cifar۱۰

محمدرضا صاحب زاده


دانشگاه علم و صنعت ایران

فهرست مطالب

۱ CNN

۱.۱ تعریف

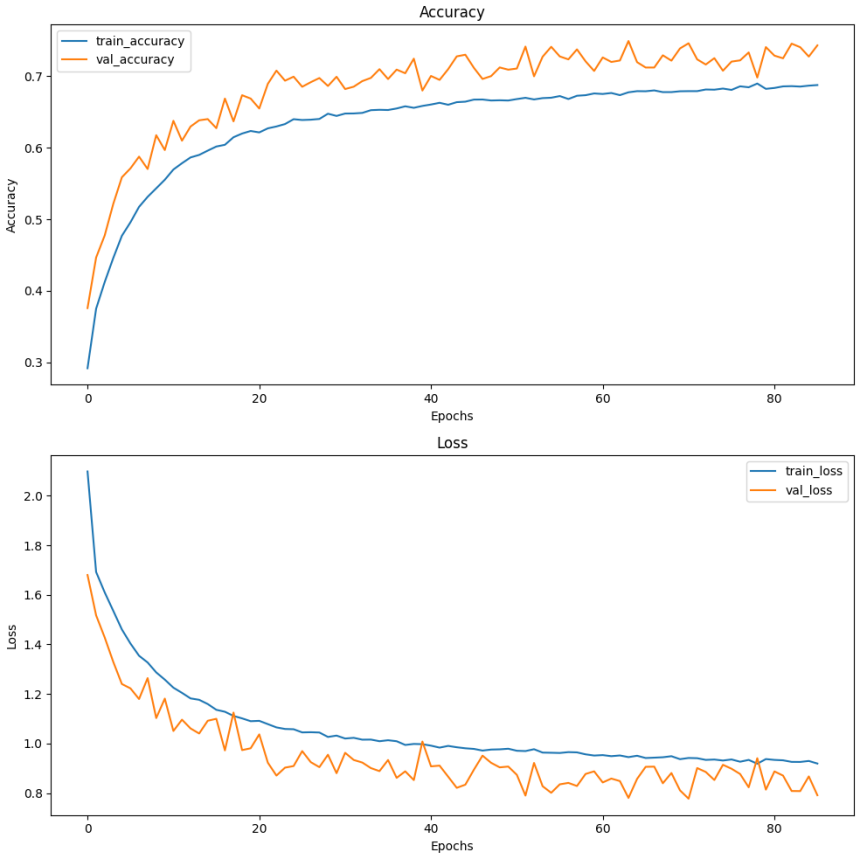
طبق شکل ۱ شبکه های کانولوشنی به این ترتیب هستند که در لایه های اول ویژگی های ساده ای مانند رنگ و لبه یابی و ... انجام میشود و در لایه های عمیق تر ویژگی های سطح بالایی مانند چشم انسان و تایر ماشین تشخیص داده میشوند در لایه های pooling max هم با انتخاب بیشترین مقدار در هر پنجره کمک به کم شدن پارامتر ها میکند

```
model = Sequential([
    Conv2D(32, (3, 3), activation=model_params['activation_functions'][0], input_shape=(32, 32, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation=model_params['activation_functions'][1]),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3),  activation=model_params['activation_functions'][2]),
    Flatten(),
    Dense(64,  activation=model_params['activation_functions'][3]),
    Dense(10,  activation=model_params['activation_functions'][4])
])
```

شکل ۱:

همانطور که در؟؟می‌بینید به نظرم آمد که خیلی به اورفیت نزدیک نشده پس حدس زدم با مقدار صبر بیشتر به دقت بیشتری میرسیم پس مقدار صبر برای خاتمه فرایند آموزش را از عدد شش به عدد پانزده تغییر دادم و دوباره آموزش را شروع کردم

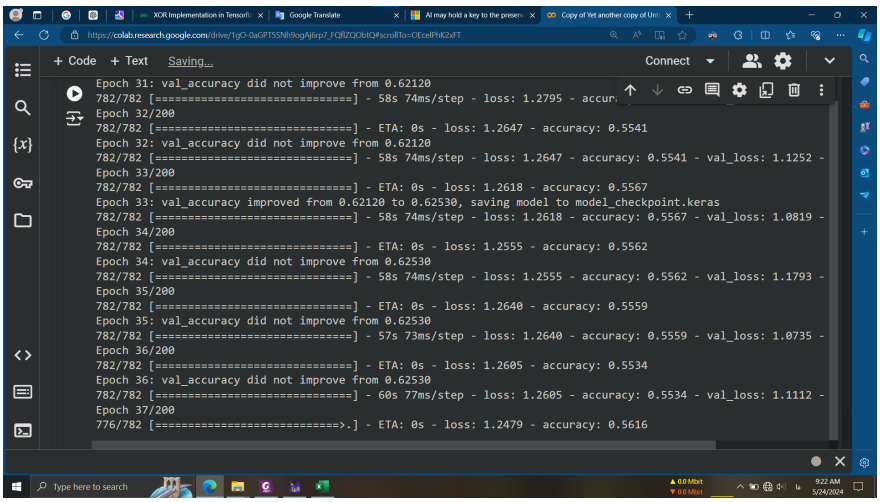
همانطور که در شکل ۲ میبینیم با صرف شدن وقت زیاد و ایپاک های خیلی بیشتر ، دقت ما ۴ درصد بهبود یافت



شکل ۲:

در مرحله بعدی ما تعداد توابعی که باعث افرایش داده میشود را بیشتر کردیم و به دلیل افزایش داده خیلی کند در حال پیشروی بود

ولی اطلاعاتش سیو نشد به دلیل تمام شدن جی پی یو

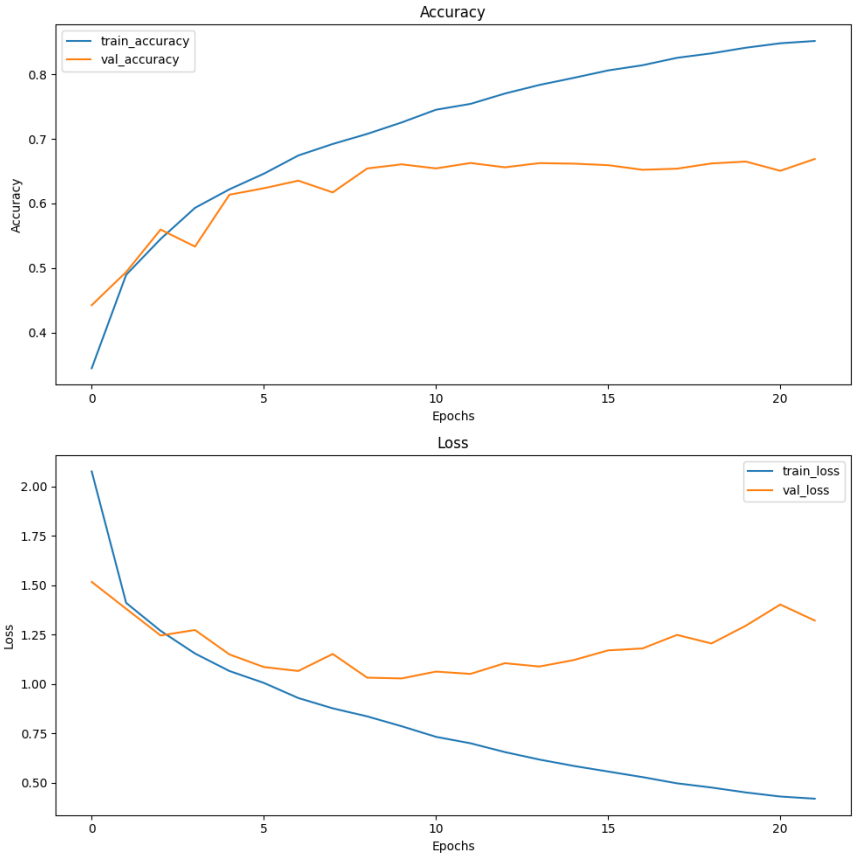


شکل ۳:

model 5 ۲.۱

در مدل شماره پنج به این موضوع پی میبرم که بهتر است به جای اینکه val accuracy را مانیتور کنم val loss را مانیتور کنم

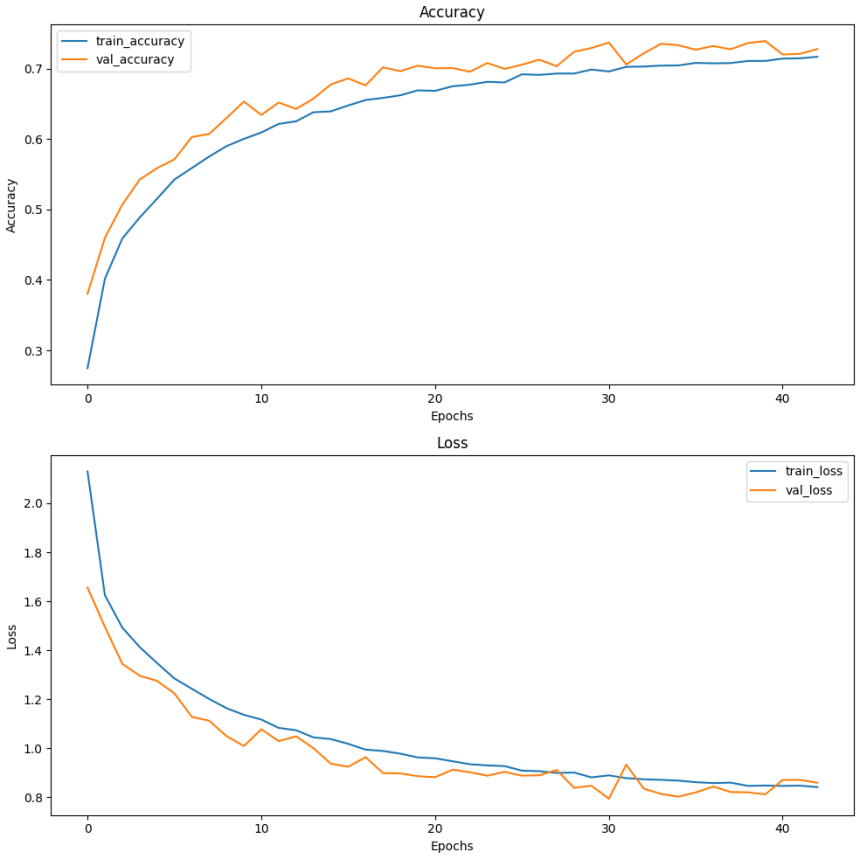
همانطور که در شکل ۴ میبینید در مدل شماره ۵ ما به کلی اضافه کردن دیتا را پاک کردیم و به همین دلیل هم ایپاک ها سریع جلو میرفتند ولی خب دقت هم به جای خوبی نرسید



شکل ۴:

model 6 ۳.۱

در این مرحله با دوباره اضافه کردن قسمت data augmentatoin باز فرق خاصی حس نمیشود

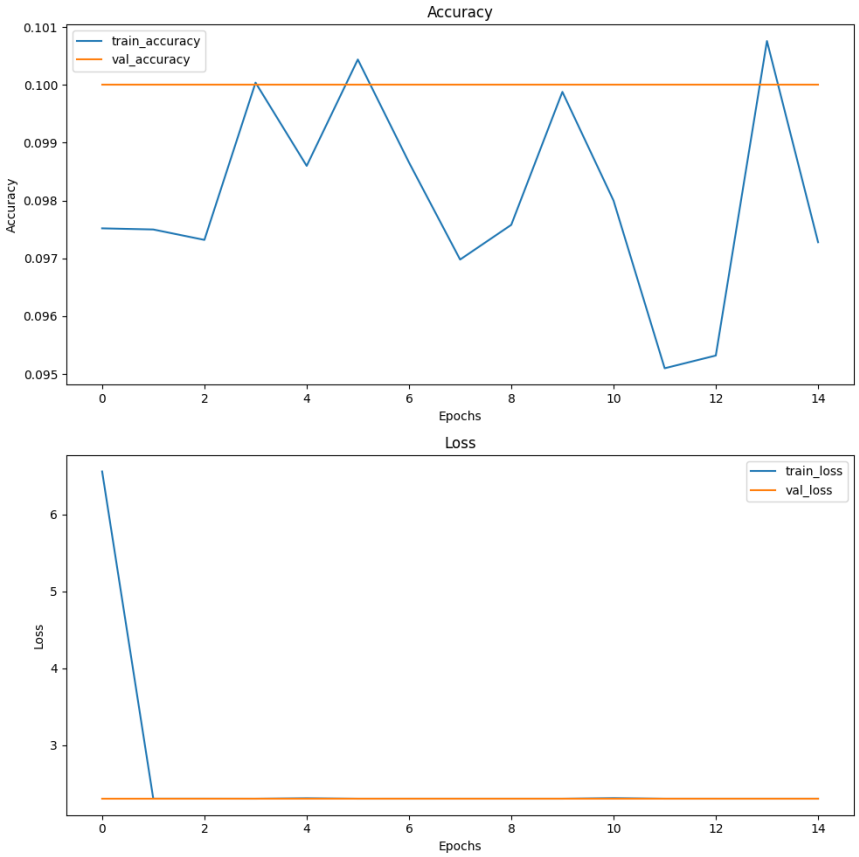


شکل ۵:

model 7,8 ۴.۱

در این مرحله با انتخاب کردن مدلی ساده تر و اولین بار با اضافه کردن کمی داده شکل ۶و بار بعد یعنی مدل ۸ مشاهده میکنیم که نتیجه اصلا خوب نشده

متاسفانه با رسیدن به محدودیت جی پی یو در گوگل کولب عکس مدل هشت و فایل csv چند مدل پاک شد



شکل ۶:

۲ more complex cnn

۱.۲ تعریف

این مدل کمی از مدل قبل پیچیده تر است و نتیجه بهتری هم دارد

توابعی که اضافه شدن

GlobalAveragePooling۲D

مانند ماکس پولینگ هست ولی برروی کل تصویر و میانگین حساب میکند نه این که از هر پنجره ماکسیمم مقدار

Dropout

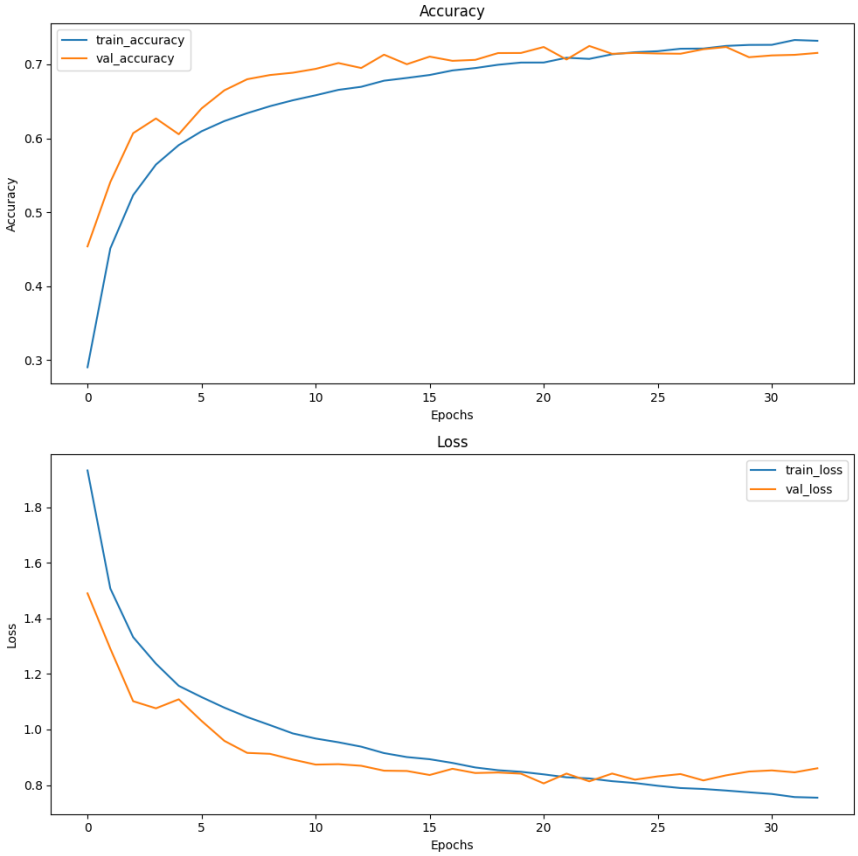
تابعی برای جلوگیری از اورفیتینگ هست و در واقع با احتمالی بعضی نورون ها رو خاموش می‌کنه

```python
#for model number 9 and 10
model = keras.Sequential(
    [
        keras.layers.Input(shape=(32, 32, 3)),
        keras.layers.Conv2D(32, kernel_size=(3, 3), activation=model_params['activation_functions'][0]),
        keras.layers.Conv2D(32, kernel_size=(3, 3), activation=model_params['activation_functions'][1]),
        keras.layers.MaxPooling2D(pool_size=(2, 2)),
        keras.layers.Conv2D(64, kernel_size=(3, 3), activation=model_params['activation_functions'][2]),
        keras.layers.Conv2D(64, kernel_size=(3, 3), activation=model_params['activation_functions'][3]),
        keras.layers.GlobalAveragePooling2D(),
        keras.layers.Dropout(0.5),
        keras.layers.Dense(number_of_output_class, activation=model_params['activation_functions'][4]),
    ]
)
```

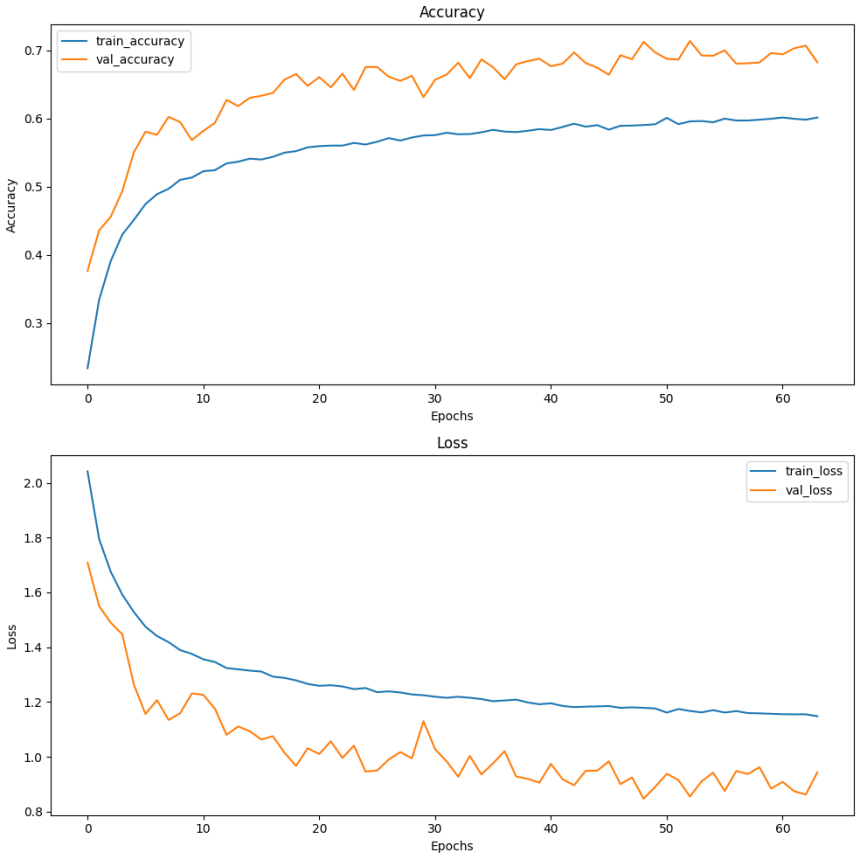
شکل ۷:

۲.۲ model 9,10

در مدل ۹ و ۱۰ مدلی پیچیده تر را روی کار میاوریم و اول بدون داده اضافی و سپس با داده اضافی نتایج را ثبت میکنیم



شکل ۸:



شکل ۹:
۱۰

۳ more complex cnn

۱.۳ تعریف

این مدل خیلی پیچیده تر هست و نتیجه بهتری هم دارد

از تابع جدیدی به نام BatchNormalization استفاده شده که برای همگرایی سریع تر استفاده میشد و به فرایند آموزش سرعت می‌بخشد

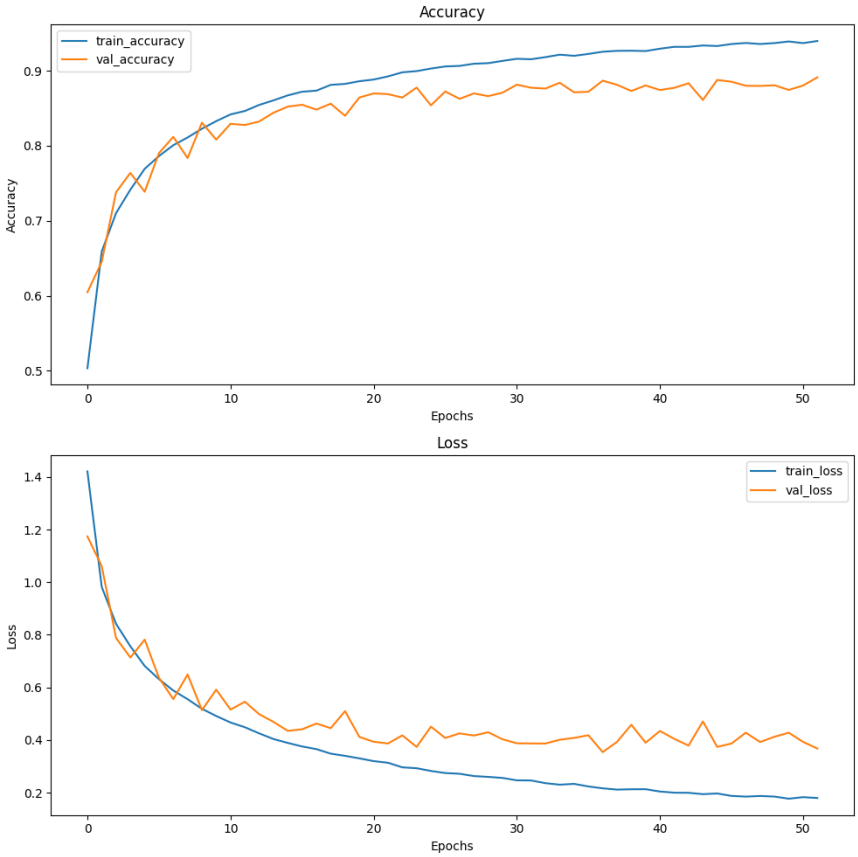
ولی برخلاف مدل های دیگر در صورت اضافه نکردن داده به آن نتیجه بدتری به ما میدهد

```python
# K = len(set(y_train))
K = number_of_output_class

# Build the model using the functional API
# input layer
i = Input(shape=x_train[0].shape)
x = Conv2D(32, (3, 3), activation=model_params['activation_functions'][0], padding='same')(i)
x = BatchNormalization()(x)
x = Conv2D(32, (3, 3), activation=model_params['activation_functions'][1], padding='same')(x)
x = BatchNormalization()(x)
x = MaxPooling2D((2, 2))(x)

x = Conv2D(64, (3, 3), activation=model_params['activation_functions'][2], padding='same')(x)
x = BatchNormalization()(x)
x = Conv2D(64, (3, 3), activation=model_params['activation_functions'][3], padding='same')(x)
x = BatchNormalization()(x)
x = MaxPooling2D((2, 2))(x)

x = Conv2D(128, (3, 3), activation=model_params['activation_functions'][4], padding='same')(x)
x = BatchNormalization()(x)
x = Conv2D(128, (3, 3), activation=model_params['activation_functions'][5], padding='same')(x)
x = BatchNormalization()(x)
x = MaxPooling2D((2, 2))(x)

x = Flatten()(x)
x = Dropout(0.2)(x)

# Hidden layer
x = Dense(1024, activation=model_params['activation_functions'][6])(x)
x = Dropout(0.2)(x)

# last hidden layer i.e.. output layer
x = Dense(K, activation=model_params['activation_functions'][7])(x)

model = Model(i, x)
```

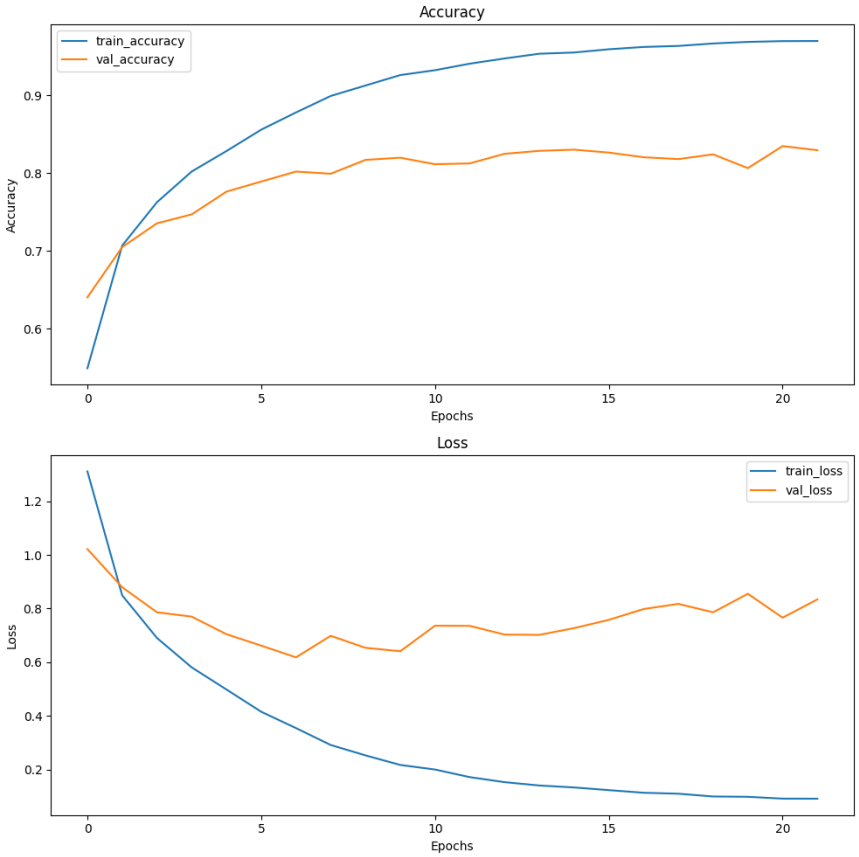
شکل ۱۰:

۲.۳ model 11,12

بازهم به سراغ مدل پیچید تری میرویم و نتایج زیر ثبت میشود


Accuracy
train_accuracy
val_accuracy
Accuracy
Epochs
Loss
train_loss
val_loss
Loss
Epochs

شکل ۱۱:
۱۱



شکل ۱۲:
۱۲

model 13,14 ۳.۳

همان مدل قبلی هست ولی با عوض کردن تابع فعال ساز

در مدل ۱۳ داده افزایی زیاد ودر مدل ۱۴ داده افزایی کمتر شده