МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Доцент |  |  |  | В. Ю. Скобцов |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №5 |
| классификация на основе обработки текстовой информации. сверточные и рекурентные сети. полносвязные сети с предобучением и дообучением |
| по курсу: интеллектуальный анализ данных на основе методов машинного обучения |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ гр. № | 4134К |  |  |  | Д.В. Самарин,  Р. Р. Усов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

1. **Цель работы**

Дан набор данных из Keras: IMDB (Internet Movie Database) - множеством

из 50000 самых разных отзывов к кинолентам в интернет-базе фильмов. Набор

разбит на 25 000 обучающих и 25 000 контрольных отзывов, каждый набор на

50% состоит из отрицательных и на 50% из положительных отзывов.

Необходимо решить задачу эмоциональной окраски отзывов путем

классификации отзывов на фильмы на положительные и отрицательные,

опираясь на анализ текста отзывов. Задачу классификации решать при помощи

полносвязной нейросетевой модели и нейросетевой модели, указанной в

варианте.

Во всех наборах данных выделить обучающую, проверочную

(валидационную) и тестовую выборки данных.

Сравнить качество полносвязной нейросетевой классификационной

модели и классификационной нейросетевой модели, указанной в варианте, на

обучающей, валидационной и тестовой выборках. Для оценки качества моделей

использовать метрики: accuracy, F1 метрику (как по всей выборке, так и

отдельно по классам). Построить соответствующие графики метрик и ошибок.

1. **Вариант**





1. **Ход работы**

Загрузим и разобъём датасет IMDB

(X\_train, y\_train), (X\_test, y\_test) = imdb.load\_data(num\_words=20000)

word\_index = imdb.get\_word\_index()

max\_len = 100

X\_train = pad\_sequences(X\_train, maxlen=max\_len)

X\_test = pad\_sequences(X\_test, maxlen=max\_len)

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X\_train, y\_train, test\_size=0.2, random\_state=42)

input\_dim = 100

encoding\_dim = 64

input\_layer = Input(shape=(input\_dim,))

encoded = Dense(encoding\_dim, activation='relu')(input\_layer)

decoded = Dense(input\_dim, activation='sigmoid')(encoded)

autoencoder = Model(input\_layer, decoded)

encoder = Model(input\_layer, encoded)

autoencoder.compile(optimizer='adam', loss='mse')

autoencoder.fit(X\_train, X\_train, epochs=10, batch\_size=32, shuffle=True)

data\_transformed = encoder.predict(X\_train)

data\_transformed = encoder.predict(X\_test)

model\_dense = Sequential([

    Embedding(input\_dim=20000, output\_dim=128, input\_length=max\_len),

    Flatten(),

    Dense(128, activation='relu'),

    Dense(1, activation='sigmoid')

])

model\_dense.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

model\_cnn = Sequential([

    Embedding(input\_dim=20000, output\_dim=128, input\_length=max\_len),

    Conv1D(filters=32, kernel\_size=3, activation='relu', input\_shape=(max\_len, encoding\_dim)),

    MaxPooling1D(pool\_size=2),

    Flatten(),

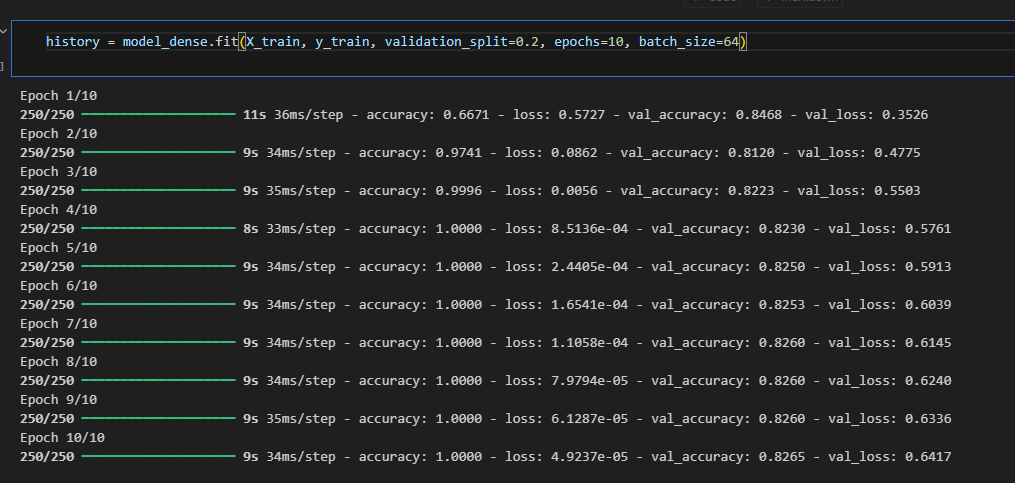
    Dense(1, activation='sigmoid')

])

model\_cnn.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

**Выполним обучение полносвязной модели**

history = model\_dense.fit(X\_train, y\_train, validation\_split=0.2, epochs=10, batch\_size=64)



plt.plot(history.history['accuracy'], label='Train Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.legend()

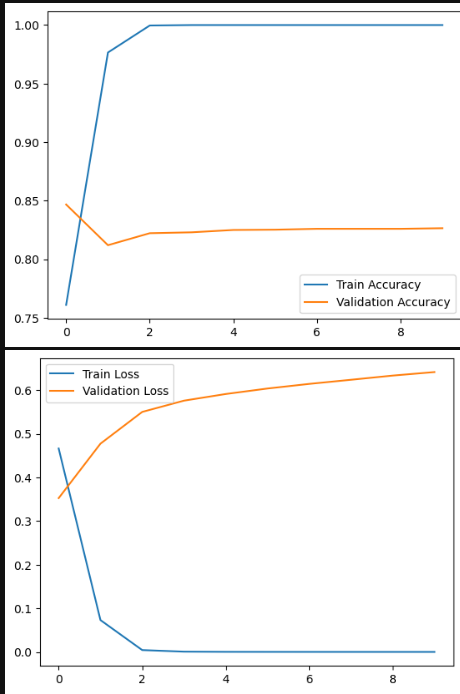
plt.show()

plt.plot(history.history['loss'], label='Train Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

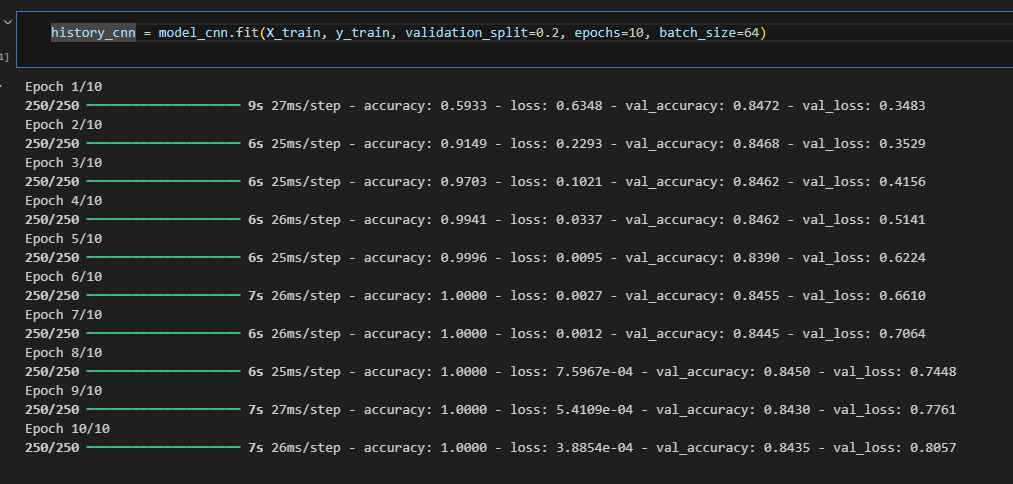
plt.legend()

plt.show()



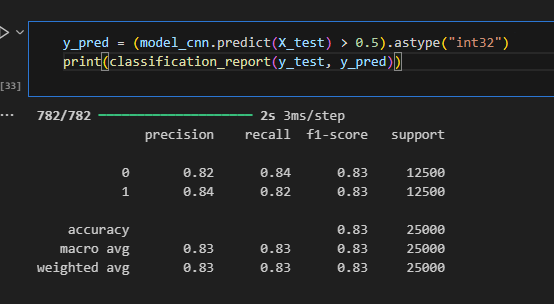
**Выполним обучение сверточной модели**

history\_cnn = model\_cnn.fit(X\_train, y\_train, validation\_split=0.2, epochs=10, batch\_size=64)



y\_pred = (model\_cnn.predict(X\_test) > 0.5).astype("int32")

print(classification\_report(y\_test, y\_pred))



plt.plot(history\_cnn.history['accuracy'], label='Train Accuracy')

plt.plot(history\_cnn.history['val\_accuracy'], label='Validation Accuracy')

plt.legend()

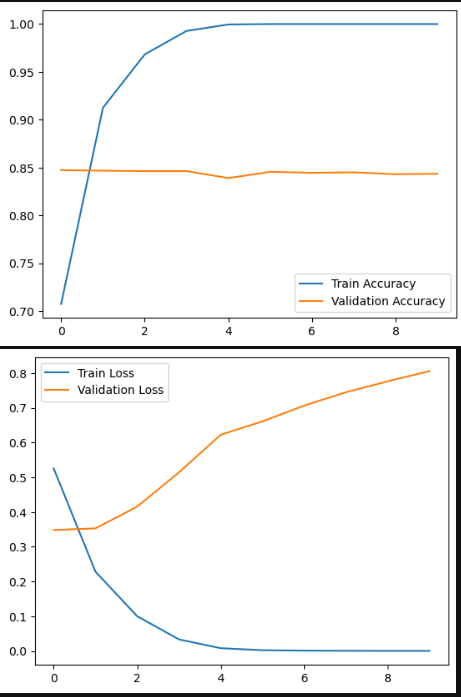
plt.show()

plt.plot(history\_cnn.history['loss'], label='Train Loss')

plt.plot(history\_cnn.history['val\_loss'], label='Validation Loss')

plt.legend()

plt.show()



**Сравним метрики F1, Loss, Accurancy для получившихся моделей**

y\_pred = (model\_dense.predict(X\_test) > 0.5).astype("int32")

y\_pred\_cnn = (model\_cnn.predict(X\_test) > 0.5).astype("int32")

acc = model\_dense.evaluate(X\_test, y\_test)

acc\_cnn = model\_cnn.evaluate(X\_test, y\_test)

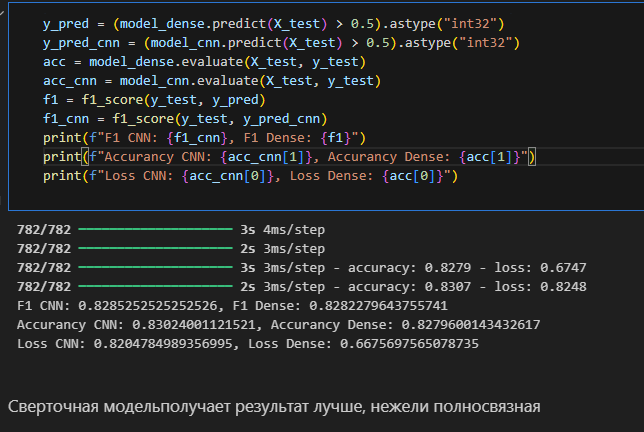
f1 = f1\_score(y\_test, y\_pred)

f1\_cnn = f1\_score(y\_test, y\_pred\_cnn)

print(f"F1 CNN: {f1\_cnn}, F1 Dense: {f1}")

print(f"Accurancy CNN: {acc\_cnn[1]}, Accurancy Dense: {acc[1]}")

print(f"Loss CNN: {acc\_cnn[0]}, Loss Dense: {acc[0]}")



1. **Выводы**

В ходе выполнения лабораторной работы был произведён сравнительный анализ двух нейронных сетей, в ходе которого сверточная сеть показала лучший результат