

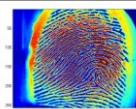
Algorytmy Przetwarzania Obrazów

Algorytmy rozpoznawania obrazów i testowanie działania aplikacji

WYKŁAD 4

Dla studiów niestacjonarnych 2025/2026

Dr hab. Anna Korzyńska, prof. IIBiB PAN



- Rozpoznawanie obrazów jest związane z innymi niż przetwarzania i analiza obrazów dziedzinami nauk komputerowych: uczeniem maszynowym UM, sztuczną inteligencją, komunikacją człowiek-komputer i naśladuje rozpoznawanie obrazów wykonywana przez ludzki mózg

Zastosowania:

- Bioidentyfikacja (oczy, uszy, odciski palców, głos)
- Kontrola jakości produktów, kontrola samochodów na drogach (rozpoznawanie tablic rejestracyjnych), roboty i manipulatory
- Badania przesiewowe (w diagnostyce medycznej)
- Symulatory do nauki prowadzenia pojazdów (samolotów, pojazdów kosmicznych, samochodów wyścigowych, wież kontrolnej lotów)
- Marketing (Yamaha Motor)
- Rozpoznawanie twarzy

3

Rozpoznawanie obrazów

Sztuka odpowiadza na pytanie:

Co przedstawia obraz w kontekście celu rozpoznawania?

Dwa podejścia:

- Klasyczne** oparte na **cechach wybranych przez programiste (rady ekspertów i analizą dyskryminacyjną cech)** i składające się z dwóch etapów:
 - Analizy - czyli ekstrakcji cech wybranych na etapie programowania
 - Właściwego rozpoznawania - zastosowania matematycznych formalizmów do oceny podobieństwa/odległości/korelacji lub innych metod klasyfikacji
- Uczenie maszynowe** - bazujące na **cechach automatycznie wybranych przez oprogramowanie** w procesie optymalizacji zwany również procesem uczenia i jest stosowany w procesie inferencji

2

Rozpoznawanie obrazów

Cel:

Wspomaganie ludzkich decyzji za pomocą informacji obrazowej lub informacji ekstrahowanej z obrazów

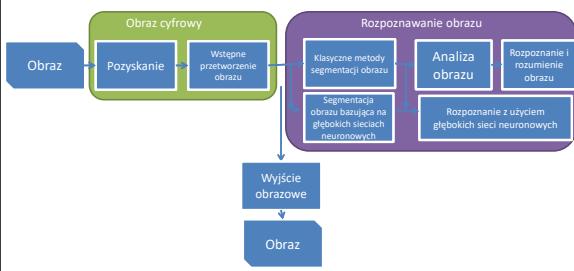
Proces rozpoznawania jest **wieloetapowy**, zawiera dwa typy operacji na obrazach:

- ukierunkowane (detekcja dopasowania, analiza kształtu, pomiar wielkości lub odległości)
- nieukierunkowane (filtracja obrazu, zamiana na obraz monochromatyczny, wyodrębnianie krawędzi)

Może następować ze względu na **cechy** wskazane przez specjalistów lub/i wybrane przez algorytmy dyskryminacji cech lub cechy wybrane w procesie uczenia się sieci neuronowej najczęściej nie mające jednoznacznej interpretacji.

4

Schemat procesu rozpoznawania obrazu



5

Rozpoznawanie obrazów

Jest to proces składający się z następujących operacji:

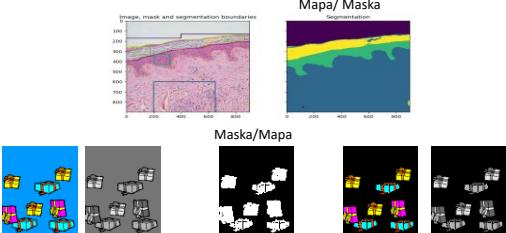
- Pozyskanie (akwizycja)** obrazu i przetworzenie do postaci cyfrowej;
- Wstępne przetworzenie obrazu**, jego filtracja i wyostrzanie, a także jego binaryzacja;
- Podejście klasyczne:**
 - Segmentacja obrazu** i wydzielenie poszczególnych obiektów oraz ich fragmentów (np. krawędzi i innych linii);
 - Analiza obrazu i wyznaczenie cech obiektów oraz informacji o ich lokalizacji;**
 - Rozpoznanie** przez identyfikację klas
- Podejście bazujące na sieciach neuronowych**
 - Segmentacja instancyjna lub semantyczna obrazu**
 - Rozpoznanie** przez identyfikację klas
- Segmentacja/detekcja** wraz z klasyfikacją do klas czyli **rozpoznawaniem**

3

Segmentacja

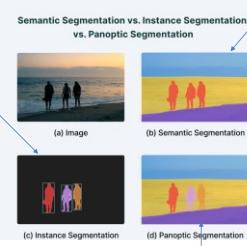
Przygotowanie obrazu do etapu właściwego rozpoznawania obiektów, określenia relacji przestrzennych pomiędzy zidentyfikowanymi obiektami i pomiaru ich wybranych cech, ich klasyfikacji.

Segmentacja stanowi poziom pośredni pomiędzy poziomem *wstępne przetwarzanie* a poziomem *analizy obrazu* w klasycznym ujęciu lub *finalny produkt wskaźnika obiektów z poszczególnych klas* dla segmentacji opartej na sztucznej inteligencji.



Segmentacja semantyczna, instancyjna i panoptyczna

Segmentacja według pojedynczych obiektów
(ang. Instance segmentation)



Segmentacja według znaczenia
(ang. Semantic segmentation)

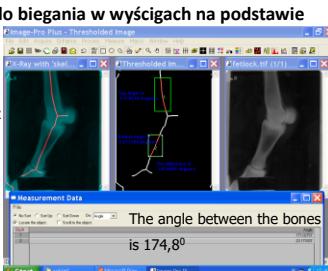
Semantyczne segmentacja daje mapę klas obiektów na obrazie bez rozróżnienia pojedynczych obiektów z danej klasą. Pikiel należące do obiektów danej klasie nie kumulują żadnej informacji o położeniu przyleganiu lub nakładaniu obiektów.

Segmentacja panoptyczna (ang. Panoptic segmentation)

Przykład

Jak ocenić zdolność konia do biegania w wyścigach na podstawie zdjęcia rentgenowskiego stawu skokowego?

Podejście klasyczne :
Analiza obrazu związana jest z ilościowym opisem danych zawartych w obrazie, tj. pomiarem wielkości, kształtu, koloru, określaniem relacji między zidentyfikowanymi elementami obrazu lub badaniem rozproszenia elementów w przestrzeni.



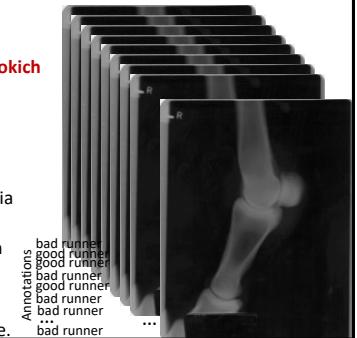
Na podstawie cech geometrycznych przestrzeni i obiektów znajdujących się w obrazie w procesie analizy wyodrębniamy „Znaczenie/treść informacyjna” obrazu mając na uwadze cel jakim jest rozpoznanie obrazu.

Pożądana wartość kąta pomiędzy kośćmi powinna wynosić 176 stopni. ⁹

Przykład

Jak ocenić zdolność konia do biegania w wyścigach na podstawie zdjęcia rentgenowskiego stawu skokowego?

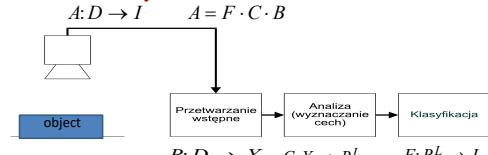
Uczenie maszynowe głębokich sieci neuronowych



Mając duży zestaw przykładów zdjęć rentgenowskich stawów skokowych koni, ze znajomym poziomem zdolności konia do biegania (w formie adnotacji), możliwe jest prezentowanie wszystkich danych sieci neuronowej (NN) i zastosowanie narzędzi w celu zoptymalizowania odpowiedzi NN na pytanie.

Klasyczne metody rozpoznawania obrazów

Schemat klasycznej metody rozpoznawania obrazów



Where:

X - przestrzeń cech ,

$C(x)$ – funkcja przynależności (miara przynależności x do klasy i -th)

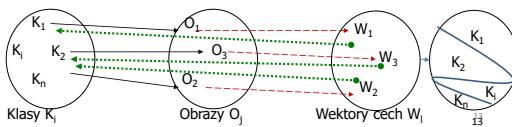
R^L - L liczba wymiarów,

I - zbiór indeksów klas. $i \in I$

Etapy procesu klasycznego rozpoznawania

Faza wstępna

Ekstrakcja **cech charakterystycznych** dla danego **obiektu** w sensie celu rozpoznawania, konstrukcja klasyfikatora na podstawie obrazów za zbiorem uczącym

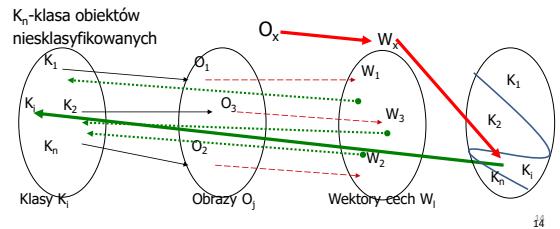


13

Etapy procesu klasycznego rozpoznawania

Faza zasadnicza

Klasifikacja nowych obiektów/obrazów, dokonywana na podstawie tych cech



14

Metody rozpoznawania

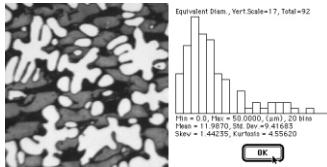
- Deterministyczna (dokładna znajomość wzorców – np. liter)
- Stochastyczna (nie istnieje jednoznaczny wzorzec; niejednoznaczność powoduje, że zakłada się pewien poziom błędu w klasyfikacji)
- Oparta na sieciach neuronowych
- Korelacyjna
- Lingwistyczna
- K-NN

15

Klasyczne podejście do rozpoznawania wymaga analizy obrazu, czyli kwantyfikacji dla każdego obiektu zbioru cech które prowadzą do poprawnej klasyfikacji

Analiza prowadzi do redukcji informacji opisującej obraz do informacji istotnej z punktu widzenia celu

- Ilościowe
 - Intensywność
 - Odległość
 - Rozmiary i wielkości (pole powierzchni, obwód)
 - Rozmiar fraktylny
 - Harmoniczne
- Jakościowe
 - Istnienie wzorców/struktur i symboli
 - Lokalizacja bezwzględna lub wzajemna wzorców/struktur



16

Charakterystyczne cechy obiektów

Analiza kształtu

wskaźniki prezentujące niezmiennosć, invariантność względem obrotów, przesunięć, zmiany skali

Ilość obiektów (lista obiektów)

policzenie przez etykietowanie

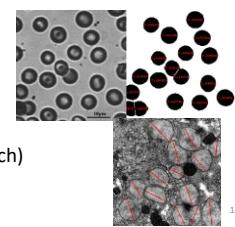
Pole powierzchni

zliczenie pikseli w obiekcie

Oś i długości rzutów

Wzajemne położenia

(drzewko opisu relacji przestrzennych)



18

Analiza kształtu

Współczynniki kształtu

Liczone na podstawie pola powierzchni S i obwodu L obiektu stanowią zgrubne przybliżenie kształtu

Momenty geometryczne

Pozwalają na lepsze rozróżnienie obiektów niż współczynniki kształtu, ale wymagają dłuższych obliczeń

Ani współczynnik kształtu ani moment nie mogą być użyte jako jedyna miara opisująca kształt obiektów (rozpoznanie byłoby niejednoznaczne)

19

Wyznaczanie cech obiektów

Typowa własność cech odniesionych do kształtu obiektu:

- niezmienniczość względem:
 - obrotu.
 - przesunięcia.
 - skali.

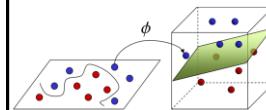
21

Metody klasyfikacji opartej na wektorach wspierających (SVM)

Maszyna wektorów nośnych/wspierających/podpierających

Maszyna wektorów podpierających (*ang. Support Vector Machine – SVM*) to metoda klasyfikacji, która dokonuje transformacji wielowymiarowej przestrzeni cech.

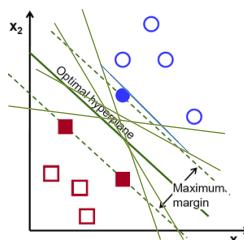
Transformacja zapewnia, że skonstruowana na podstawie przykładów (w naszym przypadku obrazów uczących) hiperplaszczyzna rozdziela klasy (zwaną hiperplaszczyzną decyzyjną) w nowej przestrzeni przebiegała w taki sposób, aby zapewnić oddzielone klas maksymalnym marginesem.



<http://enroute.pl/klasifikacja-metoda-wektor-nosnych-supporting-vector-machines-svm/>

Prosty SVM

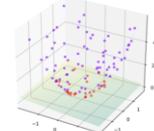
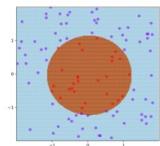
Generujemy hiperplaszczyzny, które segregują na klasy. Następnie wybieramy właściwą hiperplaszczyzny o maksymalnej segregacji (równodległe od najbliższych punktów różnych klas. Z założenie hiperplaszczyzny powinny być funkcjami liniowymi



Zaawansowany SVM

Gdy mamy do czynienia z nierozdzielonymi danymi, czyli takimi, których nie można podzielić używając funkcji liniowej w danej przestrzeni algorytm:

- Ignoruje niektóre punkty w przestrzeni cech, uznając je za nieistotnych outsiderów w grupie
- Przenosimy dane do jeszcze wyższych wymiarów (co nazywamy **kernel trick**) z użyciem transformacji opisanej przy użyciu tzw. funkcji jądrowych (**kernel functions**), takich jak: funkcja wielowymiarowa, funkcja Gaussa, hiperboliczna, itp.



https://www.youtube.com/watch?v=efR1C6CvhmE&feature=emb_rel_end
<https://tomaszkacmajor.pl/index.php/2016/04/17/support-vector-machine/>

Zalety SVM

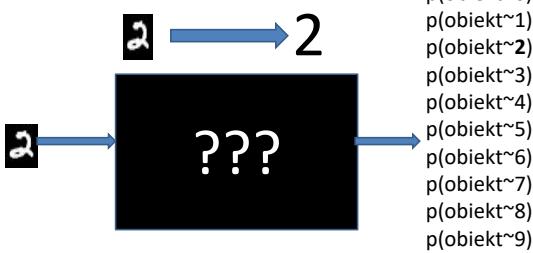
- Zaletą SVM jest wykorzystywanie do danych i problemów liniowych i nieliniowych
- Obsługuje zmienne ilościowe (ciągłe i z kategoryzowanymi) oraz jakościowe
- SVM generuje optymalną hiperpłaszczyznę decyzyjną w sposób powtarzalny, który jest wykorzystywany do minimalizowania błędu.
- Znajduje maksymalne odległości (margins) pomiędzy grupami punktów
- Efektywna obliczeniowo - złożoność rośnie tylko liniowo wraz z liczbą wymiarów

SVM wykorzystujemy nie tylko w klasyfikacji obiektów na obrazach, ale również, w analizie regresji i w klasteryzacji.

O rozpoznawaniu z pomocą sieci neuronowych

Rozpoznawanie cyfr zapisanych ręcznie np.: na czekach - Case study

Jak z obrazu uzyskać informację o cyfrze?



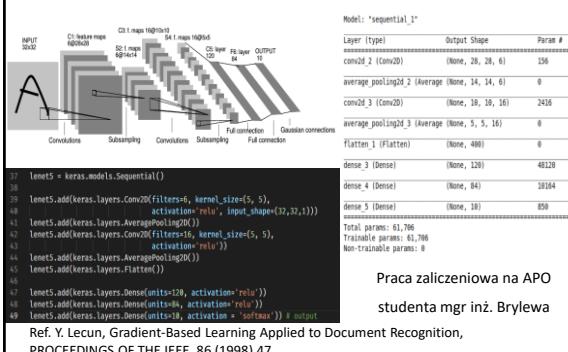
Baza obrazów MNIST

(Modified National Institute of Standards and Technology)

3	4	2	1	9	5	6	2	1	8
8	9	1	2	5	0	0	6	4	
6	7	0	1	6	3	6	3	7	0
3	7	7	9	4	6	6	1	8	2
2	9	3	4	3	9	8	7	2	5
1	5	9	8	3	6	5	7	2	3
9	3	1	9	1	5	8	0	8	4
5	6	2	6	8	5	8	8	9	9
3	7	7	0	9	4	8	5	4	3
7	9	6	4	1	0	6	9	2	3

28 px 28 x 28 = 784 pikseli
 Używamy do nauki:
 THE MNIST DATABASE of
 handwritten digits
<http://yann.lecun.com/exdb/mnist/>

Sieć LeNet

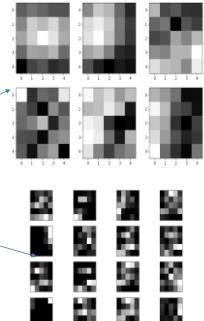


Uczenie się sieci to optymalizacja ze względu na cel

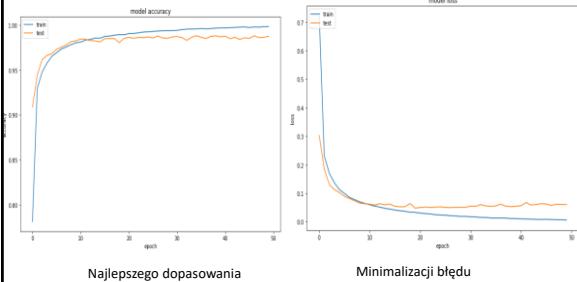
Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 28, 28, 6)	156
average_pooling2d_2 (AveragePooling2D)	(None, 14, 14, 6)	0
conv2d_3 (Conv2D)	(None, 14, 14, 16)	2436
average_pooling2d_3 (AveragePooling2D)	(None, 5, 5, 16)	0
flatten_1 (Flatten)	(None, 400)	0
dense_3 (Dense)	(None, 128)	48120
dense_4 (Dense)	(None, 64)	1024
dense_5 (Dense)	(None, 10)	65

Total params: 61,706
 Trainable params: 61,706
 Non-trainable params: 0

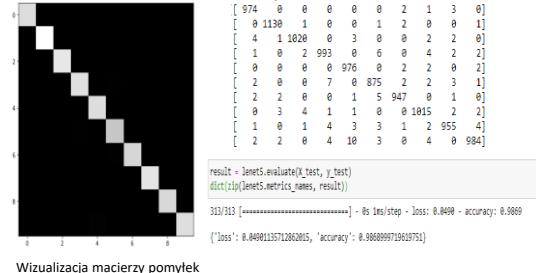
[-0.0789532 -0.02418987 0.10222547 0.00066162 -0.04760809
 0.09279951 -0.01421791 -0.07095297 0.05741353 -0.01717872]



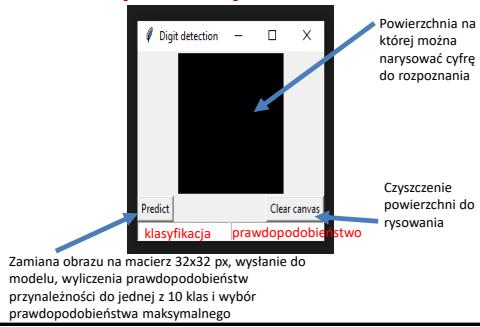
Parametry wyuczenia sieci



Kontrola błędów



Moduł wykonawczy do rozpoznawania znaków narysowanych przez użytkownika



Projekt 51

- Implementacja funkcji detekcji całej twarzy i oczu lemurów za pomocą kaskadowego klasyfikatora opartego na cechach Haara



Testowanie aplikacji

Laboratorium 5

- Analiza błędów, które wykazały **testy sprawdzające poprawność generowanych obrazów**. Test podlegają wzajemnej ocenie czyli przeprowadzają je studenci na oprogramowaniu przez przekazanemu przez wyznaczonego studenta.
- Prezentacja aplikacji rozwijanych na zajęciach.
- Przygotowanie opisu programu tworzonego na zajęciach.
- Omówienie zasad oddawanie i oceny mini-projektu egzaminacyjnego.

Testowanie aplikacji

to **proces** zapewnienia odpowiedniej jakości oprogramowania związanego z **weryfikacją** oraz **validacją** oprogramowania:

- **weryfikacja** oprogramowania pozwala skontrolować, czy wytwarzone oprogramowanie jest **zgodne ze specyfikacją**, a
- wykonanie **analizy statystycznej** oceniającej **działanie oprogramowania** jest jego **validacją**.

- Testowanie nie jest w stanie wykryć wszystkich defektów oprogramowania, jednak może dostarczyć informacji o jego zgodności z wymaganiami klienta, czy też z jego oczekiwaniemi.
- Testowanie nie sprawdza oprogramowania pod kątem wszelkich możliwych warunków początkowych, lecz jedynie w wyselekcjonowanych warunkach.
- Testowanie może we wczesnych fazach projektu wykryć defekty nie tylko oprogramowania, ale i specyfikacji wymagań czy projektu (niedojednoznaczny lub sprzeczności wymagań uniemożliwiające określić poprawnego zachowanie systemu)
- Testowanie oprogramowania – validacja oprogramowania - to analiza statyczna występowania poprawnych i niepoprawnych odpowiedzi oprogramowania za pomocą tzw.: [tablicy pomyłek](#)

Rodzaje testów oprogramowania

Testy można podzielić na kilka sposobów:

- ze względu na weryfikowane obiekty (przykładowo testy klas, komponentów, podsystemów, systemu lub zintegrowanych systemów)
- na **białoskrzynekowe (strukturalne)** - weryfikujące kod źródłowy oraz **czarnoskrzynekowe** testujące warstwę interfejsu
- bazujące na wymaganiach:
 - testy weryfikujące zgodność implementacji z wymaganiami, np. testy funkcjonalne, testy graficznego interfejsu użytkownika),
 - testy niefunkcjonalne – np.: **FURPS(+)** zdefiniowana w ramach Rational Unified Process (RUP) (Funkcjonalność (Funkcjonalność), Usability (Używalność), Reliability (Niezawodność), Performance (Wydajność), Supportability (Wsparcie))
- ze względu na metodę weryfikacji z wyróżnieniem:
 - testów statycznych, bez uruchomienia aplikacji
 - testów dynamicznych wymagających pracę na uruchomionym oprogramowaniu
- dodatkowo można wyróżnić testy wykonane w określonym celu:
 - retesty – testy poprawek defektów
 - testy regresywne – testy oprogramowania po wykonaniu zmian, niekoniecznie w kodzie (przykładowo po wykonaniu aktualizacji wersji systemu operacyjnego)

Testy funkcjonalne.

Polegają one na tym, że wcielamy się w rolę użytkownika, traktując oprogramowanie jak „czarną skrzynkę”, która wykonuje określone zadania. Nie wnikamy w ogóle w techniczne szczegóły działania programu. Testy są nazywane **testami czarnej skrzynki jeśli nie wymagają patrzenia w kod**

Uwaga! Częstym błędem jest stawianie znaku równości między testami funkcjonalnymi, a testami czarnej skrzynki. Testy funkcjonalne mogą wymagać umiejętności czytania kodu źródłowego, czego nie wymaga się przy testach interfejsów zewnętrznych.

Testy strukturalne.

Tym razem tester ma dostęp do kodu źródłowego oprogramowania, może obserwować jak zachowują się różne części aplikacji, jakie moduły i biblioteki są wykorzystywane w trakcie testu. Te testy czasami są nazywane **testami białej skrzynki**

Walidacja oprogramowanie z dziedziny przetwarzania obrazów

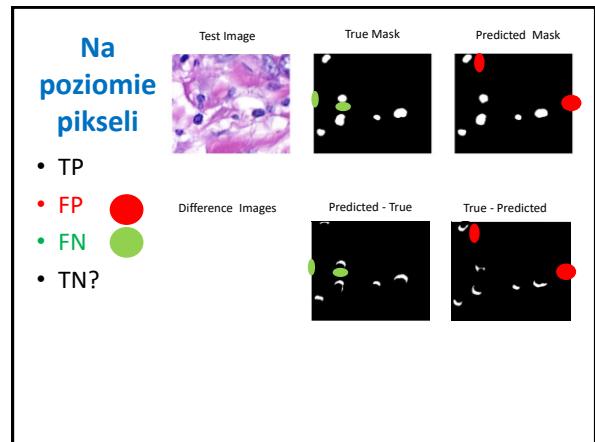
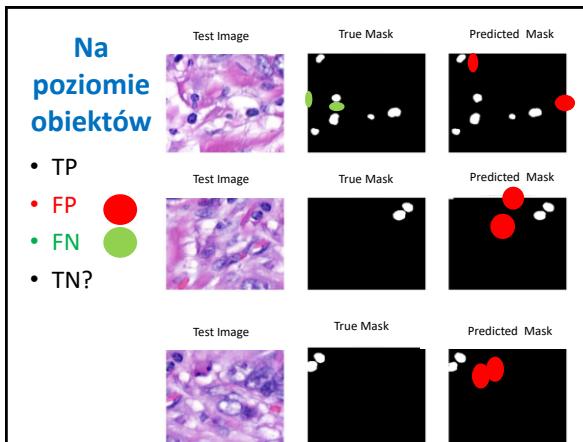
Metryki oceny jakości działania oprogramowania oparte na tablicy pomyłek

Tablica pomyłek służy do oceny jakości klasyfikacji (ang Confusion table/matrix)

- **True-Positive (TP – prawdziwie pozytywna):** przewidywanie pozytywne, faktycznie zaobserwowana klasa pozytywna
- **True-Negative (TN – prawdziwie negatywna):** przewidywanie negatywne, faktycznie zaobserwowana klasa negatywna
- **False-Positive (FP – fałszywie pozytywna):** przewidywanie pozytywne, faktycznie zaobserwowana klasa negatywna
- **False-Negative (FN – fałszywie negatywna):** przewidywanie negatywne, faktycznie zaobserwowana klasa pozytywna

- Obiekty znalezione/sklasyfikowane
- Obiekty faktyczne istniejące

		Stan faktyczny „Ground true”	
		P	N
		P	TP True-Positive
		N	FN False-Negative
		TN True-Negative	FP False-Positive

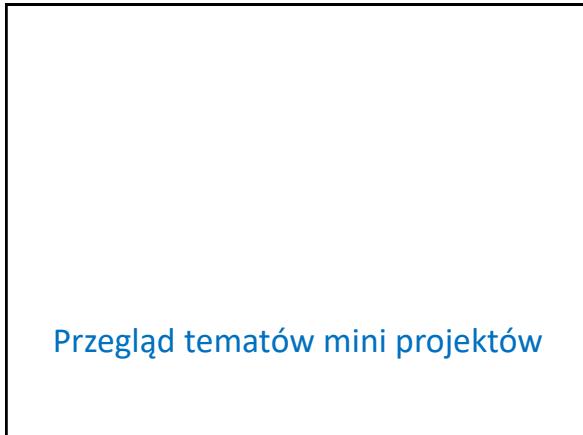


		Klasa predykowana – wynik testu		Częstość występowania, chorobowość $\frac{\sum \text{stan pozytywny}}{\sum \text{populacja}}$
	Populacja	Klasifikacja pozytywna	Klasifikacja negatywna	
Klasa rzeczywista	Stan pozytywny	prawdziwie dodatnia, TP (błęd drugiego rodzaju, FN)	falszywie ujemna (błęd pierwszego rodzaju, FP)	czułość, TPR $\frac{\sum \text{TP}}{\sum \text{TP} + \sum \text{FN}}$
	Stan negatywny	falszywie dodatnia (błęd pierwszego rodzaju, FP)	prawdziwie ujemna, TN	specjalność, SPC, TNR $\frac{\sum \text{TN}}{\sum \text{FP} + \sum \text{TN}}$
	dokładność, ACC $\frac{\sum \text{TP} + \sum \text{TN}}{\sum \text{populacja}}$	pracyja, PPV $\frac{\sum \text{TP}}{\sum \text{TP} + \sum \text{FP}}$	FOR $\frac{\sum \text{FN}}{\sum \text{FN} + \sum \text{TN}}$	LR+ $\frac{\text{TPR}}{\text{FPR}}$
		FDR $\frac{\sum \text{FP}}{\sum \text{TP} + \sum \text{FP}}$	NPV $\frac{\sum \text{TN}}{\sum \text{FN} + \sum \text{TN}}$	LR- $\frac{\text{FNR}}{\text{TNR}}$
				DOR $\frac{\text{LR+}}{\text{LR-}}$

Podsumowując:

Testowania umożliwia wykrycie błędów we wczesnych stadiach rozwoju oprogramowania, co pozwala zmniejszyć koszty usuwania tego błędu.

- Warto testować na każdym etapie tworzenia oprogramowania.
- Testować należy jak najwcześniej, ponieważ im później wykryty zostanie błąd tym większy jest koszt jego usunięcia.



	Tytuł projektu dla APOZ 2025/2026	Nr indeksu studenta	Wybrany język programowania
1	Udoskonalenie oprogramowania przygotowanego na zajęciach przez dołączanie oprogramowania do segmentacji obiektów na podstawie wskazanego na wszystkich kanałach zakresu intensywności oraz zliczania wysegmentowanych obiektów z powstalej mapy binarnej.	23557	Python
2	Udoskonalenie oprogramowania przygotowanego na zajęciach przez implementację funkcji detekcja całej twarzy i oczu za pomocą kaskadowego klasyfikatora opartego na cechach Haara	23091	Python
3	Udoskonalenie oprogramowania przygotowanego na zajęciach przez: dołożenia operacji erozji i dyfuzji działających na dowolnie zdefiniowanym elemencie strukturyzującym – proszę zaprojektować i wykonać edytor elementu strukturyzującego.	23241	Python
4	Udoskonalenie oprogramowania przygotowanego na zajęciach przez przygotowanie funkcji liczenie FFT i IFFT oraz edytora manipulacji widmem amplitudowym.	16958	Python
5	Segmentacja obrazu monochromatycznego/binarnego zawierającego nuty i rozpoznanie nut.		
6	Program prezentacji zasad przebiegu procesu wprowadzania i korekcji zniekształceń radiometrycznych z wykorzystaniem obrazów szaro odciennowych oraz ich fragmentów w postaci obrazowej i także tabeli liczb.	20668	Python
7	Udoskonalenie oprogramowania przygotowanego na zajęciach przez przygotowanie funkcji wyznaczającej kontur obiektu w binarnym obrazie: porównanie rozwiązań opartych na metodach morfologii matematycznej i konwolucyjnych operacji konturuowania.	22388	Python
8	Udoskonalenie oprogramowania przygotowanego na zajęciach przez przygotowanie funkcji zmieniającej obraz według przekształcenia opisanego przemieszczeniem trzech nie współliniowych punktów.	23337	Python
9	Udoskonalenie oprogramowania przygotowanego na zajęciach przez przygotowanie funkcji przyjęcia (kadrówanię) obrazu według protokołu niekoniecznie równoległego do osi X i Y wyznaczonego odpowiednim narzędziem wskazywaną położenia.	23391	Python
10	Przeniesienie oprogramowania stworzonego na zajęciach tak, aby działał dla systemu operacyjnego iOS.	23625	

11 Oprogramowanie do wykonyania transformaty Hough'a	20468	Python
12 Oprogramowanie do nakładania zniekształceń geometrycznych na obrazy monochromatyczne i edytora doboru funkcji korygujących takie zniekształcenia.		
13 Udoskonalenie oprogramowania przygotowanego na zajęciach przez: doliczenie operacji wyliczenia transformaty odległościowej dla obrazu binarnego oraz implementację rozdzielania obiektów dotyczących się z wykorzystaniem tej transformaty.		
14 Segmentacja obrazu monochromatycznego/binarnego zawierającego znaki specjalne i symbole oraz wyszukiwanie znaków o kształtach określonych np.: ☰, E, *, ◆, ☷, &, itp..	18629	Java
15 Segmentacja z obrazów monochromatycznych/binarnych zawierających wybrane 5 typów symboli o różnym rozmiarze, ich rozpoznanie metodami klasycznymi lub opartymi o sieci neuronowe.	17677	Java
16 Segmentacja obrazów z wykorzystaniem klasteryzacji szarych odcięć.	20667	Python
17 Udoskonalenie oprogramowania przygotowanego na zajęciach przez implementację progowanie obrazu prawdopodobieństwa przypasowania do zadanej tekstuury.		
18 Udoskonalenie oprogramowania przygotowanego na zajęciach przez implementację nowego narzędzie do tworzenia panoramy na postawie serii zdjęć.	9876	Python
19 Implementacja lini profili i wycinanie nie będących prostokątem fragmentów z istniejących obrazów monochromatycznych.		
20 Udoskonalenie oprogramowania przygotowanego na zajęciach przez wykonanie narzędzia do ekstrakcji linii pionowych i poziomych za pomocą operacji morfologicznych.	Skłosowy any	Skłosowy ny
21 Udoskonalenie oprogramowania przygotowanego na zajęciach przez przygotowanie funkcji wykonywania wyrównania histogramu obrazu kolorowego zapisanego z wykorzystaniem modelu L*a*b*.	21432	Python
22 Udoskonalenie oprogramowania przygotowanego na zajęciach przez: -implementacja operacji filtracji logicznych na obrazach binarnych; -rozwiniecie możliwości wyświetlania i zapisywania jako obraz fragmentów obrazu powstały na podstawie wskazanych, niekoniecznie sąsiednich, fragmentów histogramu.	21346	Python

23 Segmentacja obrazu monochromatycznego zawierającego drobne obiekty metalowe o różnych kształtach zarejestrowane na taśmie produkcyjnej przez kamerę monochromatyczną.	21213	Python
24 Program prezentacji sposobu działania metody α -NN (α najbliższych sąsiadów) z wizualizacją przestrzeni cech przed i po fazie uczenia.	21296	C#
25 Udoskonalenie oprogramowania przygotowanego na zajęciach przez implementację operacji przenikania dwóch obrazów monochromatycznych.	21075	Python
26 Udoskonalenie oprogramowania przygotowanego na zajęciach przez wykonanie narzędzia do rozcigania i zauważania histogramy z możliwym zastosowaniem funkcji liniowej lub funkcji gamma (analogicznie jak w Corelu PhotoPaint-cie).	21918	C#
27 Udoskonalenie oprogramowania przygotowanego na zajęciach przez doliczenie operacji otoczkii wypuklej obiektu – operacja morfologii matematycznej		
28 Segmentacja obrazu monochromatycznego/binarnego zawierającego znaki specjalni i symbole oraz wyszukiwanie znaków o kształtach wypukłych np.: ☰, E, *, ◆, ☷, &, itp..		
29 Udoskonalenie oprogramowania przygotowanego na zajęciach przez doliczenie operacji tworzenia histogramu dwuwymiarowego z obrazu monochromatycznego i jego matematycznego przekształcenia. Implementacja narzędzia do segmentacji na podstawie histogramu dwuwymiarowego.		
30 Udoskonalenie oprogramowania przygotowanego na zajęciach przez doliczenie operacji zimplementowanie operacji rekonstrukcji morfologicznej.	21611	Python
31 Udoskonalenie oprogramowania przygotowanego na zajęciach przez doliczenie operacji zmniejszenia udziału szumu przez uszczelnianie kilku obrazów zebranych w takich samych warunkach oraz operacji logicznych na zasumionych obrazach binarnych.	21371	Python

21 osób: 17-Python; 2-C#; 2-JAVA

Zaliczenie przedmiotu

- Laboratorium zalicza 26-50 punktów zdobytych na zajęciach na podstawie ocenionej przez prowadzącego aplikacji przesłanej na Teamsach (na dysku P - w odpowiednim katalogu).
- Do egzaminu mogą przystąpić Ci studenci, którzy mają minimum 26 i którzy przedstawili działające oprogramowanie projektu egzaminacyjnego w postaci filmu i pełnej wersji oprogramowania rozwijanego na zajęciach z dołączonym mini-projektem.

Egzamin

Obrona przygotowanego samodzielnie projektu następuje po jego przykazaniu:

- Projekt należy przed egzaminem (data zostanie ogłoszona – najprawdopodobniej na 3 dni robocze przed terminem egzaminu) w formie wykonywalnej wraz z kodem. W większości przypadków będzie to ten sam projekt z rozwiniętymi lub dodanymi funkcjonalnościami.
- Oprócz projektu egzaminacyjnego, proszę wgrać: **instrukcję dla użytkownika** sporządzoną na laboratorium 6 oraz **7-10 minutowy film** z wykładem prezentującym projekt. Film będzie oceniany według kryteriów:

Co będzie oceniane

- Zrozumienia stosownalności i założień projektu
- Kompletność prezentacji – podstawy, metody i funkcjonalności
- Sposób prezentacji
- Organizacja prezentacji

Kod będzie sprawdzany pod kątem

- Struktury oprogramowania
- Poziomu komentowania kodu
- Opisu zmiennych, parametrów globalnych i lokalnych
- Interface graficzny - funkcjonalność
- Interface graficzny - walory graficzne
- Oceny zgodności rezultatów z założeniami

Ocena dokumentacji

- Organizacja dokumentacji
- Kompletność dokumentacji
- Szata graficzna
- Dobór obrazów demonstracyjnych
- Dobór zrzutów z ekranów
- Spis treści i wyszukiwanie informacji

Za całość będzie można dostać 50 punktów

Zaliczenie przedmiotu to suma punktów za laboratoria i z egzaminu przeliczona na stopnie

Materiał:

- M.Doros, Przetwarzanie obrazów, skrypt WSISIZ
- Materiały wykładowe POBZ z zeszłego roku na UBIKu
- T.Pavlidis, Grafika i Przetwarzanie Obrazów, WNT Warszawa 1987.
- I.Pitas, Digital image processing, algorithms and applications, John Wiley & Sons, Inc. 2000, pp. 162-166 (w katalogu ...\\APOZ\\Materiały na UBIKu).

Literatura dodatkowa:

- W.Zielinski, M.Strzelecki: Komputerowa analiza obrazu biomedycznego, PWN Warszawa-Lódź 2002, str. 178-214; segmentacja z wykorzystaniem analizy tekstur, mozaika Voronoi (Voronoi tesselation), segmentacja metodą określenia działów wodnych (watershed transform)
- T.Pavlidis: Grafika i Przetwarzanie Obrazów, WNT Warszawa 1987
- R.Tadeusiewicz, P.Korohoda, Komputerowa analiza i przetwarzanie obrazów, Wydawnictwo Fundacji Postępu Telekomunikacji, Kraków 1997. <http://wnnibg.bog.agh.edu.pl/skrypty2/0098/>
- Zasoby sieciowe:**
- Segmentacja (w szczególności wododziałowa (*watershed*))
<https://www.mathworks.com/company/newsletters/articles/the-watershed-transform-strategies-for-image-segmentation.html>
- Definicja tekstury:
http://ai.stanford.edu/~ruzon/tex_seg/node1.html