В заключение необходимо отметить, что электронная библиотека становится важным элементом современной образовательной среды факультета математики и информатики Гродненского государственного университета им. Янки Купалы, обеспечивая удобство и доступность современной ИТ-литературы для студентов и преподавателей.

**Список литературы**

1. Обзор функционала Joomla! [Электронный ресурс] / Сайт проекта Joomla!. – URL: https://joomla.ru/docs/articles/cms-joomla/1821-joomla-opportunities (дата обращения: 08.11.2023).

2. Малюшкин, Р. NLP для людей. Часть 1 [Электронный ресурс] / Р. Малюшкин // Medium – Where good ideas find you. – URL: https://medium.com/stseusp/nlp-for-people-1-c9b54ffce13f (дата обращения: 09.11.2023).

3. Котюбеев, Р. Предобработка текста в NLP [Электронный ресурс] / Р. Котюбеев // PYTHON SCHOOL. – URL: https://python-school.ru/blog/nlp-text-preprog716754540 (дата обращения: 09.11.2023).

4. Mikolov, Tomas; et al. Efficient Estimation of Word Representations in Vector Space [Электронный ресурс] / arXiv.org e-Print archive. – URL: https://arxiv.org/pdf/1301.3781 (дата обращения: 08.11.2023).

УДК 004.8

## ПОСТРОЕНИЕ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ ДЛЯ РЕШЕНИЯ ЗАДАЧ КЛАССИФИКАЦИИ С ПОМОЩЬЮ БИБЛИОТЕКИ PYTORCH

*И. В. Савицкий*
*“Гродненский государственный университет им. Янки Купалы”, г.Гродно*
*Научный руководитель: И. Б. Просвирнина, кандидат физико-математических наук, доцент*

Сверточные нейронные сети (CNN) являются мощным инструментом для анализа и распознавания изображений. Они могут быть применены во множестве сфер человеческой жизни, где требуется классификация объектов или процессов по определенным критериям. Особый интерес представляет их применение в медицине, где большой объем данных и сложные взаимодействия между переменными способствуют эффективной моделированию с помощью глубокого обучения. Использование CNN в анализе медицинских изображений и сигналов позволяет достичь высокой точности диагностики и улучшить качество медицинской практики.

Для поиска наборов данных будем использовать платформу Kaggle. Выберем датасет Chest X-Ray Images (Pneumonia), организованный в 3 папки (train, test, val) и содержащий подпапки для каждой категории изображений (пневмония/норма). Имеется 5863 рентгеновских изображения (JPEG) и 2 категории (пневмония/норма).

Приступим к этапу загрузки, очистки и организации данных. Импортируем все необходимые для глубокого обучения библиотеки, такие как numpy, torch, torchvision, pandas, pathlib, matplotlib.

Зададим последовательность необходимых преобразований, используя объекты класса Compose из библиотеки torchvision.transforms. По умолчанию изображения имеют 3 канала (RGB) и для упрощения построения модели сделаем их черно-белыми (1 канал). Также уменьшим размер изображения до размера 60 на 60 пикселей. Также к выборкам применим аугментацию данных. Аугментация выполняется с использованием алгоритма TrivialAugment, который случайным образом изменяет яркость, контраст, насыщенность и цвет изображений. transforms.ToTensor() преобразует входное изображение или видео из формата PIL (Python Imaging Library) в формат тензора PyTorch. Он также нормирует тензор к диапазону значений от 0 до 1, деля все значения пикселей на 255.

Загрузим данные. При загрузке применим трансформацию. Данные датасета уже поделены на тренировочные, валидационные и тестовые.

Для глубокого обучения для бинарной классификации снимков мы будем использовать модель, созданную по подобию VGG16. VGG16 — это архитектура сверточной нейронной сети, достигающая точности 92.7% - топ-5, при тестировании на ImageNet в задаче распознавания объектов на изображении. Архитектура модели VGG16 включает 16 слоев, включая 13 сверточных слоев и 3 полносвязных слоя.

Математический подход к определению сверточных нейронных сетей включает использование свертки, операции max-pooling, активационных функций и алгоритма прямого и обратного распространения ошибки.

Свертка - это математическая операция, которая плавно перемещает одну функцию по другой и измеряет интеграл их точечного умножения. В PyTorch свертка реализована с помощью nn.Conv2d() со следующими параметрами: in_channels(int) - количество каналов во входном изображении, out_channels(int) - количество выходных каналов, созданных сверткой, kernel_size(int or tuple) - размер сворачиваемого ядра/фильтра, stride(int or tuple, optional) - насколько большой шаг выполняет свертывающее ядро.

Max-pooling - операция в сверточных нейронных сетях, которая выполняет субдискретизацию изображения путем выбора максимального значения из каждого окна области на изображении и присваивания этого значения новому пикселю на выходе. Применяется после сверточных слоев, чтобы уменьшить количество параметров модели и предотвратить переобучение. Отрицательным эффектом max-pooling может быть потеря некоторой информации о местоположении объектов на изображении.

Активационные функции в сверточных слоях обычно являются нелинейными, например, rectified linear unit (ReLU), которая обнуляет все отрицательные значения, а положительные оставляет без изменений.

Алгоритм прямого распространения - это алгоритм машинного обучения, который используется в нейронных сетях для расчета выходных значений на основе входных данных. При работе нейронной сети, входные данные, которые

представлены в виде вектора, передаются через каждый узел сети, где происходит вычисление значений. Результат вычислений передается на вход следующего узла, и процесс продолжается до вычисления выходного значения нейронной сети.

Алгоритм обратного распространения - это алгоритм машинного обучения, который используется для обучения нейронных сетей путем корректировки весов на основе вычисленной ошибки между предсказанными и реальными значениями.

Для обучения модели существуют функции потерь, основной задачей которых является минимизация ошибок между фактическими и предсказанными значениями в процессе обучения. Для обновления параметров модели в процессе обучения с целью минимизации функции потерь необходимы особые алгоритмы - оптимизаторы. Они определяют, как модель должна обновлять веса и смещения в процессе обучения, чтобы достичь наилучшего качества предсказаний.

Существует множество реализаций функций потерь в зависимости от задачи машинного обучения и типа предсказаний модели. В нашей модели будем использовать BCE Loss. Binary Cross-Entropy Loss - используется в задачах бинарной классификации, когда необходимо получить вероятность принадлежности к одному из двух классов.

В качестве оптимизатора для нашей модели будем использовать SGD. Стохастический градиентный спуск (SGD) - наиболее распространенный оптимизатор, который использует градиент для обновления параметров. Для оценки качества модели выберем метрику accuracy. Accuracy (точность) - мера того, насколько хорошо модель правильно классифицирует данные.

Теперь приступаем к построению собственной модели на основе уже рассмотренной ранее VGG16. По архитектуре она будет представлять собой уменьшенную и упрощенную копию своего «донора», чтобы модель не переобучалась из-за слишком усложненной и громоздкой архитектуры, которой по своей сути VGG16 и является для данного набора данных.

Начнем с инициализации модели. Для большей гибкости модели создадим подкласс nn.Module. Для него необходимо описание функции forward(). Создадим все экземпляры всех слоев, и затем воспользуемся всеми этими экземплярами один за другим в функции forward(). Всего модель EightModel содержит два сверточных слоя (Conv2d), и два полносвязных (Linear). В первом сверточном слое имеем только один входной канал, т.к. на вход подаются черно-белые снимки, размер ядра указываем 3 на 3, padding присвоим 1, чтобы центр ядра попадал на каждый пиксель, в том числе и на крайние, размер шага также сделаем равным 1. Далее применяем операцию батч-нормализации (BatchNorm2d), которая снизит риск переобучения и ускорит процесс обучения за счет более быстрого схождения данных. После применим функцию активации nn.ReLU() и наконец используем операцию Max Pooling для уменьшения размерности данных в 2 раза и извлечения наиболее значимых признаков из исходных данных. В полносвязном слое применим nn.Flatten() для выпрямления многомерного

массива в одномерный вектор, nn.Dropout() для зануления (отключения) случайных нейронов на каждой итерации обучения для улучшения обобщающей способности модели, и также функцию активации nn.ReLU().

Таким образом входящее изображение размерностью проходит следующие преобразования: 1 * 60 * 60 => 8 * 30 * 30 => 8 * 15 * 15 => 1800 => 100 => 2.

После инициализации модели следующим шагом определим функцию для обучения модели в цикле на тренировочной выборке и вычисления значения функции потерь и метрики точности на каждой эпохе обучения. Сначала настроим модель в режим обучения. На каждой эпохе, данные подаем в модель и полученное предсказание используется для вычисления значения функции потерь. Затем градиенты обнуляются и выполняется обратное распространение ошибки. Оптимизатор используется для обновления параметров модели. Наконец, метрика точности вычисляется для каждого батча данных. После прохода по всем батчам данных для каждой метрики вычисляется среднее значение. Функция возвращает среднее значение функции потерь и точности на всех батчах.

Следующим шагом определим функцию, которая выполняет обучение модели в цикле, вычисляя значения функции потерь и точности на тренировочных, валидационных и тестовых выборках для каждой эпохи обучения. На вход функции подаются необходимые параметры для обучения и тестирования модели, определение функции потерь, её параметры и число эпох обучения. В цикле создается словарь для хранения результатов. В каждой эпохе происходит обучение модели на тренировочных данных, проверка производительности модели на валидационных данных и оценка качества модели на тестовых данных. Результаты выводятся на экран и обновляются в словаре. После прохода по числу эпох функция возвращает словарь с результатами.

Для начала обучения осталось лишь запустить тренировочную функцию. Тут же укажем количество эпох – 30, создадим экземпляр model_exemplar класса EightModel, установим функцию потерь (loss_fn) - кросс-энтропию и оптимизатор - стохастический градиентный спуск (SGD). Результаты работы функции train_function() присвоим переменной model_exemplar_results.

Таким образом мы обучили нашу модель до точности 88-90% на всех выборках, что можно считать приемлемой точностью. Это значит, что по 9 из 10 снимков мы получим правильно поставленный диагноз. Модель не преодолела порог в 90% точности. Остановимся на этом результате.

## Список литературы

1. Стивенс Эли, Антига Лука, Виман Томас С80 PyTorch. Освещая глубокое обучение. — СПб.: Питер, 2022. — 576 с.

2. Жерон, Орельен. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow: концепции, инструменты и техники для создания интеллектуальных систем. Пер. с англ. - СпБ.: ООО "Альфа-книга': 2018. - 688 с.