Wizja Maszynowa
Architektury splotowych sieci neuronowych - wykład 9

Adam Szmigielski
aszmigie@pjwstk.edu.pl
materiały: $ftp(public) : //aszmigie/WM$

Popularne architektury sieci splotowych

- Sieci RNN - fast RNN, faster RNN
- SSD
- YOLO

LeNet-5 1998

Warstwa	Typ	Mapy	Rozmiar	Rozmiar jądra	Krok	F. aktywacji
Out	W pełni połączona	—	10	—	—	RBF
F6	W pełni połączona	—	84	—	—	tanh
C5	Splotowa	120	1×1	5×5	1	tanh
S4	Uśr. łącząca	16	5×5	2×2	2	tanh
C3	Splotowa	16	10×10	5×5	1	tanh
S2	Uśr. łącząca	6	14×14	2×2	2	tanh
C1	Splotowa	6	28×28	5×5	1	tanh
In	Wejściowa	1	32×32	—	—	—

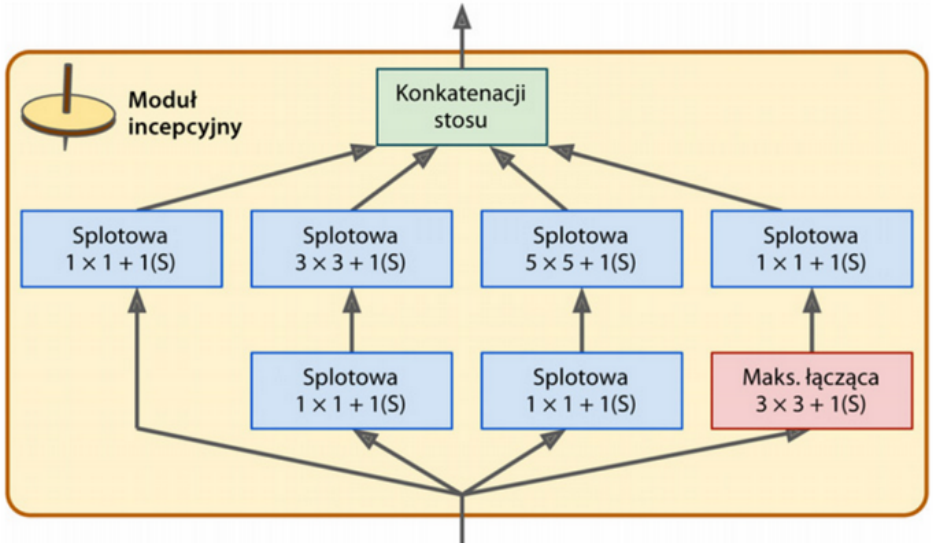
- Została stworzona przez Yanna LeCuna w 1998 roku do rozpoznawania odręcznie pisanych cyfr (MNIST)
- Architektura LeNet-5 stanowi prawdopodobnie najbardziej znany przykład sieci CNN.

AlexNet 2012

Warstwa	Typ	Mapy	Rozmiar	Rozmiar jądra	Krok	Uzup. zerami	F. aktywacji
Out	W pełni połączona	—	1000	—	—	—	Softmax
F9	W pełni połączona	—	4096	—	—	—	ReLU
F8	W pełni połączona	—	4096	—	—	—	ReLU
C7	Splotowa	256	13×13	3×3	1	SAME	ReLU
C6	Splotowa	384	13×13	3×3	1	SAME	ReLU
C5	Splotowa	384	13×13	3×3	1	SAME	ReLU
S4	Maks. łącząca	256	13×13	3×3	2	VALID	—
C3	Splotowa	256	27×27	5×5	1	SAME	ReLU
S2	Maks. łącząca	96	27×27	3×3	2	VALID	—
C1	Splotowa	96	55×55	11×11	4	SAME	ReLU
In	Wejściowa	3(RGB)	224×224	—	—	—	—

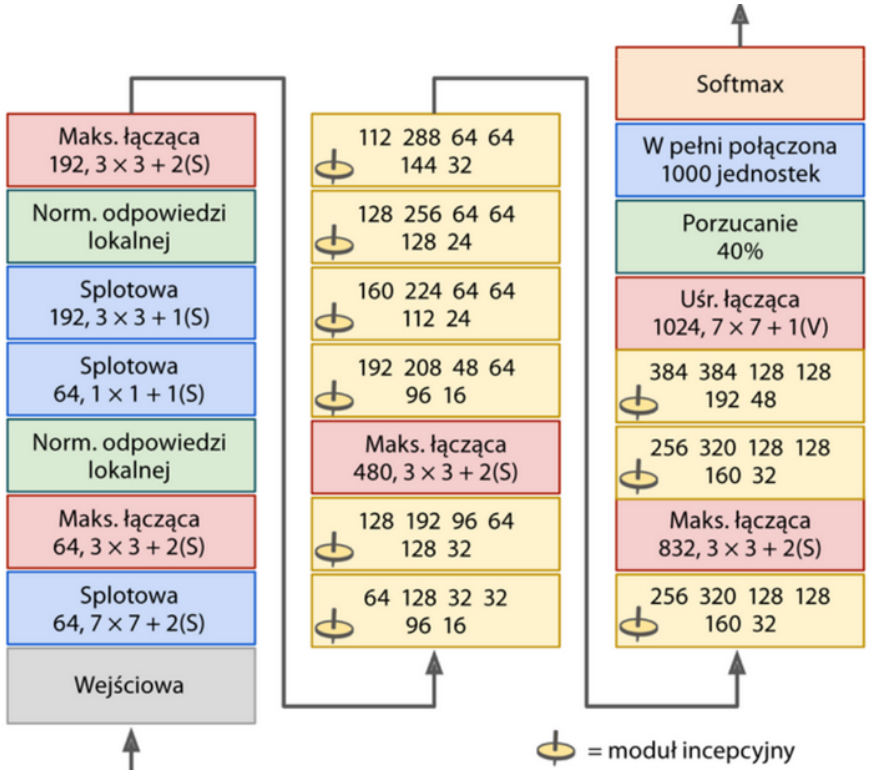
- W celu zmniejszenia przetrenowania użyto metody porzucania wobec wyjść warstw F8 i F9 oraz dogenerowali dane,
- W warstwach C1 i C3 użyto normalizacji odpowiedzi lokalnej (ang. local response normalization — LRN) - neurony o największych wagach (najsilniej aktywujące) hamują neurony znajdujące się w tym samym położeniu,

GoogLeNet 2014



- Stworzenie takiej architektury stało się możliwe dzięki wprowadzeniu podsieci zwanych **modułami incepcyjnymi** (ang. inception modules)
- Drugi zestaw warstw splotowych zawiera jądra o odmiennych rozmiarach (1×1, 3×3 i 5×5), co pozwala im wyłapywać wzorce w różnych skalach.

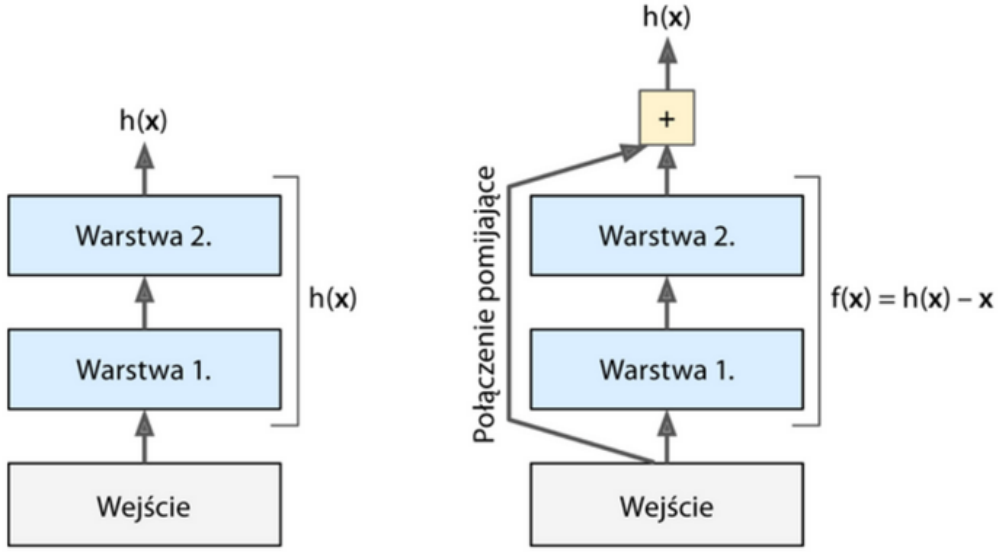
Architektura sieci GoogLeNet



- Pierwsze dwie warstwy redukują wysokość i długość obrazu czterokrotnie,
- Warstwa normalizacji odpowiedzi lokalnej sprawia, że wcześniejsze warstwy uczy się rozpoznawać bardzo zróżnicowane cechy

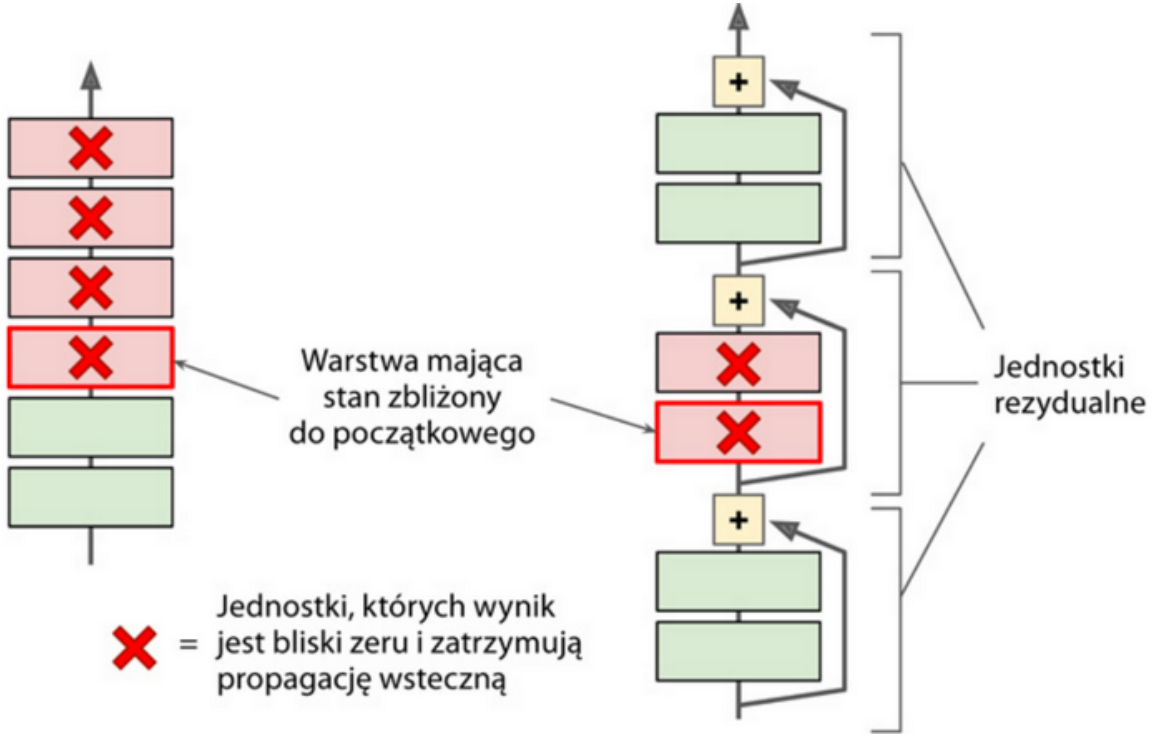
- Dwie warstwy splotowe, z której pierwsza pełni funkcję warstwy ograniczającej.
- Warstwa normalizacji odpowiedzi lokalnej,
- Maksymalizująca warstwa łącząca zmniejsza dwukrotnie wymiary obrazu,
- Wysoki stos ułożony z dziewięciu modułów incepcyjnych,
- Porzucanie służy do regularyzacji, a następnie w pełni połączona warstwa wykorzystująca funkcję aktywacji softmax do wyświetlania oszacowanych prawdopodobieństw przynależności przykładów do danej klasy.

ResNet 2015 - sieci rezydualne (ang. Residual Network)



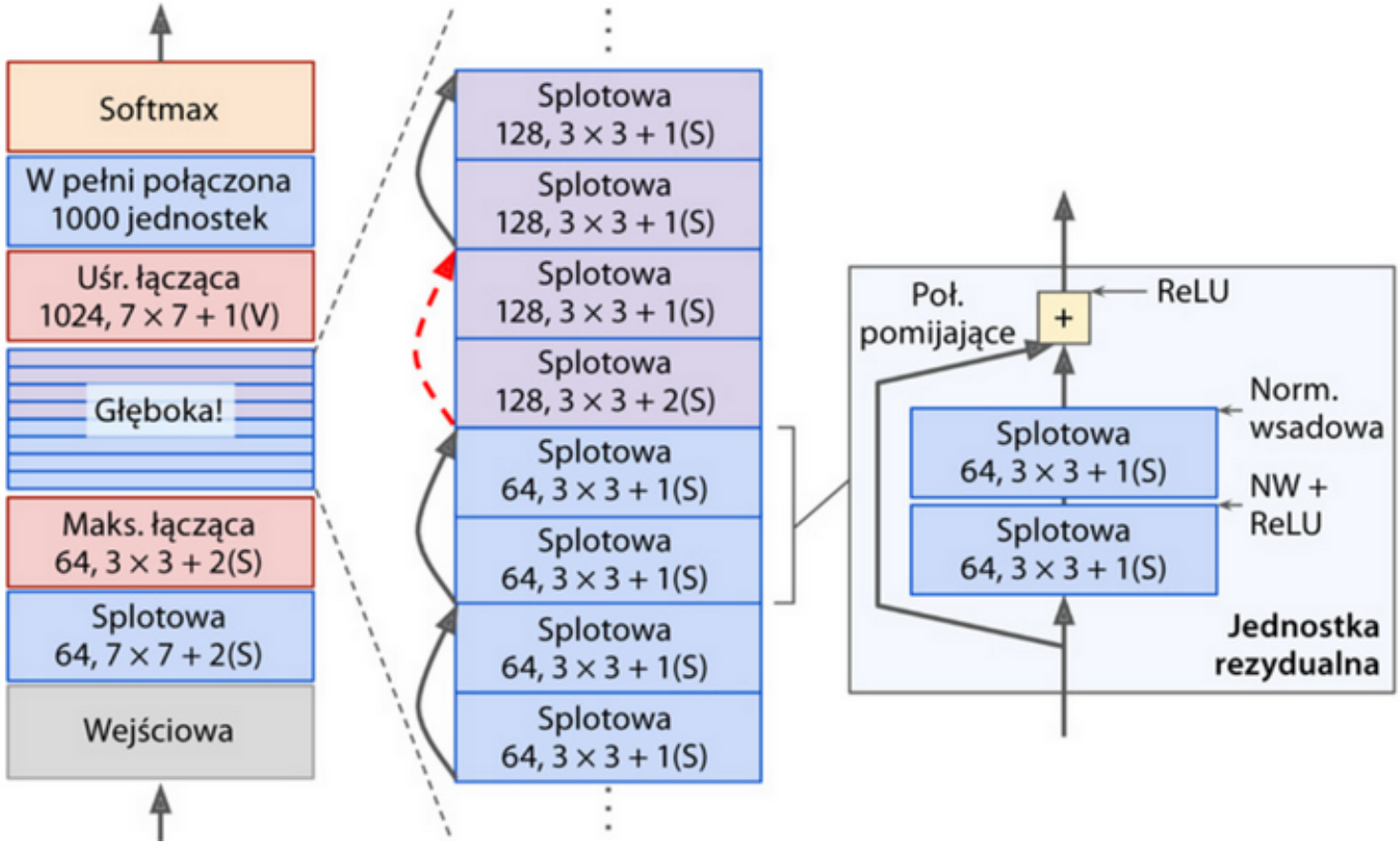
- Użyto głębokiej architektury - składającej się ze 152 warstw.
- Wykorzystano **połączenia pomijające** (zwanych również połączeniami skrótowymi — ang. shortcut connections)
- Jeżeli dodamy wejście x do wyjścia sieci (połączenia pomijającego), to sieć odwzoruje funkcję $f(x) = h(x) - x$ - **uczenie rezydualne** (resztowe; ang. residual learning).

Głęboka sieć resztowa



- Głęboka sieć resztowa może być postrzegana jako stos jednostek rezydualnych (ang. residual units), czyli niewielkich sieci neuronowych zawierających połączenie pomijające.

Architektura sieci ResNet

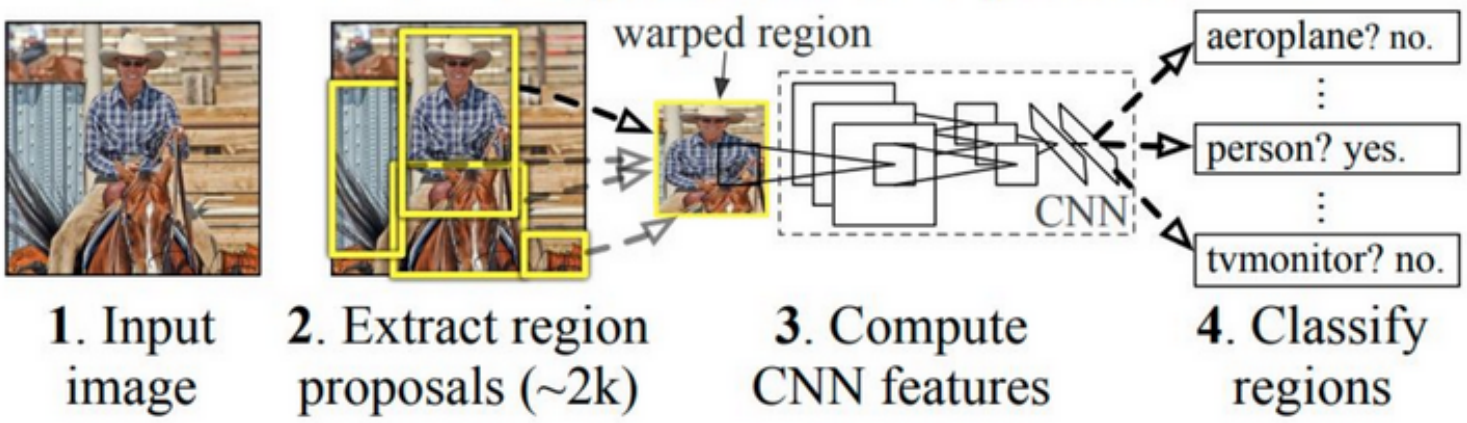


•

Detekcja obiektów w wykorzystaniem sieci splotowych

- R-CNN (Regions with CNN features) - fast RNN, faster RNN
- SSD (Single Shot Multibox Detector)
- YOLO (You Only Look Once)

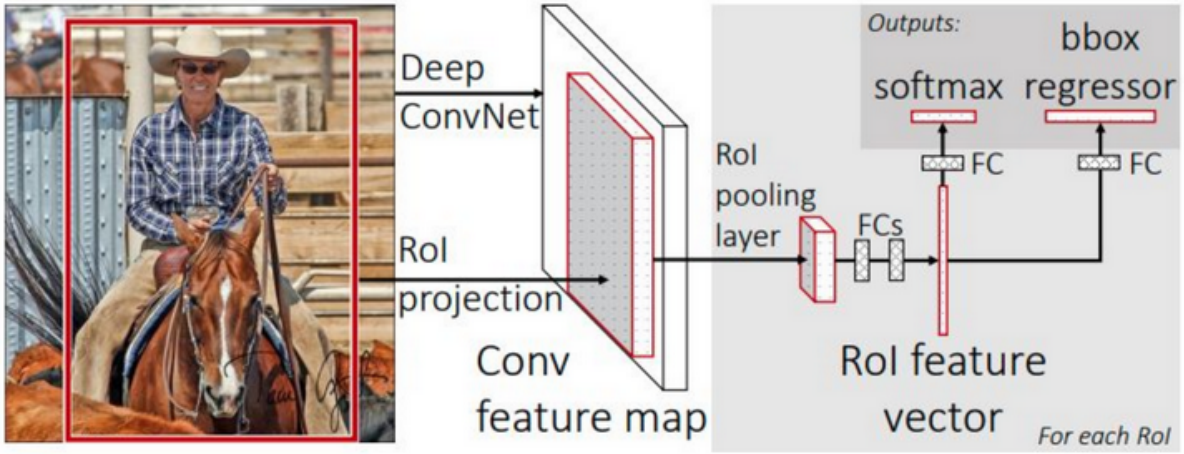
Obszarowa konwolucyjna sieć neuronowa - R-CNN



- **Propozycja regionów** - generowanie regionów, na których może być obiekt,
- **Moduł wyboru cech** - ekstrakcja cech z każdego regionu,
- **Klasyfikator** - klasyfikuje cechy do jednej ze znanych klas z wykorzystaniem klasyfikatora.

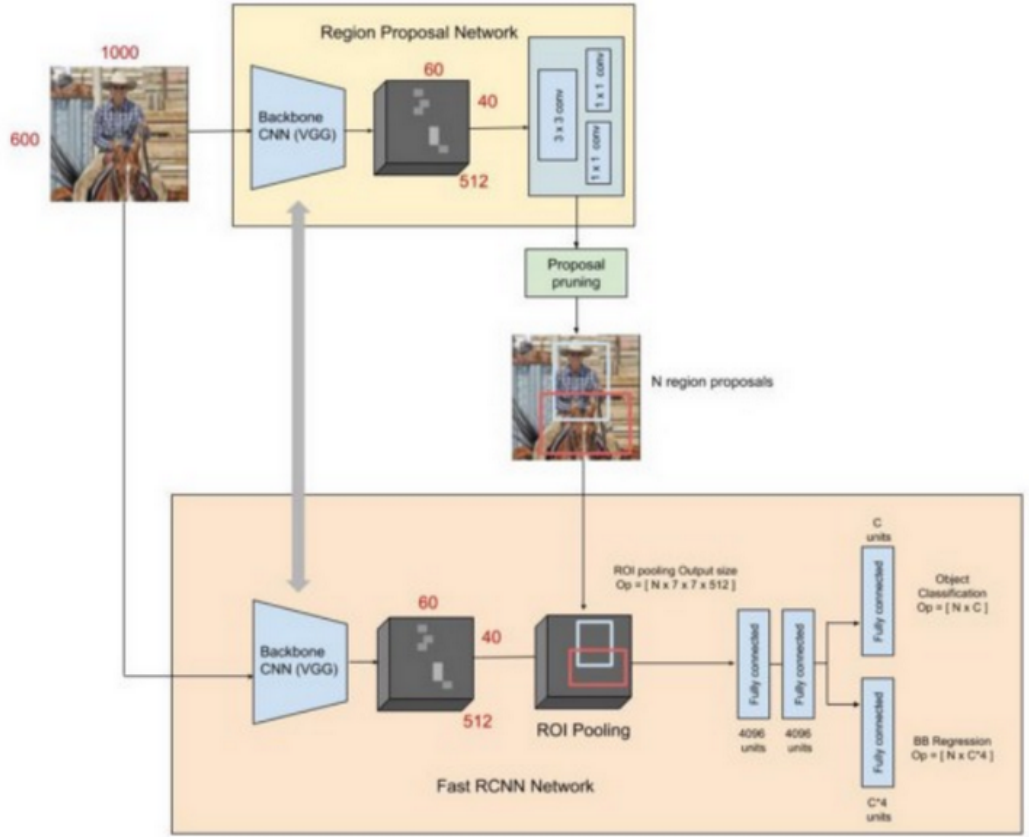
Długie szkolenie (ok. 2000 propozycji regionów dla każdego obrazu).
Wyszukiwanie regionów nie jest algorytmem uczącym się - może prowadzić do błędnych propozycji i błąd ten nie zostanie skorygowany.

Architektura modelu Fast R-CNN



- Pojedynczy model do nauki propozycji regionów i klasyfikacji za jednym razem,
- Wykorzystuje pre-trenowaną sieć do wyboru cech.
- Na końcu sieci jest warstwa zbierająca (Region of Interest Pooling Layer - RoI Pooling), która ekstrahuje cechy dla każdego proponowanego regionu.
- Następnie dane te interpretowane są w warstwie w pełni połączonej z dwoma wyjściami - klasyfikator i **bounding box**

Architektura modelu Faster R-CNN

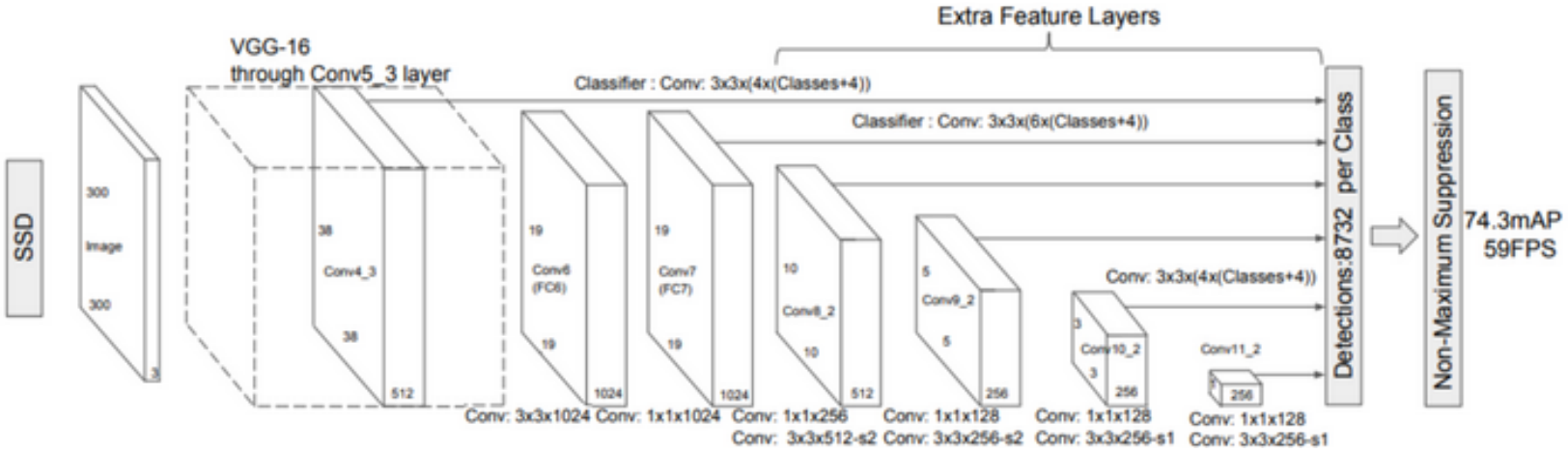


- Główną innowacją Faster R-CNN jest zastosowanie konwolucyjnej sieci do propozycyji regionów (ang. Region Proposal Network - RPN),
- Proces RPN rozpoczyna się od przepuszczenia całego obrazu przez

warstwy konwolucyjne.

- Dzięki zastosowaniu RPN zamiast selektywnego szukania, tworzymy w pełni uczącą się sieć, która ponadto może dzielić parametry pomiędzy warstwami konwolucyjnymi w RPN oraz detektorze Fast R-CNN.
- W Faster R-CNN, warstwa ROI Pooling bierze propozycje regionów nie z wyników szukania selektywnego, a procesu RPN.

Single Shot Multibox Detector SSD 2016

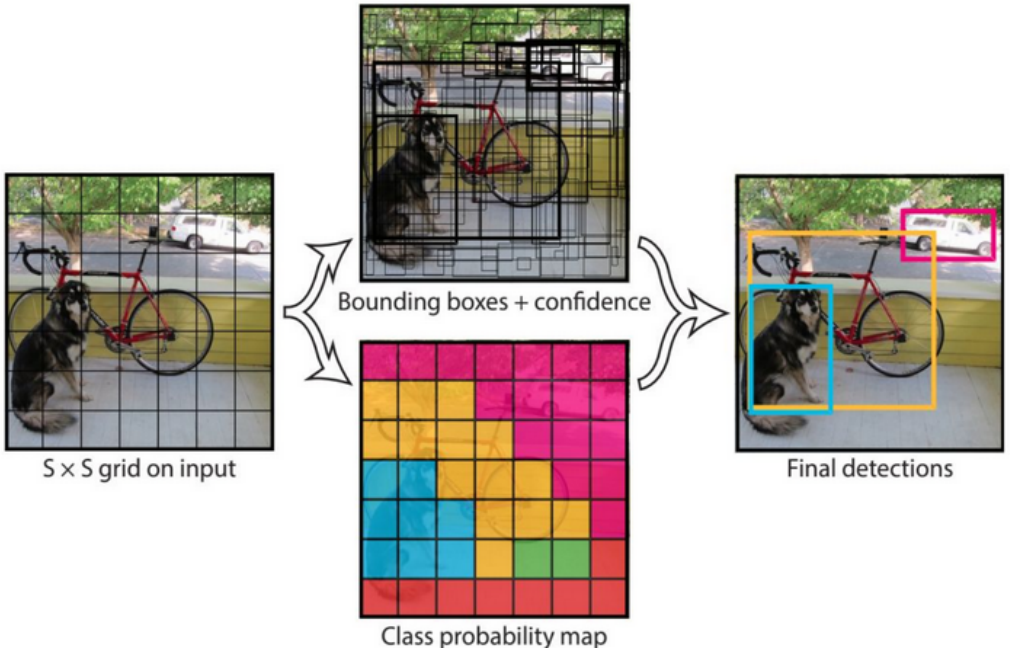


- SSD jest metodą wykrywania obiektów opartą na wykorzystaniu pojedynczej głębokiej sieci,
- W momencie predykcji sieć generuje wyniki dla obecności każdej kategorii w domyślnych obszarach
- Sieć łączy predykcje z wielu map cech z różnymi rozdzielczościami,

  SSD ma dwa komponenty szkielet modelu (ang. backbone) i warstwy SSD - dodatkowe warstwy cech (ang. Extra Feature Layers).

- Model bazowy zwykle jest pre-trenowaną siecią klasyfikującą obrazu,
- SSD nie wykorzystuje okienka przesuwającego się po obrazie, dzieli obraz na siatkę i każda komórka w tej siatce jest odpowiedzialna za wykrycie obiektów
- Jeśli nie wykryto żadnego obiektu, taka komórka staje się komórką z klasą tła.
- Narzędzia takie jak kotwica (ang. anchor box) i pole recepcyjne (ang. receptive field) zostają wykorzystane, by łatwiej wykryć wiele obiektów w obrębie jednej komórki.

YOLO - Patrzysz tylko raz (You Only Look Once) 2016



- `https://storage.googleapis.com/openimages/web/index.html`
- Wykorzystuje on cechy wytrenowane przez głębokie sieci neuronowe we frameworku Darknet.
- Jest siecią w pełni konwolucyjną (FCN - fully convolutional network)- jest obojętna na rozmiar obrazu wejściowego.

Działanie sieci YOLO

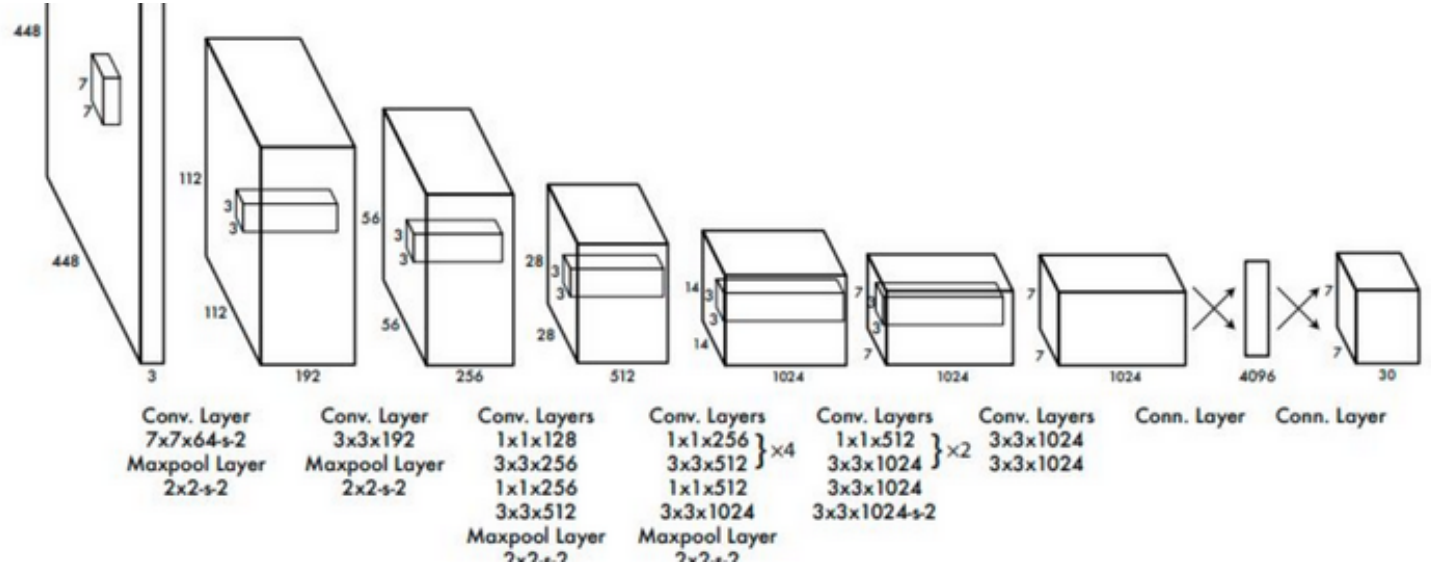
- Obraz zostaje podzielony na komórki, które tworzą siatkę $S \times S$.
- Wejściem do sieci YOLO jest obraz, a wyjściem współrzędne x, y środka obszaru zawierającego obiekt - obwiedni (ang. *bounding box*) na tym obrazie, jego szerokość, wysokość prawdopodobieństwo dla wykrytej klasy,
- Jeśli środek obiektu wypada w komórce siatki, komórka ta staje się odpowiedzialna za wykrycie tego obiektu.
- Każda komórka wykrywa B obwiedni i ich przedziały zaufania (ang. confidence scores).
- Przedziały zaufania określają niepewność detekcji obiektu i obwiedni,
- Po uzyskaniu wszystkich możliwych obwiedni, należy pozbyć się

tych, które mają niskie prawdopodobieństwo posiadania obiektu.

- Do tego stosuje się algorytm usuwania nie maksymalnych pikseli (non-maximum suppression) wyliczając indeks Jaccarda:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

YOLOv1

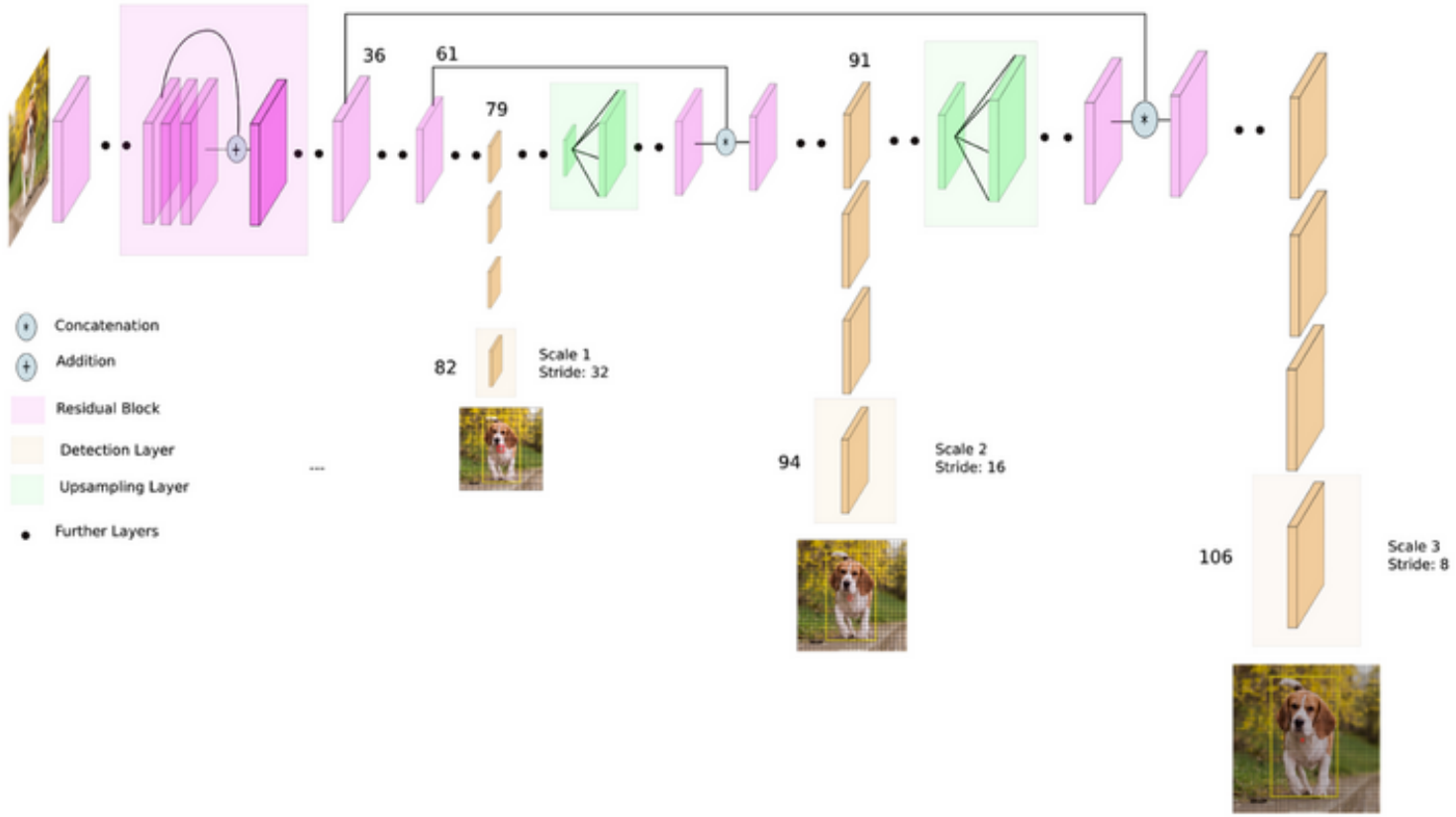


- Sieć ma 24 warstwy konwolucyjne i 2 w pełni połączone warstwy, 2 warstwy redukujące (max-pooling).
- Jest to sieć konwolucyjna, która jednocześnie tworzy predykcję wielu obszarów i prawdopodobieństwa klas dla tych obszarów.
- YOLOv1 daje słabsze wyniki, gdy na obrazie jest wiele małych obiektów np. stado ptaków

YOLOv2 - YOLO9000 2016

- W drugiej wersji algorytmu detekcja YOLO oparta jest o również ulepszony, nowy model *Darknet19*,
- Ma mniej warstw konwolucyjnych i 5 warstw redukujących oraz warstwę softmax,
- Wprowadzono kotwice bazowe obwiedni: w YOLOv2 wprowadzono obwiednie tak, jak np. w Faster R-CNN - większyło ilość obwiedni na obraz z 98, do ponad tysiąca,
- Poprawiła się detekcja małych obiektów - tworzy detekcje na mapie cech o wymiarach 13×13,
- Trenowanie na obrazach w różnych skalach.

YOLOv3 - 2018



- YOLOv3 opiera się na ulepszonej architekturze *Darknet53* - ma 53 warstwy konwolucyjne,
- Predykcje w różnych skalach - sieć w trzeciej wersji przewiduje obwiednie w trzech różnych skalach obrazu,

Keras, TensorFlow, PyTorch

- **Keras** to wydajny interfejs programowania aplikacji sieci neuronowej wysokiego poziomu, który umożliwia szybkie eksperymentowanie z głębokimi sieciami neuronowymi. Keras został zintegrowany z TensorFlow.
- **TensorFlow** to biblioteka matematyczna, która najlepiej nadaje się do programowania przepływu danych w szeregu zadań. Oferuje wiele poziomów abstrakcji do budowania i trenowania modeli.
- **PyTorch** to stosunkowo nowy framework do głębokiego uczenia oparty na Torch - prosty, łatwy w użyciu, elastyczny, wydajny w wykorzystaniu pamięci i dynamicznych wykresach obliczeniowych.

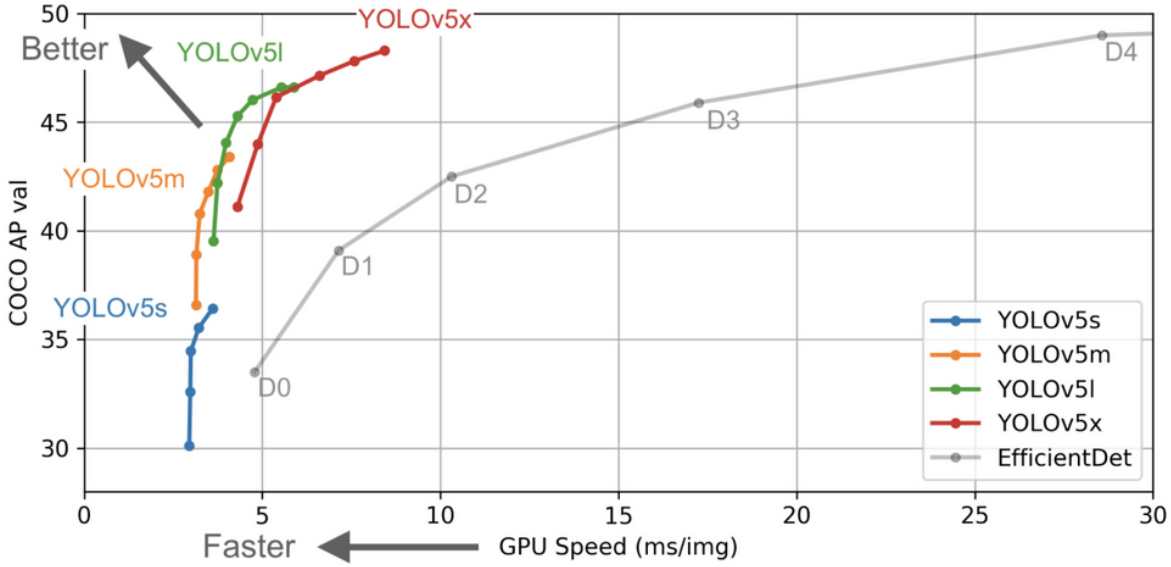
PyTorch vs TensorFlow

- PyTorch jest obecnie używany w wielu projektach głębokiego uczenia - jego popularność rośnie,
- Gdy potrzebna jest elastyczności, możliwości debugowania i krótkiego czasu szkolenia lepszy jest PyTorch. Działa na systemach Linux, macOS i Windows.
- Dzięki dobrze udokumentowanej strukturze i dużej liczbie przeszkolonych modeli TensorFlow jest bardzo popularnym narzędziem,
- TensorFlow oferuje lepszą wizualizację, która pozwala programistom na lepsze debugowanie i śledzenie procesu uczenia.

PyTorch vs TensorFlow

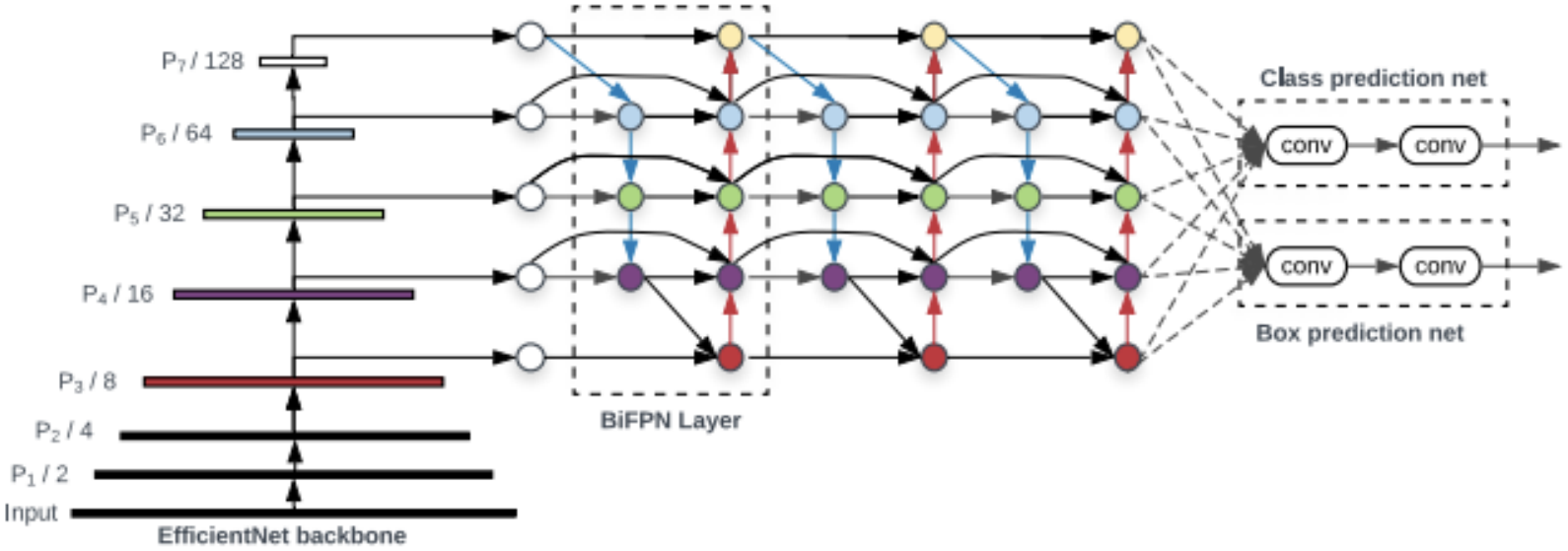
- Keras jest bardziej odpowiedni dla programistów, którzy potrzebują platformy plug-and-play, która pozwoli im szybko budować, trenować i oceniać swoje modele.
- PyTorch jest szybszy niż Keras i ma lepsze możliwości debugowania.
- Keras jest najlepszy do pracy z małymi zbiorami danych, szybkiego prototypowania.

YOLOv5



- YOLOv5 to model z rodziny modeli widzenia komputerowego You Only Look Once (YOLO). YOLOv5 jest powszechnie używany do wykrywania obiektów,
- YOLOv5 - rozszerzenie YOLOv3 PyTorch

Architektura YOLOv5



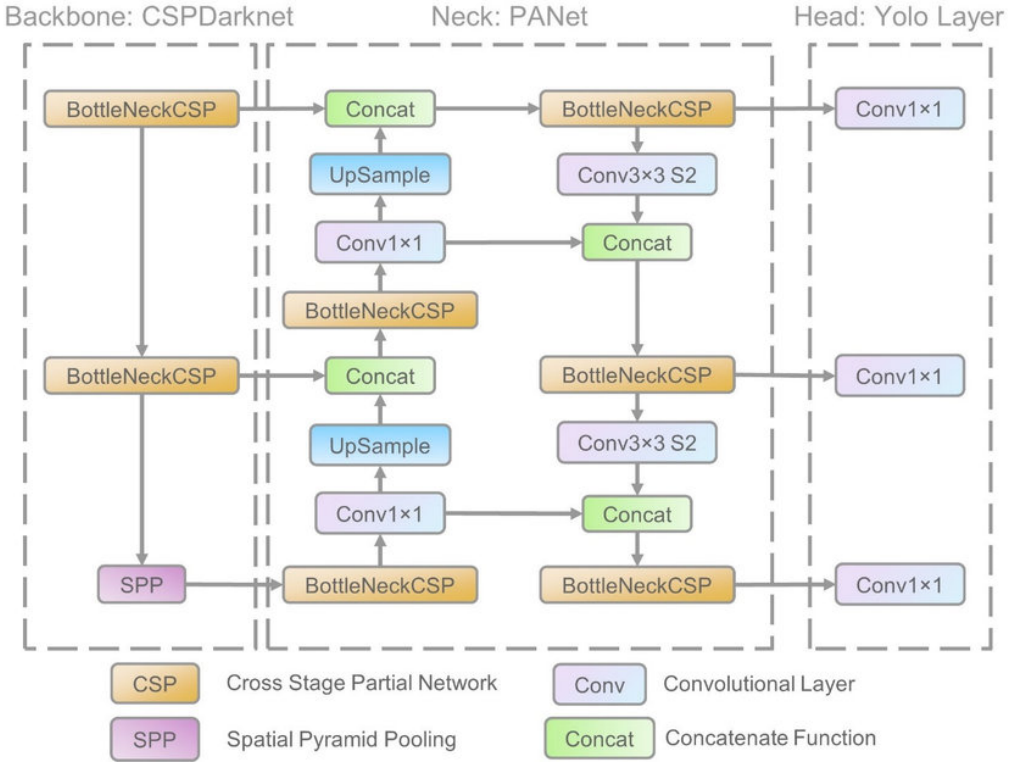
- Model YOLO był pierwszym detektorem obiektów, który połączył procedurę przewidywania ramek ograniczających z etykietami klas w sieci CNN.

Architektura YOLOv5

Sieć YOLO składa się z trzech głównych części.

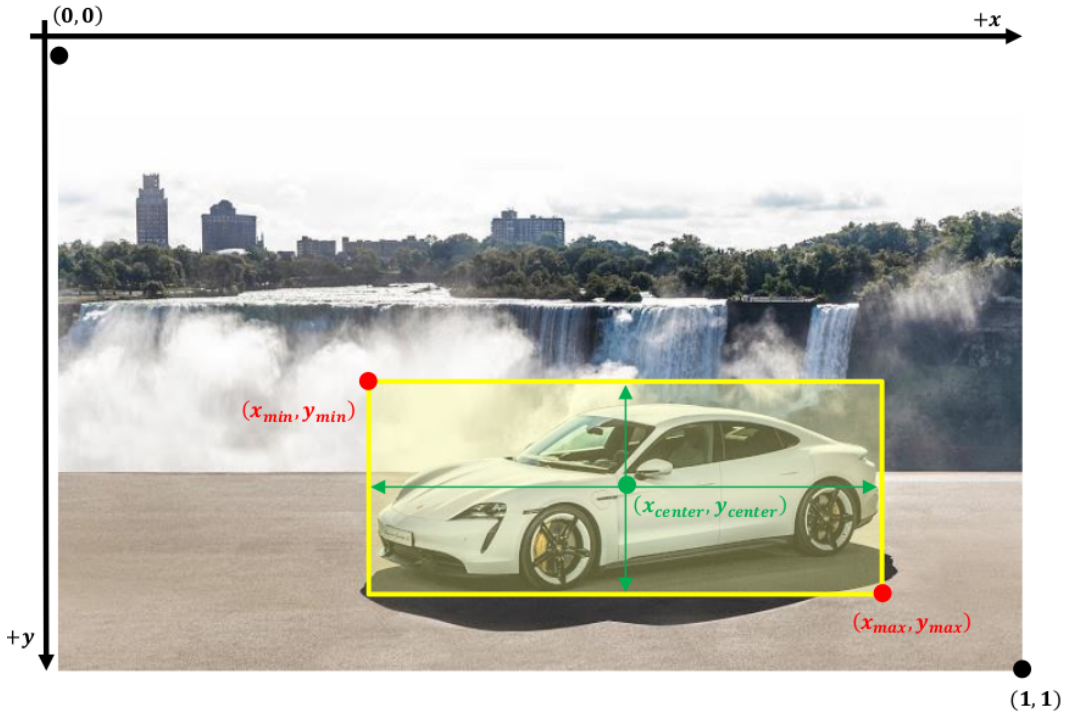
- **Backbone**: Konwolucyjna sieć neuronowa, która agreguje i tworzy cechy obrazu o różnych stopniach ziarnistości.
- **Neck**: Seria warstw do mieszania i łączenia cech obrazu w celu przekazania ich do predykcji.
- **Head**: Wykorzystuje cechy z bloku neck i dokonuje predykcji cech oraz ramek.

Architektura YOLOv5



- Dane są najpierw wprowadzane do CSPDarknet w celu wyodrębnienia cech, a następnie przekazywane do PANet w celu fuzji cech. Na koniec Yolo Layer wyświetla wyniki wykrywania (klasa, wynik, lokalizacja, rozmiar).

Format obwiedni YOLO



- Dane ramek ograniczających w YOLO są sformatowane jako [class, xcenter , ycenter , width, height].

Zadania na laboratoria

- Zapoznaj się z **PyTorch**. Zainstaluj siec **YOLOv5** lub nowszą,
- Wytrenuj sieć w oparciu o dostępną bazę danych,
- Uruchom sieć do śledzenia i wykrywania wyuczonych obiektów,
- Dodaj kolejną klasę “piła łańcuchowa”. Powiększ odpowiednio dane oraz ponownie wytrenuj sieć. Można skorzystać z dodatkowych z **narzędzi do etykietowania**,
- Sprawdź działanie sieci śledząc piłę łańcuchową na przykładzie filmu deskryptorami.