گزارش پروژه ی هوش مصنوعی

مریم حق شمار

توضیح کد: در ابتدا رنگ هارا به شکل رندم و به صورت آرایه از numpy میسازیم . هر سطر سه ستون دارد که هر رنگ را نشان میدهد و ابتدا مقدار رندمی ین ۰ تا ۲۵۵ میگیرد.

برای خواندن تصویر از کتابخانه OpenCV استفاده میکنیم. که تصویر را به شکل آرایه numpy به ما میدهد. برای انجام ادامه محاسبات تصویر را به حالت خطی درمی آوریم.
بر روی این تصویر خطی الگوریتم خود را پیاده سازی میکنیم. در اینجا از الگوریتم یادگیری hillClimbing استفاده کرده ایم که امکان خطای افتادن در مینیمم لوکال و عدم تشخیص رنگ صحیح را دارد اما به جهت بالا بودن سرعت محاسبات آن، انتخابش میکنیم.

def getNeighbours(image, colors, step)

در این تابع، رنگ های همسایه آرایه رنگ های موجود را جستجو میکنیم و مقدار فیتنس هر سری رنگ را ذخیره میکنیم. در ابتدا که هنوز یادگیری انجام نشده است، در بازه های بزرگ تر میگردیم و در استپ های نهایی بازه ی جستجو را محدود میکنیم. برای کاهش اردر زمانی ما فقط تعدادی رندم از همسایه ها را جستجو میکنیم و همه ی آن ها را بازدید نمیکنیم.

def getBestNeighbour(neighbours, fitnesses)

در این تابع فیتنس های محاسبه شده را جستجو کرده و بهترین آن ها را برمیگردانیم. همچنین سری رنگ هایی که باعث رسیدن به این فیتنس شده اند را برمیگردانیم.

```
step:  2 *** bestFitness:  6.083957396029032 *** bestNeighbour:
 [[141  79 135]
 [201 119  91]
 [222 192  44]]
********************************************************************************
```

def hillClimbing(image, colors)

در هر استپ، ابتدا همسایه های سری رنگ را جستجو میکند، سپس بهترین آن ها برمیگرداند و اگر فیتنس آن بهتر از فیتنس کل بشد، فیتنس کل را جایگزین میکند. در پایان همه ی استپ ها بهترین سری(لوکال) رنگ هارا یافته ایم.

def fitness(image, colors)

برای محاسبه فیتس طبق فرمول داده شده از تابع np.linalg که فاصله اقلیدوسی ۲ آرایه numpy را حساب میکند، استفاده کردیم.

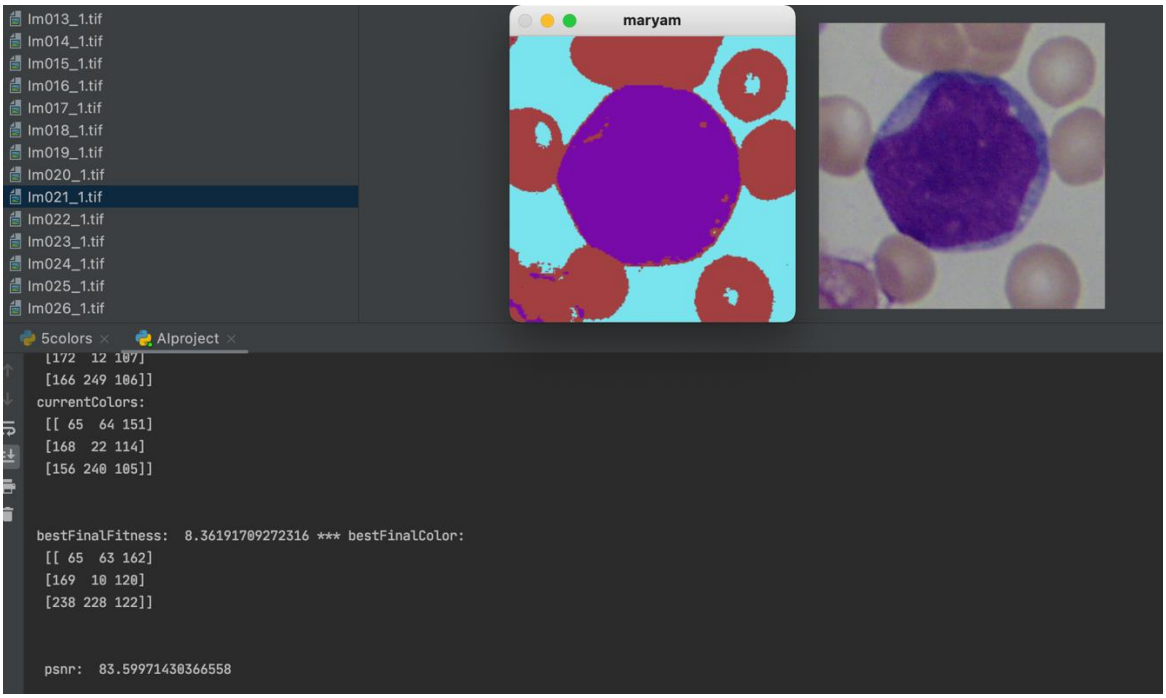
def reconstruct_image(image, bestcolor)

در این تابع بر اساس رنگ های بدست آمده و تصویر اولیه، تصویر را بازسازی میکنیم. البته توجه شود که برای افزایش سرعت محاسبات، الگورتیم یاگیری را در تصویر با اندازه ی نصف اجرا کردیم. اما بر حسب تصویر اصلی بازسازی میکنیم که این کار دقت را کم میکند.

PSNR(original, reconstructed)

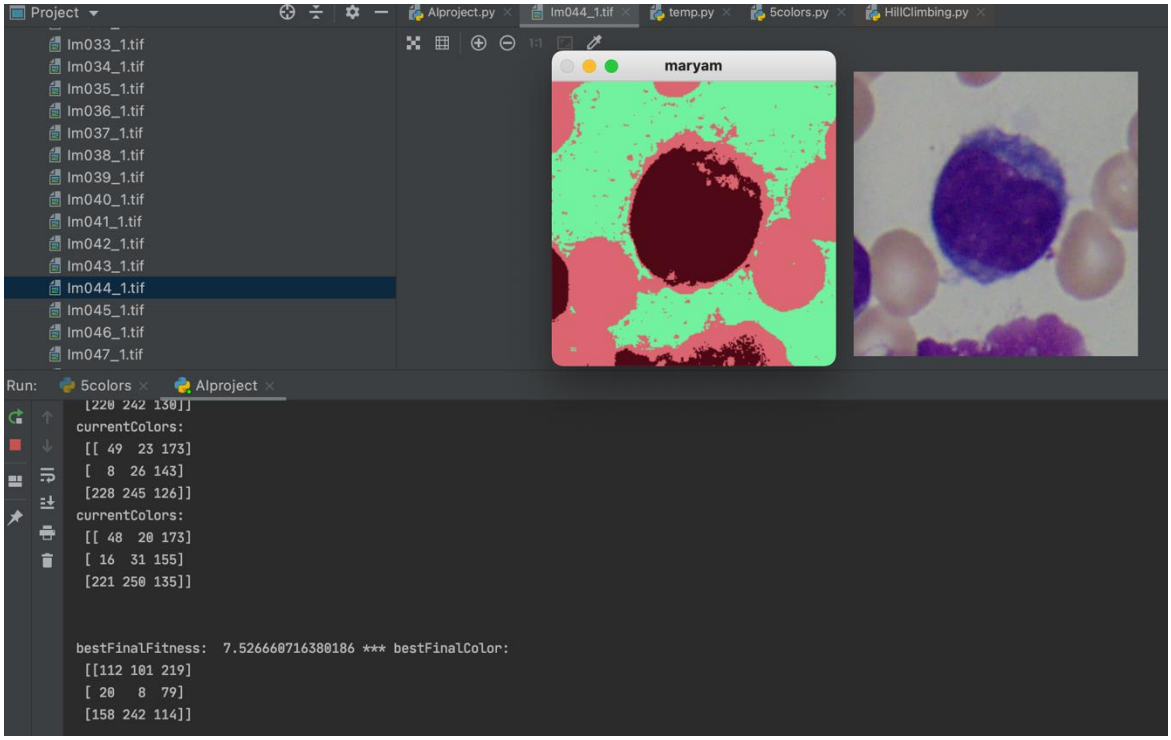
بر حسب فرمول داده شده، اختلاف تصویر بازسازی شده و تصویر اصلی را محاسبه میکنیم.

نتایج برای حالت ۳ رنگ :

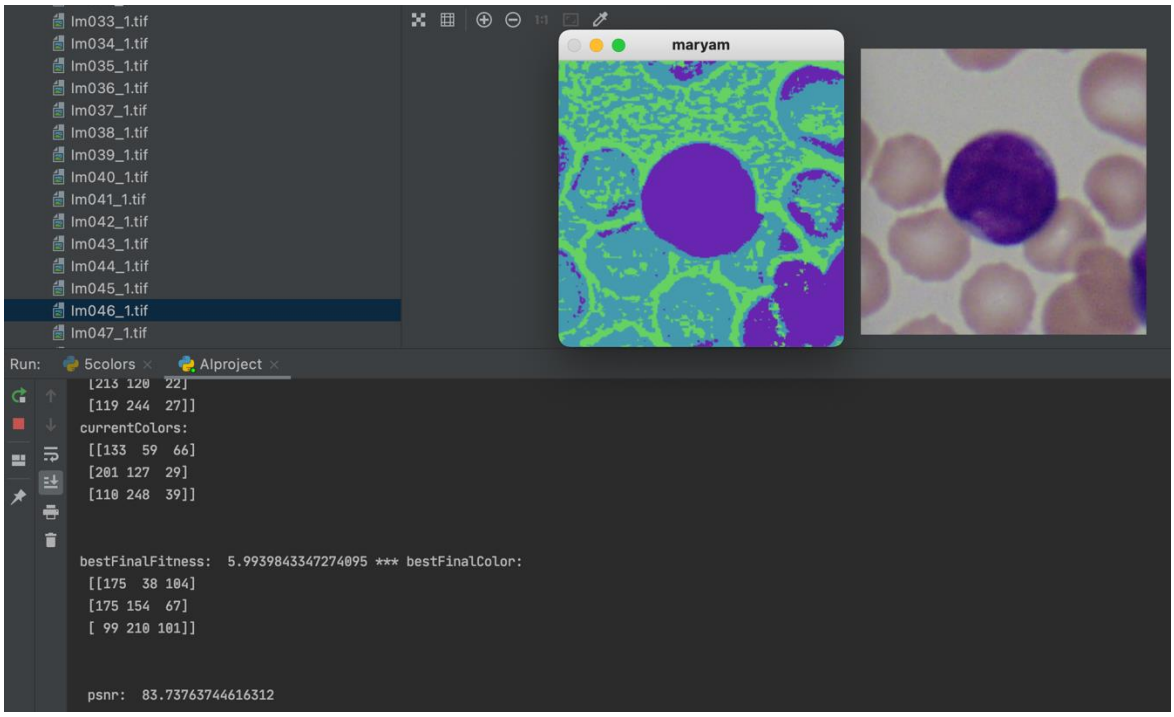


Fitness = 8.35

PSNR = 83.59

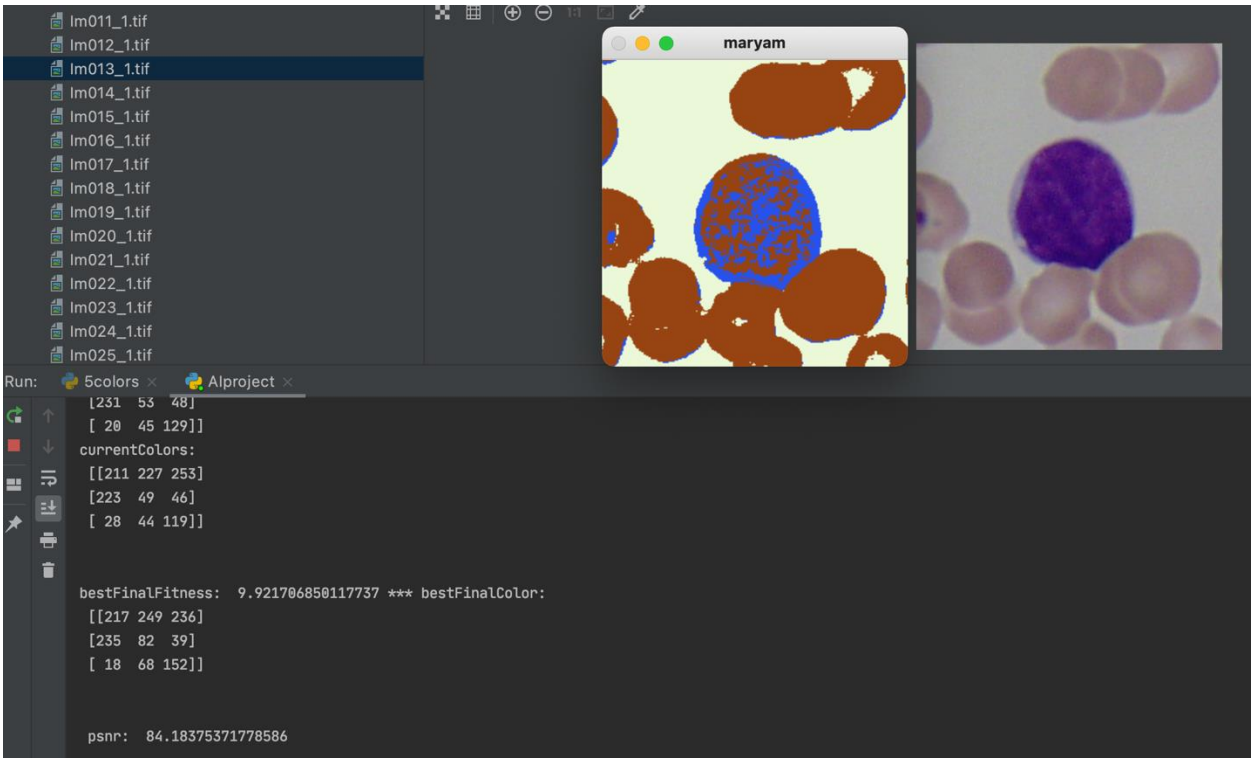


Fitness = 7.52
PSNR = 84.2



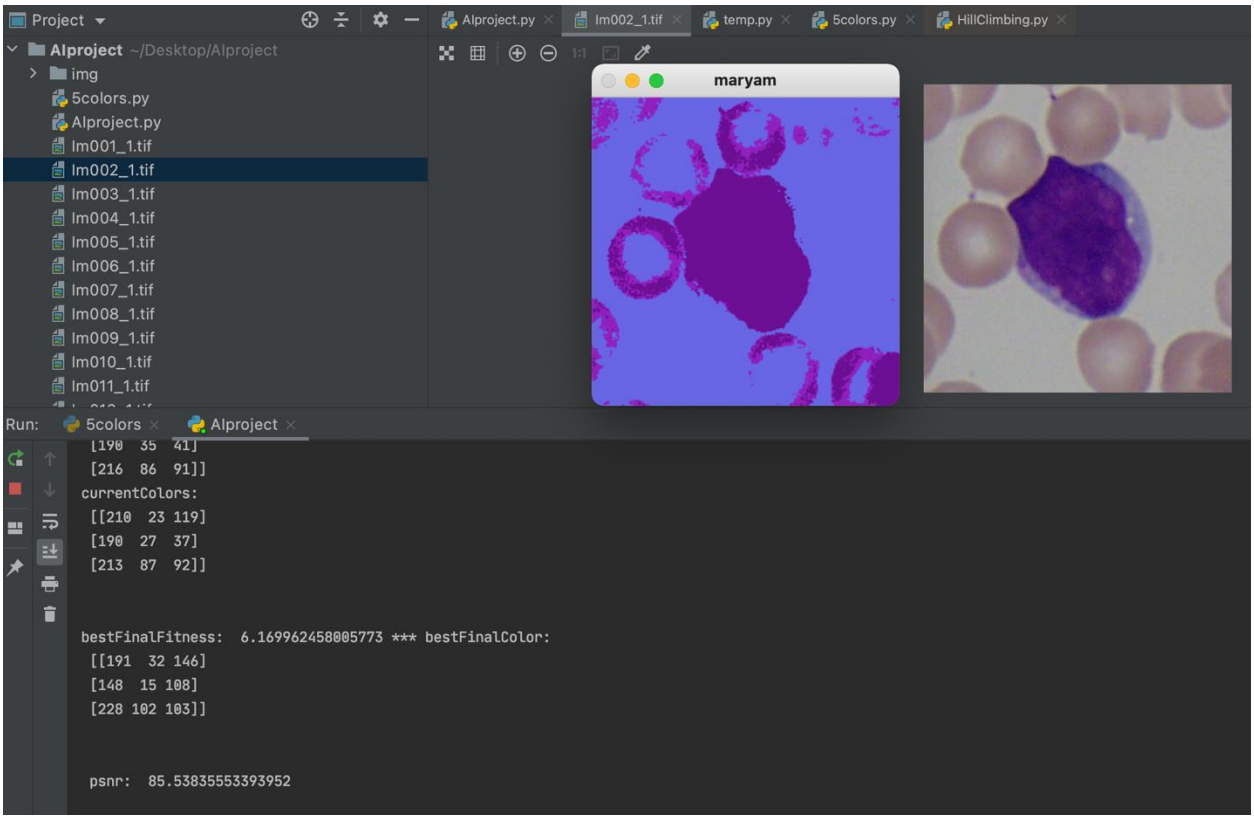
Fitness = 5.99

PSNR = 83.73



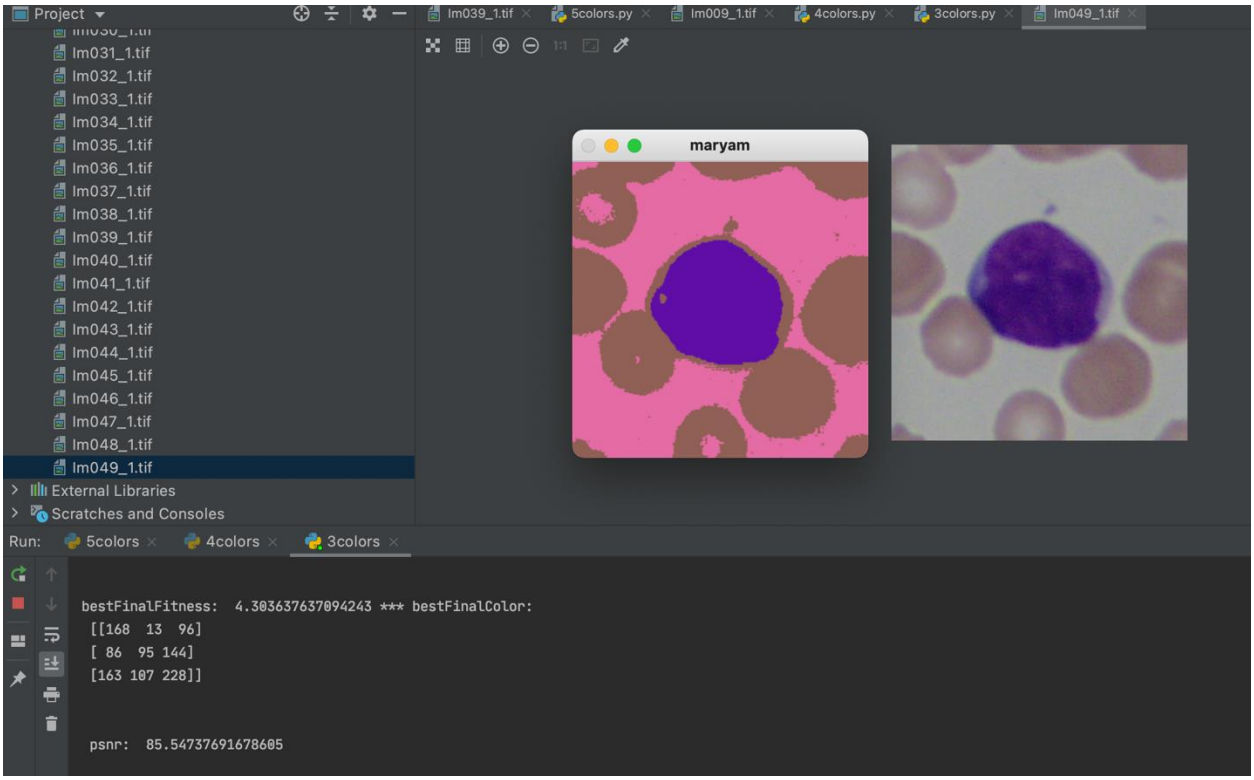
Fitness = 9.21

PSNR = 84.18



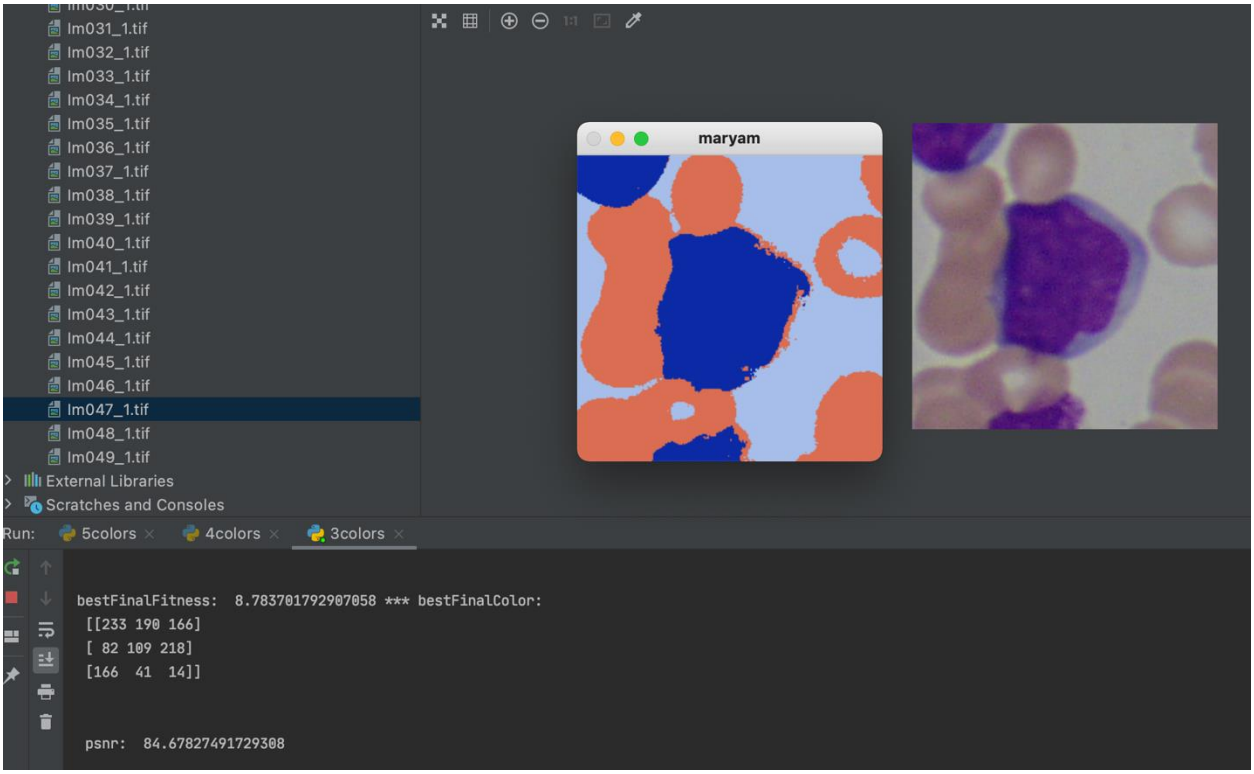
Fitness = 6.16

PSNR = 85.53



Fitness = 4.30

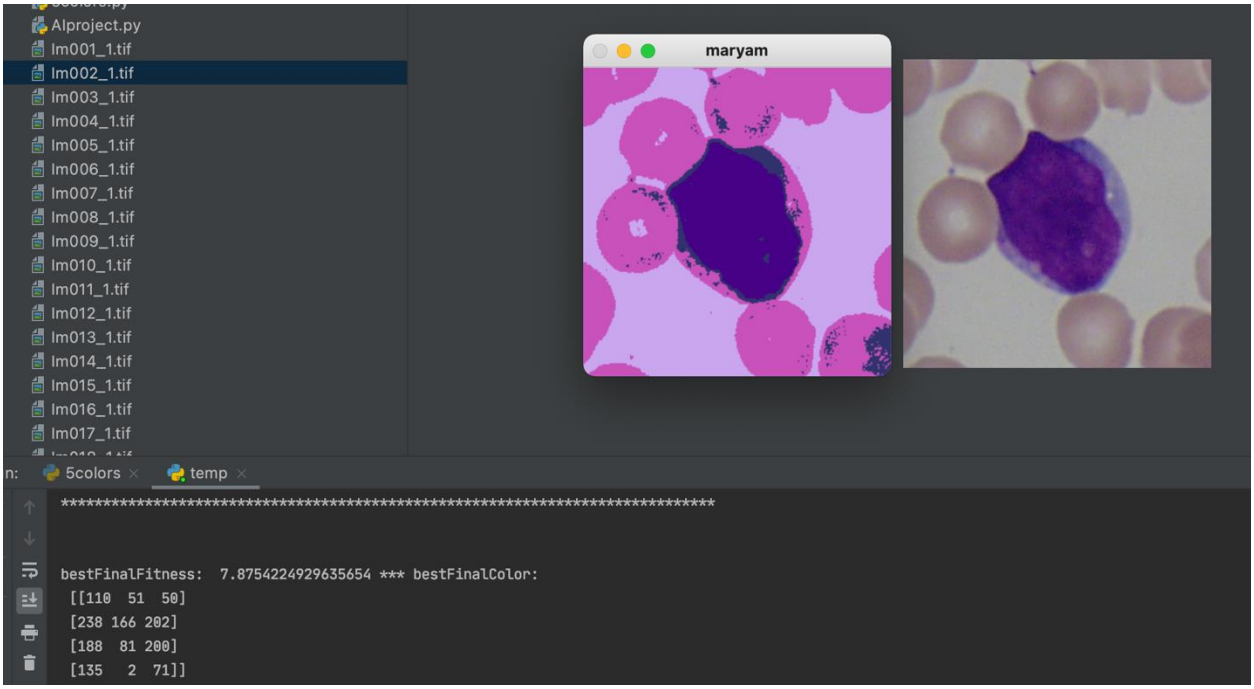
PSNR = 85.54



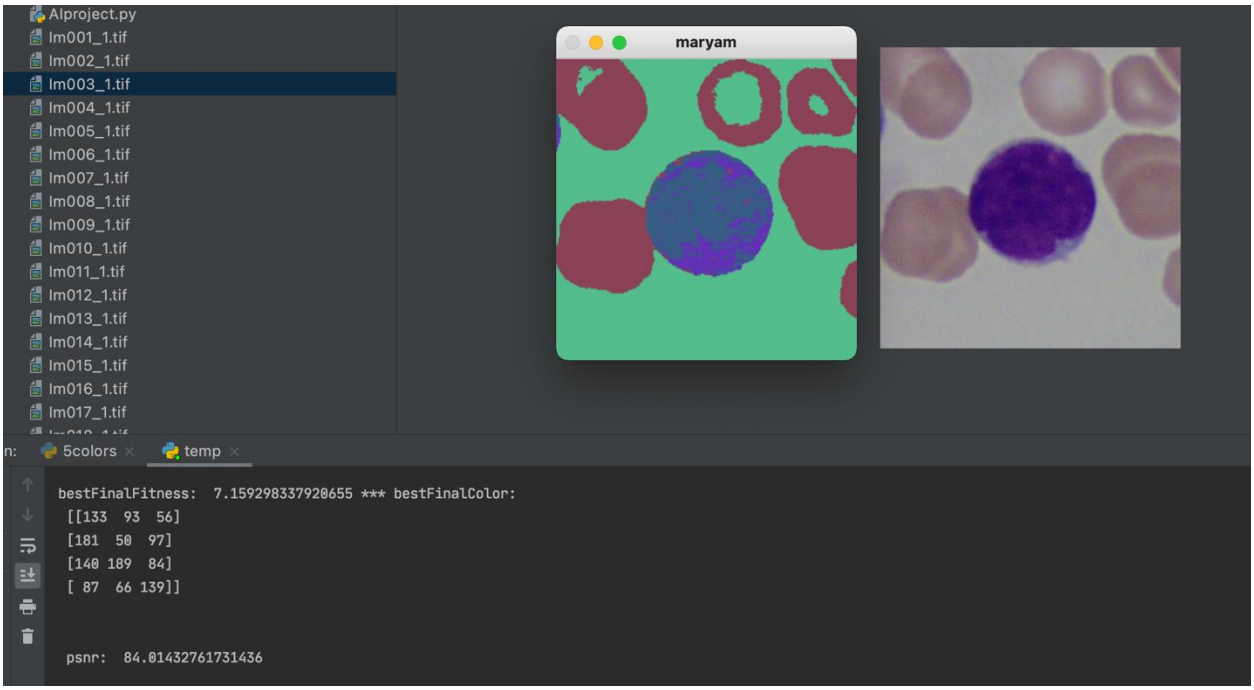
Fitness = 8.35

PSNR = 83.59

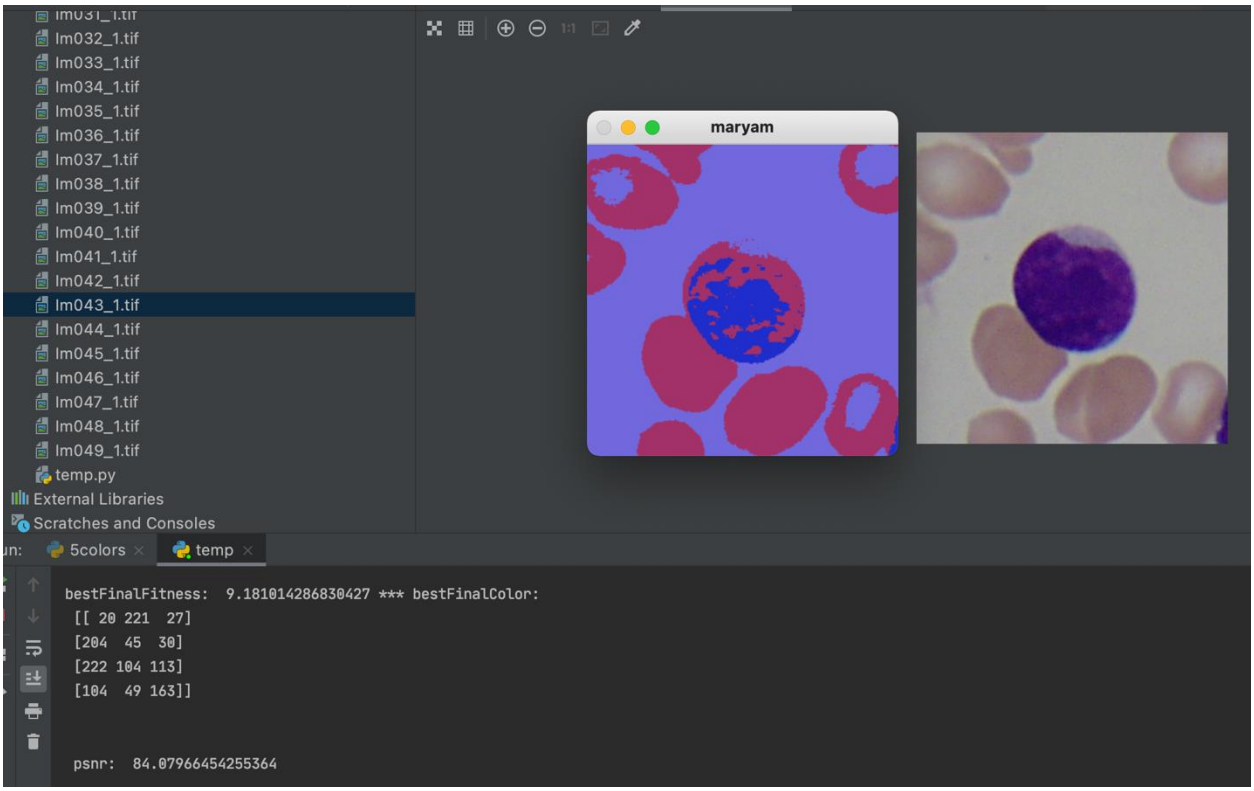
نتایج برای حالت ۴ رنگ :



Fitness = 7.87
PSNR = 85.41

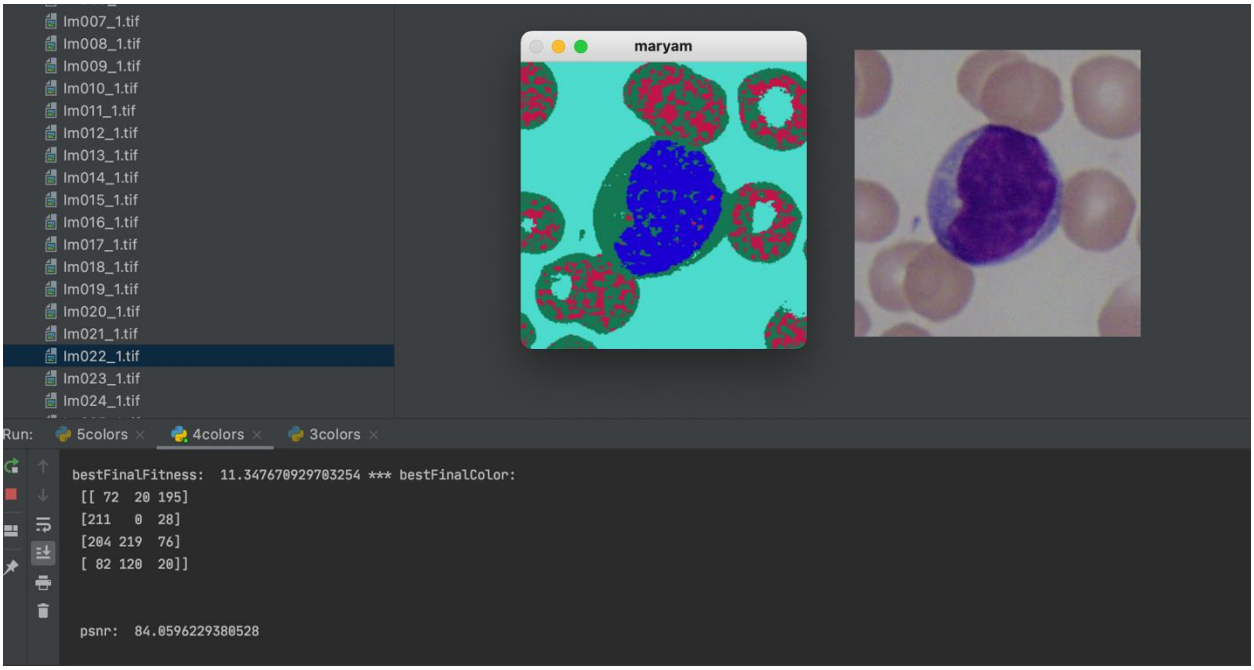


Fitness = 7.15
PSNR = 84.01



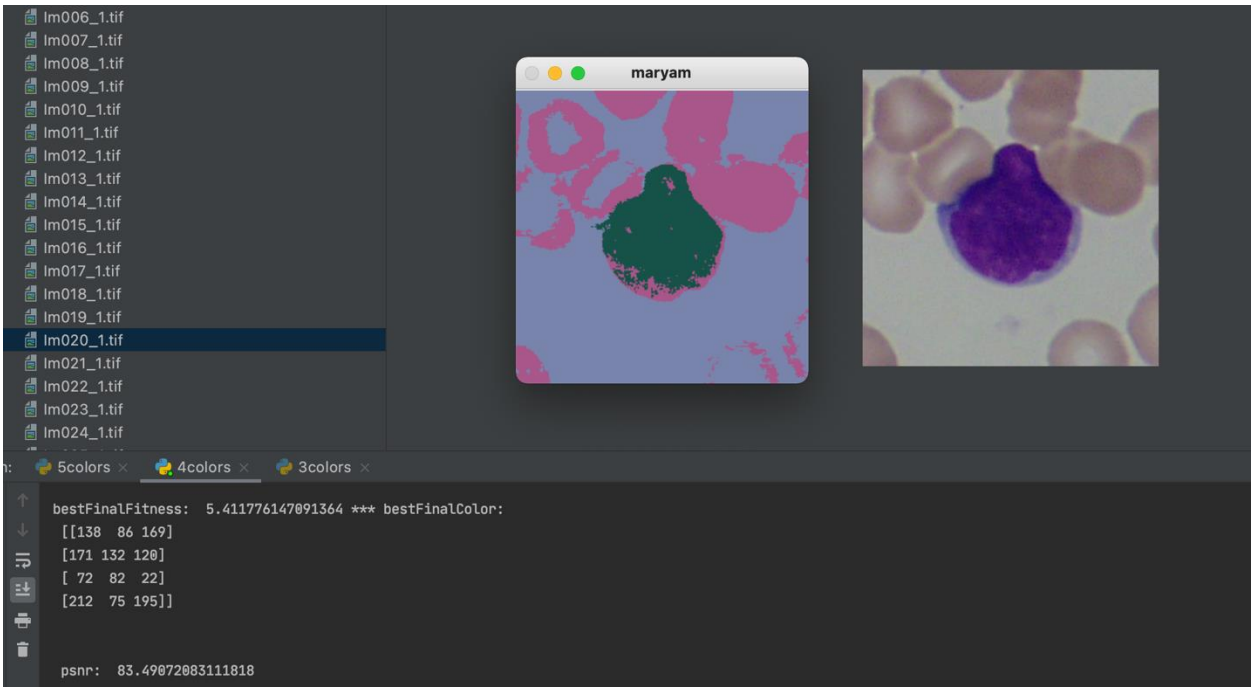
Fitness = 9.18

PSNR = 84.07



Fitness = 11.34

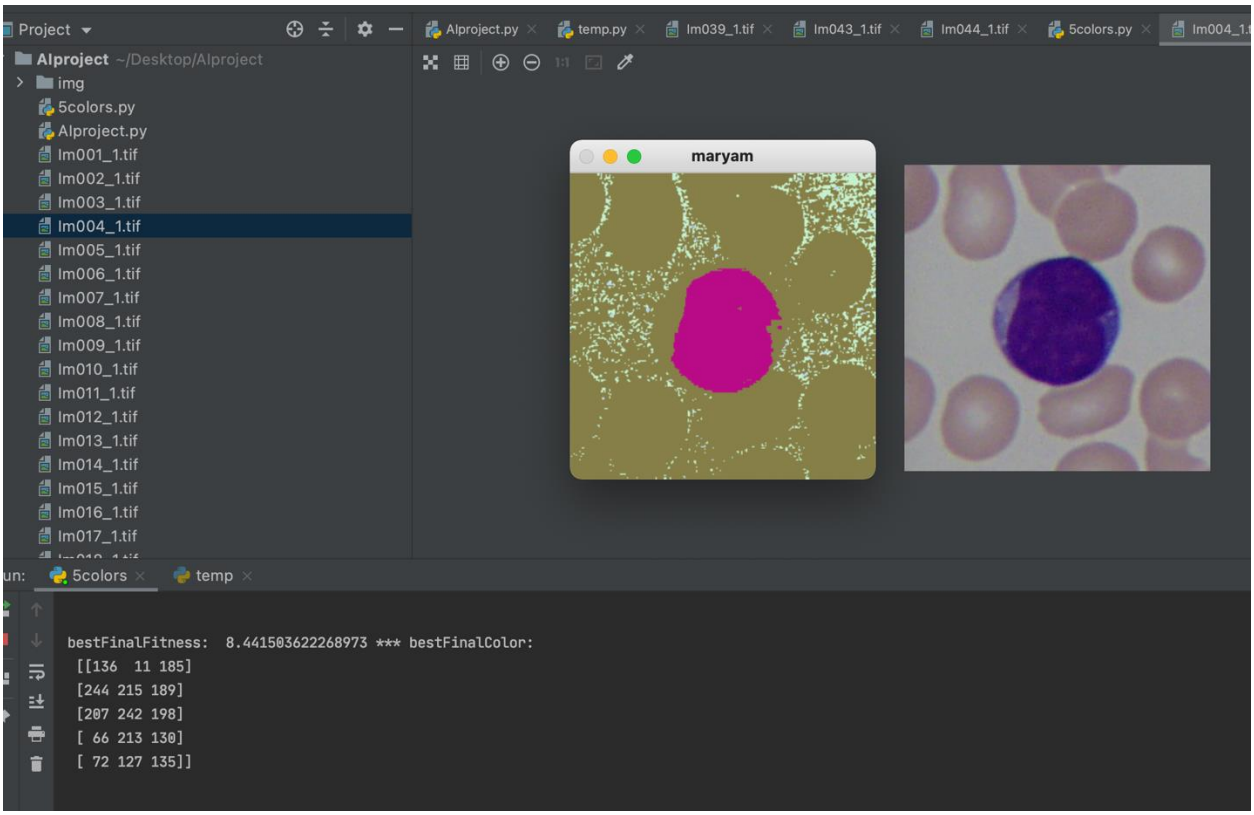
PSNR = 84.65



Fitness = 5.411

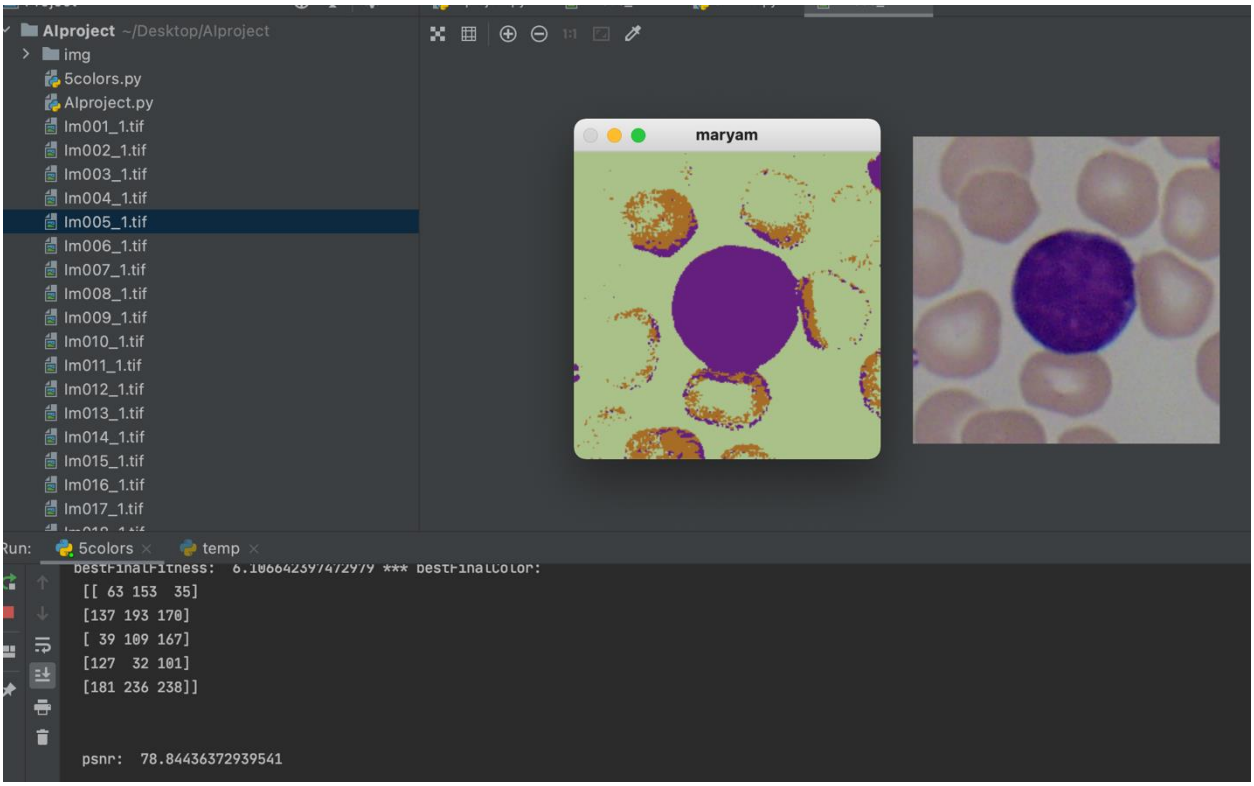
PSNR = 83.49

نتایج برای حالت ۵ رنگ :

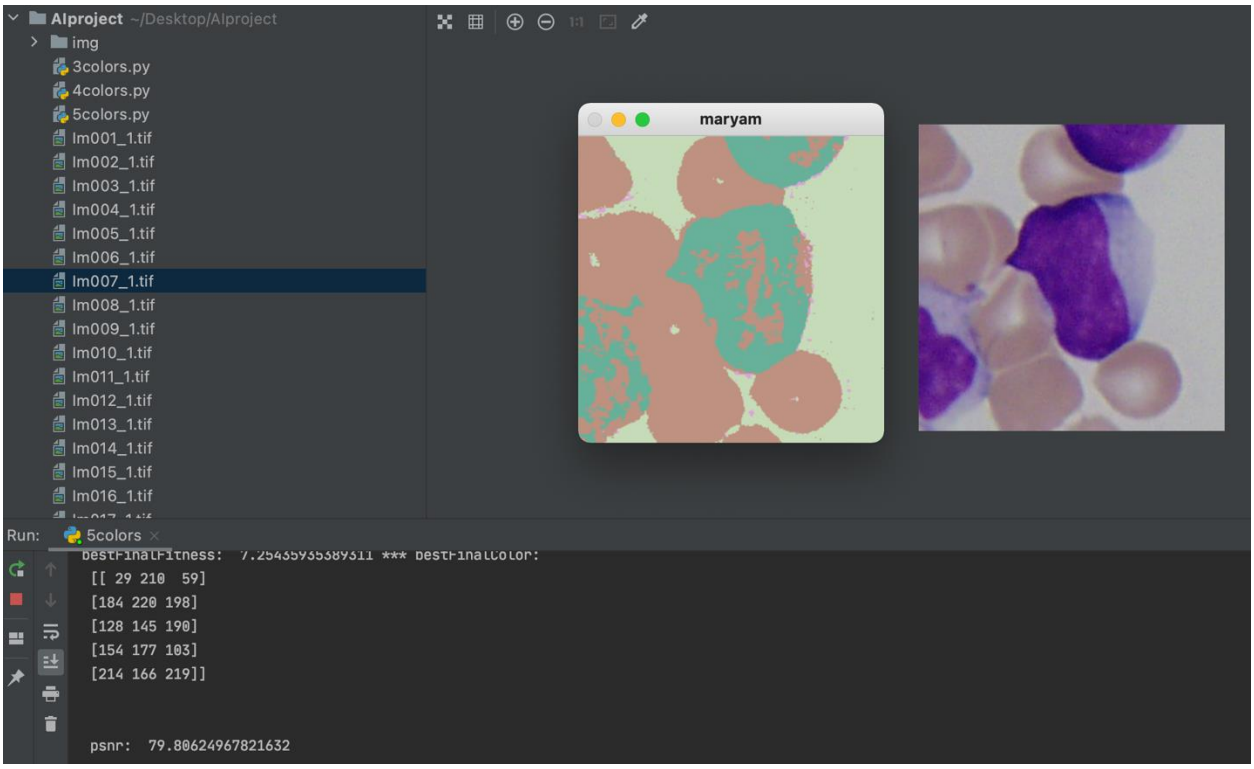


Fitness = 8.44

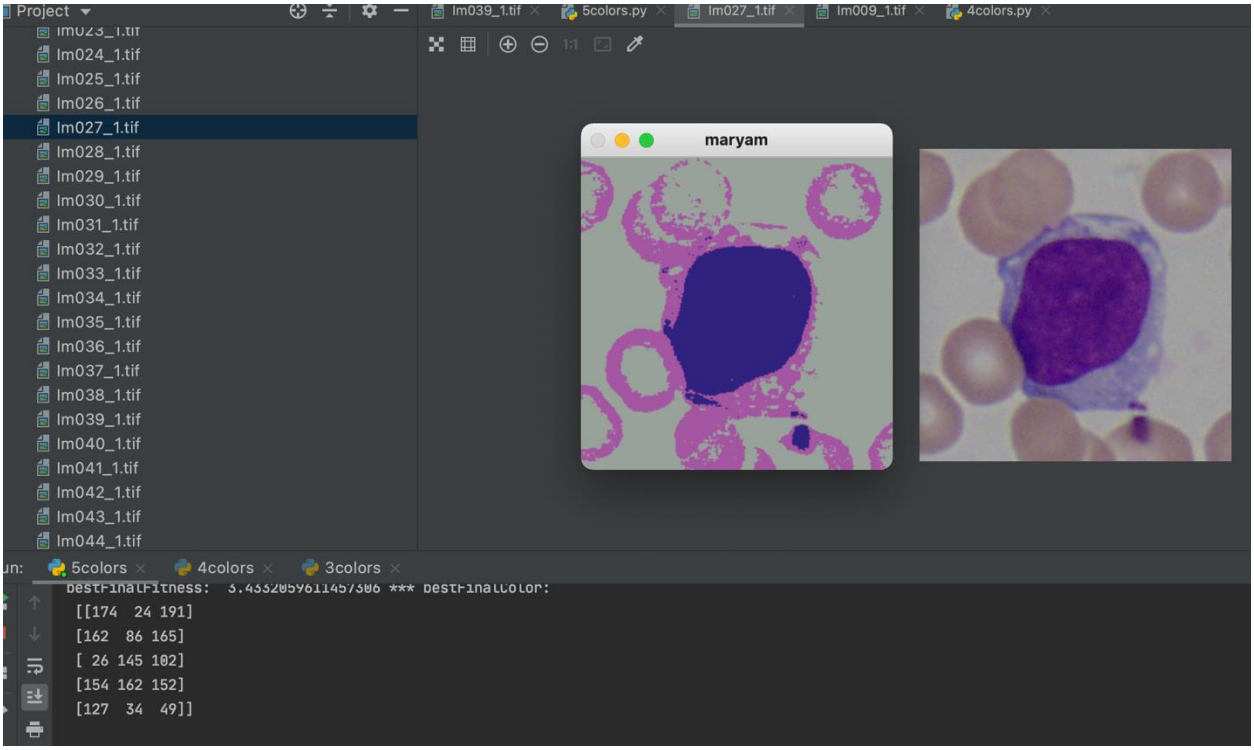
PSNR = 84.74



Fitness = 6.10
PSNR = 78.84

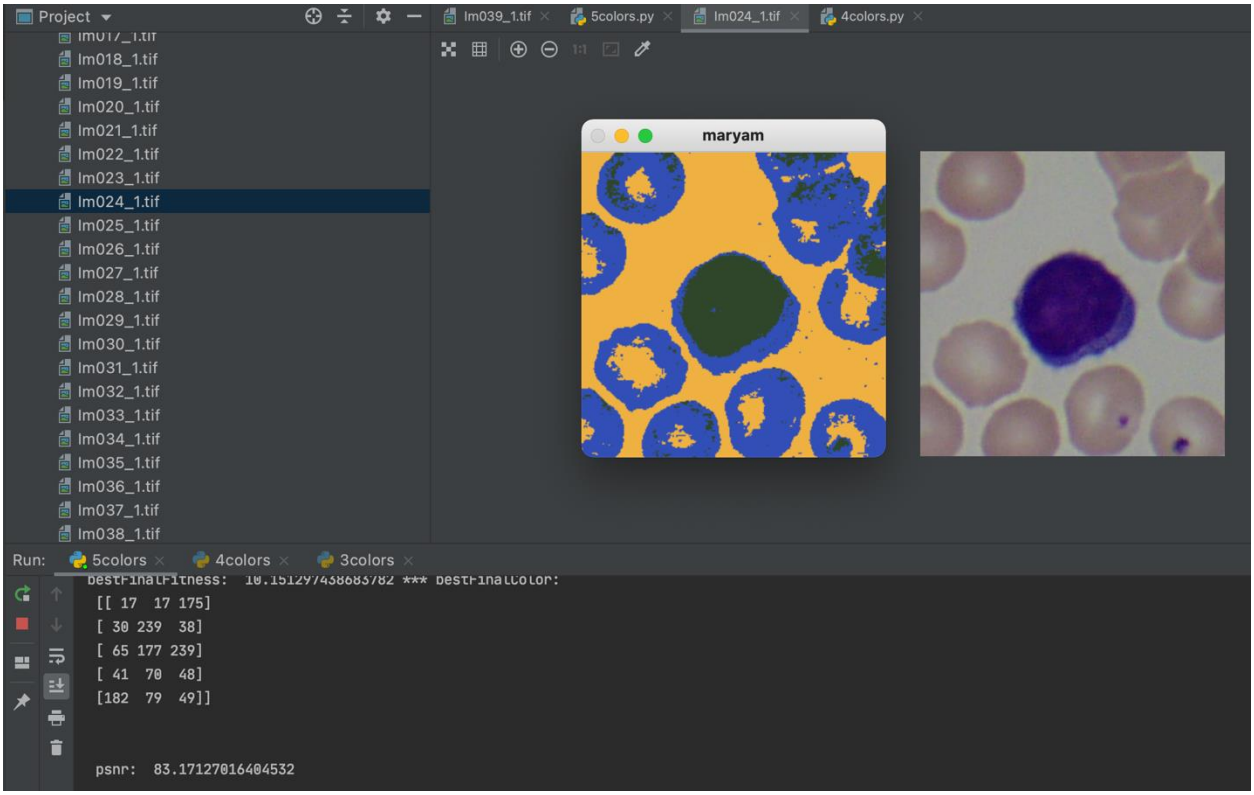


Fitness = 7.25
PSNR = 79.80



Fitness = 3.43

PSNR = 85.66



Fitness = 10.15
PSNR = 83.17

Total mean fitness = 7.42
Total mean PSNR = 84.13