|  |
| --- |
| 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 № 2**

|  |  |  |  |
| --- | --- | --- | --- |
| 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 1 3](#_Toc190869312)

[Задание 3](#_Toc190869313)

[Введение 5](#_Toc190869314)

[Шаги выполнения 5](#_Toc190869315)

[Этап 1. Общий пример MNIST 5](#_Toc190869316)

[Этап 2. Индивидуальное задание 8](#_Toc190869317)

[Этап 3. Сравнительный анализ и защита 12](#_Toc190869318)

[Результат работы: 13](#_Toc190869319)

[Вывод: 14](#_Toc190869320)

[Список использованных источников и литературы: 14](#_Toc190869321)

ПРАКТИЧЕСКАЯ РАБОТА 2

Задание

**Часть 1: Теоретический разбор и базовая реализация**

1. Ручная реализация:

* Используя базовые библиотеки (NumPy), создайте нейронную сеть с одним скрытым слоем и двумя выходами.
* Реализуйте прямое распространение (forward pass) с вычислением ошибки на основе квадратичной функции потерь.
* Реализуйте алгоритм обратного распространения ошибки вручную (без использования готовых библиотек).

1. Визуализация процесса обучения:

* Постройте графики изменения ошибки на каждой итерации обучения.

1. Вопросы для обсуждения:

* Почему обратное распространение эффективно для глубоких сетей?
* Какие ограничения имеет данный алгоритм (проблемы с градиентами и их затуханием)?

**Часть 2: Методы оптимизации и эксперименты**

1. Применение методов оптимизации:

* Постройте полносвязную нейронную сеть с двумя скрытыми слоями (можно использовать датасет MNIST или другой).
* Обучите её с использованием следующих оптимизаторов:
* SGD (градиентный спуск)
* Adam
* RMSprop
* Momentum

1. Эксперименты с параметрами:

* Проведите обучение с различными параметрами learning rate (0.01, 0.1, 0.001) и размером мини-выборки (batch size 16, 32, 64).
* Сравните скорость сходимости и итоговую точность на тестовой выборке.

1. Визуализация:

* Постройте графики изменения ошибки и точности на обучающей и тестовой выборках для каждого метода оптимизации.

**Часть 3: Анализ и защита**

1. Выполните анализ результатов:

* Как изменяется скорость сходимости в зависимости от метода оптимизации?
* Какие комбинации learning rate и batch size дали наилучшие результаты?

1. Представьте отчет с основными выводами о применении оптимизаторов в глубоких сетях.

Введение

**Цель**

Понять принципы работы алгоритма обратного распространения ошибки, реализовать его вручную и с использованием библиотек, а также исследовать влияние различных методов оптимизации на процесс обучения нейронной сети.

Шаги выполнения

Этап 1. Общий пример MNIST

1. Импортируйте необходимые библиотеки (TensorFlow/PyTorch, Matplotlib, NumPy).

Листинг 1 – Загрузка данных

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  from tensorflow import keras  from tensorflow.keras import layers  from tensorflow.keras.datasets import mnist |

1. Загрузите датасет MNIST (из библиотек keras.datasets или torchvision.datasets). Разделите его на обучающую и тестовую выборки.

Листинг 2 – Очистка данных

|  |
| --- |
| # 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)).astype('float32') / 255  x\_test = x\_test.reshape((x\_test.shape[0], 28 \* 28)).astype('float32') / 255 |

1. Постройте полносвязную нейронную сеть:

Листинг 3 – Нормализация данных

|  |
| --- |
| # Создание модели  model = keras.Sequential([      layers.Dense(128, activation='relu', input\_shape=(784,)),      layers.Dense(64, activation='relu'),      layers.Dense(10, activation='softmax')  ])  # Компиляция модели  model.compile(optimizer='adam',                loss='sparse\_categorical\_crossentropy',                metrics=['accuracy']) |

1. Обучите модель на обучающей выборке и оцените точность на тестовых данных.

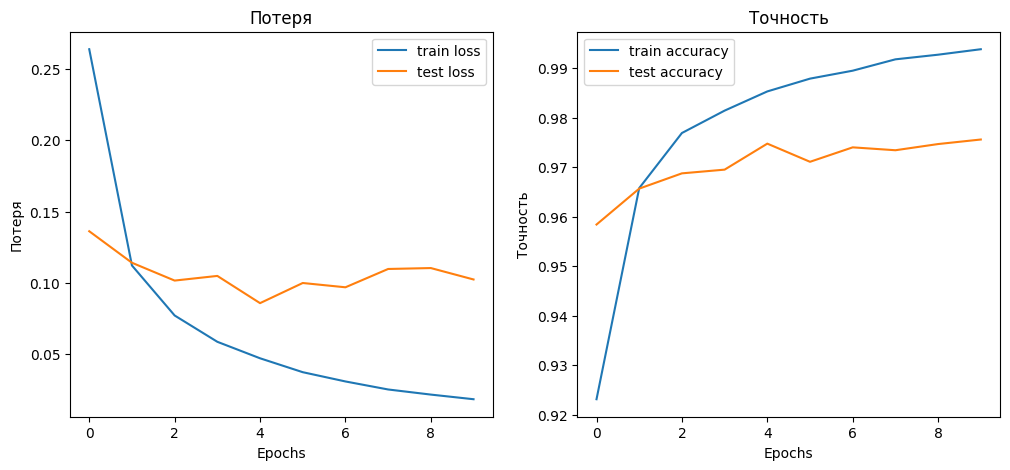
Листинг 4 – Обучение модели

|  |
| --- |
| # Обучение модели  history = model.fit(x\_train, y\_train, epochs=10, batch\_size=32, validation\_split=0.2)  test\_loss, test\_acc = model.evaluate(x\_test, y\_test)  print(f'Test accuracy: {test\_acc:.4f}') |

1. Постройте графики.

Листинг 5 – Построение графиков

|  |
| --- |
| # Графики потерь  plt.figure(figsize=(12, 5))  plt.subplot(1, 2, 1)  plt.plot(history.history['loss'], label='train loss')  plt.plot(history.history['val\_loss'], label='test loss')  plt.title('Потеря')  plt.xlabel('Epochs')  plt.ylabel('Потеря')  plt.legend()  # Графики точности  plt.subplot(1, 2, 2)  plt.plot(history.history['accuracy'], label='train accuracy')  plt.plot(history.history['val\_accuracy'], label='test accuracy')  plt.title('Точность')  plt.xlabel('Epochs')  plt.ylabel('Точность')  plt.legend()  plt.show() |

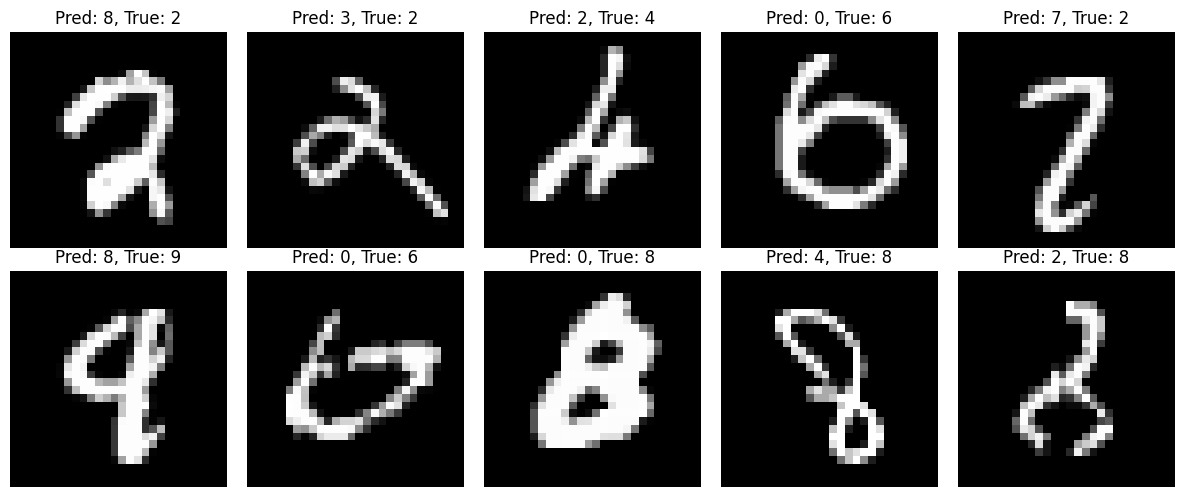


**Рисунок 1 – Графики потерь и точности**

1. Добавьте визуализацию ошибок классификации (например, изображения неверно классифицированных цифр).

Листинг 6 – Добавление изображении

|  |
| --- |
| # Предсказание классов на тестовой выборке  predictions = model.predict(x\_test)  predicted\_classes = np.argmax(predictions, axis=1)  # Найдем неверно классифицированные примеры  incorrect\_indices = np.where(predicted\_classes != y\_test)[0]  # Визуализация первых 10 неверно классифицированных изображений  plt.figure(figsize=(12, 5))  for i, idx in enumerate(incorrect\_indices[:10]):      plt.subplot(2, 5, i + 1)      plt.imshow(x\_test[idx].reshape(28, 28), cmap='gray')      plt.title(f'Pred: {predicted\_classes[idx]}, True: {y\_test[idx]}')      plt.axis('off')  plt.tight\_layout()  plt.show() |



**Рисунок 2 – Визуализация ошибок классификации**

Этап 2. Индивидуальное задание

1. Индивидуальный набор данных из Kaggle и постройте полносвязную нейронную сеть для своей задачи классификации.

Листинг 7 – Загрузка данных

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  from tensorflow import keras  from tensorflow.keras import layers  from tensorflow.keras.datasets import fashion\_mnist  # Загрузка данных Fashion MNIST  (X\_train, y\_train), (X\_test, y\_test) = fashion\_mnist.load\_data()  # Нормализация данных  X\_train = X\_train.astype('float32') / 255.0  X\_test = X\_test.astype('float32') / 255.0  # Преобразование данных из 3D в 2D (объединение высоты и ширины)  X\_train = X\_train.reshape(-1, 28 \* 28)  X\_test = X\_test.reshape(-1, 28 \* 28)  # Преобразование меток в категориальный формат  y\_train = keras.utils.to\_categorical(y\_train, num\_classes=10)  y\_test = keras.utils.to\_categorical(y\_test, num\_classes=10) |

1. Постройте полносвязную нейронную сеть для своей задачи классификации

Листинг 8 – Создание модели без Dropout

|  |
| --- |
| # Создание модели без Dropout  model\_no\_dropout = keras.Sequential()  model\_no\_dropout.add(layers.Dense(256, activation='relu', input\_shape=(28 \* 28,)))  model\_no\_dropout.add(layers.Dense(128, activation='relu'))  model\_no\_dropout.add(layers.Dense(10, activation='softmax'))  # Компиляция модели  model\_no\_dropout.compile(optimizer='adam',                            loss='categorical\_crossentropy',                            metrics=['accuracy'])  # Обучение модели без Dropout  epochs = 20  batch\_size = 32  history\_no\_dropout = model\_no\_dropout.fit(X\_train, y\_train,                                              epochs=epochs,                                              batch\_size=batch\_size,                                              validation\_split=0.2) |

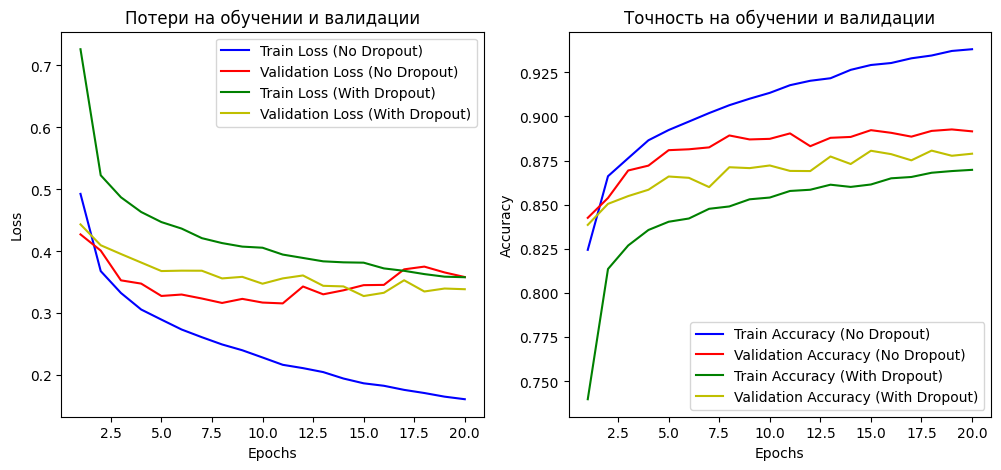
Листинг 9 – Создание модели с Dropout

|  |
| --- |
| # Создание модели с Dropout  model\_with\_dropout = keras.Sequential()  model\_with\_dropout.add(layers.Dense(256, activation='relu', input\_shape=(28 \* 28,)))  model\_with\_dropout.add(layers.Dropout(0.5))  # Добавление Dropout  model\_with\_dropout.add(layers.Dense(128, activation='relu'))  model\_with\_dropout.add(layers.Dropout(0.5))  # Добавление Dropout  model\_with\_dropout.add(layers.Dense(10, activation='softmax'))  # Компиляция модели  model\_with\_dropout.compile(optimizer='adam',                              loss='categorical\_crossentropy',                              metrics=['accuracy'])  # Обучение модели с Dropout  history\_with\_dropout = model\_with\_dropout.fit(X\_train, y\_train,                                                 epochs=epochs,                                                 batch\_size=batch\_size,                                                 validation\_split=0.2) |

1. Визуализируйте процесс обучения с использованием Matplotlib или TensorBoard и Проведите анализ переобучения. Включите Dropout в архитектуру сети и сравните результаты с его использованием и без него:

Листинг 10 – Визуализация

|  |
| --- |
| train\_loss\_no\_dropout = history\_no\_dropout.history['loss'] # Train Loss (No Dropout)  val\_loss\_no\_dropout = history\_no\_dropout.history['val\_loss'] # Validation Loss (No Dropout)  train\_accuracy\_no\_dropout = history\_no\_dropout.history['accuracy'] # Train Accuracy (No Dropout)  val\_accuracy\_no\_dropout = history\_no\_dropout.history['val\_accuracy'] # Validation Accuracy (No Dropout)  train\_loss\_with\_dropout = history\_with\_dropout.history['loss'] # Train Loss (With Dropout)  val\_loss\_with\_dropout = history\_with\_dropout.history['val\_loss'] # Validation Loss (With Dropout)  train\_accuracy\_with\_dropout = history\_with\_dropout.history['accuracy'] # Train Accuracy (With Dropout)  val\_accuracy\_with\_dropout = history\_with\_dropout.history['val\_accuracy'] # Validation Accuracy (With Dropout)  import matplotlib.pyplot as plt  epochs = range(1, len(train\_loss\_no\_dropout) + 1)  plt.figure(figsize=(12, 5))  # График потерь  plt.subplot(1, 2, 1)  plt.plot(epochs, train\_loss\_no\_dropout, 'b', label='Train Loss (No Dropout)')  plt.plot(epochs, val\_loss\_no\_dropout, 'r', label='Validation Loss (No Dropout)')  plt.plot(epochs, train\_loss\_with\_dropout, 'g', label='Train Loss (With Dropout)')  plt.plot(epochs, val\_loss\_with\_dropout, 'y', label='Validation Loss (With Dropout)')  plt.xlabel('Epochs')  plt.ylabel('Loss')  plt.legend()  plt.title('Потери на обучении и валидации')  # График точности  plt.subplot(1, 2, 2)  plt.plot(epochs, train\_accuracy\_no\_dropout, 'b', label='Train Accuracy (No Dropout)')  plt.plot(epochs, val\_accuracy\_no\_dropout, 'r', label='Validation Accuracy (No Dropout)')  plt.plot(epochs, train\_accuracy\_with\_dropout, 'g', label='Train Accuracy (With Dropout)')  plt.plot(epochs, val\_accuracy\_with\_dropout, 'y', label='Validation Accuracy (With Dropout)')  plt.xlabel('Epochs')  plt.ylabel('Accuracy')  plt.legend()  plt.title('Точность на обучении и валидации')  plt.show() |



**Рисунок 3 – Графики потерь и точности**

Этап 3. Сравнительный анализ и защита

1. Сравните модели с разными функциями активации

Листинг 11 – Сравнение модели

|  |
| --- |
| activation\_functions = ['relu', 'sigmoid', 'tanh']  histories = {}  models = {}  for activation in activation\_functions:  model = keras.Sequential()  model.add(layers.Dense(256, activation=activation, input\_shape=(28 \* 28,)))  model.add(layers.Dense(128, activation=activation))  model.add(layers.Dense(10, activation='softmax'))  model.compile(optimizer='adam',  loss='categorical\_crossentropy',  metrics=['accuracy'])  history = model.fit(X\_train, y\_train,  epochs=20,  batch\_size=32,  validation\_split=0.2,  verbose=0)  histories[activation] = history  models[activation] = model |

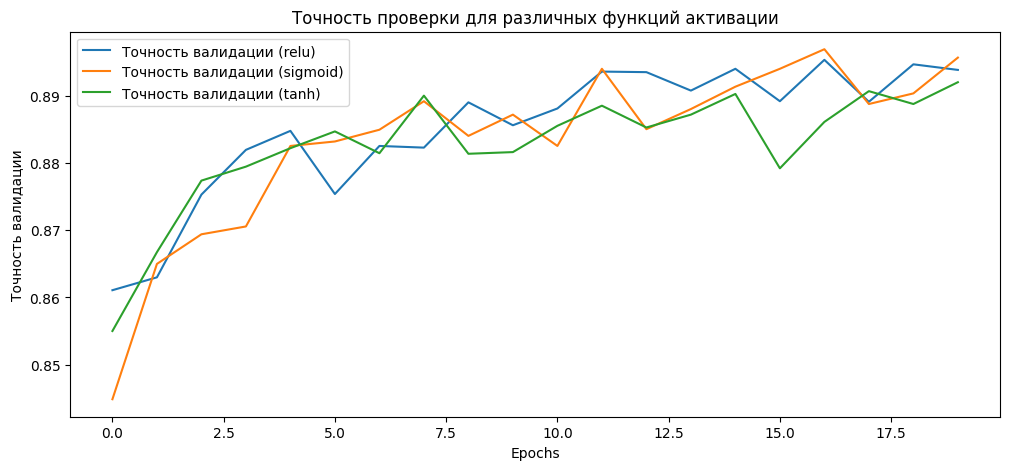
1. Выполните защиту результатов

Листинг 12 – Метрики

|  |
| --- |
| from sklearn.metrics import classification\_report  from sklearn.metrics import f1\_score  for activation in activation\_functions:  y\_pred = models[activation].predict(X\_test)  y\_pred\_classes = np.argmax(y\_pred, axis=1)  y\_true\_classes = np.argmax(y\_test, axis=1)  accuracy = np.mean(y\_pred\_classes == y\_true\_classes)  f1 = f1\_score(y\_true\_classes, y\_pred\_classes, average='weighted')  print(f'Activation: {activation}')  print(f'Accuracy: {accuracy:.4f}')  print(f'F1 Score: {f1:.4f}\n') |

Листинг 13 – Сравнительные графики потерь и точности

|  |
| --- |
| plt.figure(figsize=(12, 5))  for activation in activation\_functions:      plt.plot(histories[activation].history['val\_accuracy'], label=f'Точность валидации ({activation})')  plt.title('Точность проверки для различных функций активации')  plt.xlabel('Epochs')  plt.ylabel('Точность валидации')  plt.legend()  plt.show() |



**Рисунок 4 – Сравнительные графики потерь и точности**

Результат работы:

Данную работу можете увидеть в блокноте Google Colab:

<https://colab.research.google.com/drive/1HO9TiEkhZbzCSMmfBr4VX0nh2X1JIviK?usp=sharing>

Вывод:

Можно сделать вывод, что выбор функций активации, гиперпараметров и методов регуляризации (таких как Dropout) существенно влияет на производительность нейронной сети. ReLU является наиболее эффективной функцией активации для скрытых слоев, а Dropout помогает предотвратить переобучение.

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