

Algorytmy Przetwarzania Obrazów

Algorytmy rozpoznawania obrazów i testowanie działania aplikacji

WYKŁAD 4

Dla studiów niestacjonarnych 2025/2026

Dr hab. Anna Korzyńska, prof. IBIB PAN

Rozpoznawanie obrazów

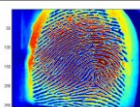
Sztuka odpowiedzi na pytanie:

Co przedstawia obraz w kontekście celu rozpoznawania?

Dwa podejścia:

- Klasyczne** oparte na **cechach** wybranych przez programistę (rady ekspertów i analiza dyskryminacyjna 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 zwanym również procesem uczenia i jest stosowany w procesie inferencji

2



- 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, wieży kontrolnej lotów)
 - Marketing (Yamaha Motor)
 - Rozpoznawanie twarzy

3

Rozpoznawanie obrazów

Cel:

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

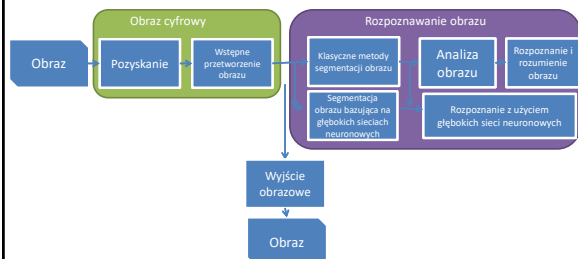
Proces rozpoznawania jest **wieloletowy**, zawiera dwa typy operacji na obrazach:

- 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 wyodrębnienie, a także jego binaryzacja;
- Podejście klasyczne**:
 - 1. Segmentacja obrazu** i wydzielenie poszczególnych obiektów oraz ich fragmentów (np. krawędzi i innych linii);
 - 2. Analiza obrazu** i wyznaczenie cech obiektów oraz informacji o ich lokalizacji;
 - 3. Rozpoznanie i rozumienie obrazu** (identyfikacja klasy).
- Podejście bazujące na sieciach neuronowych**
 - 1. Segmentacja instancyjna lub semantyczna obrazu**
 - 2. Rozpoznanie** przez identyfikację klasy
 - 3. Segmentacja/detekcja** wraz z klasyfikacją do klas czyli rozpoznawaniem

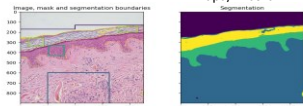
3

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 *wstępnego przetwarzania* a poziomem *analizy obrazu* w klasycznym ujęciu lub *finalny produkt wskazania obiektów z poszczególnych klas* dla segmentacji opartej na sztucznej inteligencji.

Mapa/ Maska



Maska/ Mapa

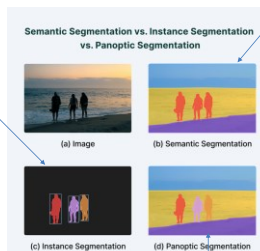


Segmentacja semantyczna, instancyjna i panotypyczna

Segmentacja według pojedynczych obiektów (ang. Instance segmentation)

Segmentacja według znaczenia (ang. Semantic segmentation)

Segmentacja według pojedynczych obiektów daje mapę obiektów danej kategorii i każdego obiektu danej kategorii oddzielnie. Jest operacją trudną wymagającą oddzielenia obiektów przyległych i nakładających się. Odzwierciedla reprezentację obiektów na poziomie pikseli.



Semantyczne segmentacja daje mapę klas obiektów na obrazie bez rozróżnienia pojedynczych obiektów z danej klasy. Piksele należące do obiektów danej klasy nie kumulują żadnej informacji o położeniu przylegania lub nakładaniu obiektów.

Segmentacja panotypyczna (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, określeniem 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 informacyjną” obrazu mając na uwadze cel jakim jest rozpoznanie obrazu.

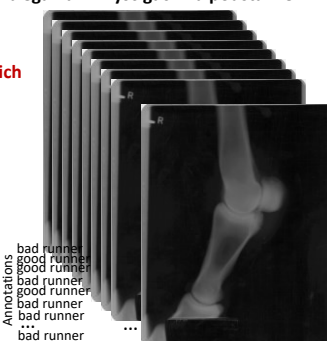
Pożądana wartość kąta pomiędzy kośćmi powinna wynosić 176 stopni. 9

Przykład

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

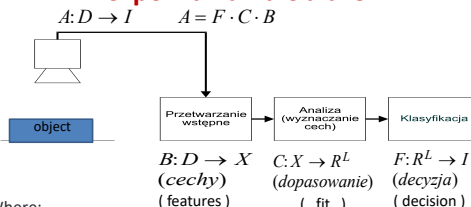
Uczenie maszynowe głębokich sieci neuronowych

Mając duży zestaw przykładów zdjęć rentgenowskich stawów skokowych koni, ze znanym 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

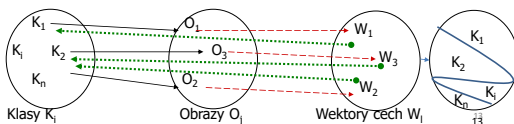
Schemat klasycznej metody rozpoznawania obrazów



Etapy procesu klasycznego rozpoznawania

Faza wstępna

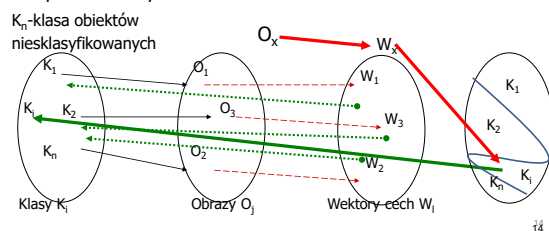
Ekstrakcja **cech charakterystycznych** dla danego **obiektu** w sensie celu rozpoznawania, konstrukcja klasyfikatora na podstawie obrazów za zbioru uczącego



Etapy procesu klasycznego rozpoznawania

Faza zasadnicza

Klasyfikacja nowych obiektów/obrazów, dokonywana na podstawie tych cech



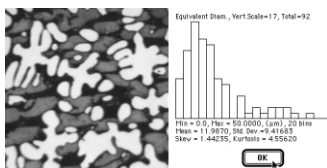
Metody rozpoznawania

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

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 fraktalny
 - Harmoniczne
- Jakościowe
 - Istnienie wzorców/struktur i symboli
 - Lokalizacja bezwzględna lub wzajemna wzorców/struktur



Charakterystyczne cechy obiektów

Analiza kształtu

wskaźniki prezentujące niezmienność, inwariantność względem obrotów, przesunięć, zmiany skali

Ilość obiektów (lista obiektów)

policzenie przez etykietowanie

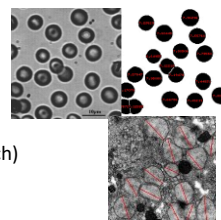
Pole powierzchni

zliczenie pikseli w obiekcie

Osie i długości rzutów

Wzajemne położenia

(drzewko opisu relacji przestrzennych)



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:

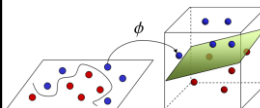
- niezmienniczość względem:
 - obrotu.
 - przesunięcia.
 - skali.

21

Metody klasyfikacji opartej na wektorach wspierających (SVM)

Maszyna wektorów nośnych/wspierających/podpierających

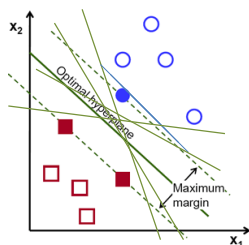
Maszyna wektorów podpierających (*ang.* Support Vector Machine – SVM) to metoda klasyfikacji, która dokonuje transformacji wielowymiarowej przestrzeni cech. Transformacja zapewnia, że skonstruowana na podstawie przykładów (w naszym przypadku obrazów uczących) hiperpłaszczyzna rozdzielająca klasy (zwaną hiperpłaszczyzną decyzyjną) w nowej przestrzeni przebiegała w taki sposób, aby zapewnić oddzielone klasy maksymalnym marginesem.



<http://enroute.pl/klasyfikacja-metoda-wektorow-nosnych-supporting-vector-machines-svm/>

Prosty SVM

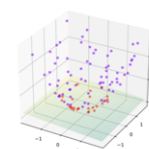
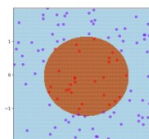
Generujemy hiperpłaszczyzny, które segregują na klasy. Następnie wybieramy właściwą hiperpłaszczyznę o maksymalnej segregacji (równoodległe od najbliższych punktu różnych klas. Z założenie hiperpłaszczyzny powinny być funkcjami liniowymi



Zaawansowany SVM

Gdy mamy do czynienia z nierozdzielnymi 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=efR1C6CvnmE&feature=emb_rel_end
<https://tomaszkacmajor.pl/index.php/2016/04/17/support-vector-machine/>

Zalety SVM

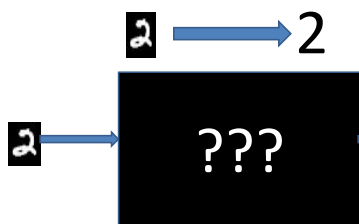
- Zaletą SVM jest wykorzystywanie do danych i problemów liniowych i nieliniowych
- Obsługuje zmienne ilościowe (ciągłe i zkatęgoryzowane) oraz jakościowe
- SVM generuje optymalną hiperpłaszczyznę decyzyjną w sposób powtarzalny, który jest wykorzystywany do minimalizowania błędu.
- Znajduje maksymalne odległości (marginesy) 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?



p(obiekt~0)
p(obiekt~1)
p(obiekt~2)
p(obiekt~3)
p(obiekt~4)
p(obiekt~5)
p(obiekt~6)
p(obiekt~7)
p(obiekt~8)
p(obiekt~9)

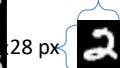
Baza obrazów MNIST

(Modified National Institute of Standards and Technology)



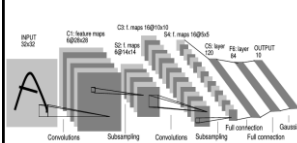
- 70 000 obrazów ręcznie pisanych cyfr
- 55 000 uczenie, 5 000 walidacja
- 10 000 testy
- 28x28 pikseli
- Wydany w 1999 roku
- 500 różnych osób
- Obraz + etykieta (jaka to cyfra)
- Skala kolorów znormalizowana do 0.0...1.0

28 px 28 x 28 = 784 pikseli



Używamy do nauki:
THE MNIST DATABASE of
handwritten digits
<http://yann.lecun.com/exd/b/mnist/>

Sieć LeNet



```
17 lenet5 = keras.models.Sequential()
18 lenet5.add(keras.layers.Conv2D(filters=6, kernel_size=(5, 5),
19 activation='relu', input_shape=(32,32,1)))
20 lenet5.add(keras.layers.AveragePooling2D())
21 lenet5.add(keras.layers.Conv2D(filters=16, kernel_size=(5, 5),
22 activation='relu'))
23 lenet5.add(keras.layers.AveragePooling2D())
24 lenet5.add(keras.layers.Flatten())
25 lenet5.add(keras.layers.Dense(units=120, activation='relu'))
26 lenet5.add(keras.layers.Dense(units=84, activation='relu'))
27 lenet5.add(keras.layers.Dense(units=10, activation='softmax'))
```

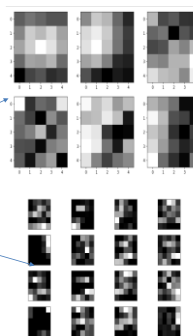
Ref. Y. Lecun, Gradient-Based Learning Applied to Document Recognition,
PROCEEDINGS OF THE IEEE. 86 (1998) 47.

Model: "sequential_1"		
Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 28, 28, 6)	156
average_pooling2d_2 (Average)	(None, 14, 14, 6)	0
conv2d_3 (Conv2D)	(None, 16, 16, 16)	2416
average_pooling2d_3 (Average)	(None, 8, 8, 16)	0
flatten_1 (Flatten)	(None, 400)	0
dense_3 (Dense)	(None, 120)	48120
dense_4 (Dense)	(None, 84)	33564
dense_5 (Dense)	(None, 10)	850
Total params:	61,796	
Trainable params:	61,796	
Non-trainable params:	0	

Praca zaliczeniowa na APO
studenta mgr inż. Brylewa

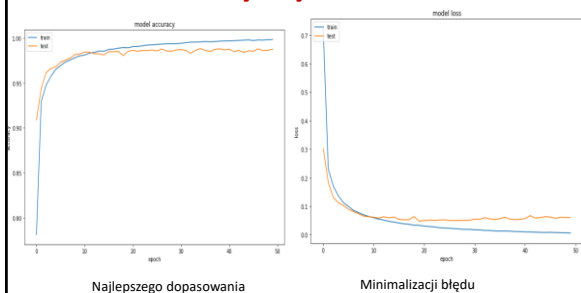
Uczenie się sieci to optymalizacja ze względu na cel

Model: "sequential_1"		
Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 28, 28, 6)	156
average_pooling2d_2 (Average)	(None, 14, 14, 6)	0
conv2d_3 (Conv2D)	(None, 16, 16, 16)	2416
average_pooling2d_3 (Average)	(None, 8, 8, 16)	0
flatten_1 (Flatten)	(None, 400)	0
dense_3 (Dense)	(None, 120)	48120
dense_4 (Dense)	(None, 84)	33564
dense_5 (Dense)	(None, 10)	850
Total params:	61,796	
Trainable params:	61,796	
Non-trainable params:	0	

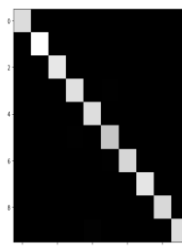


[-0.0789532 -0.02418987 0.10222547 0.00066162 -0.04760809
0.09279951 -0.01421791 -0.07095297 0.05741353 -0.01717872]

Parametry wyuczenia sieci



Kontrola błędów



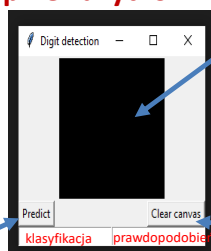
Wizualizacja macierzy pomyłek

```
[ 974  0  0  0  0  0  2  1  3  0]
[  0 1130  1  0  0  1  2  0  0  1]
[  4  1 1020  0  3  0  0  2  2  0]
[  1  0  2 993  0  6  0  4  2  2]
[  0  0  0  0 976  0  2  2  0  2]
[  2  0  0  7  0 875  2  2  3  1]
[  2  2  0  0  1  5 947  0  1  0]
[  0  3  4  1  1  0  0 1815  2  2]
[  1  0  1  4  3  3  1  2 955  4]
[  2  2  0  4 10  3  0  4  0 984]
```

```
result = lenet5.evaluate(X_test, y_test)
dict(zip(lenet5.metrics_names, result))

313/313 [*****] - 8s 1ms/step - loss: 0.0400 - accuracy: 0.9869
{'loss': 0.04001135712862015, 'accuracy': 0.9868899719619751}
```

Moduł wykonawczy do rozpoznawania znaków narysowanych przez użytkownika



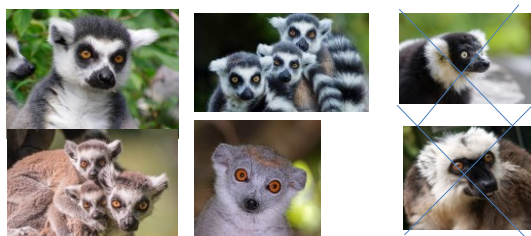
Powierzchnia na której można narysować cyfrę do rozpoznania

Czyszczenie powierzchni do rysowania

Zamiana obrazu na macierz 32x32 px, wysłanie do modelu, wyliczenia prawdopodobieństw przynależności do jednej z 10 klas i wybór prawdopodobieństwa maksymalnego

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 oddawania i oceny mini-projektu egzaminacyjnego.

Testowanie aplikacji

to **proces** zapewnienia odpowiedniej **jakości oprogramowania** związany z **weryfikacją oraz walidacją** oprogramowania:

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

- Testowanie **nie jest w stanie wykryć wszystkich defektów oprogramowania**, jednak może dostarczyć informacji o jego zgodności z wymaganiami klienta, czy też z jego oczekiwaniami.
- 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 (niejednoznaczności lub sprzeczności wymagań uniemożliwiające określić poprawnego zachowanie systemu)
- Testowanie oprogramowania – walidacja 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 **białoskrzynkowe (strukturalne)** - weryfikujące kod źródłowy oraz **czarnoskrzynkowe** 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)
- ze względu na metodę weryfikacji z wyróżnieniem:
 - testów statycznych, bez uruchomienia aplikacji
 - testów dynamicznych wymagającej 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)

Functionality
(Funkcjonalność)
Usability
(Używalność)
Reliability
(Niezawodność)
Performance
(Wydajność)
Supportability
(Wsparcie)

• 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/matix)

- **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

Przewidywanie/znalezienie/klasyfikowanie

- Obiekty znalezione/klasyfikowane
- Obiekty faktycznie istniejące

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



Na poziomie obiektów

- TP
- FP
- FN
- TN?

Test Image

True Mask

Predicted Mask

Test Image

True Mask

Predicted Mask

Test Image

True Mask

Predicted Mask

Na poziomie pikseli

- TP
- FP
- FN
- TN?

Test Image

True Mask

Predicted Mask

Difference Images

Predicted - True

True - Predicted

Metryki budowane na podstawie tablicy pomyłek

		Klasa predykowana – wynik testu		Częstość występowania, chorobowość	
		Populacja	Klasyfikacja pozytywna	Klasyfikacja negatywna	
Klasa rzeczywista	Stan pozytywny	prawdziwie dodatnia, TP	falszywie ujemna (błąd drugiego rodzaju, FN)	$\frac{\sum TP}{\sum TP + \sum FN}$ czułość, TPR	$\frac{\sum FN}{\sum TP + \sum FN}$ FNR
	Stan negatywny	falszywie dodatnia (błąd pierwszego rodzaju, FP)	prawdziwie ujemna, TN	$\frac{\sum FP}{\sum FP + \sum TN}$ FPR	$\frac{\sum TN}{\sum FP + \sum TN}$ specificity, SPC, 1-FPR
	$\frac{\sum TP + \sum TN}{\sum poplarz}$ dokładność, ACC	$\frac{\sum TP}{\sum TP + \sum FP}$ precyzja, PPV	$\frac{\sum FN}{\sum FN + \sum TN}$ FOR	$\frac{\sum TP}{\sum TP + \sum TN}$ LR+	$\frac{\sum TN}{\sum FP + \sum TN}$ DOR
		$\frac{\sum FP}{\sum FP + \sum TP}$ FOR	$\frac{\sum TN}{\sum FN + \sum TN}$ NPV	$\frac{TPR}{FPR}$ LR-	$\frac{FNR}{TNR}$ 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.

Przegląd tematów mini projektów

Tytuł projektu dla APOZ 2025/2026		Nr indeksu studenta	Wybrany język programowania
1	Udoskonalenie oprogramowania przygotowanego na zajęciach przez dołączenie oprogramowania do segmentacji obiektów na podstawie wskazanego na wszystkich kanałach zakresu intensywności oraz zliczania wysegmentowanych obiektów z powstałej mapy binarnej.	23557	Python
2	Udoskonalenie oprogramowania przygotowanego na zajęciach przez implementację funkcji detekcji całej twarzy i oczu za pomocą kaskadowego klasyfikatora opartego na cechach Haara	23091	Python
3	Udoskonalenie oprogramowania przygotowanego na zajęciach przez: dołożenie 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 liczenia FFT i IFFT oraz edytora manipulacji widmem amplitudowym.	16058	Python
5	Segmentacja obrazu monochromatycznego/binarnego zawierającego nuty i rozpoznawanie nut.	20608	Python
6	Program prezentacji zasad przebiegu procesu wprowadzania i korekty zniekształceń radiometrycznych z wykorzystaniem obrazów szaro odcieniowych oraz ich fragmentów w postaci obrazowej a także tablic liczb.	23388	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 konturowania.	23337	Python
8	Udoskonalenie oprogramowania przygotowanego na zajęciach przez przygotowanie funkcji zmieniającej obraz według przekształcenia opisanego przemieszczeniem trzech nie współliniowych punktów.	23391	Python
9	Udoskonalenie oprogramowania przygotowanego na zajęciach przez przygotowanie funkcji przycięcia (kadrowanie) obrazu według prostokąta niekoniecznie równoległego do osi X i Y wyznaczonego odpowiednim narzędziem wskazywania położenia.	23425	
10	Przeniesienie oprogramowania stworzonego na zajęciach tak, aby działał dla systemu operacyjnego iOS.		

11	Oprogramowanie to wykonania transformaty Hough'a	20668	Python
12	Oprogramowanie do nakładania zniekształceń geometrycznych na obrazy monochromatyczne i edytora doboru funkcji korygujących takie zniekształcenie.		
13	Udoskonalenie oprogramowania przygotowanego na zajęciach przez: dołożenie operacji wyliczenia transformaty odległościowej dla obrazu binarnego oraz implementację rozdzielania obiektów dotykających się z wykorzystaniem tej transformaty.		
14	Segmentacja obrazu monochromatycznego/binarnego zawierającego znaki specjalne i symbole oraz wyszukiwanie znaków o kształtach <u>wkleśłych</u> np.: ☒, ☐, ☐, ☐, ☐, ☐, ☐, itp.	18620	Jawa
15	Segmentacja z obrazów monochromatycznych/binarnych zawierających wybrane 5 typów symboli o różnym rozmiarze, ich rozpoznawania metodami klasycznymi lub opartymi o sieci neuronowe.	17677	Jawa
16	Segmentacja obrazów z wykorzystaniem klasteryzacji szarych odcieni.	20667	Python
17	Udoskonalenie oprogramowania przygotowanego na zajęciach przez implementację progowanie obrazu prawdopodobieństwa przypasowania do zadanej tekstury.		
18	Udoskonalenie oprogramowania przygotowanego na zajęciach przez implementację nowego narzędzie do tworzenia panoramy na postawie serii zdjęć.	9876	Python
19	Implementacja linii profilu i wycinanie nie będących prostokątem fragmentów z istniejących obrazów monochromatycznych.		
20	Udoskonalenie oprogramowania przygotowanego na zajęciach przez wykonanie narzędzia do ekstrakcja linii pionowych i poziomych za pomocą operacji morfologicznych.	042050	Skasowa 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: implementację operacji filtracji logicznych na obrazach binarnych; rozwinięcie możliwości wyświetlania i zapisywania jako obraz fragmentów obrazu powstałych na podstawie wskazanych, niekoniecznie spójnych, fragmentów histogramu.	21346	Python

23	Segmentacja obrazu monochromatycznego zawierających 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.		
25	Udoskonalenie oprogramowania przygotowanego na zajęciach przez implementację operacji przenikania dwóch obrazów monochromatycznych.	21296	C#
26	Udoskonalenie oprogramowania przygotowanego na zajęciach przez wykonanie narzędzia do rozciągania i zawężania histogramy z możliwym zastosowaniem funkcji liniowej lub funkcji gamma (analogicznie jak w Corelu PhotoPaint-cie).	21075	Python
27	Udoskonalenie oprogramowania przygotowanego na zajęciach przez dołożenie operacji otoczki wypukłej obiektu – operacja morfologii matematycznej	21918	C#
28	Segmentacja obrazu monochromatycznego/binarnego zawierającego znaki specjalne i symbole oraz wyszukiwanie znaków o kształtach <u>wypukłych</u> np.: ☉, ☪, ☐, ☐, ☐, ☐, ☐, itp.		
29	Udoskonalenie oprogramowania przygotowanego na zajęciach przez dołożenie 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: zaimplementowanie operacji rekonstrukcji morfologicznej.	21611	Python
31	Udoskonalenie oprogramowania przygotowanego na zajęciach przez dołożenie operacji zmniejszenia udziału szumu przez uśredniania kilku obrazów zebranych w takich samych warunkach oraz operacji logicznych na zasumowanych 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 przedstawiли 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 stosowalności i założeń 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. Zieliński, M. Strzelecki: Komputerowa analiza obrazu biomedycznego, PWN Warszawa-Lódź 2002, str. 178-214; segmentacja z wykorzystaniem analizy tekstur, mozaika Voronoi (Voronoi tessellation), 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://winntbg.bg.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