MSI - Projekt
(Semestr 24L)

Kryteria zaliczania

Projekt jest rozliczany na podstawie:

- raportu wstępnego (0 - 5 pkt.) i
- rozliczenia końcowego (0 - 25 pkt.)

Projekt rozliczany jest poprzez stronę przedmiotu MSI (realizacja 24L) na platformie dydaktycznej leia.pw.edu.pl – poprzez zakładki typu „Zadanie": „Raport wstępny", „Projekt końcowy".

W **raporcie wstępnym** należy:

A. Przedstawić zadaną **metodykę** lub **algorytm** Sztucznej Inteligencji (w tym podać szczegółowy pseudokod algorytmu i skomentować go własnymi słowami); (0-2 p.)

B. Przedstawić **koncepcję rozwiązania** zadanego problemu z wykorzystaniem zadanej metodyki lub algorytmu Sztucznej Inteligencji, tzn. wyrazić zadany problem w sposób wymagany przez zadaną metodykę Szt. Int. (wymagana reprezentacja problemu, wymagane funkcje interfejsu pomiędzy metodą a reprezentacją problemu - w tym wyróżnić w pseudo-kodzie metody taki interfejs) i omówić przewidywany cel i sposób weryfikacji poprawności proponowanego rozwiązania (0-3p.).

W **rozliczeniu końcowym** projektu należy:

C. Przekazać **raport końcowy** (0-6 p.) zawierający:

- opis struktury i głównych funkcji zrealizowanego programu, w tym opis implementacji zadanej metodyki Szt. Int.
- sposób uruchomienia programu i korzystania z niego,
- podsumowanie weryfikacji poprawnego działania programu na kilku przypadkach danych i przedstawienie wniosków.

D. Przekazać **program** (0-12p.) i jego **źródła** (0-5p.) (kody programu) oraz **plik monitorujący** (0-2p.) wykonanie programu z przykładami wykonania.

Wymagania wobec programu:

1) W programie należy wyróżnić trzy moduły – **Metoda**, **Problem**, **Sterowanie**.

   Metoda stanowi implementację zadanej metodyki Sztucznej Inteligencji wykonaną w sposób uniwersalny - niezależnie od dziedziny zastosowania (konkretnego problemu).

   Problem stanowi implementację problemu wyrażoną w sposób wymagany przez metodykę Sztucznej Inteligencji.

   Sterowanie zawiera punkt wejścia do programu, steruje przebiegami wykonania programu i zawiera zwykłe tekstowe menu użytkownika umożliwiające weryfikację zakresu działania programu.

2) Program **nie powinien zawierać** interfejsu graficznego użytkownika ani żadnych dodatkowych bibliotek, które nie są dostępne Prowadzącemu, w tym jego wykonanie nie powinno wymagać oprogramowania dedykowanej karty graficznej.

3) W typowym rozwiązaniu program posiada możliwość zadawania argumentów, wczytuje dane z przygotowanych plików i zapisuje uzyskiwane wyniki w pliku monitorującym jego wykonanie a także w terminalu tekstowym programu.

4) Rozmiar rozliczenia projektu końcowego ograniczony jest do 20 MB.
5) Implementacja programu powinna być wykonana w znanym uniwersalnym języku programowania: C/C++, C#, Java, Matlab lub Python, a program wykonywalny (potrzebny dla języków C/C++, C# i Java) (exe, jar) powinien działać w MS Windows 32- lub 64-bit.

Wybór tematu projektu

Wybór tematu odbywa się na „forum wyboru projektu" utworzonym na stronie przedmiotu. Forum będzie dostępne od terminu podanego w harmonogramie zajęć.

Wybór tematu polega na udzieleniu odpowiedzi w wątku danego tematu, utworzonym przez Prowadzącego.

Jedna osoba realizuje jeden temat. Jeden temat może być obierany najwyżej dwukrotnie, pod warunkiem różnych języków implementacji – przy wybieraniu tematu należy określić język jego implementacji. Ewentualne kolizje wyboru tematu rozstrzygane są według zasady „*kto pierwszy - ten lepszy*.

Podczas wyboru tematu należy sprawdzić, czy wpis z naszą odpowiedzią może zostać zaakceptowany przez Prowadzącego. Jeśli zostaliśmy wyprzedzeni przez wpisy co najmniej dwóch innych osób, należy wybrać inny, jeszcze wolny temat. Temat jest uznany za przydzielony po wybraniu go przez Wykonawcę i akceptacji przez Prowadzącego.

Tematy (30)

- **Część W (wnioskowanie)**
  Tematy W2 – W3
- **Część P (przeszukiwanie przestrzeni stanów)**
  Tematy P3 – P8
- **Część S (poszukiwanie rozwiązania)**
  Tematy S2 – S5
- **Część L (planowanie)**
  Tematy L2 – L3
- **Część U (uczenie przez indukcję)**
  Tematy U1 – U9
- **Część N (sieci neuronowe)**
  Tematy N4 – N10

Opisy tematów

Część W (wnioskowanie)

Projekt W2. Rachunek sytuacyjny i wnioskowanie wstecz.

Wykonać program realizujący symulację środowiska i działanie autonomicznego robota mobilnego (agenta) na piętrze nieznanego budynku o wymiarach N × N „komórek". Celem działania agenta jest znalezienie „zagrożenia" (np. bomby), jego podniesienie i wywiezienie przez zadane wyjście.

- Problem: Wykonać symulator środowiska odpowiedzialny za dostarczanie obserwacji agentowi i za śledzenie wykonywanych przez niego akcji. Agent „widzi" ścianę w sąsiadującej z nim kratce i powinien jej unikać, gdy wiezie bombę, a ewentualny wjazd w nią powoduje rozbicie ściany (i eksplozję ewentualnej bomby) ale także „unieruchamia" agenta na pewien czas. Agent „czuje" zagrożenie (bombę) na odległość do 2 komórek, jeśli w prostej linii drogi do niego nie ma ściany. Wynik działania agenta ma postać sumarycznego kosztu sekwencji wykonanych akcji ważonych czasami ich wykonania.
- Metoda: Wyrazić wiedzę agenta w **rachunku sytuacyjnym** języka predykatów (można przyjąć własny uproszczony zapis formuł).

  Wykonać i zastosować algorytm **wnioskowania wstecz** (odpowiedni do przyjętej uproszczonej postaci formuł) w celu wnioskowania o nowych faktach i wnioskowania o wyborze akcji.
- Sterowanie: Sprawdzić działanie programu dla różnych *instancji* środowiska.

  Przedstawić wyniki szeregu prób i podać wnioski.

Projekt W3. Logika predykatów. Wnioskowanie w przód.

Zrealizować program demonstrujący działanie agenta reaktywnego ze stanem, którego zadaniem jest przejazd **korytarzem** i dotarcie do zadanych drzwi, połączone z omijaniem przeszkód. Jeśli nie można objechać przeszkody, może ona w ostateczności być usuwana z drogi.

- Problem: Program ma generować losowo aktualne środowisko, czyli rozmieszczenie przeszkód i drzwi, symulować ruchy przeszkód oraz pokazywać wykonane kroki wnioskowania i akcje agenta. Można przyjąć dyskretną postać środowiska w postaci macierzy prostokątnych komórek-stanów oraz skończoną liczbę możliwych akcji agenta.
- Metoda: Realizacja agenta ma postać systemu z **bazą wiedzy** wyrażoną w **logice predykatów** i zaopatrzoną w mechanizm **wnioskowania w przód.**

  Wyrazić wiedzę agenta w logice predykatów (można przyjąć własny uproszczony zapis tekstowy formuł).

  Wykonać i zastosować algorytm wnioskowania w przód (odpowiedni do przyjętej uproszczonej postaci formuł) w celu wnioskowania o nowych faktach i wnioskowania o wyborze akcji.
- Sterowanie: Zweryfikować działanie programu wykonując szereg prób i podając charakterystyki takie, jak: stopień nadmiarowości zrealizowanej ścieżki, częstość zderzeń, itp.

W raporcie przedstawić też wyniki szeregu prób działania agenta i podać wnioski.

Część P (przeszukiwanie przestrzeni stanów)

Projekt P3

Wykonać program rozwiązujący problem komiwojażera dla N miast algorytmem **przeszukiwania poinformowanego A*** stosując dwie różne nietrywialne heurystyki kosztów resztkowych.

a) Problem: Dla zadawanej wartości parametru N wykonać losowy generator przestrzeni stanów (graf połączeń N miast).

   Zdefiniować funkcje A* zależne od problemu (funkcje kosztu, generacji następników).

   Zdefiniować i zaimplementować funkcje wyznaczania dwóch nietrywialnych heurystyk kosztów resztkowych (w oparciu o graf połączeń), które są poprawne dla algorytmu przeszukiwania A*.

b) Metoda: Zaimplementować ogólną strategię A* przeszukiwania przestrzeni stanów problemu.

c) Sterowanie: Sprawdzić wykonanie programu dla różnej liczby miast N i obu heurystyk (wartości heurystyk, liczba generowanych i liczba wizytowanych węzłów drzewa przeszukiwania dla każdej heurystyki, znaleziona ścieżka i jej koszt).

   Przedstawić wyniki i wnioski.

Projekt P4. Przeszukiwanie A* i IDA*. Problem „przejścia przez labirynt 3D".

Wykonać program dla rozwiązania problemu „przejścia przez labirynt 3D**"** stosując zamiennie algorytmy: „przeszukiwanie **A*** i **IDA*"**. Problem jest ułatwiony faktem znajomości współrzędnych komórek początku i celu ścieżki.

Zrealizować zagadnienia:

a) Metoda: Zaimplementować obie zadane strategie przeszukiwania przestrzeni stanów problemu.

b) Problem: Umożliwić użytkownikowi losowe generowanie 3-wymiarowego labiryntu o parametrach zadawanych przez użytkownika (rozmiarze „n x n x N") i generować odpowiadającą labiryntowi przestrzeń problemu w postaci zbioru stanów i ich heurystycznej oceny. W labiryncie losowo występują komórki zajęte przez ścianki. Pojedyncza akcja polega na przejściu z aktualnej komórki do jednej z jej potencjalnych sąsiadów. Koszt akcji odpowiada odległości między komórkami. Jeśli wybrana komórka jest zajęta to koszt jest wyższy, gdyż dodatkowo akcja obejmuje wtedy przebicie dziury w ściance.

c) Sterowanie: Wykonać testowanie działania programu. Porównać wyniki uzyskane przez obie strategie dla różnych liczb „n", „N" i dla różnych heurystyk.

Przedstawiać podstawowe charakterystyki dotyczące poszukiwań (ilość sukcesów, średnia długość ścieżki, ilość rozwijanych węzłów). Sformułować wnioski.

Projekt P5. Przeszukiwanie RTA* (wersja +). Nawigacja robota mobilnego.

Wykonać program realizujący nawigację **robota mobilnego** (symulując mapę 2D budynku) stosując metodą **przeszukiwania RTA* (wersja +).** Zrealizować zagadnienia:

a) Problem: Mapa 2D ma charakter prostokąta o M x N komórkach. Komórki są zaznaczone jako wolne lub zajęte (przeszkody). Jedna komórka jest miejscem początkowym, a inna – miejscem końcowym ścieżki. Robot w zasadzie nie powinien „wjechać" na przeszkodę, ale może to zrobić „płacąc" wyższym kosztem akcji. Założyć,

że minimalny zbiór ruchów robota to:

[jazda na wprost o wielokrotność „a=1” jednostki kratki (trzy akcje: a, 2a, -a), obrót w lewo lub w prawo o wielokrotność „b”= 45 stopni kątowych (trzy akcje: b, 2b, -b)].

b) Metoda: Zaimplementować strategię przeszukiwania RTA* przestrzeni stanów problemu, oddzielając część niezależną od problemu od funkcji zależnych od problemu.

c) Sterowanie: Umożliwić użytkownikowi zdefiniowanie problemu w postaci mapy 2D i parametrów zbioru akcji zbioru stanów, zbioru akcji, kosztów akcji i warunku stopu.

   Podawać rozwijaną ścieżkę i drzewo przeszukiwania (decyzyjnego) - długość i koszt ścieżki, liczbę węzłów danego poziomu drzewa.

   Sprawdzić działanie programu. Przyjąć, że liczba przeszkód wynosi od 20% do 50% liczby komórek – zbadać rozwiązania przy różnej stopie przeszkód z tego zakresu.

Porównać wyniki przeszukiwania w zależności od wartości parametrów. Przedstawić podstawową charakterystykę tych wyników (długość i koszt ścieżki; liczba rozwijanych węzłów; częstość uzyskania sukcesu, czyli optymalnej ścieżki).

Projekt P6. Przeszukiwanie z heurystyką o globalnym lub lokalnym horyzoncie.

Wykonać program porównujący ze sobą rozwiązania problemu znalezienia rozwiązania - komórki mapy 2D o **maksymalnej** wartości funkcji 2 zmiennych, z = f(*x,y*), algorytmami przeszukiwania **zachłannego z heurystyką** i **przez wspinanie („hill climbing”)**. Dziedzina funkcji ma postać dyskretnej mapy 2D, zdefiniowanej dla przedziałów wartości *x* i *y*, o zadanym rozmiarze komórki, a wartość funkcji jest etykietą każdej komórki w takiej mapie.

- Problem: Program umożliwia losowe generowanie różnych funkcji o znanym typie (np. sinusoida, funkcja wielomianowa) – losowe są jej parametry - o kilku maksimach lokalnych i losowe generowanie pozycji początkowej. Zdefiniowanie heurystyki wymaga znajomości wartości maksymalnej dla zadanej funkcji – automatycznie wyznaczać tę wartość dla konkretnej funkcji i przyjąć jej wartość za znaną w procesie przeszukiwania przestrzeni stanów dla problemu.
- Metoda: Zaimplementować oba algorytmy przeszukiwania oddzielając główną funkcję - część niezależną od problemu – od funkcji zależnych od problemu.
- Sterowanie: utworzyć proste menu tekstowe umożliwiające Użytkownikowi zadawanie parametrów testowania i zapisywania wyników.
- Sterowanie: Sprawdzić wykonanie programu dla obu algorytmów przeszukiwania dla kilku różnych funkcji i rozmiarów mapy 2D. Wynik ma postać wygenerowanej mapy 2D środowiska i sekwencji wizytowanych komórek.

Przedstawić zbiorczą charakterystykę uzyskanych wyników i sformułować wnioski.

Projekt P7. Przeszukiwanie ślepe i z heurystyką.

Wykonać program porównujący ze sobą rozwiązania **problemu poszukiwania ścieżki robota mobilnego** w przestrzeni o wymiarze N × N „komórek” algorytmami: a) **ślepego przeszukiwania z jednorodną** funkcja kosztu i b) **przeszukiwania zachłannego z heurystyką („najbliższy celowi najpierw”)**. Komórki mogą zawierać ścianki przez które nie można przejść – przyjąć liczbę ścianek z przedziału 20% – 30% liczby komórek.

- Metoda: Zaimplementować obie wymagane strategie przeszukiwania przestrzeni stanów.
- Problem: Wykonać losowy generator środowiska (w tym każdorazowo generuje on stan początkowy i cel). Przyjąć, że akcja polega na przejściu do jednej z 8 możliwych sąsiednich komórek, pod warunkiem, że komórka nie zawiera ścianki.

Zdefiniować funkcje strategii przeszukiwania zależne od problemu (od konkretnej mapy 2D i zbioru możliwych akcji robota). W szczególności zdefiniować dwie różne i nietrywialne heurystyki, dopuszczalne (poprawne) dla strategii przeszukiwania poinformowanego (dla strategii zachłannej z heurystyką).

- Sterowanie: Wykonać menu tekstowe umożliwiające generowanie lub wczytywanie danych o środowisku i parametrów wykonania. Sprawdzić wykonanie programu dla obu heurystyk i różnych rozmiarów środowiska. Przedstawić statystykę wyników i wnioski.

Projekt P8. Dwie wersje przeszukiwania RTA*.

Wykonać program realizujący poszukiwanie najlepszej ścieżki rozwiązania względnie najlepszego rozwiązania dla przejścia po (mapie labiryntu), stosując alternatywnie dwie wersje metody **przeszukiwania RTA*.** Wersje różnią się znakiem (+ lub -) stosowanym dla uwzględnienia kosztu powrotu z aktualnego węzła do węzła rodzica. („+" daje najlepsze rozwiązanie, „-„ najlepszą ścieżkę).

Mapa 2D ma charakter prostokąta o M x N komórkach. Komórki są zaznaczone jako wolne lub zajęte (przeszkody). Jedna komórka jest miejscem początkowym, a inna – miejscem końcowym ścieżki. Robot nie może „wjechać" na przeszkodę.

Komórka początkowa „losowana" jest przy ścianie „zachodniej" lub „północnej", a komórka końcowa – przy ścianie południowej lub wschodniej. Rozmiar N jest agentowi znany.

Oczywiście zdefiniować należy też sensowną heurystykę. Założyć, że akcje agenta pozwalają na przejście do jednej z kratek sąsiednich aktualnej kratki.

Zrealizować zagadnienia:

a) Metoda: Zaimplementować obie wersje strategii przeszukiwania RTA* przestrzeni stanów problemu, oddzielając część niezależną od problemu od funkcji zależnych od problemu.

b) Problem: Umożliwić użytkownikowi zdefiniowanie problemu w postaci mapy 2D (zbioru stanów i kosztów akcji).

c) Sterowanie: Sprawdzić działanie programu na różnych danych o problemie i parametrach wykonania. Umożliwić użytkownikowi sterowanie programem poprzez menu tekstowe. Przyjąć, że liczba przeszkód wynosi od 20% do 50% liczby komórek – zbadać oba rozwiązania przy różnej stopie przeszkód z tego zakresu.

   Podawać rozwijaną ścieżkę i drzewo przeszukiwania (decyzyjnego) - długość i koszt ścieżki, liczbę węzłów danego poziomu drzewa.

Porównać wyniki obu algorytmów przeszukiwania pod względem kosztu ścieżki i liczby wykonanych akcji oraz liczby rozwijanych węzłów drzewa decyzyjnego.

Przedstawić podstawowe charakterystyki tych wyników (długość i koszt ścieżki, liczba rozwijanych węzłów, optymalność wyniku).

Część S (poszukiwanie rozwiązania)

Projekt S2

Wykonać program umożliwiający testowanie algorytmu przeszukiwania drzewa Mini-Maks w wersji „**Mini-Maks z cięciami alfa-beta**" w grze **przypominającej grę w warcaby** (tzn. warcaby ograniczone w ten sposób, że pionek zamieniany na damkę zdobywa punkt i znika z

gry) o sparametryzowanym rozmiarze planszy N × N (i odpowiedniej liczbie pionków).

a) Metoda: Zaimplementować zadaną strategię „Mini-Maks z cięciami alfa-beta" poszukiwania optymalnego ruchu, oddzielając część niezależną od problemu od funkcji zależnych od problemu.

b) Problem: Wykonać symulator gry – nadzorowania stanu planszy, wykonywania ruchów własnych i przeciwnika. W szczególności symulować alternatywne dwie kategorie gracza przeciwnego (poprzez wybieranie dla niego suboptymalnych lub przypadkowych akcji).

c) Sterowanie: Sprawdzić wykonanie programu dla różnych rozmiarów N (liczba wykonywanych ruchów, rozmiar drzewa Mini-Max, rezultat gry).

d) Podsumować uzyskane wyniki i przedstawić wnioski.

Projekt S3

Wykonać program umożliwiający testowanie algorytmu **„stochastycznego obciętego Mini-Maksu"** w grze **przypominającej grę w warcaby** (tzn. warcaby ograniczone w ten sposób, że pionek zamieniany na damkę zdobywa punkt i znika z gry) o sparametryzowanym rozmiarze planszy N × N (i odpowiedniej liczbie pionków). Do standardowego algorytmu „obciętego Mini-Maksu" należy wprowadzić stochastyczny element polegający na losowym wybieraniu na poziomie drzewa K (odpowiadającego progowi obcięcia) P-procentowej populacji węzłów, dla których rozwijane są ścieżki drzewa o długości 2K zamiast K.

a) Metoda: Zaimplementować zadaną strategię poszukiwania optymalnej akcji, oddzielając część niezależną od problemu od funkcji zależnych od problemu.

a) Problem: Wykonać symulator gry – nadzorowania stanu planszy, wykonywania ruchów własnych i przeciwnika. Symulować alternatywne dwie kategorie gracza przeciwnego (przez uruchamianie dla niego tej samej strategii co dla gracza własnego, ale w pierwszym przypadku o mniejszej wartości K, a w drugim o większym K niż dla gracza własnego).

b) Sterowanie: Sprawdzić wykonanie programu dla różnych wartości parametrów K i P (liczba wykonywanych ruchów, rozmiar drzewa przeszukiwania, rezultat gry).

c) Podsumować uzyskane wyniki i przedstawić wnioski.

Projekt S4. „Problem komiwojażera" jako problem „optymalnego CSP".

Rozszerzyć algorytm **przeszukiwania przyrostowego z nawrotami** zdefiniowany dla CSP o koszty wykonywanych akcji (np. kosztu przypisania miasta jako kolejnego etapu podróży w problemie „komiwojażera") i pamiętanie kosztu najlepszego dotychczasowego rozwiązania. Każda ścieżka w drzewie decyzyjnym może zostać „przycięta" w momencie, gdy można uznać, że nie prowadzi do rozwiązania o niższym koszcie niż już znalezione rozwiązanie.

Zrealizować zagadnienia:

- Metoda: Zaimplementować rozszerzoną strategię przeszukiwania przyrostowego w głąb z nawrotami dla problemu „optymalnego CSP". Jak należy zmodyfikować Uwzględnić alternatywne rozwiązanie stosujące (ewentualnie zmodyfikowane) techniki „najbardziej ograniczona zmienna", „najbardziej ograniczająca wartość" i „podglądania wprzód".
- Problem: Wprowadzać opis „problemu komiwojażera" w postaci grafu ograniczeń (np. brak powtórnego wyboru miasta na trasie podróży).

  Wprowadzanie do programu opisów konkretnych „problemów komiwojażera" – grafu miast, miasta startu i kosztów połączeń pomiędzy miastami.
- Sterowanie: Realizacja tekstowego interfejsu użytkownika. Wykonanie testów działania programu dla różnych grafów miast i połączeń.
- Podsumowanie wyników (koszt uzyskanej ścieżki, liczba rozwijanych węzłów drzewa przeszukiwania, itp.) i wnioski.

Projekt S5. Problem CSP. Zadawanie ograniczeń operacji arytmetycznej (kryptografia). Rozwiązanie problemu CSP (problemu z ograniczeniami) dzięki implementacji **przeszukiwania przyrostowego z nawrotami**.

Zrealizować zagadnienia:

- Metoda: Implementować strategię przeszukiwania przyrostowego w głąb z nawrotami z wykorzystaniem usprawnień algorytmu sposobami „najbardziej ograniczona zmienna", „najbardziej ograniczająca zmienna" i „podglądanie (tzn. sprawdzanie) wprzód".
- Problem: Wczytanie i interpretacja opisu problemu z ograniczeniami oraz wygenerowanie grafu ograniczeń dla wczytanego problemu. Zdefiniowanie opisów konkretnych problemów kryptograficznych.
- Sterowanie: Realizacja tekstowego interfejsu użytkownika. Wykonanie testów działania programu dla różnych obliczeń kryptograficznych i zebranie wyników (szybkość uzyskania wyniku, liczba rozwijanych węzłów drzewa przeszukiwania, liczba nawrotów, itp.).
- Podsumowanie wyników i podanie wniosków.

Część L (planowanie)

Projekt L2. Planowanie „Graphplan" w języku logiki. Problem „zakupowy".

Tworzenie **planu działania** dla rozwiązania problemu agenta „zakupowego" z wykorzystaniem algorytmu „**Graphplan**".

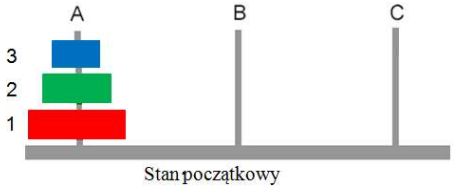
Zrealizować zagadnienia:

1. Metoda: Implementacja **strategii „Graphplan"** przeszukiwania przestrzeni planów.
2. Problem: Umożliwienie wprowadzenia (wczytania) opisu problemu planowania w postaci operatorów języka ADL. Zdefiniować przykładowe opisy problemu agenta „zakupowego".
3. Sterowanie: Monitorowanie procesu tworzenia planu po każdym kroku. Weryfikacja działania programu dla różnej liczby produktów (m=3-8) i różnej liczby sklepów (n=2,3,4).

Analiza osiągniętych wyników i podanie wniosków.

Projekt L3. Planowanie PCzU w języku logiki. Problem „wież Hanoi".

Tworzenie **planu częściowo-uporządkowanego** dla problemu „wież Hanoi" z wykorzystaniem metodologii STRIPS.



(ilustracja dla N=3)

Problem „wież Hanoi:

- dane są trzy patyki (o nazwach A, B i C) na które wkładać można N krążków o różnej średnicy (oznaczmy je od największego do najmniejszego jako 1, 2 … N);
- należy umieścić N krążków na patyku docelowym C w taki sposób, aby największy krążek 1 znalazł się na dole, na nim krążek 2, itd., a na samej górze – najmniejszy krążek N. Jednocześnie można przemieszczać tylko jeden krążek z patyka na patyk.

W szczególności należy zrealizować następujące zadania:

1. Metoda: Implementacja **strategii przeszukiwania przestrzeni planów** – przeszukiwanie w głąb z nawrotami. Implementacja konwersji planu nieuporządkowanego na postać planu częściowo-uporządkowanego.
2. Problem: Zdefiniować opis problemu „wież Hanoi" dla *N*=3 i dowolnego *N*, w postaci operatorów języka STRIPS.
3. Sterowanie: Monitorowanie procesu tworzenia planu po każdym kroku. Weryfikacja działania programu dla liczby krążków *N*=3 i N > 3 (np. 4, 5).

Analiza osiągniętych wyników i podanie wniosków.

Część U (uczenie przez indukcję)

Projekt U1. Klasteryzacja X-średnich i aproksymacja funkcji. Zrealizować algorytm segmentacji monochromatycznego obrazu składający się z trzech etapów: 1) detekcja obrazu krawędziowego, 2) klasteryzacja wektora cech pikseli krawędziowych, 3) aproksymacja elementów klastra linią prostą w obrazie. Do realizacji pierwszego etapu można zastosować bibliotekę OpenCV. Zrealizować zadanie dla zbioru obrazów samochodów (np. „Stanford Car Dataset").

- Problem 1: detekcja obrazu krawędziowego zadanego obrazu, np. operatorem Sobela; w jego wyniku każdy element krawędziowy o współrzędnych (x, y) jest charakteryzowany parą cech [siła krawędzi S, kierunek R];
- Metoda 1: zaimplementować metodę X-średnich dla klasteryzacji (grupowania) zbioru pikseli krawędziowych w oparciu o parę cech (S, R);
- Problem 2: wybrać „dominujące" klastry spełniające warunki: S(klastra) > 0.5 Smax i (liczba elementów klastra) > (średnia liczba elementów w klastrach);
- Metoda 2: każdy wybrany „dominujący" klaster należy zaaproksymować funkcją liniową (metodą regresji LSE) w układzie współrzędnych obrazu xy jako: y = f(x), biorąc pod uwagę kierunek i współrzędne punktów należących do tego samego klastra.
- Sterowanie: zrealizować proste tekstowe menu umożliwiające Użytkownikowi zadawanie parametrów model i testowania;
- Sterowanie: Sprawdzić działanie programu dla zadanych obrazów wejściowych o długich i prostych krawędziach. Wynik ma postać: liczba klastrów, wybrane klastry, ich zbiory krawędzi i proste, y = f(x), dla każdego klastra.

W raporcie przedstawić też i podsumować uzyskane wyniki oraz podać wnioski.

Projekt U2. Klasyfikator wg. wielomianowej **funkcji potencjału** dla 3 klas.

Zrealizować program uczenia i testowania klasyfikatora stosującego funkcje potencjału o postaci wielomianu 2 stopnia dla 3 klas w 2-wymiarowej przestrzeni cech – poprzez rozwiązanie odpowiedniego zadania optymalizacji MNK i rozwiązanie docelowych układów równań liniowych.

a) Metoda: Zaimplementować wymagany algorytm uczenia klasyfikatora i funkcję decyzyjną klasyfikatora. Uczymy dla każdej klasy z osobna, stosując podział próbek uczących na dwa podzbiory – próbek należących do danej klasy i należących do innych klas.
b) Problem: Automatycznie generować zbiory próbek – próbki to wektory cech o 2 wymiarach – np. na podstawie dwóch cech pikseli obrazu wyliczonych na podstawie ich lokalnego otoczenia 3x3 lub 5x5 i klasyfikacji jako piksel wewnętrzny obszaru,

brzegowy (krawędziowy) lub narożnik.

c) Sterowanie: Sprawdzić działanie programu na zbiorze testowym. Wynik ma postać wygenerowanego zbioru próbek uczących, nauczonych parametrów klasyfikatora i skuteczności klasyfikacji dla próbek testowych.

d) Przedstawić graficznie uzyskane wyniki, podać skuteczność klasyfikacji i macierze pomyłek między-klasowych.

Projekt U3. Uproszczona metoda uczenia klasycznego klasyfikatora „SVM".

Zrealizować program do uczenia i testowania klasyfikatora SVM dla przypadku 3-wymiarowych cech i niezbyt licznego zbioru próbek. Założyć liniowy charakter klasyfikatora SVM z możliwymi przekłamaniami klasyfikacji, modelowanymi jako szum.

- Metoda: Zaprojektować i wykonać uproszczoną metodę wyznaczana parametrów klasyfikatora w postaci wyszukiwania w dziedzinie pierwotnej najlepszego podzbioru czterech „wektorów nośnych" („podpierających") dla SVM. Zastanowić się nad wstępnymi kryteriami ograniczającymi proces przeszukiwania, czyli ograniczającymi liczbę podzbiorów - możliwych kandydatów rozwiązania.

  Zrealizować klasyfikator SVM dla wyznaczonych wektorów nośnych.
- Problem: Pozyskać zbiór próbek (uczących i testowych) i przekazać je do klasyfikatora.
- Sterowanie: Przeprowadzić proces uczenia i sprawdzić działanie programu dla różnych podziałów zbioru próbek na część uczącą i testową. Wynik ma postać wygenerowanego zbioru wektorów nośnych i skuteczności klasyfikacji dla próbek testowych.
- Zebrać i podsumować uzyskane wyniki. Przedstawić wnioski.

Projekt U4. Algorytm **budowy drzewa decyzyjnego – kryterium ilorazu zysku informacji**

Wykonać program realizujący budowę drzewa decyzyjnego dla pojęć (klas) reprezentujących figury geometryczne o różnym kształcie (trójkąt, okrąg, elipsa, kwadrat, prostokąt, wielokąt wklęsły, wielokąt wypukły, itp.), wykryte wcześniej w obrazie.

- Problem: Określić predykaty potrzebne do odróżnienia klas od siebie. Wygenerować wystarczająco duży zbiór próbek uczących.
- Metoda: Zaimplementować algorytm budowy drzewa decyzyjnego, stosując kryterium ilorazu **zysku informacji** (ang. SplitInfoGain) dla wyboru atrybutów.
- Sterowanie: Wykonać proces uczenia się drzewa decyzyjnego dla zadanych klas w oparciu o etykietowany zbiór uczący Sprawdzić działanie programu dla powyższych klas i różnych podzbiorów zbioru próbek uczących. Wynik ma postać uzyskanego drzewa decyzyjnego.
- Podsumować wyniki i przedstawić wnioski.

Projekt U5. Metoda Levenberga-Marquardta. Zrealizować algorytm Levenberga-Marquardta do nieliniowej optymalizacji MNK i zastosować go do aproksymacji funkcji dwóch zmiennych:

- Założyć dwuwymiarową przestrzeń cech (x_1, x_2) i wygenerować dane uczące (lub przyjąć dane z istniejącego rzeczywistego zbioru danych). Ograniczyć dziedzinę obu zmiennych.
- Przyjąć model funkcji nieliniowej o postaci wielomianu drugiego stopnia:

  $y = A\,x_1^2 + B\,x_2^2 + C\,x_1\,x_2 + D\,x_1 + E\,x_2 + F$
- Metoda: zaimplementować metodę Levenberga-Marquardta (L-M);
- Sterowanie: dla zadanego zbioru uczącego wykonać wielokrotnie metodę L-M i określać średni błąd kwadratowy aproksymacji przez każdy wynik optymalizacji. Zebrać uzyskane wyniki i przedstawić (graficznie poza programem) przy pomocy narzędzia graficznego

dane i uzyskane aproksymacje.

- Problem: wygenerować ciekawe dane uczące – np. przybliżenie powierzchni górzystego terenu geograficznego lub sztucznie wygenerowane punkty istniejącego wielomianu 2-go stopnia.
- Podsumować uzyskane wyniki i sformułować wnioski.

Projekt U6. Algorytm **budowy drzewa decyzyjnego – kryterium zysku informacji.**

Wykonać program realizujący budowę drzewa decyzyjnego dla pojęć (klas) reprezentujących choroby człowieka w oparciu o atrybuty przedstawiające sobą różne symptomy tych chorób, w tym uwzględnić wielowartościowe atrybuty.

- Problem: Określić predykaty potrzebne do odróżnienia klas od siebie. Wyszukać w Internecie lub wygenerować samemu wystarczająco duży zbiór próbek uczących.
- Metoda: Zaimplementować algorytm budowy drzewa decyzyjnego, stosując kryterium **zysku informacji** (ang. InfoGain) dla wyboru atrybutów.
- Sterowanie: Wykonać proces uczenia się drzewa decyzyjnego dla zadanych klas w oparciu o etykietowany zbiór uczący. Sprawdzić działanie programu dla powyższych klas i kilku różnych podzbiorów zbioru próbek uczących. Wynik ma postać uzyskanego drzewa decyzyjnego.
- Przedstawić uzyskane wyniki i podać wnioski.

Projekt U7. **Klasteryzacja** stochastyczna **GMM** (mieszanina rozkładów Gaussa) i **klasyfikacja** binarna nagrania audio.

Celem jest odróżnienie tego czy plik z nagraniem audio zawiera mowę naturalną czy syntezowaną programem syntezy mowy typu „text-to-speech”. Klasyfikację należy oprzeć o uprzednie stworzenie dwóch modeli GMM dla nagrań rzeczywistych i syntezowanych. Reguła decyzyjna klasyfikatora polega na maksymalizacji prawdopodobieństwa przynależności badanej próbki (nagrania testowego) do danej klasy, zgodnie z modelami GMM. Do wyznaczania cech ramek sygnału audio można użyć biblioteki Librosa lub innej (np. HTK toolkit).

- Problem: zebrać dostateczną liczbę przykładów – plików audio zawierających mowę rzeczywistą i syntezowaną (np. baza ASVspoof 2019 lub 2021) – przeznaczyć część większościową zbioru do uczenia się dwóch modeli GMM a pozostałą część do testowania procesu klasyfikacji - stosując funkcje biblioteki Librosa (lub podobnej) wyznaczyć wektory cech MFCC (oraz ich pochodne - cechy „delta MFCC”) dla krótkookresowych ramek (po ok. 16-20 ms) sygnału audio, dla wszystkich przykładów (zebranych plików audio);
- Metoda: zaimplementować metodę klasteryzacji stochastycznej GMM dla zadanego zbioru wektorów cech ramek sygnału; zrealizować funkcje klasyfikacji w oparciu o regułę maksymalizacji prawdopodobieństwa przynależności testowanej próbki do klasy;
- Sterowanie: zrealizować proste menu tekstowe umożliwiające Użytkownikowi zadawanie parametrów przetwarzania (procentowy podział zbioru przykładów na uczący i testowy, liczba cech MFCC – od 12 do 30; liczba klastrów modelu GMM – od 12 do 40, warunek zakończenia procesu uczenia);
- Sterowanie: wykonać klasteryzację GMM dla dwóch podzbiorów uczących uzyskując po jednym modelu GMM dla każdej z dwóch klas (mowy rzeczywistej i syntezowanej), zrealizować testowanie – klasyfikację przykładów z podzbioru testowego i wyznaczyć miary dokładności klasyfikacji.

W raporcie należy też przedstawić i podsumować uzyskane wyniki oraz podać wnioski.

Projekt U8. Klasyfikacja emocji mówcy klasyfikatorem Bayesa wyuczonym na zbiorze nagrań RAVDESS.

Celem jest wyuczenie klasyfikatora Bayesa na zbiorze RAVDESS i wyznaczenie jednej z 8 klas stanu emocjonalnego mówcy w nagraniach testowych. Reguła decyzyjna klasyfikatora polega na maksymalizacji prawdopodobieństwa „a posteriori" przynależności badanej próbki (nagrania testowego) do danej klasy – stanu emocjonalnego mówcy. Do wyznaczania cech ramek sygnału audio można użyć biblioteki Librosa lub innej (np. HTK toolkit)

- Problem: pobrać z Internetu zbiór nagrań RAVDESS – przeznaczyć część większościową zbioru do wyznaczenia parametrów modeli klas wymaganych w klasyfikatorze Bayesa, a pozostałą część do testowania procesu klasyfikacji - stosując funkcje biblioteki Librosa (lub podobnej) wyznaczyć wektory cech MFCC (oraz ich pochodne - cechy „delta MFCC") dla ramek sygnału audio (trwających po ok. 30 ms) a następnie przekształcić je w docelowy wektor zawierający wartości średnie i wariancje poszczególnych cech w jednym nagraniu – wyznaczyć takie wektory cech dla wszystkich przykładów (plików audio);
- Metoda: zaimplementować metodę uczenia się parametrów rozkładów prawdopodobieństwa Gaussa wymaganych w projekcie klasyfikatora Bayesa; zrealizować funkcję klasyfikacji w oparciu o regułę maksymalizacji prawdopodobieństwa „a posteriori" przynależności testowanej próbki do jednej z klas;
- Sterowanie: zrealizować proste menu tekstowe umożliwiające Użytkownikowi zadawanie parametrów przetwarzania (procentowy podział zbioru przykładów na uczący i testowy, stopień przeplotu ramek sygnału, liczba cech MFCC – od 12 do 30, warunek zakończenia procesu uczenia);
- Sterowanie: wykonać uczenie klasyfikatora Bayesa dla podzbioru uczącego; wykonać klasyfikację dla podzbiorów uczącego i testowego dla wielu różnych wartości parametrów i wyznaczać miary dokładności klasyfikacji.

W raporcie należy też przedstawić i podsumować uzyskane wyniki oraz podać wnioski.

Projekt U9. Klasyfikacja wieku mówcy klasyfikatorem kNN (k najbliższych sąsiadów) wyuczonym na podzbiorze języka polskiego bazy CommonVoice.

Celem jest wyuczenie klasyfikatora kNN (dla 3-6 klas przedziału wieku) na zbiorze języka polskiego bazy nagrań CommonVoice. Reguła decyzyjna klasyfikatora powinna być sparametryzowana tak, aby uwzględniać wartość „k" sąsiadów od k=1 do k=5. Należy też umożliwić ograniczenie liczby reprezentantów w przestrzeni cech.

Do wyznaczania cech ramek sygnału audio można użyć biblioteki Librosa lub innej (np. HTK toolkit).

- Problem: pobrać z Internetu zbiór języka polskiego bazy nagrań CommonVoice i przeznaczyć część większościową zbioru do wyznaczenia model klasyfikatora kNN, a pozostałą część do testowania procesu klasyfikacji - stosując funkcje biblioteki Librosa (lub podobnej) wyznaczyć wektory cech MFCC (oraz ich pochodne - cechy „delta MFCC") dla ramek sygnału audio (trwających po ok. 20 ms) a następnie przekształcić je w docelowy wektor zawierający wartości średnie i wariancje poszczególnych cech w jednym nagraniu – wyznaczyć takie wektory cech dla wszystkich przykładów (plików audio);
- Metoda: zaimplementować metodę uczenia się modelu kNN o zadanej liczbie reprezentantów N mniejszej lub równej liczbie przykładów uczących; zrealizować funkcję klasyfikacji kNN dla różnych wartości k (od 1 do 5);
- Sterowanie: zrealizować proste menu tekstowe umożliwiające Użytkownikowi zadawanie parametrów przetwarzania (procentowy podział zbioru przykładów na uczący i testowy, stopień przeplotu ramek sygnału, liczba cech MFCC – od 12 do 30, liczba reprezentantów

N);

- Sterowanie: wykonać uczenie klasyfikatora kNN dla podzbioru uczącego; wykonać klasyfikację dla podzbiorów uczącego i testowego dla wielu różnych wartości parametrów i wyznaczać miary dokładności klasyfikacji.

W raporcie należy też przedstawić i podsumować uzyskane wyniki oraz podać wnioski.

Część N (sieci neuronowe)

Projekt N4. Sieć CNN do klasyfikacji obrazów ubrań.

Wykonać implementację podstawowej splotowej sieci neuronowej CNN i jej algorytmu uczenia, z warstwą wyjściową typu softmax, przeznaczoną do klasyfikacji monochromatycznych obrazów o znormalizowanej rozdzielczości, zawierających pojedyncze wzorce ubrań 10 typów (klas). Zastosować bazę danych Fashion-MNIST lub podobną. Wygenerować własnych kilka obrazów każdej klasy w celach testowych.

- Metoda: Zaprojektować i zaimplementować algorytm działania sieci CNN (aproksymacji nauczonej funkcji) i algorytm uczenia tej sieci.
- Problem: Zastosować bazę danych Fashion-MNIST lub podobną do uczenia i testowania klasyfikatora. Dodać własne obrazy testowe. Wczytywać obrazy, dopasowywać ich rozdzielczość i typ do wymagań sieci neuronowej i przekazywać je na wejście (obraz) i do wyjścia (etykiety klas) sieci.
- Sterowanie: Zrealizować uczenie i testowanie kilku modeli różniących się liczbą warstw i filtrów. Określać stopę błędu klasyfikacji na zbiorze uczącym i testowym po każdej epoce procesu uczenia. Wyznaczyć macierz pomyłek (między klasowych) procesu klasyfikacji zbioru testującego po zakończeniu uczenia.
- Przedstawić i skomentować uzyskane wyniki uczenia i klasyfikacji.

Projekt N5. Sieć CNN do klasyfikacji obrazów znaków drogowych.

Wykonać implementację podstawowej splotowej sieci neuronowej CNN i jej algorytmu uczenia, z warstwą wyjściową typu softmax, przeznaczoną do klasyfikacji kolorowych obrazów o znormalizowanej rozdzielczości, zawierających pojedyncze tablice znaków drogowych 10 typów (klas). Zastosować dostępną bazę danych lub wygenerować własny zbiór przynajmniej po 10 obrazów każdej klasy.

- Metoda: Zaprojektować i zaimplementować algorytm działania sieci CNN (aproksymacji nauczonej funkcji) i algorytm uczenia tej sieci.
- Problem: Zastosować bazę obrazów do uczenia i testowania klasyfikatora. Wczytywać obrazy, dopasowywać ich rozdzielczość i typ do wymagań sieci neuronowej i przekazywać je na wejście (obraz) i do wyjścia (etykiety klas) sieci.
- Sterowanie: Zrealizować uczenie i testowanie kilku modeli różniących się liczbą warstw i filtrów. Określać stopę błędu klasyfikacji na zbiorze uczącym i testowym po każdej epoce procesu uczenia. Wyznaczyć macierz pomyłek (między klasowych) procesu klasyfikacji zbioru testującego po zakończeniu uczenia.
- Przedstawić i skomentować uzyskane wyniki uczenia i klasyfikacji.

Projekt N6. Sieć CNN do klasyfikacji obrazów znaków.

Wykonać implementację podstawowej splotowej sieci neuronowej CNN i jej algorytmu uczenia, z warstwą wyjściową typu softmax, przeznaczoną do klasyfikacji monochromatycznych obrazów o znormalizowanej rozdzielczości, zawierających pojedyncze znaki pisane odręcznie (minimum 10 klas). Zastosować bazę danych obrazów odręcznie pisanych znaków MNIST lub podobną. Wygenerować własnych kilka obrazów każdej klasy w celach testowych.

- Metoda: Zaprojektować i zaimplementować algorytm działania sieci CNN (aproksymacji nauczonej funkcji) i algorytm uczenia tej sieci.
- Problem: Zastosować bazę danych odręcznie pisanych znaków (cyfr lub liter) MNIST lub podobną do uczenia i testowania klasyfikatora. Dodać własne obrazy testowe. Wczytywać obrazy, dopasowywać ich rozdzielczość i typ do wymagań sieci neuronowej i przekazywać je na wejście (obraz) i do wyjścia (etykiety klas) sieci.
- Sterowanie: Zrealizować uczenie i testowanie kilku modeli różniących się liczbą warstw i filtrów. Określać stopę błędu klasyfikacji na zbiorze uczącym i testowym po każdej epoce procesu uczenia. Wyznaczyć macierz pomyłek (między klasowych) procesu klasyfikacji zbioru testującego po zakończeniu uczenia.
- Przedstawić i skomentować uzyskane wyniki uczenia i klasyfikacji.

Projekt N7. Sieć CNN do klasyfikacji obrazów ptaków.

Wykonać implementację podstawowej splotowej sieci neuronowej CNN i jej algorytmu uczenia, z warstwą wyjściową typu softmax, przeznaczoną do klasyfikacji kolorowych obrazów ptaków (minimum 10 typów). Zastosować bazę danych obrazów „225 Bird Species Dataset" z Kaggle lub podobną. Wybrać obrazy minimum 10 typów ptaków. Dodać inne obrazy ptaków w celach testowych.

- Metoda: Zaprojektować i zaimplementować algorytm działania sieci CNN (aproksymacji nauczonej funkcji) i algorytm uczenia tej sieci.
- Problem: Zastosować bazę danych „225 Birs Species" lub podobną do uczenia i testowania klasyfikatora. Dodać własne obrazy testowe. Wczytywać obrazy, dopasowywać ich rozdzielczość i typ do wymagań sieci neuronowej i przekazywać je na wejście (obraz) i do wyjścia (etykiety klas) sieci.
- Sterowanie: Zrealizować uczenie i testowanie kilku modeli różniących się liczbą warstw i filtrów. Określać stopę błędu klasyfikacji na zbiorze uczącym i testowym po każdej epoce procesu uczenia. Wyznaczyć macierz pomyłek (między klasowych) procesu klasyfikacji zbioru testującego po zakończeniu uczenia.
- Przedstawić i skomentować uzyskane wyniki uczenia i klasyfikacji.

Projekt N8. Sieć neuronowa typu „Deep Ranking" do identyfikacji obrazów twarzy.

Wykonać implementację sieci neuronowej typu „Deep Ranking" i wytrenować ją do identyfikacji obrazów twarzy. Zastosować bazę obrazów „CelebFaces Attributes (CelebA) dataset" z Kaggle lub podobną. Sieć trenowana jest w sposób dyskryminacyjny, stosując trzy-składową funkcję straty („tripple loss function"). Sieć wyznacza miarę podobieństwa obrazu z wzorcem.

- Metoda: Zaprojektować i zaimplementować algorytm działania sieci neuronowej typu „Deep ranking" i algorytm uczenia tej sieci.
- Problem: Zastosować bazę danych „CelebFaces Attributes (CelebA)" lub podobną do uczenia i testowania klasyfikatora. Wczytywać obrazy, dopasowywać ich rozdzielczość i

typ do wymagań sieci neuronowej i przekazywać je na wejście (obraz) i do wyjścia (identyfikatory osób) sieci.

- Sterowanie: Zrealizować uczenie i testowanie kilku modeli różniących się liczbą warstw i wag. Określać stopę błędu identyfikacji na zbiorze uczącym i testowym.
- Przedstawić i skomentować uzyskane wyniki uczenia i klasyfikacji.

Projekt N9. Sieć rekurencyjna RNN do rozpoznawania pisma.

Wykonać implementację podstawowej rekurencyjnej sieci neuronowej RNN i jej algorytmu uczenia, z warstwą wyjściową typu softmax, przeznaczoną do rozpoznawania pisma (prostych znaków lub liter pisanych) na podstawie sekwencji położeń pióra podczas pisania.

Zastosować bazę danych z pismem lub wygenerować własną. Wybrać minimum 10 typów znaków.

- Metoda: Zaprojektować i zaimplementować algorytm działania sieci RNN i algorytm uczenia tej sieci.
- Problem: Zastosować bazę danych z pismem odręcznym lub wygenerować własną, niedużą bazę z danymi, na potrzeby uczenia i testowania klasyfikatora. Przeprowadzić uczenie i testowanie sieci.
- Sterowanie: Zrealizować uczenie i testowanie kilku modeli różniących się liczbą warstw i wag. Określać stopę błędu klasyfikacji na zbiorze uczącym i testowym po każdej epoce procesu uczenia. Wyznaczyć macierz pomyłek (między klasowych) procesu klasyfikacji zbioru testującego po zakończeniu uczenia.
- Przedstawić i skomentować uzyskane wyniki uczenia i klasyfikacji.

Projekt N10. Sieć typu autoenkoder do semantycznej klasyfikacji obrazów.

Wykonać implementację sieci typu autoenkoder zbudowanej na podstawowych warstwach splotowych sieci neuronowej (CNN) przeznaczonej do klasyfikacji pikseli obrazów o znormalizowanej rozdzielczości przedstawiających środowiska określonego typu (np. wnętrza mieszkania z obszarami typu podłoga, łóżko, drzwi, stół):

- Metoda: Zaprojektować i zaimplementować algorytm działania sieci enkodera CNN i algorytm uczenia tej sieci.
- Problem: Wygenerować (zebrać) obrazy uczące, walidujące i testowe oraz wyznaczyć dla nich posegmentowane obrazy referencyjne.
- Sterowanie: Przeprowadzić uczenie i testowanie sieci. Określać błąd klasyfikacji na zbiorze uczącym, walidacyjnym i testowym po każdej epoce procesu uczenia.
- Przedstawić i skomentować uzyskane wyniki.