|  |
| --- |
| 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 № 6**

|  |  |  |  |
| --- | --- | --- | --- |
| 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 6

Задание

**Часть 1: Описание результатов обучения нейронной сети (1 пара)**

1. Выберите один из завершенных проектов из предыдущих практических занятий (например, полносвязная сеть, сверточная сеть или автокодировщик).
2. Подготовьте следующие графики:

* Потери (loss) на обучающей и тестовой выборках.
* Точность классификации или метрики качества (точность, F1-мера) на тестовой выборке.
* Визуализация ошибок классификации или восстановления данных.

1. Оформите результаты в виде краткого отчета с основными выводами о качестве модели, возможных проблемах и их решениях.

**Часть 2: Шаги реализации проекта на основе нейронных сетей (1 пара)**

1. Опишите процесс реализации проекта в формате шагов:

* Постановка задачи: краткое описание задачи, набор данных, цель.
* Предварительная обработка данных: нормализация, преобразования.
* Архитектура модели: выбор слоев, функций активации, оптимизаторов.
* Обучение модели: настройка гиперпараметров, визуализация процесса обучения.
* Оценка модели: метрики, анализ ошибок.

1. Укажите, как данные шаги могут быть применены в рамках выпускной работы.
2. Подготовьте сводную таблицу с параметрами, метриками и рекомендациями по улучшению модели.

**Часть 3: Итоговая защита и обсуждение (1 пара)**

1. Представьте подготовленный отчет на защите.
2. Обсудите:

* Какую роль играет предварительная обработка данных?
* Какие улучшения можно внести в архитектуру модели?
* Какие проблемы возникли в процессе выполнения проекта, и как они были решены?

Введение

**Цель**

Научиться систематизировать результаты работы нейронных сетей, правильно оформлять графики, таблицы и выводы для ВКР. Освоить базовые шаги при реализации проектов на основе нейронных сетей и подготовке научных отчетов.

Шаги выполнения

Этап 1. Выбор завершённого проекта

1. Подготовка данных.

Листинг 1 – Загрузка данных

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  import seaborn as sns  from tensorflow import keras  from tensorflow.keras import layers  from tensorflow.keras.datasets import mnist  from sklearn.metrics import classification\_report, confusion\_matrix  # Загрузка датасета MNIST  (x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()  # Преобразование изображений в векторы и нормализация  x\_train = x\_train.reshape((x\_train.shape[0], 28, 28, 1)).astype('float32') / 255  x\_test = x\_test.reshape((x\_test.shape[0], 28, 28, 1)).astype('float32') / 255  # Загрузка датасета MNIST  (x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()  # Преобразование изображений в векторы и нормализация  x\_train = x\_train.reshape((x\_train.shape[0], 28, 28, 1)).astype('float32') / 255  x\_test = x\_test.reshape((x\_test.shape[0], 28, 28, 1)).astype('float32') / 255 |

1. Архитектура нейронной сети

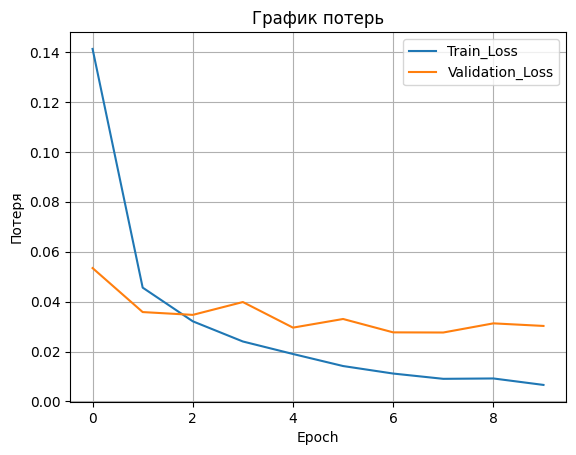
Листинг 2 – Создание, обучение нейронной сети

|  |
| --- |
| # Создание сверточной модели (CNN)  model = keras.Sequential([  layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(28, 28, 1)),  layers.MaxPooling2D((2, 2)),  layers.Conv2D(64, (3, 3), activation='relu'),  layers.MaxPooling2D((2, 2)),  layers.Flatten(),  layers.Dense(64, activation='relu'),  layers.Dense(10, activation='softmax')  ])  model.compile(optimizer='adam',  loss='sparse\_categorical\_crossentropy',  metrics=['accuracy'])  # Обучение модели  history = model.fit(x\_train, y\_train,                      epochs=10,                      batch\_size=32,                      validation\_data=(x\_test, y\_test))  # Оценка модели  test\_loss, test\_acc = model.evaluate(x\_test, y\_test)  print(f'Test accuracy: {test\_acc:.4f}') |

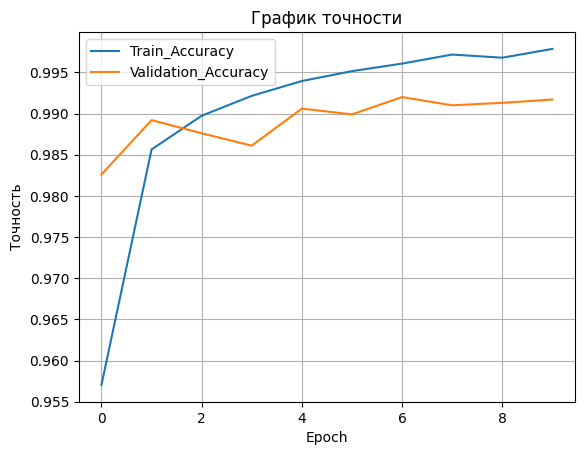
1. Результаты

Листинг 3 – Графики потерь и точности

|  |
| --- |
| # Визуализация результатов  # График потерь  plt.plot(history.history['loss'], label='Train\_Loss')  plt.plot(history.history['val\_loss'], label='Validation\_Loss')  plt.xlabel('Epoch')  plt.ylabel('Потеря')  plt.title('График потерь')  plt.legend()  plt.grid()  plt.show()  # График точности  plt.plot(history.history['accuracy'], label='Train\_Accuracy')  plt.plot(history.history['val\_accuracy'], label='Validation\_Accuracy')  plt.xlabel('Epoch')  plt.ylabel('Точность')  plt.title('График точности')  plt.legend()  plt.grid()  plt.show() |



**Рисунок 2 – График потерь**

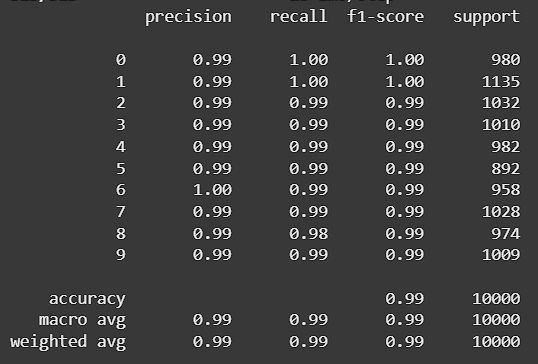


**Рисунок 2 – График точности**

1. Таблица с основными метриками качества

Листинг 4 – Таблица метрик

|  |
| --- |
| # Отчет классификации  y\_pred = model.predict(x\_test)  y\_pred\_classes = np.argmax(y\_pred, axis=1)  print(classification\_report(y\_test, y\_pred\_classes)) |

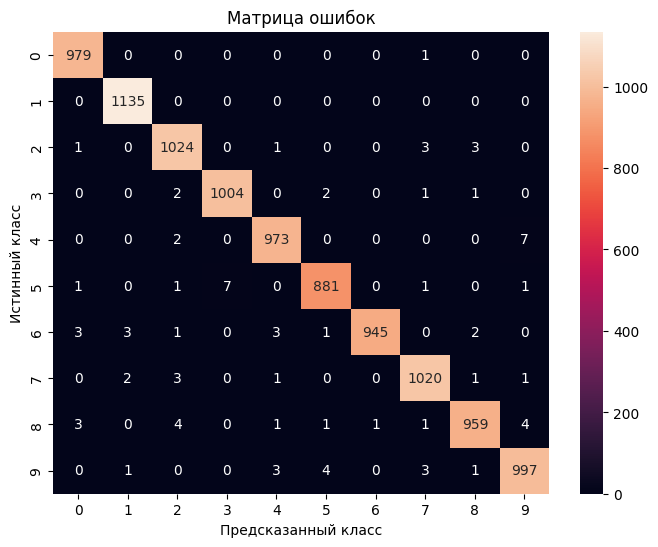


**Рисунок 3 – Отчет классификации**

1. Матрица ошибок

Листинг 5 – Визуализация ошибок классификации

|  |
| --- |
| # Матрица ошибок  cm = confusion\_matrix(y\_test, y\_pred\_classes)  plt.figure(figsize=(8, 6))  sns.heatmap(cm, annot=True, fmt='d')  plt.title('Матрица ошибок')  plt.xlabel('Предсказанный класс')  plt.ylabel('Истинный класс')  plt.show() |

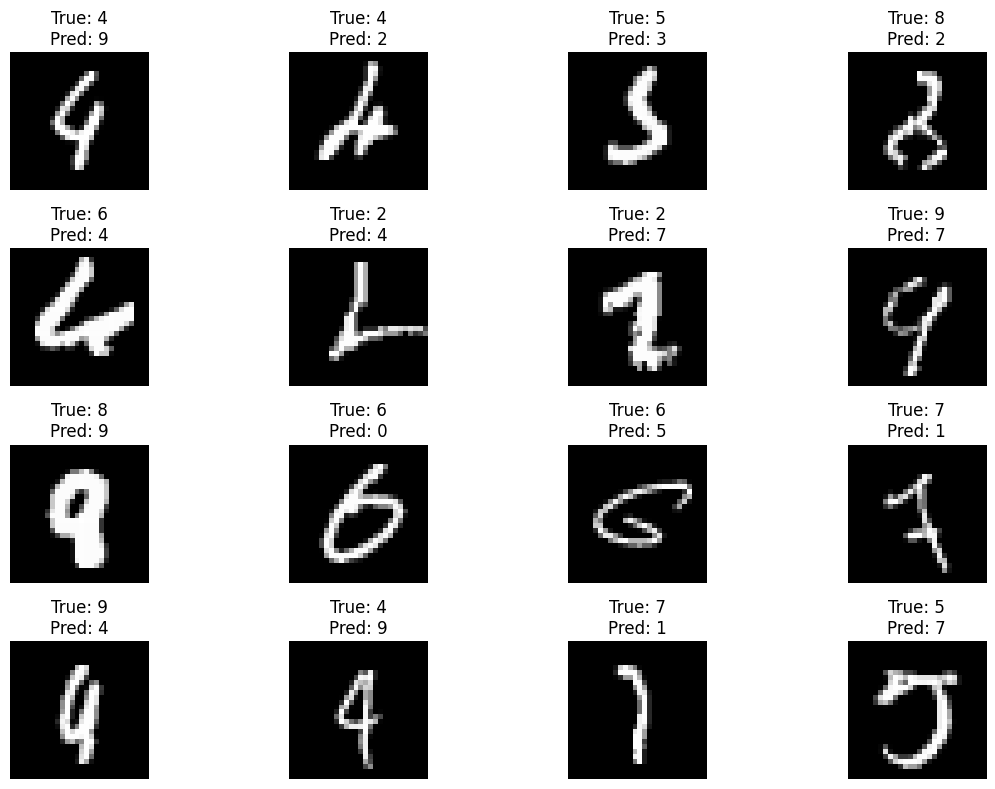


**Рисунок 4 – Матрица ошибок**

1. Визуализация изображений

Листинг 6 – Визуализация цифр

|  |
| --- |
| # Визуализация примеров ошибок  errors = (y\_pred\_classes != y\_test)  error\_images = x\_test[errors]  error\_pred = y\_pred\_classes[errors]  error\_true = y\_test[errors]  plt.figure(figsize=(12, 8))  for i in range(16):  plt.subplot(4, 4, i+1)  plt.imshow(error\_images[i].reshape(28, 28), cmap='gray')  plt.title(f'True: {error\_true[i]}\nPred: {error\_pred[i]}')  plt.axis('off')  plt.tight\_layout()  plt.show() |



**Рисунок 5 – Визуализация примеров**

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

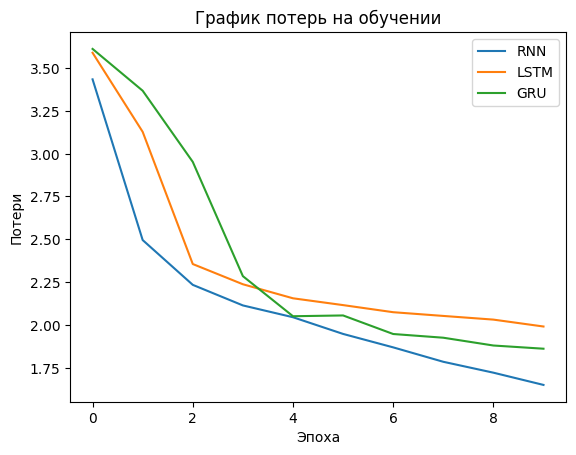
Листинг 8 – Расширение модели:

|  |
| --- |
| # 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) |

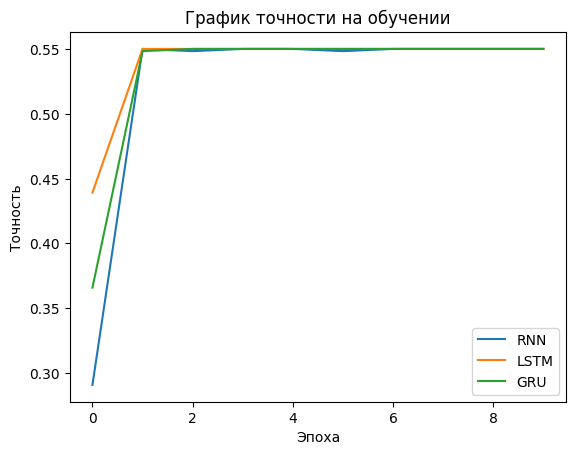
1. Визуализация результатов

Листинг 9 – Визуализация точности и 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()  import matplotlib.pyplot as plt  plt.plot(history\_rnn.history['accuracy'], label='RNN')  plt.plot(history\_lstm.history['accuracy'], label='LSTM')  plt.plot(history\_gru.history['accuracy'], label='GRU')  plt.title("График точности на обучении")  plt.xlabel("Эпоха")  plt.ylabel("Точность")  plt.legend()  plt.show() |



**Рисунок 5 – График ошибки**



**Рисунок 6 – График точности**

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