دانشکده مهندسی کامپیوتر

پروژه پایانی

دسته‌بندی تصاویر به "پلاک"، "غیرپلاک" و "پلاک مخدوش"

نام درس
مبانی بینایی کامپیوتر

نام دانشجویان
زهرا انوریان
محمدمهدی عبداله‌پور

نام استاد درس
دکتر محمدرضا محمدی

زمستان ۱۳۹۹

۱ تعریف مسئله

توسعه یک الگوریتم بینایی کامپیوتر برای دسته‌بندی تصاویر ورودی به سه دسته "پلاک سالم"، "پلاک مخدوش " و "غیرپلاک" است که نمونه‌هایی از این تصاویر در شکل زیر نشان داده شده است. تصاویری که در دسته پلاک سالم قرار می‌گیرند شامل یک پلاک بدون پوشش هستند و تصاویری که شامل یک پلاک باشند که بخشی از آن پوشانده شده باشد در دسته پلاک مخدوش قرار می‌گیرند و تصاویری که شامل هیچ پلاکی نیستند در دسته غیرپلاک قرار می‌گیرند. توجه داشته باشید که پلاک (سالم یا مخدوش) در هر بخشی از تصویر ممکن است وجود داشته باشد و لزوما در مرکز تصویر وجود ندارد.

۲ مقدمه

ابتدا با تحلیل مجموعه‌داده متوجه آن شدیم که دسته‌ی پلاک دارای ۱۸۷۶ عکس و دسته‌ی غیرپلاک دارای ۵۶۸ عکس و دسته‌ی پلاک مخدوش دارای ۲۸۳ عکس می‌باشد. باتوجه به این مقادیر تصمیم گرفتیم، معماری‌هایی را برای بررسی انتخاب کنیم که دارای پارامتر کمتر و درصد دقت بالاتر هستند.

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88

باتوجه به جدول بالا و توضیحات داده شده، ما معماری‌های Xception، ResNet50، MobileNet را برای تست و تحلیل درنظر گرفتیم و در ادامه به هرکدام با جزئیات بیشتر می‌پردازیم.
ما برای ارزیابی مدلمان معیارهای f1-score، precision، recall و accuracy را در نظر گرفتیم.

$$Accuracy = \frac{TP + TN}{TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 - Score = 2 * \frac{Precision * recall}{precision + recall}$$

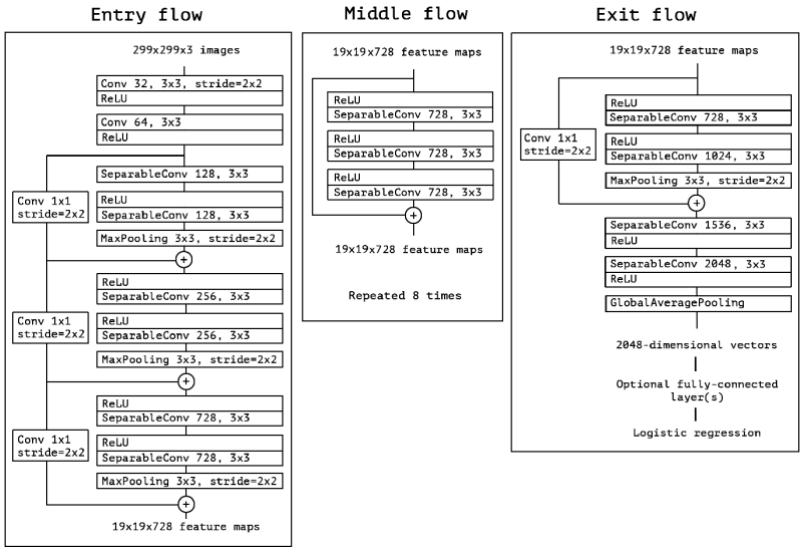
۳ تحلیل معماری‌های مختلف

این نکته نیز قابل ذکر است که ما حالت فریزکردن قسمتی از شبکه را نیز آزمایش کردیم اما چون نتیجه‌ی بهتری از آن نسبت به از ابتدا آموزش دیدن بدست نیاوردیم، معماری‌های مختلفی که مورد بررسی قرار دادیم به کلی آموزش دیده‌اند. حال در ادامه معماری‌ها را بررسی کرده و نتایج آن‌ها را مشاهده می‌کنیم.

تمامی مدل‌های بررسی شده‌ای که در ادامه می‌بینید را می‌توانید از اینجا دانلود کنید.

۳.۱ معماری Xception

این معماری دارای ساختاری به شکل زیر است و همانطور که مشاهده می‌کنیم در این معماری از Depth wise Separable Convolution استفاده شده است که باعث کاهش پارامتر و بهبود در عملکرد می‌شود.



ما به این صورت عمل کردیم که وزن‌های اولیه‌ی مدلمان را با وزن‌های آموزش دیده شده روی مجموعه‌داده imagenet مقداردهی کردیم و با استفاده از بهینه‌ساز[1] Adam و SGD و همچنین اضافه کردن چند لایه‌ی کاملا متصل و dropout که در شکل زیر می‌توان ساختار مدل را دید، آن را آموزش دادیم. لازم به ذکر است که برای کاهش overfit مدل، از داده‌افزایی استفاده کردیم و درنهایت توانستیم به نتایجی که در ادامه آمده است، برسیم.

```
def build_model():
  baseModel = Xception(
        include_top=False,
        weights="imagenet",
        input_tensor=Input(shape=(IMG_HEIGHT, IMG_WIDTH, 3)),
        pooling='avg',
    )

  headModel = baseModel.output
  headModel = Dropout(0.2)(headModel)
  headModel = Dense(1024, activation="relu")(headModel)
  headModel = Dropout(0.2)(headModel)
  headModel = Dense(1024, activation="relu")(headModel)
  headModel = Dropout(0.2)(headModel)
  headModel = Dense(512, activation="relu")(headModel)
  headModel = Dropout(0.2)(headModel)
  headModel = Dense(512, activation="relu")(headModel)
  headModel = Dropout(0.2)(headModel)
  headModel = Dense(n_classes, activation="softmax")(headModel)

  model = Model(inputs=baseModel.input, outputs=headModel)

  return model
```

نتایج بدست آمده از بهینه‌ساز Adam:

```
best_model.evaluate(validation_generator, batch_size=batch_size)
```

```
17/17 [==============================] - 14s 663ms/step - loss: 0.2108 - accuracy: 0.9773 - f1_m:
0.9773 - precision_m: 0.9773 - recall_m: 0.9773
```

[1] Optimizer

```
Found 544 images belonging to 3 classes.
Confusion Matrix
[[372   3   0]
 [  9  45   2]
 [  0   2 111]]
Classification Report
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       375
           1       0.90      0.80      0.85        56
           2       0.98      0.98      0.98       113

    accuracy                           0.97       544
   macro avg       0.95      0.93      0.94       544
weighted avg       0.97      0.97      0.97       544
```

نتایج بدست آمده از بهینه‌ساز SGD:

```
best_model.evaluate(validation_generator, batch_size=batch_size)
```

```
17/17 [==============================] - 14s 656ms/step - loss: 0.1275 - accuracy: 0.9771 - f1_m:
0.9760 - precision_m: 0.9770 - recall_m: 0.9750
```

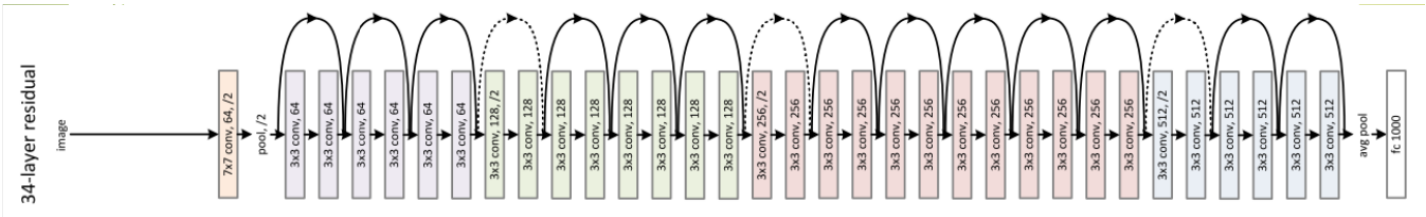
```
Found 544 images belonging to 3 classes.
Confusion Matrix
[[375   0   0]
 [ 10  44   2]
 [  2   2 109]]
Classification Report
              precision    recall  f1-score   support

           0       0.97      1.00      0.98       375
           1       0.96      0.79      0.86        56
           2       0.98      0.96      0.97       113

    accuracy                           0.97       544
   macro avg       0.97      0.92      0.94       544
weighted avg       0.97      0.97      0.97       544
```

۳.۲ معماری ResNet50

این معماری دارای ساختاری به شکل زیر است.



نتایج بدست آمده از بهینه‌ساز SGD:

```
best_model.evaluate(validation_generator, batch_size=batch_size)
```

```
9/9 [==============================] - 50s 4s/step - loss: 0.0991 - accuracy: 0.9815 - f1_m: 0.981
8 - precision_m: 0.9818 - recall_m: 0.9818
```

```
Found 544 images belonging to 3 classes.
Confusion Matrix
[[370   4   1]
 [  7  47   2]
 [  0   0 113]]
Classification Report
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       375
           1       0.92      0.84      0.88        56
           2       0.97      1.00      0.99       113

    accuracy                           0.97       544
   macro avg       0.96      0.94      0.95       544
weighted avg       0.97      0.97      0.97       544
```

۳.۳ معماری MobileNet

این معماری دارای ساختاری به شکل زیر است.

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

نتایج بدست آمده از بهینه‌ساز Adam:

```
1 best_model.evaluate(validation_generator, batch_size=BATCH_SIZE)
```

```
10/10 [==============================] - 6s 622ms/step - loss: 0.1347 - accuracy: 0.9724 - f1_m: 0.9517 - precision_m: 0.9517 - recall_m: 0.9517
[0.13467468321323395,
 0.9724264740943909,
 0.951666533946991,
 0.9516666531562805,
 0.9516666531562805]
```

```
Found 544 images belonging to 3 classes.
Confusion Matrix
[[371   3   1]
 [  6  49   1]
 [  2   1 110]]
Classification Report
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       375
           1       0.92      0.88      0.90        56
           2       0.98      0.97      0.98       113

    accuracy                           0.97       544
   macro avg       0.96      0.95      0.95       544
weighted avg       0.97      0.97      0.97       544
```

۴ تلاش برای بهبود عملکرد مدل‌ها

ما برای بهبود نتایج بالا و همچنین آزمایش راه‌های مختلف و شهود بیشتر، چندیدن ایده در نظر داشتیم که هر کدام را اجرا کردیم و به نتایجی رسیدیم که در ادامه به هرکدام با جزئیات بیشتر می‌پردازیم.

۴.۱ افزایش داده‌های دسته‌ی پلاک مخدوش

باتوجه به آن که داده‌های دسته‌ی پلاک مخدوش دارای تعداد کمی داده است و میزان دقت این کلاس به نسبت دیگر کلاس‌ها پایین‌تر است، تصمیم گرفتیم که داده‌های این دسته را افزایش دهیم بنابراین به طور دستی تعدادی از داده‌های دسته‌ی پلاک را مخدوش کردیم و حدود ۱۰۰ عکس دیگر به دسته پلاک های مخدوش اضافه کردیم تا مدل عکس‌های بیشتری از این دسته ببیند و یاد بگیرد.

مجموعه‌داده جدید را می‌توانید از اینجا دانلود کنید.

نتایج بدست آمده از بهینه‌ساز Adam با معماری Xception:

```
1 best_model.evaluate(validation_generator, batch_size=BATCH_SIZE)
```

```
10/10 [==============================] - 9s 707ms/step - loss: 0.1623 - accuracy: 0.9623 - f1_m: 0.9630 - precision_m: 0.9637 - recall_m: 0.9623
[0.1581851989030838,
 0.9593639373779297,
 0.9602853655815125,
 0.9610973596572876,
 0.9594871401786804]
```

```
Found 566 images belonging to 3 classes.
Confusion Matrix
[[361  11   3]
 [ 14  63   1]
 [  0   0 113]]
Classification Report
              precision    recall  f1-score   support

           0       0.96      0.96      0.96       375
           1       0.85      0.81      0.83        78
           2       0.97      1.00      0.98       113

    accuracy                           0.95       566
   macro avg       0.93      0.92      0.92       566
weighted avg       0.95      0.95      0.95       566
```

نتایج بدست آمده از بهینه‌ساز SGD با معماری ResNet50:

```
1 best_model.evaluate(validation_generator, batch_size=BATCH_SIZE)
```

```
10/10 [==============================] - 10s 777ms/step - loss: 0.3194 - accuracy: 0.9421 - f1_m: 0.9420 - precision_m: 0.9420 - recall_m: 0.9420
[0.3799374997615814,
 0.9328621625900269,
 0.9323077201843262,
 0.9323077201843262,
 0.9323077201843262]
```

```
Found 566 images belonging to 3 classes.
Confusion Matrix
[[371   3   1]
 [ 29  48   1]
 [  0   0 113]]
Classification Report
              precision    recall  f1-score   support

           0       0.93      0.99      0.96       375
           1       0.94      0.62      0.74        78
           2       0.98      1.00      0.99       113

    accuracy                           0.94       566
   macro avg       0.95      0.87      0.90       566
weighted avg       0.94      0.94      0.93       566
```

نتایج بدست آمده از بهینه‌ساز Adam با معماری MobileNet:

```
1 best_model.evaluate(validation_generator, batch_size=BATCH_SIZE)
```

```
10/10 [==============================] - 7s 618ms/step - loss: 0.2403 - accuracy: 0.9632 - f1_m: 0.9637 - precision_m: 0.9642 - recall_m: 0.9633
[0.23752467334270477,
 0.9650735259056091,
 0.9681919813156128,
 0.9697701334953308,
 0.9666666984558105]
```

```
Found 544 images belonging to 3 classes.
Confusion Matrix
[[370   4   1]
 [ 11  42   3]
 [  0   0 113]]
Classification Report
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       375
           1       0.91      0.75      0.82        56
           2       0.97      1.00      0.98       113

    accuracy                           0.97       544
   macro avg       0.95      0.91      0.93       544
weighted avg       0.96      0.97      0.96       544
```

باتوجه به نتایج بدست آمده متوجه‌ی پسرفت دقت مدل در دسته‌بندی این دسته شدیم و دلیل این امر نیز ممکن است از ناهمخوانی پلاک‌های مخدوش اضافه شده با پلاک‌های موجود در مجموعه داده باشد. با بررسی دوباره‌ی این داده‌ها و مقایسه با داده‌های اصلی این دسته متوجه سخت‌تر بودن داده‌های اضافه شده می‌شویم که خود می‌تواند دلیلی بر کاهش دقت مدل باشد. همچین تعداد داده‌های این دسته همچنان بسیار کمتر از داده‌های دسته اول است. overfit شدن مدل روی داده‌های اولیه نیز می‌تواند از دلایل این امر باشد.

۴.۲ تفکیک مجموعه‌داده به داده‌ی Train, Validation, Test

تا به اینجا داده‌ها را طبق کد زیر که بر اساس داکیومنتیشن keras نوشته شده است به دو دسته‌ی Train و Validation تقسیم می‌کردیم اما طبق این issue متوجه شدیم که keras به اشتباه داده‌های Validation را نیز داده‌افزایی می‌کند. همچنین نبود داده‌ی تست منجر به کاهش ارزش ارزیابی‌هایی که بعد از یادگیری مدل انجام می‌دادیم، می‌شد.

```
train_datagen_aug = ImageDataGenerator(preprocessing_function=preprocess_input,
                                       shear_range=0.25,
                                       width_shift_range=0.25,
                                       height_shift_range=0.25,
                                       zoom_range=0.2,
                                       rotation_range=40,
                                       validation_split=0.2)

train_generator_aug  = train_datagen_aug.flow_from_directory(DATA_TRAIN_PATH,
                                                             target_size=(IMG_WIDTH, IMG_HEIGHT),
                                                             batch_size=BATCH_SIZE,
                                                             class_mode='categorical',
                                                             subset='training')

validation_generator  = train_datagen_aug.flow_from_directory(DATA_TRAIN_PATH,
                                                              target_size=(IMG_WIDTH, IMG_HEIGHT),
                                                              batch_size=BATCH_SIZE,
                                                              class_mode='categorical',
                                                              subset='validation')
```

در نتیجه همانطور که در شکل زیر مشاهده می‌کنید، تصمیم گرفتیم به طور کلی ساختار داده‌ها را تغییر دهیم به این صورت که داده‌های Train, Validation, Test از ابتدا به نسبت ۸۰-۱۰-۱۰ جدا کردیم و برای هر کدام از قسمت‌ها یک generator با مسیرهای مربوطه در نظر گرفتیم و بر اساس این ساختار فقط بر روی داده‌های آموزشی داده‌افزایی اعمال می‌شود.

مجموعه داده‌ی جدید را می‌توانید از اینجا دانلود کنید.


```
train_datagen_aug = ImageDataGenerator(preprocessing_function=preprocess_input,
                                       shear_range=0.25,
                                       width_shift_range=0.25,
                                       height_shift_range=0.25,
                                       zoom_range=0.2,
                                       rotation_range=40)

validation_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)

test_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)

train_generator_aug  = train_datagen_aug.flow_from_directory(TRAIN_PATH,
                                                             target_size=(IMG_WIDTH, IMG_HEIGHT),
                                                             batch_size=BATCH_SIZE,
                                                             class_mode='categorical')
validation_generator  = validation_datagen.flow_from_directory(VALIDATION_PATH,
                                                               target_size=(IMG_WIDTH, IMG_HEIGHT),
                                                               batch_size=BATCH_SIZE,
                                                               class_mode='categorical')
test_generator = test_datagen.flow_from_directory(TEST_PATH,
                                                  target_size=(IMG_WIDTH, IMG_HEIGHT),
                                                  batch_size=BATCH_SIZE,
                                                  shuffle=False,
                                                  class_mode='categorical')
```

نتایج بدست آمده از بهینه‌ساز Adam با معماری Xception:

```
1 best_model.evaluate(test_generator, batch_size=BATCH_SIZE)
```

```
5/5 [==============================] - 3s 564ms/step - loss: 0.0877 - accuracy: 0.9819 - f1_m: 0.9833 - precision_m: 0.9833 - recall_m: 0.9833
[0.08772743493318558,
 0.9818840622901917,
 0.9833332896232605,
 0.98333340883255,
 0.98333340883255]
```

```
Confusion Matrix
[[189   0   0]
 [  4  24   1]
 [  0   0  58]]
Classification Report
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       189
           1       1.00      0.83      0.91        29
           2       0.98      1.00      0.99        58

    accuracy                           0.98       276
   macro avg       0.99      0.94      0.96       276
weighted avg       0.98      0.98      0.98       276
```

نتایج بدست آمده از بهینه‌ساز SGD با معماری ResNet50:

```
1 best_model.evaluate(test_generator, batch_size=BATCH_SIZE)
```

```
5/5 [==============================] - 1s 210ms/step - loss: 0.1203 - accuracy: 0.9746 - f1_m: 0.9744 - precision_m: 0.9744 - recall_m: 0.9744
[0.12034094333648682,
 0.9746376872062683,
 0.9744445085525513,
 0.9744445085525513,
 0.9744445085525513]
```

```
Confusion Matrix
[[187   2   0]
 [  3  25   1]
 [  1   0  57]]
Classification Report
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       189
           1       0.93      0.86      0.89        29
           2       0.98      0.98      0.98        58

    accuracy                           0.97       276
   macro avg       0.96      0.94      0.95       276
weighted avg       0.97      0.97      0.97       276
```

نتایج بدست آمده از بهینه‌ساز Adam با معماری MobileNet:

```
1 best_model.evaluate(test_generator, batch_size=BATCH_SIZE)
```

```
5/5 [==============================] - 167s 41s/step - loss: 0.1581 - accuracy: 0.9694 - f1_m: 0.9704 - precision_m: 0.9704 - recall_m: 0.9704
[0.2589297294616699,
 0.9601449370384216,
 0.9633333086967468,
 0.9633333086967468,
 0.9633333086967468]
```

```
Confusion Matrix
[[186   3   0]
 [  4  21   4]
 [  0   0  58]]
Classification Report
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       189
           1       0.88      0.72      0.79        29
           2       0.94      1.00      0.97        58

    accuracy                           0.96       276
   macro avg       0.93      0.90      0.91       276
weighted avg       0.96      0.96      0.96       276
```

۴.۳ تغییر میزان روشنایی عکس‌های مجموعه داده

ایده‌ی دیگری که داشتیم، تغییر میزان روشنایی عکس‌ها برای افزایش داده‌های موجود در مجموعه داده و بالابردن دقت و همچنین کاهش overfit مدل می‌باشد. بنابراین کدی که در زیر مشاهده می‌کنید این عملیات تغییر میزان روشنایی را انجام می‌دهد. به این صورت که ابتدا عکس‌ها را می‌خواند و آن‌ها را به سطح رنگی HSV می‌بریم و سپس میانگین مولفه‌ی V تصویر را بدست می‌آوریم و اگر میانگین از ۹۰ بیشتر بود V آن را کاهش می‌دهیم یعنی تصویر را تیره می‌کنیم و اگر میانگین از ۱۶۰ کمتر بود V آن را افزایش می‌دهیم یعنی تصویر را روشن می‌کنیم. هدف از انجام این کار این است که تصاویری که به شدت تیره/روشن هستند را تیره‌تر/روشن‌تر نکنیم.

مجموعه‌داده‌ی جدید را می‌توانید از اینجا دانلود کنید.

```python
for c in range(3):
    for img_file in all_images[c]:
        img = cv2.imread(join(f'./correct_data/train/{c}', img_file))
        img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
        avg_v = np.sum(img[:,:,2]) / (img.shape[0] * img.shape[1])
        if (avg_v > 90):
            img_aug = np.copy(img)
            img_aug[img_aug[:,:,2] <= 50, 2] = 0
            img_aug[img_aug[:,:,2] > 50, 2] = img_aug[img_aug[:,:,2] > 50, 2] - 50
            img_aug = cv2.cvtColor(img_aug, cv2.COLOR_HSV2BGR)
            aug_path = join(f'./correct_data_aug/train/{c}', f'darker_{img_file}')
            cv2.imwrite(aug_path, img_aug)
        if (avg_v < 160):
            img_aug = np.copy(img)
            img_aug[img_aug[:,:,2] >= 205, 2] = 255
            img_aug[img_aug[:,:,2] < 205, 2] = img_aug[img_aug[:,:,2] < 205, 2] + 50
            img_aug = cv2.cvtColor(img_aug, cv2.COLOR_HSV2BGR)
            aug_path = join(f'./correct_data_aug/train/{c}', f'lighter_{img_file}')
            cv2.imwrite(aug_path, img_aug)
```

نتایج بدست آمده از بهینه‌ساز Adam با معماری Xception:

```
1 best_model.evaluate(test_generator, batch_size=batch_size)
```

```
5/5 [==============================] - 51s 10s/step - loss: 0.2380 - accuracy: 0.9638 - f1_m: 0.9688 - precision_m: 0.9688 - recall_m: 0.9688
```

```
Confusion Matrix
[[186   2   1]
 [  3  22   4]
 [  0   0  58]]
Classification Report
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       189
           1       0.92      0.76      0.83        29
           2       0.92      1.00      0.96        58

    accuracy                           0.96       276
   macro avg       0.94      0.91      0.92       276
weighted avg       0.96      0.96      0.96       276
```

نتایج بدست آمده از بهینه‌ساز SGD با معماری ResNet50:

```
1 best_model.evaluate(test_generator, batch_size=BATCH_SIZE)
```

```
5/5 [==============================] - 3s 318ms/step - loss: 0.1062 - accuracy: 0.9807 - f1_m: 0.9813 - precision_m: 0.9813 - recall_m: 0.9813
[0.13046230375766754,
 0.97826087474823,
 0.9799998998641968,
 0.9800000190734863,
 0.9800000190734863]
```

```
Confusion Matrix
[[185   4   0]
 [  2  27   0]
 [  0   0  58]]
Classification Report
              precision    recall  f1-score   support

           0       0.99      0.98      0.98       189
           1       0.87      0.93      0.90        29
           2       1.00      1.00      1.00        58

    accuracy                           0.98       276
   macro avg       0.95      0.97      0.96       276
weighted avg       0.98      0.98      0.98       276
```

نتایج بدست آمده از بهینه‌ساز Adam با معماری MobileNet:

```
1 best_model.evaluate(test_generator, batch_size=BATCH_SIZE)
```

```
5/5 [==============================] - 56s 14s/step - loss: 0.1234 - accuracy: 0.9635 - f1_m: 0.9640 - precision_m: 0.9640 - recall_m: 0.9640
[0.1530923992395401,
 0.9528985619544983,
 0.9544442892074585,
 0.9544445276260376,
 0.9544445276260376]
```

```
Confusion Matrix
[[183   6   0]
 [  3  25   1]
 [  0   3  55]]
Classification Report
              precision    recall  f1-score   support

           0       0.98      0.97      0.98       189
           1       0.74      0.86      0.79        29
           2       0.98      0.95      0.96        58

    accuracy                           0.95       276
   macro avg       0.90      0.93      0.91       276
weighted avg       0.96      0.95      0.95       276
```

۴.۴ اضافه کردن داده‌ی مخدوش جدید به نسبت ۸۰-۱۰-۱۰

در این تلاش ما حدود ۱۰۰ تصویر مخدوش جدید را که خودمان ساختیم را به نسبت متناسب بین داده‌های Train, Validation و Test اضافه کردیم تا اندکی از میزان unbalanced بودن مجموعه‌داده و بایاس شدن مدل بکاهیم.

مجموعه داده‌ی جدید را می‌توان از اینجا دانلود کرد.

نتایج بدست آمده از بهینه‌ساز Adam با معماری Xception:

```
1 best_model.evaluate(test_generator, batch_size=BATCH_SIZE)
```

```
5/5 [==============================] - 3s 424ms/step - loss: 0.1239 - accuracy: 0.9662 - f1_m: 0.9628 - precision_m: 0.9677 - recall_m: 0.9580
[0.16122640669345856,
 0.9651567935943604,
 0.9645064473152161,
 0.9691408276557922,
 0.9600000381469727]
```

```
Confusion Matrix
[[186   3   0]
 [  3  34   3]
 [  1   0  57]]
Classification Report
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       189
           1       0.92      0.85      0.88        40
           2       0.95      0.98      0.97        58

    accuracy                           0.97       287
   macro avg       0.95      0.94      0.94       287
weighted avg       0.96      0.97      0.96       287
```

۵ مشاهده‌ی برخی از داده‌های به اشتباه پیش‌بینی شده

با استفاده از کد زیر عکس‌هایی که توسط مدل به اشتباه تشخیص داده شده است را نمایش می‌دهیم و در ادامه نمونه‌هایی از این تصاویر را آورده‌ایم.

```
test_generator = test_datagen.flow_from_directory(
    test_data_dir,
    target_size=(img_height, img_width),
    batch_size=500,
    class_mode='categorical',
    shuffle=False
)

labels = np.array(test_generator.classes)
print(labels)
print(len(labels))

Y_pred = best_model.predict(test_generator)
y_pred = np.argmax(Y_pred, axis=1)
idx = np.where(y_pred != labels)[0]
np.random.shuffle(idx)
print(idx)

for i in range(10):
    img = test_generator[0][0][idx[i]]
    print(f'correct: {labels[idx[i]]}, predicted: {y_pred[idx[i]]}')
    plt.figure()
    plt.imshow(img)
    plt.xticks([])
    plt.yticks([])
    plt.show()
```

correct: 1, predicted: 0

علت احتمالی: مشکل بودن داده تست به دلیل همرنگی و میزان کم مخدوش کردن

#309 correct: 1, predicted: 2

علت احتمالی: تغییر شدید ابعاد تصویر توسط preprocessor و رنگ پلاک

#318 correct: 1, predicted: 0

علت احتمالی: سخت بودن داده برای تشخیص درست (استفاده از رنگ سفید برای مخدوش‌سازی پلاک و شباهت آن قسمت با عدد ۱)

correct: 1, predicted: 0

علت احتمالی: رنگ پلاک و وجود حرف (نوشتار) در المان مخدوش کننده

#10 correct: 0, predicted: 1

علت احتمالی: نورافتادن در پلاک باعث خطای مدل در تشخیص

۶ جمع‌بندی

با توجه به مشاهدات ما، اختلاف ۲ تا ۳ درصدی که موجود بین دقت‌ها به دلیل کیفیت و کمیت کم دیتاست نشانگر و معیار درستی برای تصمیم‌گیری و قضاوت در مورد مدل‌ها نمی‌باشد و در هر بار آموزش و اجرا و عوض کردن داده‌ها ممکن است اندکی تغییر داشته باشند. از نظر ما مدلی برتری مشخص و محکمی نسبت به مدل دیگر نداشته و شاید بتوان گفت مدل Xception اندکی عملکرد بهتری از خود نشان داده‌است. مبنای اصلی ما مجموعه‌داده‌ای است که در بخش ۴.۲ به آن رسیدیم اما تمام نتایج و عملیات انجام شده را به طور خلاصه در جدول زیر آورده‌ایم.

#	model name	optimizer	data dir format	lighting aug	add new data	accuracy	f1-score macro avg
1	Xception	Adam	original	0	0	97.73	94
2	Xception	SGD	original	0	0	97.71	94
3	ResNet50	SGD	original	0	0	98.15	95
4	MobileNet	Adam	original	0	0	97.24	95
5	Xception	Adam	corrected	0	0	98.19	96
6	ResNet50	SGD	corrected	0	0	97.46	95
7	MobileNet	Adam	corrected	0	0	96.94	91
8	Xception	Adam	original	0	1 (tr/v)	96.23	92
9	ResNet50	SGD	original	0	1 (tr/v)	94.21	90
10	MobileNet	Adam	original	0	1 (tr/v)	96.32	93
11	Xception	Adam	original	1 (tr/v)	0	99.64	99
12	Xception	Adam	corrected	1 (tr)	0	96.38	92
13	ResNet50	SGD	corrected	1 (tr)	0	98.07	96
14	MobileNet	Adam	corrected	1 (tr)	0	96.40	91
15	Xception	Adam	corrected	1 (tr) (keras)	1 (tr/v/te)	96.62	94

۷ کارهای آینده

- سیاه سفید کردن تمام تصاویر و استفاده از الگوریتم otsu:

با بررسی داده‌ها این ایده به ذهن می‌رسد که شاید از بین بردن اختلاف رنگی بین تصاویر باعث بهبود عملکرد مدل شود زیرا در اکثر تصاویر پلاک‌ها به رنگ سفید بوده و قسمت آبی آن جز بسیار کمی از تصویر را تشکیل می‌دهد. همچنین از بین بردن رنگ تصاویر و نرمالیزه کردن روشنایی با استفاده از otsu این امکان را می‌دهد که مدل هیچ وقت روی طیف رنگی و شدت نور overfit انجام ندهد. به هرحال میدانیم که توانایی تشخیص این ۳ کلاس هنگامی که تصاویر سیاه و سفید شده باشد برای انسان قابل انجام است و به این صورت نیست که اطلاعات کلیدی از دست برود به طوری که نتوان در تصویری پلاک مخدوش را به دلیل سیاه و سفید شدن تشخیص نداد. همانطور که در تصاویری که در شب گرفته شده است این موضوع وجود دارد.

- مربعی کردن تصاویر بدون از بین بردن نسبت ابعاد:

با بررسی تصاویر اشتباه تشخیص داده شده به این نتیجه می توان رسید که deformation حاصل از مربعی کردن تصاویر در بعضی از تصاویر که نسبت ابعاد با مربعی بودن فاصله زیادی دارد باعث عملکرد بد مدل می‌شود. برای حل این موضوع می توان تصاویر را ابتدا با padding به یک تصویر مربعی تبدیل کرد و سپس resize کرد. این موضوع در کراس به دلیل اینکه resize کردن قبل از preprocess functionها انجام می‌شود به سختی قابل انجام است. این کار را با استفاده از opencv می‌توان انجام داد.

- اضافه کردن انواع نویز به تصاویر از جمله نویز نمک و فلفل یا نویز گوسی:

اضافه کردن این نوع نویزها به تصاویر تا حدی که توانایی تشخیص کلاس‌ها از یکدیگر از بین نرود می‌تواند به کم کردن overfitting مدل کمک کند.

- بهبود دیتاست:

یکی از مشکلات اصلی دیتاست در دسترس balance نبودن تعداد داده‌های کلاس‌هاست. به خصوص که داده‌های کلاس پلاک مخدوش که بسیار اهمیت دارد و سخت‌ترین کلاس برای یادگیری مدل است تعداد بسیار بسیار کمی داده دارد. برای این کار می‌توان ابتدا تعدادی تصویر را از کلاس پلاک سالم گرفت و محل پلاک را تگ زد و سپس با یک کد ساده روی هر عکس در محدوده‌ی پلاک تصویری رندوم را اضافه کرد به صورتی که پلاک مخدوش شود. قسمت تگ زدن این کار زمان‌گیر است اما نتیجه‌ی حاصل از این کار می‌تواند به شدت به افزایش دقت مدل کمک کند.

این نکته قابل ذکر است که یادگیری کلاس غیر پلاک برای مدل نسبتا آسان است و در نتیجه اضافه کردن تصاویر جدید به این کلاس می تواند از اولویت پایینی برخوردار باشد.

۸ منابع استفاده شده در کد

https://datascience.stackexchange.com/questions/45165/how-to-get-accuracy-f1-precision-and-recall-for-a-keras-model