Wykrywanie naczyń siatkówki oka

Aleksandra Jarzyńska 136722
Grzegorz Bryk 136686

Spis treści

1 Zastosowany język programowania oraz dodatkowe biblioteki

Zastosowaliśmy język **Python** w wersji 3.6 (lub wyższej).

1.1 Biblioteki

- OpenCV – do przetwarzania obrazu
- matplotlib – do wyświetlania wyników przetwarzania w formie graficznej
- multiprocessing – do zrównoleglania obliczeń na CPU
- joblib – do serializacji wyników, modeli, scalerów
- scikit-learn – do tworzenia, uczenia i wykorzystania modelu sieci neuronowej, scalerów
- numpy – do operacji matematycznych oraz operacji na macierzach
- itertools – do łączenia tablic bez powielania ich w pamięci

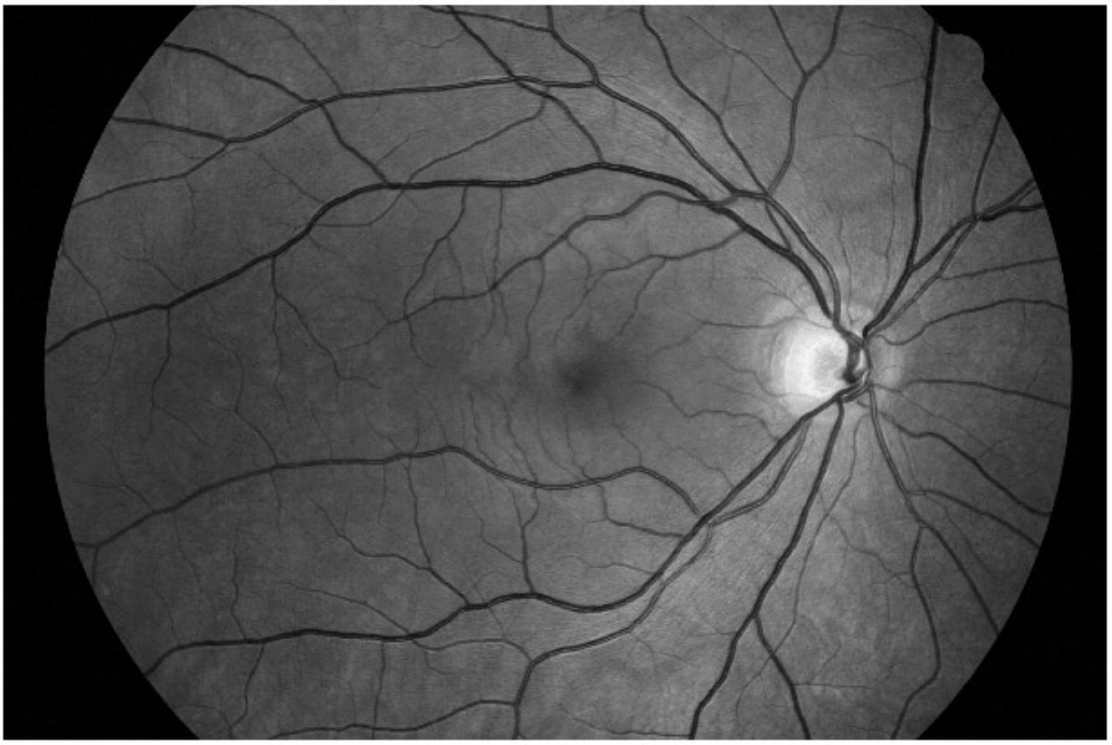
2 Opis zastosowanych metod

2.1 Przetwarzanie obrazów

2.1.1 Przetwarzanie przedwstępne

```
@staticmethod
def _basic_processing(image):
    image = (image[::, ::, 1] * 2.3) - (image[::, ::, 0] * 1.7)
    image = image * 255 / image.max()
    image = image.clip(0, 255)
    image = image.astype('uint8')
    return image
```

najpierw obraz jest rzutowany do jednego kanału wg wyznaczonego eksperymentalnie w GIMPie wzoru: 2,3G – 1,7R, następnie wynik jest normalizowany do poziomu max=255, obcinany do zakresu 0-255, aby zniwelować ułamki powyżej 255, i rzutowany do typu uint8. W wyniku powstaje następujący obraz:



Ilustracja 1: Obraz przetworzony przedwstępnie

2.1.2 Przetwarzanie wstępne do podejścia klasycznego

```
kernel = np.ones((6, 6), np.uint8)
image = cv.erode(image, kernel)
kernel = np.ones((3, 3), np.uint8)
image = cv.dilate(image, kernel)

image = cv.bilateralFilter(image, 5, 210, 30)
image = cv.medianBlur(image, 9)
```

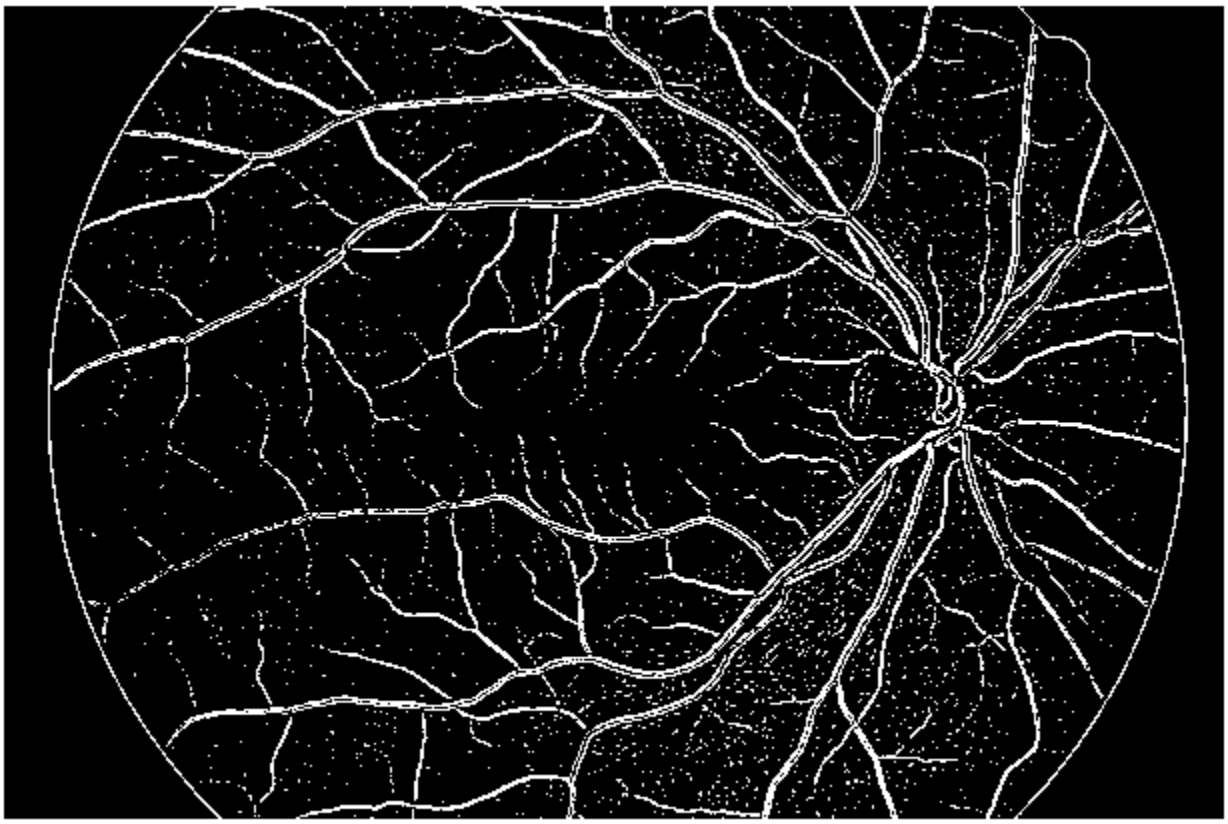
stosowane są kolejno: erozja, dylatacja, filtrowanie Bilinearne (aby uwydatnić krawędzie i pozbyć się szumów) oraz rozmycie medianowe dla wygładzenia i usunięcia najdrobniejszych szumów. W wyniku przykładowy obraz wygląda nastepująco:

Ilustracja 2: Obraz przetoworzony wstępnie

2.1.3 Przetwarzanie właściwe

```
image = cv.adaptiveThreshold(image, 255, cv.ADAPTIVE_THRESH_GAUSSIAN_C,
cv.THRESH_BINARY_INV, 31, 3)
```

Przetworzenie obrazu do maski binarnej następuje przez użycie progowania adaptacyjnego Gaussowskiego, z wynikiem odwrotnym (gdyż na wejściu naczynia krwionośne są ciemniejsze niż reszta siatkówki). Przykładowy wynik progowania:

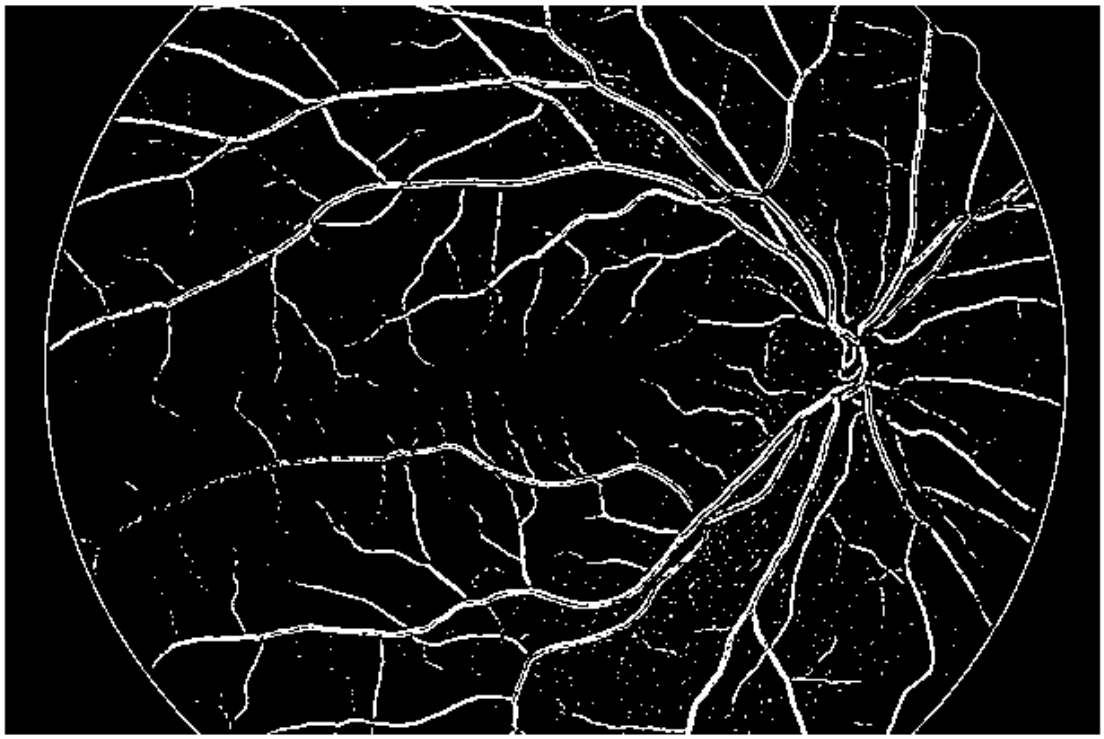


Ilustracja 3: Maska wygenerowana podejściem klasycznym

2.1.4 Przetwarzanie finalne

```
image = cv.bilateralFilter(image, 5, 70, 10)
image = cv.medianBlur(image, 9)
```

ponownie aplikowany jest filtr bilateralny oraz równanie medianowe. Po wyrównaniu otrzymujemy maskę końcową:

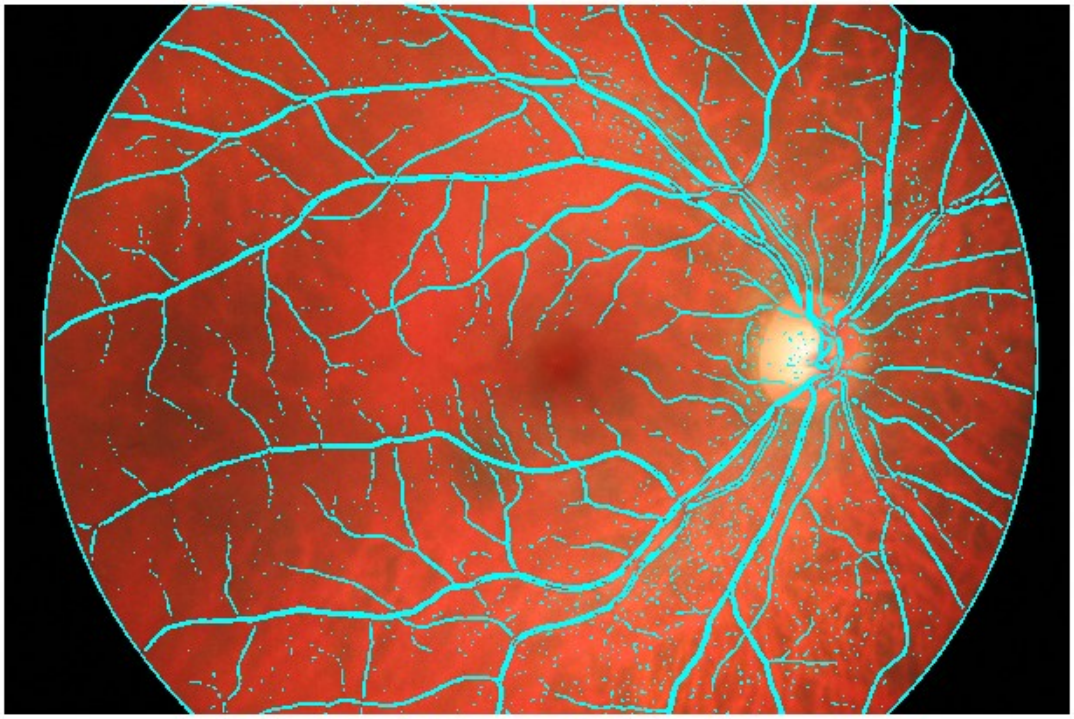


Ilustracja 4: Przefiltrowana maska otrzymana podejściem klasycznym

2.1.5 Nałożenie maski na obraz źródłowy

```
im = np.array(orig_image)
im[image > 50] = [0, 255, 255]
```

Wynik:



Ilustracja 5: Obraz dna siatkówki z nałożoną maską otrzymaną podejściem klasycznym

2.2 Uczenie maszynowe

Podejście z uczenie maszynowym opiera się na podziale obrazu na fragmenty o rozmiarach 5x5px. Jako klasyfikator wybraliśmy sieć neuronową MLPClassifier z pakietu scikit-learn.
Przy wczytywaniu obrazy są skalowane do stałego rozmiaru 700x700px, co zapewnia w miarę szybkie i w miarę dokładne przetwarzanie.

2.2.1 Nauka

Aby nauczyć sieć, tworzymy dataset składający się z wektorów, których elementami są wartości pikseli z danego fragmentu (25 skalarów), przymioty całości zbioru (średnia, minimum, maksimum, mediana) oraz wyliczone z całości obrazka momenty Hu. Daje to łącznie 36 skalarów na wejściu klasyfikatora. Każdy wektor podawany jest w parze z etykietą – True, jeśli środkowy piksel fragmentu wg maski eksperckiej należy do naczyń krwionośnych, False w przeciwnym przypadku. Przed podaniem na wejście algorytmu uczącego sieć (mechanizmem propagacji wstecznej), dane muszą zostać przeskalowane. Powodem jest wrażliwość sieci na różnice rzędu wielkości między poszczególnymi parametrami wejścia. W tym celu wykorzystujemy Scaler z pakietu scikit-learn, który automatycznie dopasowuje się do danych, a następnie zapisujemy go w wersji zserializowanej obok modelu, aby dane podane do sieci w procesie przewidywania mogły być tak samo przeskalowane.

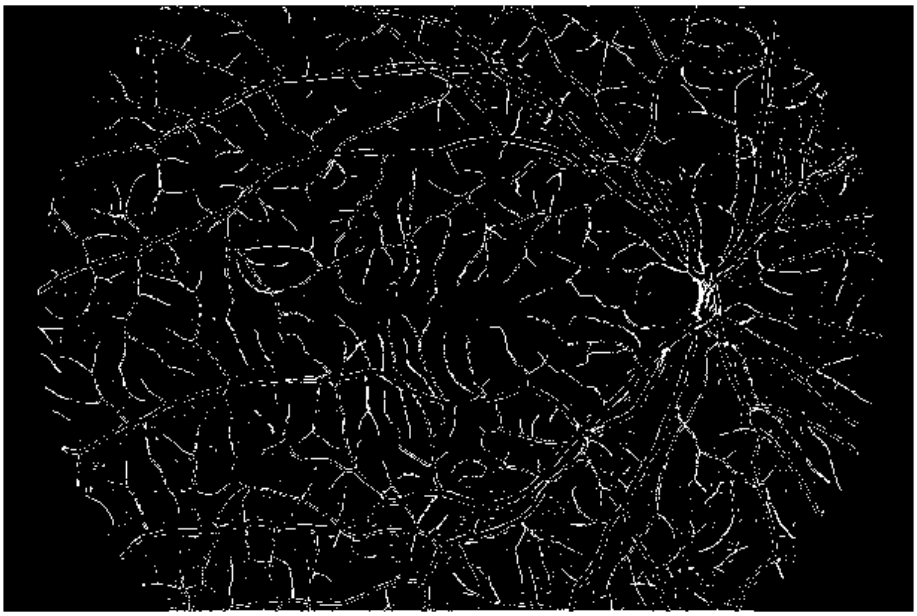
Uczenie przeprowadziliśmy na danych pochodzących z datasetu HRF. Ponieważ w podziale 15 obrazków na fragmenty 5x5px około 90% fragmentów należało do klasy negatywnej (fragmenty siatkówki poza naczyniami krwionośnymi), dataset utworzony w ten sposób określamy jako wysoce niezbalansowany. Jednak dane na wejściu sieci w trakcie rozpoznawania również nie będą zbalansowane, więc postanowiliśmy sprawdzić jak sieć zachowa się przy uczeniu na różnych poziomach zbalansowania. W tym celu użyliśmy downsaplingu na klasie negatywnej. Wykonaliśmy uczenie na zbiorach na poziomach zbalansowania od 1:1 do 1:5.

Sama sieć ma 3 warstwy ukryte, o liczności 15, 7, 2. Funkcja aktywacji to relu, a solver – adam.

Ilustracja 6: Obraz dna oka z nałożoną maską otrzymaną przy użyciu sieci neuronowej

2.2.2 Miara jakości

Aby porównać między sobą różne modele sieci (np. wytrenowane przy użyciu różnie zbalansowanych zbiorów uczących), a także w dalszej kolejności porównać wyniki sieci neuronowej z wynikami prostego przetwarzania obrazu należało obrać miarę jakości rozwiązania. U nas za tę miarę przyjęliśmy posumowaną wartość funkcji cv.absdiff między naszą maską, a maską ekspercką.



Ilustracja 7: Zobrazowanie różnicy między maską otrzymaną siecią neuronową a maską ekspercką

2.2.3 Eksperymentalne wyznaczenie najlepszego poziomu zbalansowania datasetu

Aby dowiedzieć się jaki poziom zbalansowania daje najlepsze wyniki, wytrenowaliśmy sieć na kazdym z poziomów zbalansownia od 1:1 do 1:7 i każdą sieć przetestowaliśmy na dwóch obrazach spoza zbioru uczącego. Oto zapis przebiegu eksperymentu:

Results negative: 2204223
Results positive: 217857
Desired negative sample: 217857
0.865301860160885
Balance 1:1 Try 1 absdiff: 23135130
Balance 1:1 Try 2 absdiff: 24967305

Results negative: 2204223
Results positive: 217857
Desired negative sample: 435714
0.8858967983781509
Balance 1:2 Try 1 absdiff: 14866245
Balance 1:2 Try 2 absdiff: 20954370

Results negative: 2204223
Results positive: 217857
Desired negative sample: 653571
0.9013001618030134
Balance 1:3 Try 1 absdiff: 14549280
Balance 1:3 Try 2 absdiff: 17806395

Results negative: 2204223
Results positive: 217857
Desired negative sample: 871428
0.9167114207943743
Balance 1:4 Try 1 absdiff: 13145505
Balance 1:4 Try 2 absdiff: 17682210

Results negative: 2204223
Results positive: 217857
0.9246755333187978
Balance 1:5 Try 1 absdiff: 13747305
Balance 1:5 Try 2 absdiff: 18852915

Results negative: 2204223
Results positive: 217857
Desired negative sample: 1307142
0.9330688524590164
Balance 1:6 Try 1 absdiff: 13139640
Balance 1:6 Try 2 absdiff: 18353370

Results negative: 2204223
Results positive: 217857
Desired negative sample: 1524999
0.938853378928887
Balance 1:7 Try 1 absdiff: 13447935
Balance 1:7 Try 2 absdiff: 17933385

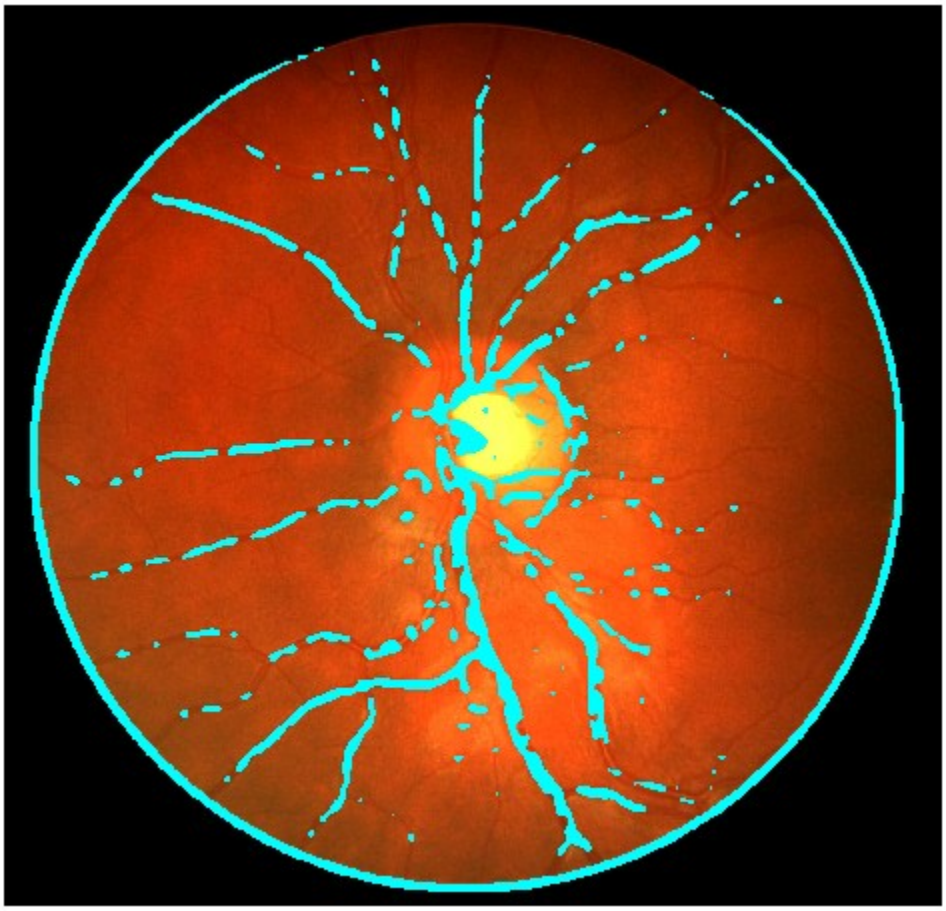
Z eksperymentu wynikło, że najbardziej zbliża się do maski eksperckiej sieć trenowana na zbiorze zbalansowanym w proporcjach 1:4. Taką sieć wykorzystaliśmy.

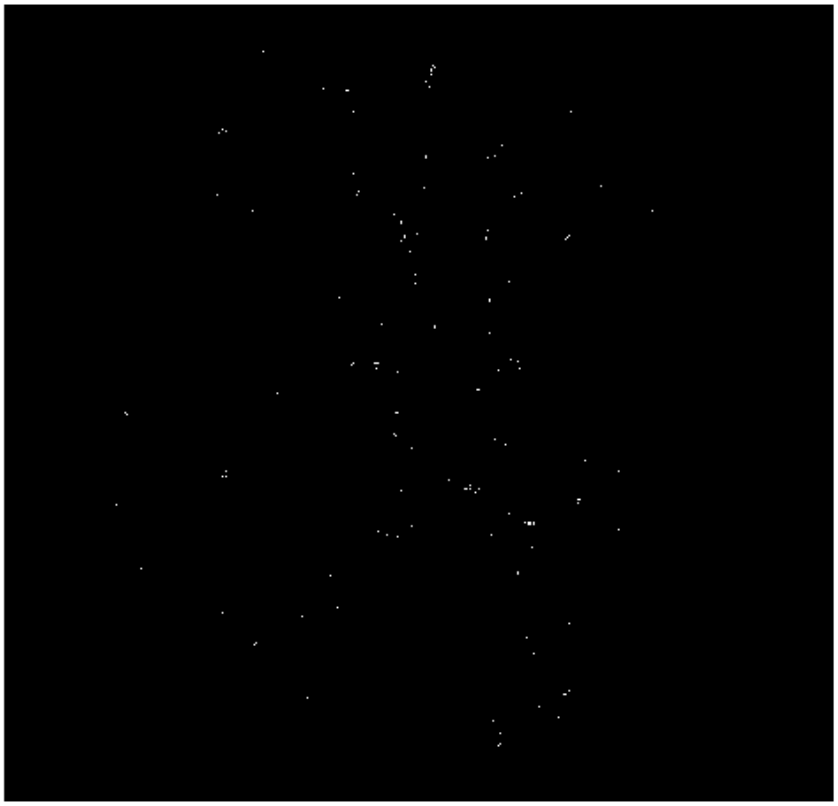
3 Wizualizacja wyników działania algorytmów

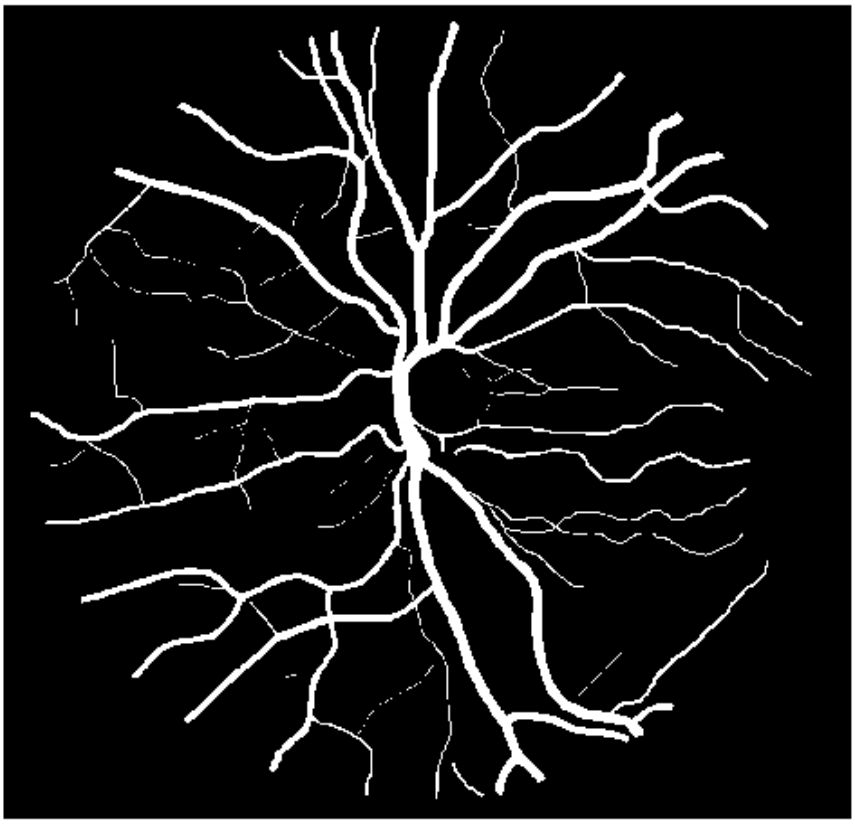
W każdym punkcie obrazki będą dodane w zadanej kolejności:

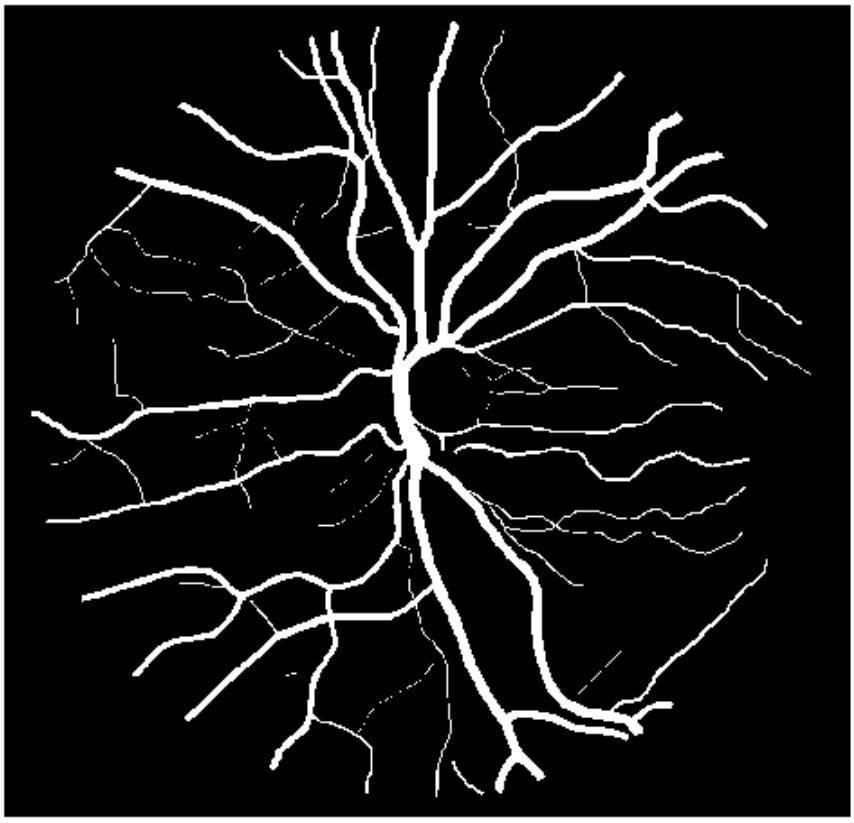
1. wynik działania algorytmu klasycznego,
2. wynik działania algorytmu ML,
3. wizualizacja różnicy między wynikami algorytmów,
4. wizualizacja różnicy między wynikiem algorytmu klasycznego a maską ekspercką,
5. wizualizacja różnicy między wynikiem algorytmu ML a maską ekspercką.

3.1

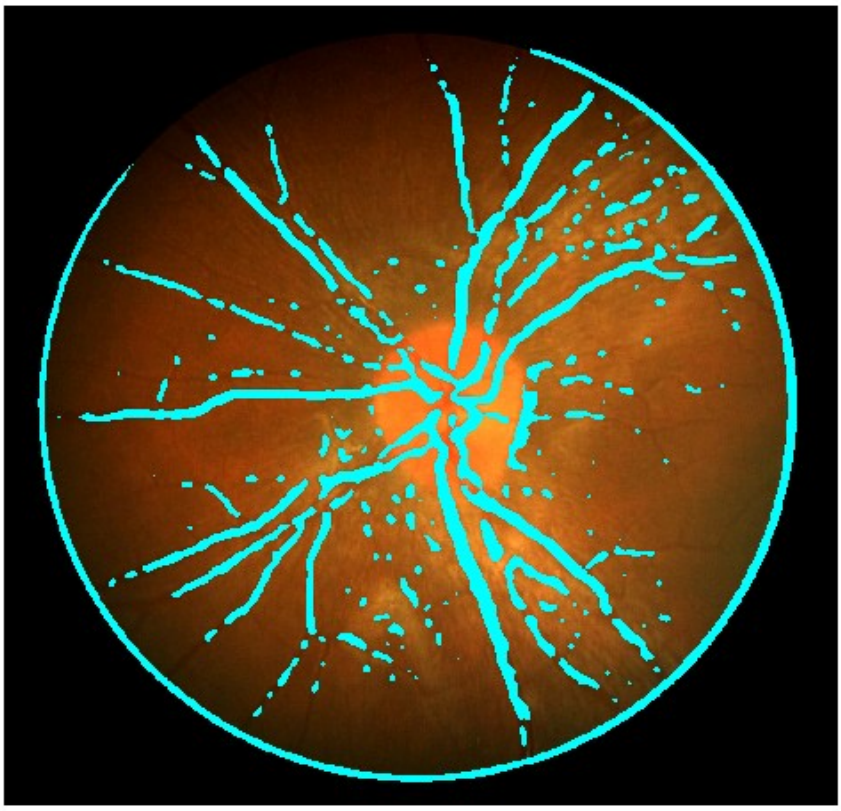


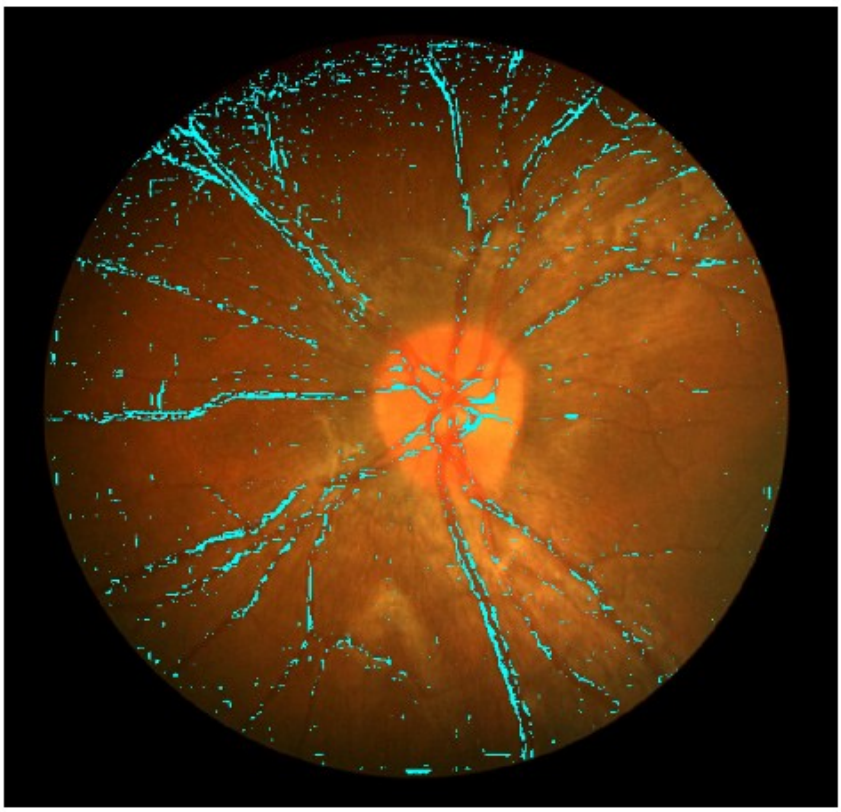


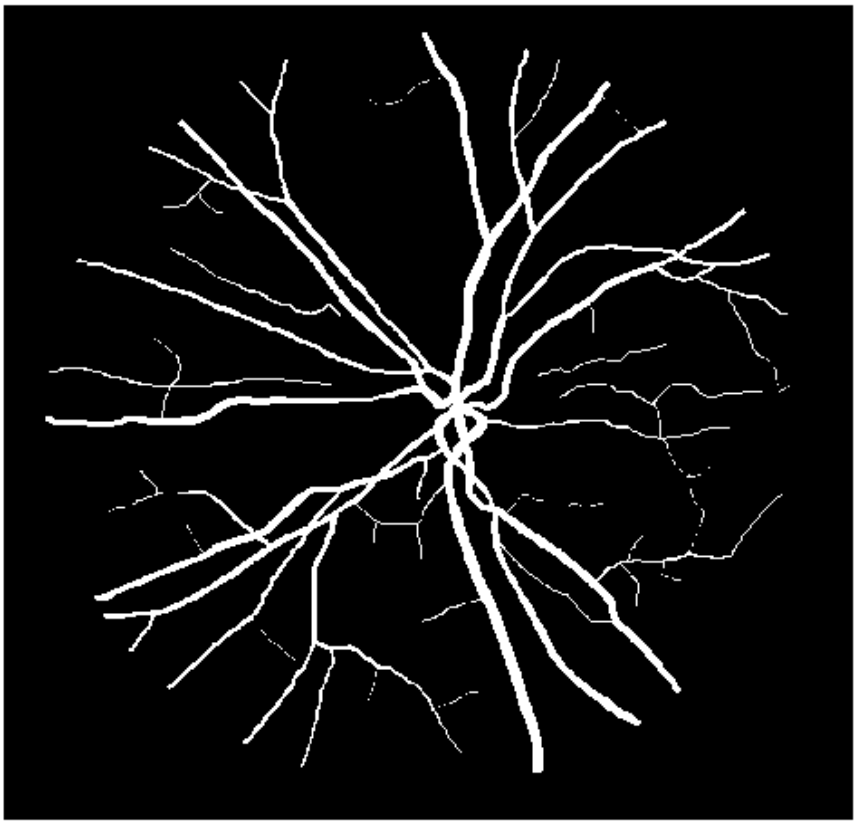


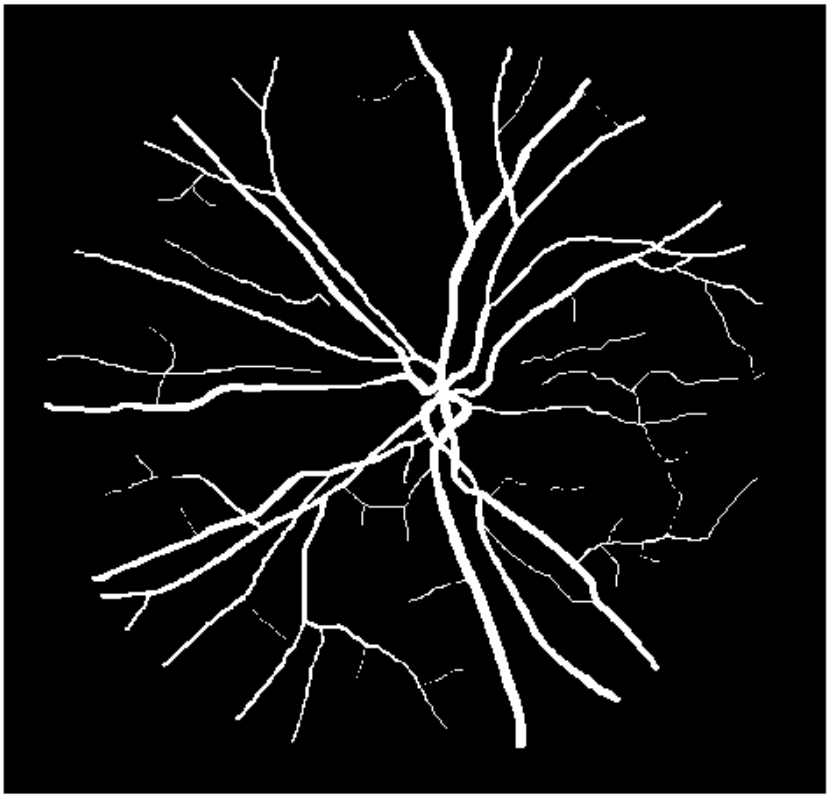


3.2

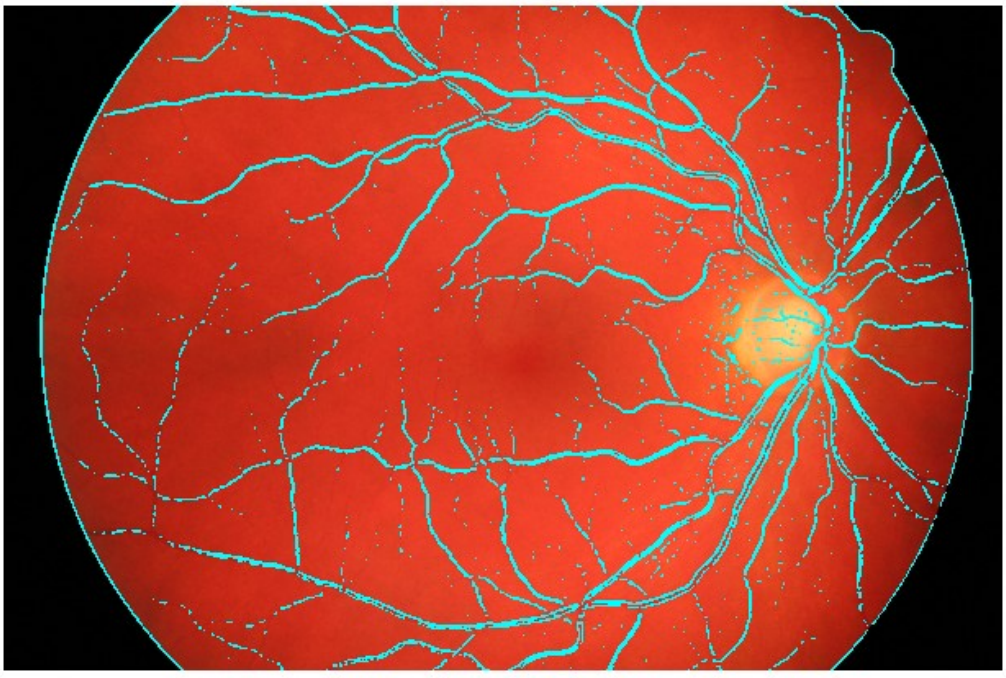


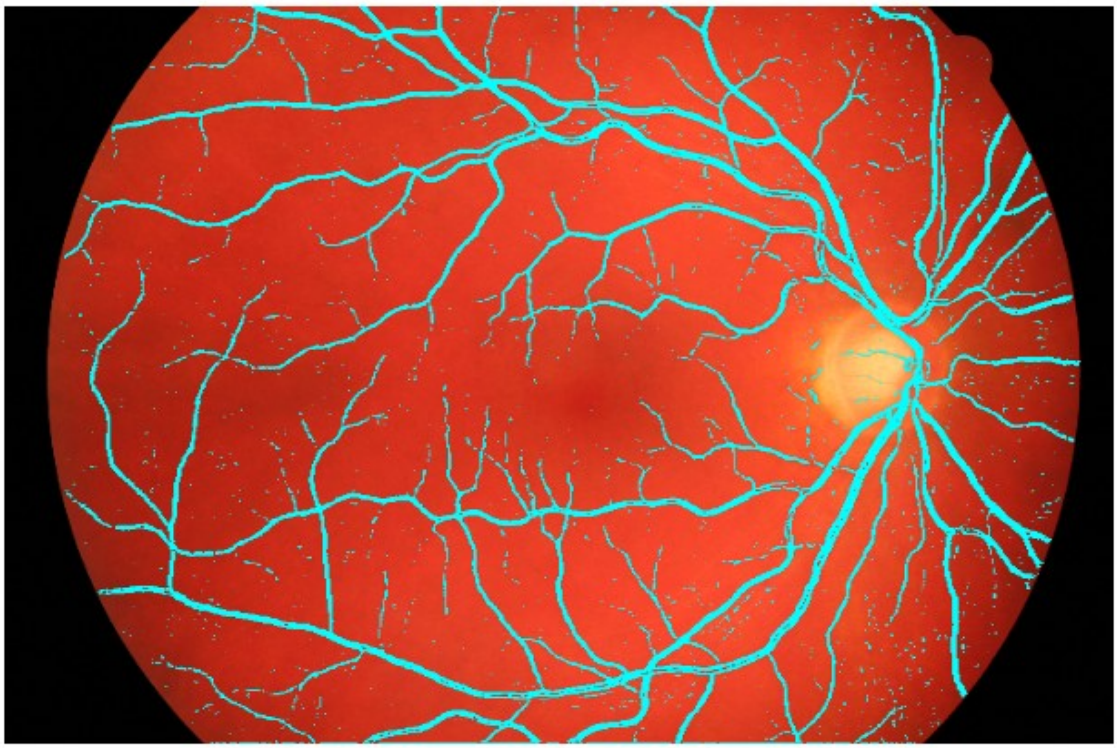


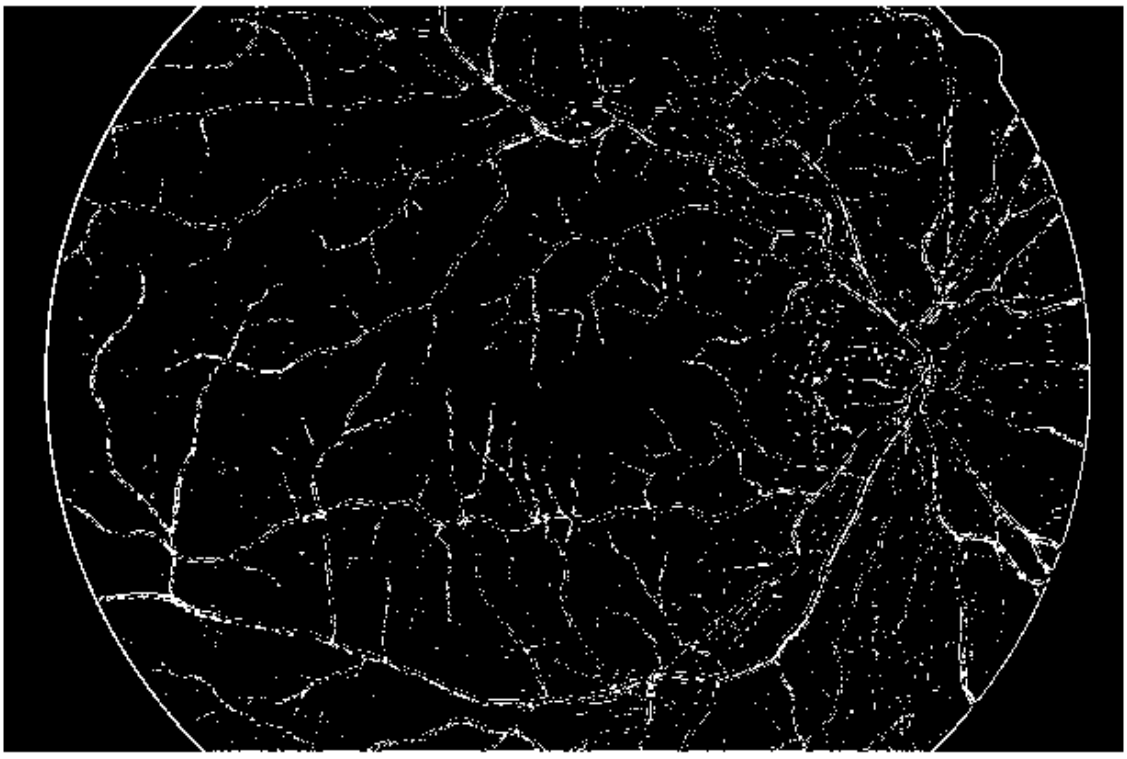


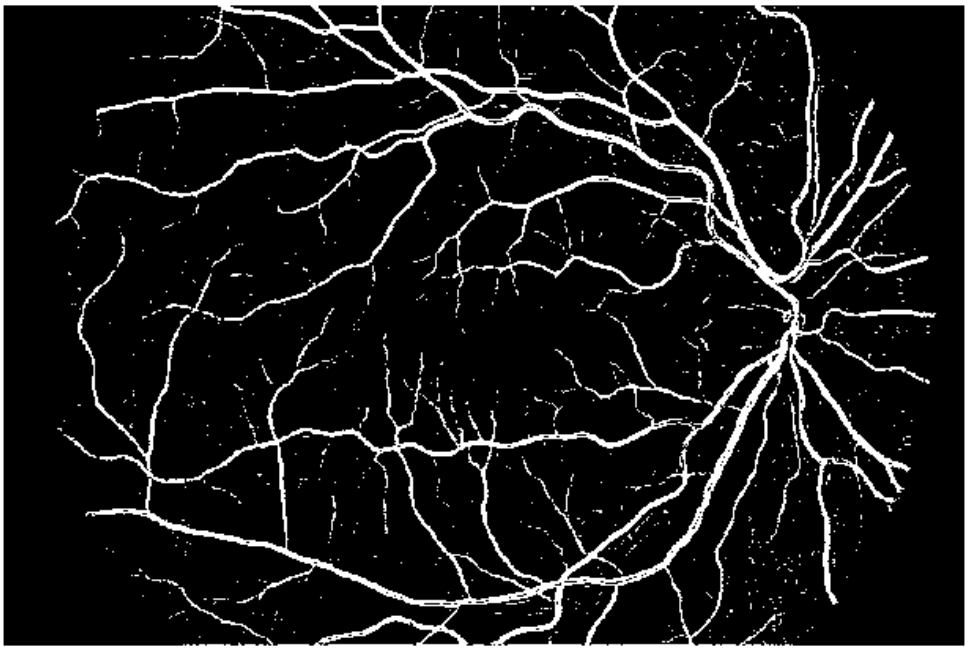


3.3

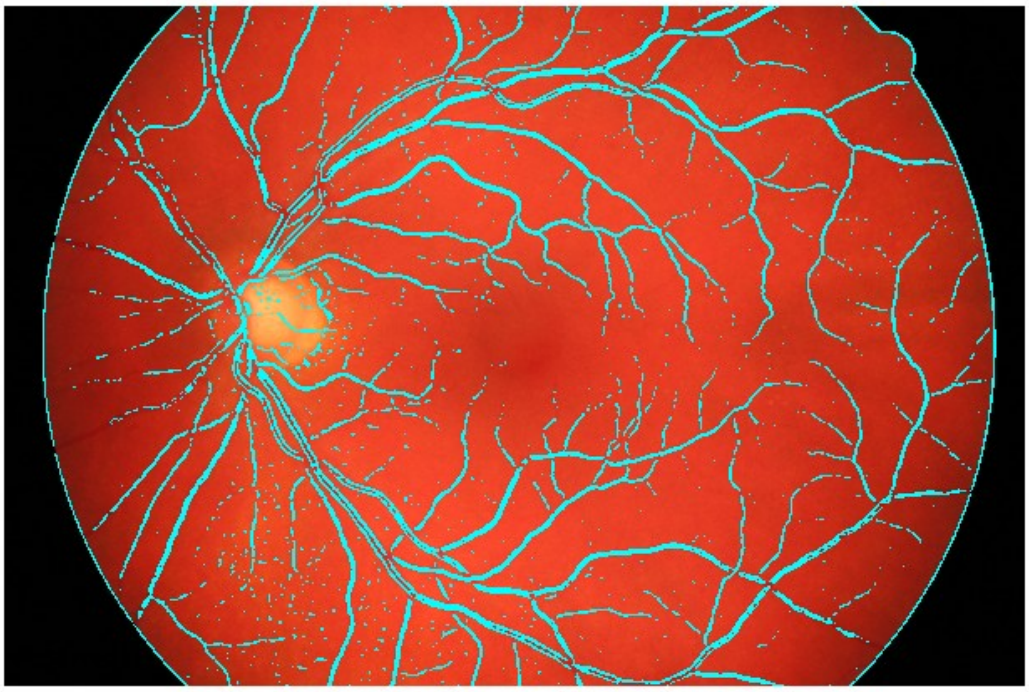


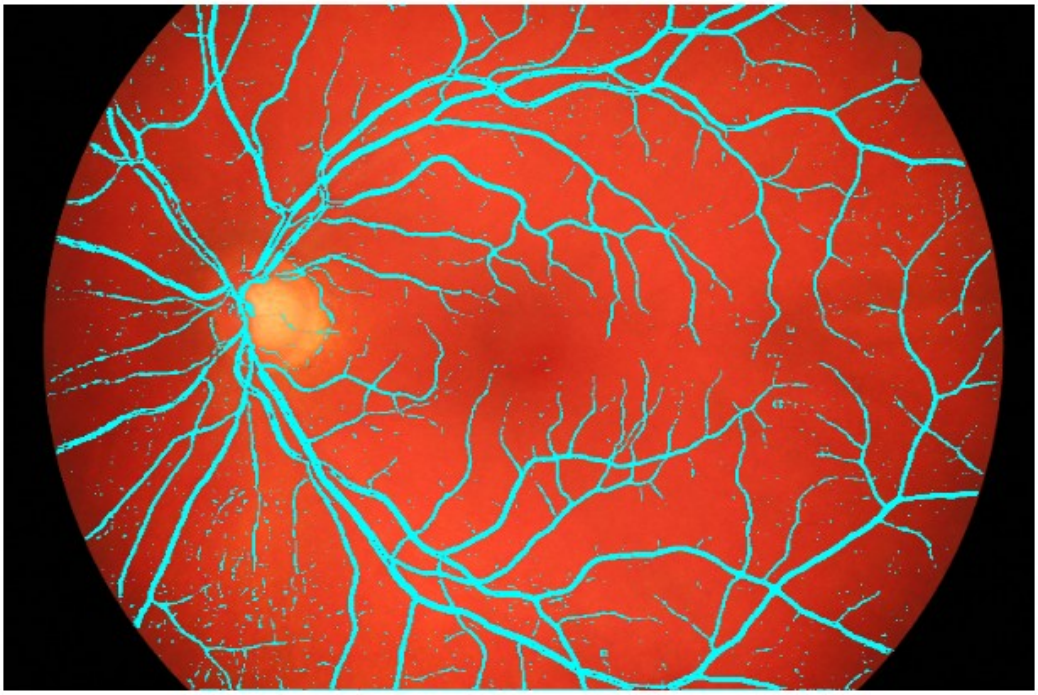


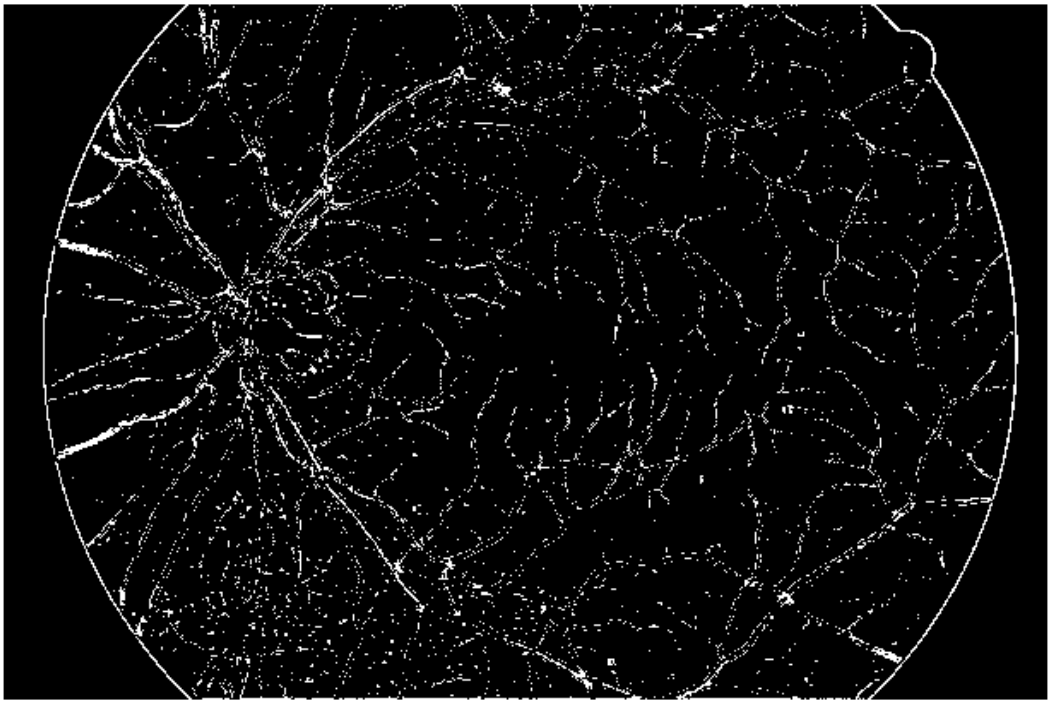


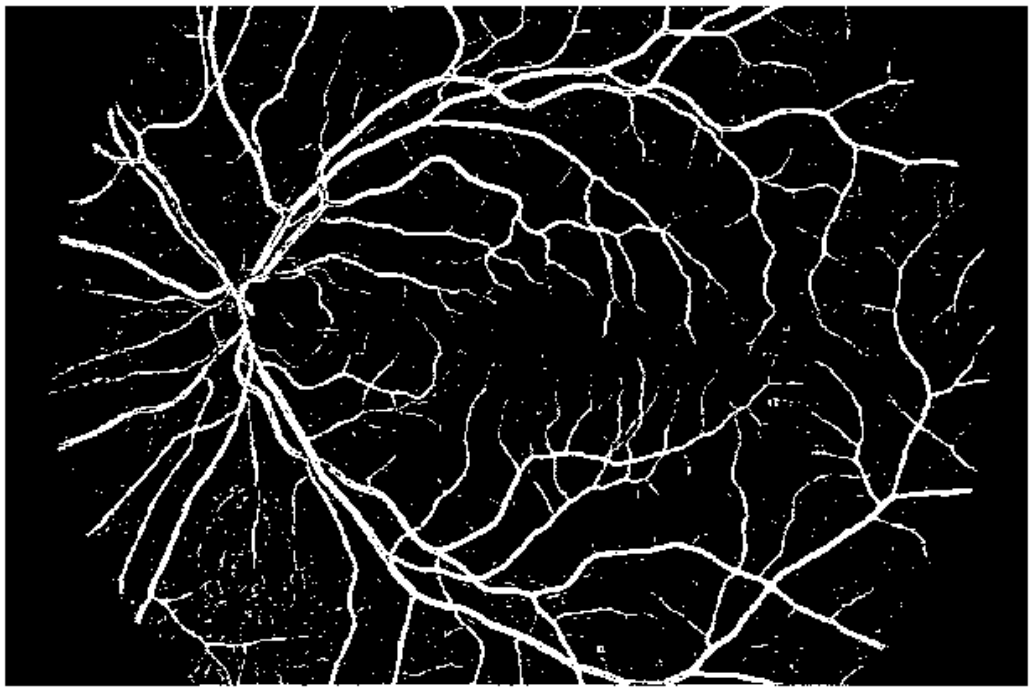


3.4

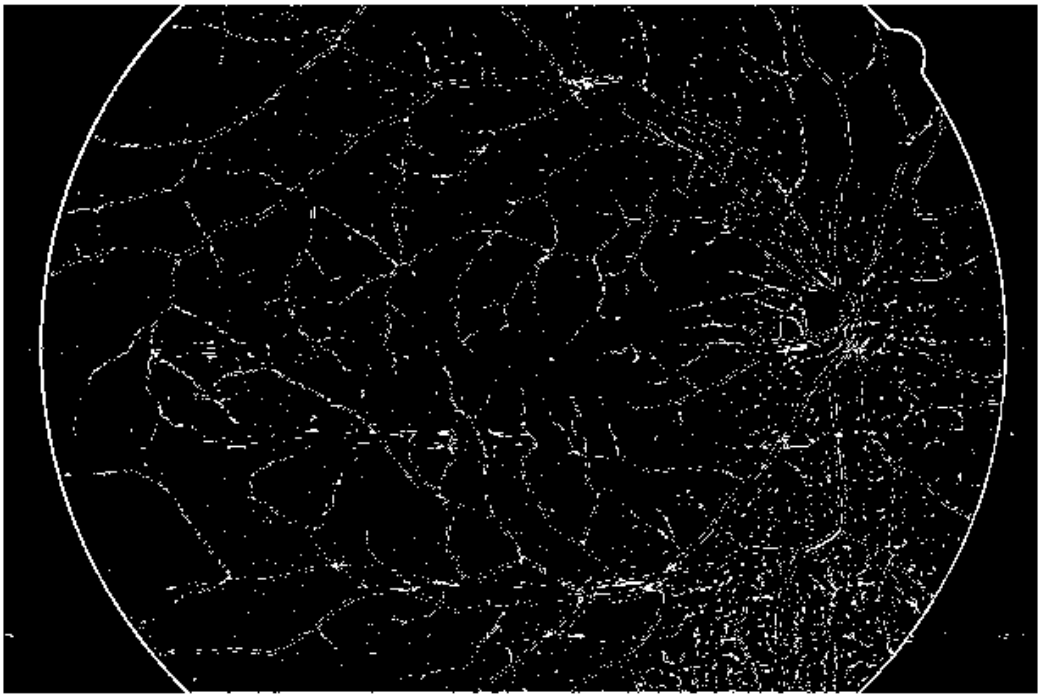


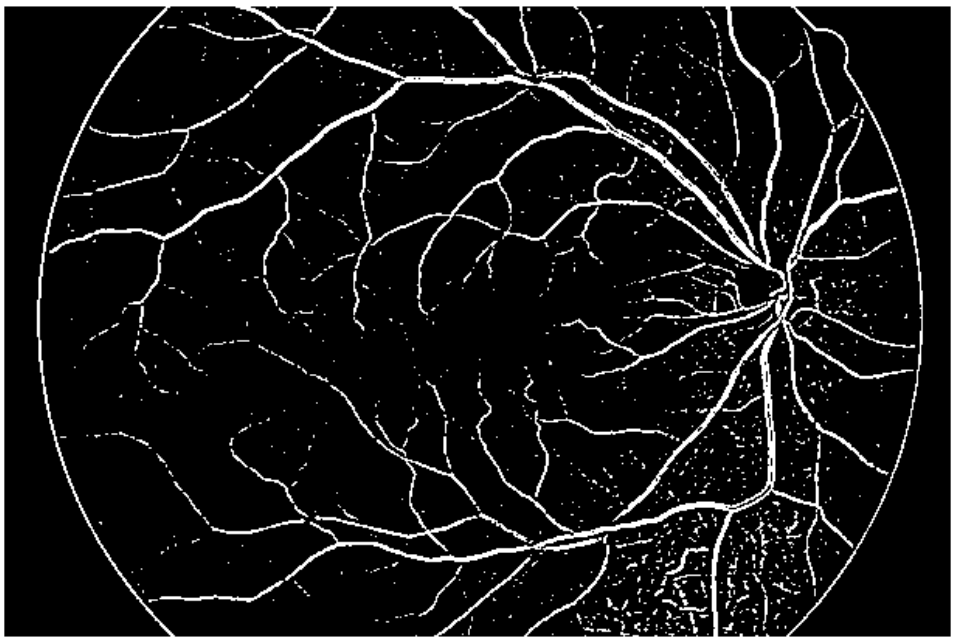






3.5





4 Podsumowanie

Porównanie wyników wzkazuje, że mniejsze różnice względem maski eksperckiej uzyskujemy metodą klasyczną. Jednak sieć neuronowa daje lepsze wyniki w naszych, ludzkich oczach. Występuje mniejsze zaszumienie, metoda jest mniej podatna na różnice w jasności miejscowej obrazu. Mimo, że progowanie adaptacyjne bierze pod uwagę niejednolitą jasność, nie radzi sobie tak dobrze, jak sieć, która interpretuje wyniki po normalizacji skalerem. Widać to po zaznaczonych końcówkach naczyń, na brzegach siatkówki, których zaznaczenia często brakuje w wynikach algorytmu klasycznego.

Ogólnie wyniki działania obu metod nie są idealne, odstępstwa od maski eksperckiej są wyraźne, szczególnie w obrębie drobniejszych naczynek. Prawdopodobnie sytuację można by dalej poprawić, stosując inne, specjalizowane rodzaje sieci neuronowych. MLPClassifer z pakietu scikit-learn jest prostym modelem sieci, dobrym to prototypowania, jednak działa stosunkowo powoli i trudno jest wpłynąć na jego zachowanie. Nie byliśmy tego świadomi przed rozpoczęciem prac, ale teraz wybralibyśmy inną bibliotekę do realizacji algorytmu ML.