Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5
по дисциплине ««Основы систем ИИ»

Тема Основы работы с ИНС

Студент Батуев А.Г.

Группа ИУ7-36Б

Преподаватели Строганов Ю.В.

Москва, 2025

Содержание

ВВЕДЕНИЕ

Цель: классифицировать данные из mnist dataset, используя нейросетевой подход с ReLU функцией активации и KL Divergence функцией потерь.

Задачи:

— определить состояние переобучения и недообучения для различного соотношения обучающей и тестовой выборок.

— использовать различное количество скрытых слоёв.

— по неравенству Чебышёва рассчитать аналитически необходимый размер обучающей выборки.

1 Аналитическая часть

Нейронные сети (*Artificial Neural Networks, ANN*) представляют собой одну из ключевых технологий в области машинного обучения, которая имитирует принципы работы биологических нейронов. Эти модели состоят из набора взаимосвязанных узлов (*нейронов*), организованных в слои. Нейроны в слоях применяют к данным последовательность линейных и нелинейных преобразований, что позволяет выявлять сложные зависимости в данных и решать задачи, такие как классификация, регрессия и прогнозирование.

Процесс обучения нейронной сети включает несколько этапов. Одним из важнейших понятий здесь является эпоха (*epoch*), которая соответствует одному полному прохождению всех тренировочных данных через модель. На каждой эпохе веса модели обновляются с целью минимизации ошибки между предсказанными и истинными значениями. Слишком малое количество эпох может привести к недообучению (*underfitting*), а слишком большое — к переобучению (*overfitting*).

1.1 Классификация данных и архитектура нейронной сети

Для решения задачи классификации, нейронная сеть обрабатывает входные данные через последовательность слоёв. Каждый слой играет свою роль:

— входной слой преобразует данные в формат, пригодный для дальнейших вычислений. Например, для изображений это может быть нормализация пикселей и их представление в виде плоского вектора.

— скрытые слои (*hidden layers*) выполняют извлечение признаков, которые помогают выявить сложные закономерности в данных.

— выходной слой (*output layer*) определяет вероятности принадлежности объекта к каждому из классов с использованием функции активации, например, *softmax*.

Для оценки качества классификации используется метрика точности (*accuracy*), определяемая как доля правильных предсказаний от общего числа примеров:

$Accuracy = \frac{\text{Количество правильных предсказаний}}{\text{Общее количество примеров}}$ [10]

Эта метрика показывает, насколько хорошо модель справляется с задачей классификации.

1.2 MNIST dataset

MNIST (*Modified National Institute of Standards and Technology*) — это один из наиболее известных и часто используемых наборов данных для задач классификации изображений. Датасет включает в себя:

— 60,000 тренировочных изображений и 10,000 тестовых изображений.

— изображения размером 28×28, представляющие рукописные цифры от 0 до 9.

Каждое изображение ассоциируется с меткой класса, что делает MNIST подходящим для

обучения моделей и проверки их качества. Его простота и стандартизированность позволяют использовать датасет для тестирования различных подходов к классификации.

1.3 Функция активации и функция потерь

Одним из важнейших компонентов нейронной сети является функция активации, определяющая, как нейроны реагируют на входные данные. В данном исследовании была использована функция активации ReLU (*Rectified Linear Unit*), которая задаётся формулой:

$f(x) = \max(0, x)$ [11]

ReLU популярна благодаря своей способности справляться с эффектом затухающих градиентов (*vanishing gradients*), что делает обучение глубоких моделей более эффективным. Однако у неё есть недостаток: некоторые нейроны могут стать "мёртвыми то есть их выход всегда будет равен нулю.

Для задачи классификации вероятностных распределений была применена функция потерь KL Divergence (*Kullback-Leibler Divergence*). Она измеряет, насколько одно вероятностное распределение P (истинные метки) отличается от другого распределения Q (предсказания модели):

$D_{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$ [12]

Эта мера помогает оценивать, насколько точно модель предсказывает вероятности классов, что особенно полезно при использовании функции *softmax* в выходном слое.

Вывод

Нейронные сети являются мощным инструментом для решения задач классификации, таких как распознавание цифр. Использование функции активации ReLU способствует эффективному обучению модели, а KL Divergence позволяет точно оценивать её предсказания.

2 Конструкторская часть

Разработанная программа предназначена для классификации изображений рукописных цифр из набора данных MNIST. Её работа основана на использовании нейронных сетей, где качество классификации исследуется в зависимости от архитектуры сети и соотношения обучающей и тестовой выборок.

2.1 Принцип действия программы

Программа состоит из нескольких последовательных этапов. На первом этапе данные предобрабатываются: изображения нормализуются путём деления значений интенсивности пикселей на 255, что приводит их в диапазон $[0, 1]$. Метки классов преобразуются в формат one-hot encoding, который представляет каждый класс в виде вектора, где значение 1 соответствует истинному классу, а остальные элементы равны 0.

Далее строится архитектура нейронной сети. На вход подаются изображения размером 28×28 с одним каналом, которые сначала преобразуются в плоский вектор с помощью слоя Flatten. Затем к сети добавляются скрытые слои, состоящие из 128 нейронов каждый.

В качестве функции активации используется ReLU (см. 1.3). ReLU минимизирует проблему исчезающих градиентов, что особенно важно при обучении глубоких сетей. Выходной слой сети имеет 10 нейронов с функцией активации softmax, что позволяет интерпретировать выход модели как вероятности принадлежности к каждому из десяти классов:

$P(y_i \mid x) = \frac{\exp(z_i)}{\sum_{j=1}^{10} \exp(z_j)}$ [13] ,где z_i — значение на выходе i-го нейрона.

Обучение модели осуществляется с использованием функции потерь KL-дивергенции (см. 1.3)

2.2 Результаты и их интерпретация

Для исследования влияния архитектуры сети и размера обучающей выборки были проведены исследования с различным числом скрытых слоёв (0, 1, 5) и соотношением обучающей и тестовой выборок (от $10\% - 90\%$ до $90\% - 10\%$). В процессе обучения фиксировались значения точности и потерь на обучающей, валидационной и тестовой выборках.

Также была проведена аналитическая оценка минимально необходимого размера обучающей выборки на основе неравенства Чебышёва: $n \geq \frac{\sigma^2}{\epsilon^2 \delta}$ [14]

Вывод

Разработана программа для классификации изображений из набора MNIST с использованием нейронных сетей. Реализована возможность исследования влияния количества скрытых слоёв и соотношения обучающей и тестовой выборок на качество классификации. Программа использует ReLU-активацию, функцию потерь KL-дивергенции и метод оптимизации Adam. Также предусмотрен аналитический расчёт минимального размера обучающей выборки по нера-

венству Чебышёва. Итоги создают основу для дальнейших исследований и анализа.

3 Технологическая часть

3.1 Средства написания программы

Для реализации программного обеспечения были использованы следующие средства:

— среда разработки PyCharm 2023 community edition [6]

— язык разработки Python 3.10 [7]

Используемые библиотеки:

— numpy — для быстрой работы с массивами, сохранения промежуточных данных и связи их с другими библиотеками [1]

— matplotlib — для отображения графиков [2]

— pathlib — доступ к файлам системы [5]

— tensorflow — для реализации нейросетей [4]

— sklearn — для работы с данными [3]

3.2 Функции

Листинг 3.1 — Нормализация данных

```
def normalize_data(image, label):
    image = tf.cast(image, tf.float32) / 255.0
    label = to_categorical(label, num_classes=10)
    return image, label
```

Листинг 3.2 — Разделение данных

```
def split_dataset(dataset, train_size):
    dataset_list = list(dataset.unbatch().as_numpy_iterator())
    images, labels = zip(*dataset_list)
    images = np.array(images)
    labels = np.array(labels)

    x_train, x_val, y_train, y_val = train_test_split(images, labels,
        train_size=train_size)
    train_ds = tf.data.Dataset.from_tensor_slices((x_train, y_train)).
        batch(BATCH_SIZE)
    val_ds = tf.data.Dataset.from_tensor_slices((x_val, y_val)).batch(
        BATCH_SIZE)
    return train_ds, val_ds
```

Листинг 3.3 — Создание модели

```
def build_model(hidden_layers):
    model = models.Sequential()
```

```
    model.add(layers.Input(shape=(28, 28, 1)))
    model.add(layers.Flatten())

    for _ in range(hidden_layers):
        model.add(layers.Dense(128, activation='relu'))

    model.add(layers.Dense(10, activation='softmax'))
    return model
```

Листинг 3.4 — Обучение и оценка модели

```
def train_and_evaluate(train_ds, val_ds, test_ds, hidden_layers):
    model = build_model(hidden_layers)
    model.compile(optimizer='adam', loss=KLDivergence(), metrics=['
        accuracy'])
    history = model.fit(train_ds, validation_data=val_ds, epochs=
        EPOCHS, verbose=0)
    test_loss, test_acc = model.evaluate(test_ds, verbose=0)
    return history, test_loss, test_acc
```

Листинг 3.5 — Функция расчета размера выборки по неравенству Чебышева

```
def chebyshev_sample_size(epsilon, delta, sample_variance):
    if epsilon <= 0 or delta <= 0 or delta >= 1:
        raise ValueError("Epsilon and delta must be greater than zero
            and delta must be less than one.")

    n = math.ceil(sample_variance / (epsilon ** 2 * delta))
    return n
```

Листинг 3.6 — Визуализизация результатов обучения

```
def plot_results(results):
    for hidden_layers in HIDDEN_LAYER_CONFIGS:
        plt.figure(figsize=(10, 6))
        for split in DATA_SPLITS:
            res = [r for r in results if r['hidden_layers'] ==
                hidden_layers and r['split'] == split]
            if res:
                acc = res[0]['history'].history['accuracy']
                val_acc = res[0]['history'].history['val_accuracy']
                plt.plot(acc, label=f'Train Acc (split {split})')
                plt.plot(val_acc, linestyle='--', label=f'Val Acc (
                    split {split})')
```

```
        title = f'Results for {hidden_layers} hidden layers'
        output_file = IMG_DIR / f'{title}.png'

        plt.title(title)
        plt.xlabel('Epoch')
        plt.ylabel('Accuracy')
        plt.legend()
        plt.savefig(output_file)
```

Вывод

Были разработаны методы исследования влияние архитектуры нейронной сети и размера обучающей выборки на эффективность классификации.

4 Исследовательская часть

Цель исследования — классифицировать данные из mnist dataset, используя нейросетевой подход с ReLU функцией активации и KL Divergence функцией потерь.

4.1 Оборудование

Характеритстики ноутбука:

— процессор intel-core i5-12500H [9]

— ОЗУ 16 Гб DDR4

— ОС Windows 11 [8]

4.2 Результаты исследования

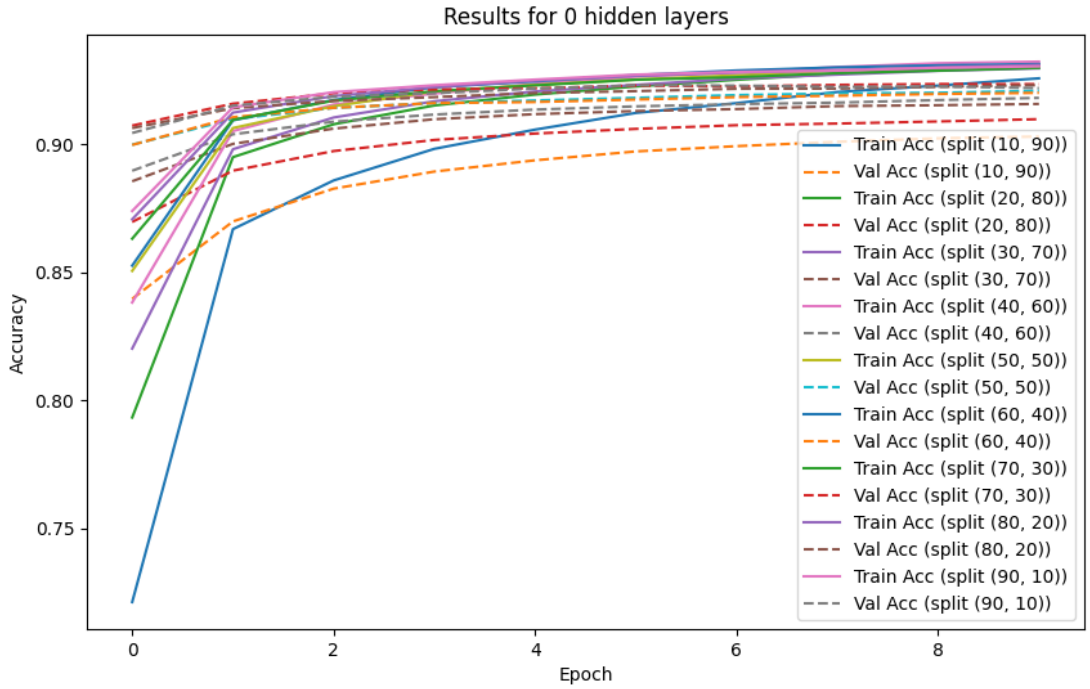

Рисунок 4.1 — Влияние соотношения обучающей и валидационной выборки на обучение модели без скрытых слоёв

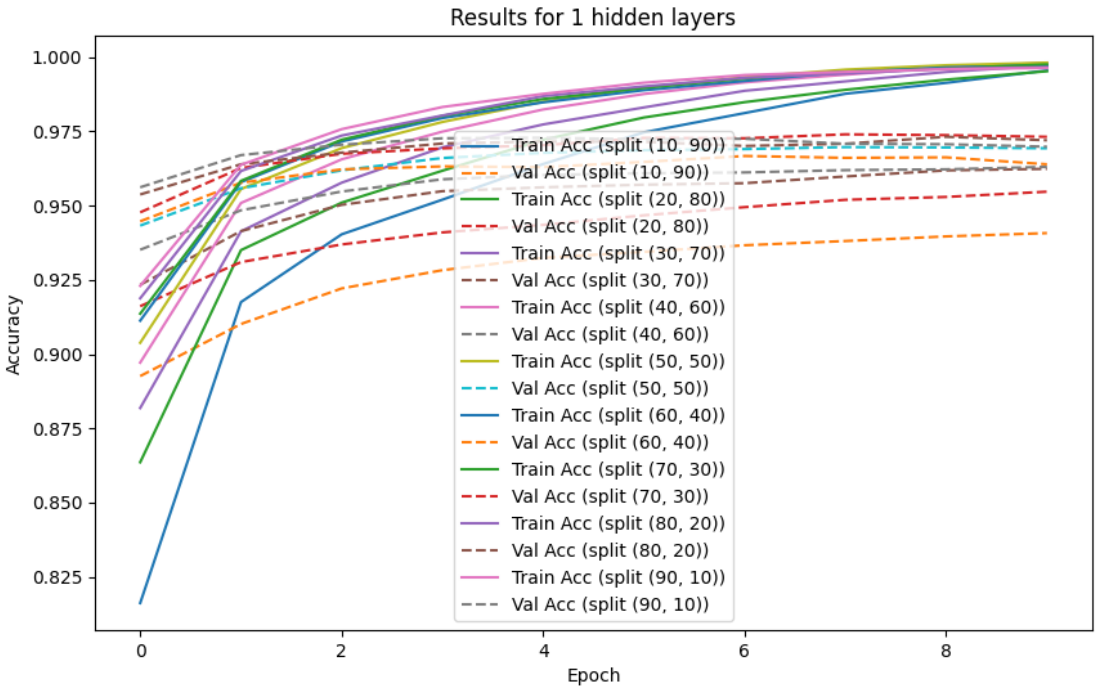

Рисунок 4.2 — Влияние соотношения обучающей и валидационной выборки на обучение модели с 1 скрытым слоем

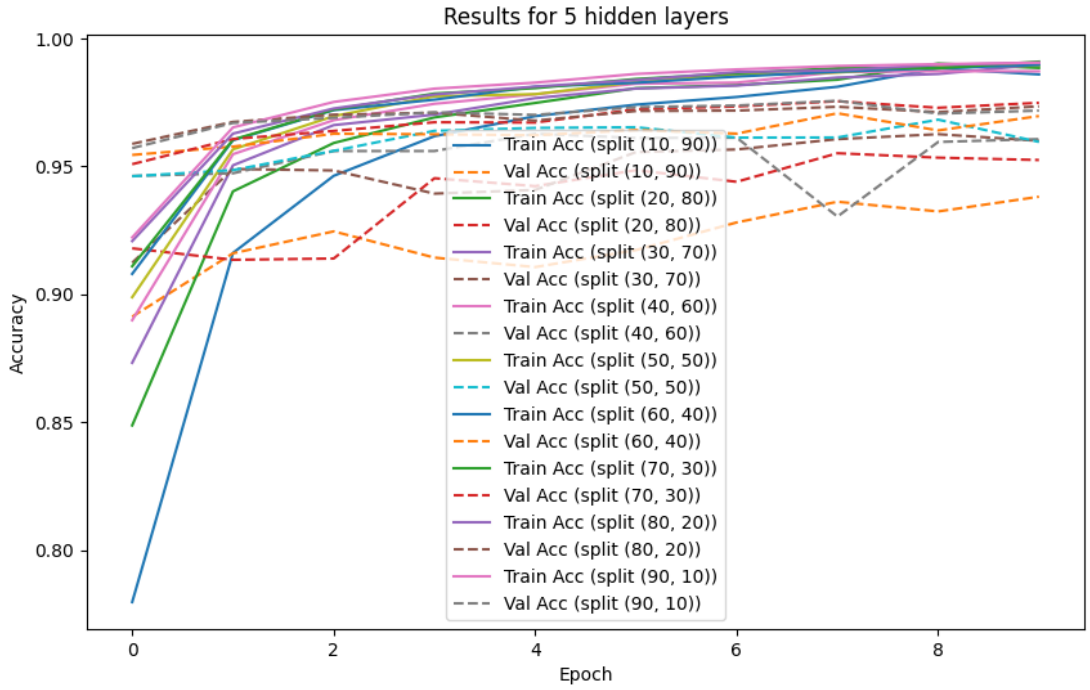

Рисунок 4.3 — Влияние соотношения обучающей и валидационной выборки на обучение модели с 5 скрытыми слоями

Таблица 4.1 — Размер выборки по Чебышёву для разных разбиений трен./валид. выборки

Соотношение трен./валид. выборки	Чебышёв Размер
(10, 90)	48
(20, 80)	42
(30, 70)	46
(40, 60)	42
(50, 50)	45
(60, 40)	44
(70, 30)	47
(80, 20)	44
(90, 10)	45

Таблица 4.2 — Результаты тестирования для разных разбиений и количества скрытых слоёв

Соотношение трен./валид. выборки	Скрытые слои	Потери	Точность
(10, 90)	0	0.3216	0.9096
(10, 90)	1	0.1931	0.9430
(10, 90)	5	0.2894	0.9429
(20, 80)	0	0.2943	0.9172
(20, 80)	1	0.1499	0.9568
(20, 80)	5	0.2032	0.9521
(30, 70)	0	0.2860	0.9213
(30, 70)	1	0.1250	0.9673
(30, 70)	5	0.1629	0.9622
(40, 60)	0	0.2777	0.9237
(40, 60)	1	0.1230	0.9665
(40, 60)	5	0.1497	0.9649
(50, 50)	0	0.2716	0.9234
(50, 50)	1	0.1175	0.9712
(50, 50)	5	0.1991	0.9576
(60, 40)	0	0.2750	0.9241
(60, 40)	1	0.1270	0.9644
(60, 40)	5	0.1325	0.9701
(70, 30)	0	0.2704	0.9249
(70, 30)	1	0.0997	0.9752
(70, 30)	5	0.1270	0.9742
(80, 20)	0	0.2700	0.9253
(80, 20)	1	0.1064	0.9728
(80, 20)	5	0.1070	0.9764
(90, 10)	0	0.2681	0.9285
(90, 10)	1	0.1084	0.9721
(90, 10)	5	0.1214	0.9740

Вывод

Проведённые исследования показали, что увеличение количества скрытых слоёв улучшает точность и снижает потери на тестовой выборке, однако прирост качества становится

незначительным при переходе от одного скрытого слоя к пяти. Оптимальной в данном случае можно считать архитектуру с одним скрытым слоем, которая обеспечивает баланс между качеством и сложностью модели.

Кроме того, распределение данных между обучающей и валидационной выборками оказывает влияние на результаты. Более высокие значения точности достигаются при большем объёме данных для обучения (Split: 90/10 и 80/20).

ЗАКЛЮЧЕНИЕ

В рамках данного исследования была выполнена классификация данных из MNIST датасета с использованием нейросетевого подхода, функции активации ReLU и функции потерь KL Divergence. Исследование было направлено на достижение поставленной цели — классификации данных

Для оценки состояния переобучения и недообучения проведён анализ точности на обучающих и тестовых выборках при различных пропорциях их разбиения. В рамках исследования были использованы следующие соотношения: от 10:90 до 90:10. Наблюдалось, что увеличение размера обучающей выборки приводит к увеличению точности модели на обучающих данных. При этом важно соблюдать баланс между обучающей и тестовой выборками, чтобы не приводить модель к недообучению или переобучению.

Увеличение количества скрытых слоев увеличивает точность модели на обучающих данных. Но здесь, также важно соблюдать баланс между количеством скрытых слоев, потому что она может вести к ненужному увеличению сложности модели.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. NumPy Documentation URL: https://numpy.org/doc/ (Дата обращения 07.01.2025)
2. Matplotlib 3.9.2 documentation URL: https://matplotlib.org/stable/index.html (Дата обращения 07.01.2025)
3. Документация sklearn URL: https://scikit-learn.org/stable/ (Дата обращения 07.01.2025)
4. Документация Tensorflow URL: https://www.tensorflow.org/api_docs (Дата обращения 07.01.2025)
5. Документация Pathlib https://docs.python.org/3/library/pathlib.html (Дата обращения 07.01.2025)
6. Среда разработки Pycharm 2023 Community edition URL: https://www.jetbrains.com/ru-ru/pycharm/download/other.html
7. Язык разработки Python 3.10 URL: https://www.python.org/downloads/release/python-3100/
8. ОС Windows 11 URL: https://www.microsoft.com/ru-ru/software-download/windows11
9. Intel i5-12500H URL: https://www.intel.com/content/www/us/en/products/sku/96141/intel-core-i512500h-processor-18m-cache-up-to-4-50-ghz/specifications.html?wapkw=12500h
10. Введение в архитектуру нейронных сетей URL: https://cyberleninka.ru/article/n/vvedenie-v-arhitekturu-neyronnyh-setey (Дата обращения 07.01.2025)
11. Функия активации URL: https://cyberleninka.ru/article/n/sravnenie-sposobov-vychisleniya-proizvodnoy-aktivatsii-vypryamitel-pri-obuchenii-neyronnoy-seti (Дата обращения 07.01.2025)
12. Функция потерь URL: https://cyberleninka.ru/article/n/why-deep-learning-methods-use-kl-divergence-instead-of-least-squares-a-possible-pedagogical-explanation (Дата обращения 07.01.2025)
13. Функция активации softmax URL: https://cyberleninka.ru/article/n/ispolzovanie-modeli-neyronnoy-seti-glubokogo-obucheniya-dlya-resheniya-problem-klassifikatsii-nezhelatelnogo-kontenta-v-sotsialnyh (Дата обращения 07.01.2025)
14. Неравенство Чебышева URL: https://cyberleninka.ru/article/n/k-obosnovaniyu-metoda-ustoychivogo-otsenivaniya-posredstvom-neravenstva-chebyshyova (Дата обращения 07.01.2025)