Wydział Elektryczny
POLITECHNIKA WARSZAWSKA

Systemy Wizyjne
Politechnika Warszawska
Wydział Elektryczny
Automatyka i Robotyka Stosowana, II stopień

Rozpoznawanie śrub z użyciem głębokich sieci neuronowych

Opiekun: dr inż. Sławomir Skoneczny

Członkowie zespołu:
Nadia Kerrouche
Magda Koprowska

Wstęp

Tematem niniejszego projektu jest stworzenie systemu rozpoznawania śrub z wykorzystaniem głębokich sieci neuronowych w środowisku Matlab. Aby stworzyć tego rodzaju system, należy wytrenować wybraną sieć głęboką wykorzystując reprezentatywny zbiór uczący.

Zbiór uczący powinien zostać stworzony ze zbioru obrazów reprezentujących poszczególne rodzaje (klasy) obiektów, które będą wykrywane i rozpoznawane. Zbiór ten powinien być obszerny oraz zróżnicowany, ponieważ od tego istotnie zależą wyniki wytrenowania sieci.

Trenowanie sieci głębokich jest tzw. uczeniem z nauczycielem, tzn. sieć uczy się rozpoznawać obiekty poszczególnych klas bazując na wcześniej przedstawionych jej obrazach zawierających informacje, do której klasy należy obiekt znajdujący się na obrazie. Aby stworzyć tego rodzaju obrazy obiektów, należy przeprowadzić proces etykietyzacji surowego zbioru obrazów obiektów.

Zaetykietyzowany zbiór obrazów powinien zostać podzielony na zbiór uczący oraz zbiór walidacyjny.

- Zbiór uczący zostanie podany na wejście sieci, która będzie dobierała iteracyjnie wagi poszczególnych neuronów ją tworzących, na podstawie etykiet zawartych w tym zbiorze.

- Zbiór walidacyjny będzie wykorzystywany do ewaluacji sieci, w celu sprawdzenia poprawności dobranych w procesie iteracyjnym wag.

Posiadając zaetykietyzowany zbiór obrazów z wyodrębnionym zbiorem uczącym, należy przeprowadzić proces uczenia sieci. Niezwykle istotnym aspektem są parametry sieci oraz jej uczenia - rozmiar obrazów wejściowych, rozmiar partii, ilość warstw sieci, architektura sieci czy ilość epok uczenia. Wszystkie te parametry wpływają istotnie na czas uczenia i jego wyniki.

Proces uczenia sieci oraz wyniki jej działania można oceniać stosując standardowe metryki ewaluacyjne - precyzje, czułość oraz średnią precyzję, jak również z wykorzystaniem dostępnych w środowisku Matlab narzędzi.

Projekt systemu wizyjnego został podzielony na poszczególne etapy przedstawione poniżej.

Etapy realizacji projektu:

1. Stworzenie zbioru obrazów reprezentujących poszczególne klasy obiektów.
2. Przeprowadzenie procesu etykietyzacji zbioru obrazów.
3. Stworzenie zbioru uczącego.
4. Wytrenowanie głębokich sieci neuronowych do detekcji obiektów.
5. Ewaluacja wyników trenowania
6. Ewaluacja wyników detekcji.

1. Stworzenie zbioru obrazów reprezentujących poszczególne klasy obiektów.

Stworzony przez autorów zbiór reprezentuje 5 klas - 5 rodzajów śrub, które różnią się od siebie długością, średnicą oraz wykonaniem główki śruby. Wybrane elementy zbioru zostały dobrane tak, aby w jak najbardziej się różniły, jednocześnie zwiększając szanse na poprawną klasyfikację. Dla każdej klasy zostało wykonane 50 zdjęć w rozdzielczości 3048 na 2048 pikseli, które zapisane w formacie jpg.

Zdjęcia zostały wykonane z różnych perspektyw, aby tworzyły możliwie najbardziej reprezentatywny zbiór obrazów danej klasy. Starając się przy tym aby fotografowany obiekt na wszystkich zdjęciach zajmował podobne pole w kadrze.

Niestety, pomimo próby uniknięcia zjawiska cienia obiektu na zdjęciu, nie udało się go całkowicie wyeliminować.

Poniżej zostały przedstawione przykłady zdjęć obiektów reprezentujących poszczególne klasy.

- **Klasa 1**

Rysunek 1. Przykład obrazu reprezentującego obiekt klasy 1.

- **Klasa 2**

Rysunek 2. Przykład obrazu reprezentującego obiekt klasy 2.

- **Klasa 3**

Rysunek 3. Przykład obrazu reprezentującego obiekt klasy 3.

- **Klasa 4**

Rysunek 4. Przykład obrazu reprezentującego obiekt klasy 4.

- **Klasa 5**

Rysunek 5. Przykład obrazu reprezentującego obiekt klasy 5.

2. Przeprowadzenie procesu etykietyzacji zbioru obrazów poszczególnych klas.

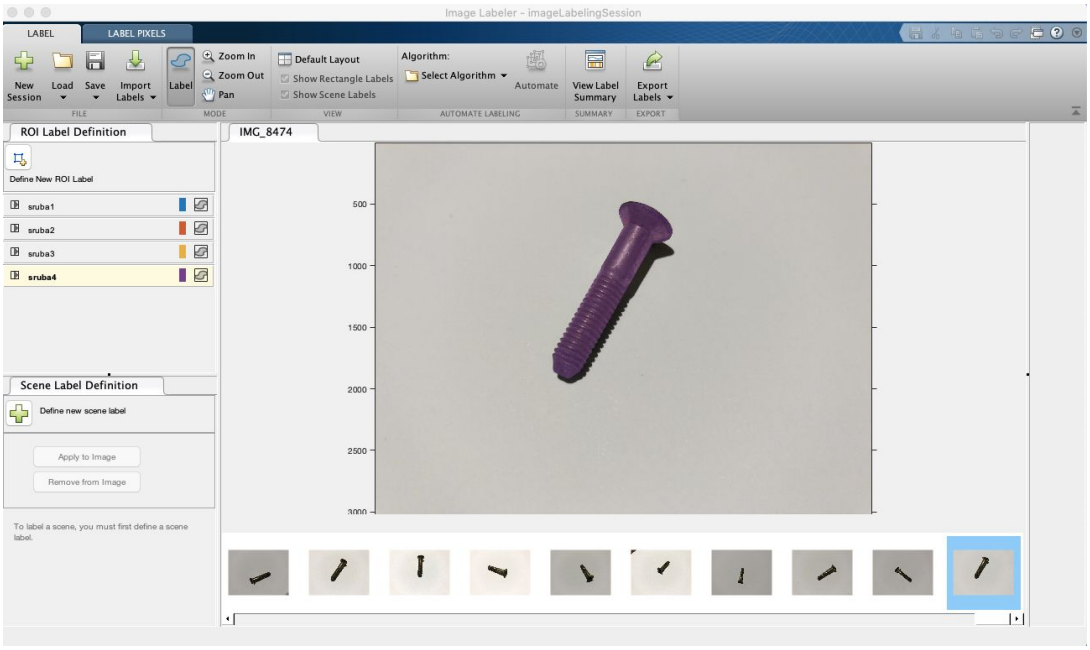
Proces etykietyzacji został przeprowadzony z wykorzystaniem aplikacji Image Labeler dostępnej z poziomu menu Matlaba.

Aplikacja ta pozwala na załadowanie zbioru zdjęć obiektów ze wskazanego przez użytkownika folderu, a następnie zdefiniowanie etykiet dla poszczególnych klas i przeprowadzenie procesu etykietyzacji.

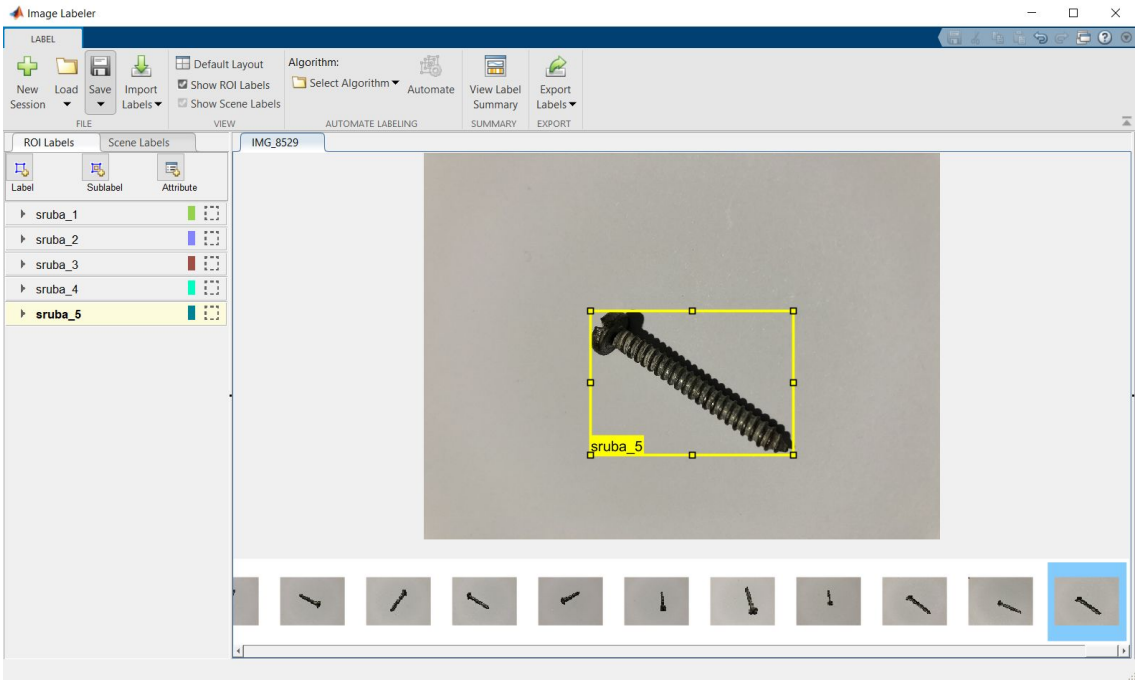
Możliwe rodzaje etykietyzacji to bazująca na oznaczaniu pikseli, z wykorzystaniem prostokątów okalających obiekt oraz etykietyzacja obiektów w kształcie linii.

W tym etapie zostały stworzone dwa zaetykietyzowane zbiory - z pomocą etykietowania pikselowego i z wykorzystaniem prostokątów, aby posiadać dowolność przy wybieraniu sieci głębokiej.

Poniżej zostały pokazane dwa obrazy przedstawiające proces etykietyzacji.



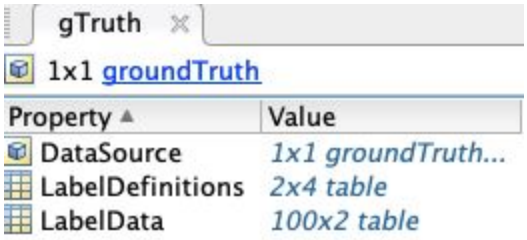
Rysunek 6. Przykład etykietowania pikselowego w aplikacji Image Labeler.



Rysunek 8. Przykład etykietowania z wykorzystaniem prostokątów okalających obiekt.

W przypadku etykietowania pikselowego wyjściowy zapis danych wraz z etykietami jest strukturą składającą się z dwóch macierzy i jednej, kolejnej struktury, co znacznie utrudnia odwołanie się do konkretnych obiektów wraz z ich etykietami w procesie uczenia sieci.

Elementy zaetykietowane metodą nakładania na obiekty prostokątów z etykietą odpowiadającą danej klasie elementu zostały zapisane w tablicy gTruth 250x6.



Rysunek 7. Podgląd struktury gTruth.

gTruth2

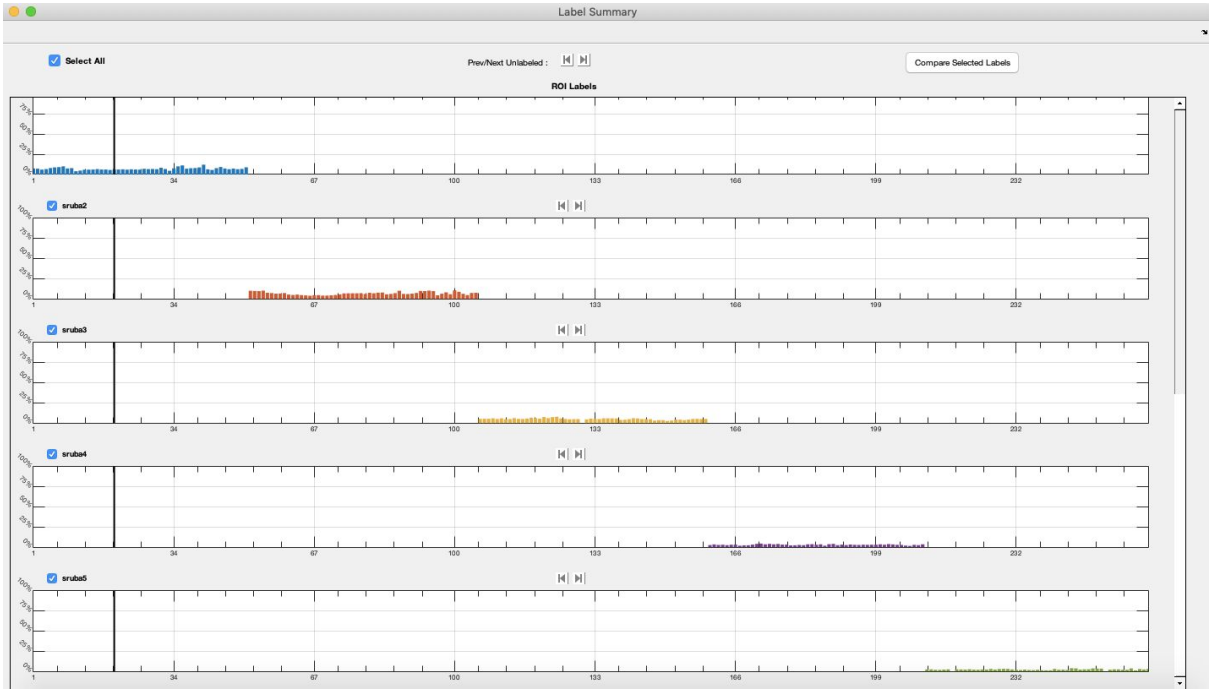
154x4 table

	1 imageFilename	2 sruba_1	3 sruba_2	4 sruba_3
1	'/Users/nadiakerrouche/śruby/śruba1	[713,529,2649,2077]	[]	[]
2	'/Users/nadiakerrouche/śruby/śruba1	[1060,256,2023,2677]	[]	[]
3	'/Users/nadiakerrouche/śruby/śruba1	[1414,86,1322,2888]	[]	[]
4	'/Users/nadiakerrouche/śruby/śruba1	[1414,168,1533,2608]	[]	[]
5	'/Users/nadiakerrouche/śruby/śruba1	[692,386,2370,2336]	[]	[]
6	'/Users/nadiakerrouche/śruby/śruba1	[365,815,3236,1784]	[]	[]
7	'/Users/nadiakerrouche/śruby/śruba1	[100,1564,3562,960]	[]	[]
8	'/Users/nadiakerrouche/śruby/śruba1	[59,1571,3385,1382]	[]	[]
9	'/Users/nadiakerrouche/śruby/śruba1	[692,440,2343,2370]	[]	[]
10	'/Users/nadiakerrouche/śruby/śruba1	[1326,188,1042,2745]	[]	[]

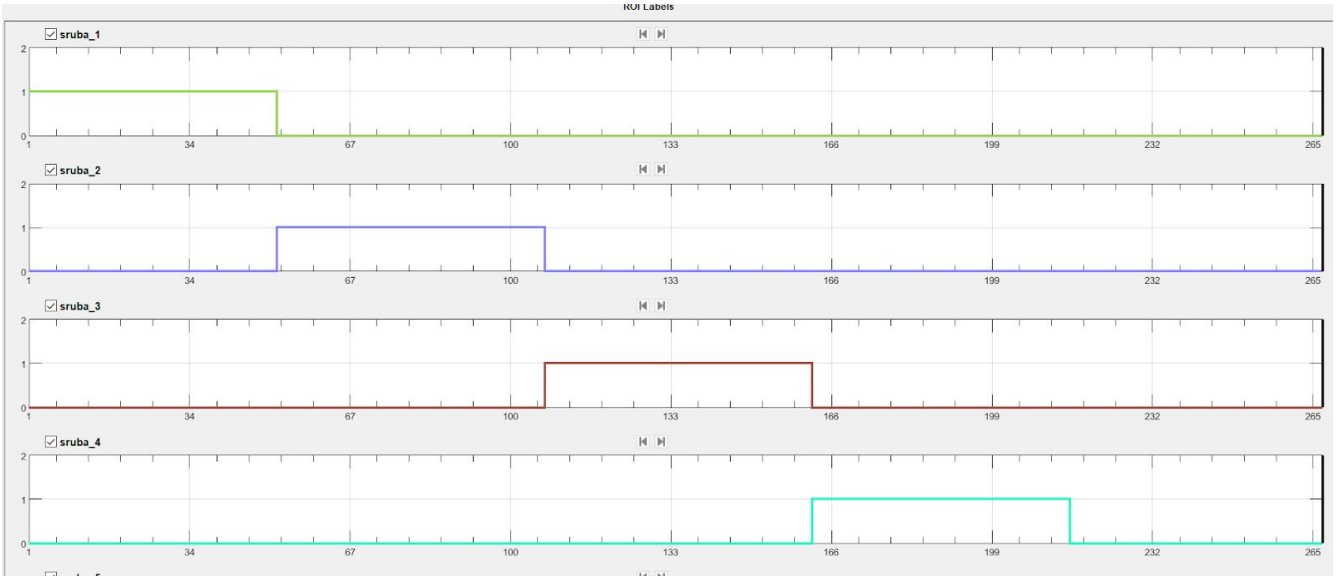
Rysunek 9. Fragment tablicy gTruth.

Aplikacja Image Labeler oferuje również możliwość prześledzenia podsumowania etykietowania, na podstawie którego możemy stwierdzić, na ilu obrazach znajduje się obiekt danej klasy oraz czy wszystkie obrazy zostały zaetykietowane.

Wynik podsumowania procesu etykietyzacji pikselowej oraz z wykorzystaniem prostokątów okalających znajdują się poniżej.



Rysunek 10. Podsumowanie procesu etykietyzacji pikselowej w aplikacji Image Labeler.



Rysunek 11. Podsumowanie procesu etykietyzacji z wykorzystaniem prostokątów okalających w aplikacji Image Labeler.

3. Stworzenie zbioru uczącego.

Stworzenie zbioru uczącego polegało na wykorzystaniu obiektu *gTruth table* z zaetykietowanymi danymi stworzonego na poprzednim etapie oraz podzieleniu go na - obrazy oraz etykiety przeznaczone na zbiór uczący i obrazy wraz z etykietami przeznaczone na zbiór walidacyjny. Zbiór podzielono w stosunku 6:4.

Następnie za pomocą funkcji imageDataStore oraz *labelDataStore* stworzono obiekty typu *DataStore* przechowujące obrazy oraz etykiety, które zostały wykorzystane do stworzenia obiektu typu *CombinedData* przy pomocy funkcji combine.

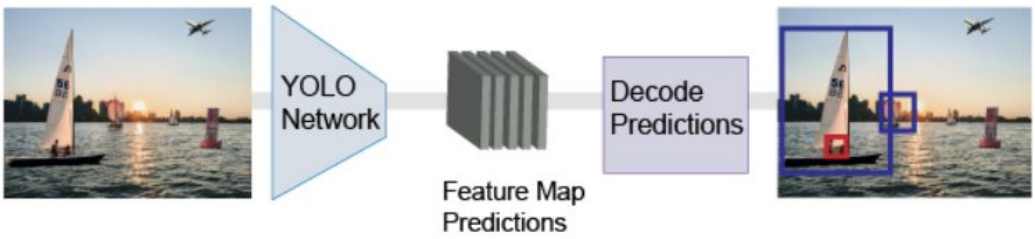
Obiekt typu *CombinedData* jest parametrem wejściowym funkcji do trenowania głębokiej sieci neuronowej.

4. Wytrenowanie głębokiej sieci neuronowej do wykrywania i rozpoznawania obiektów.

Posiadając dwa zaetykietyzowane zbiory - pierwszy metodą pikselową i drugi przy pomocy prostokątów okalających możliwe było wykorzystanie dwóch algorytmów *DeepLab v3* bazującego na segmentacjji semantycznej i *YOLOv2*.

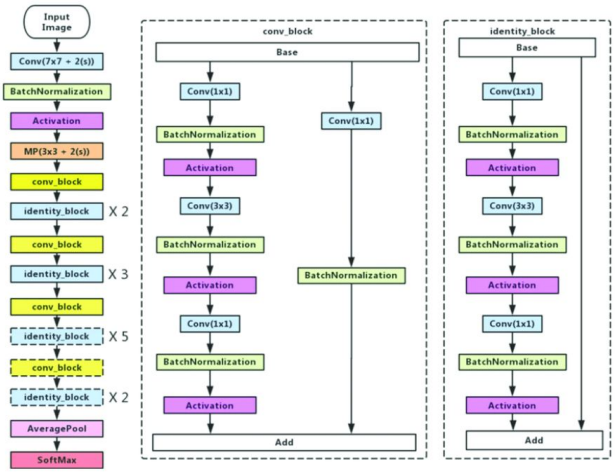
Wybranym algorytmem detekcji został *YOLOv2 (You Only Look Once).* Algorytm ten wykrywa obiekty jednoetapowo, czym różni się od standardowych algorytmów opartych na dwóch etapach - tj. *R-CNN*.

Algorytm *YOLOv2* jednocześnie przeprowadza proces klasyfikacji i wyodrębniania ramek okalających (ang. *bounding box*) w przeciwieństwie do algorytmów opartych na *R-CNN*, które wykonują te zadania dwuetapowo.



Rysunek. 12. Proces wyodrębniania ramek okalających przez algorytm YOLOv2.
Źródło:https://www.mathworks.com/help/vision/ug/getting-started-with-yolo-v2.html#mw_bcbd5375-ed75-4616-b57b-469e29139033

Jako baza został wybrany wstępnie wytrenowany rdzeń *ResNet-50*.



Rysunek 12. Architektura rdzenia ResNet-50. Źródło:
https://www.researchgate.net/figure/Left-ResNet50-architecture-Blocks-with-dotted-line-represents-modules-that-might-be_fig3_331364877

Jako warstwa *ReLU* została wybrana warstwa '*activation_40_relu*'.

Pierwszym etapem tego punktu było określenie rozmiaru obrazów, które są danymi wejściowymi dla sieci neuronowej. Rozmiary te zdefiniowano w rozmiarze 224x224 px. Określenie poprawnego rozmiaru obrazów wejściowych jest niezwykle istotne, ponieważ ma on wpływ na szybkość i wyniki wytrenowania sieci.

Następnie zostały określone parametry trenowania sieci:

- *InitialLearnRate* - wartość określona wyjściowo jako 0.01. Jeżeli jest za mała, uczenie wymaga dużej ilości czasu, jeżeli jest za duża, uczenie może osiągnąć nieoptymalny wynik lub nie być zbieżne.

- *Verbose* - określa, czy wyniki są wyświetlane cyklicznie w oknie poleceń.

- *MiniBatchSize* - rozmiar partii obrazów ze zbioru uczącego, który jest podawany na wejście sieci przy każdej iteracji. Służy on do oceny gradientu funkcji błędu i aktualizacji wag neuronów sieci.

- *MaxEpochs* - maksymalna liczba epok uczenia, które zostaną przeprowadzone podczas jednej sesji.

- *VerboseFrequency* - częstotliwość wyświetlania - określa liczbę iteracji między wyświetlaniem wyników kolejnych checkpointów w oknie poleceń.

- *Checkpointpath* - ścieżka do zapisywania 'punktów kontrolnych' sieci.

Przykładowa konfiguracja parametrów w środowisku Matlab została przedstawiona poniżej:

```
options = trainingOptions('sgdm',...
          'InitialLearnRate',0.001,...
          'Verbose',true,...
          'MiniBatchSize',16,...
          'MaxEpochs',30,...
          'Shuffle','never',...
          'VerboseFrequency',30,...
          'CheckpointPath',tempdir);
```

Rysunek 13. Konfiguracja opcji trenowania sieci.

Przy pomocy zapisywanych cyklicznie punktów kontrolnych (ang. checkpoints) możemy na bieżąco śledzić proces trenowania sieci - głównie wykres błędu trenowania od ilości przeprowadzonych iteracji.

Pozwala to na monitorowanie bieżącej wartości błędu, jak również wykrycie niepożądanych zjawisk np. przeuczenia sieci, kiedy błąd na zbiorze uczącym maleje, natomiast błąd na zbiorze testowym zaczyna rosnąć.

Poniżej został przedstawiony proces trenowania wyświetlany w oknie poleceń.

```
Training on single CPU.
Initializing input data normalization.
|========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |     RMSE     |     Loss     |      Rate       |
|========================================================================================|
|       1 |           1 |       00:02:08 |         9.05 |         81.9 |          0.0010 |
|       8 |          50 |       00:45:20 |         3.41 |         11.6 |          0.0010 |
|      15 |         100 |       01:27:34 |         3.79 |         14.3 |          0.0010 |
|      20 |         140 |       01:59:46 |         1.37 |          1.9 |          0.0010 |
|========================================================================================|
Detector training complete.
************************************************************************
```

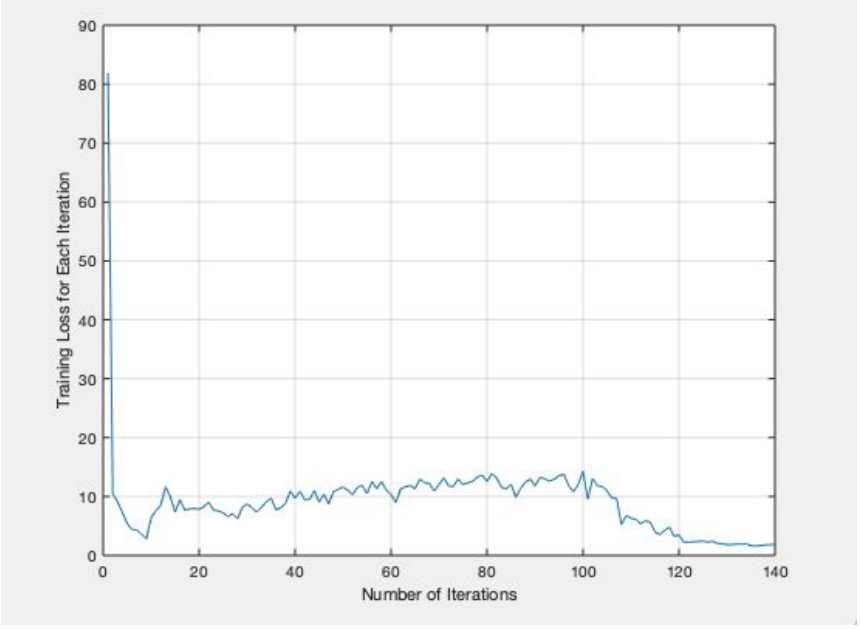
Rysunek 14. Proces trenowania detektora YOLOv2 opartego na rdzeniu ResNet-50.

Podczas procesu trenowania dostajemy na bieżąco informacje m.in. o liczbie przeprowadzonych epok trenowania, liczbie iteracji, czasu trwania trenowania, czy błędzie średniokwadratowym (ang. *RMSE - Root Mean Square Error*).

5. Ewaluacja wyników trenowania.

Ewaluacja wyników trenowania została przeprowadzona na podstawie wyników klasyfikacji wyodrębnionego wcześniej zbioru testowego (walidacyjnego) w procesie tworzenia zbioru uczącego.

W pierwszej kolejności została przeanalizowana zależność parametru błędu trenowania (ang. Training Loss) w zależności od liczby iteracji (ang. Number of Iterations).



Rysunek 15. Wykres zależności błędu trenowania od ilości iteracji.

Wyniki wskazują, iż z każdą przeprowadzoną iteracją uczenia, błąd malał, co oznacza, że wagi poszczególnych neuronów sieci określane w każdej iteracji coraz bardziej zbliżały się do wartości oczekiwanych.

Co warte uwagi, w pierwszych iteracjach błąd gwałtownie spadł, następnie utrzymywał się na podobnym poziomie, a nawet nieznacznie wzrósł, aby przy około 120 iteracji znacząco zmaleć i utrzymywać się na niskim poziomie aż do końcowej 140 iteracji. Oznacza to, iż ilość iteracji (epok) została dobrana poprawnie.

Aby móc ocenić wydajność klasyfikacji, stosuje się tablice pomyłek - tabele złożoną z dwóch wierszy i dwóch kolumn. Wiersze przedstawiają klasy predykowane a kolumny klasy rzeczywiste.

		klasa rzeczywista	
		pozytywna	**negatywna**
klasa predykowana	**pozytywna**	prawdziwie pozytywna (TP)	fałszywie pozytywna (FP)
	negatywna	fałszywie negatywna (FN)	prawdziwie negatywna (TN)

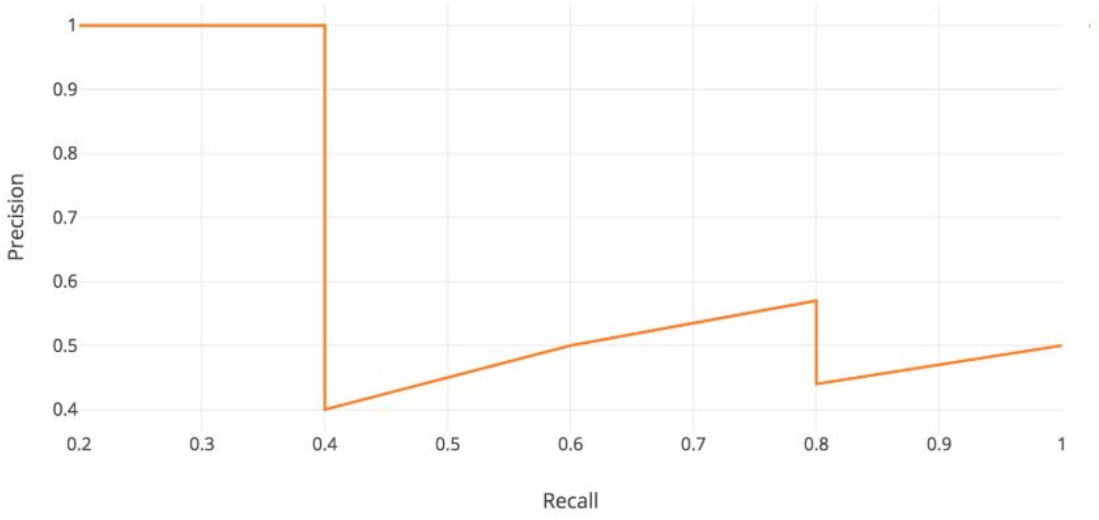
Rysunek 16. Tablica pomyłek. Źródło: pl.wikipedia.org.

Na podstawie prawdziwych i nieprawdziwych klasyfikacji można wyróżnić szereg miar oceniających wydajność danej klasyfikacji:

- Precyzja (ang. Precision) jest to stosunek wyników prawdziwie pozytywnych do sumy wyników prawdziwie pozytywnych oraz fałszywie pozytywnych. Informuje ona o tym, jak zgodne są jego wyniki w stosunku do innych pomiarów tej wielkości.

- Czułość (ang. Recall) jest to stosunek wyników prawdziwie pozytywnych do sumy wyników prawdziwie pozytywnych i fałszywie negatywnych. Informuje ona o tym, jak dobra jest jego metoda w znajdowaniu wyników pozytywnych.

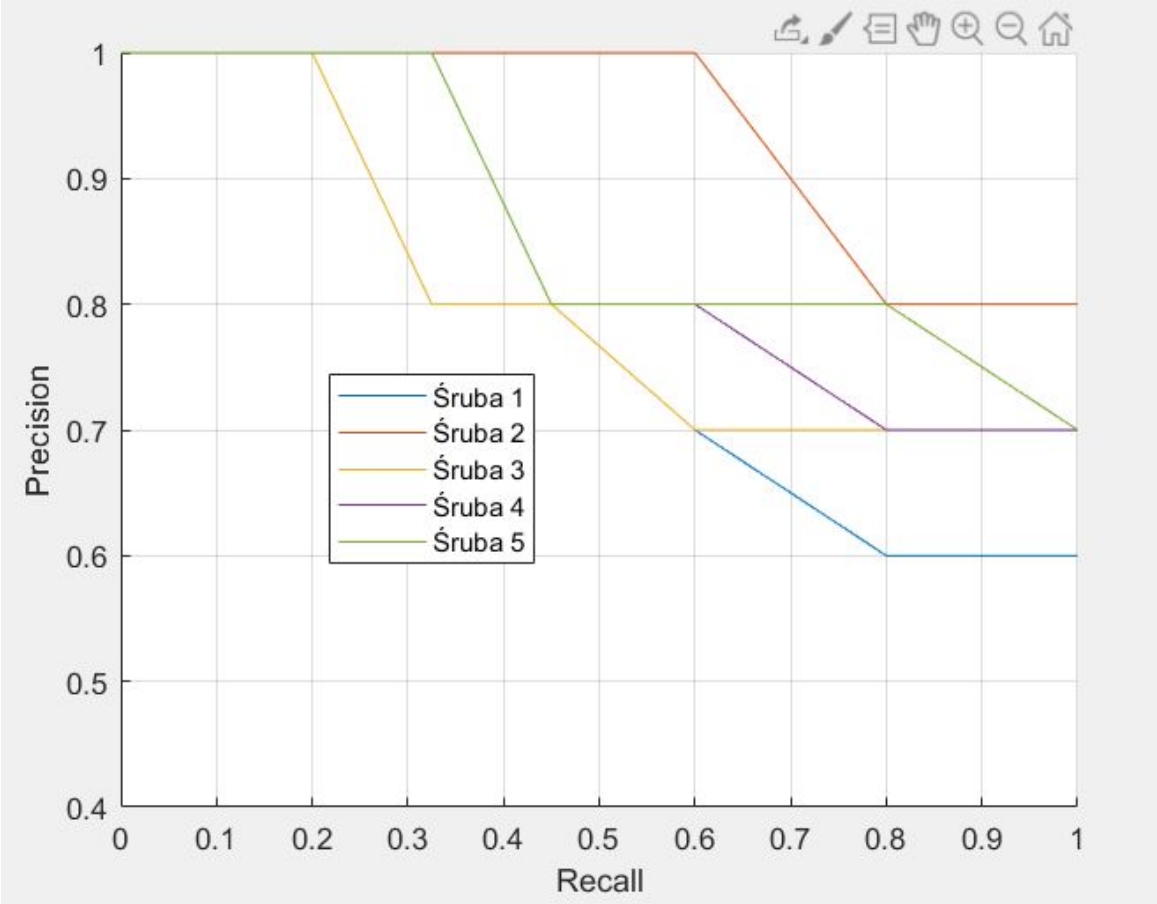
- Średnia precyzja (ang. Average Precision) jest to wskaźnik zawierający się w granicach 0-1 określający precyzję w funkcji czułości.



Rysunek 17. Przykładowy wykres średniej precyzji. Źródło: https://medium.com/@jonathan_hui/map-mean-aver

age-precision-for-object-detection-45c121a31173.

Wyniki precyzji, czułości i średniej precyzji uzyskane przez wytrenowany klasyfikator prezentują się następująco:



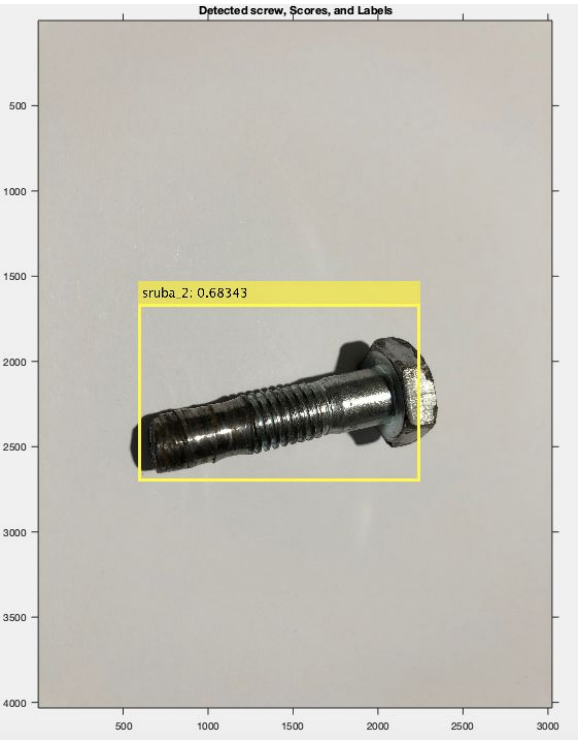
Rysunek 18. Wykres precyzji w funkcji czułości dla obiektów poszczególnych klas.

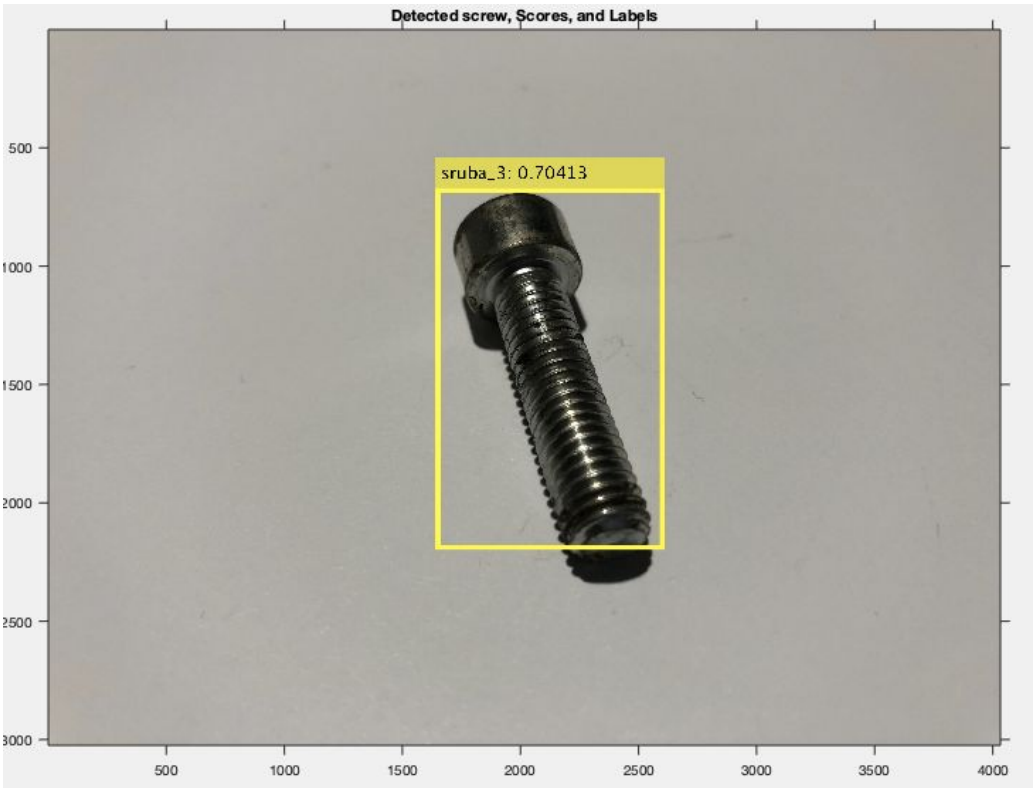
Klasa	Wartość AP (ang. *Average Precision*)
Śruba 1	0.25
Śruba 2	0.47
Śruba 3	0.23
Śruba 4	0.33
Śruba 5	0.27

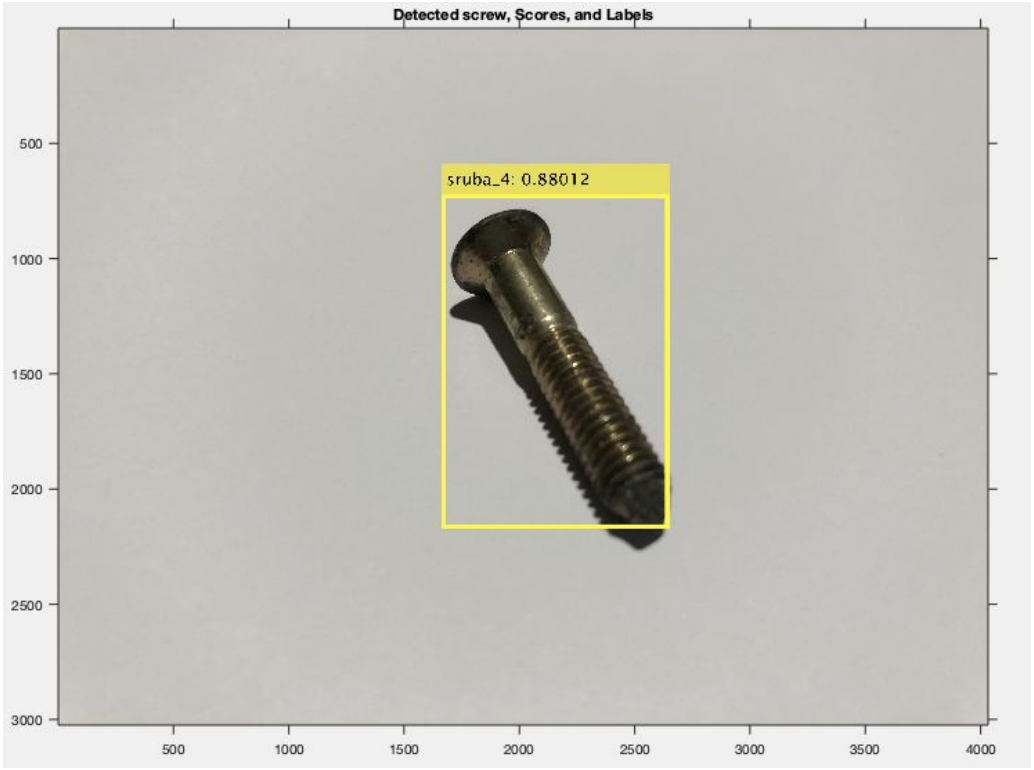
Tabela 1. Tabela prezentująca wyniki wartości średniej precyzji dla poszczególnych klas zbioru.

6. Ewaluacja wyników detekcji.

Ewaluacja wyników detekcji została przeprowadzona na podstawie obrazów ze zbioru testowego. Przeprowadzenie tego procesu miało na celu ocenę wykrywania poszczególnych obiektów oraz ocenę poprawności zakwalifikowania obiektów znajdujących się na obrazie. Wyniki zostały przedstawione na zdjęciach poniżej.







Rysunek 19. Wyniki detekcji na kilku obrazach ze zbioru testowego.

Jak widać na przedstawionych powyżej obrazach, sieć poprawnie sklasyfikowała wybrane obiekty, jednak wyniki nie są idealne.

Ramki okalające (ang. bounding box) nie stanowią idealnych obrysów obiektów - są one częściowo przesunięte - jak również procentowy wynik klasyfikacji kształtuje się na poziomie około 60 %.

Najlepsze wyniki sieć osiągnęła dla śruby nr 4. Prawdopodobnie jest to związane z jej charakterystycznym wyglądem - ma ona inną główkę niż pozostałe śrubki oraz jest dość krótka.

Drugim prawdopodobnym czynnikiem lepszego współczynnika klasyfikacji tej śruby może być większe zróżnicowanie obrazów zbioru uczącego, choć autorzy starali się wyeliminować te różnice w procesie tworzenia zbioru.

Podsumowanie

Uzyskane przez autorów wyniki wytrenowania sieci oraz osiągane przez nią wyniki detekcji na obrazach testowych są zadowalające.Również wartości standardowych metryk ewaluacyjnych zaimplementowanych w procesie ewaluacji wytrenowania sieci - precyzji, czułości oraz średniej czułości kształtują się na akceptowalnym poziomie.

W celu poprawy uzyskanych wyników należałoby zwiększyć rozmiar zbioru uczącego, całkowicie usunąć występujące na zdjęciach cienie oraz bardziej zróżnicować sceny zbioru.

Autorzy, oprócz zaetykietyzowania zbioru uczącego wykorzystując prostokąty okalające, stworzyli zbiór zaetykietyzowany pikselowo. W procesie wytrenowania sieci został wykorzystany pierwszy z nich, jednak ze względu na dokładność określenia przynależności poszczególnych pikseli do obiektu konkretnej klasy, etykietyzacja pikselowa mogłaby lepiej spełnić swoją rolę i pozwolić na stworzenie sieci dającej lepsze wyniki w procesie ewaluacji. Wiązałoby się to z wybraniem innej architektury sieci i prawdopodobnie wyższym czasem trenowania, ze względu na większą ilość danych.

Źródła

- https://www.mathworks.com/help/deeplearning/ug/object-detection-using-yolo-v2.html?s_tid=blogs_rc_6
- https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173
- https://www.researchgate.net/figure/Left-ResNet50-architecture-Blocks-with-dotted-line-represents-modules-that-might-be_fig3_331364877
- https://www.mathworks.com/help/vision/ug/getting-started-with-yolo-v2.html#mw_bcbd5375-ed75-4616-b57b-469e29139033
- https://www.mathworks.com/help/vision/ref/selectstrongestbboxmulticlass.html