Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH

Sieci neuronowe

(projekt)

Wykorzystanie sieci VGG19 do klasyfikacji owoców

Drelich Ewelina, Dziurlikowski Krzysztof,
Pawlak Iga, Pierczyk Krzysztof

Warszawa, 21 stycznia 2021

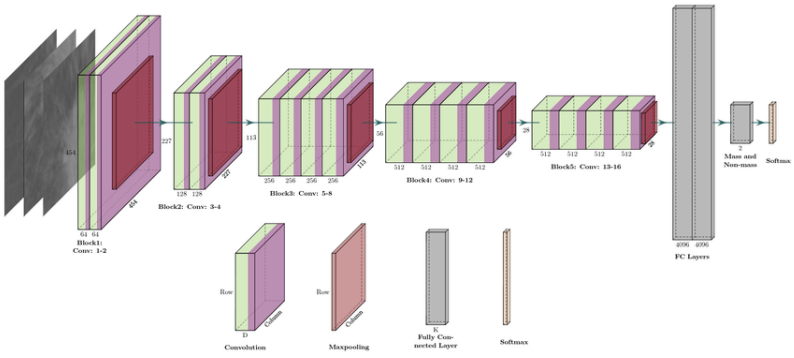
Spis treści

Sztuczne sieci neuronowe na stałe zadomowiły się w dziedzinie, którą dzisiaj powszechnie określamy mianem sztucznej inteligencji. Algorytmy tworzone przez firmy jak Google potrafią już same uczyć się operowania w tak złożonych grach jak szachy czy Starcraft II znacząco przewyższając wynikami ludzi [1]. Coraz częściej pojawiają się również w bardziej egzotycznych obszarach sterując balonami stratosferycznymi [2] czy przewidując struktury przestrzenne długich łańcuchów aminokwasowych [3].

Jednym z klasycznych zastosowań sieci neuronowych jest klasyfikacja obrazów. Wśród najpowszechniej używanych w tym celu architektur znajduje się od dłuższego czasu zaproponowana w 2014 roku *VGG*. Niniejsza praca skupia się na jednym z wariantów tego modelu - VGG19 - analizując jego możliwości w kontekście klasyfikacji obrazów owoców ze zbioru *Fruits-360*. Pierwsze trzy rozdziały stanowią opis postawionego problemu, wykorzystanej architektury oraz zbioru danych. Rozdział 4 opisuje przypadki uczenia klasyfikatorów typu perceptronowego oraz SVM bazujących na cechach generowanych przez warstwy splotowe sieci VGG19 uprzednio wytrenowanej na zbiorze *ImageNet*. Następnie przedstawiony został trening części klasyfikującej (typu perceptronowego) wraz z częścią lub wszystkimi warstawmi splotowymi. Przedostatni rozdział zgłębia analizę wytrenowanych sieci wykorzystując techniki wizualizacji obszarów uwagi oraz stopnia aktywacji poszczególnych warstw sieci.

1 Architektura VGG19

Akronim VGG pochodzi od nazwy grupy badawczej *Visual Geometry Group* z uniwersytetu w Oxfordzie będącej autorem analizowanej sieci. W 2014 roku zaproponowana przez naukowców z *VGG* architektura zdobyła pierwsze i drugie miejsce kolejno w kategorii lokalizacji i klasyfikacji konkursu ImageNet [4]. Kluczową cechą ich modelu było zminimalizowanie wielkości filtrów warstw splotowych (a co za tym idzie ilości związanych z nimi parametrów) na rzecz zwiększenia ilości warst.



Rysunek 1: Struktura sieci VGG19

VGG występuje w kilku głównych wariantach oznaczanych przez VGGx, gdzie x określa liczbę warstw sieci, których parametry są modyfikowanie w czasie treningu. Rys. 1 prezentuje strukturę VGG19. Składa się ona z trójwarstwowej części perceptronowej poprzedzonej kilkoma blokami konwolucyjnymi. W skład każdego bloku wchodzi szereg warstw

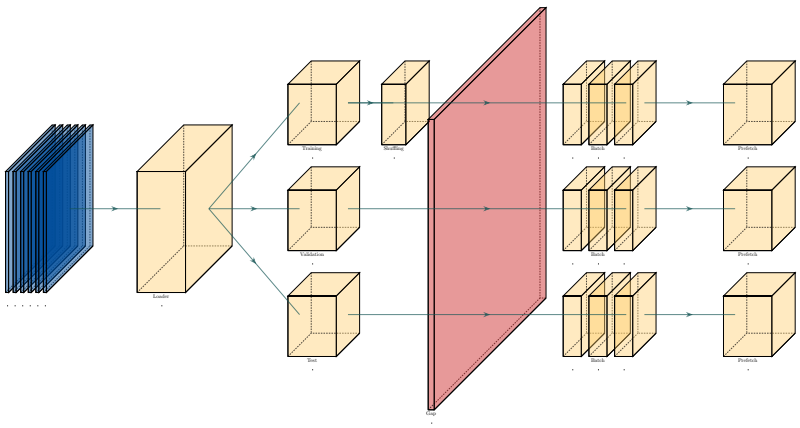
splotowych zakończony warstwą agregującą typu *max pooling*. Rozmiar pola recepcyjnego wynosi odpowiedni 3×3 dla warstw splotowych i 2×2 dla warstw agregujących. Warto w tym miejscu zauważyć, że *efektywne* pole recepcyjne dwóch połączonych warstw konwolucyjnych 3×3 ma wymiar 5×5 [5]. Podejście zaproponowane w VGG ma jednak tę zaletę, że utylizuje większą ilość warstw nieliniowych, co według twórców przekłada się na ubogacenie fuknccji decyzyjnej.

Ilość kanałów w warstwach splotowych zwiększa się dwukrotnie w każdym kolejnym bloku począwszy od 64 w bloku wejściowym. Parametr *stride* został ustalony na 1 w warstwach splotowych oraz na 2 w agregujących. Część perceptronowa składa się z trzech warstw w pełni połączonych o odpowiednio 4096, 4096 i 1000 neuronach. W warstwach ukrytych zastosowano nieliniowość typu $ReLU$. Jedynie warstwa wyjściowa (klasyfikująca) wykorzystuje funkcję typu *softmax*.

2 Zbiór danych

W projekcie wykorzystana została baza danych **Fruits-360** [6], która inkorporuje zbiór ponad 90 tys. zdjęć owoców podzielonych na 131 klas. Zdjęcia dostarczone są w formacie JPEG, a ich wielkość została ustandaryzowana do wymiaru 100×100 pikseli. Warto nadmienić, że różne warianty tych samych gatunków owoców zostały umieszczone w oddzielnych klasach. Dane zostały podzielone przez autorów na dwa podzbiory: *Training* (67692 zdjęć) i *Test* (22688 zdjęć).

2.1 Potok



Rysunek 2: Struktura potoku danych wejściowych

Prace nad projektem rozpoczęto od wyboru bibliotek dostarczających zestaw podstawowych narzędzi uczenia maszynowego. Decyzja padła na popularny framework *Tensorflow*, dzięki któremu możliwe było przygotowanie wygodnego, wysoce konfigurowalnego potoku danych wejściowych. Wykorzystanie *tf.data* API pozwoliło na dynamiczne ładowanie danych do pamięci, co znacznie zredukowało jej zużycie w procesie uczenia. Możliwość zrównoleglenia tego procesu względem obliczeń wykonywanych na procesorze graficznym

usunęła występujące początkowo wąskie gardło w postaci procesu przenoszenia danych z pamięci operacyjnej do VRAMu. Całość rozwiązania została zamknięta w postaci klasy, której parametry mogłą być ustalane z poziomu plików konfiguracyjnych. Takie podejście wyeliminowało potrzebę ingerowania w kod źródłowy na etapie eksploatacji potoku.

Rys. 2.1 przedstawia graficzną reprezentację przepływu danych. Na wejściu następuje przydzielenie danych do trzech zbiorów. Podzbiór *Training* jest w całości wykorzystywany jako zbiór treningowy. Z kolei *Test* **dzielony jest w stosunku** $1:1$ na dane walidacyjne i testowe. W kolejnym kroku nastepuje losowe przemieszanie danych treningowych. Obszar zaznaczony na rysunku na czerwono oznacza lukę. W tym miejscu na zbiory można nałożyć arbitralne modyfikacje, co wykorzystywane jest na etapie augmentacji. Następnie wszystkie zbiory zostają podzielone na porcje (ang. *mini-batches*). Ostatecznie część danych zostaje pobrana do bufora w pamięci operacyjnej (faza *prefetch*).

2.2 Augmentacja

3 Klasyfikatory

3.1 Klasyfikator perceptronowy

3.2 Maszyna Wektorów Wspierających

3.3 Porównanie wyników

4 Sieci głębokie

4.1 Uczenie ostatniej warstwy splotowej

4.2 Uczenie dwóch ostatnich warstw splotowych

4.3 Uczenie pełnej sieci

4.4 Uczenie sieci o uproszczonej strukturze

4.5 Porównanie wyników

5 Wizualizacja

5.1 Class Activation Map

5.2 Deep Dream

6 Podsumowanie

Bibliografia

[1] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap i D. Silver, „Mastering Atari, Go, chess and shogi by planning with a learned model," *Nature*, nr. 588, s. 604–609, 2020.

[2] M. G. Bellemare, S. Candido, P. S. Castro, J. Gong, M. C. Machado, S. Moitra, S. S. Ponda i Z. Wang, „Autonomous navigation of stratospheric balloons using reinforcement learning," *Nature*, nr. 588, s. 77–82, 2020.

[3] A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. W. R. Nelson, A. Bridgland, H. Penedones, S. Petersen, K. Simonyan, S. Crossan, P. Kohli, D. T. Jones, D. Silver, K. Kavukcuoglu i D. Hassabis, „Improved protein structure prediction using potentials from deep learning," *Nature*, nr. 577, s. 706–710, 2020.

[4] K. Simonyan i A. Zisserman, „Very Deep Convolutional Networks for Large-Scale Image Recognition," 2015.

[5] W. Luo, Y. Li, R. Urtasun i R. Zemel, „Understanding the Effective Receptive Field in Deep Convolutional Neural Networks," 2017.

[6] H. Muresan i M. Oltean, „Fruit recognition from images using deep learning," *Informatica*, nr. 10, s. 26–42, 2018. adr.: `http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.htm`.