SPRAWOZDANIE

Zajęcia: Nauka o danych 2

Prowadzący: prof. dr hab. inż. Vasyl Martsenyuk

Laboratorium Nr 6 **Data** 19.11.2025 **Temat:** "Projektowanie zaawansowanych architektur sieci neuronowych w TensorFlow" **Wariant** 10	Jakub Janik Informatyka II stopień, stacjonarne, 2 semestr, gr. CB

1. Cel ćwiczenia:

Celem ćwiczenia było zaimplementowanie i trenowanie Dwukierunkowej Rekurencyjnej Sieci Neuronowej (BRNN) do zadania klasyfikacji sentymentu na zbiorze danych IMDB Reviews. Głównym zadaniem było ocenienie, czy recenzja filmowa jest pozytywna, czy negatywna, wykorzystując ograniczenie długości sekwencji do 100 tokenów

2. Przebieg ćwiczenia:

Zbiór Danych

Wykorzystano zbiór danych IMDB Reviews, zawierający 25 000 recenzji treningowych i 25 000 testowych, z dwiema klasami: pozytywną i negatywną.

Proces

- Ograniczenie słownika (MAX_FEATURES): Użyto tylko 10 000 najczęściej występujących słów.
- Ograniczenie długości sekwencji (MAX_LEN): Wszystkie recenzje zostały przycięte lub dopełnione do stałej długości 100 tokenów za pomocą operacji pad_sequences

Architektura modelu

W celu optymalizacji wydajności, w architekturze BRNN zastosowano bardziej zaawansowane jednostki LSTM (Long Short-Term Memory) zamiast klasycznych SimpleRNN, które są podatne na problem zanikającego gradientu.

Model zaimplementowano w Keras/TensorFlow w następujący sposób:

Warstwa	Dane wyjściowe	Liczba parametrów	Opis
Embedding	(None, 100, 128)	1,280,000	Osadzanie 10 000 słów w 128-wymiarową przestrzeń.
Bidirectional (LSTM)	(None, 128)	98,816	Dwie warstwy LSTM (po 64 jednostki każda) przetwarzające sekwencję w obu kierunkach.
Dense	(None, 1)	129	Warstwa wyjściowa z aktywacją sigmoid do klasyfikacji binarnej.
Suma parametrów trenowalnych	-	1 378 945	

Model został skompilowany z optymalizatorem Adam i funkcją straty binary_crossentropy

Kod źródłowy:

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.datasets import imdb
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Bidirectional, LSTM, Dense
from tensorflow.keras.preprocessing.sequence import pad_sequences
import logging

# 1. Konfiguracja TensorFlow
tf.get_logger().setLevel(logging.ERROR)
tf.autograph.set_verbosity(3)

# 2. Ustawienia i ładowanie danych
MAX_FEATURES = 10000
MAX_LEN = 100
EMBEDDING_DIM = 128
BATCH_SIZE = 32
EPOCHS = 5

print(f"Ładowanie zbioru danych IMDB (tylko {MAX_FEATURES} najczęściej wystę-
pujących słów)...")
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=MAX_FEATURES)

print(f"Dane treningowe: {len(x_train)} recenzji | Dane testowe: {len(x_test)}
recenzji")

# 3. Preprocessing: Standaryzacja długości sekwencji
print(f"Standaryzacja długości sekwencji do MAX_LEN = {MAX_LEN}...")
x_train = pad_sequences(x_train, maxlen=MAX_LEN)
```

```
x_test = pad_sequences(x_test, maxlen=MAX_LEN)

print(f"Kształt x_train po paddingu: {x_train.shape}")

# 4. Budowa Modelu BRNN (Bidirectional LSTM)
model = Sequential()

# 1. Warstwa osadzania (Embedding)
model.add(Embedding(MAX_FEATURES, EMBEDDING_DIM))

# 2. Warstwa Bi-directional RNN (Bidirectional LSTM)
model.add(Bidirectional(LSTM(64)))

# 3. Warstwa klasyfikacyjna (Dense)
model.add(Dense(1, activation='sigmoid'))

# 5. Kompilacja i Podsumowanie Modelu (Z Wymuszeniem Budowy)
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accura-
cy'])

# Wymuszenie budowy modelu, aby model.summary() wyświetlił poprawną liczbę pa-
rametrów
model.build(input_shape=(None, MAX_LEN))

print("\nPodsumowanie modelu BRNN (Bidirectional LSTM):")
model.summary()

# 6. Trening Modelu
print("\nRozpoczęcie treningu...")
history = model.fit(x_train, y_train, epochs=EPOCHS, batch_size=BATCH_SIZE,
validation_split=0.2, verbose=1)

# 7. Ocena Modelu
print("\nOcena modelu na zbiorze testowym...")
loss, accuracy = model.evaluate(x_test, y_test, verbose=0)
print(f"\nWynik końcowy na zbiorze testowym:")
print(f"Loss (Strata): {loss:.4f}")
print(f"Accuracy (Dokładność): {accuracy*100:.2f}%")
```

Wyniki:

```
Ładowanie zbioru danych IMDB (tylko 10000 najczęściej występujących słów)...
Dane treningowe: 25000 recenzji | Dane testowe: 25000 recenzji
Standaryzacja długości sekwencji do MAX_LEN = 100...
Kształt x_train po paddingu: (25000, 100)

Podsumowanie modelu BRNN (Bidirectional LSTM):

Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 100, 128)	1,280,000
bidirectional_1 (Bidirectional)	(None, 128)	98,816
dense_1 (Dense)	(None, 1)	129

```
 Total params: 1,378,945 (5.26 MB)

 Trainable params: 1,378,945 (5.26 MB)

 Non-trainable params: 0 (0.00 B)

Rozpoczęcie treningu...
Epoch 1/5
625/625 ━━━━━━━━━━━━━━━━━━━━ 14s 21ms/step - accuracy: 0.8004 - loss: 0.4267 - val_accuracy: 0.8478 - val_loss: 0.3474
Epoch 2/5
625/625 ━━━━━━━━━━━━━━━━━━━━ 14s 22ms/step - accuracy: 0.8987 - loss: 0.2516 - val_accuracy: 0.8296 - val_loss: 0.3830
Epoch 3/5
625/625 ━━━━━━━━━━━━━━━━━━━━ 15s 24ms/step - accuracy: 0.9406 - loss: 0.1593 - val_accuracy: 0.8264 - val_loss: 0.4108
Epoch 4/5
625/625 ━━━━━━━━━━━━━━━━━━━━ 18s 28ms/step - accuracy: 0.9696 - loss: 0.0865 - val_accuracy: 0.8242 - val_loss: 0.5193
Epoch 5/5
625/625 ━━━━━━━━━━━━━━━━━━━━ 17s 27ms/step - accuracy: 0.9819 - loss: 0.0535 - val_accuracy: 0.8200 - val_loss: 0.6660

Ocena modelu na zbiorze testowym...

Wynik końcowy na zbiorze testowym:
Loss (Strata): 0.6578
Accuracy (Dokładność): 82.35%
```

Analiza wyników:

Model trenowano przez 5 epok. Wyniki Końcowe na Zbiorze Testowym

Metryka	Wartość
Strata	0.6578
Dokładność	82.35%

Analiza Postępu treningu

Epoka	Dokładność	Strata	Dokładność (Walidacja)
1	80.04%	0.3474	84.78%
2	89.87%	0.3830	82.96%
3	94.06%	0.4188	82.64%
4	96.96%	0.5193	82.42%
5	98.19%	0.6660	82.00%

Analiza postępu treningu wykazała, że:

- Sukces Bidirectional LSTM: Zastosowanie Bidirectional LSTM znacznie poprawiło stabilność modelu w porównaniu do SimpleRNN, osiągając dobrą początkową dokładność (84.78% w Epoce 1)
- Nadmierne Dopasowanie (Overfitting): Model wykazuje wyraźne oznaki overfittingu, które zaczynają się już po pierwszej epoce. Dokładność treningowa rośnie niemal do 98.19% (uczenie się na pamięć), podczas gdy strata walidacyjna zaczyna systematycznie rosnąć z 0.3474 do 0.6660, a dokładność walidacyjna spada z 84.78% do 82.00%
- Optymalny Punkt Zatrzymania: Optymalny moment na zatrzymanie treningu (uzyskanie najlepszego wyniku na niewidzianych danych) nastąpił na koniec Epoki 1.

Link do repozytorium:https://github.com/Uczelniane/NODII.git

3. Wnioski

Aby jeszcze bardziej poprawić wyniki i zapobiec obserwowanemu nadmiernemu dopasowaniu, w przyszłych iteracjach zaleca się zastosowanie:

- Wczesnego Zatrzymania (Early Stopping): Monitorowanie straty walidacyjnej i automatyczne zatrzymanie treningu, gdy ta metryka przestaje spadać (w tym przypadku po 1. epoce).
- Dropoutu: Dodanie warstw Dropout po warstwie Embedding lub Bidirectional, aby zredukować współzależność między neuronami.