Naczynia dna siatkówki oka

Projekt 2. Informatyka w Medycynie

Szymon Rozynek 136793
Marcin Pałasz 136778

W projekcie stworzyliśmy aplikację do automatycznego wykrywania naczyń krwionośnych dna siatkówki oka.
Do realizacji wykorzystaliśmy język programowania python oraz dodatkowe biblioteki:
-scipy
-numpy
-matplotlib
-imageio
-skimage
-math
-ipywidgets
-os
-IPython
-sklearn
-cv2

Opis zastosowanych metod:
a. przetwarzanie obrazów
i.
1. w pierwszym kroku tworzymy obraz na podstawie koloru zielonego w oryginalnym obrazku

```
def img2gray(image):
    x=image.shape[0]
    y=image.shape[1]
    img = np.zeros( (x, y), dtype='uint8')
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            if(image[i,j,1]<40):
                img[i,j]=0
            elif(sum(image[i,j])<100):
                img[i,j]=0
            else:
                img[i,j]=255-np.clip(image[i,j,1],0,140)
    return img
```

2. w drugim kroku tworzymy obraz na podstawie koloru zielonego z ograniczoną wartością maksymalną wykorzystując oryginalny obraz

```
def img2gray_1(image):
    x=image.shape[0]
    y=image.shape[1]
    img = np.zeros( (x, y), dtype='uint8')
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
          img[i,j]=np.clip(image[i,j,1],0,100)+np.sqrt(np.clip(image[i,j,1]-100,0,155))
    return img
```

3. w trzecim kroku porównaliśmy obrazy używając funkcją

```
def comp(image, image1): # comp(krok 1., krok 2.)
    x=image.shape[0]
    y=image.shape[1]
    img = np.zeros( (x, y), dtype='uint8')
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            if(image[i,j])<80:
                img[i,j]=0
            else:
                img[i,j]=np.clip(image[i,j]-image1[i,j], 40, 255)
    return img
```

4. w czwartym kroku tworzymy obraz wynikowy nakładając filtry

```
mp.dilation(sobel(sobel(sobel(feature.canny(krok3, sigma=1)))))
```

5. następnie szukamy krawędzi oka i usuwamy ją z wyniku tworząc gotową maskę

```
edge=sobel(feature.canny(img2gray(img), sigma=6))
edge=mp.dilation(edge,square(7))
mask= comp_2(krok4,edge)

#gdzie funkcja comp_2 wygląda następująco
def comp_2(image, image1):
    x=image.shape[0]
    y=image.shape[1]
    img = np.zeros( (x, y), dtype='uint8')
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            if image[i,j]-image1[i,j]>0.85*image[i,j]:
                img[i,j]=255
            else:
                img[i,j]=0
    return img
```

ii. wykorzystujac operator Sobela jesteśmy w stanie dosyć dokładnie wychwycić krawędze naczyń krwionośnych na zdjęciu, po wstępnym jego przygotowaniu

b. uczenie maszynowe

i. Do nauczenia klasyfikatora dane są przygotowywane w następujący sposób:

-obraz jest konwertowany na skalę szarości
-normalizacja pikseli (zamiast 255 maksymalna wartość piksela to 1)
-zwiększenie kontrastu
-wykrycie krawędzi operatorem Sobela

ii. Obraz dzielony jest na fragmenty o ustalonym przez nas rozmiarze 7 pikseli, z odstępem 5 pikseli (fragmenty nakładają się). Fragmenty są przeliczane na wektor składowych pikseli oraz momentów Hu. Środkowy piksel fragmentu jest etykietowany poprzez maskę ekspercką. Następnie wykorzystujemy downsampling aby ilość pikseli z etykietą “0” (brak naczynia krwionośnego) była równa ilości tych z etykietą “1” (naczynie krwionośne)

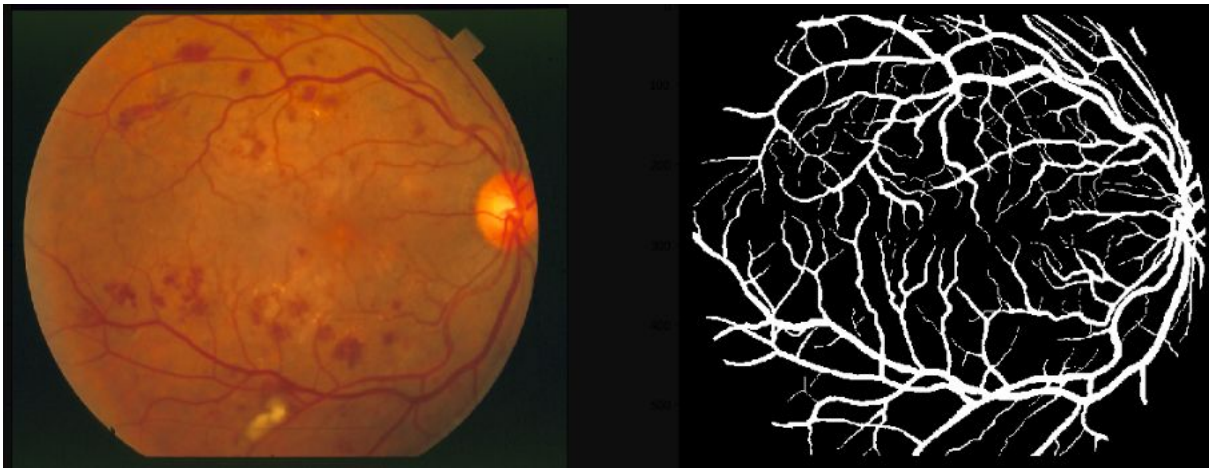
iii. Klasyfikator wybrany przez nas to las losowy zaimplementowany przez bibliotekę scikit-learn z argumentami domyślnymi

iv. Oceniamy klasyfikator przy pomocy k-fold cross validation, gdzie nasze k jest równe 5. Wstępne wyniki to około 80-85% accuracy

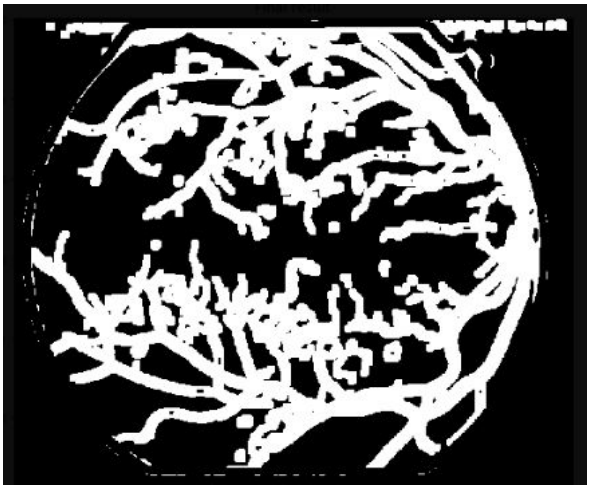
v. Las losowy jest łatwy w wykorzystaniu i nauczeniu nawet z domyślnymi parametrami, a w dodatku jest dosyć wydajny w porównaniu do innych metod które pobieżnie przetestowaliśmy (drzewo i sieć neuronowa). Zastosowanie resamplingu do wyrównania zbioru danych znacznie poprawił czułość klasyfikatora.

Wizualizacja wyników działania programu

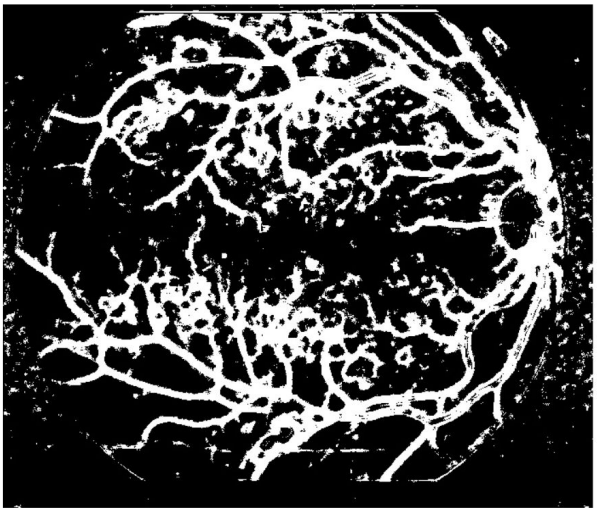
1. Zdjęcie nr 139 (oryginał + maska ekspercka)



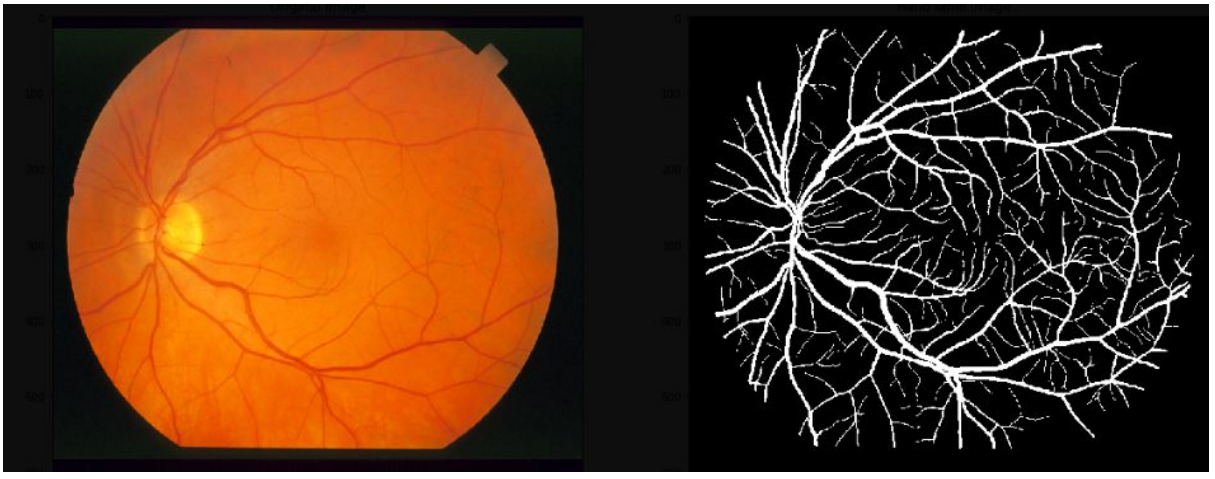
Maska prosta:



Maska zaawansowana:

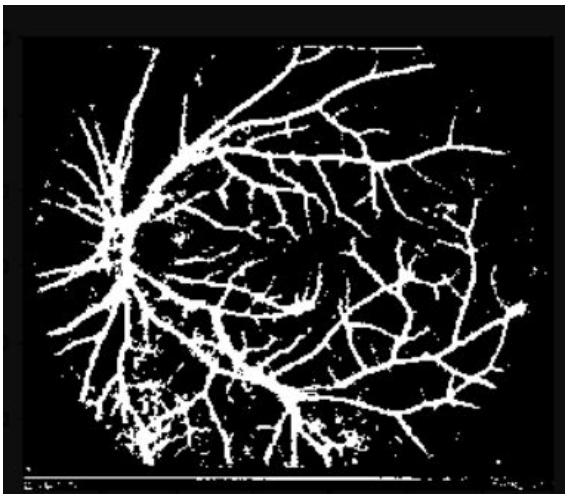


2. Zdjęcie nr 162

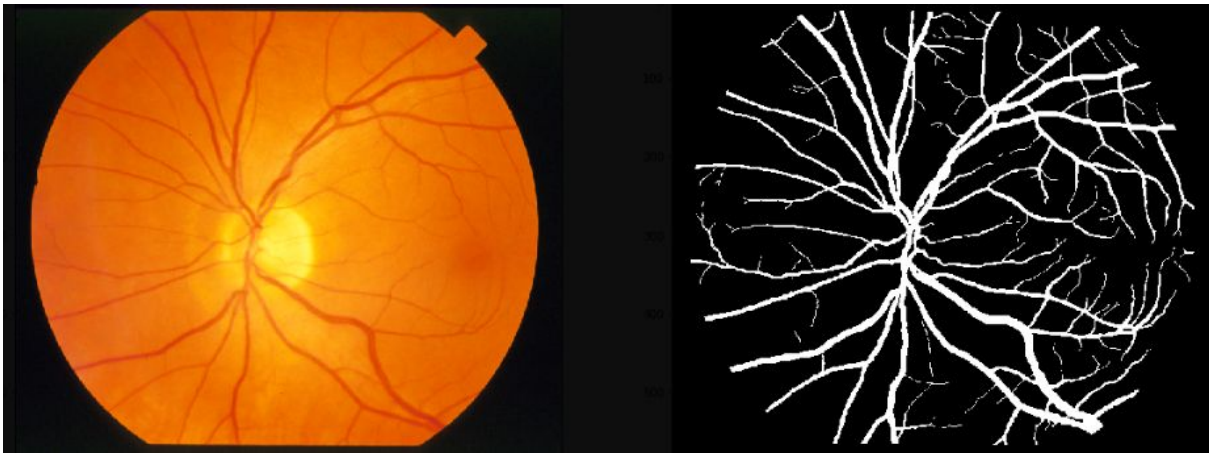


Maska prosta:

Maska zaawansowana:

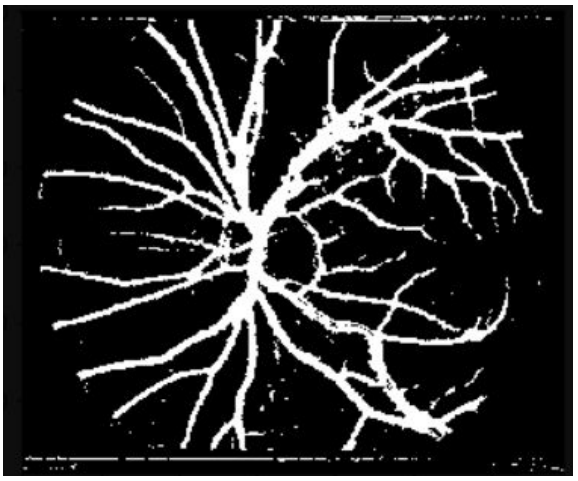


3. Zdjęcie nr 163

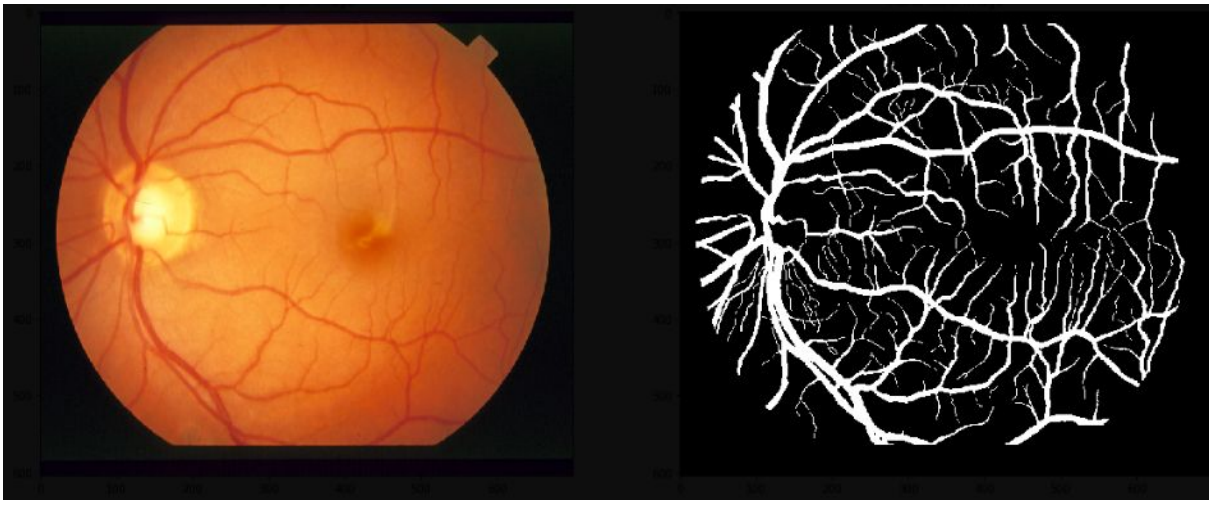


Maska prosta:

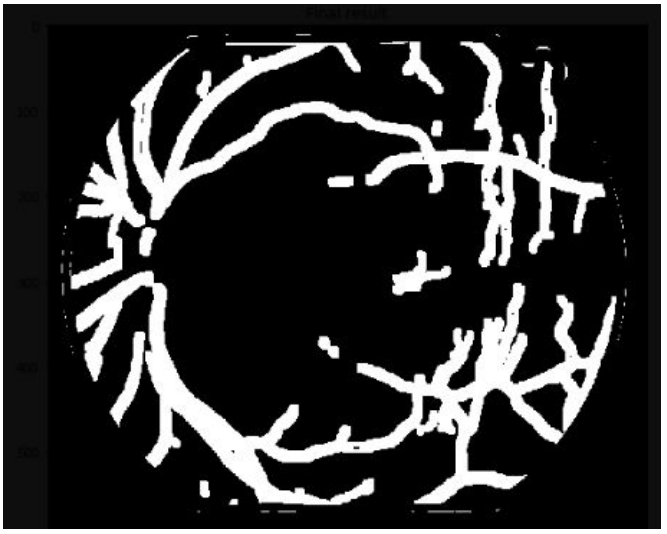
Maska zaawansowana:



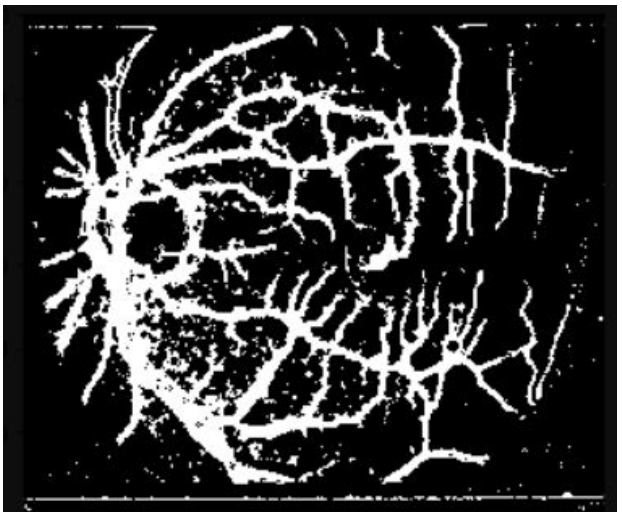
4. Zdjęcie nr 235



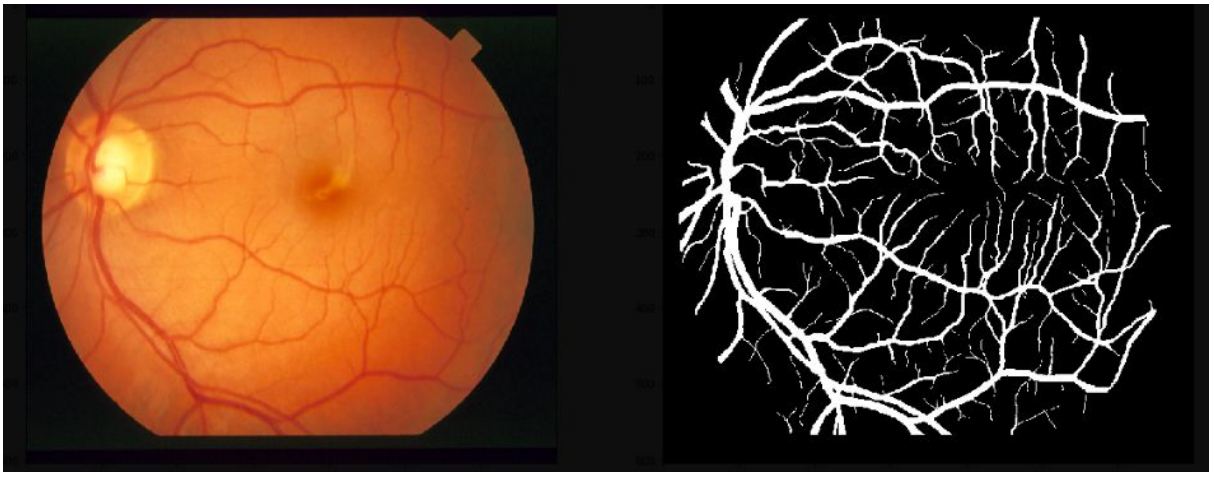
Maska prosta:



Maska zaawansowana:



5. Zdjęcie nr 236



Maska prosta:

Maska zaawansowana:

Analiza wyników działania programu

1. Zdjęcie nr 139
   Maska prosta:
   Macierz pomyłek:
   53866 93895
   10849 264890
   accuracy=0.75267
   sensitivity=0.832357
   specificity=0.738297

   Maska zaawansowana:
   Macierz pomyłek:
   41992 62638
   22723 284982
   accuracy=0.792981
   sensitivity=0.648875
   specificity=0.819808

2. Zdjęcie nr 162
   Maska prosta:
   Macierz pomyłek:
   36370 81263
   12132 293735
   accuracy=0.779468
   sensitivity=0.749865
   specificity=0.783297

   Maska zaawansowana:
   Macierz pomyłek:
   25909 44973
   22593 318860
   accuracy=0.836138

sensitivity=0.534184
specificity=0.876391

3. Zdjęcie nr 163
   Maska prosta:
   Macierz pomyłek:
   30762 47101
   18343 327294
   accuracy=0.845468
   sensitivity=0.626453
   specificity=0.874194

   Maska zaawansowana:
   Macierz pomyłek:
   32165 35070
   16940 328160
   accuracy=0.873864
   sensitivity=0.655024
   specificity=0.903449

4. Zdjęcie nr 235
   Maska prosta:
   Macierz pomyłek:
   36746 41080
   23011 322663
   accuracy=0.848663
   sensitivity=0.614923
   specificity=0.887063

   Maska zaawansowana:
   Macierz pomyłek:
   38673 50925
   21084 301653
   accuracy=0.825362
   sensitivity=0.647171
   specificity=0.855563

5. Zdjęcie nr 236
   Maska prosta:
   Macierz pomyłek:
   37343 41072
   18879 326206
   accuracy=0.858439
   sensitivity=0.664206
   specificity=0.888171

Maska zaawansowana:
Macierz pomyłek:

37138	51612
19084	304501

accuracy=0.828547
sensitivity=0.660559
specificity=0.855068

Wnioski:
Wyniki można interpretować na wiele sposobów, jednakże należy wziąć pod uwagę że sam współczynnik accuracy może niewiele mówić ponieważ cały czarny obraz ma sam w sobie ~85-90%.
Względnie najlepiej oceniony obraz przez klasyfikator to zdjęcie nr 3 - mniej FP oraz FN niż proste filtrowanie, natomiast nr 5 wypada najgorzej w porównaniu z naszym podstawowym algorytmem - posiadając więcej FP niż bardzo surowy obraz podany przez filtr.