Wydział Matematyki i Nauk Informacyjnych Politechniki Warszawskiej

Dokumentacja

Sieci konwolucyjne

Karol Ulanowski
Jakub Waszkiewicz

28 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

Sprawdzono wpływ następujących modyfikacji i usprawnień na działanie sieci [1, 2, 3]:

- **data agumentation** - sprawdzenie wpływu zastosowania różnych przekształceń obrazu w celu poszerzenia zbioru danych treningowych,
- **transfer learning** - 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**.

W dokumencie opisano architektury sieci, których konfiguracje wyżej wymienionych współczynników osiągnęły najlepsze skuteczności.

1.4 Opis danych

Do treningu i testowania zostały wykorzystane dane CIFAR-10 dostępne na platformie Kaggle [4], będące zbiorem obrazów, dla których należy dokonać klasyfikacji. Zbiór zawiera 60 tys. obrazów o rozmiarach $32px \times 32px$ 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

Zbiór treningowy będzie podzielony na zbiór treningowy oraz walidacyjny, na którym przeprowadzono obliczenia. Każda z sieci uczona była do momentu wzrostu błędu na zbiorze walidacyjnym.

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

1.6 Wykorzystane technologie

Wszelkie testy zostały wykonane przy użyciu technologii **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:

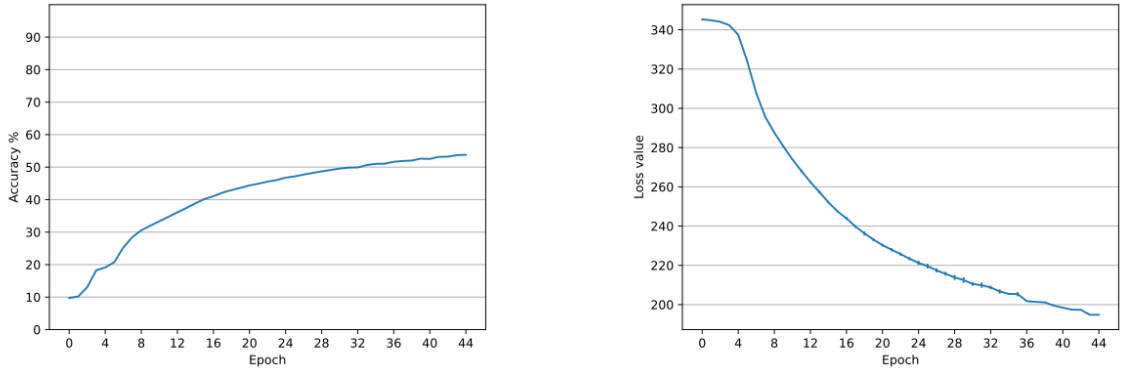
1. konwolucyjnej z 6 filtrami 5×5,
2. warstwa łącząca przy użyciu funkcji max i filtrów 2×2,
3. konwolucyjnej z 16 filtrami 5×5,
4. liniowej o 120 neuronach,
5. liniowej o 84 neuronach.

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 zaznaczono 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 [6]:

- AlexNet,
- VGG19,
- ResNet18

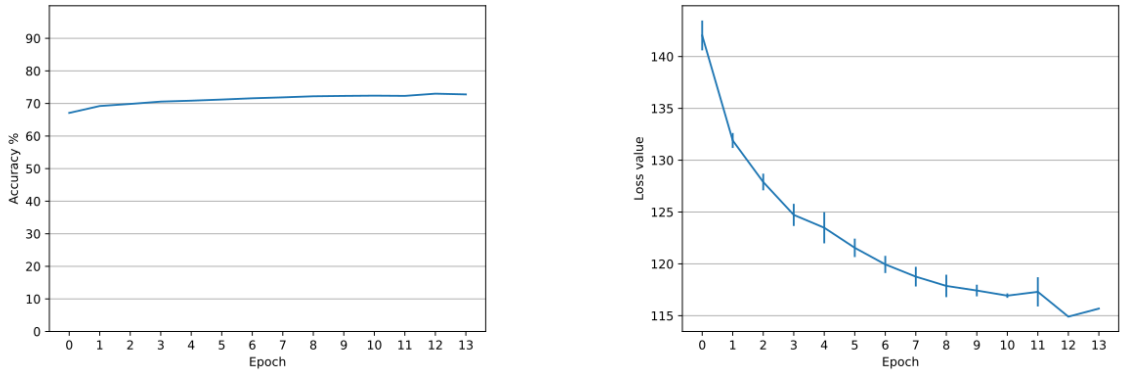
Dla każdej z tych sieci podmieniono ostatnią warstwę na warstwę liniową tak, aby sieć zwracała wyniki dla 10 klas. Dodatkowo zmieniono rozmiary obrazków wejściowych na $224px \times 224px$ tak, aby były zgodne z danymi wejściowymi przeznaczonymi dla sieci transferowanych. W testach trenowano jedynie 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

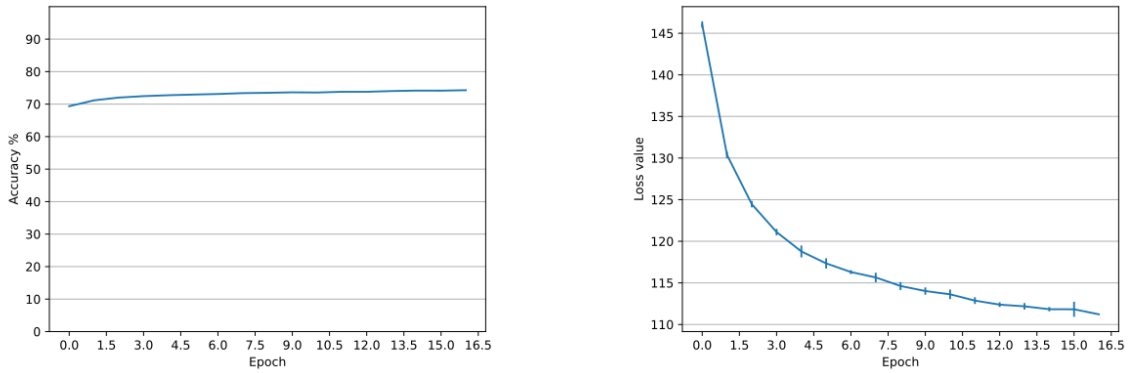
(b) Wykres funkcji straty

Rysunek 2: Wyniki dla sieci testowej zbudowanej na bazie AlexNet

Wyniki przeprowadzonych testów zostały przedstawione na rysunku 2. Dzięki wykorzystaniu od początku częściowo wytrenowanej sieci 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 VGG

Architektura VGG została stworzona w 2014 roku i w konkursie ILSVRC uzyskała błąd na poziomie 7.3%. Jest to 19 warstwowa sieć składająca się z warstw konwolucyjnych z filtrami 3×3 oraz warstw redukujących funkcją max z użyciem filtrów 2×2 [3].



(a) Wykres skuteczności

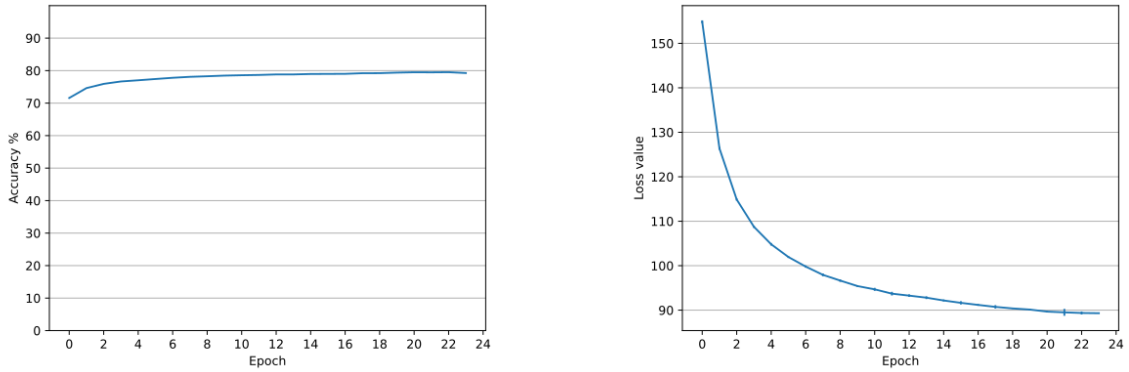
(b) Wykres funkcji straty

Rysunek 3: Wyniki dla sieci testowej zbudowanej na bazie VGG

Wyniki uczenia sieci zostały przedstawione na rysunku 3. Na zbiorze walidacyjnym osiągneła ona niecałe 75% skuteczności.

3.3 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 dodawane do pierwotnego wejścia, co w założeniu ma poprawiać postać danych zamiast tworzyć nowe, które mogą wcale nie być związane z wejściem [3].



(a) Wykres skuteczności

(b) Wykres funkcji straty

Rysunek 4: Wyniki dla sieci testowej zbudowanej na bazie ResNet

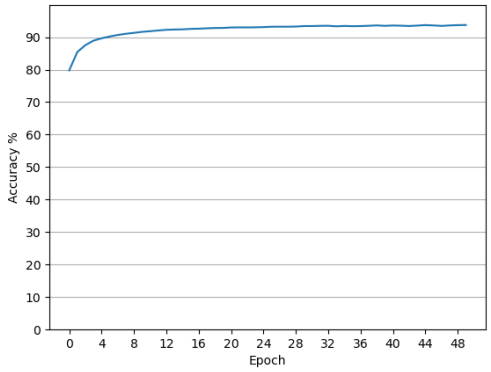
Testy zostały przeprowadzone na wariancie sieci ResNet z 18 warstwami. Jak widać na rysunku 4, sieć bazująca na ResNet uzyskała około 80% skuteczności. Już po 1 epoce sieć klasyfikowała poprawnie ponad 70% zbioru walidacyjnego.

3.4 ResNet - odblokowanie zmian wag

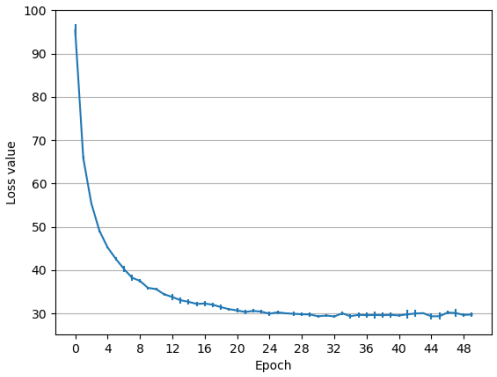
Ze wszystkich przetestowanych sieci najlepsze wyniki uzyskała sieć ResNet, dlatego też dalsze eksperymenty będą przeprowadzane na sieciach o zbliżonej architekturze.

Do tej pory trenowany był wariant tej sieci w którym jedynie wagi na ostatniej warstwie mogły zmieniać swoje wartości. Mogło to być powodem, dla którego wykres skuteczności jest bardzo "wypłaszczony" – zmiana wag na tylko jednej warstwie nie ma aż tak wyraźnego wpływu na skuteczność sieci, która nie miała możliwości dobrego dostosowania się do zbioru.

Z powyższych powodów została także przeprowadzona seria 5 testów na przetrenowanej sieci ResNet (pozostałe sieci były na tyle gorsze, że zostały wykluczone z dalszych rozważań), jednak z odblokowaną możliwością zmiany wag na wszystkich warstwach. Wyniki zostały przedstawione na rysunku 5.



(a) Wykres skuteczności

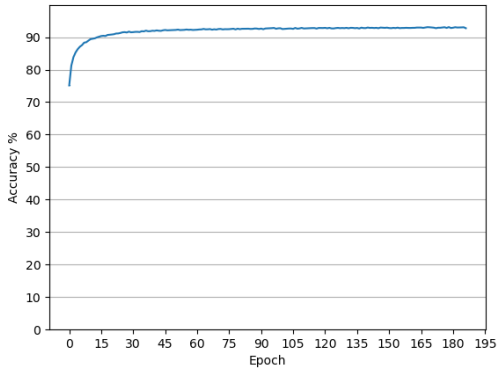


(b) Wykres funkcji straty

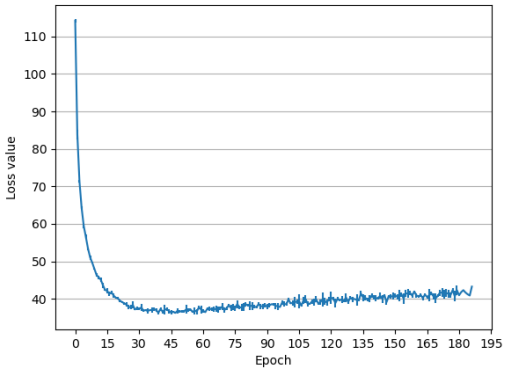
Rysunek 5: Wyniki dla sieci testowej zbudowanej na bazie ResNet z odblokowaną zmianą wag

Wyniki testów są bardzo zadowalające. Sieć uzyskała ponad 93% skuteczności na zbiorze walidacyjnym (oraz 93.37% na zbiorze testowym kaggle). Na początku procesu uczenia widzimy także wyraźny wzrost skuteczności w porównaniu do testu przeprowadzonego na tej samej sieci z zablokowaną zmianą wag, co było spodziewanym wynikiem.

Można się też zastanowić, jakie będą wyniki, gdy tylko częściowo zostanie zablokowana zmiana wag – taka zmiana może dać także dobre wyniki przy jednoczesnym szybszym procesie uczenia (nie w sensie ilości epok). Taki eksperyment został przeprowadzony na sieci ResNet18 z odblokowaną zmianą wag na ostatnich 2 warstwach z bloków rezydualnych (i ostatniej warstwie pełnej oczywiście, gdyż ona musiała być dostosowana do liczby wyjściowych klas). Wyniki zaprezentowano na rysunku 7.



(a) Wykres skuteczności



(b) Wykres funkcji straty

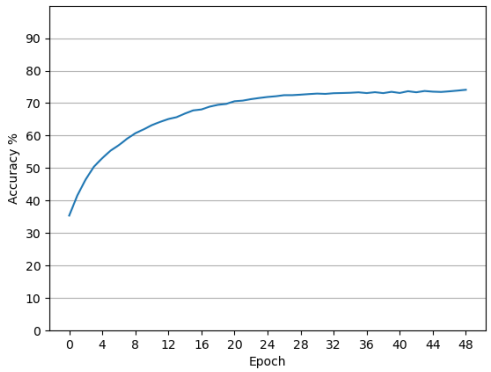
Rysunek 6: Wyniki dla sieci testowej zbudowanej na bazie ResNet z odblokowaną zmianą wag na ostanich dwóch warstwach z bloków rezydualnych

Otrzymane wyniki są niemal identyczne w porównaniu do całkowicie "odblokowanej" sieci – obie już po około 30 epokach uzyskują ponad 90% skuteczności.

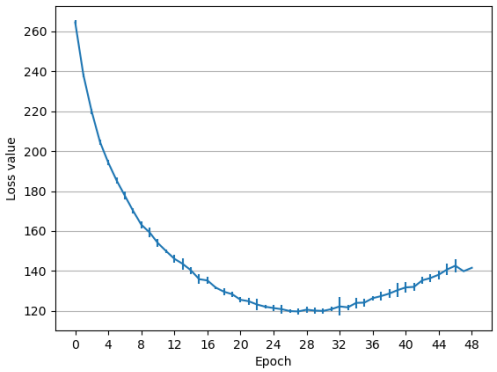
Dla częściowo zablokowanej sieci przeprowadzono jednak dużo dłuższy test. Było to spowodowane zmianą warunku zakończenia treningu, który zatrzymywał się w przypadku chwilowego wzrostu błędu walidacyjne-

go. Warunek ten jest nieco restrykcyjny i mógłby być wrażliwy na chwilowe błędy sieci, której ostateczna skuteczność mogłaby być wyższa. Dlatego też od tego momentu trening kończył się w przypadku wzrostu błędu o określoną procentową wartość w stosunku do minimalnej. Patrząc jednak na wykresy w pełni odblokowanej sieci ResNet stary warunek okazał się być wystarczający, gdyż sieć zdążyła osiągnąć wypłaszczenie oznaczające stabilizację wyniku skuteczności.

W celu sprawdzenia wpływu wstępnie wytrenowanej sieci na ostateczny wynik, spróbowano wytrenować sieć ResNet18 od początku.



(a) Wykres skuteczności



(b) Wykres funkcji straty

Rysunek 7: Wyniki dla sieci ResNet18 uczonej od początku bez wstępnie wytrenowanych wag

Wyniki jakie udało się uzyskać były o około 20% gorsze niż w przypadku sieci wstępnie wytrenowanych.

4 Głębokość sieci

Bazując na konstrukcji sieci ResNet18 zostały rozpatrzone różne jej warianty ze względu na głębokość sieci, tzn. liczbę warstw składających się z bloków rezydualnych. Docelowo ResNet18 składa się z [7]:

1. warstwy konwolucyjnej z 64 filtrami 7×7,
2. warstwy składającej się z:
   (a) warstwy redukującej z funkcją max i filtrem 3×3,
   (b) 2 równoległych bloków rezydualnych składających się z 128 filtrów 3×3 każdy,
3. warstwy składającej się z 2 równoległych bloków rezydualnych z 256 filtrami 3×3 każdy,
4. warstwy składającej się z 2 równoległych bloków rezydualnych z 512 filtrami 3×3 każdy,
5. warstwy redukującej z funkcją avg do rozmiaru 1×1.

Warto zaznaczyć, że ze względu na konstrukcję zupełnie nowej sieci bazującej jedynie na sieci ResNet proces uczenia wymaga dużo więcej epok, ponieważ nie mamy tu na wejściu obliczonych już wag.

Parametry treningowe były identyczne dla wszystkich sieci i wyglądały następująco:

- Zbiór został podzielony na treningowy oraz walidacyjny w stosunku $7 : 3$,

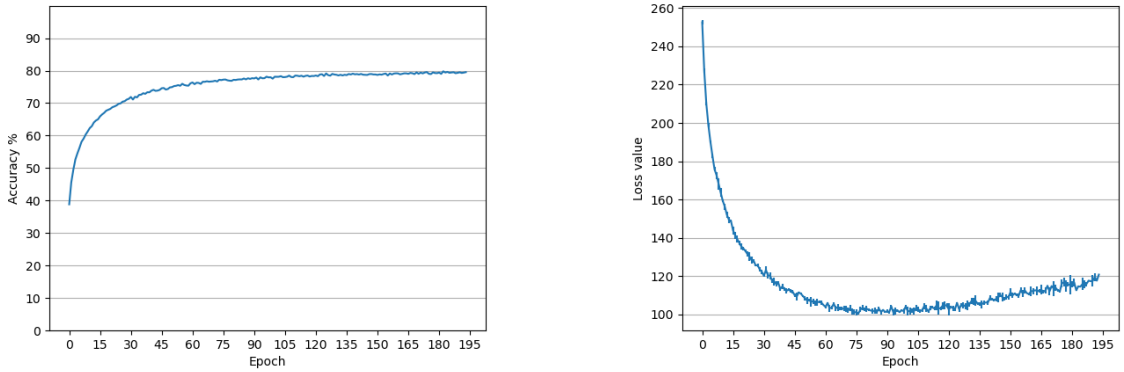
- Rozmiar batcha został ustawiony na 100,
- Współczynnik nauki miał wartość 0.0005,
- Współczynnik bezwładności miał wartość 0.9,
- Jako funkcja straty została wykorzystana CrossEntropy.

Dla każdej sieci wykonano 3 identyczne testy z wykorzystaniem różnych ziaren losowych.

4.1 Sieć płytka

Jako pierwsza sieć została przetestowana taka, która zawierała jedynie 2 warstwy z bloków rezydualnych:

1. warstwy redukującej z funkcją max i filtrem 3×3 oraz składającej się z 2 bloków z 256 filtrami 3×3,
2. składającej się z 2 bloków z 512 filtrami 3×3.



(a) Wykres skuteczności

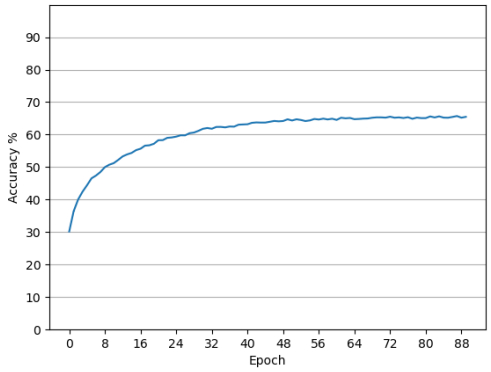
(b) Wykres funkcji straty

Rysunek 8: Wyniki dla płytkiej sieci testowej zbudowanej na bazie ResNet

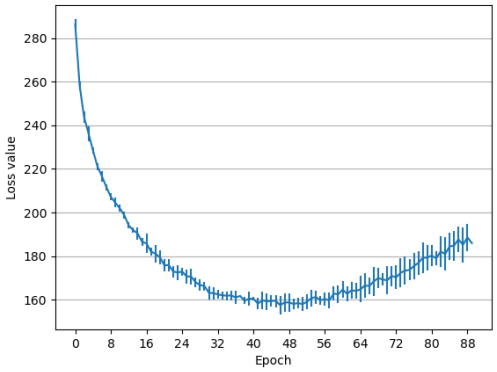
Wyniki zostały przedstawione na rysunku 8. Zgodnie z oczekiwaniami proces uczenia był dłuższy, a sieć uzyskała maksymalnie 80% skuteczności na zbiorze walidacyjnym. Minimalna wartość funkcji straty wyniosła ponad 100, gdzie wstępnie przetrenowana sieć ResNet18 uzyskała tę wartość na poziomie około 30.

4.2 Sieć głęboka

W podobny sposób została skonstruowana została głębsza sieć bazująca na ResNet, która składała się z aż 6 warstw z blokami rezydualnymi (po 2 bloki w każdej) odpowiednio z 64, 128, 256. 512. 1024, 2048 filtrami. Wyniki testów zostały zaprezentowanie na rysunku 9.



(a) Wykres skuteczności



(b) Wykres funkcji straty

Rysunek 9: Wyniki dla głębokiej sieci testowej zbudowanej na bazie ResNet

Jak widać, jej wyniki są wyraźnie gorsze od płytkiej sieci – skuteczność na zbiorze walidacyjnym spadła o ponad 10%, a po osiągnięciu minimum funkcja straty zaczęła szybko rosnąć. Taki wynik może być spowodowany zbyt skomplikowaną konstrukcją sieci, dla której zbiór mniej niż 50 tys. obserwacji był za mały, aby lepiej skorygować wartości takiej liczby wag.

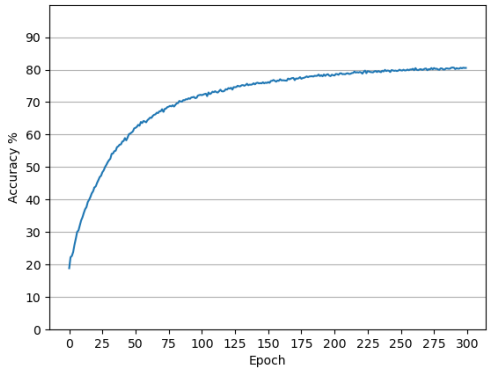
Ostatecznie to sieć płytsza uzyskała lepszą skuteczność. Jest ona jednak niższa od wyników sieci transferowanej.

5 Modyfikacje transferowanej sieci ResNet

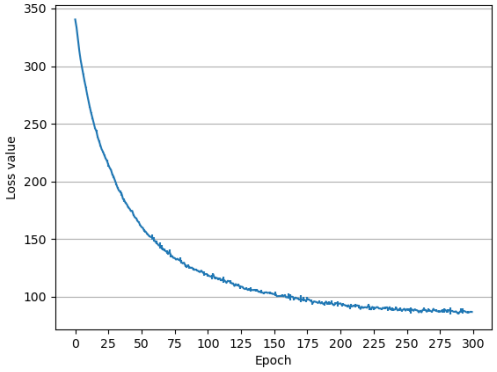
Z poprzednich testów wyszło, że lepsze wyniki uzyskała sieć bazująca na ResNet o mniejszej liczbie warstw rezydualnych. Jej skuteczność jest jednak wciąż niższa niż sieci przetransferowanej. W związku z tym przeprowadzono eksperyment na przetransferowanej sieci ResNet, jednak z usuniętymi ostatnimi 2 warstwami rezydualnymi. Jej wyniki przedstawiono na rysunku 10. Do treningu wykorzystano następującą konfigurację:

- 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 2 identyczne testy z użyciem różnych ziaren losowych



(a) Wykres skuteczności

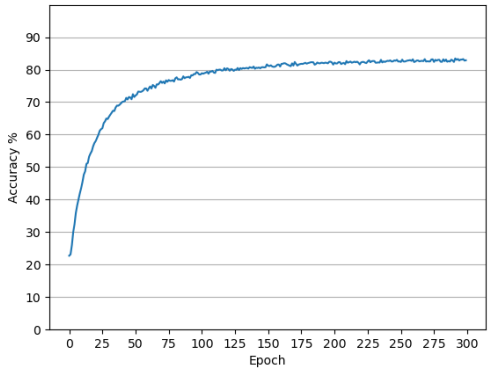


(b) Wykres funkcji straty

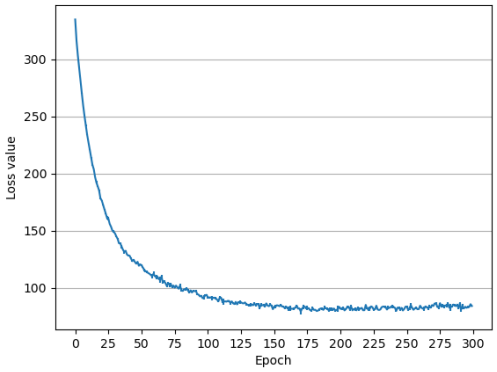
Rysunek 10: Wyniki dla transferowanej sieci ResNet z usuniętymi 2 ostatnimi warstwami rezydualnymi

Sieć uzyskała nieco ponad 80% skuteczności. Uczyła się przez stosunkowo długą liczbę epok, choć i tak trening zakończył się przez narzuconą górną granicę tej wartości (na 300).

W związku z tym, aby przyspieszyć proces uczenia, zwiększono dwukrotnie współczynnik nauki (czyli do wartości 0.001). Wyniki analogicznego testu przedstawiono na rysunku 11.



(a) Wykres skuteczności



(b) Wykres funkcji straty

Rysunek 11: Wyniki dla transferowanej sieci ResNet z usuniętymi 2 ostatnimi warstwami rezydualnymi ze zwiększonym współczynnikiem nauki

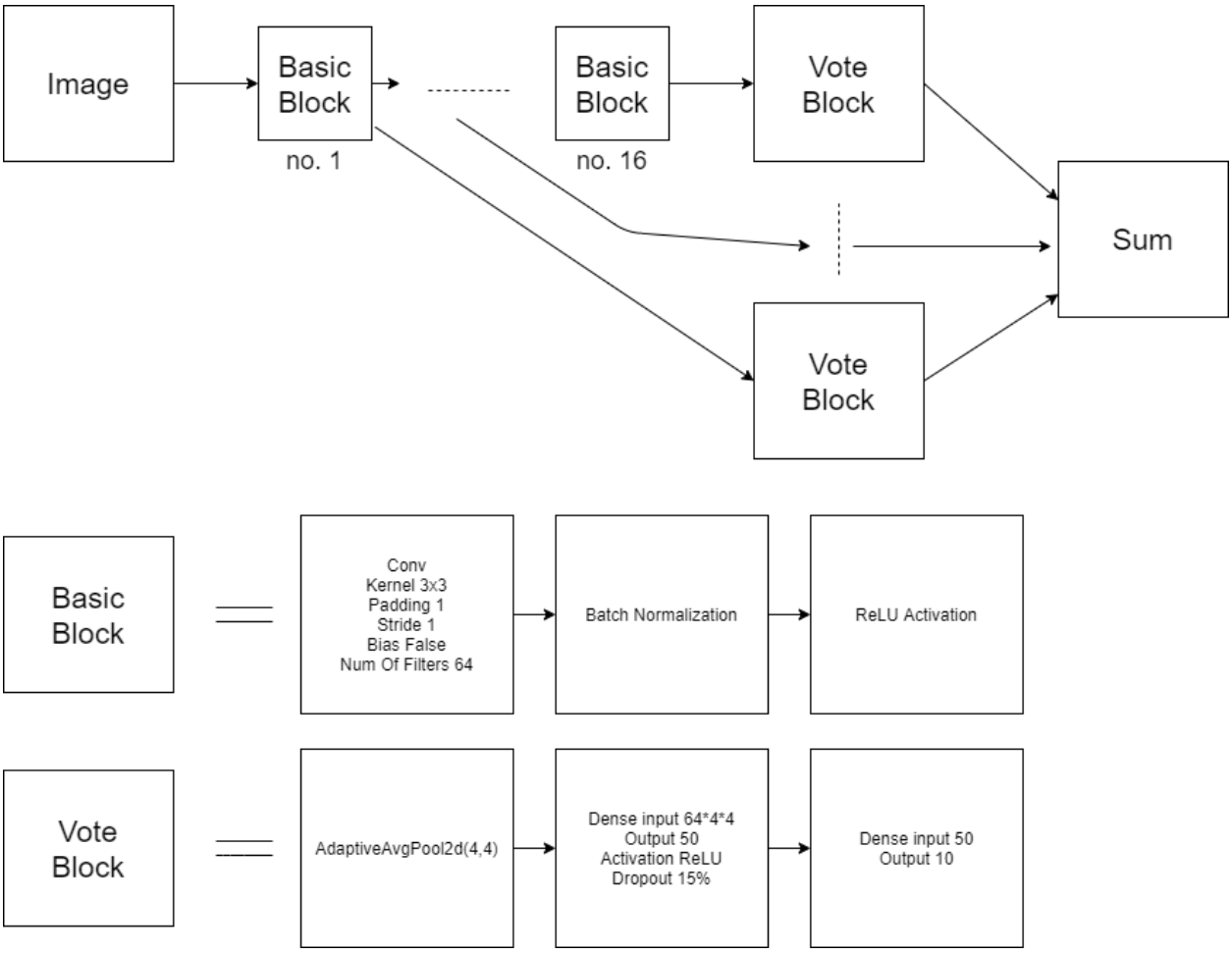
Otrzymana skuteczność jest minimalnie wyższa (82%), jednak nie jest to bardzo znacząca różnica. Zgodnie z oczekiwaniami zwiększony współczynnik nauki pozwolił szybciej uzyskać podobną skuteczność, jednak w dalszych epokach widzimy "wypłaszczenie" na wykresach zarówno skuteczności, jak i funkcji straty.

6 Własny pomysł

Korzystając z przeprowadzonych eksperymentów postanowiono zaprojektować własną architekturę sieci. Inspiracją było:

- struktura blokowa ResNet,
- przekazywanie informacji w sieciach densenet,
- idea ensamble learningu.

Zaprojektowano następującą sieć przedstawioną na rysunku 12.



Rysunek 12: Własna architektura sieci

Sieć składa się z 16 Basic Blocków. Idea sieci polega na tym, że każdy z Basic Bloków głosuje poprzez Vote Block na jakąś klasę (zwraca wektor długości 10). Wynik z Basic Blocku przekazywany jest też do następnych bloków. Na koniec, aby uzyskać ostateczny wynik wyniki (wektory) z Vote Blocków są sumowane.

Założeniem jest, aby bloki wcześniejsze oceniały do jakiej klasy należy zakwalifikować obraz na podstawie ogólnych cech. Natomiast późniejsze bloki oceniają obraz na podstawie cech bardziej skomplikowanych. Te wszystkie wyniki są następnie uwzględniane.

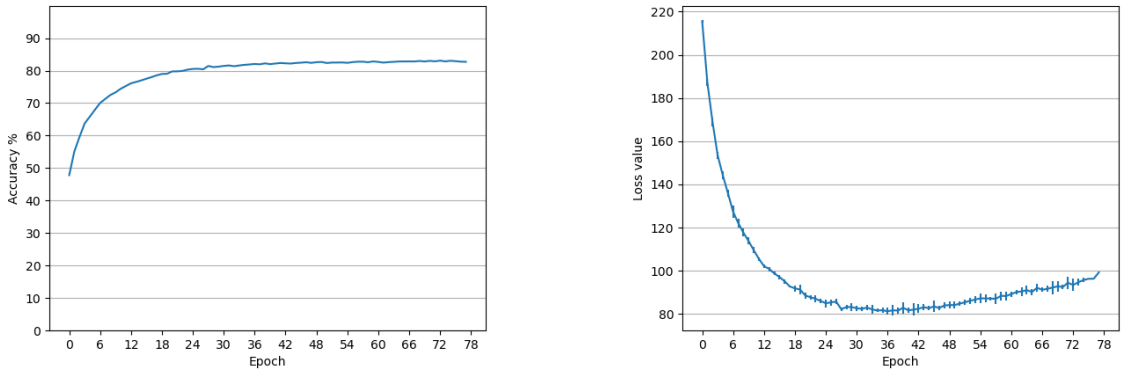
Sieć była trenowana z następującymi parametrami:

- 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.

Jako data augumentation użyto losowego okręcania zdjęć względem osi pionowej.

Tak wytrenowany model uzyskał dokładność 81.06% na zbiorze kaggle.



(a) Wykres skuteczności

(b) Wykres funkcji straty

Rysunek 13: Wyniki dla własnej sieci

7 Podsumowanie

Najlepszy wynik jaki udało się osiągnąć na platformie kaggle to 93,37%. Taki wynik uzyskała sieć ResNet18 z przetransferowanymi wagami i następnym "dostrajaniu" ich w procesie uczenia.

Na bazie sieci ResNet testowano także sieci o podobnej architekturze. Niektóre z nich także korzystały z przetransferowanych wartości wag z części warstw. Nie uzyskały one jednak skuteczności wyższej niż 83%.

Zaproponowano też własną architekturę, która uzyskała wynik 81.06% na platformie kaggle. Co wydaje się bardzo dobrym wynikiem, ponieważ siec ResNet bez przetransferowanych wag uzyskiwała skuteczność na poziomie 73%.

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

[7] *ResNet*, https://pytorch.org/assets/images/resnet.png