PROJEKT ZALICZENIOWY

SIECI NEURONOWE I DEEP LEARNING

Dominika Schabowska
Mateusz Pompa
Grzegorz Grodecki

15.06.2024

1 WYBÓR PROBLEMU I DANYCH

W ramach projektu zdecydowaliśmy się zbudować model, który będzie klasyfikował zdjęcia w celu rozpoznawania obiektów, nawet dla bardziej złożonych danych. Przy pomocy sieci neuronowych trenujemy model, który może służyć jako filtr w aplikacjach lub może być wykorzystany do maszyn sortujących, przykładowo owoców i warzyw lub paczek. Problem identyfikacji różnych, mniej lub bardziej skomplikowanych obiektów ma zastosowanie w wielu branżach.

W tym celu korzystamy ze zbioru danych CIFAR-10. Jest to podzbiór 80-milionowego zbioru danych Tiny Images, zebranych przez Alex Krizhevsky, Vinod Nair i Geoffrey Hinton, w ramach ich badań nad rozpoznawaniem obrazów i głębokim uczeniem.

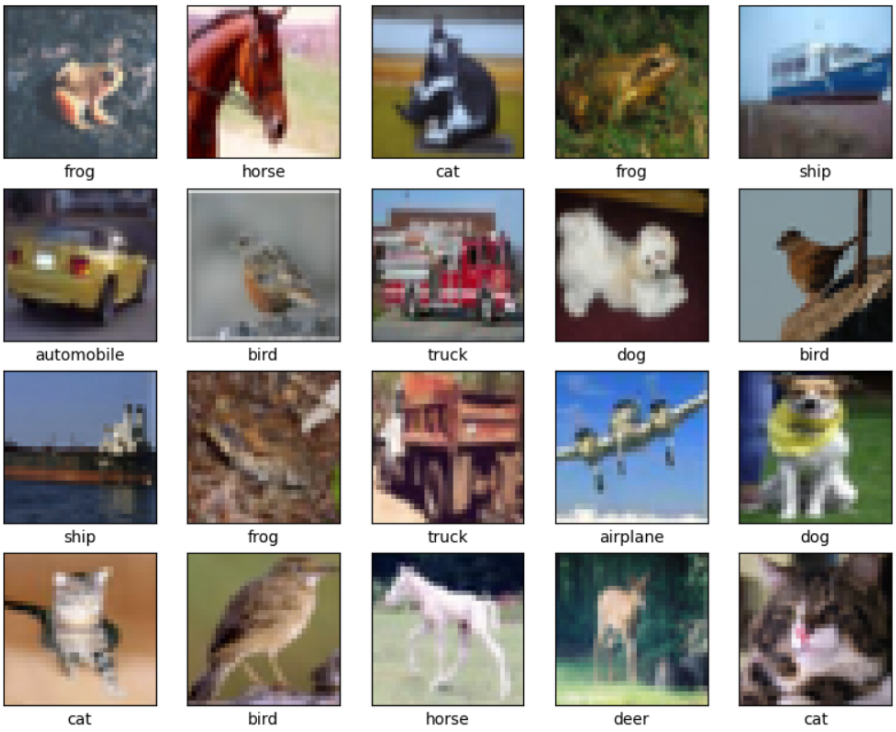
Zbiór danych wykorzystywany jest edukacyjnie, więc dostępny jest m.in. na platformie Kaggle oraz wbudowany jest w bibliotekach takich jak PyTorch czy TensorFlow.

CIFAR-10 zawiera 60.000 obrazów w 10 klasach, z czego dla każdej klasy dobranych jest po 6.000 obrazów. Obrazy oznaczone są jedną z wykluczających się klas (samoloty, samochody, ptaki, koty, jelenie, psy, żaby, konie, statki, ciężarówki). Każdy obraz ma rozdzielczość 32x32 pikseli i jest kolorowy (RGB).

Oryginalnie zbiór jest podzielony na 50.000 obrazów treningowych i 10.000 obrazów testowych, co zostanie zmienione w trackie analizy, aby dołożyć zbiór walidacyjny.

2 WSTĘPNA ANALIZA DANYCH

Analizę danych rozpoczynamy od załadowania ich z biblioteki TensorFlow. Zbiór danych nie ma żadnych braków, a obrazki w klasach rozłożone są po równo. Problematyczne dla trenowania danych może być ich jakoś, ponieważ są to bardzo małe ilustracje 32x32 pikseli. Poniżej przedstawione są przykładowe dane, wygenerowane w trakcie analizy.



Rysunek 1: Przykładowe ilustracje ze zbiory danych CIFAR-10.

Przygotowanie danych zaczynamy od normalizacji pikseli do zakresu $[0,\ 1]$, zdefiniowania klas:

$$airplane,\ automobile,\ bird,\ cat,\ deer,\ dog,\ frog,\ horse,\ ship,\ truck$$

oraz nowego podziału na zbiór treningowy, walidacyjny i testowy.

```
train_images.shape,valid_images.shape,test_images.shape,
```

```
((39000, 32, 32, 3), (9000, 32, 32, 3), (12000, 32, 32, 3))
```

Rysunek 2: Fragment kodu przedstawiający ilość ilustracji, rozmiar obrazka, liczba kanałów w każdym obrazie w zbiorze odpowiednio train, valid, test.

Obrazki są kolorowe więc zostają podzielone ze względu na trzy kanały kolorów RGB (Red, Green, Blue).

3 BUDOWA MODELU

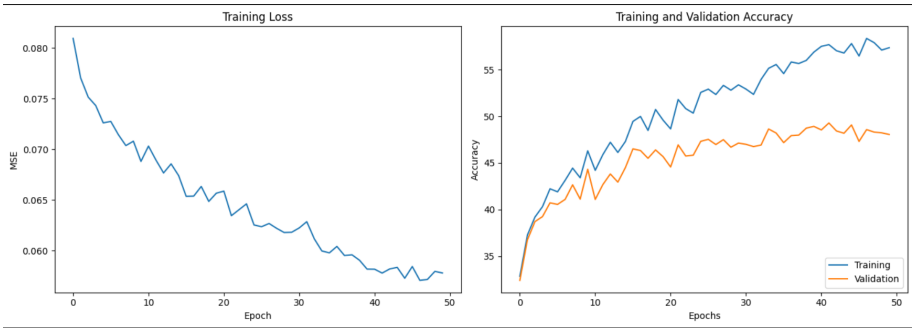
Docelowo zdecydowaliśmy się na zbudowanie konwolucyjnych sieci neuronowych (CNN), których celem jest odczytanie obiektu na zdjęciu i przypisanie go do odpowiedniej klasy. Jednak w ramach treningu oraz porównania skuteczności różnych modeli poznanych na zajęciach, w projekcie zostały również zastosowane wielowarstwowe sieci neuronowe typu MLP oraz sieć MLP w module *torch.nn*.

3.1 IMPLEMENTACJA WIELOWARSTWOWEGO PERCEPTRONU MPL

Budowę modelu zaczynamy od zaimplementowania go z jedną warstwą ukrytą, używamy funkcji sigmoidalnej jako funkcji aktywacji oraz funkcji straty błąd średniokwadratowy. Dane, jako wielowymiarowe tablice obrazów, zostają przekształcone w jednowymiarowe wektory:

Początkowy kształt: (39000, 32, 32, 3) dla train_images, gdzie 39000 to liczba obrazów.

Po przekształceniu: (39000, 32*32*3), czyli (39000, 3072).
Tworzymy obiekt klasy NeuralNetMLP ze 100 neuronami w warstwie ukrytej, a w procesie uczenia modelu wykorzystujemy funkcję *minibatch_generator*(). Kolejno implementujemy funkcję straty oraz funkcję błędu klasyfikacji i przechodzimy do trenowania modelu przy 50-ciu epokach i learning_rate = 0.5.



Rysunek 3: Wykresy funkcji straty oraz dokładność predykcji dla zbioru treningowego i walidacyjnego.

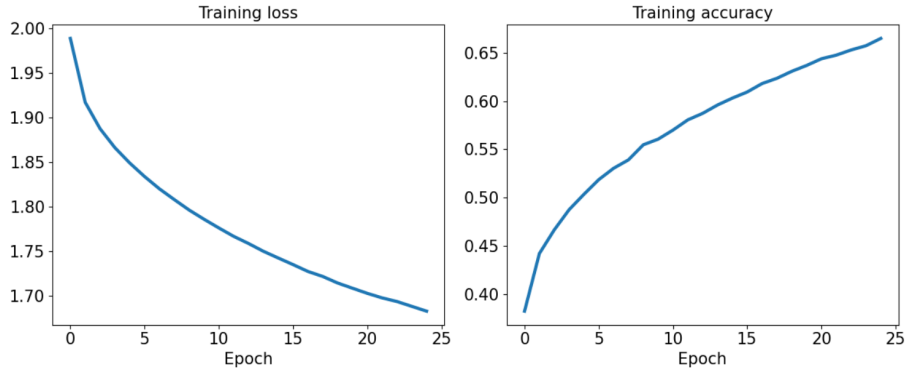
Błąd średniokwadratowy dla zbioru treningowego maleje przekraczając wartość 0.06 z kolejnymi epokami, dokładność dla tego zbioru rośnie do 58%. Dokładność dla zbioru walidacyjnego również rośnie, jednak zaczyna się utrzymywać przy 48%. Na powyższym wykresie możemy zauważyć, że wykresy dla obydwu zbiorów raczej zaczynają się oddalać przy kolejnych

epokach. Ta obserwacja oraz niewielka dokładność predykcji modelu skłania do polepszenia jego jakości oraz rozważenia innych algorytmów.

3.2 SIEĆ MPL W MODULE TORCH.NN

Analizując tę metodę decydujemy się na zaimplementowanie zbioru danych na nowo i podzielenie go na zbiór treningowy oraz testowy w proporcjach 5:1. Odpowiednio przygotowujemy dane poprzez standaryzacje pikseli i przekształcenie obrazów w jednowymiarowe wektory, jak poprzednio.

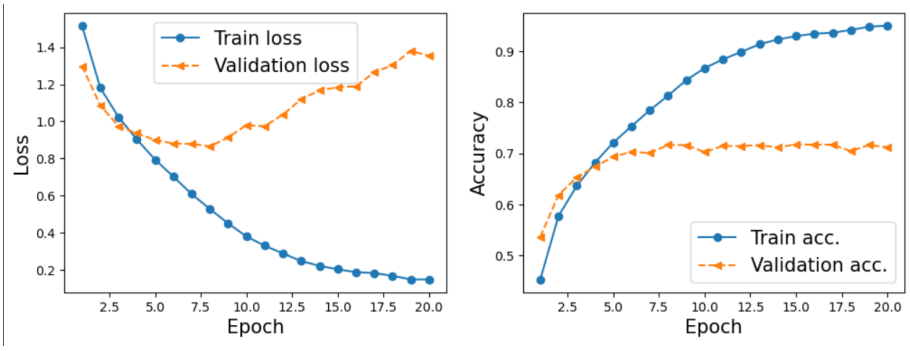
Skonstruujemy sieć z dwiema warstwami oraz metodą forward. Pierwsza warstwa przekształca cechy przez 1024 neuronów, druga warstwa na wejście otrzymuje wyjście z pierwszej warstwy i przekształca je przez 10 neuronów (ponieważ mamy 10 klas). Przy learning_rate = 0.1 oraz batch_size = 50 otrzymujemy najlepszy model, który dokładność osiąga do 66.50%, a funkcja straty maleje do wartości: 1.6825.



Rysunek 4: Wykresy funkcji straty oraz dokładność predykcji dla zbioru treningowego.

3.3 KONWOLUCYJNE SIECI NEURONOWE

W kolejnej części podejdziemy do problemu z użyciem sieci CNN. Zaczniemy od przekonwertowania danych na parę tensor+etykieta, z której będzie korzystał nasz model. Na nowo dokonujemy przemieszania danych, żeby samodzielnie stworzyć zbiory treningowy, testowy i walidacyjny. Konstruujemy sieć CNN w PyTorch, dobierając warstwy konwolucyjne oraz pooling. Następnie spłaszczamy wyjście z poprzednich warstw oraz dodajemy dwie warstwy typu *fully connected* z warstwą *dropout* pomiędzy. Trenujemy nasz model przy 20-stu epokach, wyniki prezentują się poniżej na wykresach.



Rysunek 5: Wykresy funkcji straty oraz dokładność predykcji dla zbioru treningowego (model 1 dla CNN).

Wykresy wskazują na przeuczenie modelu. Przy piątej epoce wartości dla zbioru treningowego i walidacyjnego zaczynają się rozbiegać, a od dziesiątej epoki zaczynamy obserwować wzrost funckji straty dla zbioru walidacyjnego.

Wyniki, które otrzymaliśmy, są jednak najlepsze dla naszego problemu spośród przedstawionych do tej pory metod. W kolejnej sekcji omówimy dokładniej dobór parametrów oraz sposób poprawy wydajności sieci CNN.

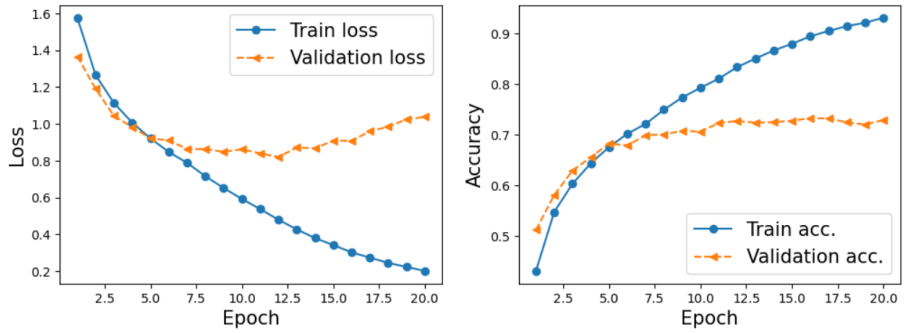
4 EWALUACJA ORAZ OPTYMALIZACJA MODELU

Na powyższych rysunkach zaobserwowaliśmy, że po kilku epokach błąd na zbiorze walidacyjnym wzrasta co może sugerować przeuczenie modelu. Wynik modelu na zbiorze testowym to 71.5% procent skuteczności, co jest dość dobrym wynikiem w porównaniu z wcześniej używanymi metodami. Aby uzyskać lepszy model, postanowiliśmy dokonać kilku zmian w naszej architekturze sieci. Każda z nich była dodawana niezależnie i samodzielnie do naszej podstawowej sieci, aby łatwiej było porównać wyniki i zobaczyć pozytywny lub negatywny wpływ danej zmiany naszego model. Utworzyliśmy trzy modele z następującymi zmianami:

- Model 2 - dodanie kolejnej warstwy konwolucyjnej, kolejnej warstwy poolingowej oraz kolejnej warstwy ReLU do naszego modelu. W tym kroku chcieliśmy sprawdzić w jaki sposób rozbudowanie sieci wpłynie na wyniki.
- Model 3 - zmiana wartości learning rate na połowę mniejszą, w celu sprawdzenia czy efekt zmniejszania dokładności na zbiorze walidacyjnym po kilku epokach można zniwelować.

- Model 4 - zastąpienie funkcji ReLU, funkcjami ELU w celu sprawdzenia poziomu działania innych funkcji.

Niewielka poprawa wyników nastąpiła w przypadku modelu 2 i modelu 3 (predykcja na zb. testowym 73% i 72.5% odpowiednio). W modelu 4 zaobserwowaliśmy pogorszenie wyniku w stosunku do modelu wyjściowego (predykcja na zb. testowym 69.5%), co sugeruje, że używanie funkcji ReLU jest optymalniejsze. W modelu trzy można zaobserwować poprawę efektu związanego z przeuczeniem:



Rysunek 6: Wykresy funkcji straty oraz dokładność predykcji dla zbioru treningowego i walidacyjnego (model 3 dla CNN).

W związku z tym jako nasz finalny model wybraliśmy model z dodatkowymi warstwami i jeszcze niższym learning rate równym 0.0003 (aby jeszcze lepiej zwalczyć efekt przeuczenia) i dzięki temu otrzymaliśmy najlepszy wynik.

4.1 WYNIK KOŃCOWY

Nasz końcowy model dał nam dokładność predykcji na przygotowanym wcześniej zbiorze testowym na poziomie 74,38 procent oraz pozwolił w dużej mierze wyeliminować efekt przeuczenia na co pozwolił niższy poziom learning rate oraz dodatkowe warstwy. Mimo iż uzyskaliśmy wynik na całkiem dobrym poziomie, wciąż pozostała przestrzeń na poprawy. Jednym z głównych problemów powstrzymujących nas przed uzyskaniem lepszych wyników wydaje się być słaba jakość zdjęć w naszym zbiorze co jest dobrze widoczne na przykładach, które zostały źle zaklasyfikowane przez nasz finalny model.