JUNE 2022 | تیر ۱۴۰۱

COMPUTER VISION

Project Report

Final Project | Phases 1, 2, 3

PRESENTED TO

Dr. Esmaeil Najafi

Javad Khoramdel

Yasamin Borhani

PRESENTED BY

Amin Mozayan

M.Amin HosseinNiya

Ali Rashedi

Table of Contents

چکیده

در درس بینایی کامپیوتر در طول ترم با روش‌های کلاسیک پردازش تصویر، کلیات ماشین لرنینگ، شبکه‌های عصبی و image augmentation آشنا شدیم و در پروژه نهایی تلاش کردیم تا در سه فاز و با استفاده از فنون ذکر شده آموخته‌های خویش را به کار بگیریم. در ادامه و در این گزارش مفصلا به توضیح هر فاز و روش انجام مراحل اجرایی آن پرداخته شده است و همچنین نتایج حاصل از انجام آنها نیز به طور کامل آورده شده است.

PHASE1

1 فاز اول

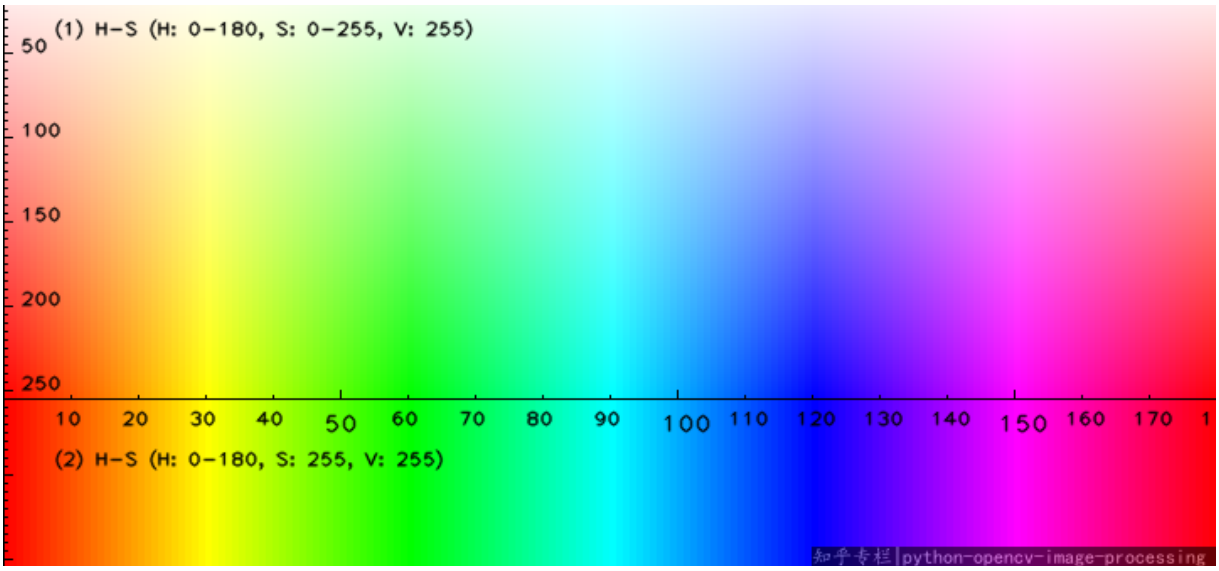
- **مقدمه**

در فاز اول پروژه قرار بود با استفاده از مطالبی که تا زمان ارائه‌ی این فاز آموزش داده شده بودند (کتابخانه‌ی open cv و روش‌های کلاسیک پردازش تصویر) رنگ چراغ راهنمایی را در ۶۰ تصویر مختلف تشخیص دهیم. مدل رنگ HSV این امکان را برای ما فراهم می‌کرد.

- **شرح کار**

در ابتدا دیتاستی شامل ۶۰ تصویر مختلف از چراغ‌های راهنمایی (۲۰ سبز، ۲۰ قرمز و ۲۰ زرد) تهیه کردیم (شکل ۱). این تصاویر با جست و جو در جست‌وجوگر گوگل و سایت پینترست به دست آمد (عکس‌ها در پوشه‌های red_light، green_light و yellow_light قرار داده شده‌اند). برای تشخیص رنگ چراغ در هر تصویر ابتدا آن را از مقیاس RGB به مقیاس HSV بردیم. در مقیاس HSV برخلاف RGB ، در بازه‌های پیوسته‌ی آرگومان‌ها (H , S, V) رنگ‌ها به طور پیوسته تغییر می‌کنند (شکل۲).

شکل ۱- نمونه عکس موجود در دیتاست تهیه شده



شکل ۲- تغییر طیف رنگی در مدل HSV با تغییر آرگومان‌ها

لذا انتظار داریم که برای مثال در یک چراغ با رنگ سبز، با توجه به شکل ۱، مقدار تعداد زیادی از پیکسل‌ها در بازه (۴۵, ۲۵, ۲۵) تا (۹۵, ۲۵۵,۲۵۵) باشد. این بازه برای رنـــــگ زرد برابر است با (۱۷, ۹۳, ۰) تا (۴۵, ۲۵۵, ۲۵۵) و برای رنـــگ قرمـــــز برابر با مجمـــوع دو بازه‌ی (۰, ۱۲۰, ۷۰) تا (۱۰, ۲۵۵, ۲۵۵) و (۱۷۰, ۱۲۰, ۷۰) تا (۱۸۰, ۲۵۵, ۲۵۵) است.

از هر عکس، سه ماسک تهیه کردیم که در اولی مقدار پیکسل‌های سبز، در دومی مقدار پیکسل‌های زرد و در سومی مقدار پیکسل‌های قرمز برابر یک بود. مقدار سایر پیکسل‌ها در هر سه تصویر صفر بود. بدین ترتیب ماسکی که بیشترین تعداد پیکسل غیر صفر را داشت نمایانگر رنگ چراغ در آن تصویر بود. این کد در فایل Project۱.py موجود است. همچنین کد Q۱_۲.py الگوریتم مشابهی را روی تمامی تصاویر پیاده می‌کند و دقت عملکرد آن را می‌سنجد.

- **نتایج**

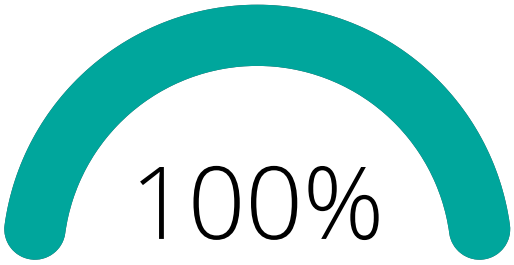
نتیجه‌ی بدست آمده برای تشخیص چراغ‌ها بدین شرح می‌باشد:

```
========= RESTART: C:\Users\ASUS\Desktop\project\Q1\green_light\Q1_2.py ========
number of true positives of green lights:  20 out of 20
>>>
========== RESTART: C:\Users\ASUS\Desktop\project\Q1\red_light\Q1_2.py =========
number of true positives of red lights:  20 out of 20
>>>
======== RESTART: C:\Users\ASUS\Desktop\project\Q1\yellow_light\Q1_2.py ========
number of true positives of red lights:  17 out of 20
```

شکل ۳- نتایج بدست آمده از دیتاست

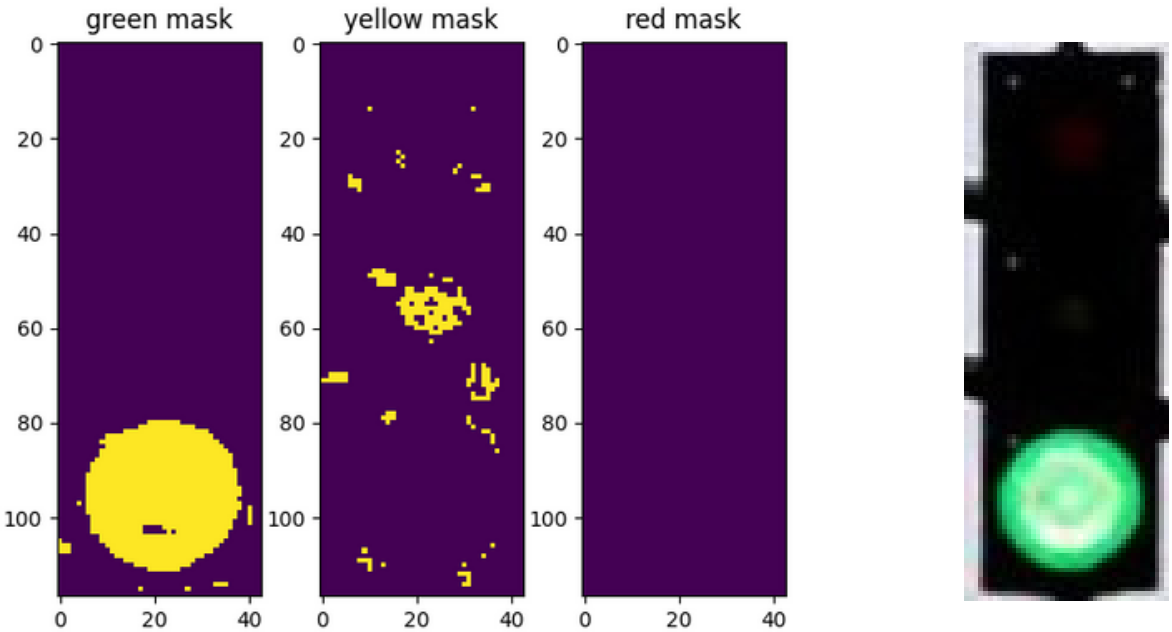
نتیجه بدست آمده برای چراغ سبز :

طبق پردازش انجام شده هر ۲۰ عکس موجود در دیتاست به درستی تشخیص داده شد و دقت تشخیص ۱۰۰ درصد شد.



نمودار ۱ - دقت بدست آمده در چراغ سبز

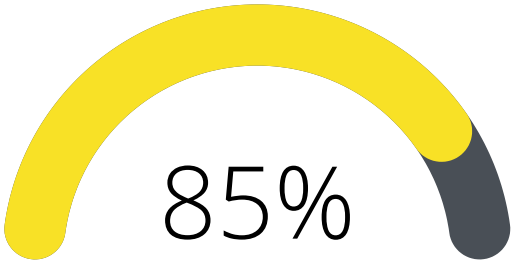
نمونه تشخیص :



شکل ۴- نمونه تشخیص چراغ سبز

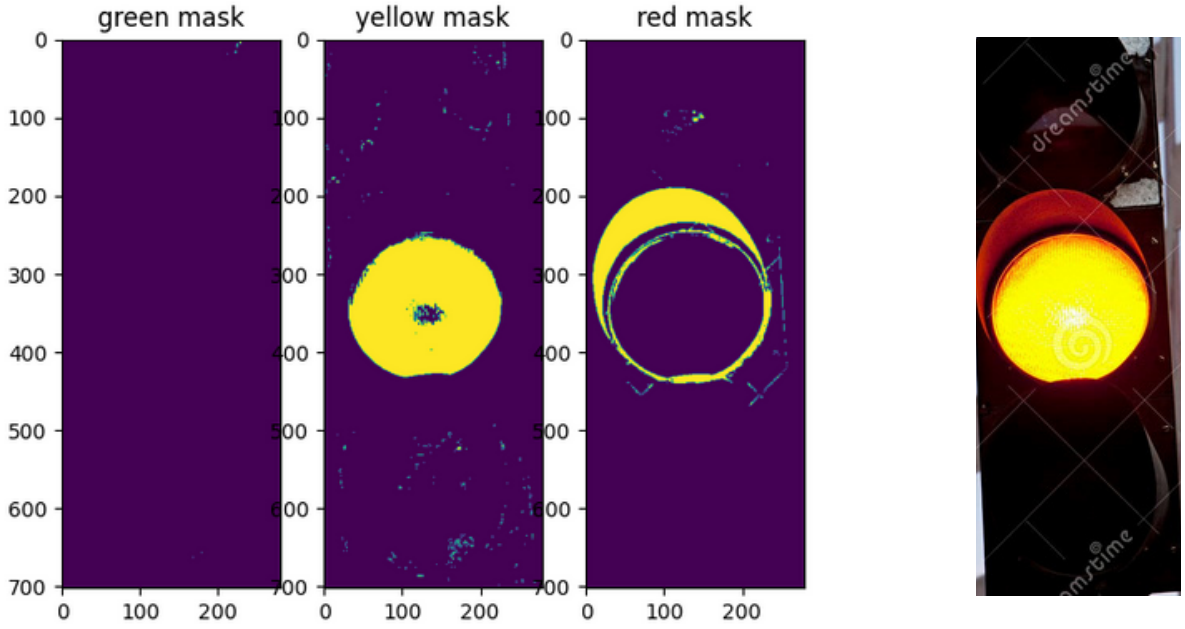
نتیجه بدست آمده برای چراغ زرد :

طبق پردازش انجام شده تعداد ۱۷ عکس از ۲۰ عکس موجود در دیتاست به درستی تشخیص داده شد و دقت تشخیص ۸۵ درصد شد. به دلیل نزدیک بودن رنگ چراغ زرد به چراغ قرمز در برخی از تصاویر که شدت رنگ زرد بیشتر بود یا به رنگ نارنجی متمایل بود، چراغ، قرمز تشخیص داده شد.



نمودار ۲ - دقت بدست آمده در چراغ زرد

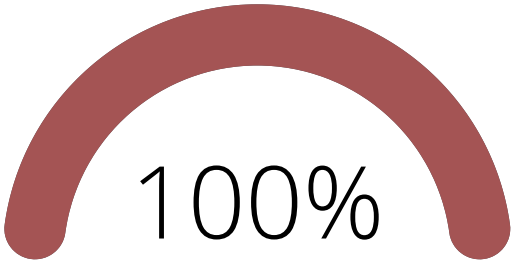
نمونه تشخیص :



شکل ۵- نمونه تشخیص چراغ زرد

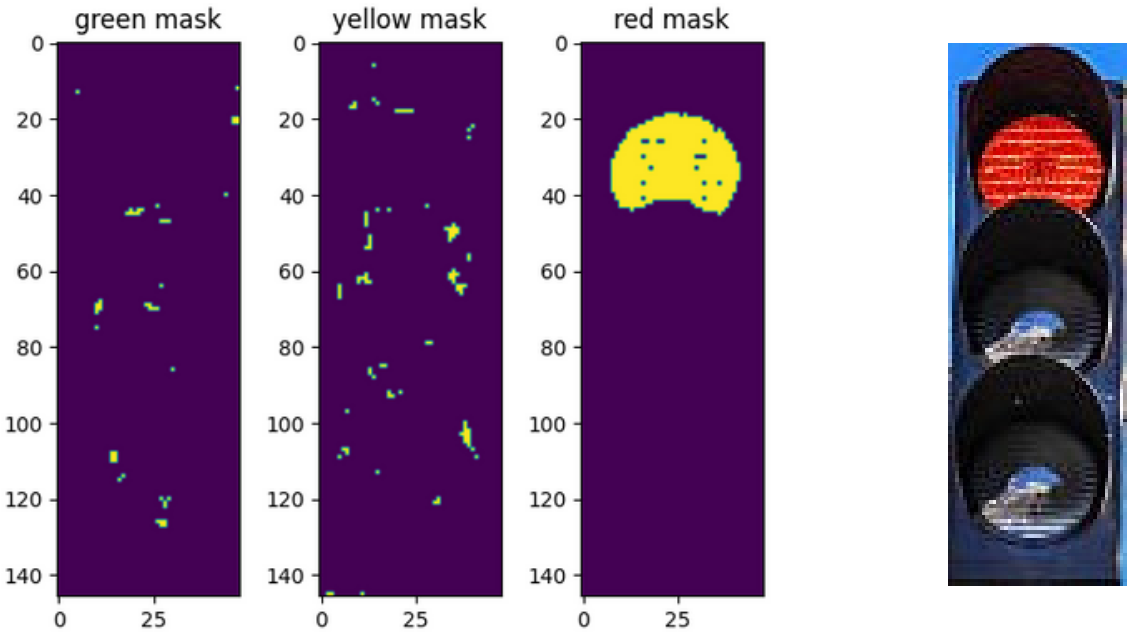
نتیجه بدست آمده برای چراغ قرمز :

طبق پردازش انجام شده تعداد ۲۰ عکس از ۲۰ عکس موجود در دیتاست به درستی تشخیص داده شد و دقت تشخیص ۱۰۰ درصد شد.



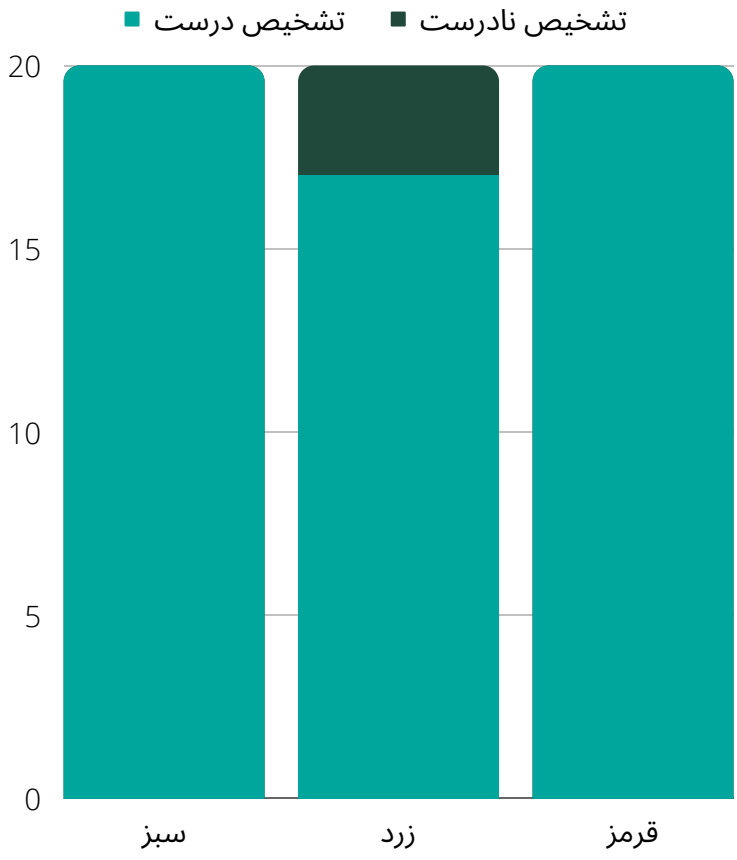
نمودار ۳ - دقت بدست آمده در چراغ قرمز

نمونه تشخیص :



شکل ۶- نمونه تشخیص چراغ قرمز

نتایج نهایی فاز اول:



نمودار ۴- جمعبندی نتیجه نهایی

PHASE2

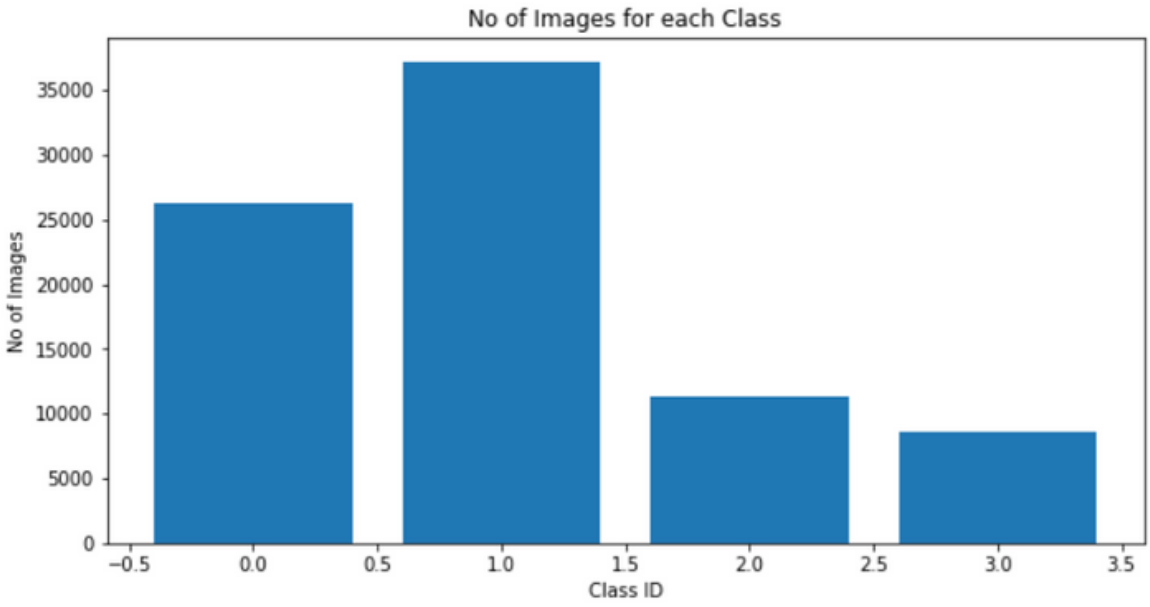
2 فاز دوم

- **مقدمه**

در فاز دوم قرار است بر روی دیتاستی کار کنیم که با استفاده از تکنیک OCT که برای تشخیص بیماری چشمی می باشد تهیه شده است. در این فاز می‌بایست دیتای موجود در ۴ دسته، کلاس‌بندی شود (این ۴ دسته شامل : Normal, CNV , DME , DRUSEN می باشد). این کار به کمک شبکه‌های CNN انجام شد و برای حل این سوال از سه مدل استفاده کردیم که شرح کار و نتایج در ادامه آمده است.

- **شرح کار**

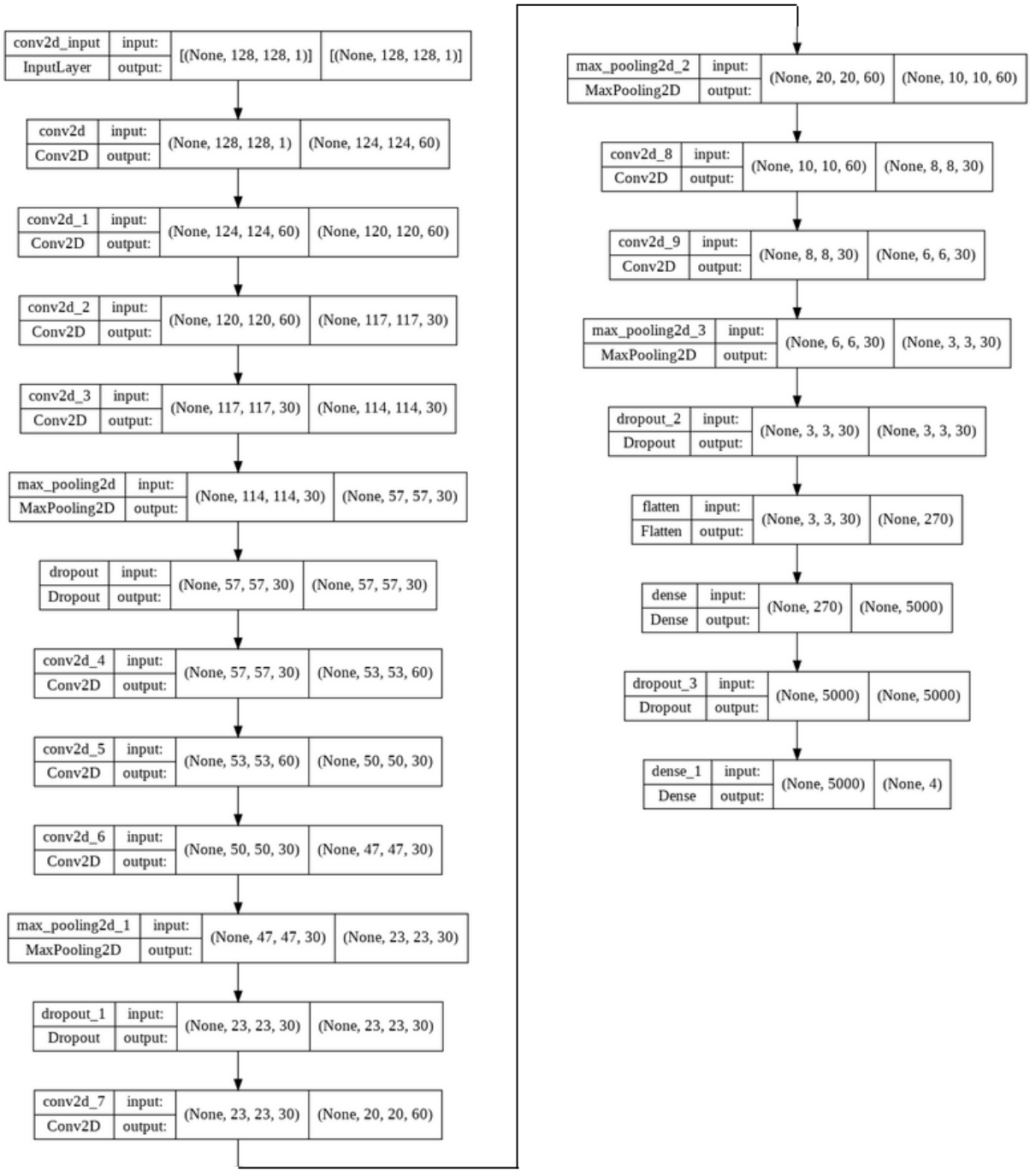
ابتدا دیتاست را خوانده و به بررسی آن پرداختیم. دیتاست شامل ۳ فولدر Test, Train, Validation بود که در هرکدام ۴ کلاس قرارداده شده بود. برای پیاده‌سازی شبکه نیاز بود ابتدا روی داده‌ها عملیات preprocessing انجام شود. ابتدا عکس‌های تمامی فولدرها را در یک متغیر ذخیره و لیبل‌های آن را از نام فولدر مربوطه استخراج کردیم. تعداد عکس‌های موجود در هر کلاس به شرح زیر است:



نمودار ۵- تعداد عکس‌های موجود در هر کلاس

- **تعریف مدل ۱**

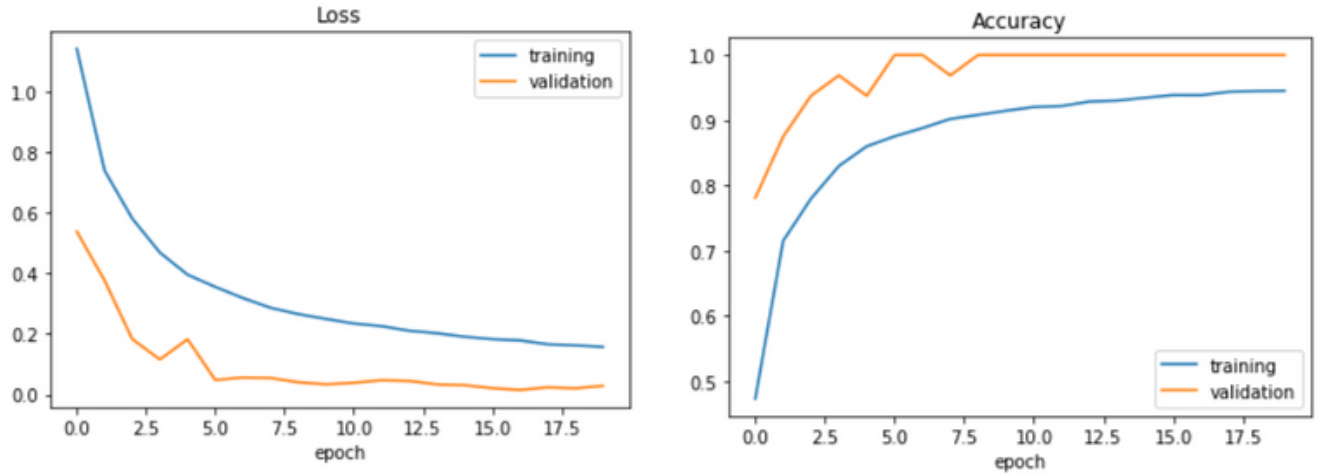
مدل استفاده شده در این فاز را با توجه به نیازهای موجود شخصی سازی کرده و نوشتیم. اکتیویشن فانکشن‌های لایه‌های میانی relu و لایه‌ی نهایی softmax انتخاب شد همچنین از اپتیمایزر Adam و loss = categorical_crossentropy استفاده کردیم. تعداد فیلترها و لایه‌های مدل در تصویر زیر قابل مشاهده می‌باشد. این مدل در فایل Q۲_۱.ipynb موجود است.



شکل ۷- مدل اول استفاده شده

نتیجه مدل ۱ :

نتایج بدست آمده از مدل ۱ به شرح زیر می‌باشد.



نمودار ۶- Accuracy , Loss

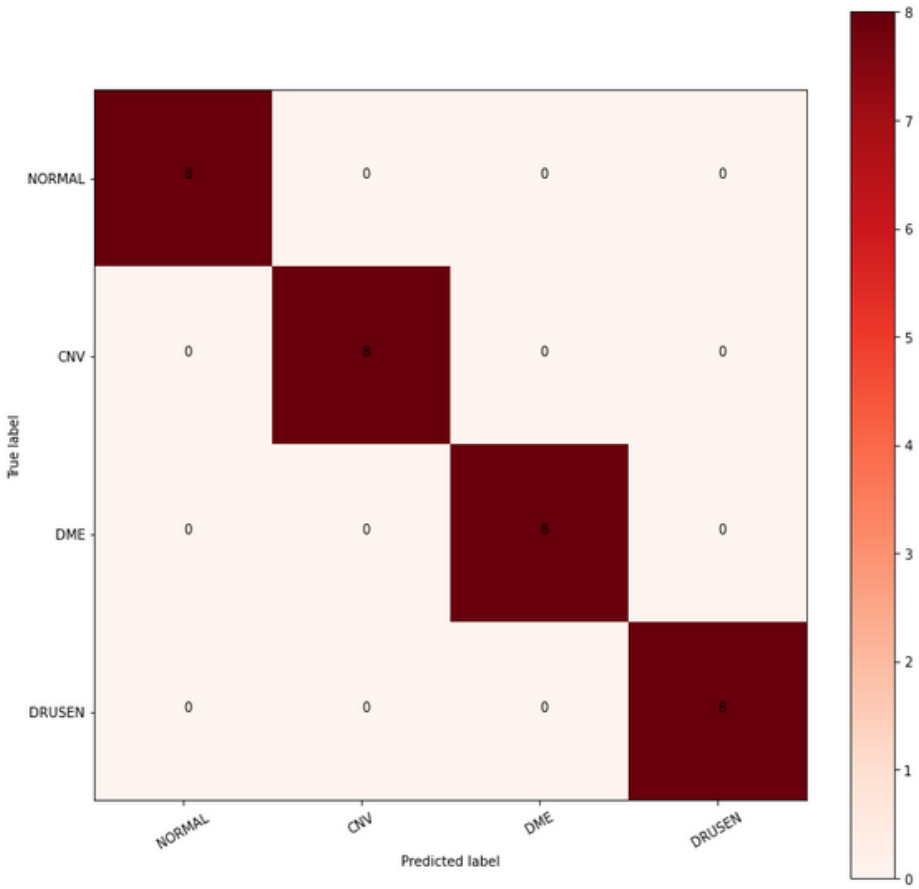
دقت داده‌های تست :

```
Test Score =  0.036954239010810 85
Test Accuracy =  0.9958677887916565
```

نتایج بدست آمده در epoch آخر:

```
Epoch 20/20
135/135 [==============================] - 99s 737ms/step - loss: 0.1575 - accuracy: 0.9452 - val_loss: 0.0284 - val_accuracy: 1.0000
```

کانفیوژن ماتریس:



شکل ۸- کانفیوژن ماتریس

Precision , Recall , F1 - Score

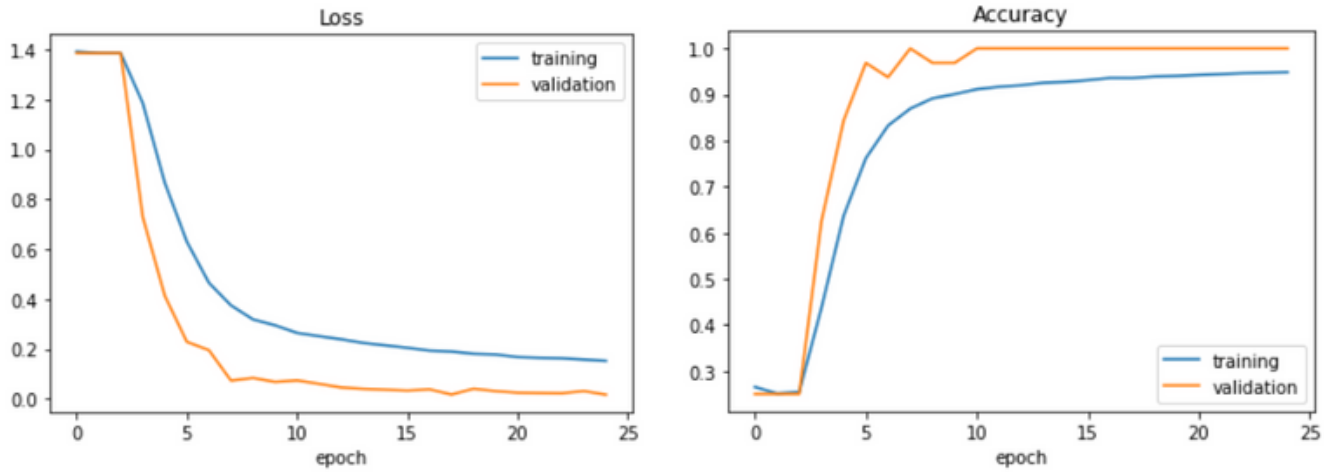
	precision	recall	f1-score	support
0	1.00	1.00	1.00	8
1	1.00	1.00	1.00	8
2	1.00	1.00	1.00	8
3	1.00	1.00	1.00	8
accuracy			1.00	32
macro avg	1.00	1.00	1.00	32
weighted avg	1.00	1.00	1.00	32

- **تعریف مدل ۲**

اگرچه در مدل ۱ به دقت بسیار بالایی دست یافتیم اما در جهت بهبود نتایج و همچنین بررسی تاثیر تغییرات مدل دوم را نیز ساختیم. این مدل مانند مدل اول می‌باشد با این تفاوت که دو لایه شامل کانولوشن، دراپ اوت و مکس پولینگ در میان آن اضافه کردیم. همچنین تعداد epochهارا از ۲۰ به ۲۵ افزایش دادیم که نتایج در ادامه قابل مشاهده می‌باشد. این مدل در فایل Q۲_۲.ipynb قرار داده شده‌است.

نتیجه مدل ۲ :

نتایج بدست آمده از مدل ۲ به شرح زیر می‌باشد.



نمودار ۷- Accuracy , Loss

دقت داده‌های تست :

```
Test Score =  0.045889340341091156
Test Accuracy =  0.9927685856819153
```

نتایج بدست آمده در epoch آخر:

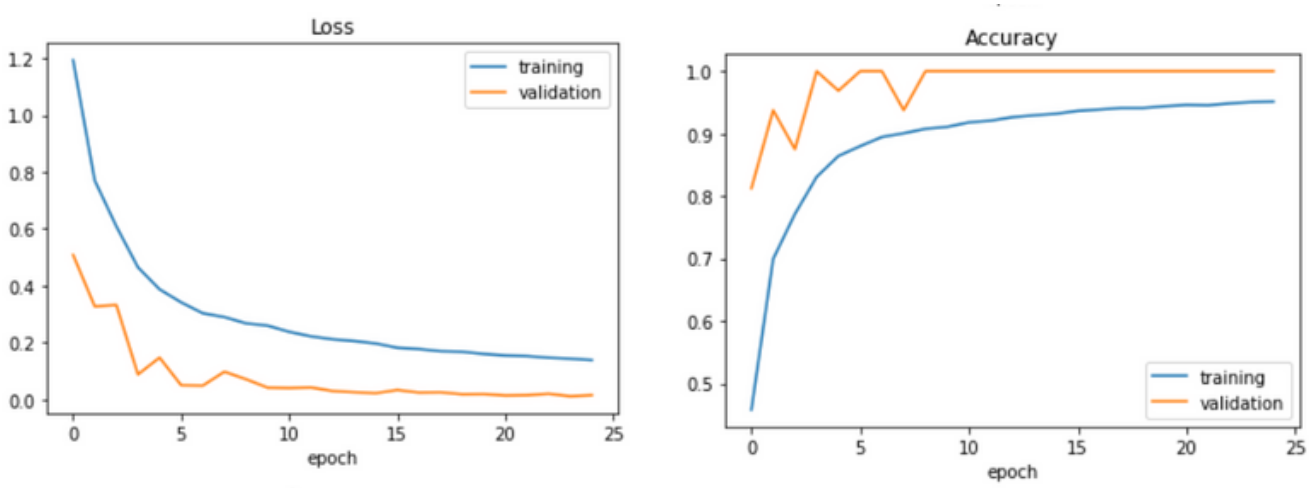
```
Epoch 25/25
135/135 [==============================] - 96s 710ms/step - loss: 0.1523 - accuracy: 0.9486 - val_loss: 0.0168 - val_accuracy: 1.0000
```

- **تعریف مدل ۳**

پس از آنکه دقت مدل دوم کمتر از مدل اول شد، مدل سوم را با تغییر در تعداد فیلترها (افزایش) مدل اول ساخته و Train کردیم. که نتایج در ادامه قابل مشاهده است. این مدل در فایل Q۲_۳.ipynb گنجانده شده.

نتیجه مدل ۳ :

نتایج بدست آمده از مدل ۳ به شرح زیر می‌باشد.



نمودار ۸- Accuracy , Loss

دقت داده‌های تست :

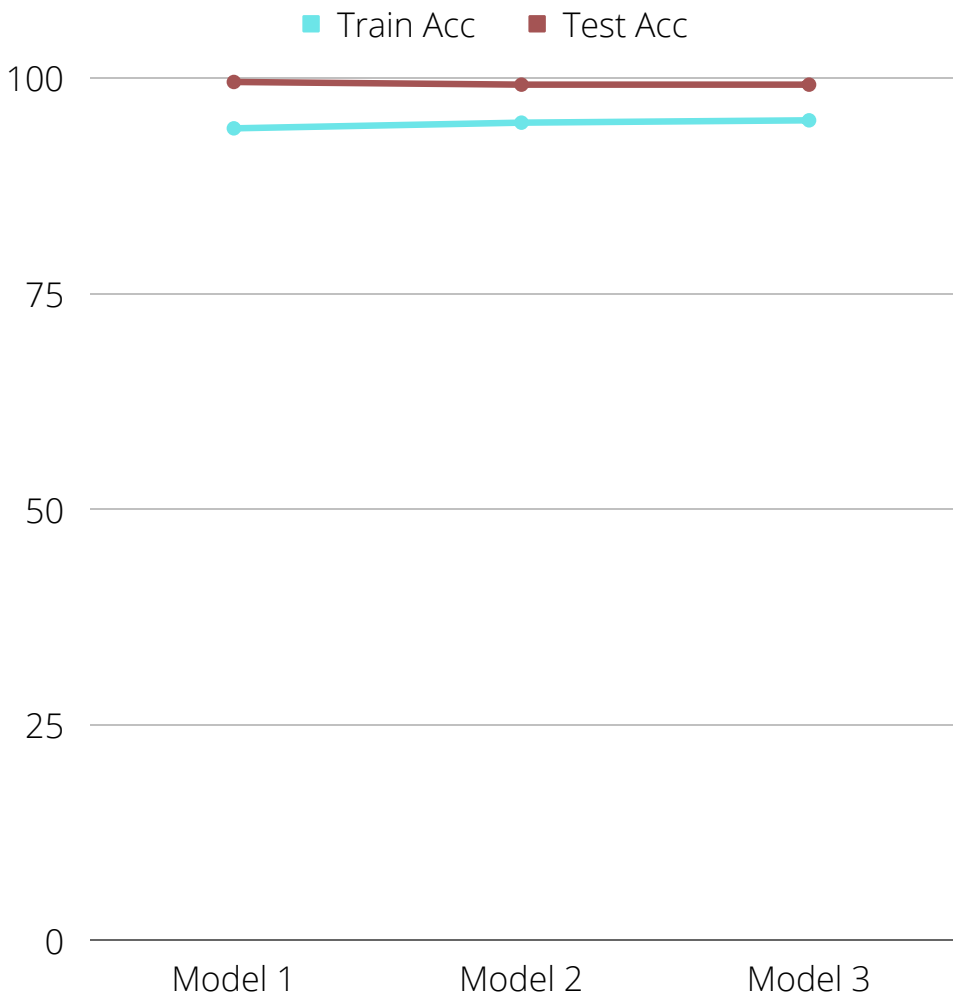
```
: Score =  0.039610143750905 99
: Accuracy =  0.9927685856819153
```

نتایج بدست آمده در epoch آخر:

```
Epoch 25/25
135/135 [==============================] - 216s 2s/step - loss: 0.1390 - accuracy: 0.9513 - val_loss: 0.0153 - val_accuracy: 1.0000
```

نتایج نهایی فاز دوم:

با توجه به نتایج بدست آمده مشخص است که با افزایش تعداد لایه‌ها در مدل دوم دقت داده‌های Train افزایش یافته است اما دقت داده‌های test اندکی کاهش یافته است. در مدل سوم نیز با افزایش تعداد فیلتر ها دقت داده‌های train از مدل ۱ و ۲ بیشتر شده اما دقت داده‌های test نسبت به مدل ۱ کاهش و نسبت به مدل ۲ تغییری نداشته است.
بهترین دقت بدست آمده برای مدل اول با ۹۹.۵۸ درصد بدست آمد.



نمودار ۹- مقایسه مدل های ۱ و۲ و۳

PHASE 3

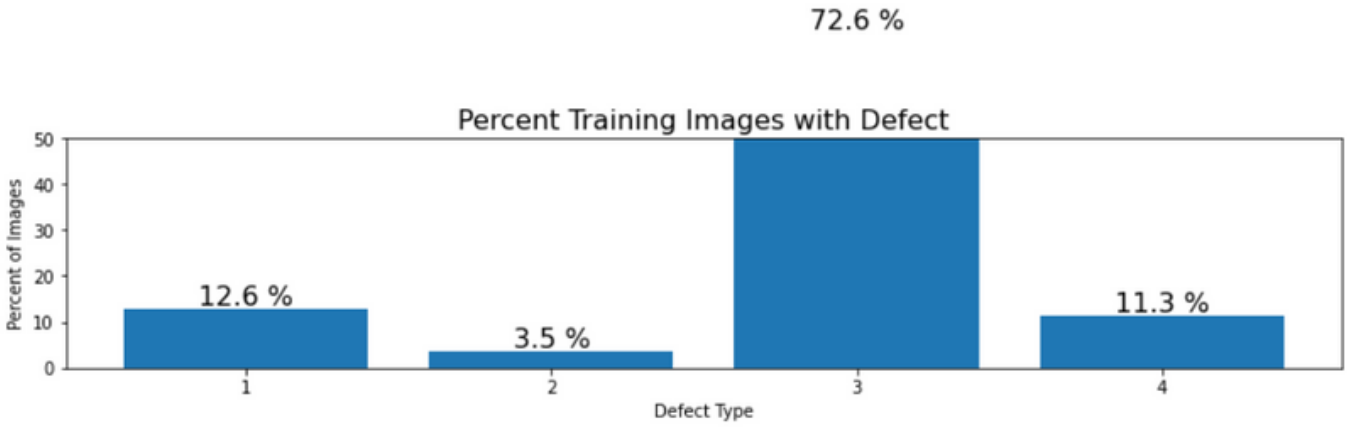
3 فاز سوم

- **مقدمه**

در فاز سوم قرار است بر روی دیتاستی کار کنیم که شامل تصاویری از ورقه‌های فولادی دارای سه نوع نقص می باشد. در این فاز می‌بایست ایرادات موجود شناسایی، محل آنها مشخص و نوع آن‌ها کلاس‌بندی شود. در انجام این فاز از Image Segmentation و دو مدل FUNet و Unet استفاده شده است که شرح کار و نتایج در ادامه آمده است.

- **شرح کار**

ابتدا دیتاست را خوانده و به بررسی آن پرداختیم. برای پیاده‌سازی شبکه نیاز بود ابتدا روی داده‌ها عملیات preprocessing انجام شود. در نهایت داده‌ها بصورت مورد نیاز ما درامد که در بخش مربوط به هر مدل دیتای اولیه و دیتای پردازش شده قابل مشاهده خواهد بود.



نمودار ۱۰- درصد عکس‌های موجود در هر کلاس

- **مدل U-net**

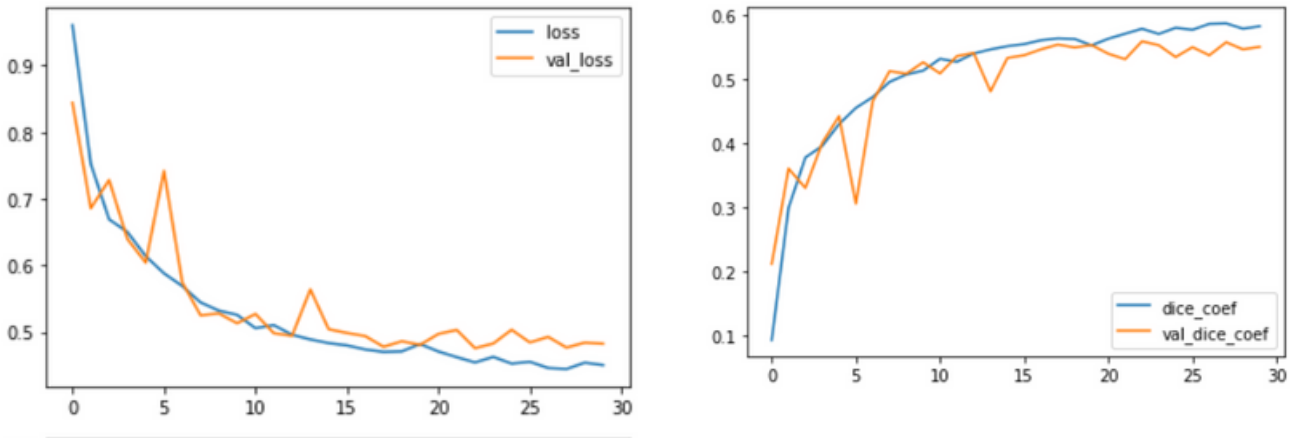
مدل اول :

مدل استفاده شده در این فاز مـــــــدل u-net می باشد. در این مـــــــدل از اپتیمایزر Adam و loss = bce_dice_loss و metrics = [dice_coef] استفاده کردیم. این مدل در فایل Q۳_۱.ipynb قرار دارد.

این مدل با توجه به نوت بوک موجود در سایت kaggle با batch size = ۱۶ به دقت حدود ۴۰ درصد می‌رسید. به منظور افزایش دقت این مدل را با batch size = ۸ ترین کردیم و نتایج بدست آمده به شرح زیر است.

نتیجه مدل ۱ :

نتایج بدست آمده از مدل ۱ به شرح زیر می‌باشد.



نمودار ۱۱- Loss , dice_coef

نتایج به دست آمده در epoch آخر:

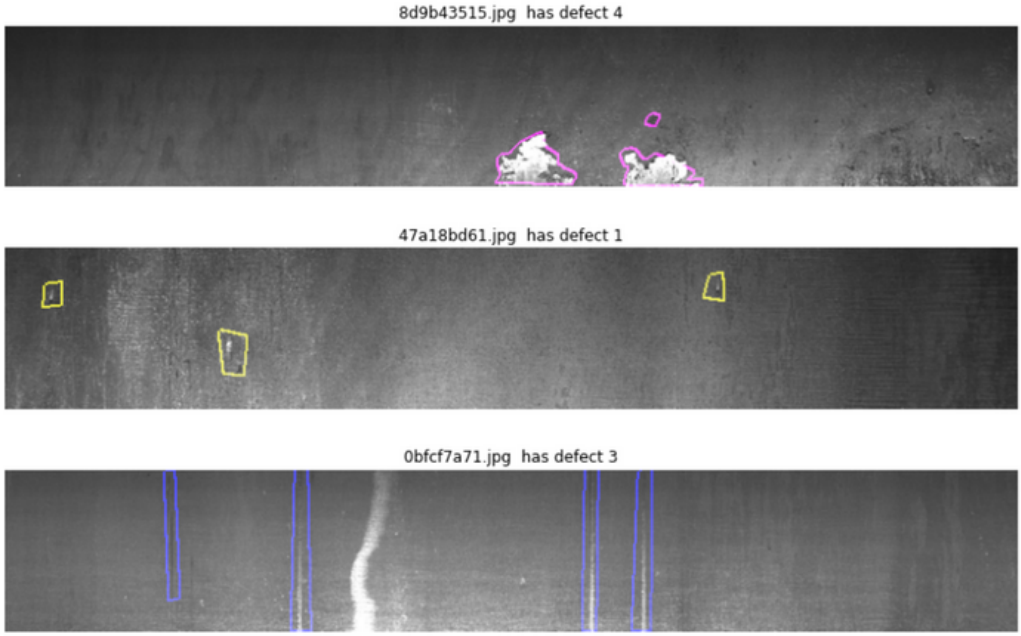
```
Epoch 30/30
1335/1335 [==============================] - 425s 318ms/step - loss: 0.4503 - dice_coef: 0.5831 - val_loss: 0.4824 - val_dice_coef: 0.5510
```

توجه

این مدل با batch size = ۴ نیز ترین شد اما به دلیل پایین بودن دقت از آوردن نتایج خودداری می‌کنیم.

مدل دوم :

مدل استفاده شده در این فاز مدل u-net می باشد که از کتابخانه‌ی segmentation_models فراخوانی شده است. در این مدل از اپتیمایزر Adam و loss = bainary_crossentropy و metrics = [dice_coef] استفاده کردیم. این مدل با ترنسفر لرنینگ و استفاده از وزن‌های بدست آمده از ترین‌کردن مدل resnet-۵۰ روی دیتاست image-net آموزش داده شده‌است. این مدل را می‌توانید در فایل Q۳_۲.ipynb مشاهده کنید.



شکل ۹- نمایش ماسک تعدادی از داده‌ها

نتیجه مدل ۲ :

نتایج بدست آمده از مدل ۲ به شرح زیر می‌باشد.

```
Epoch 20/20
354/354 - 246s - loss: 0.0157 - dice_coef: 0.6532 - val_loss: 0.0259 - val_dice_coef: 0.5708 - 246s/epoch - 694ms/step
```

مشاهده می‌شود که دقتمان به نسبت مدل اول ، ۲ درصد افزایش یافته است.

مجددا، اپتیمایزر را به SGD تغییر دادیم اما تغییر مشهودی مشاهده نشد. این اقدام در فایل Q۳_۳.IPYNB انجام شد.

و در نهایت loss را به bce_dice_loss تغییر دادیم و بالاترین دقت بدست آمد. شایان ذکر است که bce_dice_loss را خارج از loss های کتابخانه keras و بصورت دستی تعریف کردیم. در نوت‌بوک Q۳_۴.ipynb می‌توانید این تلاش را ببینید.

```
Epoch 50/50
354/354 - 242s - loss: 0.2377 - dice_coef: 0.7882 - val_loss: 0.3806 - val_dice_coef: 0.6671 - 242s/epoch - 683ms/step
```

مشاهده می‌شود که دقت ۱۰ درصد افزایش یافته است.

- **مدل FCN-8**

مدل اول :

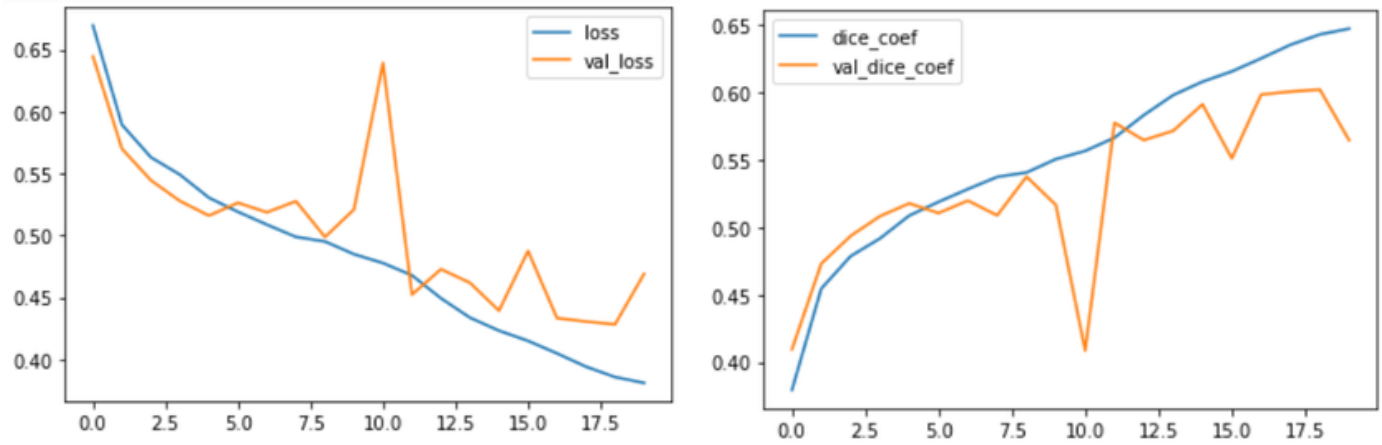
از مدل FCN-۸ استفاده کردیم. اپتیمایزر Adam را به کار گرفتیم و از loss = bce_dice_loss و metrics = [dice_coef] استفاده کردیم.

بچ‌سایز را برابر ۸ قرار دادیم. بیشترین دقت مدل برابر ۶۰ درصد به دست آمد. فایل Q۳_۵.ipynb حاوی این مدل است.

```
Epoch 19/20
1335/1335 [==============================] - 459s 344ms/step - loss: 0.3857 - dice_coef: 0.6433 - val_loss: 0.4282 - val_dice_coef: 0.6024
```

نتیجه مدل ۱ :

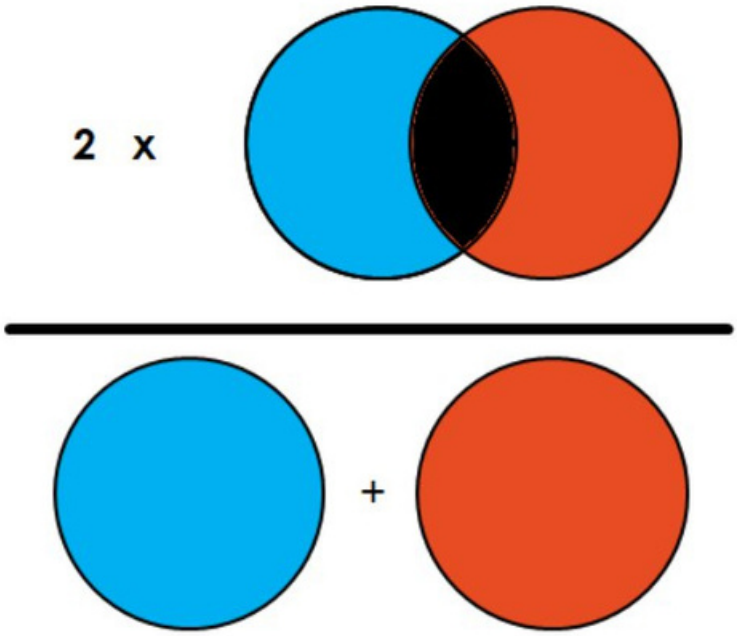
نتایج بدست آمده از مدل ۱ به شرح زیر می‌باشد.



نمودار ۱۲- dice_coef , Loss

معیار ارزیابی عملکرد Segmentation Model:

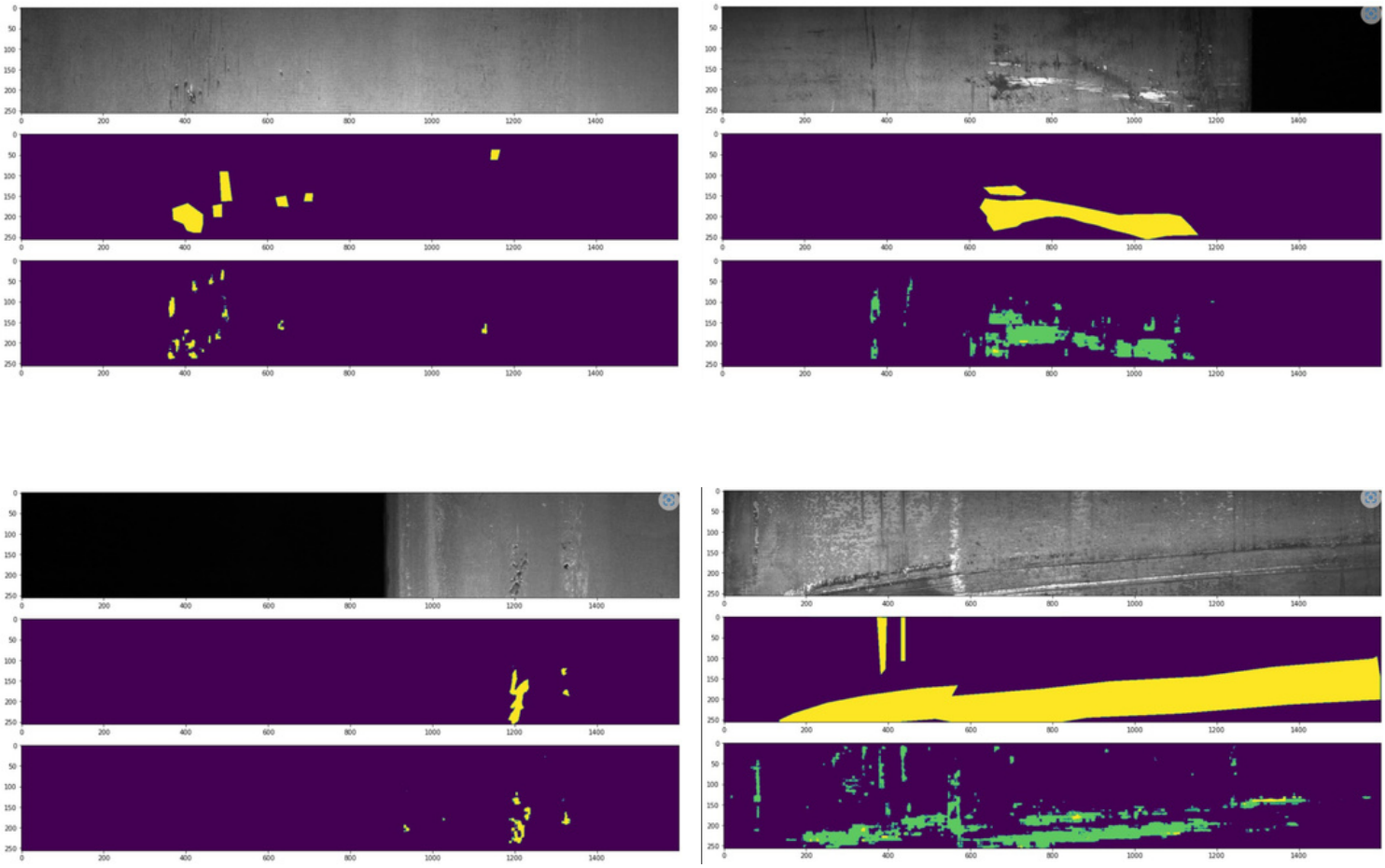
معیار Dice Coefficient عبارت است از دوبرابر مساحت اشتراک دو ناحیه، تقسیم بر اجتماع آن دو. این معیار بسیار شبیه به معیار IoU است؛ به طوری که اگر مطابق یکی از این دو معیار عملکرد یک مدل در سگمنتیشن یک عکس بهتر از مدل دیگری باشد، آن‌گاه مطابق معیار دیگر هم همین شرایط برقرار است. هم Dice Coefficient و هم IoU از صفر تا یک تغییر می‌کنند. یک نمایانگر بهترین عملکرد، و صفر نمایانگر ضعیف‌ترین عملکرد است.



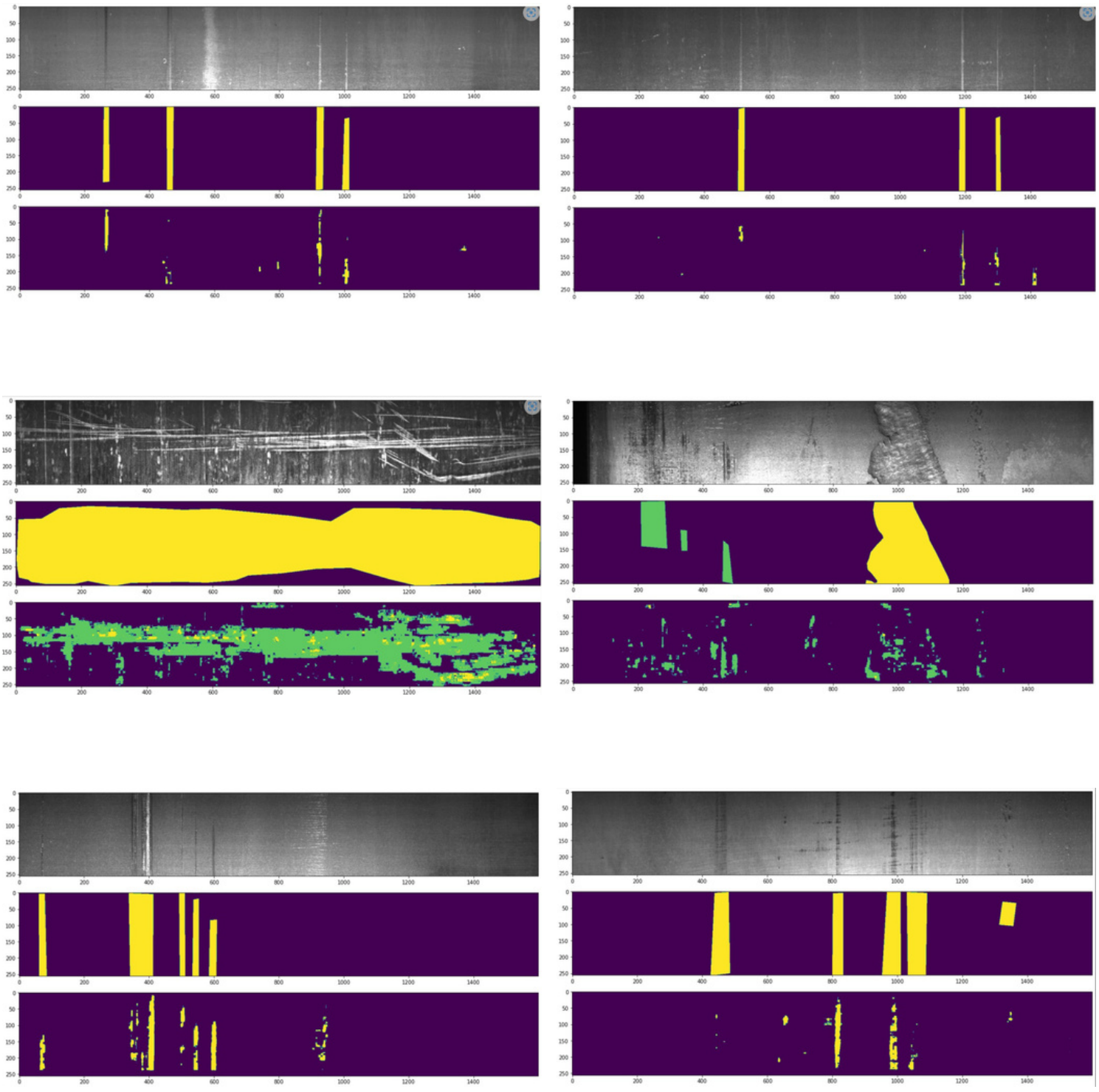
شکل ۱۰- معیار ارزیابی عملکرد Segmentation Model

نتیجه‌گیری:

نتایج prediction مدل FCN-۸ روی ده عکس خواسته‌شده در فاز سوم (موجود در فایل Q۳_۶.ipynb):



شکل ۱۱ تا ۱۴ - نتایج FCN ۸



شکل ۱۵ تا ۲۰ - نتایج FCN ۸

References

https://www.ncbi.nlm.nih.gov/books/NBK554044/

https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2

https://kaggle.com

https://pinterest.com

Presented by:

Thanks for your attention