Задание:

Цель: получить базовые навыки работы с одной из библиотек глубокого обучения (Caffe, Torch, TensorFlow или MXNet на выбор) на примере полностью связанных нейронных сетей.

Задачи:

Выполнение практической работы предполагает решение следующих задач:

1. Выбор библиотеки для выполнения практических работ курса.

2. Установка выбранной библиотеки на кластере.

3. Проверка корректности установки библиотеки. Разработка и запуск тестового примера сети, соответствующей логистической регрессии, для решения задачи классификации рукописных цифр набора данных MNIST (пример разобран в лекционных материалах).

4. Выбор практической задачи компьютерного зрения для выполнения практических работ.

5. Разработка программ/скриптов для подготовки тренировочных и тестовых данных в формате, который обрабатывается выбранной библиотекой.

6. Разработка нескольких архитектур полностью связанных нейронных сетей (варьируются количество слоев и виды функций активации на каждом слое) в формате, который принимается выбранной библиотекой.

7. Обучение разработанных глубоких моделей.

8. Тестирование обученных глубоких моделей.

9. Сделать вывод относительно разработанных архитектур.

10. Подготовка отчета, содержащего минимальный объем информации по каждому этапу выполнения работы.

Выполнение задач:

1. В ходе данного проекта я освоила базовые навыки работы с библиотекой глубокого обучения TensorFlow на примере построения и обучения полностью связанных нейронных сетей для задачи бинарной классификации. Полностью связная нейронная сеть может быть описана последовательностью слоев с весами. Например, для сети с одним скрытым слоем:

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Содержимое, созданное искусственным интеллектом, может быть неверным.

где:

* W — матрица весов скрытого слоя;
* b — вектор смещений скрытого слоя;
* Wout — вес для выходного слоя;
* bout — смещение для выходного слоя;
* f — функция активации для скрытого слоя (например, ReLU);
* g — функция активации для выходного слоя, используемая для бинарной классификации (обычно это сигмоид):

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Содержимое, созданное искусственным интеллектом, может быть неверным.

Для бинарной классификации используется бинарная кросс-энтропийная функция потерь:

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Содержимое, созданное искусственным интеллектом, может быть неверным.

где:

* y — истинная метка для образца i;
* ypred — предсказанная вероятность принадлежности к классу 1 для образца i.

Обучение сети заключается в минимизации функции потерь с помощью метода градиентного спуска или его вариантов (например, Adam или аналогов):

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Содержимое, созданное искусственным интеллектом, может быть неверным.

где:

* θ — параметры модели (веса и смещения);
* η — скорость обучения;
* ∇L(θ) — градиент функции потерь по параметрам.

2. Были использованы данные подмножество набора данных Food-101. Боссард, Лукас, Матье Гийомен и Люк Ван Гул. «Food-101 – Mining Discriminative Components with Random Forests». Изображения разделены на две папки: pizza и not\_pizza, имеющее равное количество изображений (по 983 шт.). Все изображения были масштабированы таким образом, чтобы максимальная длина стороны составляла 512 пикселей (однако в коде длина стороны была уменьшена для экономии ОЗУ). Содержание папок соответствует их названию. Данные разделены: Количество обучающих изображений: 1376 Количество валидационных изображений: 294 Количество тестовых изображений: 296

3.Была использована метрика качества Accuracy, которая определяется следующим образом:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Содержимое, созданное искусственным интеллектом, может быть неверным.

где:

* TP (True Positives) — количество истинных положительных предсказаний (правильно классифицированные объекты класса 1);
* TN (True Negatives) — количество истинных отрицательных предсказаний (правильно классифицированные объекты класса 0);
* FP (False Positives) — количество ложных положительных предсказаний (объекты класса 0, ошибочно классифицированные как класс 1);
* FN (False Negatives) — количество ложных отрицательных предсказаний (объекты класса 1, ошибочно классифицированные как класс 0).

Данные для работы были взяты с сайта kaggle, хранятся локально в двух папках с файлами в формате Jpg.

5. Я использовала создание датасета с помощью встроенной функции tensorflow.keras.preprocessing.image\_dataset\_from\_directory, в нее подавались пути к изображениям и изображения, а на выходе получили Кортеж (features, label): где features — это тензор с признаками, а label — это тензор с меткой класса. После этого я использовала загрузчик данных DataLoder. В него подается полученный на предыдущем этапе датасет, а в итоге получаем пакеты данных (batch), состоящие из кортежей (features, labels) для каждого образца в пакете.

6. Мной была разработана функция для обучения нейронной сети:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Содержимое, созданное искусственным интеллектом, может быть неверным.

В приведенной функции происходит процесс обучения модели на тренировочном наборе данных с последующей валидацией на валидационном наборе:

1) инициализируем списки для хранения потерь и метрики качества;

2) переводим модель в режим обучения, создаем переменную для накопления потерь в текущей эпохе;

3) проходим циклом по каждому батчу. В цикле обнуляем градиенты, получаем предсказания, вычисляем потери, вычисляем градиенты на основе потерь, обновление параметров модели на основе градиентов, производим накапливание потерь;

4) вычисляем средние потери за эпоху и сохраняем их;

5) тестируем модель на валидационных данных и сохраняем результат;

6) Пункты 2-5 выполняются по количеству эпох.

7. В ходе работы были разработаны три нейронные сети.

* простая модель. Один скрытый линейный слой, функция активации скрытого слоя – RELU;

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Содержимое, созданное искусственным интеллектом, может быть неверным.

* Средняя модель (четыре скрытых слоя, функция активации скрытого слоя – тангенс):

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Содержимое, созданное искусственным интеллектом, может быть неверным.

* Сложная модель (четыре скрытых слоя, используем нормализацию, обнуление нейронов и LeakyReLU в качестве активации):

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Содержимое, созданное искусственным интеллектом, может быть неверным.

8. Результаты:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Модель | Время обучения | Кол-во скрытых слоев | Функция активации | Accuracy |
| create\_simple\_nn | 4671 | 1 | ReLU | 67,45 |
| create\_medium\_nn | 18463 | 4 | Tanh | 64,69% |
| create\_complex\_nn | 21316 | 4 | LeakyReLU | 68% |

Вывод:

Сreate\_simple\_nn () оказалась наиболее успешной из предоставленных данных с максимальной точностью 69.5%. Её простота позволила избежать излишней сложности и проблем с настройкой гиперпараметров, но она ограничена в способности улавливать сложные закономерности, что выражается в плато на поздних эпохах.

Сreate\_medium\_nn() (предположительно) могла показать схожие или чуть лучшие результаты благодаря четырём слоям, но активация tanh, вероятно, замедлила обучение из-за исчезающего градиента.

Сreate\_complex\_nn() неожиданно показала худший результат (66.36%), несмотря на сложную архитектуру с Batch Normalization, Dropout и LeakyReLU. Это может быть связано с неоптимальной скоростью обучения (0.001), избыточным Dropout или общей неподходящей архитектурой для изображений. Однако она лучше контролировала переобучение.

Основная проблема всех моделей: Полносвязные сети не подходят для классификации изображений, так как теряют пространственную информацию. Это ограничивает точность на уровне 66–69%, независимо от сложности архитектуры. Для сравнения, свёрточные нейронные сети (CNN) могли бы достичь точности 80–90% на аналогичных задачах, если данные содержат явные визуальные различия между классами.