Projekt Własny CPO

# Krótkie wprowadzenie teoretyczne do deep learning

Deep learning to poddziedzina uczenia maszynowego, koncentruje się na algorytmach wzorowanych na strukturze mózgu, zwanych sztucznymi sieciami neuronowymi (artificial neural networks). W przeciwieństwie do tradycyjnego uczenia maszynowego, które często wymaga ręcznego wyodrębniania cech, metody głębokiego uczenia automatycznie odkrywają reprezentacje potrzebne do wykrywania lub klasyfikacji cech na podstawie surowych danych. Jej rdzeniem są głębokie sieci neuronowe (DNN) składające się z wielu warstw wzajemnie połączonych neuronów. Uczenie obejmuje dostosowywanie wag połączeń w celu zminimalizowania funkcji strat.

Uczenie głębokie odniosło niezwykły sukces w takich dziedzinach, jak widzenie komputerowe, przetwarzanie języka naturalnego i uczenie się przez wzmacnianie. Konwolucyjne sieci neuronowe (CNN) doskonale radzą sobie z zadaniami związanymi z obrazem dzięki warstwom splotowym, podczas gdy rekurencyjne sieci neuronowe (RNN) skutecznie radzą sobie z danymi sekwencyjnymi. Dostępność dużych zbiorów danych i zasobów obliczeniowych o wysokiej wydajności, takich jak procesory graficzne, napędza rozwój coraz bardziej złożonych modeli, umożliwiających osiągnięcie wydajności na poziomie ludzkim w zadaniach takich jak rozpoznawanie obrazów i tłumaczenie językowe.

# Krótki opis istniejących rozwiązań - jakie narzędzia są stosowane

Konwolucyjne sieci neuronowe (CNN) stanowią podstawę zadań związanych z obrazem. Sieci CNN wykorzystują warstwy splotowe do automatycznego i adaptacyjnego uczenia się hierarchii przestrzennych w obrazach, dzięki czemu są bardzo skuteczne w zadaniach takich jak rozpoznawanie i klasyfikacja obrazów. Do wybitnych architektur CNN należą AlexNet, VGG i ResNet, a każda z nich rozwija tę dziedzinę dzięki innowacjom w zakresie głębokości i struktury sieci.

Rekurencyjne sieci neuronowe (RNN) są przeznaczone do przetwarzania danych sekwencyjnych, co czyni je idealnymi do zadań obejmujących szeregi czasowe lub przetwarzanie języka naturalnego. RNN utrzymują formę pamięci poprzez swoje ukryte stany, co pozwala im uchwycić zależności czasowe.

Model YOLO jest znany ze swoich możliwości przetwarzania w czasie rzeczywistym, skutecznego wykrywania obiektów na obrazach poprzez dzielenie ich na siatki i bezpośrednie przewidywanie obwiedni i prawdopodobieństw klas. Dzięki temu YOLO jest bardzo skuteczny w zastosowaniach wymagających szybkiego i dokładnego wykrywania obiektów.

Tradycyjne metody, takie jak histogram zorientowanych gradientów (HOG), również odgrywają rolę w widzeniu komputerowym. Deskryptory HOG służą do wykrywania obiektów poprzez przechwytywanie rozkładu orientacji gradientów w zlokalizowanych częściach obrazu. Choć nie jest tak elastyczny jak modele głębokiego uczenia się, HOG jest skuteczny w zadaniach takich jak wykrywanie pieszych.

Zbiory danych takie jak MNIST (baza danych Zmodyfikowanego Narodowego Instytutu Standardów i Technologii) mają fundamentalne znaczenie w opracowywaniu i porównywaniu modeli głębokiego uczenia się. Zbiór danych MNIST, składający się z odręcznych obrazów cyfrowych, jest standardem oceny wydajności algorytmów klasyfikacji obrazów. Pomimo swojej prostoty pozostaje punktem odniesienia do testowania nowych algorytmów i modeli.

### Narzędzia

**Obliczeniowe** – Pandas, numpy

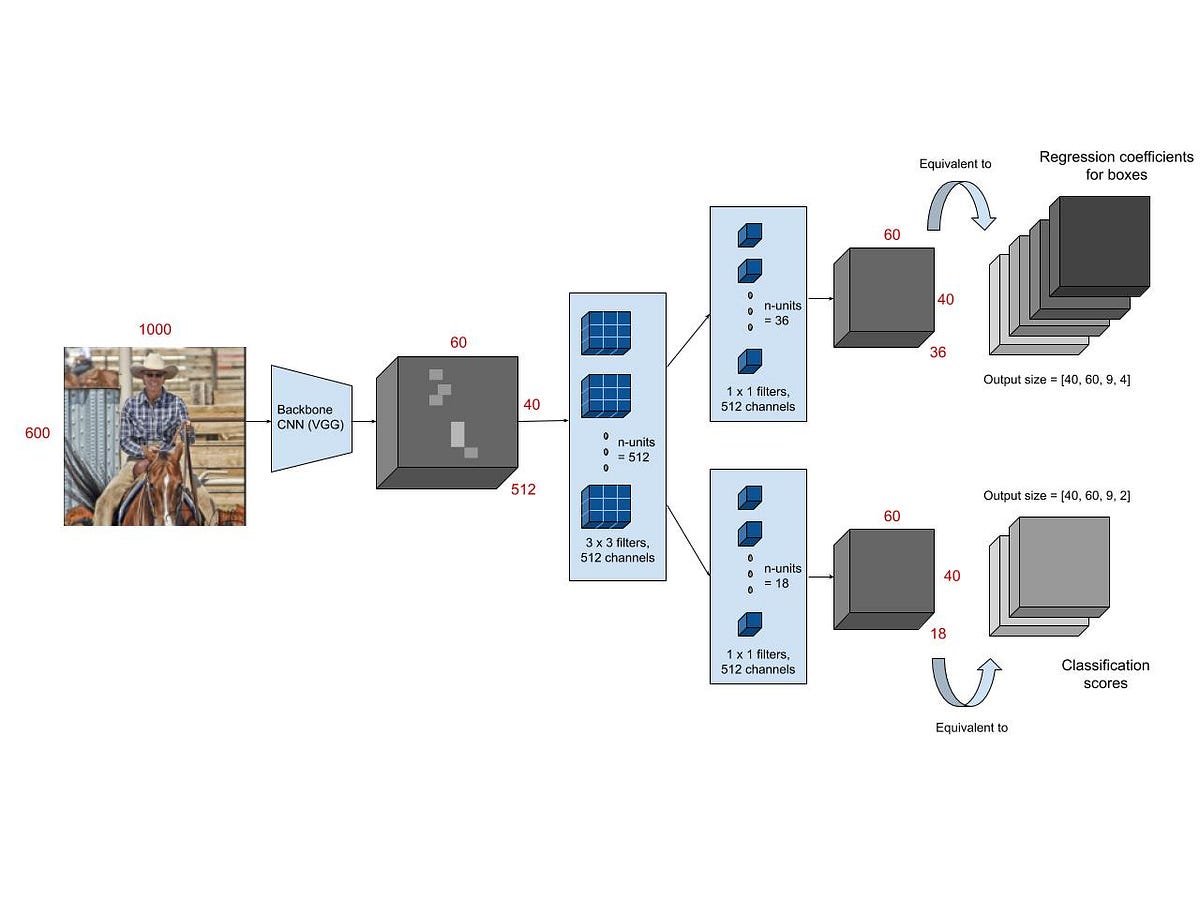
**Wykresowe/obrazowe** – cv2, matplotlib

**Deep learningowe** – keras, pytorch, scikit-learn, tensorflow

Liderami w dziedzinie są TensorFlow i PyTorch, oba open source’owe framework’i, szeroko stosowane ze względu na ich elastyczność i obszerne biblioteki. Keras, interfejs API wysokiego poziomu działający na bazie TensorFlow, upraszcza budowanie i szkolenie sieci neuronowych. Scikit-learn (sklearn) to szeroko stosowana biblioteka w ekosystemie uczenia maszynowego, która oferuje proste i wydajne narzędzia do eksploracji i analizy danych, w tym wstępnego przetwarzania, wyboru modelu i oceny.

# Opis technologii zastosowanych w projekcie ze szczególnym uwzględnieniem wybranej architektury

Wybrana architektura to Faster R-CNN (Region-based Convolutional Neural Network), model zaprojektowany specjalnie do zadań wykrywania obiektów. Architektura ta płynnie integruje sieci propozycji regionów (Region Proposal Network - RPN) z klasyfikacją obiektów, umożliwiając precyzyjną lokalizację i identyfikację obiektów na obrazach. Łącząc te komponenty, Faster R-CNN osiąga wyjątkową dokładność przy jednoczesnym zachowaniu dużej szybkości przetwarzania, co czyni go idealnym wyborem do zastosowań w czasie rzeczywistym.



1. Architektura faster R-CNN

Konwolucyjne sieci neuronowe (CNN) to klasa głębokich sieci neuronowych zaprojektowanych specjalnie do przetwarzania ustrukturyzowanych danych siatkowych, takich jak obrazy. Składają się z wielu warstw, w tym warstw splotowych, warstw łączących i warstw w pełni połączonych. Oto jak działają CNN:

### Warstwy splotowe:

Warstwy splotowe nakładają zestaw uczalnych filtrów (jąder) na obraz wejściowy. Filtry te oplatają obraz, wyodrębniając takie cechy, jak krawędzie, tekstury i kształty.

### Warstwy Łączące (Pooling):

Warstwy łączenia zmniejszają próbkowanie map obiektów tworzonych przez warstwy splotowe, redukując wymiary przestrzenne danych wejściowych. Typowe operacje łączenia obejmują maksymalne łączenie i średnie łączenie, które zachowują najważniejsze funkcje, jednocześnie zmniejszając złożoność obliczeniową.

### W pełni połączone warstwy:

W pełni połączone warstwy przetwarzają spłaszczone mapy obiektów z poprzednich warstw i wykonują zadania klasyfikacji lub regresji. Uczą się złożonych relacji między cechami, aby móc przewidywać.

Sieci CNN przodują w hierarchicznym uczeniu się cech, w którym niższe warstwy przechwytują proste cechy (np. Krawędzie), a wyższe warstwy wychwytują bardziej abstrakcyjne cechy (np. Części obiektów). Ta hierarchiczna reprezentacja umożliwia CNN osiągnięcie niezwykłej wydajności w zadaniach związanych z obrazem, w tym w klasyfikacji, segmentacji i wykrywaniu obiektów. 

2. Diagram CNN

Region Proposal Networks(RPN) to kluczowy element architektury Faster R-CNN, odpowiedzialny za generowanie propozycji regionów dla potencjalnych lokalizacji obiektów na obrazie.

### Skrzynki kotwiczne:

RPN wykorzystują zestaw predefiniowanych bloków kontrolnych o różnych rozmiarach i proporcjach, równomiernie rozmieszczonych na obrazie. Te skrzynki kontrolne służą jako pola referencyjne dla potencjalnych lokalizacji obiektów.

### Okno przesuwne:

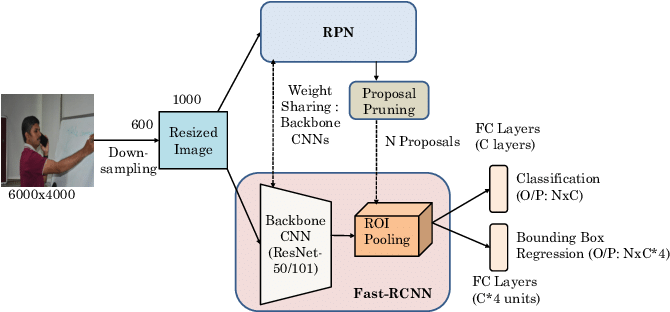
RPN przesuwa małe okienko (zwykle 3x3) na mapy obiektów utworzone przez warstwy splotowe CNN. W każdej pozycji okno wyodrębnia cechy i przewiduje, czy obiekt jest obecny, czy nie.

### Regresja ramki ograniczającej:

Oprócz oceny obiektywności RPN przewiduje również korekty (przesunięcia) współrzędnych pola zakotwiczenia, aby lepiej dopasować je do dokładnych lokalizacji obiektów. Proces ten nazywany jest regresją obwiedni.

### Tłumienie inne niż maksymalne (Non-Maximum Suppression - NMS):

Aby wyeliminować zbędne propozycje regionów, RPN stosuje niemaksymalne tłumienie w celu połączenia nakładających się propozycji i zatrzymania tylko najbardziej pewnych.



3. Diagram RPN

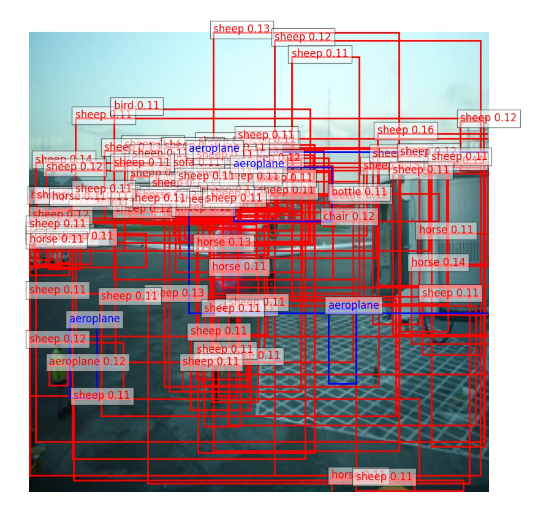
Łącząc sieci CNN do ekstrakcji cech i RPN do generowania propozycji regionów, architektura Faster R-CNN osiąga wysoką dokładność w zadaniach wykrywania obiektów. Sieci CNN skutecznie przechwytują cechy obrazu, podczas gdy sieci RPN skutecznie generują potencjalne lokalizacje obiektów, tworząc solidny i skuteczny system wykrywania obrazu.

# Trenowanie sieci – opis procesu uczenia, doboru hiperparametrów sieci, testowanie, ewaluacja rezultatów

## Proces uczenia

Nasz proces uczenia sieci rozpoczyna się od przygotowania zestawu danych składającego się z oznaczonych obrazów i odpowiednich adnotacji. Podzieliliśmy zbiór danych na zbiory szkoleniowe i walidacyjne, aby zapewnić zdolność modelu do generalizacji.

Proces uczenia wykonuje iterację po zbiorze danych dla wielu epok, przy czym każda epoka zawiera kilka partii obrazów i adnotacji. W każdej iteracji model oblicza funkcję straty, która składa się z różnych składników, w tym utraty wykrycia obiektu i składników regularyzacji. Następnie optymalizator aktualizuje parametry modelu, aby zminimalizować straty, stopniowo poprawiając wydajność modelu.



4. Identyfikacja bez treningu

## Dobór hiperparametrów

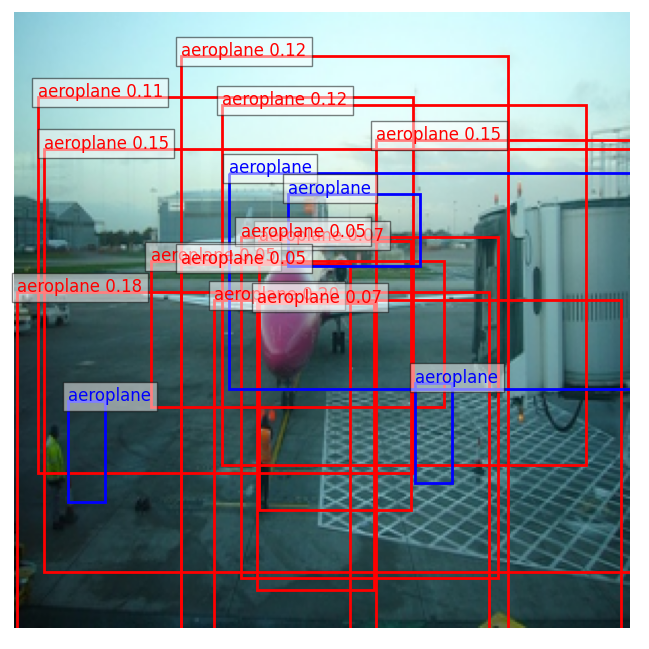
Dobór hiperparametrów sieci odgrywa kluczową rolę w określaniu wydajności modelu i szybkości konwergencji. Starannie dostrajamy hiperparametry, takie jak szybkość uczenia się, wielkość partii i siła regularyzacji, poprzez eksperymenty empiryczne i monitorowanie wydajności. Dodatkowo wykorzystujemy techniki takie jak planowanie szybkości uczenia się i wczesne zatrzymywanie, aby zapobiec nadmiernemu dopasowaniu i zapewnić optymalną zbieżność.

Parametry nr 1:

BATCH\_SIZE = 2

NUM\_EPOCHS = 5

dataset = Subset(dataset, range(100))



5. Identyfikacja z parametrami nr 1

## Testowanie i ewaluacja

Po zakończeniu uczenia oceniamy wyszkolony model na oddzielnym zestawie testowym, aby ocenić jego wydajność na niewidocznych danych. Mierzymy różne wskaźniki oceny, w tym precyzję, przypominanie i średnią precyzję średnią (mAP), aby określić ilościowo dokładność i niezawodność modelu. Dodatkowo wizualizujemy przewidywania modelu na przykładowych obrazach, aby uzyskać wgląd w jego zachowanie i potencjalne obszary wymagające poprawy.

Ocena wyników dostarcza cennych informacji na temat mocnych i słabych stron modelu. Analizujemy wydajność modelu w różnych kategoriach obiektów, identyfikując klasy, w których model może mieć problemy. Badamy również przypadki błędów, aby zrozumieć pierwotne przyczyny błędnych klasyfikacji i badamy strategie dalszego udoskonalenia.

Systematycznie powtarzając proces uczenia sieci, dostrajając hiperparametry i rygorystycznie oceniając wyniki, zapewniamy rozwój solidnego i dokładnego systemu wykrywania obrazu. Nasze podejście kładzie nacisk zarówno na eksperymenty empiryczne, jak i zrozumienie teoretyczne, co pozwala nam osiągnąć najnowocześniejszą wydajność przy jednoczesnym zachowaniu możliwości interpretacji i uogólniania.

# Bibliografia:

<https://towardsdatascience.com/faster-r-cnn-for-object-detection-a-technical-summary-474c5b857b46>

<https://www.researchgate.net/figure/Schematic-diagram-of-a-basic-convolutional-neural-network-CNN-architecture-26_fig1_336805909>

<https://www.researchgate.net/figure/Flow-Diagram-of-RPN_fig3_343685243>

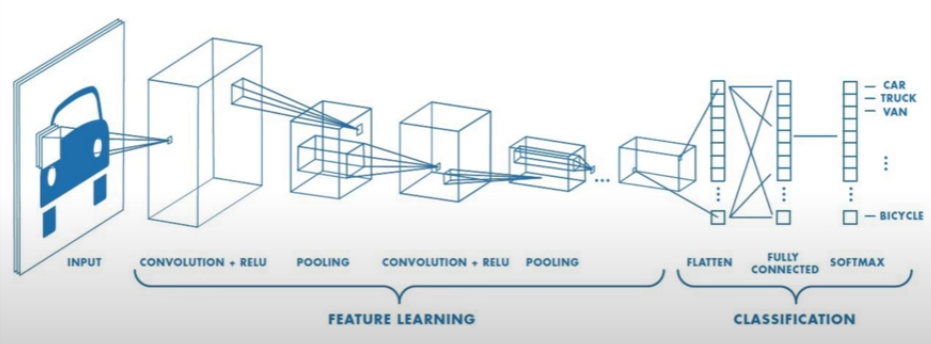
<https://www.baeldung.com/cs/learning-rate-batch-size#:~:text=The%20batch%20size%20affects%20some,and%20a%20good%20initial%20choice>.

<https://www.youtube.com/watch?v=pDdP0TFzsoQ>

<https://stackoverflow.com/questions/56023786/intuition-behind-fluctuating-training-loss#:~:text=If%20the%20model%20output%20is,validation%20loss%20would%20remain%20constant>.

# Niestety nie chciało grać, ani działać ani ogólnie nic robić

Poprzez brak czasu, sił oraz wiedzy zdecydowałem się zmienić model, i sposób działania. Podstawa całego programu pochodzi z kodu Laboratorium 6. Pojedyncza epoka trwała 21 godzin na mojej maszynie a nawet 500 obrazków trwało też za długo by można było na jakąś normalną liczbę epok odpalić.



6. Architektura użyta w poniższym problemie

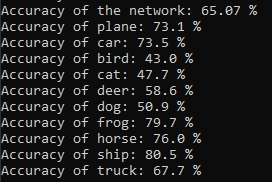
Cifar10:

1. Podstawowe hiperparametry:

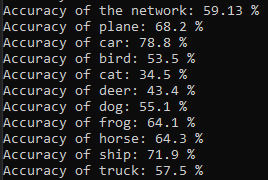
batch\_size = 4

epochs = 30

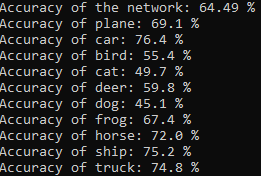
lr = 1e-3



1. 80 epok:



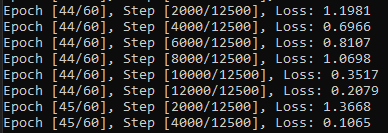
1. 60 Epok:



1. 2 i dodatkowo Batch size i lr zwiększone z 4 i 1e-3 do 16 i 4e-3
2. 3 i dodatkowo epok 40

## Obserwacje:

Zauważyłem gigantyczne wahania w stracie pomiędzy epokami dla



7. Wahania dla hiperparametrów 3

Na początku epoki 44 strata była mniejsza niż na początku epoki 45. Podczas epoki te wartości też się wahają, ale zazwyczaj kończą w lepszym miejscu niż na początku, ale nie jest to brane pod uwagę. Jakby latało wokół optimum.

If the model output is constant, the training loss would fluctuate( because of different samples in different batches) whereas the validation loss would remain constant. Maybe try altering the learning rate/batch size or add more data(to validation set as well).