Wydział Matematyki i Nauk Informacyjnych Politechniki Warszawskiej

Dokumentacja

Sieci konwolucyjne

Karol Ulanowski
Jakub Waszkiewicz

21 kwietnia 2020

Spis treści

1 Wstęp

1.1 Problem badawczy

Problemem projektu jest zbadanie sieci konwolucyjnych. Zawiera się w nim przegląd architektury takich sieci oraz zmierzenie wpływu jej poszczególnych czynników oraz parametrów na proces uczenia oraz uzyskaną skuteczność.

1.2 Cel pracy

Celem projektu jest przygotowanie sieci konwolucyjnej, która uzyska jak najlepszy wynik na platformie Kaggle dla zadania CIFAR-10.

1.3 Metody

Do stworzenia projektu planowane jest wykorzystanie biblioteki PyTorch. Dodatkowo sprawdzony zostanie wpływ następujących modyfikacji i usprawnień na działanie sieci [1, 2, 3]:

- **data agumentation** - sprawdzenie wpływu zastosowanie różnych przekształceń obrazu w celu poszerzenia zbioru danych treningowych,
- **transfer learning** - planowane jest sprawdzenie sieci AlexNet, VGG i ResNet,
- **dropout** - sprawdzenie wyłączania neuronów na różnych warstwach na skuteczność sieci oraz zjawisko przetrenowania,
- **pooling** - sprawdzenie różnych metod redukcji warstw,
- **stride i padding** - zbadanie wpływu tych parametrów na jakość uzyskanych cech,
- **rozmiar kerneli** - przetestowanie jak rozmiary filtrów wpływają na dokładność sieci,
- **liczba zastosowanych filtrów**,
- **rozmiar sieci**.

1.4 Opis danych

Do treningu i testowania zostaną wykorzystane dane CIFAR-10 dostępne na platformie Kaggle [4], będące zbiorem obrazów, dla których należy dokonać klasyfikacji. Zbiór zawiera 60 tys. obrazów o rozmiarach 32×32 w formacie .PNG podzielonych na 10 równolicznych klas: **airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck**.

Dane są podzielone na treningowe oraz testowe o liczności odpowiednio 50 i 10 tys. Przynależność do klas jest przedstawiona w formie tabeli o 2 kolumnach:

1. id – identyfikator (nazwa pliku) obrazu,
2. label – klasa, do której obraz jest zaklasyfikowany.

Dla zbioru testowego nie mamy dostępu do informacji o przydzielonych klasach dla poszczególnych obrazów. Dodatkowo na platformie Kaggle zbiór testowy jest uzupełniony o dodatkowe 290 tys. obserwacji, dla których wyniki klasyfikacji nie będą uwzględniane w ocenie skuteczności, a mają jedynie stanowić dodatkowe zabezpieczenie przed oszukiwaniem.

1.5 Sposób weryfikacji rezultatów

Z powodu braku etykiet dla zbioru testowego, zbiór treningowy będzie podzielony na zbiór treningowy oraz walidacyjny, na których będą przeprowadzane obliczenia. Każda z sieci będzie uczona do momentu wzrostu błędu na zbiorze walidacyjnym.

Eksperymenty dla identycznych konfiguracji zostaną przeprowadzone wielokrotnie, a wyniki zostaną przedstawione na postawie wartości oczekiwanej i odchylenia standardowego dokładności oraz funkcji straty dla serii. Wyniki otrzymane przez najlepszy model zostaną opublikowane na platformie Kaggle.

1.6 Wykorzystane technologie

Wszelkie testy zostały wykonane przy użyciu technolgii **Python**. Sieci zostały skonstruowane przy użyciu biblioteki **PyTorch**.

2 Przykładowy test

W tym rozdziale zostały przedstawione wyniki testów przeprowadzonych na przykładowej sieci konwolucyjnej, udostępnionej na oficjalnej stronie PyTorch [5]. Jest ona skonstruowana z następujących warstw ukrytych:

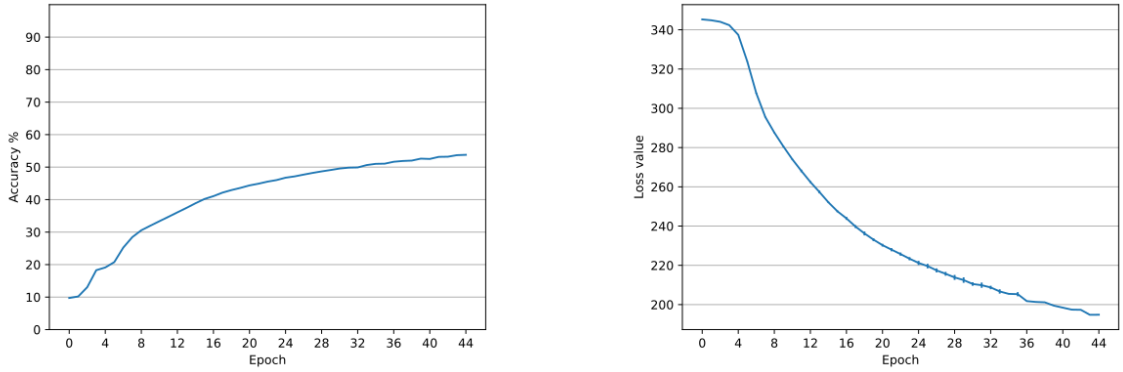
1. konwolucyjnej z 6 filtrami 5×5,
2. warstwa łącząca przy użyciu funkcji max i filtrów 2×2,
3. konwolucyjnej z 16 filtrami 5×5,
4. liniowej o 120 neuronach,
5. liniowej o 84 neuronach.

Należy także wspomnieć o tym, że dla wszystkich testów wartości wejściowe do sieci zostały znormalizowane do wartości średniej 0.5 i odchylenia 0.5 dla wartości każdego z kanałów.

Do przeprowadzenia testu została wykorzystana następująca konfiguracja:

- Zbiór został podzielony na treningowy oraz walidacyjny w stosunku $7 : 3$,
- Rozmiar batcha został ustawiony na 100,
- Współczynnik nauki miał wartość 0.0005,
- Współczynnik bezwładności miał wartość 0.9,
- Jako funkcja straty została wykorzystana CrossEntropy.

Przeprowadzono 20 testów dla różnego ziarna losowego. Na rysunku 1 przedstawiono wykresy średniej skuteczności oraz funkcji straty na zbiorze walidacyjnym w kolejnych epokach.



(a) Wykres skuteczności

(b) Wykres funkcji straty

Rysunek 1: Wyniki dla sieci testowej

Na obu wykresach także odchylenie standardowe. W przypadku skuteczności przyjęło ono wartości rzędu 0.001, dlatego są one praktycznie niewidoczne. Ostatecznie sieć uzyskała dla zbioru walidacyjnego około 53% skuteczności oraz wartość funkcji straty około 195.

3 Transfer Learning

Przetestowano następujące modele w ramach metody transfer learning:

- AlexNet,
- VGG19,
- ResNet18

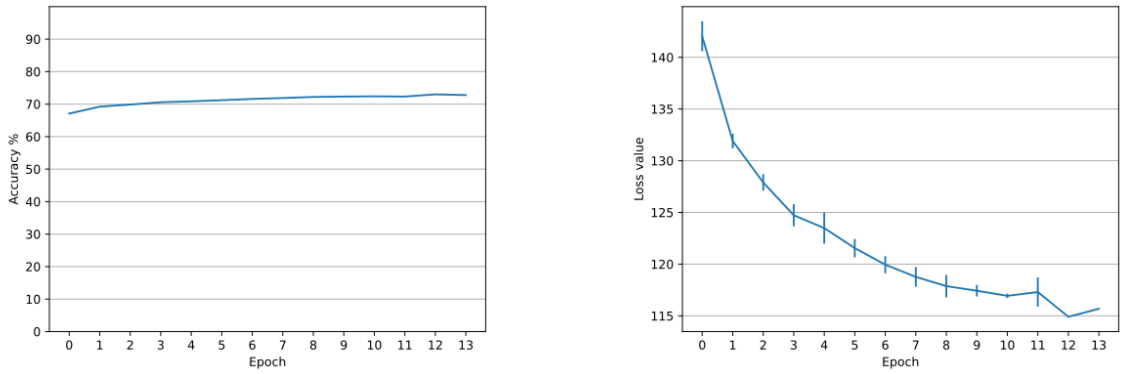
Dla każdej z tych sieci podmieniono ostatnią warstwę na warstwę liniową tak, aby sieć zwracała wyniki dla 10 klas. Dodatkowo zmieniono rozmiary obrazków wejściowych na $224px \times 224px$ tak, aby były zgodne z danymi wejściowymi przeznaczonymi dla sieci transferowanych. W testach trenowano jedynie wagi dla ostatniej warstwy. Pozostałe wagi były zamrożone. Parametry treningowe były identyczne dla wszystkich sieci i wyglądały następująco:

- Zbiór został podzielony na treningowy oraz walidacyjny w stosunku $7:3$,
- Rozmiar batcha został ustawiony na 100,
- Współczynnik nauki miał wartość 0.0005,
- Współczynnik bezwładności miał wartość 0.9,
- Jako funkcja straty została wykorzystana CrossEntropy.

Dla każdej sieci wykonano 10 identycznych testów z wykorzystaniem różnych ziaren losowych.

3.1 AlexNet

AlexNet to jedna z najstarszych znanych architektur sieci konwolucyjnych , której autorami są Krizhevsky, Sutskever, Hinton. W 2012 roku wygrała ILSVRC (*ImageNet Large-Scale Visual Recognition Challenge*) uzyskując błąd na poziomie 15.4%. Składa się się z 5 warstw konwolucyjnych, warstw łączących (z funkcją max), warstw dropout i 3 warstw ukrytych w pełni połączonych [3].



(a) Wykres skuteczności

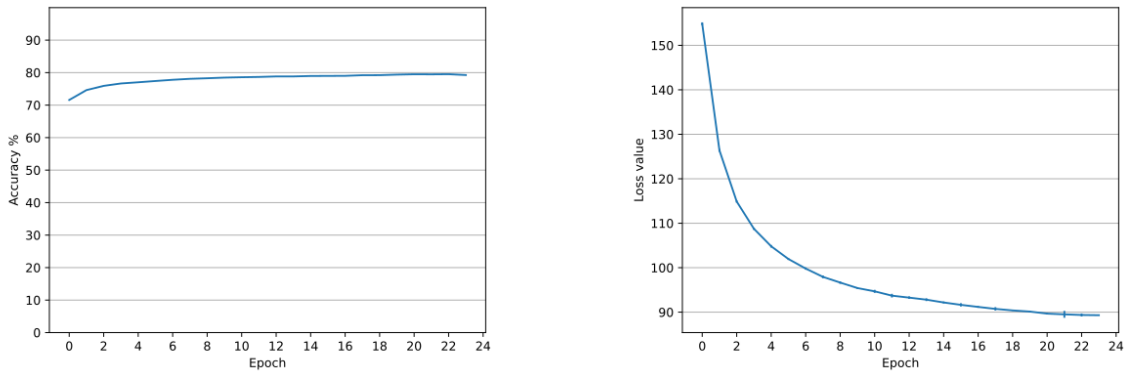
(b) Wykres funkcji straty

Rysunek 2: Wyniki dla sieci testowej zbudowanej na bazie AlexNet

Wyniki przeprowadzonych testów zostały przedstawione na rysunku 2. Dzięki wykorzystaniu od początku już częściowo wytrenowanej sieci już już po pierwszej epoce uzyskała ona skuteczność ponad 60%, jednak w kolejnych epokach wzrost skuteczności był stosunkowo wolny, ostatecznie dochodząc do około 73%.

3.2 ResNet

ResNet jest siecią zaprojektowaną przez Microsoft, która w 2015 roku wygrała ILSVRC uzyskując błąd na poziomie 3.6%. Jest skonstruowana z tzw. **bloków resydualnych**, w których to wejście do bloku jest przekształcane przez kilka warstw konwolucyjnych, a następnie dodawany do pierwotnego wejścia, co w założeniu ma poprawiać postać macierzy wejściowej zamiast tworzyć nową, która może wcale nie być związana z wejściową [3].



(a) Wykres skuteczności

(b) Wykres funkcji straty

Rysunek 3: Wyniki dla sieci testowej zbudowanej na bazie ResNet

Testy zostały przeprowadzone na wariancie sieci ResNet z 18 blokami resydualnymi. Jak widać na rysunku 3, sieć bazująca na ResNet uzyskała około 80% skuteczności. Już po 1 epoce sieć klasyfikowała poprawnie ponad 70% zbioru walidacyjnego.

Literatura

[1] Adit Deshpande, *A Beginner's Guide To Understanding Convolutional Neural Networks Part 1*, https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/

[2] Adit Deshpande, *A Beginner's Guide To Understanding Convolutional Neural Networks Part 2*, https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/

[3] Adit Deshpande, *The 9 Deep Learning Papers You Need To Know About (Understanding CNNs Part 3)*, https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html

[4] *CIFAR-10 - Object Recognition in Images*, https://www.kaggle.com/c/cifar-10

[5] *Training a Classifier*, https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

[6] *Transfer Learning*, https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html