ГУАП

КАФЕДРА № 41

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

Ассистент		А.С. Раскопина
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1

ИСПОЛЬЗОВАНИЕ НЕЙРОННЫХ СЕТЕЙ ПРЯМОГО РАСПРОСТРАНЕНИЯ ДЛЯ РЕЩЕНИЯ ЗАДАЧ КЛАССИФИКАЦИИ

по курсу: МАШИННОЕ ОБУЧЕНИЕ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4217		Д.М. Никитин
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2025

Цель работы

Изучение основ работы с нейронными сетями прямого распространения (FNN) для классификации данных, обучение модели на подготовленном датасете, анализ и оценка полученных результатов.

Задачи

1. Ознакомиться с принципом работы сети прямого распространения (FNN) и её применением в задачах классификации.

2. Подготовить датасет для обучения модели.

3. Реализовать и обучить нейронную сеть прямого распространения (FNN) с использованием выбранного инструмента (PyTorch, TensorFlow или Keras).

4. Провести обучение сети на подготовленных данных.

5. Оценить точность работы модели и проанализировать полученные результаты.

6. Составить отчет, в котором будет описан процесс работы и выводы.

Последовательность выполнения работы:

1. Подготовка данных:

Выбрать датасет в зависимости от варианта

Разметить данные, если не размечены.

Провести предварительную обработку данных: нормализация, кодирование, разделение на обучающую и тестовую выборки.

2. Построение нейронной сети:

Создать архитектуру сети прямого распространения (определить количество слоев и нейронов).

Использовать фреймворк (PyTorch/TensorFlow/Keras) для реализации модели.

Применить функцию активации и функцию потерь.

3. Обучение сети:

Настроить параметры обучения: количество эпох, размер батча, скорость обучения.

Обучить модель на подготовленных данных.

4. Оценка и анализ результатов:

Оценить точность модели с использованием тестовой выборки.

Проанализировать ошибки, выявить возможные улучшения.

5. Отчет:

Оформить отчет, в котором будет описан процесс работы, результаты и выводы.

Вариант задания

Вариант 3: "Классификация текстов (IMDb Reviews)".

Описание задачи:

Используем датасет IMDb Reviews, который состоит из текстовых отзывов о фильмах, помеченных как положительные (positive) или отрицательные (negative). Нужно обучить нейронную сеть для классификации текстов на основе этих данных.

Шаги работы:

1. Загрузка данных: датасет IMDb Reviews доступен через Keras и на Kaggle.

2. Предобработка данных: тексты в датасете представлены в виде индексов слов в словаре. Надо преобразовать их обратно в текстовый формат для дальнейшей обработки. Нужно преобразовать индексы в массивы с использованием pad_sequences, чтобы тексты были одинаковой длины

3. Обучение модели:

Создать нейронную сеть для классификации текста на положительный или отрицательный отзыв.

4. Тестирование: протестировать модель на тестовых данных. Оценить точность модели на тестовой выборке.

Решение

Сначала был проведён просмотр датасета с помощью кода ниже. Результат см. на рис. 1.

```
import pandas as pd
```

```
dataset = pd.read_csv("IMDB Dataset.csv")
dataset.head(10)
```

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive
5	Probably my all-time favorite movie, a story o...	positive
6	I sure would like to see a resurrection of a u...	positive
7	This show was an amazing, fresh & innovative i...	negative
8	Encouraged by the positive comments about this...	negative
9	If you like original gut wrenching laughter yo...	positive

Рисунок 1 – Часть датасета в табличном представлении

Далее был очищен текст (удалены лишние символы, приведён к нижнему регистру) с помощью кода ниже. Получившийся датасет показан на рисунке 2.

```
import re

# Очистка текста
def clean_text(text: str) -> str:
  # Удаление тегов HTML
  text: str = re.sub(r'<.*?>', '', text)

  # Удаление символов, не являющихся буквой, цифрой, нижним
подчёркиванием или
  # пробелом
  text: str = re.sub(r'[^\w\s]', '', text)

  text: str = text.lower()

  return text

# Превращение positive и negative оценок в 1 и 0 соответственно
def sentiment_to_number(sentiment: str) -> int:
  if sentiment == "positive":
    return 1
  else:
    return 0

dataset['sentiment'] =
dataset['sentiment'].apply(sentiment_to_number)
dataset['review'] = dataset['review'].apply(clean_text)
```

```
dataset
```

	review	sentiment
0	one of the other reviewers has mentioned that ...	1
1	a wonderful little production the filming tech...	1
2	i thought this was a wonderful way to spend ti...	1
3	basically theres a family where a little boy j...	0
4	petter matteis love in the time of money is a ...	1
...	...	...
49995	i thought this movie did a down right good job...	1
49996	bad plot bad dialogue bad acting idiotic direc...	0
49997	i am a catholic taught in parochial elementary...	0
49998	im going to have to disagree with the previous...	0
49999	no one expects the star trek movies to be high...	0

50000 rows × 2 columns

Рисунок 2 – Обработанный датасет

Далее текст был преобразован в числовой формат или же токенизирован с помощью кода ниже. Токенизация — это процесс преобразования текста в последовательность чисел (токенов), которые могут быть обработаны нейронной сетью. В данном случае используется Tokenizer из библиотеки TensorFlow/Keras, который создаёт словарь из наиболее часто встречающихся слов и присваивает каждому слову уникальный индекс. Слова, не вошедшие в словарь, заменяются на специальный токен (Out Of Vocabulary). После этого тексты преобразуются в последовательности чисел, а затем дополняются нулями или обрезаются до одинаковой длины с помощью функции pad_sequences, чтобы все входные данные имели единый размер для обработки моделью.

```
import tensorflow as tf

vocab_size = 10000 # Размер словаря
max_length = 200 # Максимальный размер последовательности

# Создание токенизатора - редкоиспользуемые слова были помечены
как <OOV>
tokenizer =
tf.keras.preprocessing.text.Tokenizer(num_words=vocab_size,
```

```
oov_token="<OOV>")
# Обучение токенизатора
tokenizer.fit_on_texts(dataset['review'])

# Преобразование текста в последовательности чисел
sequences = tokenizer.texts_to_sequences(dataset['review'])

# Обрезанные и продлённые последовательности одной длины
padded_sequences =
tf.keras.preprocessing.sequence.pad_sequences(sequences,
maxlen=max_length, padding='post', truncating='post')
```

Далее данные были разделены на обучающую и тестовую выборки с помощью кода ниже.

```
from sklearn.model_selection import train_test_split

# 80% датасета используется для тренировки, 20% - для
тестирования
X_train, X_test, y_train, y_test =
train_test_split(padded_sequences, dataset['sentiment'],
test_size=0.2, random_state=0)
```

Создание нейронной сети начинается с определения её архитектуры с помощью tf.keras.Sequential, которая позволяет последовательно добавлять слои. Первый слой — Embedding, преобразует целочисленные индексы слов в плотные векторы фиксированной размерности (в данном случае 64), что позволяет модели работать с семантическим значением слов. Далее идёт слой Flatten, который "выравнивает" многомерные данные в одномерный вектор, подготавливая их для полносвязных слоёв. Затем добавляется полносвязный слой Dense с 64 нейронами и функцией активации ReLU, который помогает модели выявлять сложные зависимости в данных. Выходной слой Dense с одним нейроном и функцией активации sigmoid используется для бинарной классификации, выдавая вероятность принадлежности к классу 1 (например, положительный отзыв). После создания архитектуры модель компилируется с использованием оптимизатора Adam, функции потерь binary_crossentropy (для бинарной классификации) и метрики accuracy для оценки точности. Метод model.summary() выводит структуру модели, включая информацию о каждом слое, форме выходных данных и количестве обучаемых параметров. Это производится с помощью кода ниже. Результат его выполнения на рисунке 3.

```
# Создание модели
```

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(input_dim=vocab_size,
output_dim=64, input_length=max_length,
input_shape=(max_length,)), # Слой Embedding
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid') # Выходной
слой для бинарной классификации
])

# Компиляция модели
model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
    )

# Вывод структуры модели
print(model.summary())
```

Layer (type)	Output Shape	Param #
embedding_2 (Embedding)	(None, 200, 64)	640,000
flatten_2 (Flatten)	(None, 12800)	0
dense_4 (Dense)	(None, 64)	819,264
dense_5 (Dense)	(None, 1)	65

Total params: 1,459,329 (5.57 MB)
Trainable params: 1,459,329 (5.57 MB)
Non-trainable params: 0 (0.00 B)
None

Рисунок 3 – Структура созданной нейросети

Вывод говорит о том, что модель была создана верно. Отображены все 4 слоя, заданные в коде.

Слой embedding преобразует целочисленные индексы слов в плотные векторы фиксированной размерности. В данном случае каждое слово является вектором из 64 чисел.

Слой flatten преобразует её в вектор длины 200 * 64 = 12,800.

Верхний dense - полносвязный слой, который обучается на данных. Каждый нейрон в этом слое принимает входные данные от всех элементов предыдущего слоя и применяет к ним веса и смещения. Вход: вектор длины 12,800. Выход: вектор длины 64. Функция активации relu (Rectified Linear Unit) добавляет нелинейность: f(x) = max(0, x).

Нижний dense - выходной слой для бинарной классификации. Выдаёт вероятность принадлежности к классу 1. Вход: вектор длины 64. Выход: одно число (от 0 до 1). Функция активации sigmoid преобразует выход в вероятность: $f(x) = 1 / (1 + \exp(-x))$.

Далее было проведено обучение модели на данных для обучения. С помощью следующего кода.

```
# Обучение модели
history = model.fit(
    X_train, y_train,
    epochs=5,
    batch_size=512,
    validation_data=(X_test, y_test)
)
```

Во время выполнения команды model.fit происходит процесс обучения нейронной сети. Модель последовательно обрабатывает обучающие данные (X_train и y_train) в течение 5 эпох (полных проходов по всему набору данных). На каждом шаге данные разбиваются на батчи размером 512 примеров, что позволяет эффективно использовать память и ускорить обучение. В процессе обучения модель выполняет прямое распространение (forward pass), вычисляет ошибку (loss) с помощью функции потерь binary_crossentropy, а затем обновляет свои веса с помощью оптимизатора Adam, минимизируя ошибку. После каждой эпохи модель оценивается на валидационных данных (X_test и y_test), что позволяет отслеживать её обобщающую способность и избегать переобучения. Результаты обучения (ошибка и точность на тренировочных и валидационных данных) сохраняются в объекте history, который можно использовать для анализа и визуализации процесса обучения. Чем больше батчей, тем быстрее процесс обучения, но требуется больше памяти.

Далее была проведена оценка модели. Вывод кода ниже смотри на рисунке 4.

```
# Оценка модели
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Точность на тестовых данных: {accuracy:.2f}")
```

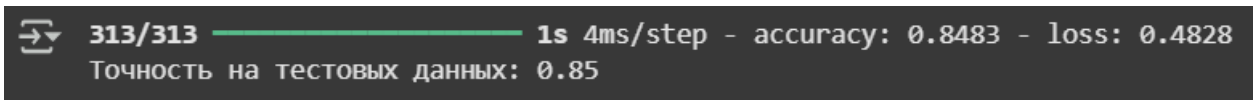

Рисунок 4 – Точность модели

Команда model.evaluate используется для оценки обученной модели на тестовых данных (X_test и y_test). В процессе выполнения модель обрабатывает тестовые данные, вычисляет значение функции потерь (loss) и точность (accuracy), которые показывают, насколько хорошо модель справляется с предсказаниями на новых, ранее не виденных данных. В данном случае выводится точность модели на тестовых данных, округлённая до двух знаков после запятой. Например, если точность составляет 0.85, это означает, что модель правильно классифицировала 85% тестовых примеров. Этот этап позволяет оценить качество модели и её способность обобщать знания на реальных данных.

Ссылка на Google Colab

https://colab.research.google.com/drive/1ZR7Ku73BtpU7rJvdfmnu2LEr-aivz5nS?usp=sharing

Выводы

В процессе данной работы была создана простая четырёхслойная сеть прямого распространения (FNN) с помощью TensorFlow. Модель была обучена и протестирована на датасете IMDb Reviews, содержащем текстовые отзывы о фильмах, размеченные как положительные или отрицательные. В ходе работы были выполнены все поставленные задачи: проведена предобработка данных (очистка текста, токенизация, padding), создана и обучена нейронная сеть, а также оценена её точность на тестовых данных. Модель показала точность 0.85, что свидетельствует о её способности успешно классифицировать отзывы. Это очень хороший результат для данной модели, так как она не сильно требовательная к аппаратному обеспечению и относительно проста в реализации. В процессе обучения использовались 5 эпох с размером батча 512, что позволило эффективно обучать модель, избегая переобучения. Результаты работы демонстрируют, что нейронные сети

прямого распространения могут быть эффективно применены для задач классификации текстов, а также подчёркивают важность корректной предобработки данных и настройки параметров модели для достижения высоких результатов. В дальнейшем, возможно, можно улучшить модель, добавив дополнительные слои, регуляризацию или используя более сложные архитектуры, такие как LSTM или GRU, для работы с последовательностями.