

Przedmiot - **Rozpoznawanie i przetwarzanie obrazów**
Temat - Rozpoznawanie/klasyfikacja obrazów
za pomocą głębokich sieci neuronowych

dr inż. Andrzej Burda
a.burda@vizja.pl

- ❑ **Rozpoznawanie obrazów** (*Image Recognition*) to wyodrębnianie konkretnych informacji z obrazów, np. tekstu, emocji czy gestów.
- ❑ Przykłady - motoryzacja - odczytywanie tablic rejestracyjnych, zagadnienia bezpieczeństwa - rozpoznawanie twarzy i emocji.
- ❑ **Przetwarzanie i modyfikacja obrazów** obejmuje swoim zakresem zmienianie lub ulepszanie obrazów.
- ❑ Rodzaje - *Super-resolution* - poprawianie rozdzielczości obrazów, *Style transfer* - nakładanie stylu jednego obrazu na drugi, *Inpainting* - wypełnianie brakujących fragmentów obrazu.
- ❑ Przykłady - usuwanie szumu z obrazów, tworzenie obrazów o artystycznym wyglądzie.
- ❑ **Generowanie obrazów** (*Image Generation*), na pozór podobne zagadnienie, ale ze względu na fakt, że obraz powstaje na podstawie opisu słownego, klasyfikowany jest jako zadanie **multimodalne**, ponieważ łączy w sobie dwa różne rodzaje danych - tekstowe (język naturalny) i wizualne (obrazy).

- ❑ Poprzedni wykład poświęcony był metodyce i technikom rozwiązywania zadania klasyfikacji przy pomocy sieci konwolucyjnych. Należy jednak pamiętać, że możliwości **CNN** nie ograniczają się tylko do tej klasy problemu.
- ❑ Reasumując rozważania poprzedniego wykładu, klasyfikacją w kontekście naszego kursu, **klasyfikacja obrazów** (*Image Classification*), to przypisywanie obrazu do jednej lub większej liczby kategorii.

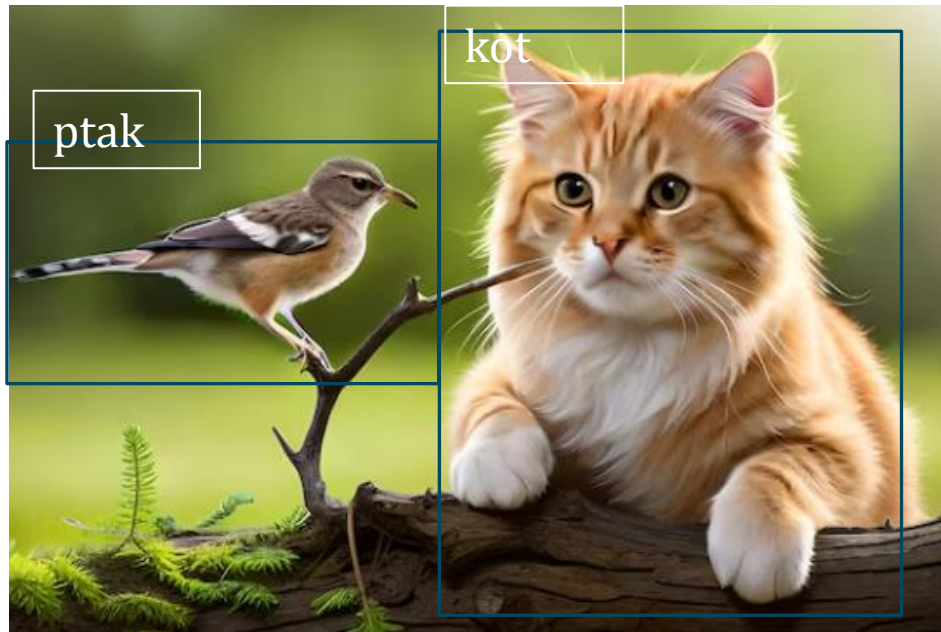


- kot
- pies
- ptak
- koń
- owca

- zabawa
- odpoczynek
- obserwacja
- zainteresowanie

- sztuka
- natura
- fotografia
- urbanistyka

- ❑ **Detekcja obiektów** (*Object Detection*) polega na lokalizowaniu i klasyfikowaniu obiektów występujących na obrazach za pomocą ramek ograniczających tzw. „bounding boxów” (*bounding boxes*).
- ❑ Zastosowania praktyczne to - w motoryzacji - wykrywanie pieszych, samochodów oraz znaków drogowych w systemach/samochodach autonomicznych, w logistyce - lokalizowanie produktów na półkach sklepów/magazynów.



- ❑ **Segmentacja obrazów** (*Image Segmentation*), to przyporządkowanie każdego piksela obrazu do konkretnej kategorii.
- ❑ Rodzaje - **Segmentacja semantyczna** - grupowanie pikseli należących do tej samej klasy (np. tło, samochód, pies) oraz **segmentacja instancji** - identyfikowanie każdej osobnej instancji obiektu (np. dwa psy jako osobne segmenty).



- ❑ Ta klasa problemu decyzyjnego ma bardzo szerokie zastosowanie w obszarze bezpieczeństwa państwa, medycynie, sporcie, czy sztuce.



- ❑ Ocena poprawności lokalizacji detekcji obiektów polega na porównaniu prognozowanej lokalizacji obiektów przez model z rzeczywistymi, ręcznie oznaczonymi lokalizacjami w zbiorze testowym.
- ❑ Najczęściej do tego celu wykorzystuje się miarę Jaccarda (**IoU** – *Intersection over Union*) -

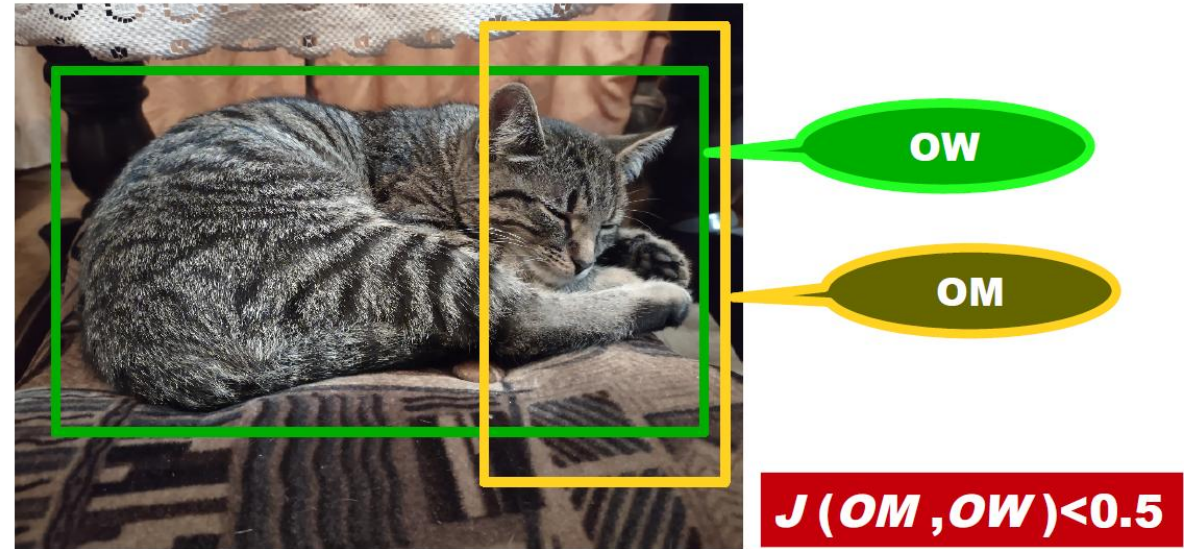
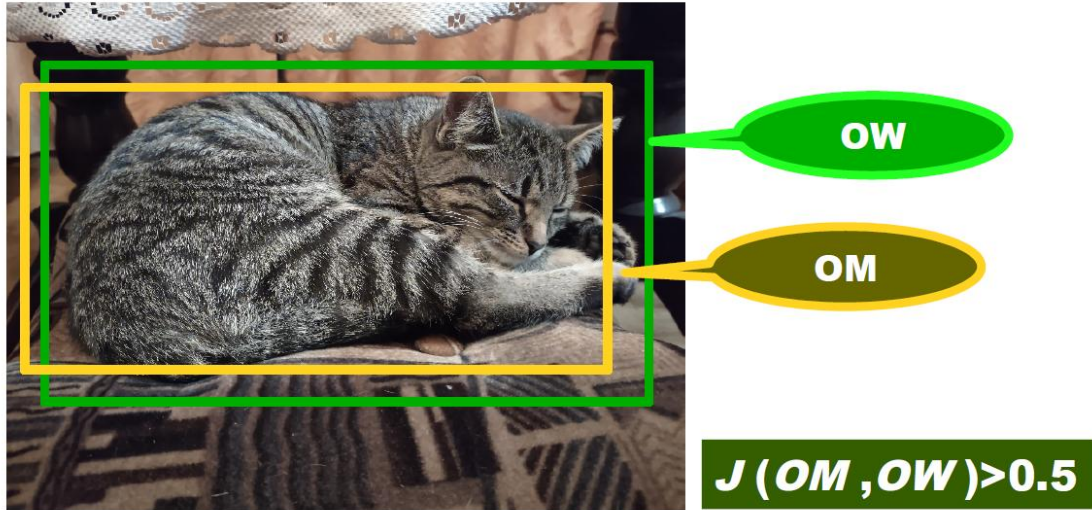
$$J(OM, OW) = \frac{|OM \cap OW|}{|OM \cup OW|}$$

OM – obrys obiektu wyznaczony przez **m**odel

OW – obrys **w**zorcowy

- ❑ Jest to stosunek powierzchni przecięcia dwóch prostokątnych obszarów (detekcji i rzeczywistej lokalizacji) do powierzchni ich unii (połączenie obu obszarów).
- ❑ $J(OM, OW) = 1$, gdy obrys OM pokrywa się idealnie z obrysem OW , $J(OM, OW) = 0$, gdy obrys OM jest całkowicie rozłączny z obrysem OW . Najczęściej przyjmuje się, że obiekt został poprawnie zlokalizowany na obrazie przez model, jeśli $J(OM, OW) > 0.5$.

Przykłady lokalizacji obiektów na obrazie



K. Pancerz - Zaawansowane metody sztucznej inteligencji, KUL, materiały pomocnicze do wykładu.

- ❑ Wstępne przetwarzanie obrazu – przygotowanie obrazu do analizy
- ❑ Ekstrakcja cech – wydobywanie istotnych cech obrazu
- ❑ Generowanie regionów propozycji **ROIs** (*Regions of Interest*)
– wybór obszarów, które mogą zawierać obiekty
- ❑ Klasyfikacja obiektów – przypisanie obiektowi odpowiedniej klasy
- ❑ Lokalizacja obiektów – wyznaczenie prostokąta otaczającego obiekt
- ❑ Post-processing NMS (*Non-maximum suppression*) – eliminowanie zbędnych detekcji
- ❑ Ocena wyników — mierzenie skuteczności detekcji (**IoU, Precision, Recall**)

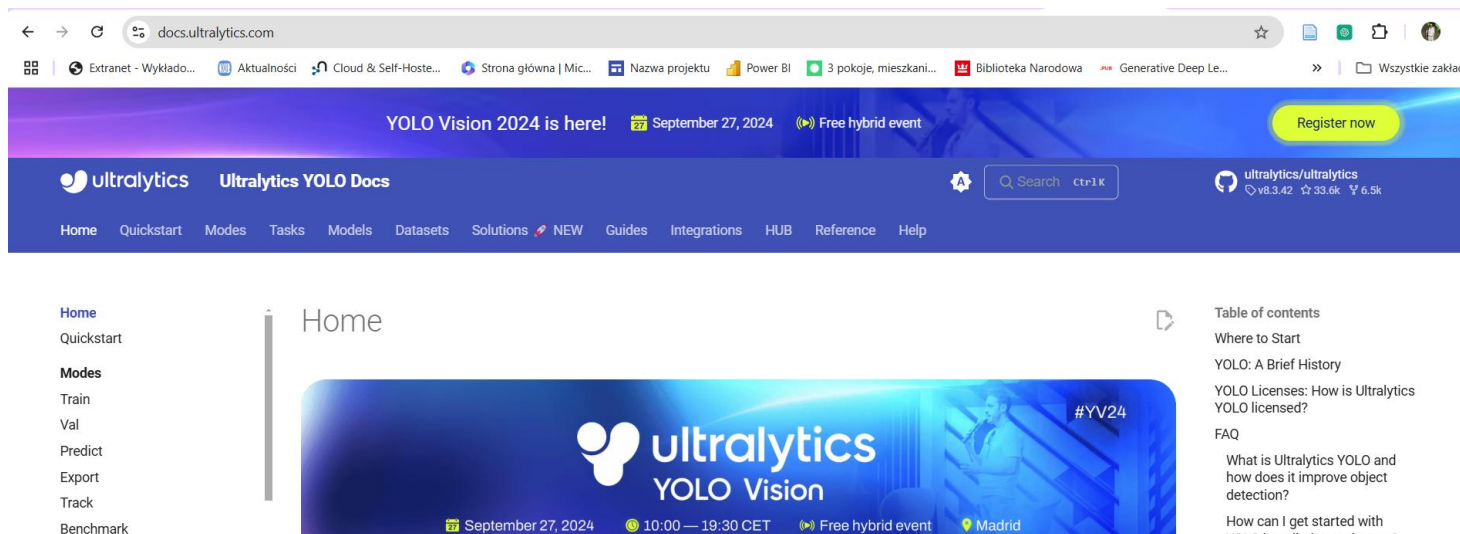
- ❑ **Inicjalizacja okna** - okno o określonym rozmiarze jest początkowo ustawione w lewym górnym rogu obrazu.
- ❑ **Przesuwanie okna** - okno przesuwana się po obrazie w poziomie i pionie, analizując różne fragmenty obrazu.
- ❑ **Kroki przesunięcia** - Okno może przesuwać się o jeden piksel lub większy krok (np. co drugi piksel), w zależności od wymagań.
- ❑ **Podział na regiony** - Każda pozycja okna generuje region propozycji (*region of interest* - ROI).
- ❑ **Analiza zawartości okna** - dla każdego regionu obrazu, który znajduje się w oknie, stosuje się klasyfikator, aby określić, czy obiekt jest obecny w tym regionie. Jeśli klasyfikator wykryje obiekt w danym regionie, zaznaczany jest jako detekcja.
- ❑ **Rejestracja wyników**: Wszystkie wykryte obiekty są zapisywane, a następnie mogą podlegać dalszemu procesowi, np. *post-processingowi*.

- ❑ Metoda okna przesuwne jest kosztowna obliczeniowo.
- ❑ W przypadku zastosowania warstw konwolucyjnych, operacje konwolucji powtarzane są dla każdego analizowanego podobszaru.
- ❑ **Detektory jednoetapowe** (bez propozycji **ROI**) – YOLO, SSD
Wykonywane jest jedno przejście przez obraz w celu detekcji i lokalizacji obiektów na obrazach, są **bardziej efektywne obliczeniowo**, są **mniej dokładne**, wykorzystywane są szczególnie **w aplikacjach czasu rzeczywistego**.
- ❑ **Detektory dwuetapowe** (z propozycją **ROI**) – R-CNN, Fast R-CNN, Faster R-CNN, R-FCN, Mask R-CNN
Wykonywane są dwa przejścia przez obraz w celu detekcji i lokalizacji obiektów na obrazach, w pierwszym przejściu generowane są propozycje obszarów **ROI**, w drugim przejściu obszary ROI są czyszczone i klasyfikowane, są **mniej efektywne obliczeniowo**, są **bardziej dokładne**, wykorzystywane są szczególnie **w aplikacjach wymagających większej dokładności**.

- ❑ YOLO (*You Only Look Once*) - jego nazwa oddaje kluczową cechę działania — model przetwarza obraz jednokrotnie, aby wykryć i zlokalizować obiekty na obrazie, co wyróżnia go od starszych metod detekcji obiektów, które wymagały wielokrotnego przeszukiwania obrazu. Dzięki temu YOLO jest znacznie szybszy i bardziej wydajny w czasie rzeczywistym, co jest kluczowe dla aplikacji, takich jak monitorowanie, autonomiczne pojazdy, czy drony.
- ❑ YOLO dzieli obraz wejściowy na siatkę, gdzie każda kratka ma za zadanie wykryć obiekty, które są w jej zakresie. Model generuje kilka (zwykle 2-3) propozycje *bounding boxów* dla każdego obszaru siatki, przewidując jednocześnie:
 - ❑ centrum obiektu (jego współrzędne w siatce),
 - ❑ szerokość i wysokość prostokąta,
 - ❑ pewność co do wykrycia obiektu w danej kratce,
 - ❑ klasę obiektu.

Przykład detekcji z użyciem modelu YOLO

- ❑ YOLO często generuje wiele nakładających się propozycji prostokątów dla tego samego obiektu. Aby usunąć nadmiarowe detekcje, stosowana jest technika *Non-Maximum Suppression*, która wybiera propozycję z najwyższym prawdopodobieństwem i odrzuca te nakładające się.
- ❑ YOLO trenowany na dużych zbiorach danych, takich jak *ImageNet*, może rozpoznawać tysiące kategorii obiektów ogólnego użytku, w tym zwierzęta, pojazdy, budynki, instrumenty muzyczne, sprzęt kuchenny i wiele innych.
- ❑ Model YOLOv8 został wydany przez firmę Ultralytics w styczniu 2023 roku.

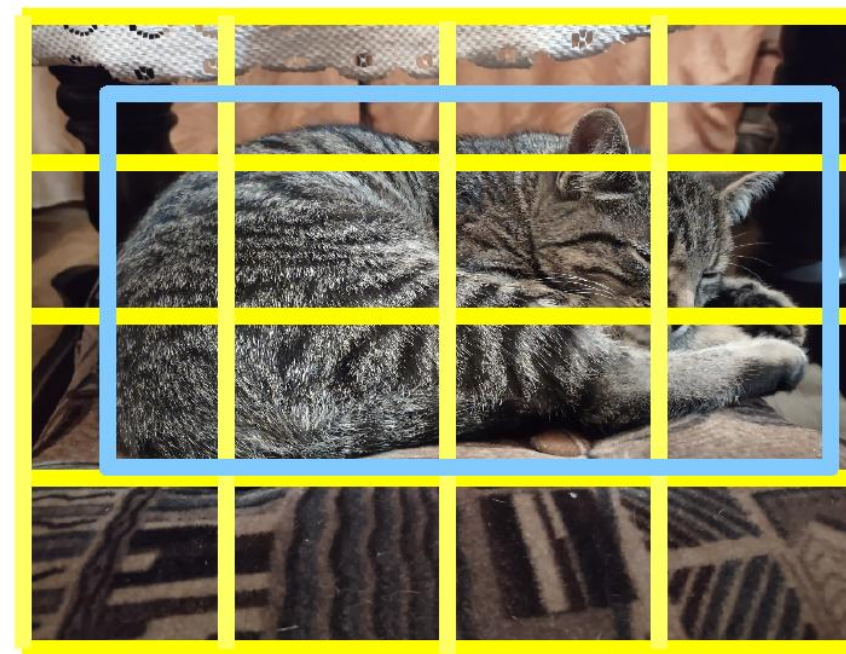


<https://docs.ultralytics.com/>

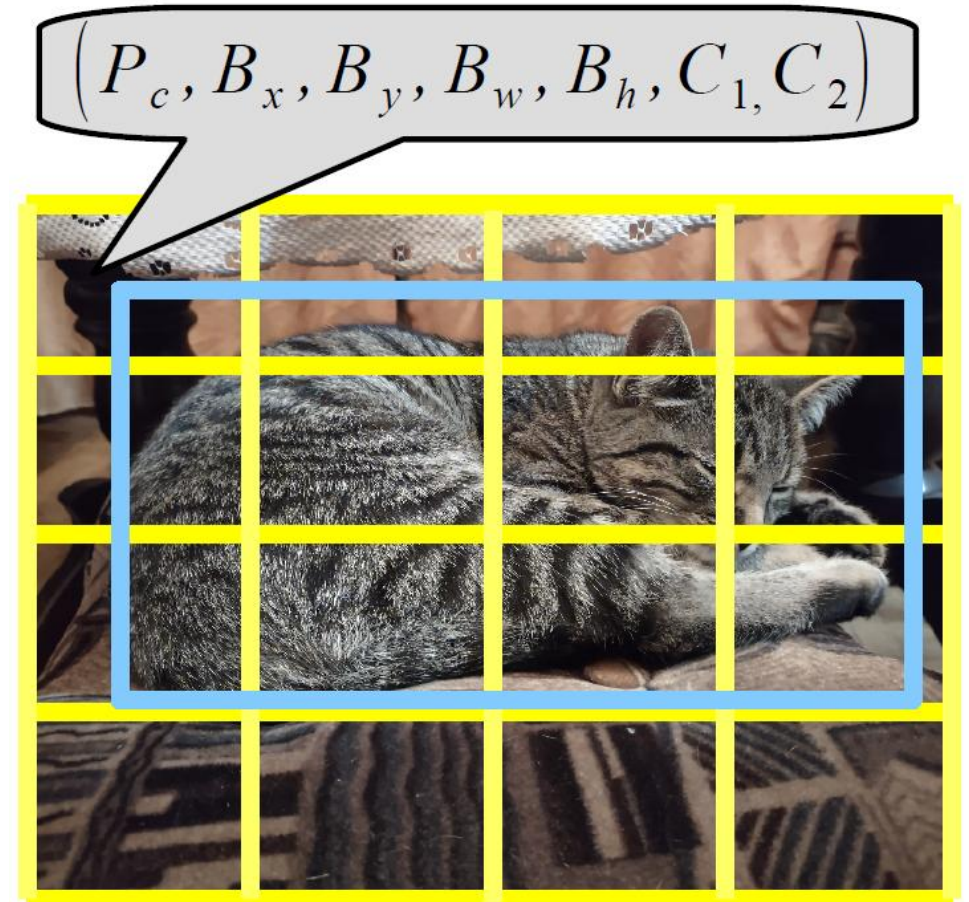
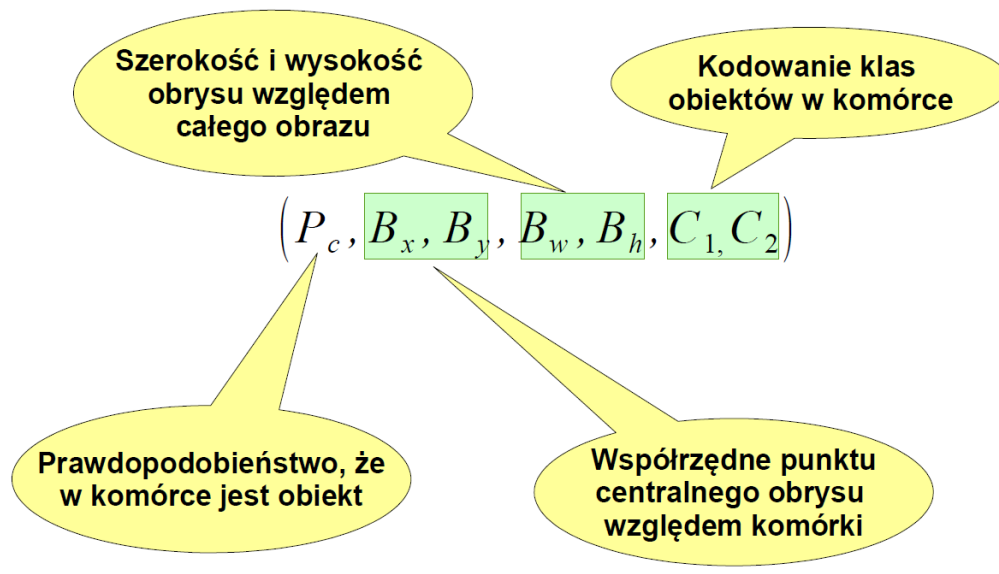
- ❑ Obrys otaczający (*bounding box*) charakteryzowany jest przez cztery wartości:
 - ❑ (B_x, B_y) – współrzędne centralnego punktu obrysu,
 - ❑ B_w – szerokość obrysu,
 - ❑ B_h – wysokość obrysu.
- ❑ YOLO dzieli obraz na siatkę składającą się z $S \times S$ komórek, gdzie $S \times S$ zależy od architektury i poziomu przeskalowania obrazu wejściowego.
- ❑ Jeśli wejściowy obraz ma rozdzielczość 416×416 , a YOLO używa siatki 13×13 , każda komórka siatki reprezentuje obszar 32×32 pikseli (zakładając podział na równe części).

Podział obrazu siatką o rozmiarze $S \times S$:

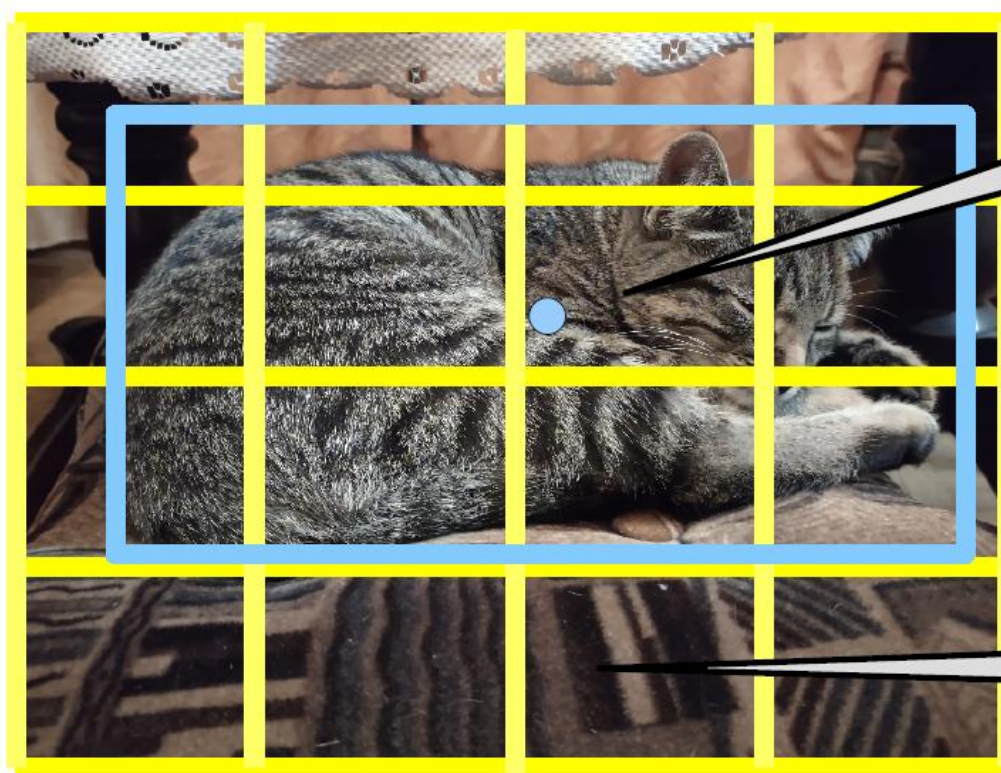
$S=4$



- ❑ Obrys otaczający (*bounding box*) charakteryzowany jest przez cztery wartości:
 - ❑ (B_x, B_y) – współrzędne centralnego punktu obrysu,
 - ❑ B_w – szerokość obrysu,
 - ❑ B_h – wysokość obrysu.
- ❑ Każda komórka siatki etykietowana jest co najmniej 7 wartościami:



- Model jest uczony w taki sposób, że każda komórka siatki odpowiada za dokładne przewidywanie informacji tylko dla jednego obiektu (najczęściej tego, którego środek masy znajduje się w jej obszarze). Pozostałe komórki siatki uczą się ignorować dany obiekt.



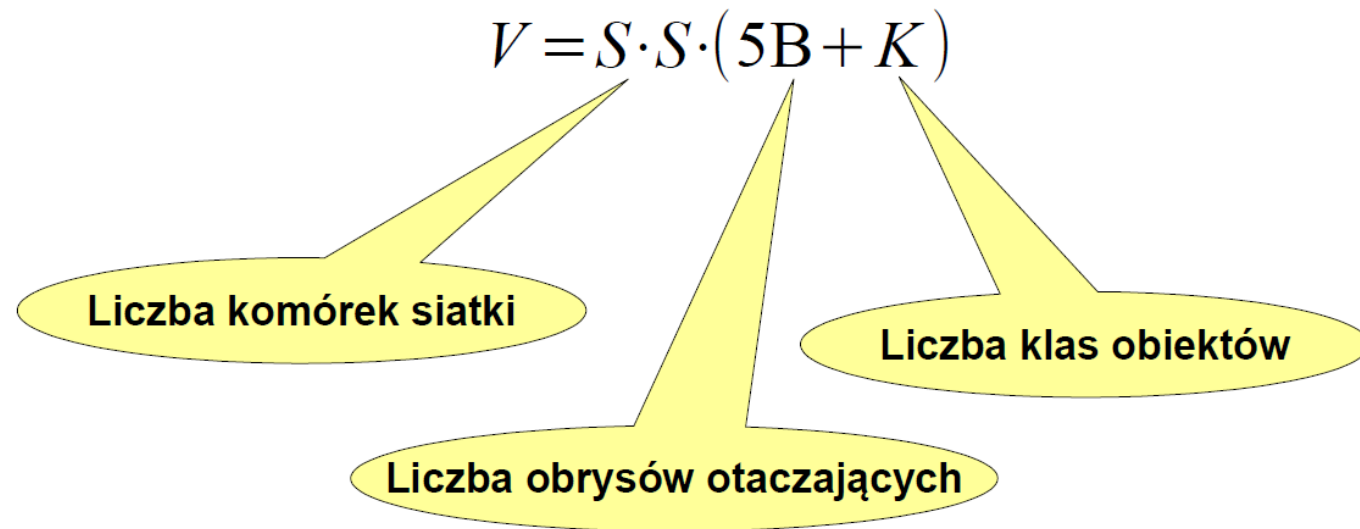
$(1, 0.05, 0.2, 0.85, 0.7, 1, 0)$

C_1, C_2, \dots, C_n przyjmują wartości w przedziale $[0,1]$, a ich suma dla wszystkich klas powinna wynosić 1 (jeśli traktujemy każdą klasę jako niezależną).

Obiekt na pewno należy do klasy C_1 i na pewno nie należy do klasy C_2 .

$(0, 0, 0, 0, 0, 0, 0)$

- Całkowita liczba wartości do predykcji:





```
# Montowanie dysku  
from google.colab import drive  
drive.mount('/content/drive', force_remount=True)
```



```
# Klonowanie repozytorium YOLO  
!git clone https://github.com/ultralytics/ultralytics
```



```
import os  
  
# Sprawdzenie bieżącego katalogu roboczego  
print("Obecny katalog:", os.getcwd())
```

```
# Zmiana katalogu roboczego  
os.chdir('/content/ultralytics/')  
print("Obecny katalog:", os.getcwd())
```

```
➔ Obecny katalog: /content/ultralytics
```

```
# Instalowanie zależności  
!pip install -qe . # e- instaluje pakiet w trybie „edytowalnym” q- tryb "cichy"
```

```
# Import przydatnych bibliotek  
from ultralytics import utils  
from matplotlib import pyplot as plt  
from matplotlib import image
```

```
[46] yolo_detect=YOLO('yolov8n.pt')
      results_detect = yolo_detect('/content/Puzzle.jpg', conf=0.25, iou=0.7)
      results_detect[0].save(filename="Obraz_detect.jpg")
```



```
# Sprawdzenie wyników detekcji
for result in results_detect:
    for box in result.boxes: # Przechodzi przez wykryte obiekty
        class_id = box.cls.item() # Klasa
        confidence = box.conf.item() # Prawdopodobieństwo
        bbox = box.xyxy.cpu().numpy() # Współrzędne bounding boxa

        # Wyświetlenie informacji [x_min, y_min, x_max, y_max]
        print(f"Etykieta: {result.names[int(class_id)]}, Prawdopodobieństwo: {confidence:.2f}, Bounding Box: {bbox}")
```



```
Etykieta: dog, Prawdopodobieństwo: 0.73, Bounding Box: [[ 709.56  104.68 1543.5 1336.4]]
Etykieta: dog, Prawdopodobieństwo: 0.56, Bounding Box: [[ 32.646 117.01 835.05 1404]]
Etykieta: dog, Prawdopodobieństwo: 0.53, Bounding Box: [[1274.8 184.7 1977.6 1393.6]]
Etykieta: dog, Prawdopodobieństwo: 0.49, Bounding Box: [[ 461.3 983.19 1032.2 1396.5]]
Etykieta: dog, Prawdopodobieństwo: 0.30, Bounding Box: [[ 713.19 702.71 1533 1370.3]]
```




```
[49] # Liczba wartości do predykcji  
      S=13  
      B=5  
      K=2  
      v=S*S*(5*B+K)  
      v
```

⇒ 4563

Przykład detekcji i lokalizacji



```
[72] yolo_segment = YOLO('yolov8n-seg.pt')
      results_segment = yolo_segment('/content/Puzzle.jpg')
      results_segment[0].save(filename="Obraz_segment.jpg")
```

```
# Wyświetl maski segmentacji i odpowiadające im klasy
for result in results_segment:
    # Uzyskujemy klasy wykrytych obiektów
    classes = result.boxes.cls.cpu().numpy() # Pobiera identyfikatory klas
    names = result.names # Nazwy klas (np. osoba, samochód itp.)

    # Iterujemy przez maski i klasy
    for i, mask_data in enumerate(result.masks.data): # Przechodzimy przez maski
        class_id = int(classes[i]) # Identyfikator klasy dla danej maski
        class_name = names[class_id] # Nazwa klasy na podstawie identyfikatora

        print(f"Etykieta: {class_name}")

# Wyświetlenie obrazu z segmentacją
image_with_segmentation = results_segment[0].plot()
plt.imshow(image_with_segmentation)
plt.axis('off')
plt.show()
```

```
...2ms
... at shape (1, 3, 480, 640)
```

Wynik segmentacji i detekcji



Etykieta: dog
Etykieta: cat
Etykieta: dog
Etykieta: dog
Etykieta: dog
Etykieta: teddy bear




```
[77] yolo_pose = YOLO('yolov8n-pose.pt')  
     results_pose = yolo_pose('/content/sport.jpg')  
     results_pose[0].save(filename="Obraz_pose.jpg")
```

```
▶ # Narysowanie wykrytych pozycji na obrazie  
  image_with_pose = results_pose[0].plot() # Rysuje kluczowe punkty na obrazie  
  
  # Wyświetlenie obrazu z wykrytymi pozycjami  
  plt.imshow(image_with_pose)  
  plt.axis('off') # Wyłączenie osi  
  plt.show()
```







```
# Inicjalizacja modelu YOLOv8 z plikiem wag
yolo_classify = YOLO('yolov8n-cls.pt')

# Użycie modelu do klasyfikacji obrazu
results_classify = yolo_classify('/content/drive/MyDrive/KUL/ZMSI/Lab_6/Obrazy/muzyka_renesans.jpg')

# Zapisanie na dysku obrazu z rezultatem klasyfikacji
results_classify[0].save(filename="Obraz_classify.jpg")

# Wyświetlenie obrazu z rezultatem klasyfikacji
results_classify[0].show()
```

 `print(yolo_classify)`

 `YOLO(
 (model): ClassificationModel(
 (model): Sequential(
 (0): Conv(
 (conv): Conv2d(3, 16, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
 (act): SiLU(inplace=True)
)
 (1): Conv(
 (conv): Conv2d(16, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
 (act): SiLU(inplace=True)
)
 (2): C2f(
 (cv1): Conv(
 (conv): Conv2d(32, 32, kernel_size=(1, 1), stride=(1, 1))
 (act): SiLU(inplace=True)
)
 (cv2): Conv(
 (conv): Conv2d(48, 32, kernel_size=(1, 1), stride=(1, 1))
 (act): SiLU(inplace=True)
)
 (m): ModuleList(
 (0): Bottleneck(
 (cv1): Conv(`



Modele **YOLO** można dotrenować na nowych danych, dostosowując je do specyficznych zastosowań lub niestandardowych zbiorów danych. Proces dotrenowania modelu YOLO nazywa się ***fine-tuningiem*** lub ***transfer learningiem***.

Jest to często stosowana metoda, szczególnie gdy nie dysponujemy wystarczającą ilością danych, aby trenować model od podstaw.



```
# Sprawdzenie wyników klasyfikacji
for result in results_classify:
    predicted_class = result.probs.top1 # Klasa o najwyższym prawdopodobieństwie
    confidence = result.probs.top1conf # Wartość prawdopodobieństwa

    # Wyświetlenie etykiety klasy oraz prawdopodobieństwa
    print(f"Etykieta: {result.names[predicted_class]}")
    print(f"Prawdopodobieństwo: {confidence:.2f}")
```



```
Etykieta: harp
Prawdopodobieństwo: 0.91
```


- ❑ Każda klasa problemów decyzyjnych wymaga innego etykietowania obrazów/przykładów uczących.
- ❑ W przypadku klasyfikacji etykietowanie jest proste. Przypisujemy obrazowi jedną bądź więcej etykiet tekstowych lub numerycznych.
- ❑ W pozostałych przypadkach problem ten jest bardziej skomplikowany.
- ❑ Istnieją gotowe rozwiązania, którymi można się wspomagać przy tworzeniu dużych zbiorów etykietowanych obrazów. Jednym z nich jest Roboflow.

