Sztuczna Inteligencja

*Projekt*

**Skład zespołu:**

Łukasz Dudziak, Marta Żuławska, Anna Żurawik, Aleksander Smiatacz

**Wybrany temat:**

Wykrywanie i zliczanie wystąpień samochodów na zdjęciach ze skrzyżowań

kamer miejskich za pomocą sieci neuronowej.

***Sformułowanie problemu:***

W dzisiejszych czasach analiza i klasyfikacja (np. pod względem natężenia) ruchu drogowego na skrzyżowaniach jest stosunkowo często poruszanym problemem, wiążącym się ze stałym rozwojem ośrodków miejskich i rosnącą liczbą uczestników ruchu miejskiego. Poprawne podejście do omawianego problemu pozwala na usprawnienie ruchu, a co za tym idzie redukcję zanieczyszczeń, zwiększenie bezpieczeństwa itp.

W naszym problemie postanowiliśmy skoncentrować się przede wszystkim na liczeniu pojazdów na zdjęciach, jednak bardziej ogólne, a nawet zupełnie odmienne, podejścia również są możliwe; część z nich omówiona została bardziej szczegółowo w dalszej części pracy (np. klasyfikacja względem typu pojazdu).

***Analiza problemu:***

Wybrany przez nas temat wiąże się z dużą liczbą problemów, które będzie trzeba rozwiązać w trakcie tworzenia projektu. Pierwszym z nich jest odpowiednie przetworzenie danych wejściowych – tutaj będzie trzeba zająć się oddzieleniem samochodów od otoczenia, żeby skupić się na obiektach, które nas interesują. Ponadto istotnym kłopotem jest nakładanie się pojazdów w związku z ich licznymi wystąpieniami na zdjęciach, jak i ich różne typy samochodów (osobowe, ciężarowe, z przyczepami itd.), co może negatywnie wpływać na podanie poprawnego wyniku. Problemem może być również kwestia perspektywy (przy metodzie sliding window), jak i oświetlenia szukanych przez nas środków transportu. Wiązać się to będzie z licznymi uproszczeniami w modelu wzorcowym tego projektu. Uproszczenia najprawdopodobniej będą się wiązać nie tylko z grafiką, ale i budową samej sieci neuronowej. Tutaj istotnym problemem będzie również dobranie odpowiedniej funkcji oceniającej (kryterium), dla wcześniej spreparowanych danych wejściowych, jak i przygotowanie odpowiednio dużego zbioru uczącego, aby można było zwrócić wiarygodne wyniki.

Dokładna realizacja zadania zależy od przyjętej metody i ewentualnych uproszczeń związanych z ww. problemami i nie może być jednoznacznie określona na tak wczesnym etapie projektu. Kwestie związane ze wstępnym przetwarzaniem obrazu najprawdopodobniej wiązać się będą z wykorzystaniem biblioteki OpenCV. Dane wejściowe przypuszczalnie będą również związane z dużymi uproszczeniami, jak np. mniejsze zagęszczenie drogi przy odpowiednim oświetleniu. Jeśli chodzi o samą budowę sieci neuronowej, istnieje biblioteka FANN (ang. Fast Artificial Neural Network), która również może nam pomóc przy pracy nad rozwiązaniem postawionych sobie problemów. Ponadto istotną pomocą będą istniejące już rozwiązania, które są opisane poniżej.

***Analiza istniejących rozwiązań:***

# Multi-Net for Object Detection (MNOD):

Metoda została opisana tutaj: <http://www.academia.edu/1272586/Learning_Object_Detection_using_Multiple_Neural_Networks>

MNOD odnosi się do różnych obiektów, nie tylko samochodów. Multi-net to multisieć - sieć sieci neuronowych, z których każda jest konfigurowana i uczona osobno. Analiza tej metody pozwoliła stwierdzić, że jest ona lepsza w przypadkach zmian perspektywy oraz, gdy jakiś obiekt pojawia się na zdjęciach w różnym położeniu (od tyłu, od boku itp.). Informacje podawane na wejście każdej sieci pochodzą bezpośrednio z obrazu lub z wyjść inny sieci. Oczywiście, aby wytrenować sieć, musimy najpierw zrobić to z sieciami-dziećmi. W tej metodzie, gdy chcemy przeanalizować jeden obraz, poddajemy go różnym operacjom podkreślającym jego pewne istotne cechy, na przykład poprzez zmianę nasycenia, odcieni, jasności czy zastosowaniu różnych filtrów, przez co powstaje wiele różnych obrazów, tworzących później wejścia różnych (często kilku) węzłów multisieci. Rozmiar wejściowego obrazka jest ustalany taki sam dla każdej sieci. Obraz jest odpowiednio skalowany, wykorzystuje się też tzw. sliding window - okienko przesuwające się po obrazie. Zaletą MNOD jest to, że pozwala na podział problemu na podproblemy. Niestety ma też wadę - w przypadku dużej różnicy w skali obiektu nie działa zbyt dobrze.

# Vehicle Detection, Tracking and Counting:

Metoda została opisana tutaj:

<https://www.behance.net/gallery/Vehicle-Detection-Tracking-and-Counting/4057777>

[Andrews Sobral](https://www.behance.net/andrewssobral), autor tej metody zajął się również śledzeniem samochodów na filmach z kamerek, nas interesuje tylko wykrywanie i liczenie. W swoim projekcie wspomina on o dwóch metodach wykrywania obiektów. Pierwszą jest Haar Cascades a dokładniej Haar-like features z wykorzystaniem Cascade Classifier. Polega ona na znajdywaniu na obrazie kilku sąsiadujących, prostokątnych obszarów, których różnice sum jasności wszystkich pikseli spełniają pewne warunki. Metoda ta wymaga uczenia klasyfikatora. Andrews Sobel korzysta z gotowej implementacji klasyfikatora OpenCV. Drugim sposobem wykrywania obiektów jest Background Substraction. W tej metodzie potrzebne są sekwencje ramek filmu lub zdjęć. Są na nich wykonywane pewne operacje a później od każdej ramki jest odejmowane tło, wyznaczane na podstawie wszystkich ramek. Autor metody sam korzysta tu z biblioteki stworzonej przez siebie, ale w OpenCV również jest dostępne narzędzie odejmowania tła. Ten sposób nieco lepiej się sprawdza niż Haar Cascades, ale przy obu z nich pojawiają się problemy w przypadku dużej liczby samochodów obok siebie, np. gdy jest korek.  
  
Inne prace:

- <https://cseweb.ucsd.edu/classes/fa05/cse252c/dlagnekov_belongie.pdf> (jednak tu autor skupia się bardziej na rozpoznawaniu konkretnych marek aut)

***Koncepcje rozwiązania problemu:***

1. Rozwiązania wstępne

a. Omówienie

Wstępna realizacja projektu ma na celu zapoznanie się ze środowiskiem i wiąże ze sobą liczne uproszczenia. Na tym etapie projektu postaramy się zrozumieć działanie i zbudować fundamenty pod dalszą realizację projektu.

b. Wstępne założenia

Wyżej wspomniane uproszczenia oznaczają w dużym stopniu zmniejszenie wymagań odnośnie danych wejściowych do sieci neuronowej. Będą nimi obrazki złożone jedynie z dwóch kolorów pikseli - czarnych i białych. Dane uczące sieć neuronową nie będą zawierały wszystkich kombinacji pikseli, jednak będą zawierać określoną ilość obrazków z poprawnymi odpowiedziami. W przeciwnym wypadku sieć neuronowa znałaby wszystkie możliwe kombinacje i prawidłową odpowiedź na zadane dane wejściowe, gdyż zgodnie z założeniami odpowiedź algorytmu sieci neuronowej na danych testowych powinien zwracać 100% poprawnych odpowiedzi. Najważniejszym założeniem do danych wejściowych na tym etapie jest to, że będą zliczane białe pola na czarnym tle.

c. Generowanie obrazów i danych wejściowych

Wcześniej wspomniane obrazki będą generowane losowo. Ich ilość jak i rozmiar są parametrami do ustalenia w kodzie generującym obrazki. Poprawna odpowiedź zostanie zapisana w nazwie pliku, a on sam będzie już interpretowany w postaci tabeli zawierającej kolory pikseli.

d. Sieć neuronowa - budowa i cele

Sieć neuronowa w tej fazie pracy będzie polegała na wykorzystaniu wcześniej wspomnianej biblioteki FANN. Jej cel to zliczanie białych pól, biorąc pod uwagę czy dane dwa białe piksele są ze sobą w dowolnym sąsiedztwie pikseli. Jeśli tak, to będą one zaliczane do jednej grupy. Późniejsza rozbudowa sieci neuronowej jak i przetwarzania obrazów pozwoli na bardziej zaawansowane działania, co jest celem naszego projektu.

e. Przykłady



Przedstawiony powyżej rysunek jest rozmiaru 50x50 pikseli i może być przykładem danych wejściowych do sieci neuronowej. Małe zagęszczenie białych pól ułatwia w tym wypadku ich zliczanie. Poprawną odpowiedzią do zadanego przykładu jest 95.

2. Plan ew. rozbudowy projektu

a. Omówienie

Po zaimplementowaniu i sprawdzeniu działania poprawności wcześniejszego etapu, spróbujemy nieco bardziej rozbudować projekt o niżej opisane punkty.

b. Przetwarzanie obrazu

i. Przekształcenie obrazu do postaci z pierwszego etapu można wykonać na dwa sposoby. Pierwszym jest użycie obrazu referencyjnego. Uznaje się że nie ma na nim interesujących nas obiektów - na przykład jest to obrazek przedstawiający pustą ulicę. Następnie tworzy się obrazek będący absolutną różnicą dwóch obrazków - referencyjnego i tego na którym chcemy rozpoznać samochody. Można też zastosować rożne filtry, poprawiające uzyskany wynik, jak filtr medianowy. Omówione operacje na obrazach można wykonać za pomocą biblioteki OpenCV. Przykładowy wynik takiej różnicy:



Aby znaleźć na takim obrazie konkretne elementy powinno się usunąć z niego konsekwencje naturalnych różnic między obrazem referencyjnym a tym na którym chcemy wykryć obiekty, takie jak zmieniające się światło i cień czy zmiany wynikające z podmuchu wiatru. Można skorzystać z funkcji threshold, dzięki której nie tylko pozbędziemy się tych niewielkich różnic ale także uzyskamy obraz na którym są tylko dwa możliwe kolory - czarny lub biały, dzięki czemu kolejne operacje będą łatwiejsze. Problemem może być dobór odpowiedniej wartości jasności piksela, powyżej której jest on kwalifikowany jako biały, szczególnie, że czasem samochody mogą mieć inne wartości różnicy w różnych miejscach, co powoduje, że cały samochód może się pojawić jako dwie grupy pikseli . Widać to na poniższym wyniku wspomnianej funkcji przedstawionym niżej. Pokazuje on ponadto że rzeczywiście, niewielkie różnice obrazów zostały wyeliminowane, chociaż nadal część z nich pozostało, np. część wynikająca z przesuwającego się cienia.

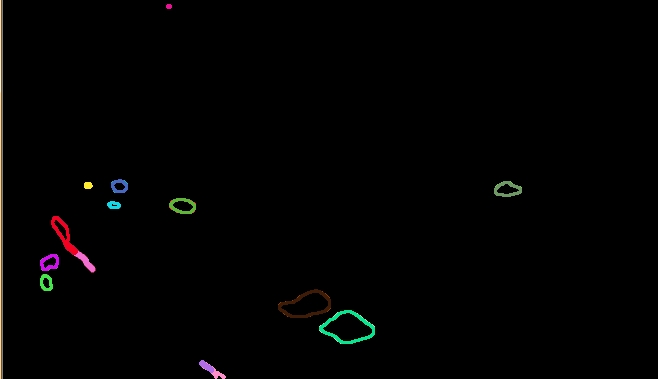


ii. Drugim sposobem otrzymania obrazu podobnego do poprzedniego jest skorzystanie z klasy [BackgroundSubtractor](http://docs.opencv.org/master/d7/df6/classcv_1_1BackgroundSubtractor.html), która sama na podstawie sekwencji ramek tworzy model tła. Korzysta także z różnicy i funkcji threshold ale wykorzystuje ponadto mieszaniny gaussowskie - bardziej skomplikowaną metodę.



Oba sposoby pozwalają w dużym stopniu zredukować obraz z kamery skrzyżowania do problemu, który występował wcześniej. To z której z nich skorzystamy może zależeć od wielu czynników, np. od tego czy ktoś posiada obrazek referencyjny.   
 Po przerobieniu obrazków to postaci grup białych pikseli na czarnym tle największym możliwym problemem będzie tu zakwalifikowanie tych grup jako jednego samochodu, jak i odróżnienie ich od innych białych punktów, jednak może nam w tym pomóc metoda omówiona poniżej.

iii. Wykrywanie konturów



Wykrywanie konturów pozwala zaklasyfikować zbiór danych pikseli jako jeden fragment. Pomijając kontury o zbyt małym kształcie i nakładające się samochody, które mogą być sklasyfikowane jako jeden kontur, będzie można znaleźć miejsca „podejrzane” o bycie samochodem na zadanym obrazku. Dzięki temu sieć neuronowa zostanie odciążona przez mniejszą ilość danych wejściowych, które mogłyby być kłopotem przy metodzie sliding window i przetwarzaniu całego obrazu od początku.

c. Rozbudowanie sieci neuronowej (SOM)

Istnieje możliwość rozbudowania sieci neuronowej o inny typ – SOM ( Self-organising map), która pozwala na rozpoznawanie podobnych obrazków. Wynikiem działania sieci jest klasyfikacja przestrzeni w sposób grupujący zarówno przypadki ze zbioru uczącego, jak i wszystkie inne wprowadzenia po procesie uczenia. Może to również usprawnić pracę działania programu.

***Problemy implementacyjne (2-3 strony):***

Wybór języka programowania.

Zdecydowaliśmy się na język C++. Jest dedykowanych dla niego kilka bibliotek, które spełniają wymagania projektu i znacznie ułatwią implementację programu. Oprócz tego, każdy z uczestników projektu ma doświadczenie z tym językiem.

Biblioteki

* OpenCV

Jest to biblioteka funkcji wykorzystywanych do przetwarzania obrazów. W naszym projekcie wykorzystujemy funkcje wykonujące operacje takie jak usuwanie szumów, progowanie obrazu, wykrywanie krawędzi.

* OpenCL

Jest to framework wspomagający pisanie programów równoległych, działających na platformach składających się z różnego rodzaju jednostek obliczeniowych (CPU, GPU). Umożliwi on szybsze wykonywanie programu.

* Qt

Aby ułatwić obsługę programu, zdecydowaliśmy się na użycie biblioteki graficznej. Qt jest biblioteką programistyczną, służąca do budowy graficznego interfejsu programu. Jest biblioteką dedykowaną m.in. dla języka C++.

Sieć neuronowa

W projekcie będzie wykorzystywana stworzona przez nas sztuczna sieć neuronowa. Będzie ona rozpoznawać, czy na fragmencie zdjęcia znajduje się samochód. Zostanie ona nauczona poprzez podawanie grup losowych zdjęć z kamer ulicznych.

Rozważamy stworzenie osobnych sieci do różnych rozmiarów obrazów, co pozwoli uniknąć problemów związanych z pogorszeniem jakości powiększonych zdjęć.