Sprawozdanie z laboratorium:
Komunikacja Człowiek Komputer

Przetwarzanie obrazów – aplikacja

Prowadzący: Agnieszka Mensfelt

Autorzy: Jakub Frąszczak 136704
Jakub Borowiak 136684

Zajęcia wtorkowe 16:50

Wstęp

Celem realizowanego projektu była implementacja aplikacji przetwarzającej zdjęcia planszy do gry w sudoku oraz rozwiązywanie go na podstawie odczytanych cyfr.

Zastosowane techniki przetwarzania obrazów

W celu ekstrakcji najważniejszych informacji z obrazów wykorzystywane są podstawowe operatory oraz filtry wykorzystywane do przetwarzania obrazów takie jak:

- **Gaussian Blur** – Filtr redukujący szum występujący na obrazie
- **Adaptive Thresholding** – Filtr umożliwiający separację pierwszego i drugiego planu obrazu na podstawie obliczonej granicy intensywności pikseli. Sam threshold jest obliczany w sposób lokalny na podstawie najbliżej występujących pikseli poprzez, na przykład, uśrednienie ich intensywności.
- **Dylatacja** – Podstawowa operacja w morfologii matematycznej umożliwiająca „wypełnienie" wybrakowań w obrazie
- **Erozja** - Podstawowa operacja w morfologii matematycznej umożliwiająca usunięcię zanieczyszczeń występujących w obrazie
- **Transformacja Hough'a** - Metoda wykrywania regularnych kształtów, w naszym przypadku prostych.
- **Transformacja Perspektywy** – Metoda umożliwiająca na podstawie 3 lub 4 współrzędnych obrazu przetransformować obraz na inną wybraną perspektywę, poprzez podanie docelowych lokalizacji wspomnianych współrzędnych.

Schemat dokonywanych transformacji

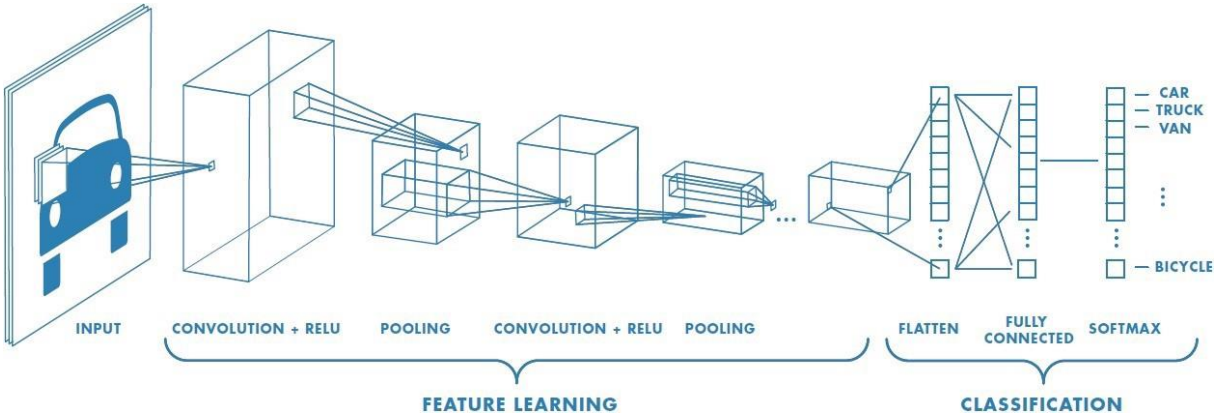
Preprocessing całego zdjęcia:

1. Gaussain Blur
2. Adaptive Threshold
3. Dylatacja
4. Transformacja Perspektywy
5. Transformacja Hough'a

Preprocessing wyciętych liczb:

1. Adaptive Threshold
2. Erozja

Architektura zastosowanej konwulsyjnej sieci neuronowej do rozpoznawania cyfr



Pierwszą warstwę stanowi warstwa konwolucyjna, mająca 30 filtrów. Jej celem jest wydobycie poszczególnych cech obrazu, które mogą być przydatne w detekcji danego obiektu. Następnie na obrazach otrzymanych po zastosowaniu filtrów stosowana jest funkcja RELU, której zadanie sprowadza się do zamiany ujemnych wartości na zerowe. Następną warstwą jest pooling zmniejszający wymiarowość map. Użyty przez nas pooling to Max Pooling o stride równym 2. Łącznie zastosowane zostały 2 warstwy konwulsyjne oraz 2 warstwy pooling. Dodatkowo w celu przeciwdziałania zjawisku overfittingu wykorzystana została technika dropout, ignorująca określony procent, losowo wybranych neuronów. W momencie, gdy neurony kluczowe w rozpoznawaniu danej cechy zostaną wyłączone pozostałe neurony są bardziej „zaangażowane" w proces uczenia, czego skutkiem jest wiele wewnętrznie niezależnych reprezentacji informacji nauczonej się przez sieć. Następnie, po nauczeniu się cech obrazu trafiają one do sztucznej sieci neuronowej posiadającej 2 warstwy ukryte o 128 i 50 neuronach. Jako funkcja aktywacyjna została wykorzystana funkcja RELU, ze względu na jej szybkie uczenie natomiast, jako funkcja kosztów categorical crossentropy. W warstwie wynikowej w celu klasyfikacji liczby, wykorzystana została funkcja aktywacyjna softmax. Otrzymany model został otrzymany poprzez trenowanie sieci przez 20 epok o rozmiarze batch'a równego 100.

Pochodzenie danych

Konwulsyjna sieć neuronowa w celu dobrania modelu rozpoznającego cyfry z sudoku, była uczona na podstawie zbioru ręcznie pisanych cyfr MNIST, będącego dostępnym z poziomu API biblioteki Keras. Zdjęcia plansz sudoku, natomiast zostały przygotowane przez nas.

Skuteczność

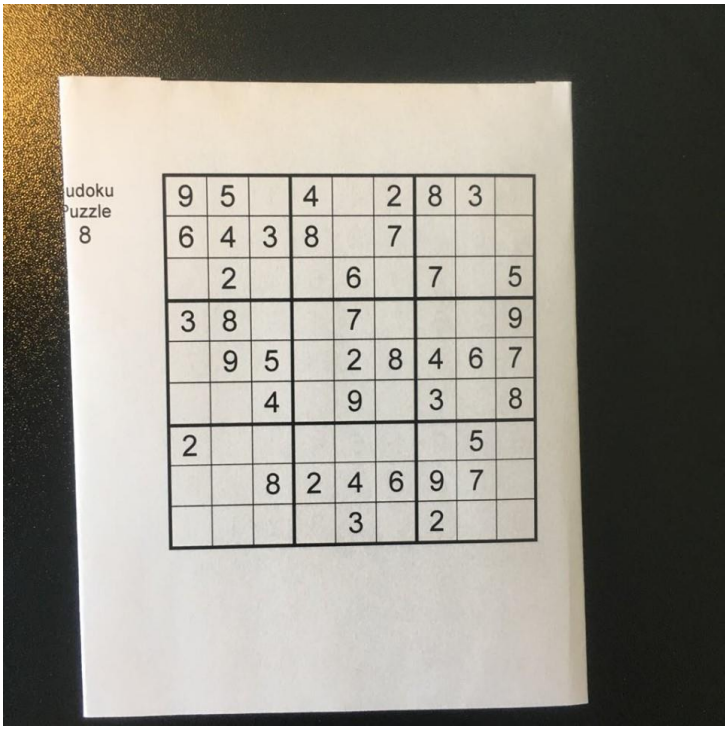
Macierz błędów

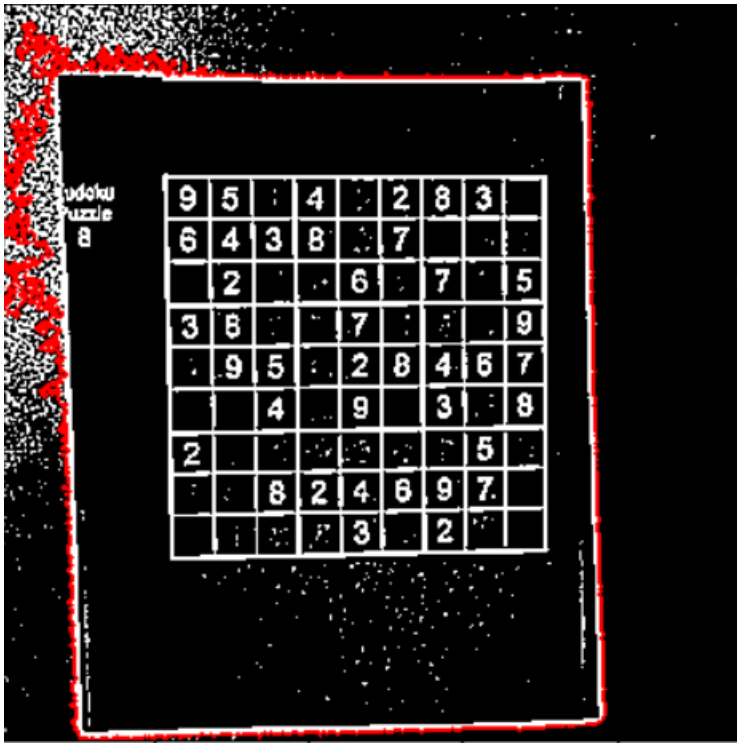
	1	2	3	4	5	6	7	8	9
Poprawnie sklasyfikowana	20	35	33	37	33	22	28	40	18
Błędnie sklasyfikowana	13	0	0	2	0	14	5	0	8
Niezakwalifikowana jako cyfra	0	0	0	0	0	0	0	0	0
Zakwalifikowana jako 8	0	0	0	0	0	14	0	x	0
Zakwalifikowana jako 7	10	0	0	0	0	0	x	0	0
Zakwalifikowana jako 3	0	0	x	0	0	0	0	0	8

Na podstawie powyższych danych można wysnuć wniosek, że stosowane przez nas metody rozpoznawania cyfr w znacznym stopniu wykonują swoje zadania poprawnie. Z cyframi 2,3,5 i 8 nie mają żadnego problemu i ich skuteczność wynosi 100%. Pomyłki pojawiają się natomiast przy rozpoznawaniu liczb 4 oraz 7, ich odsetek jest jednak bardzo niewielki. Spory problem występuje niestety przy klasyfikacji 1, 6 i 9 kiedy nawet 30% cyfr poddanych detekcji jest mylone z innymi cyframi. Podane cyfry najczęściej są mylone z kolejno: 7, 8 oraz 3.

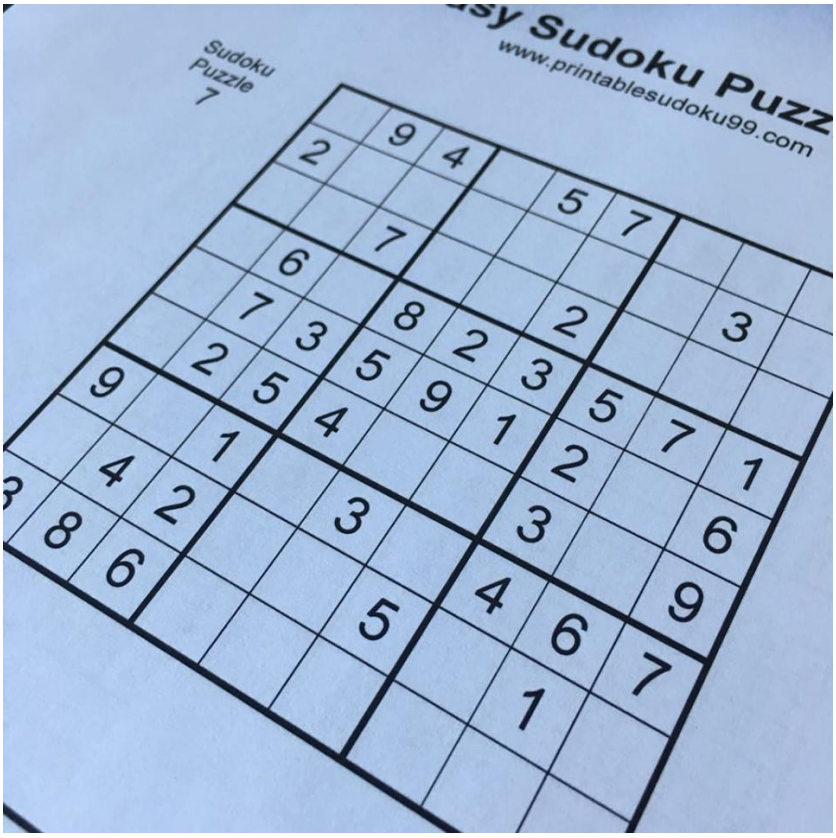
Napotykane trudności

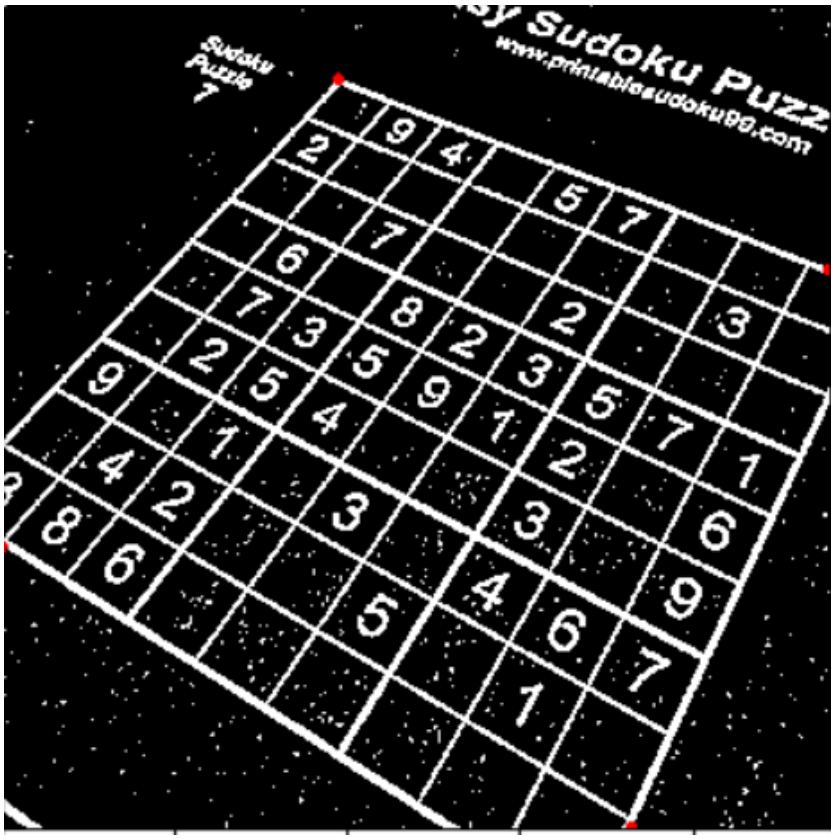
Nasz algorytm znajduje pierwotnie kontur obejmujący największą powierzchnię i zakłada, że jest to zewnętrzna krawędź planszy, dlatego w przypadku zdjęcia gdzie tło stanowią ciemne elementy, nie radzi sobie on z detekcją właściwej krawędzi. Jest to zauważalne na poniższym przykładzie.



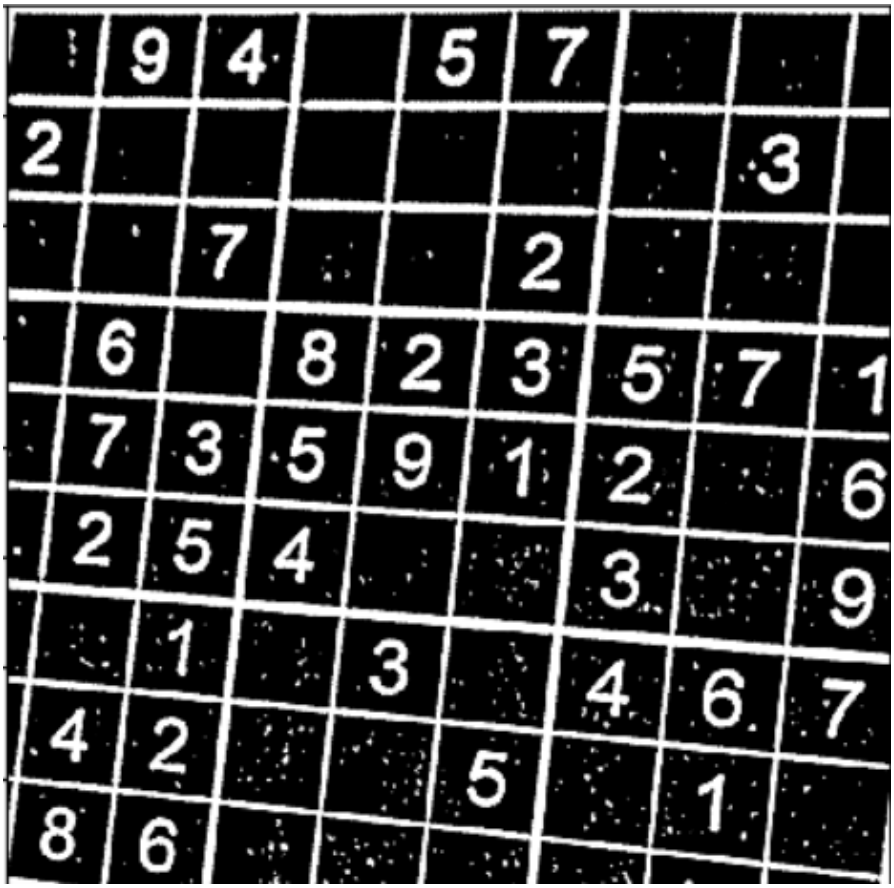


Dodatkowo po wykryciu zewnętrznej krawędzi planszy, obliczane są skrajne wierzchołki, zatem w przypadku zdjęć, gdzie wierzchołki są przycięte, algorytm nie jest w stanie prawidłowo rzutować planszy na odpowiednią perspektywę.





Przez co finalnie po transformacji, przetwarzany obraz może wyglądać następująco:



Wnioski

Na wejściu naszego programu podajemy jedynie plansze sudoku ułatwiając nieco zadanie programowi. Gdybyśmy chcieli go nieco usprawnić na szerszy użytek, moglibyśmy również pokusić się o napisanie mechanizmu sprawdzającego, czy podane na wejściu zdjęcie jest na pewno planszą sudoku. Jednak do wykonania zadania na stosowanych przez nas planszach nie jest to konieczne.

Napisany przez nas program w zdecydowanej większości poprawnie rozpoznaje cyfry znajdujące się na planszy sudoku. Z trzema cyframi występują pewne problemy, co niestety ogranicza możliwość poprawnego rozwiązania gry. Żywimy jednak przekonanie, że po przetestowaniu efektywności większej liczby modeli lub zastosowaniu poszerzonej bazy danych, jest możliwość poprawienia algorytmu, tak żeby mylił się jeszcze rzadziej.