Oskar Kacprzak s22010

Paweł Walesiak s12828

RAPORT – PROJEKT INL

Temat: Sieć neuronowa tworząca deminutywy

1. **Przygotowanie danych wejściowych**.

Dane wejściowe zostały pozyskane z http://plwordnet.pwr.wroc.pl/wordnet/ po rejestracji na stronie w celach naukowych można pobrać plik z danymi w postaci .xml.

Dane zostały pogrupowane za pomocą wyrażenia regularnego, które wyciąga relacje.

```
deminutives = re.findall(r"<lexicalrelations parent=\"([0-9]+)\"
child=\"([0-9]+)\" relation=\"56\"\n?.*valid=\"true\"
owner=\".*\"\/>", file, re.MULTILINE)
```

Regex wyciągający wszystkie słowa, które są rzeczownikami.

```
words = re.findall(r"<lexical-unit id=\"([0-9]+)\"\n?( {8})?
?name=\"(.*)\"\n?( {8})? ?pos=\"rzeczownik.*\"\n?( {8})?
?tagcount=\"[0-9]+\"\n?( {8})? ?domain=\".*\"\n?( {8})?
?desc=\".*\"\n?( {8})? ?workstate=\".*\"\n?( {8})?
?source=\".*\"\n?( {8})? ?variant=\"[0-9]+\"/>", file, re.MULTILINE)
```

2. **Architektura programu.**

Program oparty jest na bibliotece torch języka Python. Biblioteka dedykowana jest od maszynowego uczenia, stosowana głównie w aplikacjach mających na celu przetwarzanie języka naturalnego

Poprzez przygotowanie danych za pomocą wyrażenia regularnego, dodajemy do zbioru numery słów występujących w relacji. Następnie możemy numerową słowa występujące w relacjach poprzez dodanie klucza.

Następnym krokiem jest utworzenie słowników Znak – Numer oraz Numer – Znak.
Pozwala nam to na wpisanie wektoryzowanych słów do tablicy i utworzenie wektora wejściowego oraz wyjściowego.

3. **Model sieci neuronowej.**

Za pomocą modułu Numpy, który umożliwia zawansowane obliczenia matematyczne w szczególności operacje na macierzach. Charakterystyczne dla obiektów przechowywanych w macierzy array jest to, że wszystkie muszą być tego samego typu. Do obliczeń wykorzystujemy CUDA czyli równoległa architektura obliczeniowa, która zapewnia wzrost wydajności obliczeń dzięki wykorzystaniu mocy układów CPU.

Parametry wejściowe sieci neuronowej:

```
N, D_in, H, D_out = len(x_set), max_len, 512, max_len
```

Gdzie:

n- rozmiar setu wejsciowego

h-wielkość warstwy

d-in -rozmiar wektora wejściowego

d_out - rozmiar wektora wejściowego

Poniżej znajduje się implementacja modelu sieci neuronowej:

```
class NeuralNetworkModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.layer_input = nn.Linear(D_in,H)
        self.layer_hidden_one = nn.Tanh()
        self.layer_hidden_two = nn.Linear(H,H)
        self.layer_dropout = nn.Dropout(p=0.4)
        self.layer_output = nn.Linear(H,D_out)
    def forward(self, param):
        param = F.relu(self.layer_input(param))
        param = self.layer_dropout(param)
        param = F.relu(self.layer_hidden_one(param))
        param = F.relu(self.layer_hidden_two(param))
        param = F.relu(self.layer_dropout(param))
        param = F.relu(self.layer_output(param))
    return param
```

Do modelu sieci neuronowej użyto algorytmu optymalizacji „Adam”, który aktualizuje iteracje wag sieci neuronowej w oparciu o dane szkoleniowe.

Zbiór testowy oraz treningowy został podzelony wg. proporcji.

```
x_train = x[0:3000]
y_train = y[0:3000]
 x_test = x[3001:]
 y_test = y[3001:]
```

4. Prezentacja wyników.

Optymalizacja modelu sieci neurnowej została zrealizowana za pomocą doboru rozmiaru neuronów w warstwie, zmianą liczby epok oraz wartością dropoutu..

Optymalna konfiguracja sieci posiada parametry:

rozmiar neuronów w sieci `h = 512`

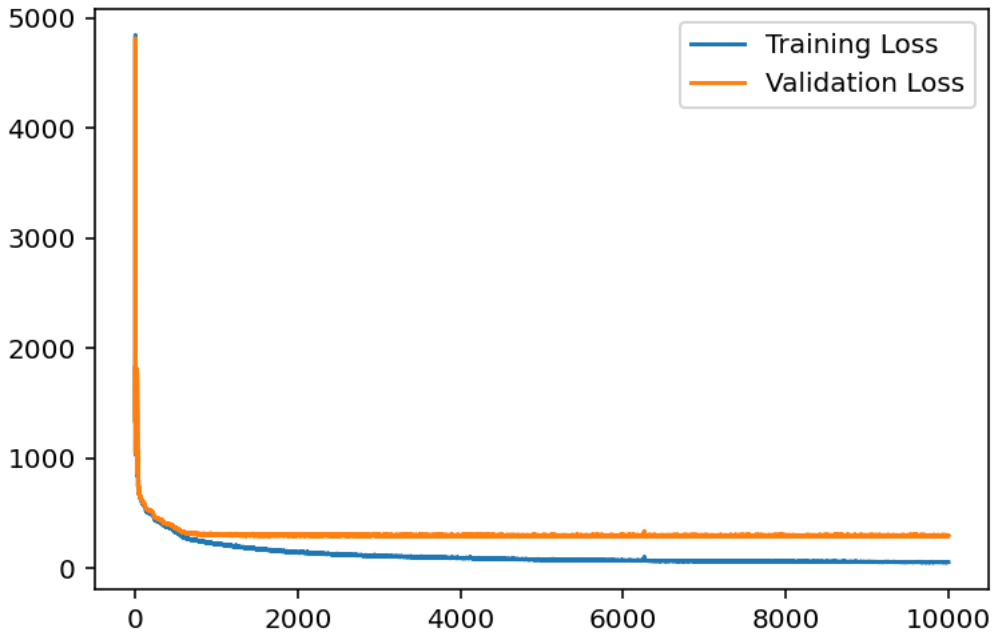
liczba `epochs = 10000`

dropout `p = 0.4`

Wyniki zostały zaprezentowane w postaci wykresu oraz inoframcji:

```
Epoch: 10000/10000..  Training Loss: 56.090..  Test Loss: 291.927..
Test Accuracy: 0.000
```

Wykres został opracowany za pomocą biblioteki `matplotlib.pyplot`



Rysunek 1.Wyres przedstawiający wynik działania sieci neurnowej

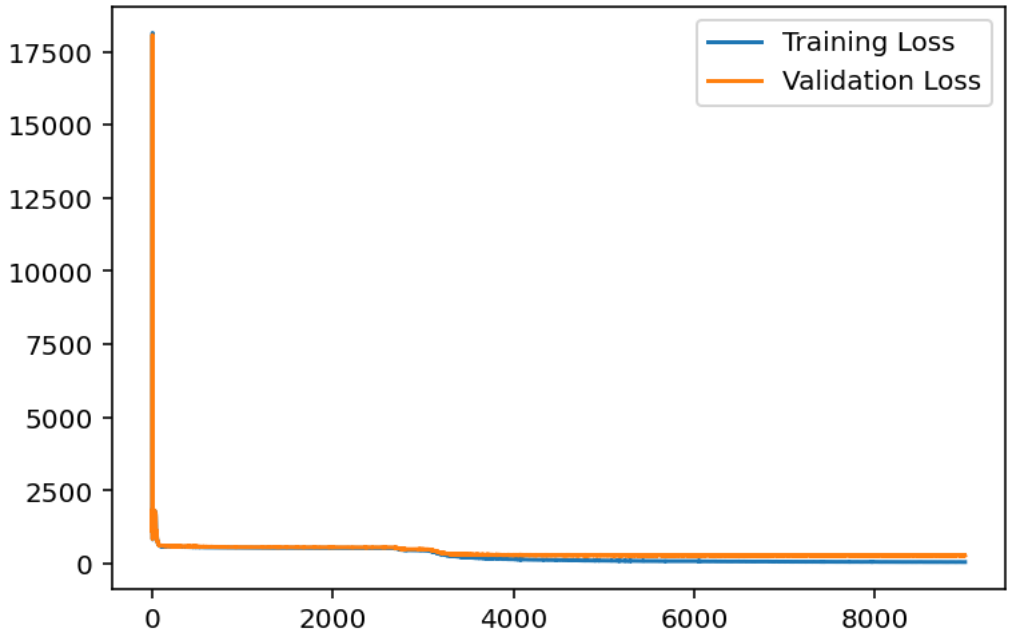
Przykład próby optymalizacji sieci:

rozmiar neuronów w sieci `h = 600`

liczba `epochs = 90000`

dropout `p = 0.4`

```
Epoch: 9000/9000..  Training Loss: 56.179..  Test Loss: 279.939..
Test Accuracy: 0.000
```



Rysunek 2. Wykres przedstawiający testy modelu sieci neuronowej

5. Wnioski

Dane wejściowe do sieci neuronowej zostały przygotowane poprawnie, również sama metodyka uczenia jest poprawna, natomiast można stwierdzić, że zaproponowany model nie jest poprawny. Budowa modelu odbywała się metodą prób i błędów bez wiedzy specjalistycznej, dlatego dokładność jest równa 0 lub bliska zeru. Aby osiągnąć zadowalający efekt należałoby posiąść głębszą wiedzę z zakresu budowy modelu sieci neuronowych oraz teorii działania. Autorzy rozwiązania nie posiadają głębokich podstaw teoretycznych jak i odpowiednio szerokiej wiedzy praktycznej.

6. Repozytorium kodu oraz wykorzystane źródła.

Środowisko do uruchomienia programu zostało utworzone za pośrednictwem Google Colab.

Bezpośredni link do projektu:
https://colab.research.google.com/drive/12mS89nF0btqcfJ1Il2Zvh3vhoKofjB_V?usp=sharing#scrollTo=62u-JEGkveqk

Repozytorium plików:

https://github.com/pawelwalesiak/INLproject.git

Źródła z których wykorzystujemy fragmenty kodu oraz czerpiemy inspiracje.
https://github.com/LanguegeEngineering

thirdpytorch.ipnb

Fashionmnist.ipnb