Konspekt projektu #2

z przedmiotu Metody Głębokiego Uczenia

Piotr Podbielski, grupa nr 3

26 marca 2019

Streszczenie

Konspekt zawiera opis zbioru oraz metod, które zostaną zastosowane do problemu generowania obrazów na podstawie zbioru *CIFAR-10*.

1 Wstęp

Niniejszy dokument powstał w wyniku drugiego projektu z przedmiotu *Metody Głębokiego Uczenia* wykładanego na kierunku *Inżynierii i Analizy Danych* w roku akademickim *2018/2019* na wydziale *Matematyki i Nauk Informacyjnych Politechniki Warszawskiej*.

Celem głównym projektu jest zaimplementowanie modelu *GAN* na podstawie artykułu Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville und Bengio (2014), który umożliwi generowanie obrazów na podstawie zbioru *CIFAR-10*. Implementacja musi spełniać następujące warunki:

- udostępnianie możliwości wyboru architektury modułów generatora i dyskryminatora w wariantach:
  - generator i dyskryminator są sieciami w pełni połączonymi,
  - generator i dyskryminator są sieciami, odpowiednio, konwolucyjną oraz "dekonwolucyjną",
- wizualizowanie funkcji kosztu na zbiorze treningowym i walidacyjnym podczas trenowania, oddzielnie dla modułu generatora i dyskryminatora,
- wizualizowanie obrazów generowanych przez moduł generatora w zależności od epoki,
- wizualizowanie par: wygenerowany obraz, obraz jemu najbliższy ze zbioru treningowego w przestrzeni pikseli,
- wizualizowanie obrazów uzyskanych przez interpolację przestrzeni liniowej.

Celami dodatkowymi, w przypadku wykonania projektu przed czasem, są:

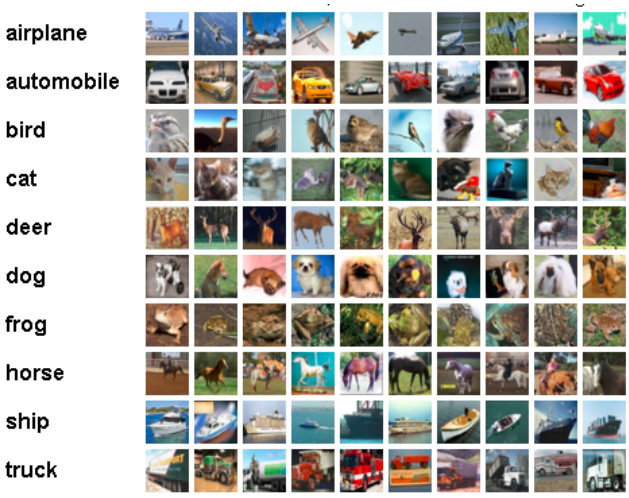
- zastosowanie architektury *DCGAN*,
- implementacja metody *Wasserstein'a*.

2 Narzędzia, jakie posłużą przy implementowaniu modelu

Model zostanie zaimplementowany w języku *Python* przy użyciu biblioteki *TensorFlow* w wersji `r1.13`. Wartości metryk oraz obrazy będą przekazywane podczas trenowania i walidacji do *TensorBoard'a*, co pozwoli na szybsze porównywanie poszczególnych modeli.

3 Zbiór danych CIFAR-10

Zbiór danych uczących i walidacyjnych dostępny jest na stronie Alex'a Krizhevsky'ego. Zbiór treningowy składa się z 50 tyś. przykładów, zaś walidacyjny z 10 tyś. Każdy z przykładów jest obrazem o rozmiarach 32x32 pikseli wraz z klasą, do której należy. Obrazy te przedstawiają środki lokomocji oraz zwierzęta. Przykładowy podzbiór przykładów widoczny jest na rysunku 1.



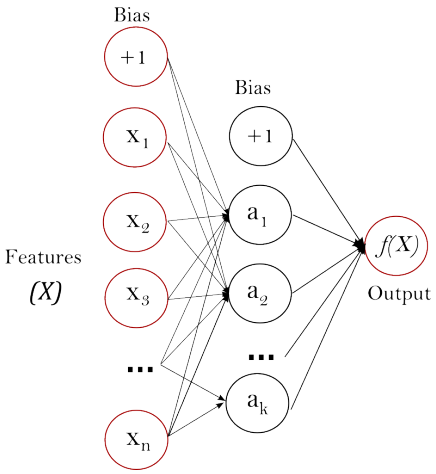
Rysunek 1: Przykładowy podzbiór przykładów ze zbioru CIFAR-10.

Każdy piksel obrazu reprezentują trzy wartości będące odwzorowaniem nasycenia w każdym z podstawowych kolorów (R, G, B). Wartości nasycenia pikseli zawierają się w skali od 0 do 255.

4 Opis metod

4.1 Perceptron wielowarstwowy (ang. *Multilayer perceptron*)

Perceptron wielowarstwowy jest typem sieci neuronowej z przepływem informacji w przód (ang. *Feedforward Neural Network*), co oznacza, że nie występują w niej pętle zwrotne. Podczas przepływu informacji *w przód*, w warstwie i wykorzystywane są tylko wartości aktywacji z warstwy $i-1$. W warstwach ukrytych *MLP* wartość aktywacji każdego neurona obliczana jest na podstawie kombinacji liniowej wszystkich wartości aktywacji neuronów z warstwy poprzedniej. Przykładowa architektura perceptronu wielowarstwowego została przedstawiona na rysunku 2.



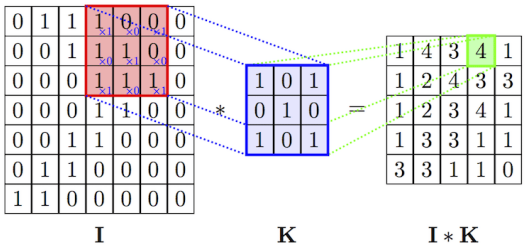
Rysunek 2: Przykładowa architektura perceptronu wielowarstwowego z jedną warstwą ukrytą. Źródło.

4.2 Convolutional Neural Networks

Sieci konwolucyjne są także typem sieci neuronowej z przepływem informacji w przód (ang. *Feedforward Neural Network*). Różnica w stosunku do *MLP* jest taka, że w sieciach konwolucyjnych dzięki użyciu splotu funkcji wartość aktywacji dla każdego z neuronów w warstwach ukrytych wyliczana jest na podstawie części neuronów z warstw poprzednich. Dzieje się to w taki sposób, że nie każdy neuron w warstwie ukrytej ma połączenie z wszystkimi neuronami z warstwy poprzedniej.

W przypadku obrazów zastosowanie konwolucji polega na przesuwaniu okna o pewnym rozmiarze po neuronach w warstwie i i zastosowywaniu na tych neuronach splotu z pewnym filtrem, który reprezentują wagi warstwy ukrytej. Filtr ten podlega nauce przez sieć podczas procesu trenowania za pomocą propagacji wstecznej.

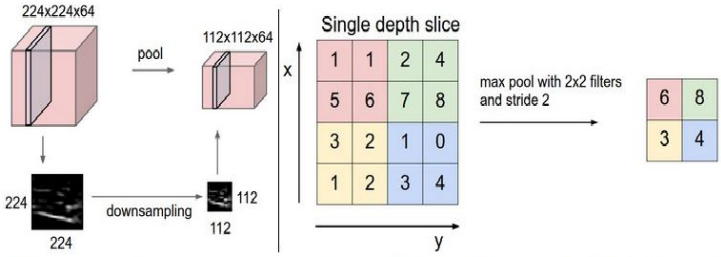
Przykład działania konwolucji (splotu) w sieciach konwolucyjnych został przedstawiony na rysunku 3.



Rysunek 3: Przykład działania funkcji konwolucji dwuwymiarowej, która oblicza wartość neuronu w kolejnej warstwie na podstawie części neuronów z poprzedniej warstwy. Źródło.

Każda z warstw konwolucji w *CNN* składa się z filtru, który przesuwany jest po neuronach w warstwie poprzedniej co pewną wartość *stride*. Dodatkowo neurony w warstwie poprzedniej mogą zostać wzbogacone o sztuczneńeurony zwane *padding'iem*. W zależności od *stride'u*, *padding'u* i kilku innych parametrów konwolucji uzyskujemy różną liczbę neuronów w warstwie konwolucji. *Padding* stosujemy wtedy, gdy chcemy, aby liczba neuronów w warstwie poprzedniej była równa liczbie neuronów w warstwie konwolucji.

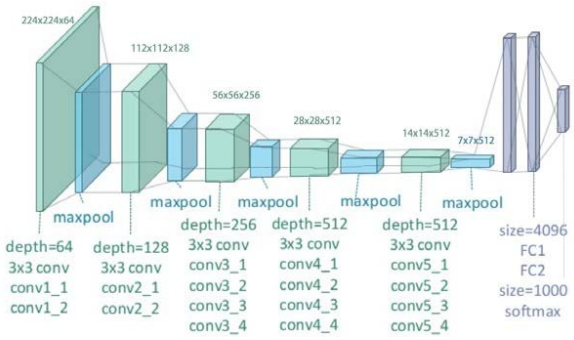
Do kazdej z warstw konwolucji dołączyć można warstwę zwaną *pooling'iem*. *Pooling* dwuwymiarowy ma za zadanie drastycznie zmniejszyć liczbę neuronów w każdej kolejnej warstwie sieci konwolucyjnej poprzez przejście pewnym zadanym oknem po neuronach warstwy poprzedniej i zagregowanie wartości z tego okna do jednej, która razem z wartościami z reszty okien tworzy wyjście z warstwy *pooling'u*. Przedstawiono to na obrazku 4.



Rysunek 4: Przykład działania *pooling'u* w sieciach konwolucyjnych. Źródło.

4.3 Deep Convolutional Neural Networks

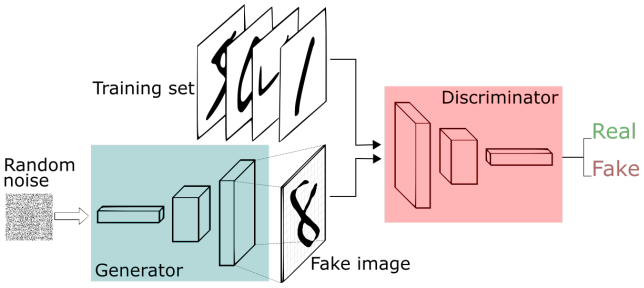
Głębokie sieci konwolucyjne są odmianą sieci konwolucyjnych opisanych w rozdziale 4.2, z dodatkowym założeniem, że muszą mieć wiele warstw ukrytych (często dziesiątki). Sieci takie potrafią mieć po kilkadziesiąt albo kilkaset milionów parametrów (parametry są to po prostu wagi, które są uczone przez sieć w trakcie trenowania). Przykładem głębokiej CNN jest architektura *VGG-19* przedstawiona na rysunku 5. Architektura ta ma 144 milionów parametrów.



Rysunek 5: Przykład głębokiej sieci konwolucyjnej *VGG-19*. Źródło.

4.4 General Adversarial Networks

GAN są dosyć nowymi sieciami, bo wymyślonymi w 2014 r. przez *Ian'a Goodfellow'a*. *GAN'y* są modelami generatywnymi i jest to swego rodzaju gra dwóch sieci (zwanych dalej modułami), które konkurują ze sobą. Moduł generatywny (ang. *generative*) odpowiedzialny jest za generowanie pewnej wyuczonej przez moduł reprezentacji z zadanego wektora wejściowej (przeważnie losowego, będącego szumem). Drugi zaś moduł, zwany dyskryminatorem (ang. *discriminator*) odpowiedzialny jest za klasyfikację tejże wygenerowanej reprezentacji, oraz reprezentacji ze zbioru treningowego w dwóch klasach: czy (1) dana reprezentacja jest prawdziwa, czy może (2) fałszywa. Trenowanie *GAN'ów* opiera się na przemiennym optymalizowaniu funkcji kosztu obu modułów. Schemat koncepcji *GAN* został przedstawiony na rysunku 6.



Rysunek 6: Koncepcja *GAN* zaproponowana przez *Ian'a Goodfellow'a* na przykładzie danych dwuwymiarowych. Źródło.

Kwestia architektury obu modułów jest kwestią wtórną. Zazwyczaj są to sieci konwolucyjne (np. *DCGAN*).

Literatura

[Goodfellow u. a. 2014] Goodfellow, Ian J. ; Pouget-Abadie, Jean ; Mirza, Mehdi ; Xu, Bing ; Warde-Farley, David ; Ozair, Sherjil ; Courville, Aaron ; Bengio, Yoshua: Generative Adversarial Networks. In: *arXiv e-prints* (2014), Jun, S. arXiv:1406.2661