Лабораторная работа 3

Рекуррентные нейронные сети

Задание.

— Разберите предоставленные фрагменты кода, дайте комментарии к происходящему в них.

— Выберите композитора. Сформируйте набор данных, аналогичный существующему набору данных «Хоралы Баха», состоящему из 382 хоралов, сочиненных Иоганном Себастьяном Бахом, где каждый хорал имеет длину от 100 до 640 временных шагов, а каждый временной шаг содержит 4 целых числа, где каждое целое число соответствует индексу ноты на фортепиано (за исключением значения 0, которое означает, что нота не воспроизводится).

— Разработайте рекуррентную модель, которая на основе вашего набора данных сможет предсказать следующий временной шаг (четыре ноты). Затем используйте эту модель для создания музыки в стиле выбранного вами композитора.

Фрагмент кода 9-1 – Код инициализации для примера прогнозирования продаж в книжном магазине

```
import numpy as np
import matplotlib.pyplot as pltimport tensorflow as
tf
from tensorflow import keras
from tensorflow.keras.models import Sequentialfrom
tensorflow.keras.layers import Dense

from tensorflow.keras.layers import SimpleRNNimport
logging tf.get_logger().setLevel(logging.ERROR)

EPOCHS = 100
BATCH_SIZE = 16
TRAIN_TEST_SPLIT = 0.8
MIN = 12
FILE_NAME = '../data/book_store_sales.csv'

def readfile(file_name):
file = open(file_name, 'r', encoding='utf-8')next(file)
data = []
for line in (file):
values = line.split(',') data.append(float(values[1]))
file.close()
return np.array(data, dtype=np.float32)

# Считывание данных и разделение их на обучающие и тестовые.
sales = readfile(FILE_NAME)
months = len(sales)
split = int(months * TRAIN_TEST_SPLIT)
train_sales = sales[0:split] test_sales =
sales[split:]
```

Фрагмент кода 9-2 – Код для создания графика исторических данных о продажах

```
# Набор данных
x = range(len(sales))
plt.plot(x, sales, 'r-', label='book sales')
plt.title('Book store sales') plt.axis([0, 339, 0.0,
3000.0])
plt.xlabel('Months') plt.ylabel('Sales (millions $)')
plt.legend()
plt.show()
```

Фрагмент кода 9-3 – Код для вычисления и построения графика наивного предсказания

```
# Наивное предсказание

test_output = test_sales[MIN:]
naive_prediction = test_sales[MIN-1:-1]
x = range(len(test_output))
plt.plot(x, test_output, 'g-', label='test_output') plt.plot(x, naive_prediction,
'm-', label='naive prediction')plt.title('Book store sales')
plt.axis([0, len(test_output), 0.0, 3000.0])plt.xlabel('months')
plt.ylabel('Monthly book store sales')plt.legend()
plt.show()
```

Фрагмент кода 9-4 – Стандартизация данных

```
# Стандартизация обучающих и тестовых данных.

# Использование только тренировочных периодов для вычисления среднего и среднеквадратичного
значения.

mean = np.mean(train_sales)
stddev =  np.mean(train_sales) train_sales_std = (train_sales -
mean)/stddevtest_sales_std = (test_sales - mean)/stddev
```

Фрагмент кода 9-5 – Распределение и заполнение тензоров для обучающих и тестовых данных

```
# Создание учебных примеров.

train_months = len(train_sales)
train_X = np.zeros((train_months-MIN, train_months-1, 1))train_y =
np.zeros((train_months-MIN, 1))
for i in range(0, train_months-MIN):
    train_X[i, -(i+MIN):, 0] = train_sales_std[0:i+MIN]train_y[i, 0] =
    train_sales_std[i+MIN]

# Создание тестовых примеров.
test_months = len(test_sales)
test_X = np.zeros((test_months-MIN, test_months-1, 1))test_y =
np.zeros((test_months-MIN, 1))
for i in range(0, test_months-MIN):
    test_X[i, -(i+MIN):, 0] = test_sales_std[0:i+MIN]test_y[i, 0] =
    test_sales_std[i+MIN]
```

Фрагмент кода 9-6 Определение двухслойной модели с одним рекуррентным слоем и одним плотным слоем

```
# Создание RNN модели

model = Sequential()
model.add(SimpleRNN(128, activation='relu',
                    input_shape=(None, 1))) model.add(Dense(1,
activation='linear')) model.compile(loss='mean_squared_error', optimizer = 'adam',
              metrics =['mean_absolute_error'])model.summary()
history = model.fit(train_X, train_y,
                    validation_data
                    = (test_X, test_y), epochs=EPOCHS,batch_size=BATCH_SIZE,
                    verbose=2, shuffle=True)
```

Фрагмент кода 9-7 Вычисление наивного предсказания, MSE и MAE на стандартизированных данных

```
# Создание наивного прогноза на основе стандартизированных данных.

test_output = test_sales_std[MIN:]
naive_prediction = test_sales_std[MIN-1:-1] mean_squared_error =
np.mean(np.square(naive_prediction - test_output))mean_abs_error =
np.mean(np.abs(naive_prediction - test_output))
print('naive test mse: ', mean_squared_error)
print('naive test mean abs: ', mean_abs_error)
```

Фрагмент кода 9-8 – Использование модели для прогнозирования результатов обучения и тестирования и дестандартизация результатов

```
# Использование обученной модели для прогнозирования тестовых данных

predicted_test = model.predict(test_X, len(test_X))predicted_test =
np.reshape(predicted_test,
                             (len(predicted_test)))predicted_test =
predicted_test * stddev + mean


x = range(len(test_sales)-MIN) plt.plot(x, predicted_test, 'm-
',
         label='predicted test_output') plt.plot(x, test_sales[-
(len(test_sales)-MIN):],
         'g-', label='actual test_output')plt.title('Book
sales')
plt.axis([0, 55, 0.0, 3000.0])
plt.xlabel('months') plt.ylabel('Predicted book sales')
plt.legend()
plt.show()
```

Фрагмент кода 9-9 – Сокращение периода возврата до 7 дней

```
# Уменьшение периода ожидания при вводе.
train_X = train_X[:, (train_months - 13):, :]

test_X = test_X[:, (test_months - 13):, :]

# Создание модели с прямой связью.

model.add(Flatten(input_shape=(12, 1))) model.add(Dense(256,
```

```
activation='relu'))model.add(Dense(1, activation='linear'))
```

Фрагмент кода 9-10 Создание входных данных и модели с двумя входными переменными на временной шаг

```
# Создание тренировочных примеров

train_months = len(train_sales)
train_X = np.zeros((train_months-MIN, train_months-1, 2))
train_y = np.zeros((train_months-MIN, 1))
for i in range(0, train_months-MIN):
    train_X[i, -(i+MIN):, 0] = train_sales_std[0:i+MIN]
    train_X[i, -(i+MIN):, 1] = train_sales_std2[0:i+MIN]train_y[i, 0] =
    train_sales_std[i+MIN]

# Создание тестовых примеров

test_months = len(test_sales)
test_X = np.zeros((test_months-MIN, test_months-1, 2))
test_y = np.zeros((test_months-MIN, 1))
for i in range(0, test_months-MIN):
    test_X[i, -(i+MIN):, 0] = test_sales_std[0:i+MIN]
    test_X[i, -(i+MIN):, 1] = test_sales_std2[0:i+MIN]
    test_y[i, 0] = test_sales_std[i+MIN]
…
model.add(SimpleRNN(128, activation='relu',
                    input_shape=(None, 2)))
```