

Rozpoznawanie i klasyfikacja obrazów

Mateusz Markiewicz

Maj 2020

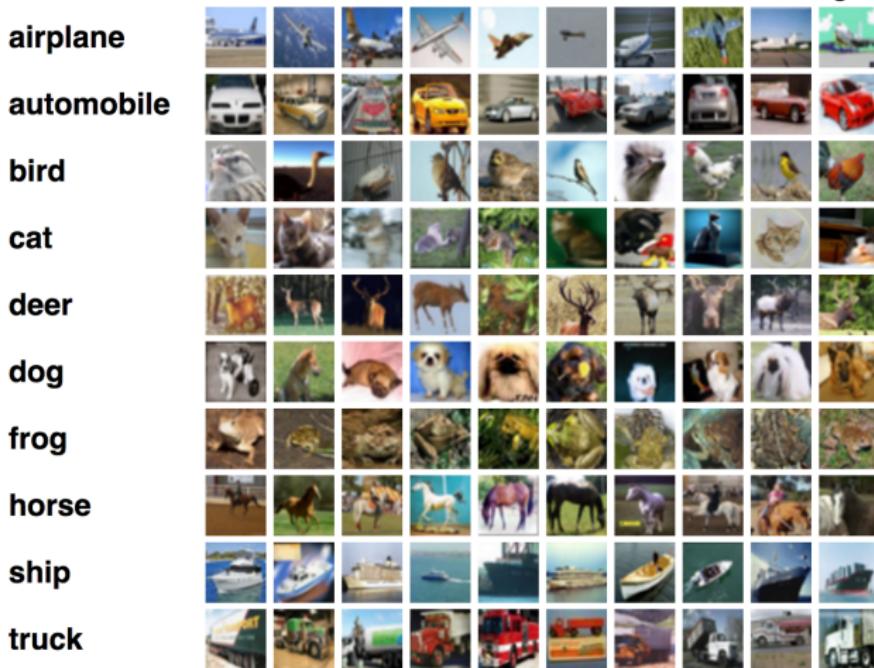
Plan

- Klasyfikacja a rozpoznawanie
- KNN
- SVM
- Regresja logistyczna
- Sieci neuronowe
- Sieci splotowe
- VGG
- Wykrywanie obiektów

Klasyfikacja obrazów

Klasyfikacja jest procesem polegającym na przypisaniu elementowi otrzymanemu jako wejście jednej klasy. Wszystkie klasy, które mogą zostać zwrócone musimy wcześniej znać.

Klasyfikacja obrazów

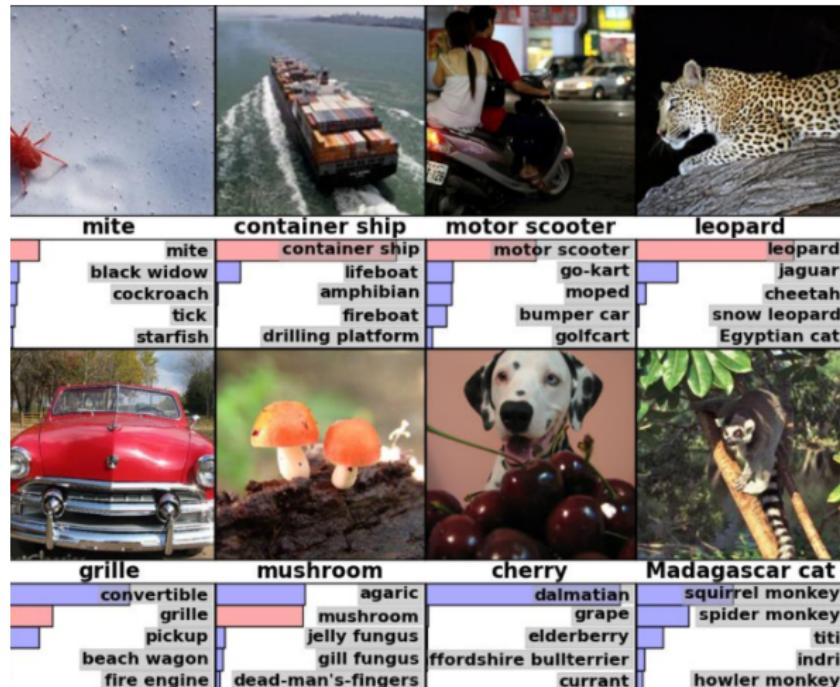


Rysunek: Źródło

Rozpoznawanie obrazów

Rozpoznawanie obrazów to proces polegający na rozpoznaniu na zdjęciu określonych osób, zwierząt, obiektów lub przedmiotów. Przykładem jest proces rozpoznania twarzy.

Rozpoznawanie obrazów



Rysunek: Źródło

Czego potrzebujemy?

Supervised learning

Uczenie nadzorowane (supervised learning) charakteryzuje się tym, że dysponujemy danymi, które są poprawnie zaklasyfikowane, przez co sam proces uczenia może być nadzorowany.

Zbiory danych

Jeśli chcemy szkolić model w sposób nadzorowany potrzebujemy więc (dużo) przykładowych danych z poprawnie przypisanymi klasami.
My użyjemy MNIST - zbioru zdjęć ręcznie zapisanych cyfr.

Data sets: Train-test split

Motywacja

Nie możemy jednak używać całych danych, którymi dysponujemy do nauki.

Naszym celem jest, by nasz model radził sobie dobrze z danymi, których nigdy wcześniej nie widział.

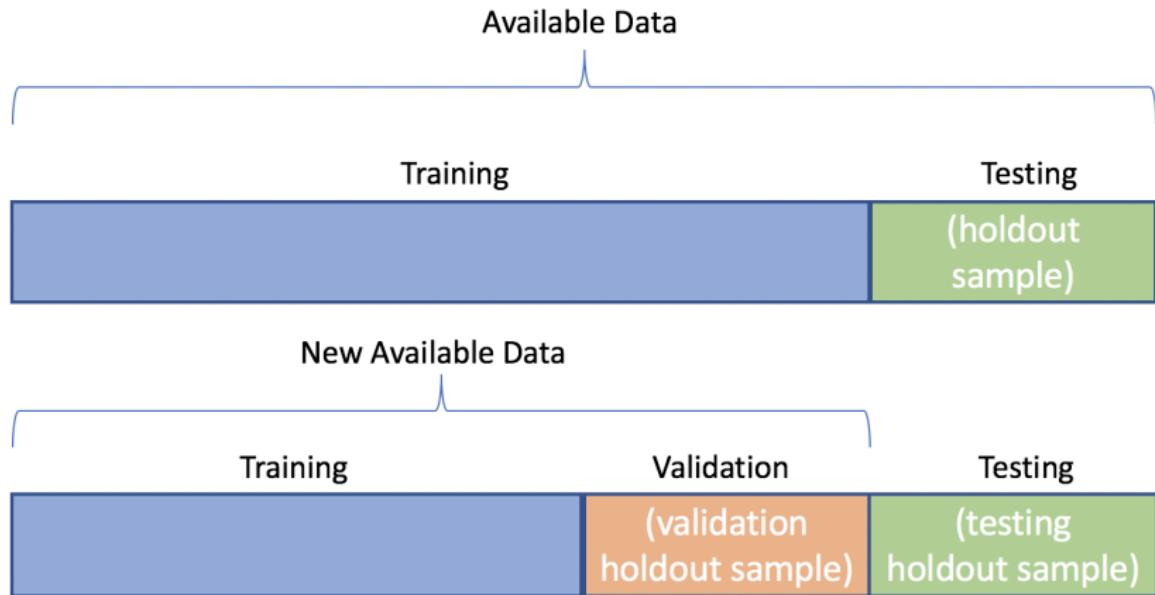
Jeśli w procesie uczenia będziemy oceniać skuteczność modelu na podstawie tych samych danych, na których ten model się uczył, to pomiar ten nie będzie wiarygodny.

Realizacja

Dzielimy nasze dane na dane treningowe, na których uczymy nasz model oraz dane testowe, na których będziemy go testować.

Jeśli nasz model ma dodatkowe parametry taki podział może być niewystarczający, wówczas dzielimy dane na 3 części: treningową, walidacyjną oraz testową. Część walidacyjna będzie nam służyła do ustalenia najlepszych parametrów naszego modelu.

Train-test split



Rysunek: Źródło

KNN (K-nearest neighbors)

Idea

Pierwszym podejściem do problemu klasyfikacji będzie algorytm KNN. Idea algorytmu jest bardzo prosta. Dla obrazka, który dostajemy do sklasyfikowania znajdujemy K najbliższych mu obrazków z danych treningowych oraz zwracamy najczęstszą spośród ich klas.

Co oznacza najbliższych w kontekście obrazów?

Obraz jest macierzą, której każdym element reprezentuje pojedynczy piksel. W przypadku MNIST piksele są w skali szarości, więc każdy reprezentowany jest jako pojedyncza wartość. Możemy rozwinąć macierz w wektor, a następnie mając dwa takie wektory obliczamy drugą normę ich różnicę.

Odległość między obrazami

Odległość między dwoma wektorami

$$dist(X^i, Y^j) = \sqrt{\sum_{k=0}^K (X_k^i - Y_k^j)^2}$$

gdzie X^i to i -ty wektor spośród danych treningowych, a Y^j to j -ty wektor spośród danych testowych.

Macierz odległości

Dla każdego wektora, któremu chcemy przyporządkować klasę potrzebujemy jego odległość do wszystkich danych treningowych, czyli macierzy D, taką że

$$D_{i,j} = Dist(X^i, Y^j)$$

Mając taką macierz możemy łatwo znaleźć K najbliższy wektorów danych treningowych dla j-tego wektora danych testowych (na przykład wykonując jednorazowo argsort)

Rezultat KNN



Rysunek: Przykład działania K-NN dla danych MNIST

Skuteczność?

97.1%

Redukcja wymiarowości

Aktualna wymiarowość

Aktualnie pracujemy na obrazach 28×28 pikseli, które rozwijamy w wektory. W ten sposób otrzymujemy wektor o 784 wymiarach.

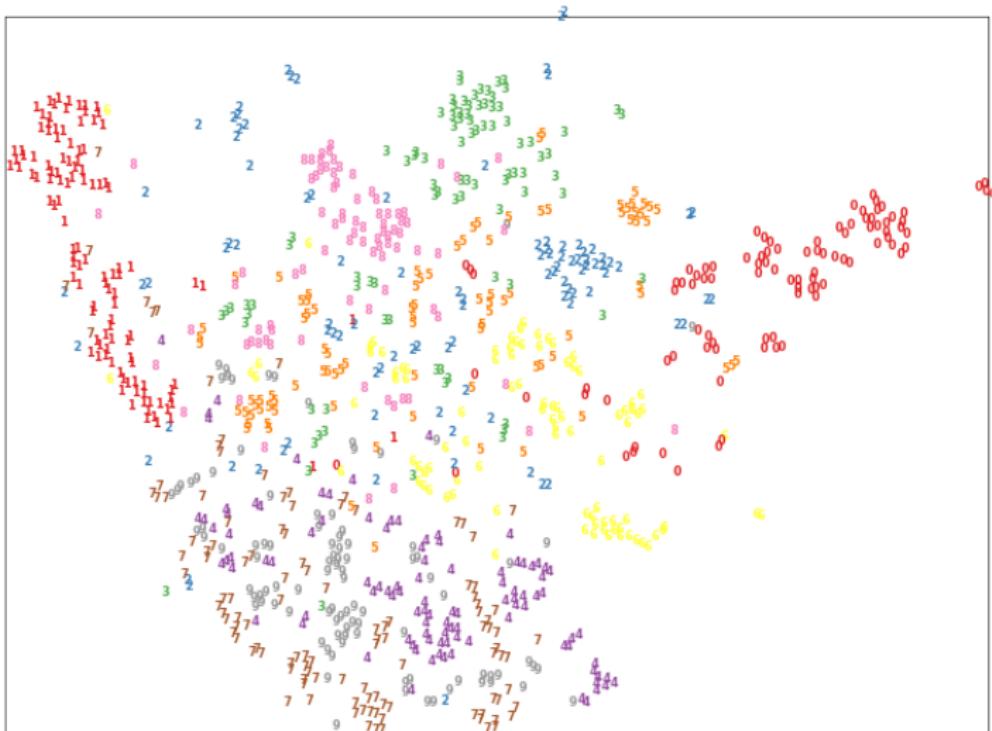
Po co redukować dane?

Czasem chcemy zredukować wymiarowość danych nawet do $2/3$ wymiarów w celu ich wizualizacji. Niektóre redukcje wymiarowości poprawiają również skuteczność klasyfikatorów.

Najpopularniejsze metody redukcji wymiarowości

- Random projection
- PCA
- K-PCA
- Isomap
- LLE
- MDS
- t-SNE

Rezultat redukcji wymiarowości



Rysunek: t-SNE

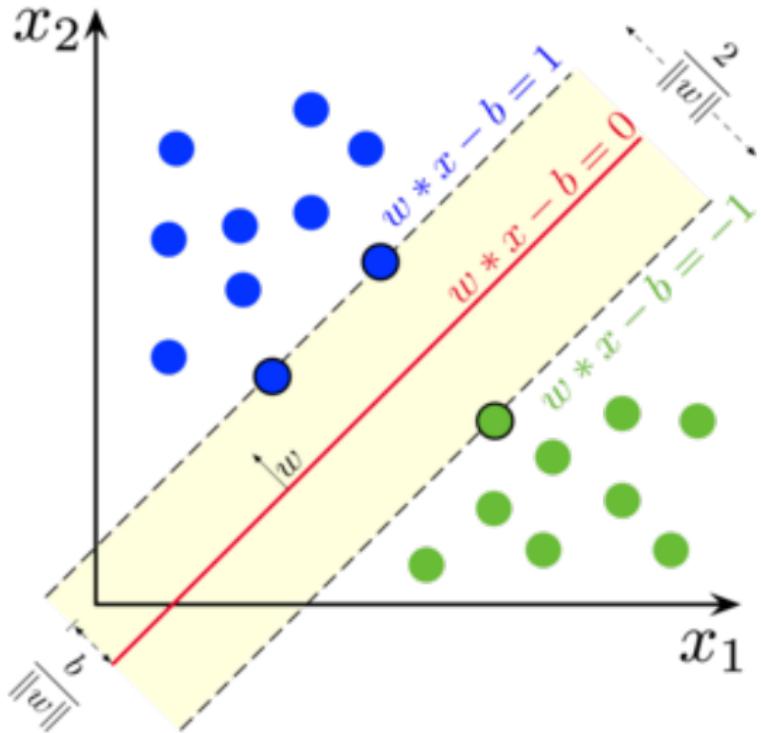
Idea

Niech X^i będzie i-tym wektorem danych (np. rozwiniętą macierzą reprezentującą obraz), a Y^i będzie klasą tego wektora, gdzie $Y \in \{-1, 1\}$. Celem SVM jest wyznaczenie klasy dla X^i na podstawie $\text{signum}(w^T X^i + b)$.

Wizualizacja

Chcemy więc znaleźć przestrzeń, która w jak najlepszy sposób oddziela od siebie obie klasy.

SVM



Rysunek: Źródło

Kernel trick

Idea

Wiemy, że nie wszystkie dane możemy rozdzielić linią. Jeśli przedstawimy je jednak w przestrzeni wyżej wymiarowej może się okazać, że będzie się dało znaleźć linie (w tej wyżej-wymiarowej przestrzeni), która je rozdziela. Chcemy wykorzystać nieliniowe przekształcenie $x \rightarrow \phi(x)$.

Złożoność obliczeniowa

Jeśli przestrzeń $\phi(x)$ będzie wysoce wymiarowa wszystkie obliczenia byłyby czasochłonne. Istnieją jednak takie $\phi(\cdot)$, że możemy obliczyć $\phi(X)^T \phi(Y)$ wykonując jedynie obliczenia w oryginalnej przestrzeni.

Jak zastosować SVM gdy mamy więcej klas?

Są dwa główne podejścia, by przystosować wcześniej opisany algorytm do radzenia sobie z danymi mającymi więcej, niż dwie klasy - One versus rest oraz One versus one.

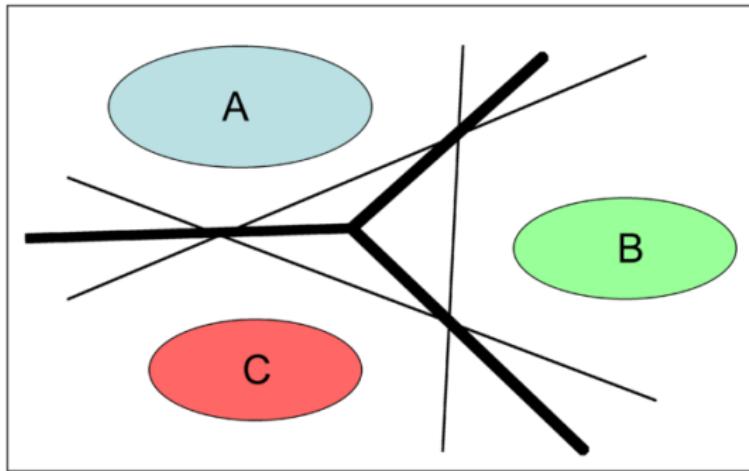
One versus rest

Mając dane składające się z K klas trenujemy K klasyfikatorów. i -ty z nich decyduje, czy klasyfikowany wektor należy do i -tej klasy, czy do reszty klas. Po skończonym treningu otrzymujemy K par (w_k, b_k) . Chcąc sklasyfikować nowy wektor X obliczamy wartości $f^k(X) = w_k^T X + b_k$, a następnie zwracamy k dla którego $f^k(X)$ osiąga maksymalną wartość.

One versus one

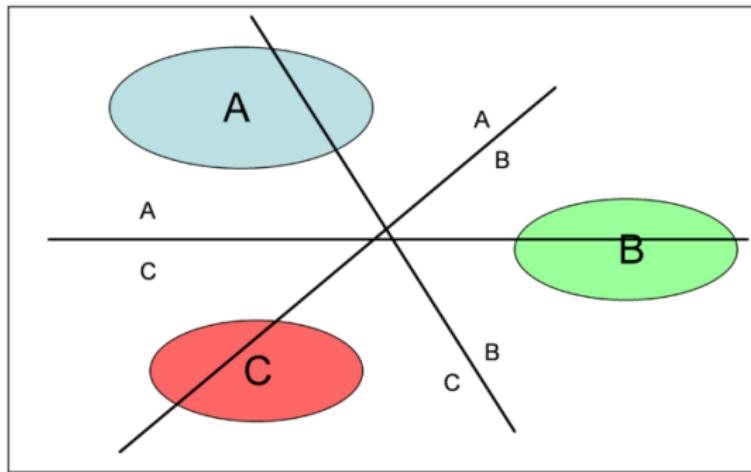
Trenujemy $K(K - 1)$ klasyfikatorów, po jednym dla każdej pary klas. Klasyfikując nowy wektor tworzymy K liczników, po jednym dla każdej klasy. Mając k -ty klasyfikator dla pary klas (i, j) , jeśli $f^k(X)$ będzie równe 1 inkrementujemy licznik i -tej klasy, w przeciwnym przypadku inkrementujemy licznik j -tej klasy. Na koniec zwracamy klasę o największej wartości licznika.

One versus rest



Rysunek: Źródło

One versus one



Rysunek: Źródło

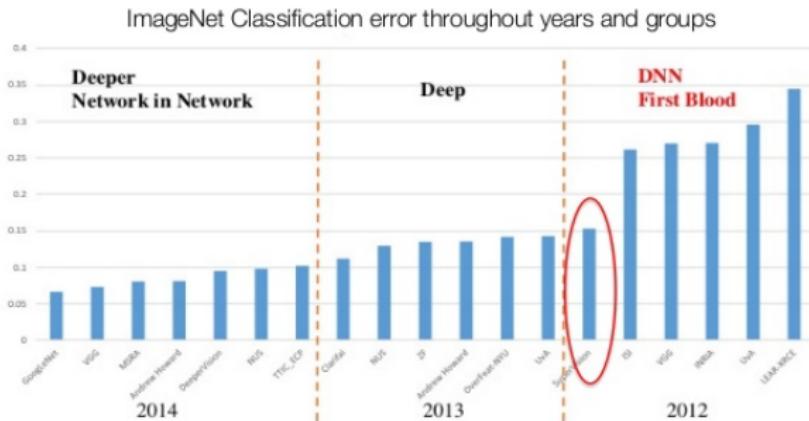
Classification Results (CLS)



Rysunek: Źródło

Przełom w skuteczności klasyfikacji

ImageNet Challenge



Li Fei-Fei: ImageNet Large Scale Visual Recognition Challenge, 2014

Rysunek: Źródło

Regresja liniowa

Regresja liniowa

Mając dany wektor danych $x = [x_1, x_2, \dots, x_n]$ regresję liniową możemy opisać za pomocą wyrażenia

$$h_{\Theta}(x) = \sum_{n=1}^N \Theta_n * x_n + \Theta_0$$

Możemy więc przyjąć, że wektor danych x przekształcamy do postaci $x = [1, x_1, x_2, \dots, x_n]$, dzięki czemu powyższe równanie sprowadza się do:

$$h_{\Theta}(x) = \Theta^T x$$

Regresja logistyczna

Idea

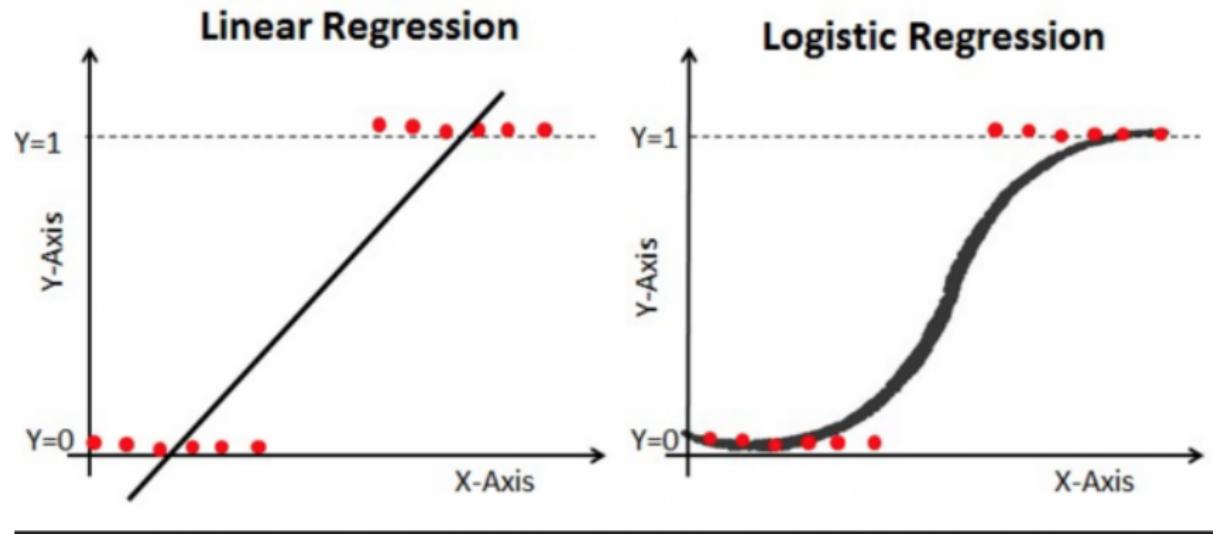
Regresja logistyczna jest modelem, który dla każdej wartości wejściowej zwraca wartość wyjściową z przedziału $[0, 1]$. Bardzo często wykorzystuje się ten fakt i stosuje się regresję logistyczną do klasyfikacji.

Regresja logistyczna

Łatwo zauważyc, że wartości $h_{\Theta}(x)$ mogą przyjmować wartości z przedziału $(-\infty, \infty)$. Możemy jednak nałożyć na tą wartość sigmoid, czyli funkcję $\sigma(z) = \frac{1}{1+e^{-z}}$. Stąd:

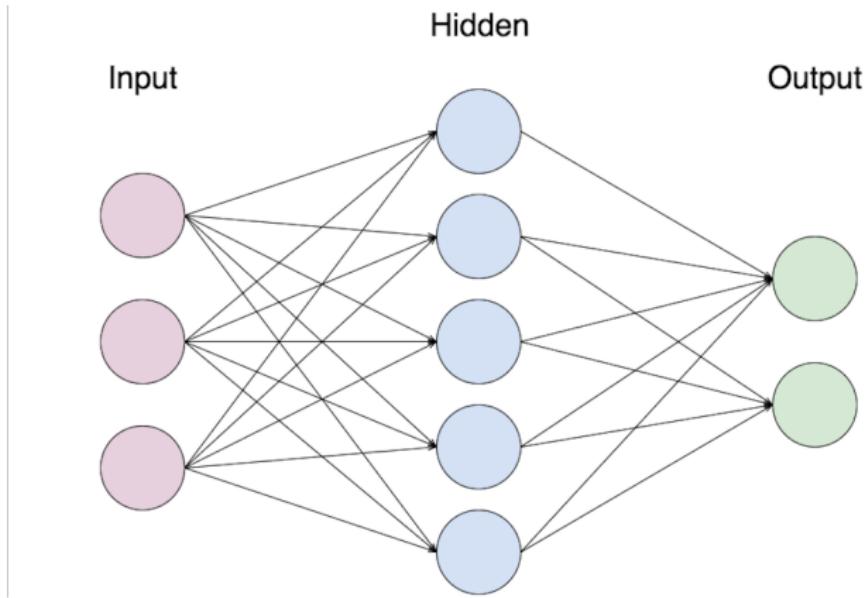
$$h_{\Theta}(x) = \sigma(\Theta^T x) = \frac{1}{1 + e^{-\Theta^T x}}$$

Regresja logistyczna



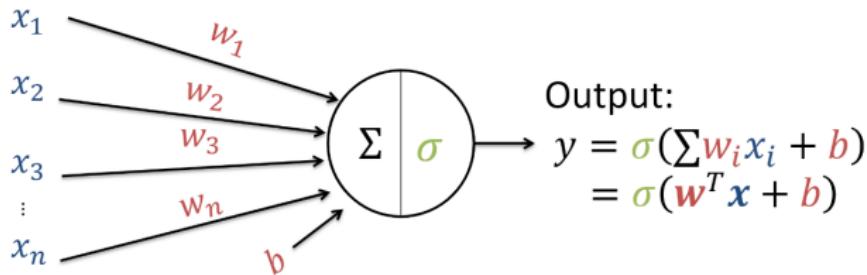
Rysunek: Źródło

Sieci neuronowe



Rysunek: Źródło

The artificial neuron (perceptron)



- x_i are the inputs
 - w_i are the weights and b the bias
 - Σ denotes the summation
 - σ is a (possibly nonlinear) activation function
- w_i, b are TUNABLE!!*

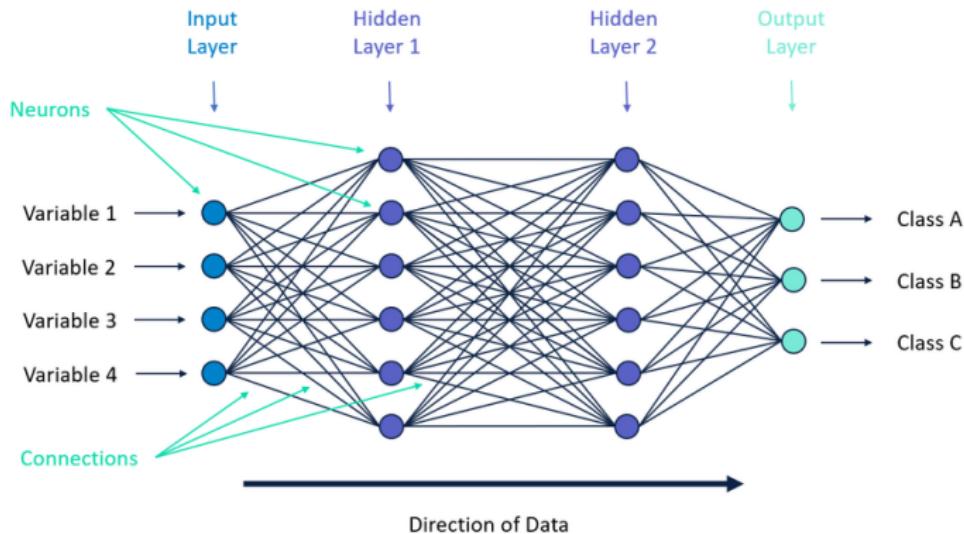
Rysunek: Źródło

Forward pass

Jak obliczyć predykcję dla sieci?

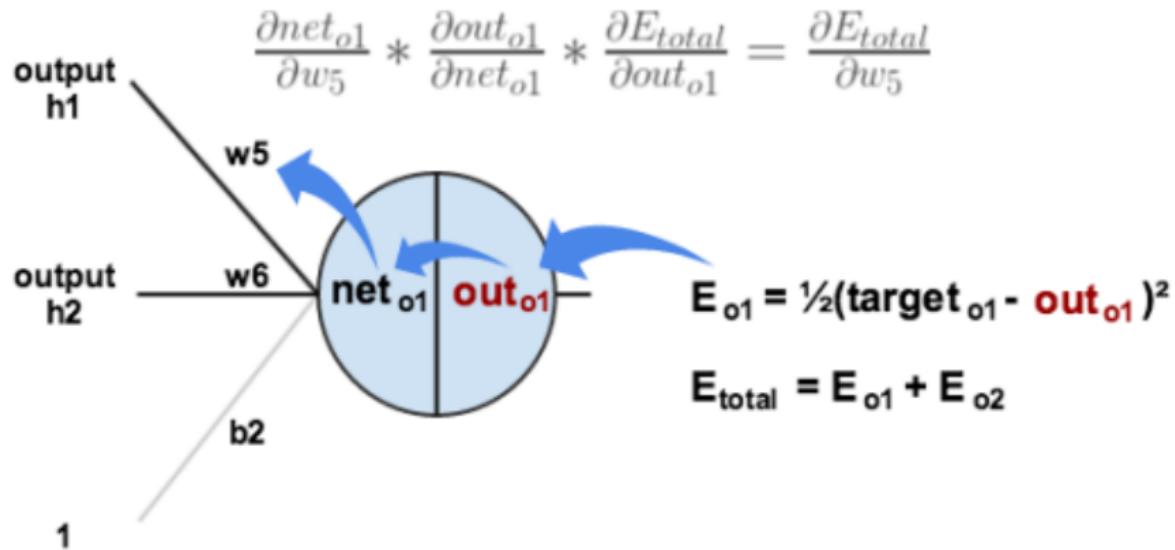
Każda kropka po lewej stronie odpowiada za jeden element wejścia, np. jeden piksel obrazu. Następnie każdy neuron z pierwszej warstwy oblicza swoją wartość w sposób pokazany na obrazku wyżej. Trzeba zwrócić uwagę, że każdy neuron ma osobną wagę dla każdego wejścia. Druga warstwa powtarza te same obliczenia, jedynie zamiast patrzyć na wejście (np. obraz) patrzy na wynik pierwszej warstwy. Ostatni warstwa ma dokładnie tyle neuronów, ile mamy możliwych klas. Ostatecznie wejście dostaje klasę, której odpowiadający neuron z ostatniej warstwy miał największą wartość. Takie przejście nazywa się forward passem.

Forward pass



Rysunek: Źródło

Backward pass

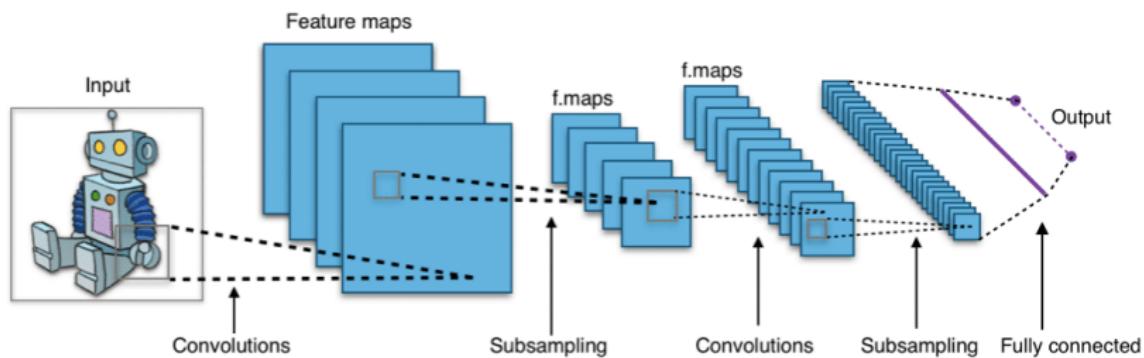


Rysunek: Źródło

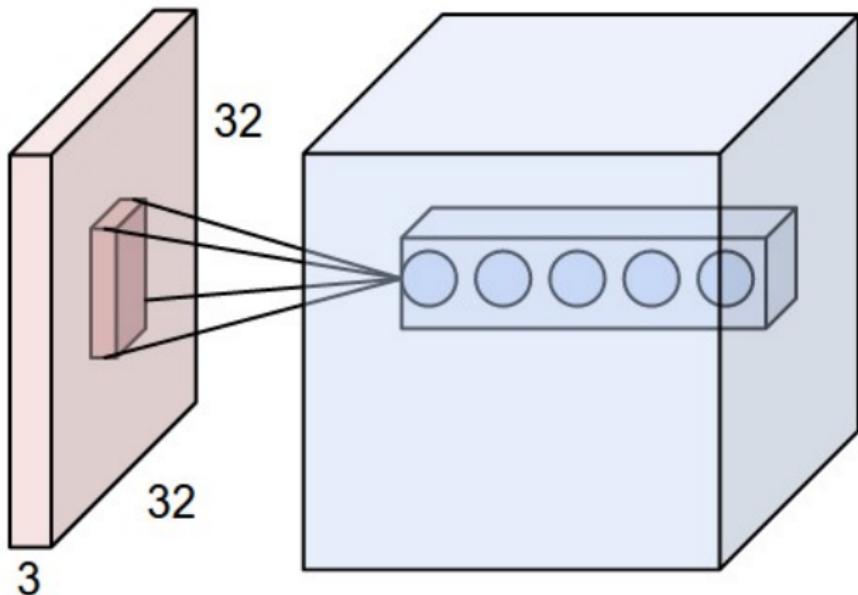
Sieci splotowe

Filtry

W sieciach splotowych (konwolucyjnych) pojedynczy neuron zamiast "patrzeć" na każdy piksel patrzy jedynie na pewien wycinek obrazu wejściowego.

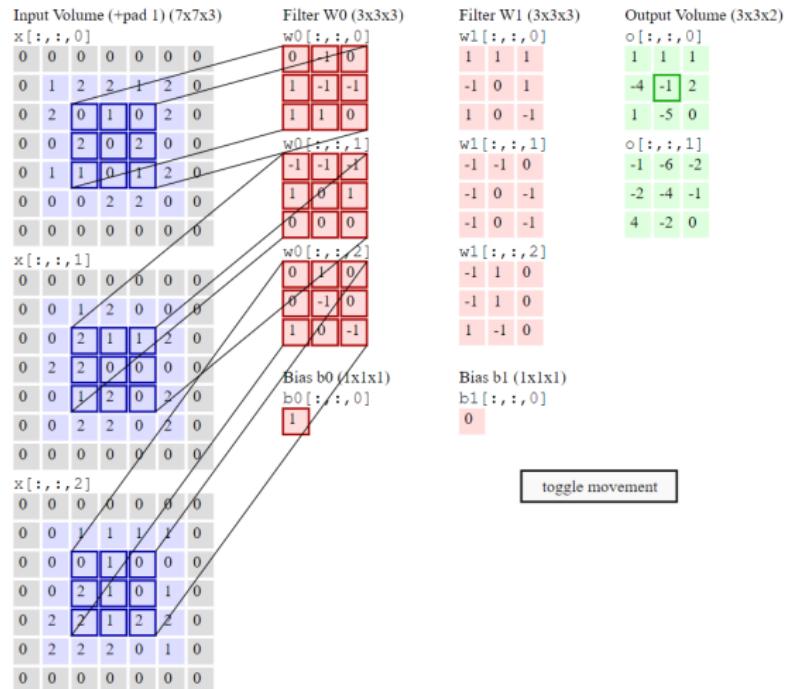


Sieci splotowe



Rysunek: Źródło

Sieci splotowe

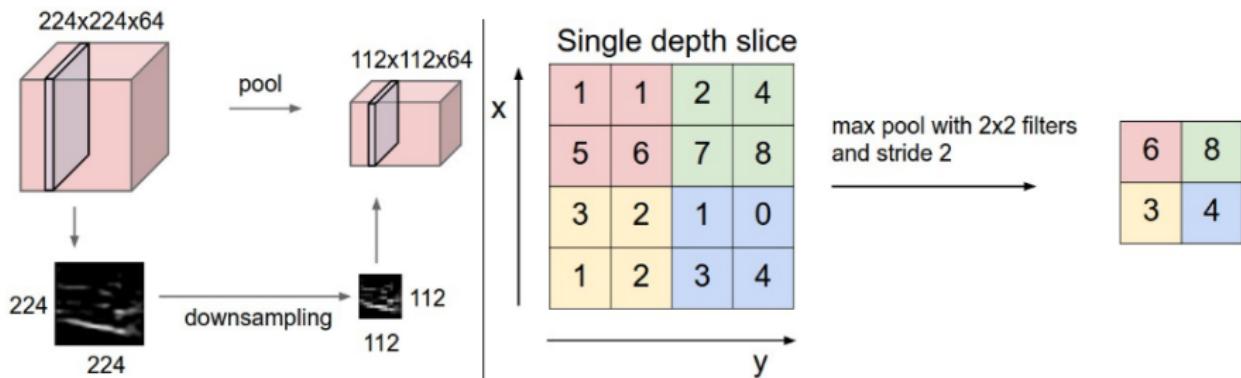


Rysunek: Źródło

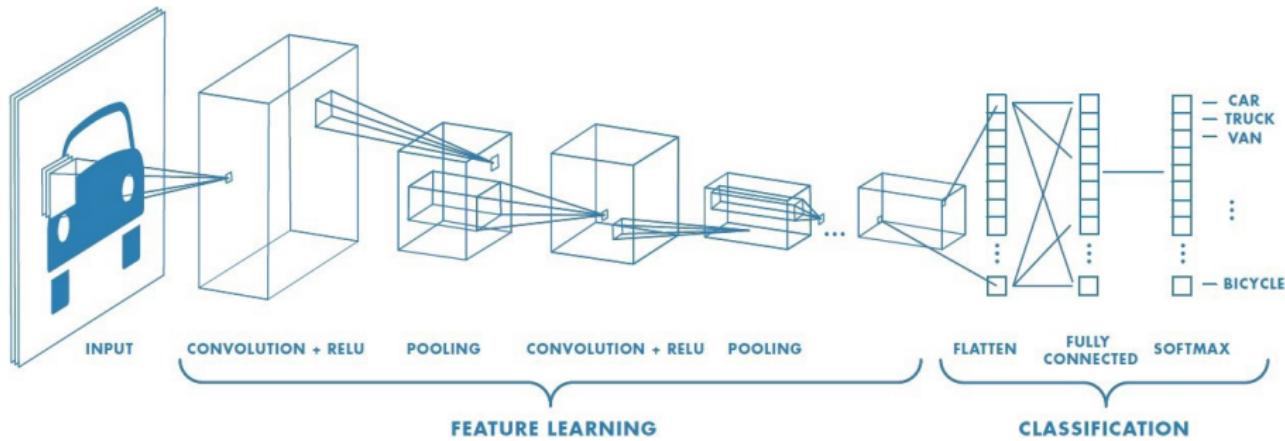
Pooling

Pooling

Pooling polega na zmniejszeniu wymiarowości w ten sposób, że patrzymy na obszar o pewnym rozmiarze i dla każdego rozpatrywanego obszaru zapisujemy min/max/avg z wartości w tym obszarze.

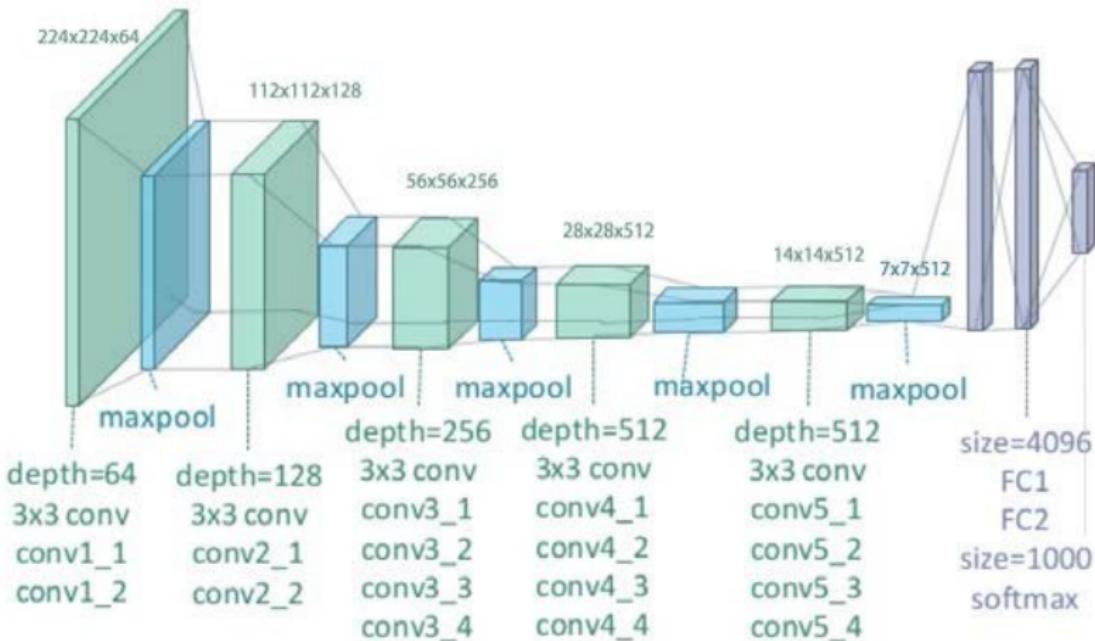


Sieci splotowe z warstwami wyciągającymi



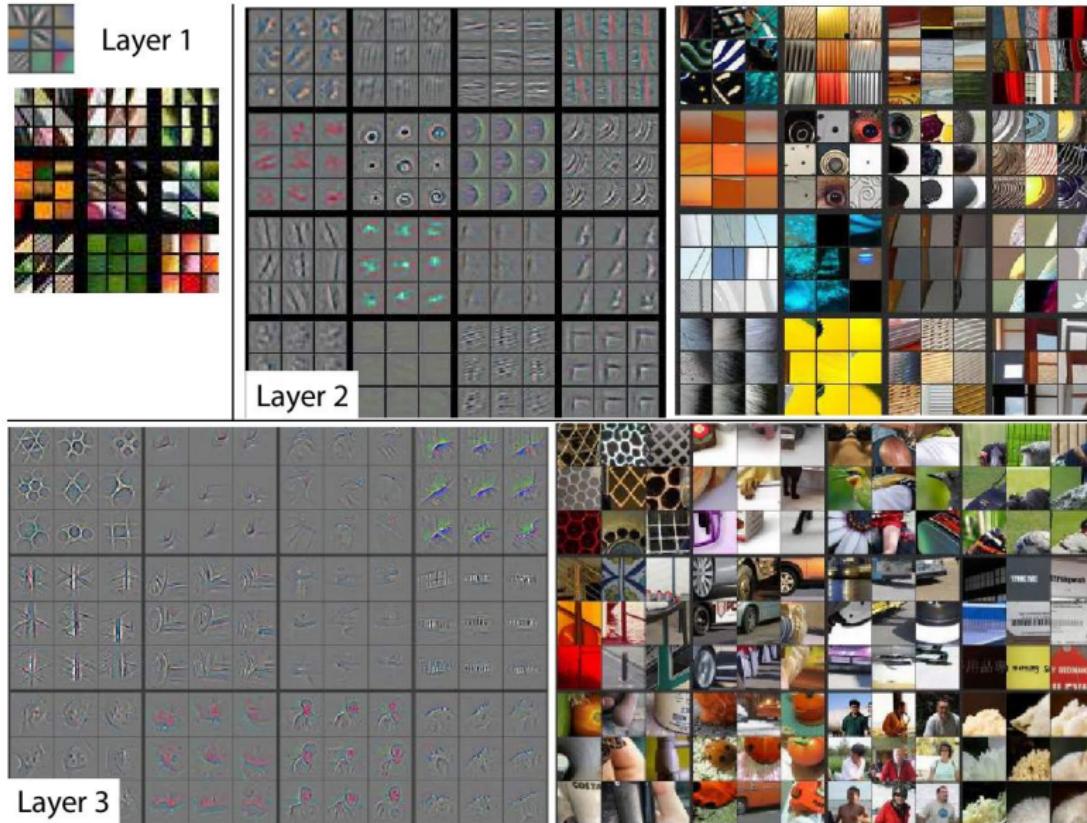
Rysunek: Źródło

VGG

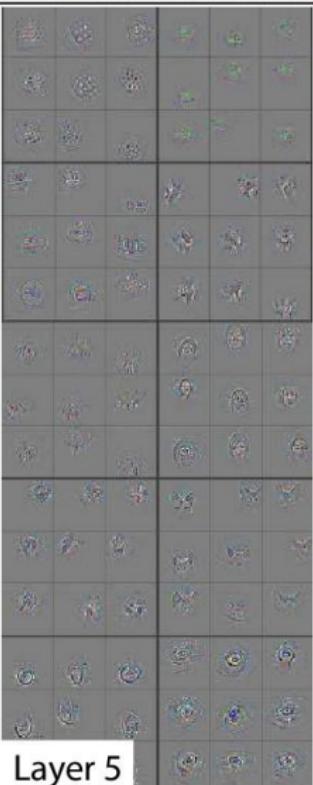
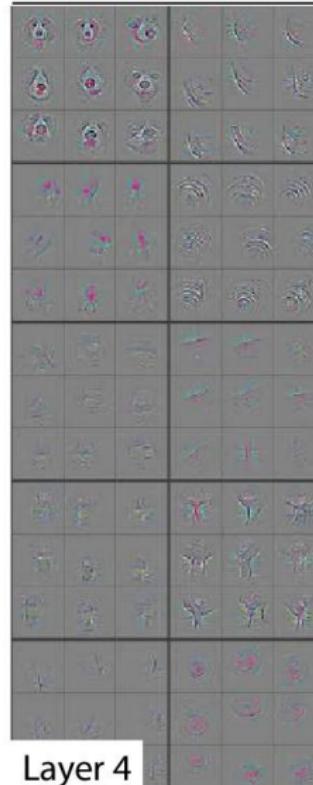


Rysunek: Źródło

Co robią pojedyncze neurony?



Co robią pojedyncze neurony?



Wykrywanie obiektów

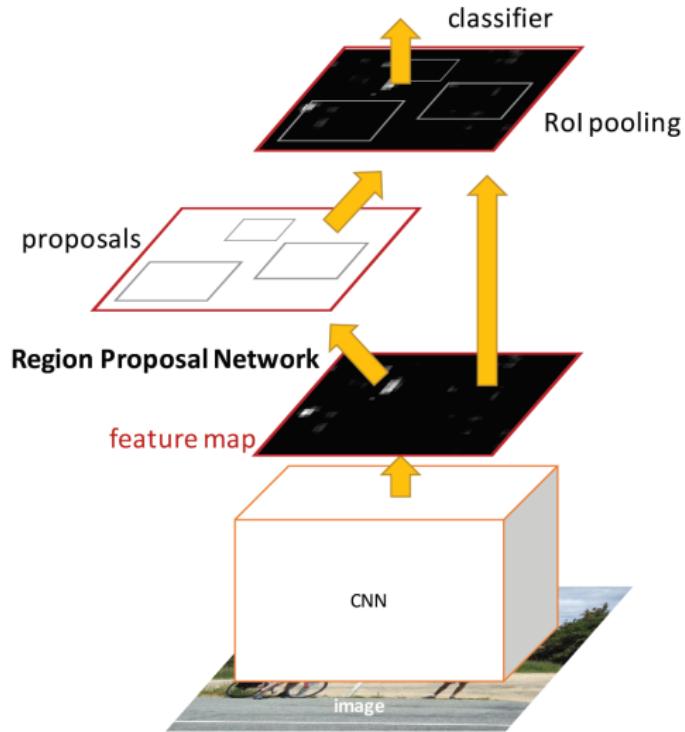
Podejście naiwne

W celu wykrycia, czy w danym miejscu zdjęcia znajduje się jakiś obiekt (i ewentualnie jaki to jest obiekt) moglibyśmy zastosować prosty, ale bardzo nieefektywny algorytm polegający na próbie klasyfikacji każdego fragmentu obrazu o wszystkich możliwych rozmiarach.

Inne podejścia

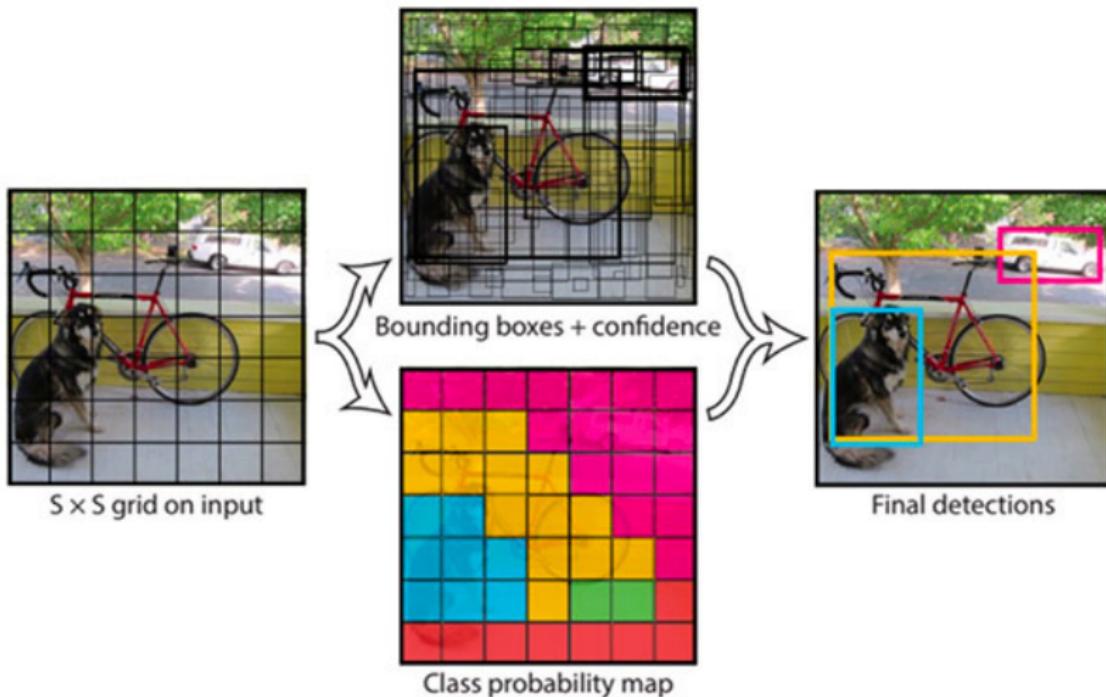
Powstało jednak wiele metod pozwalających rozwiązać ten problem w bardziej efektywny sposób. Jednymi z bardziej popularnych są metody fast R-CNN oraz YOLO.

Fast R-CNN



Rysunek: Źródło

Yolo - you only look once



Rysunek: Źródło

Źródła

- Wykłady CS Stanford
- Wykłady Machine Learning oraz Deep Learning Doktora Jana Chorowskiego
- Bishop - Pattern Recognition And Machine Learning
- <https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/svmtutorial.pdf>
- <https://arxiv.org/pdf/1311.2901.pdf>

Dziękuję za uwagę