|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра прикладной математики (ПМ)**

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**

по дисциплине «Технологии и инструментарий машинного обучения»

**Практическое занятие № 5**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент группы ИНБО-01-17 | *ИМБО-02-22, Ким Кирилл Сергеевич* | (подпись) | |
| Преподаватель | *Трушин Степан Михайлович, преподаватель* | (подпись) | |
| Отчет представлен | «\_\_\_»\_\_\_\_\_\_\_\_202\_\_г. | |  | |

Москва 2025 г.

СОДЕРЖАНИЕ

[ПРАКТИЧЕСКАЯ РАБОТА 5 3](#_Toc197877287)

[Задание 3](#_Toc197877288)

[Введение 4](#_Toc197877289)

[Шаги выполнения 5](#_Toc197877290)

[Этап 1. Подготовка данных 5](#_Toc197877291)

[Этап 2. Рекуррентные нейронные сети (1 пара) 9](#_Toc197877292)

[Вывод: 13](#_Toc197877293)

[Список использованных источников и литературы: 13](#_Toc197877294)

ПРАКТИЧЕСКАЯ РАБОТА 5

Задание

**Часть 1: Автокодировщик (1 пара)**

1. Подготовка данных:

* Загрузите датасет изображений (например, MNIST или Fashion MNIST).
* Выполните нормализацию данныхизображений.

1. Создание автокодировщика:

* Постройте сеть с симметричной архитектурой:
* Входной слой, сжимающий изображение до вектора.
* Скрытые слои, выполняющие кодирование данных.
* Декодирующие слои, восстанавливающие изображение из вектора.

1. Обучение автокодировщика:

* Обучите модель на обучающей выборке.
* Оцените качество восстановления изображений на тестовой выборке.

1. Визуализация:

* Постройте графики исходных и восстановленных изображений.
* Визуализируйте скрытые представления данных. нормализации.

1. Эксперимент:

* Изучите влияние уменьшения размера скрытого представления (вектора) на качество восстановления.

**Часть 2: Рекуррентные нейронные сети (1 пара)**

1. Подготовка данных:

* Используйте набор последовательных данных, например, текстовый датасет (предсказание следующего слова) или временной ряд (предсказание следующего значения)

1. Реализация рекуррентной сети:

* Постройте базовую рекуррентную сеть с одним рекуррентным слоем (RNN).
* Обучите её на предоставленных данных.

1. Расширение модели:

* Замените базовую RNN на LSTM и GRU.
* Сравните их поведение и точность на тестовой выборке.

1. Визуализация:

* Графики ошибки и точности для каждой архитектуры (RNN, LSTM, GRU).
* Визуализация предсказаний на временных рядах или текстах.

Введение

**Цель**

Освоить принципы работы автокодировщиков для сжатия и восстановления данных, а также изучить применение рекуррентных нейронных сетей (RNN, LSTM, GRU) для обработки временных последовательностей и текстов.

Шаги выполнения

Этап 1. Подготовка данных

1. Подготовка данных.

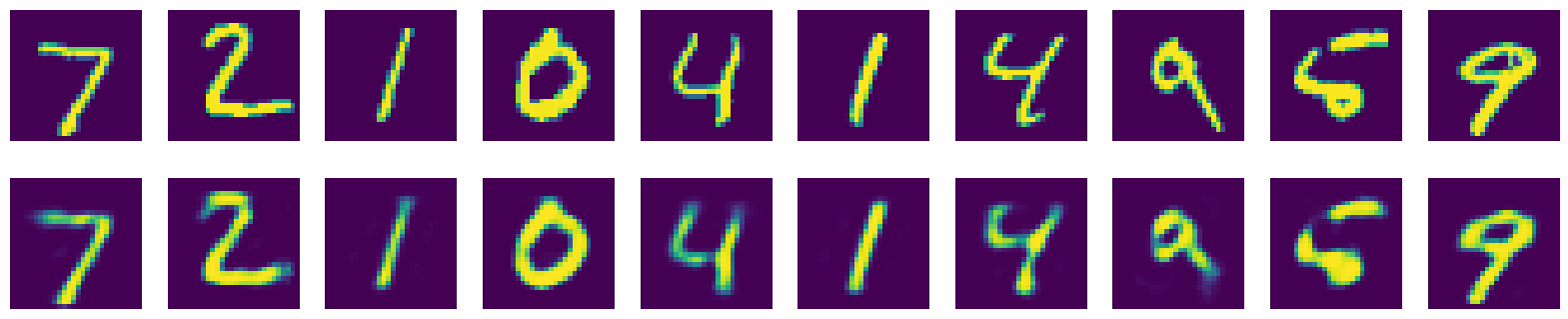
Листинг 1 – Загрузка данных

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  from tensorflow.keras.datasets import mnist  from tensorflow.keras.models import Model  from tensorflow.keras.layers import Input, Dense  (x\_train, \_), (x\_test, \_) = mnist.load\_data()  x\_train = x\_train.astype('float32') / 255.  x\_test = x\_test.astype('float32') / 255.  x\_train = x\_train.reshape((len(x\_train), 784)) # 28\*28  x\_test = x\_test.reshape((len(x\_test), 784)) |

1. Создание и обучение автокодировщика с размером encoding\_dim = 32

Листинг 2 – Создание, обучение автокодировщика и визуализация

|  |
| --- |
| encoding\_dim = 32 # размер скрытого вектора  input\_img = Input(shape=(784,))  encoded = Dense(encoding\_dim, activation='relu')(input\_img)  decoded = Dense(784, activation='sigmoid')(encoded)  autoencoder = Model(input\_img, decoded)  autoencoder.compile(optimizer='adam',  loss='binary\_crossentropy')  autoencoder.fit(x\_train, x\_train,  epochs=20,  batch\_size=256,  shuffle=True,  validation\_data=(x\_test, x\_test))  decoded\_imgs = autoencoder.predict(x\_test)  n = 10  plt.figure(figsize=(20, 4))  for i in range(n):  # Оригинальные изображения  ax = plt.subplot(2, n, i + 1)  plt.imshow(x\_test[i].reshape(28, 28))  plt.axis('off')  # Восстановленные изображения  ax = plt.subplot(2, n, i + 1 + n)  plt.imshow(decoded\_imgs[i].reshape(28, 28))  plt.axis('off')  plt.show() |

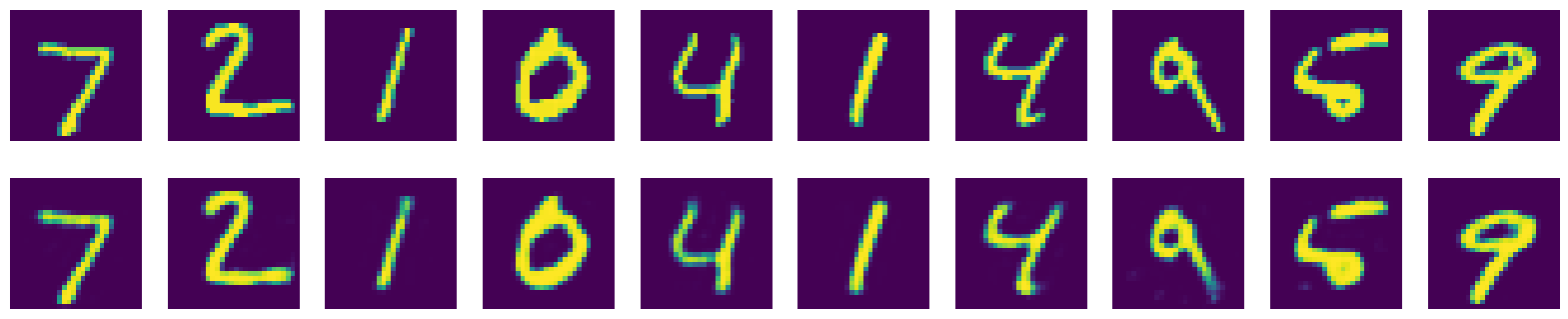


**Рисунок 1 – Оригинальные и восстановленные изображения**

1. Создание и обучение автокодировщика с размером encoding\_dim = 64

Листинг 3 – Создание, обучение автокодировщика и визуализация

|  |
| --- |
| encoding\_dim = 64 # размер скрытого вектора  input\_img = Input(shape=(784,))  encoded = Dense(encoding\_dim, activation='relu')(input\_img)  decoded = Dense(784, activation='sigmoid')(encoded)  autoencoder = Model(input\_img, decoded)  autoencoder.compile(optimizer='adam',  loss='binary\_crossentropy')  autoencoder.fit(x\_train, x\_train,                  epochs=20,                  batch\_size=256,                  shuffle=True,                  validation\_data=(x\_test, x\_test))  decoded\_imgs = autoencoder.predict(x\_test)  n = 10  plt.figure(figsize=(20, 4))  for i in range(n):      # Оригинальные изображения      ax = plt.subplot(2, n, i + 1)      plt.imshow(x\_test[i].reshape(28, 28))      plt.axis('off')      # Восстановленные изображения      ax = plt.subplot(2, n, i + 1 + n)      plt.imshow(decoded\_imgs[i].reshape(28, 28))      plt.axis('off')  plt.show() |

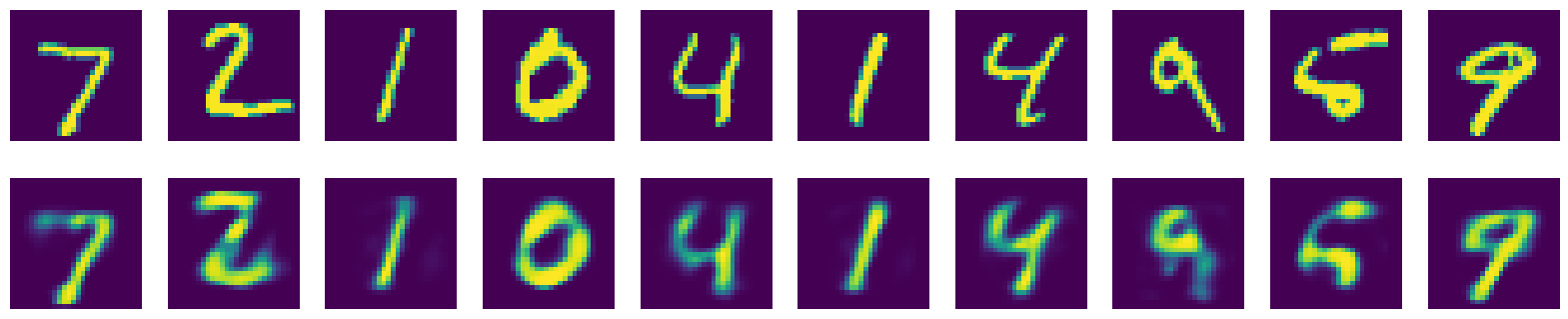


**Рисунок 2 – Оригинальные и восстановленные изображения**

1. Создание и обучение автокодировщика с размером encoding\_dim = 16

Листинг 4 – Создание, обучение автокодировщика и визуализация

|  |
| --- |
| encoding\_dim = 16 # размер скрытого вектора  input\_img = Input(shape=(784,))  encoded = Dense(encoding\_dim, activation='relu')(input\_img)  decoded = Dense(784, activation='sigmoid')(encoded)  autoencoder = Model(input\_img, decoded)  autoencoder.compile(optimizer='adam',  loss='binary\_crossentropy')  autoencoder.fit(x\_train, x\_train,  epochs=20,  batch\_size=256,  shuffle=True,  validation\_data=(x\_test, x\_test))  decoded\_imgs = autoencoder.predict(x\_test)  n = 10  plt.figure(figsize=(20, 4))  for i in range(n):  # Оригинальные изображения  ax = plt.subplot(2, n, i + 1)  plt.imshow(x\_test[i].reshape(28, 28))  plt.axis('off')  # Восстановленные изображения  ax = plt.subplot(2, n, i + 1 + n)  plt.imshow(decoded\_imgs[i].reshape(28, 28))  plt.axis('off')  plt.show() |

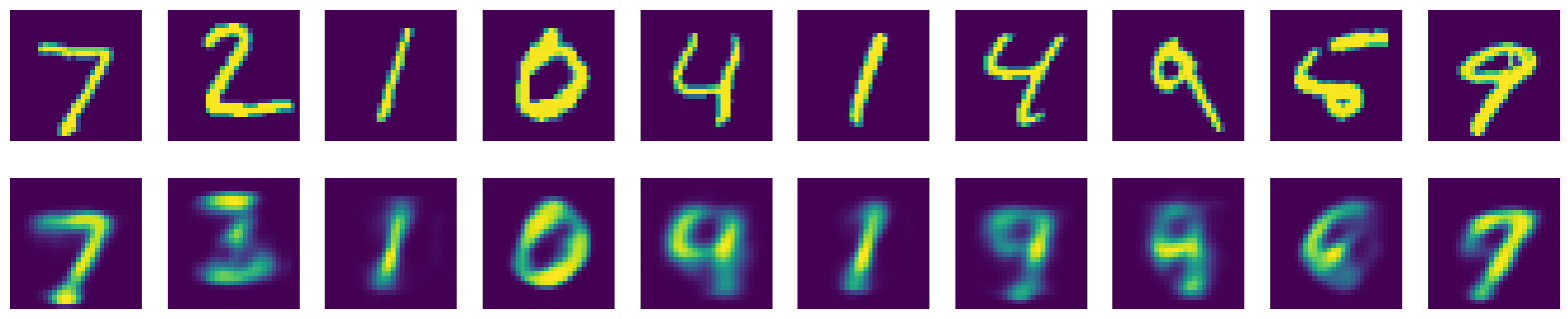


**Рисунок 3 – Оригинальные и восстановленные изображения**

1. Создание и обучение автокодировщика с размером encoding\_dim = 8

Листинг 5 – Создание, обучение автокодировщика и визуализация

|  |
| --- |
| encoding\_dim = 8 # размер скрытого вектора  input\_img = Input(shape=(784,))  encoded = Dense(encoding\_dim, activation='relu')(input\_img)  decoded = Dense(784, activation='sigmoid')(encoded)  autoencoder = Model(input\_img, decoded)  autoencoder.compile(optimizer='adam',  loss='binary\_crossentropy')  autoencoder.fit(x\_train, x\_train,                  epochs=20,                  batch\_size=256,                  shuffle=True,                  validation\_data=(x\_test, x\_test))  decoded\_imgs = autoencoder.predict(x\_test)  n = 10  plt.figure(figsize=(20, 4))  for i in range(n):      # Оригинальные изображения      ax = plt.subplot(2, n, i + 1)      plt.imshow(x\_test[i].reshape(28, 28))      plt.axis('off')      # Восстановленные изображения      ax = plt.subplot(2, n, i + 1 + n)      plt.imshow(decoded\_imgs[i].reshape(28, 28))      plt.axis('off')  plt.show() |



**Рисунок 4 – Оригинальные и восстановленные изображения**

Этап 2. Рекуррентные нейронные сети (1 пара)

1. Подготовка текста:

Листинг 6 – Подготовка текста

|  |
| --- |
| text = open("input.txt").read().lower()  seq\_length = 40  step = 3  sentences = []  next\_chars = []  for i in range(0, len(text) - seq\_length, step):  sentences.append(text[i: i + seq\_length])  next\_chars.append(text[i + seq\_length])  chars = sorted(list(set(text)))  char\_indices = dict((c, i) for i, c in enumerate(chars))  indices\_char = dict((i, c) for i, c in enumerate(chars))  X = np.zeros((len(sentences), seq\_length, len(chars)),  dtype=bool)  y = np.zeros((len(sentences), len(chars)), dtype=bool)  for i, sentence in enumerate(sentences):  for t, char in enumerate(sentence):  X[i, t, char\_indices[char]] = 1  y[i, char\_indices[next\_chars[i]]] = 1 |

1. Базовая RNN

Листинг 7 – Реализация рекуррентной сети:

|  |
| --- |
| from tensorflow.keras.models import Sequential  from tensorflow.keras.layers import SimpleRNN, Dense  model\_rnn = Sequential()  model\_rnn.add(SimpleRNN(128, input\_shape=(seq\_length,  len(chars))))  model\_rnn.add(Dense(len(chars), activation='softmax'))  model\_rnn.compile(loss='categorical\_crossentropy',  optimizer='adam', metrics=['accuracy'])  history\_rnn = model\_rnn.fit(X, y, batch\_size=128, epochs=10, validation\_split=0.2) |

1. Модификации: LSTM и GRU

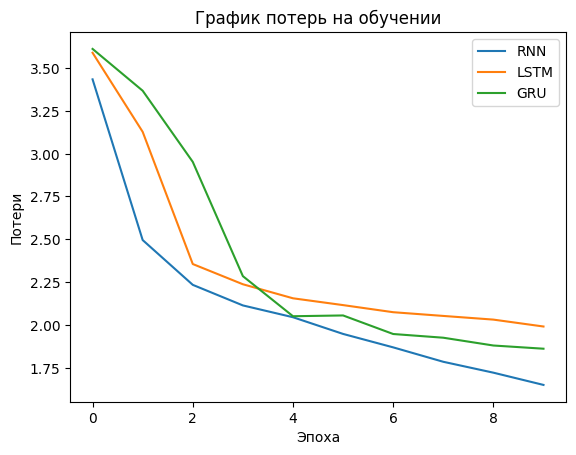
Листинг 8 – Расширение модели:

|  |
| --- |
| # Модель LSTM:  from tensorflow.keras.layers import LSTM  model\_lstm = Sequential()  model\_lstm.add(LSTM(128, input\_shape=(seq\_length, len(chars))))  model\_lstm.add(Dense(len(chars), activation='softmax'))  model\_lstm.compile(loss='categorical\_crossentropy',  optimizer='adam', metrics=['accuracy'])  history\_lstm = model\_lstm.fit(X, y, batch\_size=128, epochs=10, validation\_split=0.2)  # Модель GRU:  from tensorflow.keras.layers import GRU  model\_gru = Sequential()  model\_gru.add(GRU(128, input\_shape=(seq\_length, len(chars))))  model\_gru.add(Dense(len(chars), activation='softmax'))  model\_gru.compile(loss='categorical\_crossentropy',  optimizer='adam', metrics=['accuracy'])  history\_gru = model\_gru.fit(X, y, batch\_size=128, epochs=10,  validation\_split=0.2) |

1. Визуализация результатов

Листинг 9 – Визуализация потери

|  |
| --- |
| import matplotlib.pyplot as plt  plt.plot(history\_rnn.history['loss'], label='RNN')  plt.plot(history\_lstm.history['loss'], label='LSTM')  plt.plot(history\_gru.history['loss'], label='GRU')  plt.title("График потерь на обучении")  plt.xlabel("Эпоха")  plt.ylabel("Потери")  plt.legend()  plt.show() |



**Рисунок 5 – График ошибки**

Данную работу можете увидеть в блокноте Google Colab:

<https://colab.research.google.com/drive/1kS4TYommK70YnGpcgumXV7n6sn34o05z?usp=sharing>

Вывод:

Таким образом, автокодировщики эффективны для задач сжатия данных и выделения признаков, при этом качество восстановления напрямую зависит от размера скрытого представления. При 32, 64 нейронах качество практически идеальное, а при 8, 16 не очень.

Для работы с LSTM и GRU предпочтительнее базовой RNN благодаря их способности запоминать долгосрочные зависимости.

LSTM продемонстрировала лучшую точность благодаря механизму запоминания долгосрочных зависимостей

GRU показала сравнимую с LSTM точность при меньших вычислительных затратах.

Список использованных источников и литературы:

1. Ростовцев В.С. Искусственные нейронные сети,   
   Издательство "Лань", 2019. — 216 с. — URL: https://e.lanbook.com/book/122180
2. Араки М. Манга: Машинное обучение,   
   Издательство "ДМК Пресс", 2020. — 214 с. — URL: <https://e.lanbook.com/book/179473>
3. Платонов, А. В. Машинное обучение : учебное пособие для вузов / А. В. Платонов. — Москва : Издательство Юрайт, 2022. — 85 с. — (Высшее образование). — ISBN 978-5-534-15561-7. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/508804