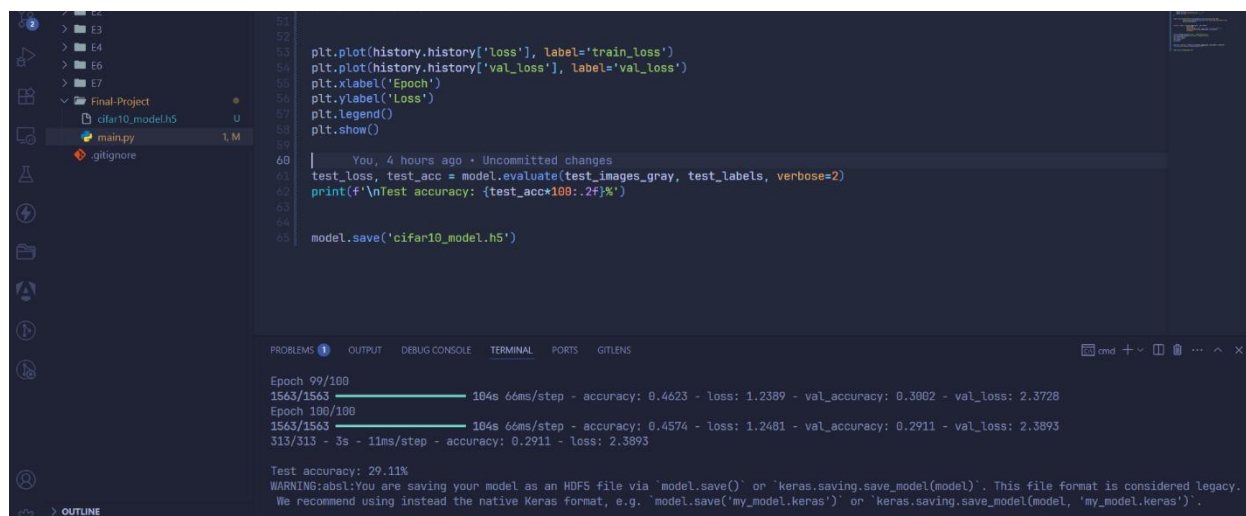
در مورد مفهوم batch size میتوان گفت تعداد نمونه هایی که قبل از به روزرسانی وزن ها پردازش می شوند. در ابتدا این مقدار را ۳۲ قرار دادم که به نسبت مقدار گفته شده در جدول دوم صورت سوال (۸ عدد) بیشتر بود مقدار بیشتر باعث کاهش میزان نویز خواهد شد این کار (افزایش مقدار batch size) میزان مصرف حافظه را بسیار بالا برد. مقدار کم batch size باعث همگرایی سریع تر میشود ولی مقدار نویز را افزایش خواهد داد.

در ادامه نمودار میزان Loss شبکه در طی ۱۰۰ اپیاک نشان داده میشود.



روند کاهشی مقدار train-loss امیدوار کننده است که مدل در حال یادگیری است اما حالت نوسانی val\_loss و روند ره به رشد آن در مجموع دو خطر مهم را گوش زد میکند.

اول اینکه احتمال overfitting وجود دارد و دوم اینکه این نوسانات میتواند به این دلیل باشد که مدل از نقاط بهینه رد میشود البته می تواند به مقدار batch size هم مربوط باشد.



```
51 plt.plot(history.history['loss'], label='train_loss')
52 plt.plot(history.history['val_loss'], label='val_loss')
53 plt.xlabel('Epoch')
54 plt.ylabel('Loss')
55 plt.legend()
56 plt.show()
57
58 You, 4 hours ago · Uncommitted changes
59 test_loss, test_acc = model.evaluate(test_images_gray, test_labels, verbose=2)
60 print(f'\nTest accuracy: {test_acc*100:.2f}%')
61
62 model.save('cifar10_model.h5')
63
```

Epoch 99/100  
1563/1563 — 104s 66ms/step - accuracy: 0.4623 - loss: 1.2389 - val\_accuracy: 0.3902 - val\_loss: 2.3728  
Epoch 100/100  
1563/1563 — 104s 66ms/step - accuracy: 0.4574 - loss: 1.2481 - val\_accuracy: 0.2911 - val\_loss: 2.3893  
313/313 - 3s - 11ms/step - accuracy: 0.2911 - loss: 2.3893  
Test accuracy: 29.11%  
WARNING:absl:You are saving your model as an HDF5 file via 'model.save()' or 'keras.saving.save\_model(model)'. This file format is considered legacy. We recommend using instead the native Keras format, e.g. 'model.save('my\_model.keras')' or 'keras.saving.save\_model(model, 'my\_model.keras')'.

مقدار test accuracy ما ۲۹/۱۱ درصد بود است که مقدار پائینی است دلیل این مقدار پائین میتواند به دلیل خاکستری کردن تصویر باشد چون ممکن است مقداری از اطلاعات را با این تبدیل از دست داده باشیم. میتوان با اصلاحاتی به نسخه بهتری دست پیدا کرد.

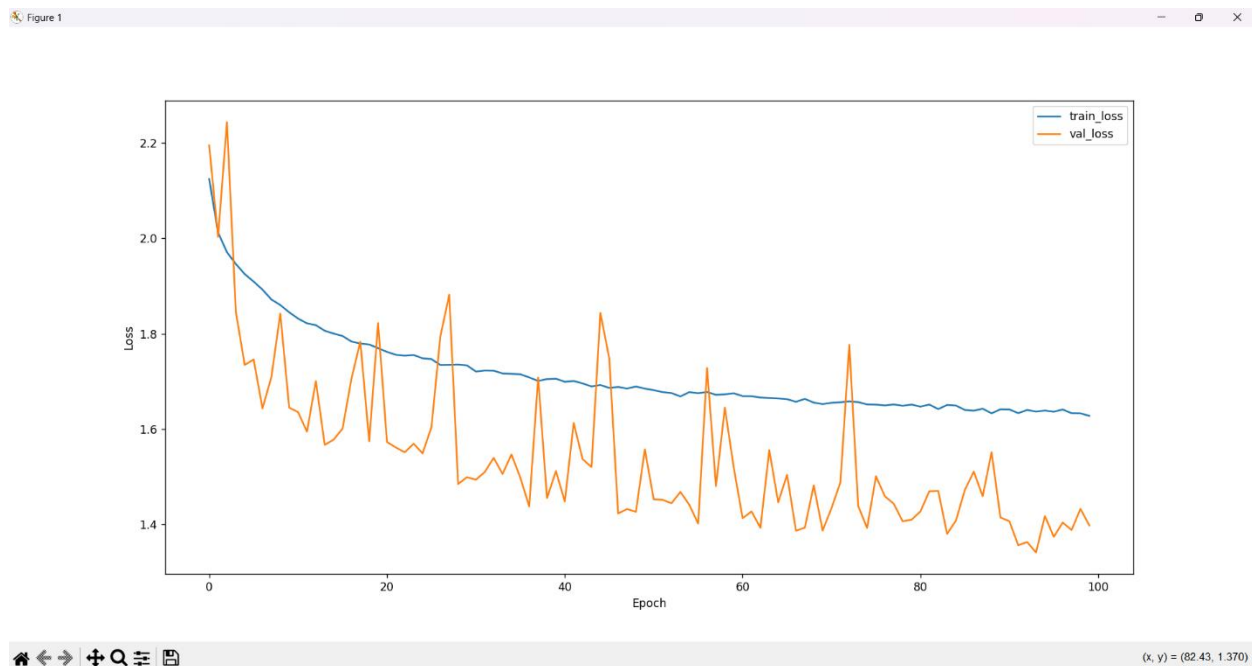
در هر لایه کانکلوشنی طبق جدول داده شده از ReLU و AveragePooling استفاده شده است. از averagePooling برای کاهش ابعاد با میانگیری به جای ماکزیمم استفاده شده است.

در ادامه هم طبق مقادیر داده شده برای مدل طبق جدول دوم نوشته شده است. batch size برابر ۸ و با learning rate ۰/۰۰۱ انجام شده است.

در ادامه هم رسم نمودار انجام شده.

در نمونه کد دوم سعی شد میزان accuracy افزایش داده شود برای این کار با روش data augmentation سعی شد که به مدل در یادگیری کمک شود.

در ادامه هم سعی شد با استفاده از GlobalAveragePooling2D از وقوع overfitting جلوگیری شود.



در این نمودار دیده میشود که هم مقدار `train_loss` و هم `val_loss` هر دو روند نزولی داشته اند که این نکته خوبی است و این یعنی مدل در حال یادگیری است. در این کد جدید میزان `accuracy` با یک بار اجرا به ۵۱ درصد رسید.

```

50     metrics=['accuracy'])
51
52
53 history = model.fit(train_images, train_labels,
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
Epoch 99/100
6250/6250 — 88s 14ms/step - accuracy: 0.4185 - loss: 1.6370 - val_accuracy: 0.4913 - val_loss: 1.4325
Epoch 100/100
6250/6250 — 87s 14ms/step - accuracy: 0.4192 - loss: 1.6257 - val_accuracy: 0.5110 - val_loss: 1.3976
313/313 - 3s - 9ms/step - accuracy: 0.5110 - loss: 1.3976

Test accuracy: 51.10%
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.save_model(model)`. This file format is considered legacy.
We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.save_model(model, 'my_model.keras')`.

```