Цифриум

Технологическая карта

Умный код на Python. Базовый уровень

Модуль 4 | Тема 1 | Урок 1

Основы нейронных сетей: архитектура, функции активации, обучение, работа в команде

Уважаемый коллега, обратите внимание!

Учёт присутствия ребёнка на уроке ведётся только по цифровому следу. Каждый ученик из группы на уроке должен передать цифровой след на нашей платформе.

Вы отвечаете за передачу следа учеником на нашей платформе во время урока.

Передавать цифровой след нужно в личном кабинете ученика.

Цифровой след засчитывается по итогу выполнения учеником трёх задач на платформе в ходе урока.

Обращаем внимание — в задачах с отправкой ответа в виде файла наша платформа поддерживает следующие форматы: .pdf, .doc, .png, .jpeg, .xlsx, mp3, .py, .ipynb, .txt, .csv, .json, .xml, .sb3, .ino, .hex.

Контролируйте время урока, чтобы не пропустить момент, когда ученики должны залогиниться в личном кабинете и перейти к решению задач на платформе.

Проследите, чтобы каждый ученик залогинился в личном кабинете и выполнил 3 задачи на платформе. В противном случае, Вам не будет засчитано, что вы провели урок, занятие не будет оплачено, а ученику не будет выставлена отметка о присутствии на уроке.

Просим вас в ходе 45 минут урока проконтролировать, чтобы:

1. Каждый из учеников залогинился в личном кабинете.
2. Каждый ученик в разделе **«Решить задания»** отправил решения по 3 задачам — «Практическое задание 1», «Практическое задание 2» и «Практическое задание 3».

О занятии

Краткое содержание занятия:

1. Узнаете, как устроен искусственный нейрон и что такое слои.
2. Разберётесь, что делают функции активации и зачем они нужны.
3. Поймёте, как нейросеть обучается через градиентный спуск.
4. Познакомитесь с датасетом MNIST.
5. Напишете простую нейросеть на PyTorch.
6. Обучите модель распознавать рукописные цифры.

Цель:

Познакомить с внутренней структурой нейросетей и принципами их обучения.

Задачи:

1. Выявить основные компоненты нейрона и их роль в сети.
2. Рассмотреть распространённые функции активации (ReLU, Sigmoid, Tanh).
3. Понять алгоритм обучения через прямой и обратный проходы на примере PyTorch.

Термины

1. **Нейрон —** базовый элемент искусственной нейронной сети, моделирующий работу биологического нейрона. Получает на вход взвешенную сумму сигналов, пропускает результат через функцию активации и передаёт дальше.

2. **Слой —** группа нейронов, выполняющих одну и ту же операцию: входной слой принимает данные, скрытые слои обрабатывают,
3. а выходной слой выдаёт финальный результат.

О занятии

4. **Функция активации —** нелинейная функция, применяемая к взвешенной сумме входов нейрона. Задаёт выход нейрона и позволяет сети моделировать сложные зависимости. Популярные примеры: ReLU, Sigmoid, Tanh.

5. **Ошибка (loss) —** мера отличия выходного сигнала сети от целевого значения. Вычисляется по всей выборке или мини-пакету и минимизируется в процессе обучения.

6. **Градиентный спуск —** алгоритм оптимизации, при котором веса сети итеративно корректируются в направлении, противоположном градиенту функции ошибки по этим весам, чтобы минимизировать ошибку.

7. **Эпоха —** один полный проход по всему набору обучающих данных. После каждой эпохи градиентный спуск обновляе веса на основе накопленного градиента.

8. **MNIST** — стандартный датасет из 70 000 изображений рукописных цифр (от 0 до 9), часто используемый для первичного тестирования и обучения моделей компьютерного зрения.

9. **PyTorch —** библиотека машинного обучения на Python для создания и обучения нейронных сетей.

10. **CrossEntropyLoss —** функция потерь для задач классификации, измеряющая расхождение между предсказанным и истинным распределением классов.

11. **Optimizer —** алгоритм оптимизации (например, SGD или Adam), который обновляет веса модели для минимизации функции потерь.

Необходимые материалы:

1. Доступ к платформе для выполнения практических заданий.

Темы и время

ЭТАП	ВРЕМЯ
Приветствие	5 мин.
Теория	25 мин.
Итоги занятия	5 мин.
Получение фидбека. Рефлексия. Проверка знаний. Домашнее задание	5 мин.
Вопрос-ответ	5 мин.
Итого	45 мин.

Ход занятия

Номер слайда	Пояснение к слайду
1	Приветствие: Добро пожаловать на урок по основам нейронных сетей! Сегодня мы погрузимся в удивительный мир искусственного интеллекта и узнаем, как работают нейросети — технологии, которые распознают лица, переводят тексты, играют в шахматы и управляют автомобилями. Дополнительно: Нейронные сети — это не магия, а математика и программирование. Вы уже умеете работать с готовыми моделями через API. Сегодня вы заглянете «под капот» и поймёте, как эти модели создаются и обучаются с нуля.
2	**Титульный слайд** Тема урока: «Основы нейронных сетей: архитектура, функции активации, обучение, работа в команде»
3	**На прошлом занятии вы:** Работа с GPT API: отправка запросов, обработка JSON-ответов Async/await и fetch-запросы для асинхронного взаимодействия с сервером Создание веб-интерфейса для взаимодействия с моделью Дополнительно: Вы научились использовать нейросети как инструмент. Сегодня вы станете создателями: напишете свою простую нейросеть, которая будет учиться распознавать рукописные цифры. Это первый шаг к пониманию глубокого обучения.
4	**Сегодня на занятии вы:** Устройство искусственного нейрона: входы, веса, сумма, функция активации Структуру нейросети: входной, скрытые и выходной слои Функции активации: ReLU, Sigmoid, Tanh, Softmax — зачем они нужны и как работают Процесс обучения: прямой проход, функция потерь, обратное распространение, градиентный спуск

Ход занятия

	Датасет MNIST: что это, как устроен, зачем используется Написание простой нейросети на PyTorch: архитектура, обучение, оценка результата
5	**Цель и задачи урока** Цель: Познакомить с внутренней структурой нейросетей и принципами их обучения, чтобы вы понимали, как ИИ «думает» и учится. Задачи: Выявить основные компоненты нейрона и их роль в обработке информации Рассмотреть функции активации (ReLU, Sigmoid, Tanh, Softmax): математика, применение, плюсы и минусы Понять алгоритм обучения через прямой и обратный проходы, функцию потерь и градиентный спуск Научиться создавать простую нейросеть на PyTorch и интерпретировать результаты обучения Дополнительно: После урока вы сможете самостоятельно модифицировать архитектуру сети, подбирать параметры обучения и оценивать качество модели. Это база для дальнейшего изучения компьютерного зрения, обработки текста и других областей ИИ.
6	**Что такое нейрон?** Искусственный нейрон — это маленькая математическая машина, которая принимает решения на основе чисел. Давай разберём её на примере, который ты точно поймёшь. Представь, что ты решаешь, пойти ли гулять с друзьями. На твоё решение влияют несколько факторов: ● Хорошая ли погода? (дождь или солнце) ● Сделал ли ты домашку? ● Есть ли у тебя свободное время? Нейрон делает почти то же самое, только с числами: 1. Входы (x) — это числа, которые описывают ситуацию. Например: погода = 1 (солнечно), домашка = 0 (не сделана), время = 1 (есть). 2. Веса (w) — это «важность» каждого фактора. Если для тебя погода очень важна, у неё будет большой

Ход занятия

	вес, например 0.8. Если домашка не так важна — вес 0.3. 3. Сумматор — нейрон умножает каждый вход на его вес и складывает результаты: (1 × 0.8) + (0 × 0.3) + (1 × 0.5) = 1.3 4. Смещение (bias) — это как личное предпочтение. Даже если все факторы «против», ты всё равно можешь захотеть пойти гулять. Смещение добавляет небольшой бонус к сумме. 5. Функция активации — это «переключатель», который решает, что делать с результатом. Например: если сумма больше 1 — выдаём «Да, иду!», если меньше — «Остаюсь дома». Или функция может выдать не просто да/нет, а уверенность в процентах. 6. Выход (y) — итоговое решение нейрона, которое передаётся дальше в сеть. Проще говоря: нейрон — это формула, которая берёт входные данные, взвешивает их по важности, добавляет личное смещение и через правило-переключатель выдаёт ответ. Дополнительно: Без функции активации нейрон выполнял бы только линейные преобразования, и вся сеть сводилась бы к одной линейной функции. Именно нелинейность позволяет нейросетям обучаться сложным закономерностям. Визуально нейрон можно представить как узел, принимающий стрелки-входы, умножающий их на веса, суммирующий и передающий результат дальше.
7	**Что такое слой?** Слой — это группа нейронов, обрабатывающих информацию на одном этапе преобразования данных. Типы слоёв: Входной слой: получает исходные данные. Для MNIST: 28×28 = 784 нейрона, каждый соответствует яркости одного пикселя (0–1). Не выполняет вычислений, только передаёт данные. Скрытые слои: промежуточные слои, которые извлекают признаки из данных. Первый скрытый слой может распознавать края, второй — формы, третий — части цифр.

Ход занятия

	Чем больше слоёв, тем «глубже» сеть (отсюда термин «глубокое обучение»). Выходной слой: выдаёт итоговый результат. Для классификации цифр: 10 нейронов, каждый соответствует вероятности принадлежности к классу 0–9. Дополнительно: Полносвязный слой (Fully Connected, Linear) — каждый нейрон слоя соединён со всеми нейронами предыдущего слоя. Это простейший тип слоя. Существуют также свёрточные слои (для изображений), рекуррентные (для последовательностей) и другие, но сегодня мы работаем с полносвязными.
8	**Функции активации** Без функций активации нейросеть, сколько бы слоёв ни имела, могла бы быть сведена к одной линейной операции. Функции активации добавляют нелинейность, позволяя сети обучаться сложным, нелинейным зависимостям. Популярные функции активации: ReLU (Rectified Linear Unit): $f(x) = \max(0, x)$ Если вход > 0, пропускает его; если ≤ 0, выдаёт 0 Плюсы: простая, быстрая, не затухает градиент при положительных значениях Минусы: «мёртвые нейроны» — если нейрон всегда получает отрицательный вход, он перестаёт обучаться Применение: стандарт для скрытых слоёв Sigmoid: $f(x) = 1 / (1 + e^{-x})$ Сжимает вход в диапазон (0, 1) Плюсы: интерпретируемый выход как вероятность Минусы: затухание градиента при больших |x|, вычисления медленнее Применение: выходной слой для бинарной классификации Tanh (гиперболический тангенс): $f(x) = (e^{x} – e^{-x}) / (e^{x} + e^{-x})$ Сжимает вход в диапазон (–1, 1), центрирована относительно нуля Плюсы: сильнее градиент, чем у sigmoid, лучше сходимость Минусы: всё ещё подвержена затуханию градиента Применение: скрытые слои в рекуррентных сетях Softmax: $f(x_i) = e^{x_i} / \Sigma\square\, e^{x}\square$ Преобразует вектор чисел в распределение вероятностей (сумма = 1) Применение: выходной слой для многоклассовой классификации (например, 10 цифр)

Ход занятия

	Дополнительно: Выбор функции активации влияет на скорость обучения и качество модели. Современный стандарт: ReLU для скрытых слоёв, Softmax + CrossEntropy для выхода в задачах классификации.
9	**Как сеть учится?** Как сеть учится? Процесс обучения пошагово Обучение нейросети — это настройка весов и смещений так, чтобы сеть минимизировала ошибку предсказаний. Шаг 1: Прямой проход (Forward Pass) Входные данные подаются на сеть Каждый слой последовательно преобразует данные На выходе получается предсказание (например, вероятности для цифр 0–9) Шаг 2: Вычисление ошибки (Loss Function) Сравниваем предсказание сети с правильным ответом (меткой) Функция потерь (Loss) количественно измеряет ошибку Для классификации: CrossEntropyLoss = $-\Sigma\, y_i \cdot \log(p_i)$, где y — истинная метка, p — предсказанная вероятность Шаг 3: Обратное распространение ошибки (Backpropagation) Алгоритм, который вычисляет градиент функции потерь по каждому весу сети Использует цепное правило дифференцирования (chain rule) Показывает, как изменение каждого веса влияет на общую ошибку Шаг 4: Обновление весов (Optimizer / Gradient Descent) Градиентный спуск: двигаем веса в направлении, противоположном градиенту, чтобы уменьшить ошибку Формула: $w_{new} = w_{old} - \eta \cdot \nabla L$, где η — скорость обучения (learning rate), ∇L — градиент потерь Оптимизаторы (Adam, SGD) автоматически управляют этим процессом, добавляя моменты, адаптивные шаги и другие улучшения Дополнительно: Один полный проход по всем данным обучения называется эпохой (epoch). Обычно сеть обучают на 5–50 эпохах. Важно разделять данные на обучающую и проверочную выборки, чтобы контролировать переобучение.

Ход занятия

10	**Что такое MNIST?** MNIST (Modified National Institute of Standards and Technology) — классический датасет для обучения и тестирования моделей компьютерного зрения. Характеристики: 70 000 изображений рукописных цифр от 0 до 9 60 000 изображений в обучающей выборке, 10 000 — в тестовой Каждое изображение: 28×28 пикселей, оттенки серого (0 = чёрный, 255 = белый) Данные нормализуются: пиксели делятся на 255, чтобы значения были в диапазоне [0, 1] Зачем используется: Простой, но нетривиальный датасет для отладки архитектур Позволяет быстро обучать модели даже на обычных компьютерах Стандартный бенчмарк: если модель не работает на MNIST, вряд ли она будет работать на сложных задачах Дополнительно: Показывайте ученикам примеры цифр — некоторые написаны неразборчиво, с наклоном, разным почерком. Это учит сеть обобщать, а не запоминать. Датасет доступен в PyTorch через torchvision.datasets.MNIST.
11	**Пишем простую сеть на PyTorch** Архитектура нашей сети для MNIST: Вход: 784 нейрона (28×28 пикселей, изображение «разворачивается» в вектор) Скрытый слой: 128 нейронов + функция активации ReLU Выходной слой: 10 нейронов (по одному на класс 0–9) import torch.nn as nn model = nn.Sequential(nn.Linear(784, 128), # Полносвязный слой: 784 входа → 128 выходов nn.ReLU(), # Функция активации nn.Linear(128, 10) # Выходной слой: 128 → 10 классов

Ход занятия

	) Дополнительные компоненты: Функция потерь: nn.CrossEntropyLoss() — комбинирует LogSoftmax и NLLLoss, подходит для многоклассовой классификации Оптимизатор: torch.optim.Adam(model.parameters(), lr=0.001) — адаптивный градиентный спуск Обучение: 1 эпоха для демонстрации, но на практике используют 5–10 Дополнительно: Перед подачей в сеть изображение 28×28 преобразуется в вектор длины 784 через .view(-1, 784) или .flatten(). PyTorch автоматически вычисляет градиенты через механизм autograd — не нужно писать backprop вручную.
12	**Что показывает результат?** Accuracy (точность) — доля правильно классифицированных примеров: Accuracy = (число верных предсказаний) / (общее число примеров) Пример: если из 10 000 тестовых цифр сеть правильно распознала 9 300 → Accuracy = 93% Что влияет на точность: Архитектура сети: больше слоёв/нейронов → выше потенциал, но риск переобучения Количество эпох: слишком мало — недообучение, слишком много — переобучение Скорость обучения: слишком высокая — расходимость, слишком низкая — медленное обучение Качество данных: шум, смещение в датасете ухудшают обобщение Дополнительно: После 1 эпохи простая сеть на MNIST обычно достигает 92–95% точности. Цифры 1 и 0 распознаются лучше всего (простая форма), а 4, 7, 9 — хуже (похожие начертания, разный почерк). Для анализа ошибок можно построить матрицу ошибок (confusion matrix) —

Ход занятия

	таблица, показывающая, какие цифры сеть путает между собой.

Практика

Вопросы для самопроверки

Что делает искусственный «нейрон»?

Зачем в нейросети нужны слои?

Что делает функция активации?

Почему важно уметь измерять ошибку?

Что такое градиентный спуск?

Задание 1

Создайте простую нейросеть в PyTorch для распознавания рукописных цифр (датасет MNIST). Используйте один скрытый слой с 128 нейронами, функцию активации ReLU и функцию потерь CrossEntropyLoss. Обучите модель одну эпоху и выведите точность.

Подсказки

Входной размер: 784 (28x28 пикселей)

Скрытый слой: 128 нейронов

Выходной слой: 10 классов (цифры от 0 до 9)

```
model = nn.Sequential(
    nn.Linear(784, 128),
    nn.ReLU(),
    nn.Linear(128, 10)
)
```

Дополнительно

Используйте `optimizer = torch.optim.SGD(...)` или `Adam`

Для потерь: `criterion = nn.CrossEntropyLoss()`

Не забудьте: `loss.backward()` и `optimizer.step()` в цикле обучения

Примеры выполнения заданий

Проверь себя по следующим критериям:

Искусственный нейрон — принимает входные сигналы, умножает их на веса, суммирует и применяет функцию активации, чтобы выдать выход.

Слои в нейросети — позволяют сети извлекать иерархические признаки: от простых к сложным (например, от краёв к объектам).

Функция активации — вводит нелинейность, позволяя сети моделировать сложные зависимости; без неё сеть сводится к линейной.

Измерение ошибки — нужно, чтобы понимать, насколько предсказания сети далеки от реальных значений и как её улучшать.

Градиентный спуск — алгоритм оптимизации, который корректирует веса сети в направлении, уменьшающем ошибку.

Задание 1

```
import torch
import torch.nn as nn
import torchvision.transforms as transforms
import torchvision.datasets as datasets
from torch.utils.data import DataLoader
# Устройство
device = torch.device('cuda' if torch.cuda.is_available()
else 'cpu')
# Модель
model = nn.Sequential(
    nn.Flatten(),
    nn.Linear(784, 128),
    nn.ReLU(),
    nn.Linear(128, 10)
).to(device)
# Данные
```

Примеры выполнения заданий

```
transform = transforms.ToTensor()

train_dataset = datasets.MNIST(root='./data', train=True,
download=True, transform=transform)

train_loader = DataLoader(train_dataset, batch_size=64,
shuffle=True)

# Оптимизатор и функция потерь

criterion = nn.CrossEntropyLoss()

optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

# Обучение (1 эпоха)

model.train()

correct = total = 0

for images, labels in train_loader:

    images = images.view(-1, 784).to(device)

    labels = labels.to(device)

    optimizer.zero_grad()

    outputs = model(images)

    loss = criterion(outputs, labels)

    loss.backward()

    optimizer.step()

    _, predicted = torch.max(outputs, 1)

    total += labels.size(0)

    correct += (predicted == labels).sum().item()

print(f"Точность после 1 эпохи: {100 * correct /
total:.2f}%")
```

Сегодня на занятии мы:

- Познакомились с внутренней структурой нейросетей и принципами их обучения.
- Выявили основные компоненты нейрона и их роль в сети.
- Рассмотрели распространённые функции активации (ReLU, Sigmoid, Tanh).
- Поняли алгоритм обучения через прямой и обратный проходы.
- Написали простую нейросеть на PyTorch.
- Обучили модель распознавать рукописные цифры на датасете MNIST.

На следующем занятии вы:

- Узнаете, что такое zero-shot, one-shot и few-shot подходы к промптингу.
- Поймёте, почему важно приводить примеры в запросах к GPT.
- Научитесь задавать понятные и точные промпты.
- Оцените разные источники данных для prompt engineering.

Получение фидбека. Рефлексия. Проверка знаний

Попросите ребят решить интерактивные задачи по теме урока.

Ответы к интерактивным задачам

Практическое задание 1

Какая функция активации применяет формулу f(x) = max(0, x)?

- **ReLU**
- Sigmoid
- Tanh
- Softmax

Практическое задание 2

Выберите ключевые компоненты процесса обучения нейронной сети

- **Функция ошибки (loss)**
- **Градиентный спуск**
- **Обновление весов**
- Компиляция программы
- Установка операционной системы

Практическое задание 3

Как называется библиотека для глубокого обучения, используемая в примере кода из лекции?

PyTorch

Вопрос-ответ

Если во время подготовки или проведения урока возникают вопросы, их необходимо адресовать в методический чат.