Przetwarzanie równoległe w systemach wbudowanych

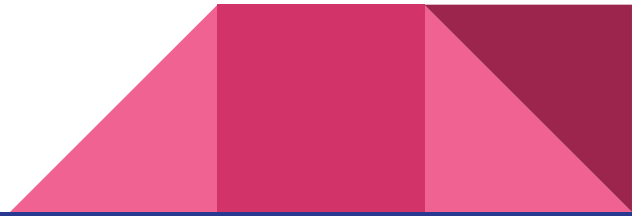
Mgr inż. Paweł Pietrzak
pawel.pietrzak@doctorate.put.poznan.pl

Informacje ogólne

- Ocena za realizację poszczególnych laboratoriów z wagą 1
- Ocena za realizację projektu z wagą 3
- Ocena końcowa to średnia ważona wszystkich ocen

Wymagania wstępne:

- Znajomość podstaw języka Python
- Znajomość podstaw sieci neuronowych

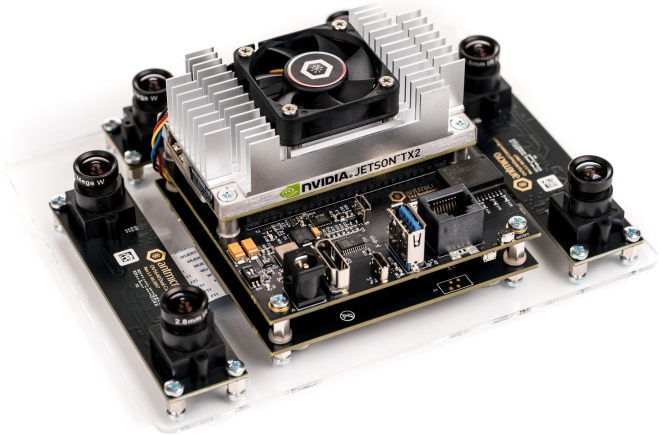


Antmicro TX2 Deep Learning Kit

- Oparty o Nvidia Jetson TX2:

GPU	256-core NVIDIA Pascal™ GPU architecture with 256 NVIDIA CUDA cores
CPU	Dual-Core NVIDIA Denver 2 64-Bit CPU Quad-Core ARM® Cortex®-A57 MPCore
Memory	8GB 128-bit LPDDR4 Memory 1866 MHx - 59.7 GB/s
Storage	32GB eMMC 5.1
Power	7.5W / 15W

- 6 kamer OV5640 z rozdzielczością 1920x1080 pikseli i 30 fps (wartość maksymalna)



CPU i GPU współdzielą tę samą przestrzeń adresową w pamięci głównej!

Wprowadzenie do podstawowych zagadnień z przetwarzania obrazów

Mgr inż. Paweł Pietrzak
pawel.pietrzak@doctorate.put.poznan.pl

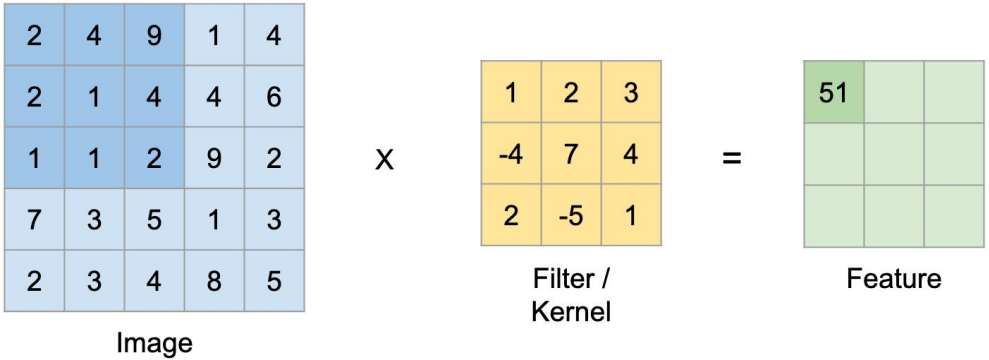
Konwolucja/splot - informacje ogólne

- Najważniejsza operacja w dziedzinie przetwarzania obrazów
- Wykorzystywana w większości filtrów dostępnych w OpenCV (bilateral, gaussian, etc)
- Wykorzystywana w **konwolucyjnych** sieciach neuronowych
- Efekty możliwe do uzyskania na obrazach dzięki niej to m.in.
  - Wygładzenie
  - Wyostrzenie
  - Redukcja szumu
  - Detekcja krawędzi
- Określenia konwolucja i splot są używane zamiennie w przetwarzaniu obrazów

Splot to mnożenie odpowiadających wartości dwóch funkcji w określonym przedziale i sumowanie wyników ze sobą.

$$(f * g)[n] = \sum_{m}^{n} f[m]\, g[n - m]$$

Konwolucja/splot - zasada działania



Funkcją pierwszą jest obraz, a funkcją drugą jest okno. Przemnożenie odpowiadających wartości i dodanie ich realizuje operację splotu matematycznego.

Kolejność działań w konwolucji:

1. Dopasowujemy okno (inaczej kernel, filtr, jądro) z “pewnymi” wartościami do obrazu tak by anchor point (zwykle punkt środkowy kernela) pokrywał się z pixelem, którego wartość będziemy modyfikować.
2. Mnożymy odpowiadające sobie wartości, po czym je dodajemy, otrzymując wynikową wartość pixela po przekształceniu oknem.
3. Proces powtarzamy dla reszty pixeli.

Konwolucja/splot - podsumowanie

Rzeczy do zapamiętania:

1. Rodzaj przekształcenia obrazu zależy od wartości w kernelu. Wartości te mogą być wyznaczane przez różne funkcje matematyczne, np. funkcję Gaussa.

2. Rozmiary okien mogą być różne. Najczęściej spotykane to 3x3, 5x5, 7x7 pikseli.

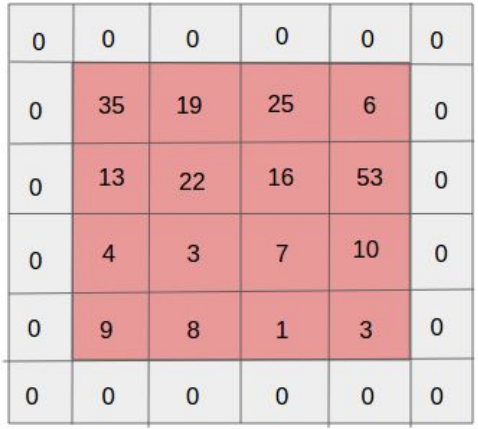
3. Niektóre metody filtrowania opierają się na dopasowywaniu okna i przesuwaniu go podobnie jak w konwolucji, ale nie stosują operacji splotu (mnożenia i dodawania). Przykładem jest filtr medianowy, który zastępuje dany pixel jego medianą z sąsiedztwa spod przyłożonego do obrazu okna.

4. Wynikową wartość pixela po konwolucji można dalej modyfikować, przykładowo poprzez podanie jej na wejście jakiejś funkcji matematycznej, np sigmoidalnej lub ReLU.

5. Nakładanie okna na obraz, robienie “pewnej” operacji w jego obszarze i przesuwanie go po całym obrazie jest najważniejszą czynnością w przetwarzaniu obrazów.

Problem konwolucji na pixelach z brzegu obrazu

Rozwiązanie: **padding**



0	0	0	0	0	0
0	35	19	25	6	0
0	13	22	16	53	0
0	4	3	7	10	0
0	9	8	1	3	0
0	0	0	0	0	0

W przypadku stosowania konwolucji na pixelach brzegowych może dojść do sytuacji, w której część okna będzie wystawała poza obraz. W takim przypadku możemy albo pominąć ten pixel, albo zastosować tak zwany **padding**, czyli dodanie dodatkowego obramowania złożonego z zer do obrazu.

Wprowadzenie do konwolucyjnych sieci neuronowych

Mgr inż. Paweł Pietrzak
pawel.pietrzak@doctorate.put.poznan.pl

Sieci neuronowe feed-forward - powtórzenie

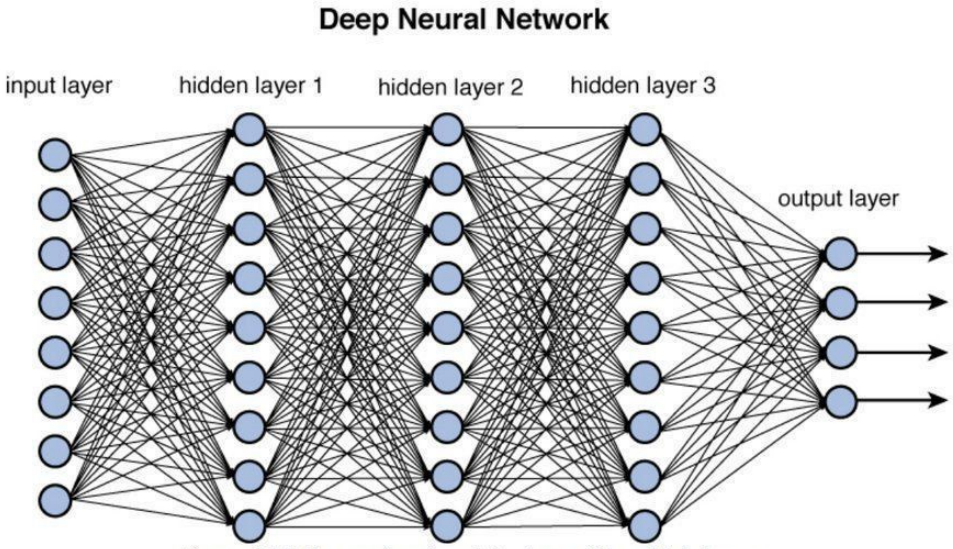


Figure 12.2 Deep network architecture with multiple layers.

Warstwy takich sieci są określane jako densely connected layers albo fully connected layers. W tych sieciach każdy neuron z poprzedniej warstwy ma po jednym połączeniu z każdym neuronem w następnej warstwie. Warstwy te można również podzielić ze względu na położenie w sieci:

- wejściowe (input layers)
- ukryte (hidden layers)
- wyjściowe (output layers)

Sieci neuronowe feed-forward - powtórzenie

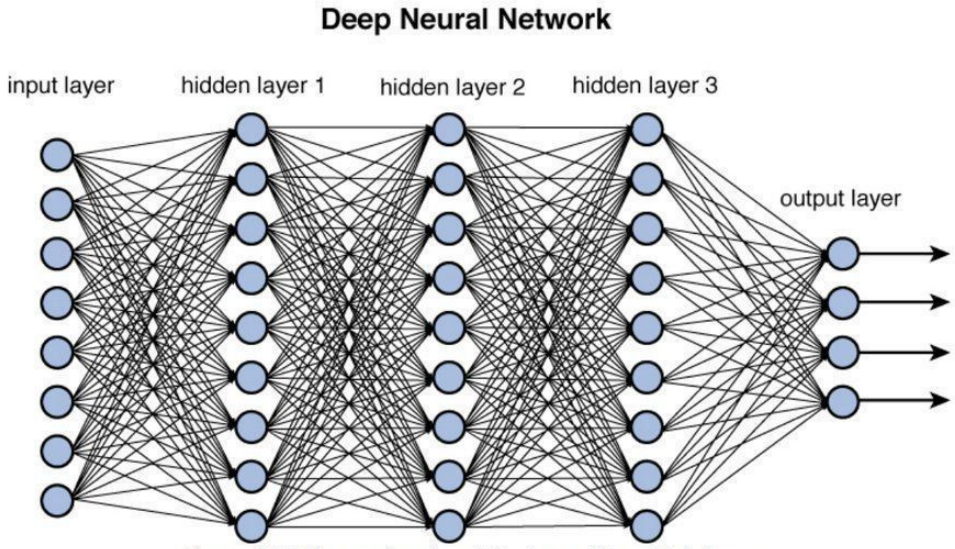
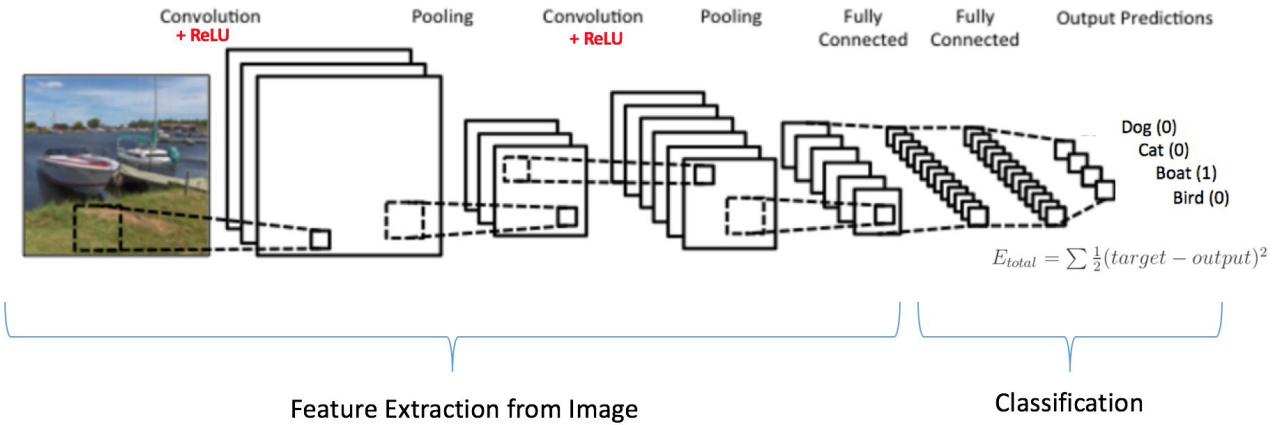


Figure 12.2 Deep network architecture with multiple layers.

Każde połączenie posiada oddzielną wagę, której wartość jest wyznaczana na podstawie algorytmu wstecznej propagacji. Głównym zadaniem tego algorytmu jest minimalizacja wartości funkcji kosztu (loss function).

Do każdego zadania, które ma realizować sieć, odpowiednia jest inna funkcja kosztu. Z tego względu jej wybór przed treningiem jest kluczowy.

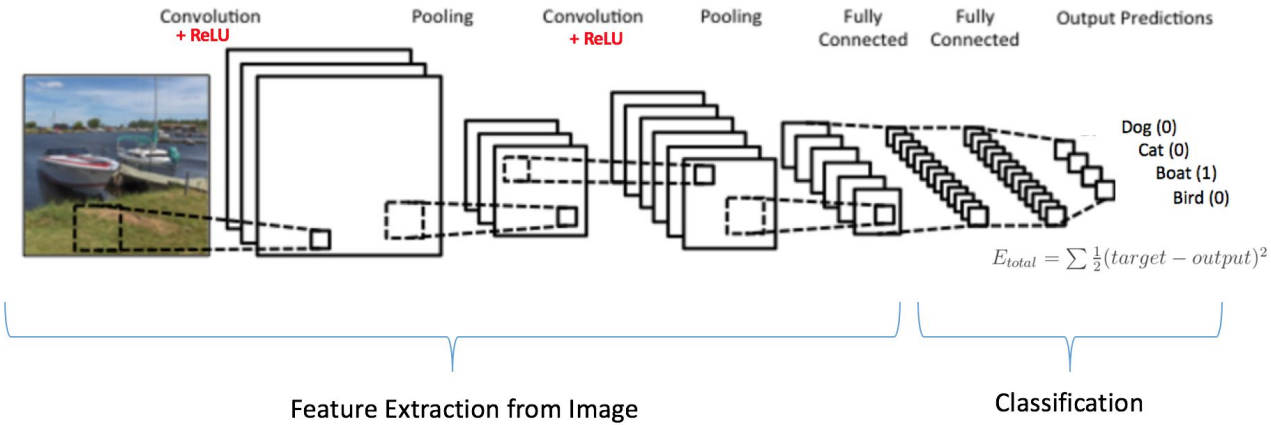
Konwolucyjne sieci neuronowe - budowa ogólna



Konwolucyjne sieci neuronowe składają się z dwóch części:

1. części odpowiedzialnej za ekstrakcję pożądanych cech w obrazie (odfiltrowanie niepotrzebnych informacji)
2. części odpowiedzialnej za wnioskowanie (zwykle jest to zadanie klasyfikacji obrazów)

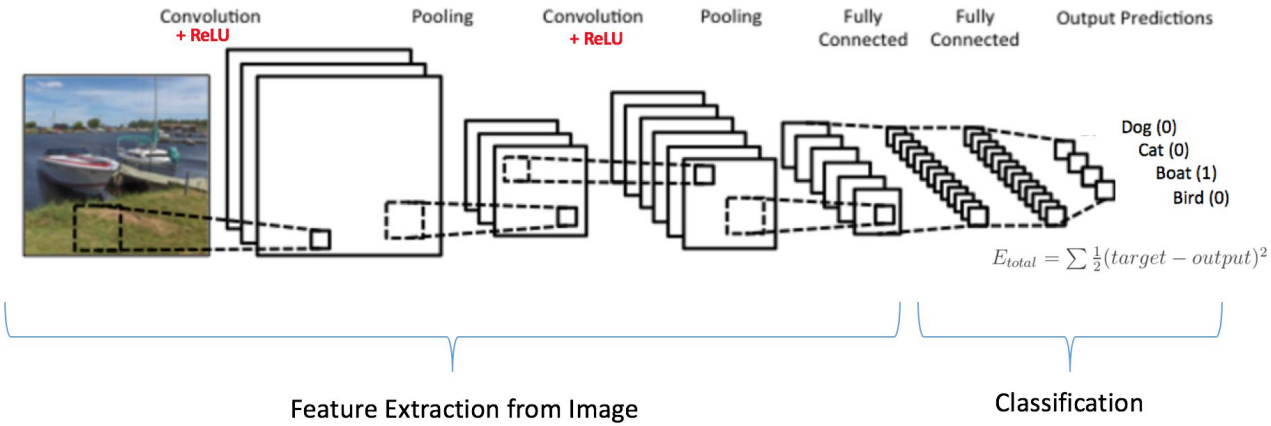
Konwolucyjne sieci neuronowe - budowa ogólna



Część pierwsza sieci neuronowej wykorzystuje wcześniej omówiony mechanizm konwolucji z tą różnicą, że wynik konwolucji jest podawany na wejście funkcji (najczęściej ReLU).

Część druga sieci neuronowej to zwykle “tradycyjna” sieć neuronowa (feed-forward neural network).

Konwolucyjne sieci neuronowe - inteligentne filtry

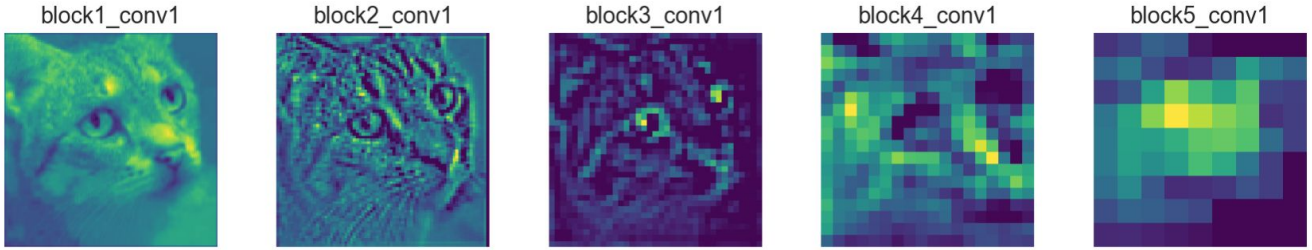


Parametry w oknach warstw konwolucyjnych są traktowane jak wagi połączeń w sieciach neuronowych feed forward i również podlegają uczeniu wcześniej wspomnianym algorytmem propagacji wstecznej.

Ten fakt sprawia, że filtry, które są realizowane przez konwolucyjne warstwy potrafią się uczyć ekstrakcji pożądanych cech w trakcie treningu.

Wytrenowane filtry w warstwach konwolucyjnych mogą realizować różne przekształcenia na obrazie. Dla przykładu pierwszy filtr może wygładzać obraz, a kolejny odfiltrowywać wszystko poza krawędziami. Nie ma tutaj specjalnej reguły.

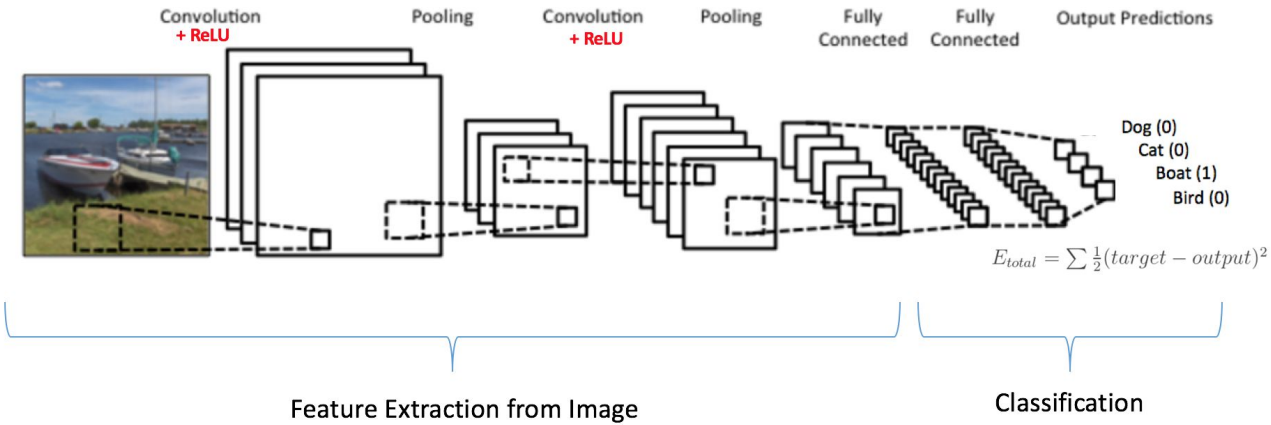
Konwolucyjne sieci neuronowe - inteligentne filtry



Wynikowymi obrazami warstwy konwolucyjnej są tak zwane mapy cech (feature maps), które prezentują wyciągnięte cechy z obrazu.

W konwolucyjnych sieciach neuronowych często stosuje się wiele warstw konwolucyjnych pod rząd, co sprawia, że z wynikowych map cech również wyciągane są mapy cech. Oprócz tego jedna warstwa konwolucyjna posiada zwykle więcej niż jeden filtr. Powyższe fakty sprawiają, że ilość obliczeń w tych sieciach jest bardzo duża i wymaga zrównoleglenia, np. poprzez wykorzystanie GPU.

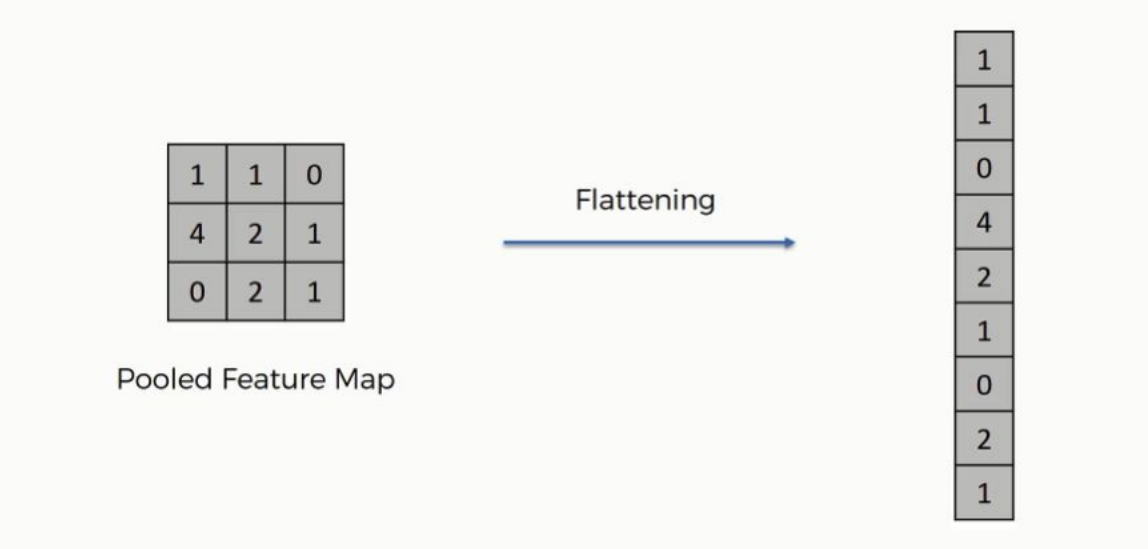
Konwolucyjne sieci neuronowe - pooling



W związku z dużą złożonością obliczeniową i nadmiarem informacji w mapach cech, stosuje się tak zwane pooling layers, czyli warstwy, które zmniejszają ilość informacji w wynikowych mapach cech.

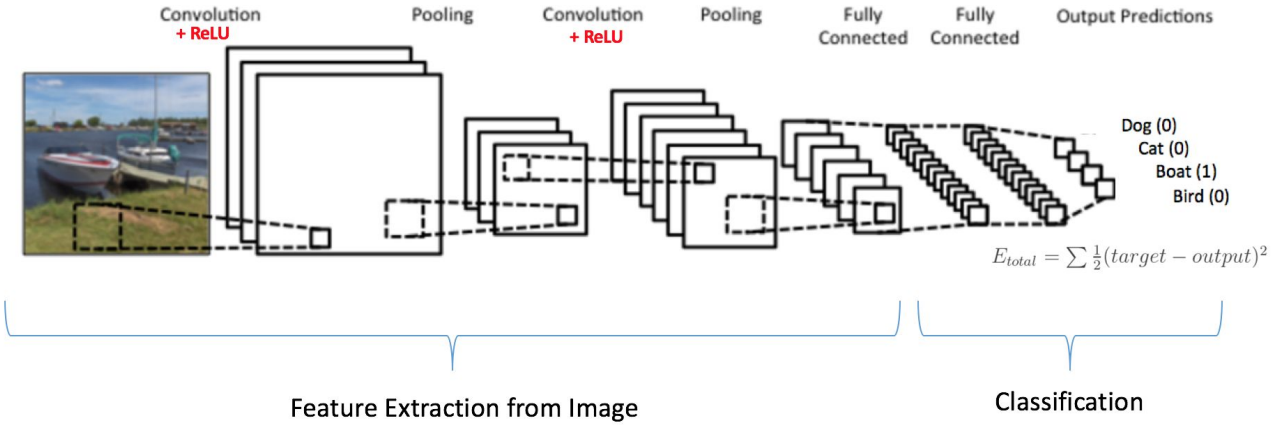
Taka warstwa używa wspomnianego mechanizmu przesuwania okna po obrazie i zastępowaniu odpowiedniego pixela inną wartością, np maksymalną z danego okna (max pooling). W tych warstwach nie stosuje się paddingu, więc wynikowe mapy cech mają zmniejszone rozmiary względem wejściowych. Przyspiesza to działanie sieci.

Konwolucyjne sieci neuronowe - wypłaszczenie



Na końcu części odpowiedzialnej za ekstrakcję cech wszystkie wynikowe feature maps są poddawane operacji wypłaszczania (flatten). Powoduje to ułożenie wszystkich wartości pixeli z map cech w jeden wektor, który wchodzi na warstwę wejściową części odpowiedzialnej za wnioskowanie.

Konwolucyjne sieci neuronowe - podsumowanie



Dobre praktyki - ściągawka

1. Przy warstwach konwolucyjnych rozmiar map cech może być mniejszy niż wejściowy obraz. Dzieje się tak, gdy nie zastosujemy paddingu do konwolucji. Zwykle lepiej jest go stosować z uwagi na fakt, że wtedy cechy bliżej granic obrazu zostaną lepiej przyswojone przez warstwy konwolucyjne.
2. Ilość filtrów (okien, kerneli) w danej warstwie konwolucyjnej jest ważnym hiper-parametrem. Z reguły przyjmuje się, że im dalsza warstwa konwolucyjna, tym więcej filtrów. Ilość filtrów często rośnie dwukrotnie w kolejnych warstwach konwolucyjnych.
3. Rozmiar okien w warstwach konwolucyjnych odpowiada za obszar, na którym dana cecha występuje. Zwykle stosuje się rozmiary okien 3x3, 5x5, a rzadziej 7x7.
4. W przypadku warstw poolingowych stosuje się zwykle rozmiary okien równe 2x2.
5. Warstwy poolingowe stosuje się tylko po warstwach konwolucyjnych.

W przeciwieństwie do rozwiązań analitycznych, w sieciach neuronowych zwykle nie jest wiadome ile wynoszą odpowiednie parametry dla danej sieci. Wyznaczyć je należy metodą eksperymentalną!

Zwykle najlepszym podejściem jest wzorowanie się na już zaimplementowanej sieci, która realizuje podobne zadanie do naszego.

Rodzaje klasyfikacji - ściągawka

Najczęściej używaną funkcją aktywacji w warstwach konwolucyjnych i ukrytych jest ReLU. W przypadku warstwy wyjściowej używa się różnych funkcji aktywacji do różnych zadań klasyfikacji:

- Binary classification (obraz przynależy lub nie do danej klasy)
  - Na wyjściu uzyskujemy prawdopodobieństwo przynależności do danej klasy
  - Wyjściowa funkcja aktywacji: sigmoid
  - Ilość neuronów w warstwie wyjściowej: 1
  - Funkcja kosztu: binary crossentropy
- Multiclass classification (obraz przynależy do jednej z klas)
  - Na wyjściu uzyskujemy prawdopodobieństwa przynależności do każdej z klas, których suma zawsze wynosi 1
  - Wyjściowa funkcja aktywacji: softmax
  - Ilość neuronów w warstwie wyjściowej: równa liczbie klas
  - Funkcja kosztu: categorical crossentropy
- Multilabel classification (obraz przynależy do żadnej, jednej lub więcej niż jednej klasy)
  - Na wyjściu uzyskujemy prawdopodobieństwa przynależności do każdej z klas, których suma nie musi wynosić 1
  - Wyjściowa funkcja aktywacji: sigmoid
  - Ilość neuronów w warstwie wyjściowej: równa liczbie klas
  - Funkcja kosztu: binary crossentropy