Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

К ЗАЩИТЕ ДОПУСТИТЬ
Зав. кафедрой АСУ,
канд. техн. наук, доцент
В.В. Романенко

(подпись)
«____»_____ 2024 г.
(дата)

УЧЕБНЫЙ КОМПЛЕКС ДЛЯ ПРОЕКТИРОВАНИЯ И ОБУЧЕНИЯ НЕЙРОННЫХ СЕТЕЙ

Бакалаврская работа
по направлению подготовки 09.03.01
«Информатика и вычислительная техника»

Выполнил: студент гр. 430-2
_____ А.А.Лузинсан
(подпись) (И.О. Фамилия)
«11» июня 2024 г.
(дата)
Руководитель:
профессор кафедры АСУ,
доктор тех. наук, профессор
(должность, ученая степень, звание)
_____ А.А. Захарова
(подпись) (И.О. Фамилия)
«___»_____ 2024 г.
(дата)

Томск 2024

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

УТВЕРЖДАЮ
Зав. кафедрой АСУ,
канд. техн. наук, доцент
В.В. Романенко

(подпись)
«____»_____ 2024 г.
(дата)

ЗАДАНИЕ

на бакалаврскую работу

студенту гр. 430-2 факультета систем управления

Лузинсан Анастасии Александровне
(Ф.И.О. студента)

1. Тема бакалаврской работы (БР): Учебный комплекс для проектирования и обучения нейронных сетей

(утверждена приказом по вузу от «___» _____ 2024 г. № _____).

2. Срок сдачи студентом законченной БР: «22» июня 2024 г.

3. Исходные данные к работе:

3.1 Интерактивный учебник «Dive into Deep Learning»,

3.2 Документация к фреймворку «Dearpygui»,

3.3 ОС ТУСУР 01-2021

4. Содержание расчетно-пояснительной записки / перечень подлежащих разработке вопросов:

4.1 Анализ требований

4.2 Определение спецификаций

4.3 Проектирование

4.4 Кодирование

4.5 Тестирование

5. Перечень графического материала (с точным указанием обязательных листов презентации):

6. Дата выдачи задания: «20» мая 2024 г.

Руководитель бакалаврской работы

профессор кафедры АСУ, доктор техн. наук, профессор		Захарова А.А.
(должность, ученая степень, звание)	(подпись)	(Ф.И.О.)

Задание принял к исполнению: «20» мая 2024 г.

Студент гр. 430-2		Лузинсан А.А.
	(подпись)	(Ф.И.О.)

Реферат

Бакалаврская работа содержит 87 страниц, 40 рисунков, 3 таблицы, 31 источник, 7 приложений.

УЧЕБНЫЙ КОМПЛЕКС, НЕЙРОННЫЕ СЕТИ, ВИЗУАЛЬНЫЙ РЕДАКТОР, ГЛУБОКОЕ ОБУЧЕНИЕ, КЛАССИФИКАЦИЯ ИЗОБРАЖЕНИЙ, СВЕРТОЧНЫЕ НЕЙРОСЕТИ

Объект разработки: процесс проектирования и обучения нейронных сетей

Предмет разработки: методы, технологии и системы проектирования и обучения нейронных сетей.

Цель работы: разработка учебного комплекса для проектирования и обучения нейронных сетей, который обеспечивает более эффективный процесс изучения глубокого обучения и формирования компетенций на основе типовых архитектур сверточных сетей.

Полученный результат работы: прикладное программное обеспечение, представляющее собой учебный комплекс для проектирования и обучения нейронных сетей.

Область применения разработки: учебный комплекс пригоден для общеобразовательных, профессиональных и высших учебных заведений, а также в качестве инструмента для повышения квалификации и профпереподготовки.

Бакалаврская работа выполнена в текстовом редакторе Microsoft Word и представлена в электронной версии в электронной образовательной среде ТУСУРа.

Abstract

Bachelor's thesis contains 87 pages, 40 figures, 3 tables, 31 sources, and 7 appendices.

EDUCATIONAL COMPLEX, NEURAL NETWORKS, VISUAL EDITOR, DEEP LEARNING, IMAGE CLASSIFICATION, CONVOLUTIONAL NEURAL NETWORKS

Object of development: the process of designing and training neural networks.

Subject of development: methods, technologies, and systems for designing and training neural networks.

Objective of the work: to develop an educational complex for designing and training neural networks that ensures a more effective process of learning deep learning and developing competencies based on typical convolutional network architectures.

Result of the work: application software representing an educational complex for designing and training neural networks.

Scope of application: the educational complex is suitable for general education, vocational, and higher education institutions, as well as a tool for advanced training and professional retraining.

The bachelor's thesis was prepared using Microsoft Word and is presented in electronic format in the electronic educational environment of TUSUR.

Оглавление

Введение

Глубокое обучение является важной и активно развивающейся областью искусственного интеллекта, которая в последние десятилетия привлекает все больше внимания как в академических кругах, так и в промышленности.

Однако обучение и работа с нейронными сетями, являющимися базовым инструментом глубокого обучения, требуют глубоких знаний в области математики, алгоритмов и программирования. Студенты часто сталкиваются с трудностями в освоении этой темы из-за ее сложности и недостатка структурированных и доступных образовательных ресурсов на русском языке.

Объект разработки представляет собой процесс проектирования и обучения нейронных сетей. Предметом разработки являются методы, технологии и системы проектирования нейронных сетей.

В связи с этим, в качестве основной задачи выступает создание учебного комплекса, который позволит студентам ознакомиться с основными концепциями, систематизировать и углубить свои знания в области глубокого обучения для задачи классификации изображений, а также приобрести практические навыки во время выполнения экспериментов с различными архитектурами нейронных сетей и анализа их результатов.

Целью выпускной квалификационной работы является разработка учебного комплекса для проектирования и обучения нейронных сетей, который обеспечивает более эффективный процесс изучения глубокого обучения и формирования компетенций на основе типовых архитектур сверточных сетей.

Для достижения поставленной цели необходимо выполнить следующие задачи:

1. Провести анализ требований.
2. Определить спецификации и функциональные возможности.
3. Выполнить проектирование интерфейса и архитектуры.
4. Реализовать учебный комплекс.

5. Протестировать учебный комплекс.

Результатом выполнения выпускной квалификационной работы является учебный комплекс для проектирования и обучения нейронных сетей. Комплекс представляет собой интерактивную среду, позволяющую студентам ознакомиться с основными концепциями и архитектурами глубокого обучения, создавать и настраивать свои собственные нейронные сети, проводить их обучение и валидацию.

1 АНАЛИЗ ТРЕБОВАНИЙ

1.1 Проблематика

В последние годы глубокое обучение и нейронные сети стали одними из наиболее динамично развивающихся областей компьютерных наук. Быстрое развитие и востребованность этих технологий обуславливают необходимость подготовки специалистов, обладающих глубокими знаниями и практическими навыками в области нейронных сетей.

На данный момент образовательные учреждения испытывают дефицит качественных и современных учебных комплексов, которые бы позволяли студентам не только изучать теоретические аспекты, но и приобретать практический опыт работы с нейронными сетями. Традиционные методы обучения часто не обеспечивают достаточной интерактивности и не позволяют студентам полностью погружаться в процесс проектирования и обучения нейронных сетей.

Сложность также представляет собой необходимость освоения фреймворков, реализующих нейронные сети, коих существует множество: TensorFlow, PyTorch, Keras, MXNet, JAX [1]. Они имеют свои особенности и области применения. Это создает дополнительную сложность для студентов, которым необходимо выбирать наиболее подходящий инструмент для конкретной задачи, что требует значительных усилий и времени на изучение их особенностей и сравнительных характеристик.

Еще одной проблемой является повторяющийся код при самостоятельном построении пайплайна обучения, тестирования и получения предсказаний. К тому же, в большинстве случаев все шаги оформляются в Jupyter Notebook [2], что снижает читаемость кода. Часто весь код находится в одном файле, что делает его сложным для понимания и сопровождения. Это особенно проблематично для новичков, которые могут легко запутаться в большом объеме несистематизированной информации.

Особую актуальность представляет собой ориентация на русскоязычный сегмент, поддерживающая суверенные технологии. Большая часть качественных курсов по глубокому обучению написана на английском языке, что создает трудности для русскоязычных студентов из-за языкового барьера. В результате возникает нехватка релевантных русскоязычных практикоориентированных учебных комплексов, которые могли бы помочь студентам освоить современные технологии и методы в области глубокого обучения. Необходимо также учитывать стратегию научно-технологического развития России, а также сквозные технологии Национальной технологической инициативы (НТИ) [3], которые интегрируются во все сферы деятельности. Создание учебных материалов, способствующих развитию компетенций в области глубокого обучения среди русскоязычных студентов, подчеркнет значимость отечественных технологий и их внедрения в различные отрасли экономики и науки. Это не только улучшит качество образования, но и поддержит подготовку высококвалифицированных специалистов, готовых к работе с передовыми технологиями.

Таким образом, одной из ключевых проблем, которую необходимо решить, является нехватка доступных и качественных образовательных комплексов на русском языке, способных эффективно и системно обучать студентов глубокому обучению и работе с нейронными сетями.

1.2 Развитие глубокого обучения для задачи классификации изображений

1.2.1 Основы глубокого обучения

В конце 1980-х годов нейронные сети стали распространенной темой в области машинного обучения и искусственного интеллекта благодаря изобретению различных эффективных методов обучения и сетевых структур. Одним из таких инновационных методов была, например, многослойная перцептронная сеть, обученная с помощью алгоритма «Обратное

распространение ошибки». Хотя нейронные сети успешно использовались во многих приложениях, в дальнейшем интерес к исследованию этой темы снизился ввиду больших вычислительных затрат, непосильных на тот момент. После этого, в 2006 году официально было представлено понятие «глубокое обучение», в основе которого лежала концепция искусственной нейронной сети [4].

Глубокое обучение представляет собой каскад нескольких уровней нелинейных фильтров для извлечения признаков, соответствующих различным уровням абстракции, который в конечном итоге должен обеспечивать сходимость некоторой целевой функции, связывающей набор входных и выходных данных [5].

В контексте данной работы рассматривается “обучение с учителем” для задачи классификации изображений. Под понятием “обучение с учителем” подразумевается процесс тренировки модели на основе заданного набора данных, который содержит метки, показывающие правильный ответ.

Типичная нейронная сеть в основном состоит из множества простых связанных обрабатывающих элементов, называемых нейронами, которые впоследствии проходят через функцию активации, получая на выходе целевое значение. Алгебраическая запись вычисления выходного сигнала простейшей нейронной сети для задачи регрессии на примере одного наблюдения, схематичное изображение которой представлено на рисунке 1.1, вычисляется по формуле:

$$y=f(\boldsymbol{w}^T\boldsymbol{x}+b), \tag{1.1}$$

где d – количество признаков;

$\boldsymbol{x}$ – входной вектор, $x_i \in \mathbb{R}^d$;

b – смещение;

$\boldsymbol{w}$ – весовые коэффициенты, $w_i \in \mathbb{R}^d$;

f – функция активации;

y – выходной сигнал.

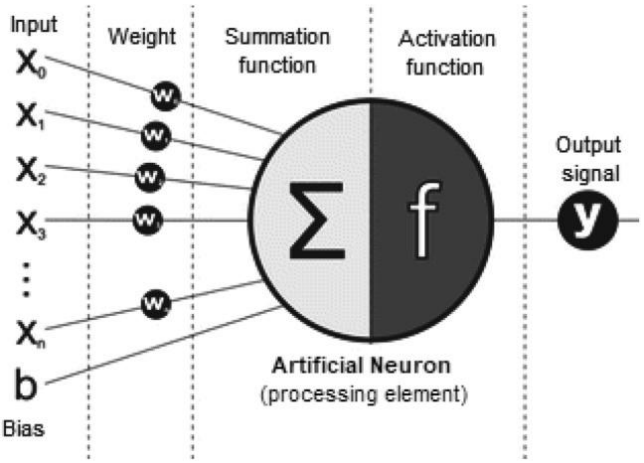

Рисунок 1.1 – Схематическое изображение простейшей нейронной сети для задачи регрессии на примере одного наблюдения

Переходя к задаче классификации применяется техника “one-hot encoding”, которая конвертирует метку данных в вектор с количеством компонент, равным количеству классов метки. В свою очередь компоненту, соответствующему конкретной категории экземпляра, будет присвоено значение 1, а всем остальным компонентам – 0. Таким образом, алгебраическая запись простейшей нейронной сети для задачи классификации на примере одного наблюдения будет выглядеть следующим образом:

$$y=f(\boldsymbol{W}\boldsymbol{x}+\boldsymbol{b}), \tag{1.2}$$

где d – количество признаков;

k – количество классов;

$\boldsymbol{x}$ – входной вектор, $x_i \in \mathbb{R}^d$;

$\boldsymbol{b}$ – смещение, $b_i \in \mathbb{R}^k$;

$\boldsymbol{W}$ – весовые коэффициенты, $w_i^j \in \mathbb{R}^{k \times d}$;

$(\boldsymbol{W}\boldsymbol{x}+\boldsymbol{b})$ – вектор условных вероятностей наблюдения (логиты, **o**);

f – функция активации;

y – выходной сигнал.

Учитывая набор данных, цель обучения – подобрать веса и смещение, которые в среднем позволяют прогнозам модели как можно ближе

соответствовать истинным меткам, наблюдаемым в данных. Методом достижения данной цели выступает функция потерь.

Функция потерь количественно определяет расстояние между реальными и прогнозируемыми значениями меток. Значение функции потерь обычно является неотрицательным числом, где чем оно меньше - тем лучше.

В задаче классификации часто вместе фигурируют понятия “Softmax” преобразования и “Cross-Entropy” функции потерь. Основными причинами, по которым, в отличие от задачи регрессии, в задаче классификации используют “Softmax” преобразование, являются: отсутствие гарантии, что сумма логитов после алгебраического преобразования будет равна единице, а также то, что все логиты являются неотрицательными. Таким образом, неотъемлемым этапом обучения нейронной сети для задачи классификации является “Softmax” преобразование, вычисляемое по формуле:

$$\hat{y}_i = \frac{exp(o_i)}{\sum_j exp(o_j)} \tag{1.3}$$

Наконец, рассматривая “Cross-Entropy” функцию потерь на примере одного наблюдения с q возможными классами меток, ее формула выглядит следующим образом:

$$l(y, \hat{y}) = \sum_{j=1}^{q} y_j \log(\hat{y}_j) \tag{1.4}$$

В случае добавления скрытых полносвязных слоев в архитектуру мы можем получить многослойный перцептрон. Однако, на деле скрытые слои будут представлять собой аффинные функции входных данных, которые тоже в свою очередь могут являться аффинными функциями. Аффинную функцию аффинной функции можно всегда свести к одному линейному преобразованию, в связи с этим добавление полносвязных слоев не привносит пользы в архитектуру.

Недостающей компонентой является нелинейная функция активации. Она применяется к каждому скрытому слою после линейного преобразования.

Самым популярным выбором благодаря простоте реализации и хорошей производительности является функция активации ReLU [6]:

$$ReLU(x) = max(x, o) \quad (1.5)$$

Причина использования ReLU заключается в предсказуемом взятии производной: либо она обнуляется (затухает), либо принимает значение единицы (активируется). Это улучшает качество оптимизации и смягчает эффект исчезновения градиента функции.

Сигмоидная функция [7] активации полезна в рекуррентных сетях. Она сжимает входные данные в диапазоне (-inf, inf), преобразуя их в значения на интервале (0, 1). При этом градиент функции обращается в нуль при больших положительных и отрицательных значениях аргументов, что является проблемой затухающего градиента. Данная функция выглядит следующим образом:

$$sigmoid(x) = \frac{1}{1 + exp(-x)} \quad (1.6)$$

Гиперболический тангенс [8], представленный формулой 1.7, также сжимает входные данные, но преобразуя их в значения на интервале (-1, 1). Он является симметричным относительно начала координат, поэтому производная данной функции в нуле принимает значение единицы. Но, как и сигмоидной функции, гиперболическому тангенсу также присуща проблема затухающего градиента.

$$tanh(x) = \frac{1 - exp(-2x)}{1 + exp(-2x)} \quad (1.7)$$

1.2.2 Основы сверточных нейронных сетей

Переходя непосредственно к задаче классификации изображений, классическим решением является использование сверточных нейронных сетей.

Для рассмотрения сверточного слоя положим, что мы имеем дело с входными данными, представленными двумерными одноканальными изображениями $\boldsymbol{X}$ и скрытым слоем $\boldsymbol{H}$ той же размерности. Оба представления имеют пространственную структуру. Теперь, пусть X_j^i и H_j^i обозначают пиксель в позиции (i, j) входного изображения и скрытого представления соответственно. Помимо этого, из-за появления пространственной структуры, необходимо перейти к весовым тензорам четвертого порядка **W** и матрице смещения **U**. Однако, этого недостаточно из-за требований компьютерного зрения: принципа трансляционной инвариантности, локальности и обобщения [9]. Для удовлетворения этих требований вводят понятие ядра свертки **V**, которое, ко всему прочему, сокращает количество обучаемых параметров. Ядро свертки представляет собой матрицу размера $\Delta \times \Delta \times c$, где Δ обычно не превышает значения 10, а c – это количество каналов. Визуализация свертки двухканального изображения до одного двумерного тензора представлена на рисунке 1.2. Рассматривая цветные изображения, получается следующая формула сверточного слоя в срезе одного пикселя:

$$H_{i,j,d} = \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} \sum_{c} V_{a,b,c,d} X_{i+a,j+b,c} \tag{1.8}$$

Форма выходного тензора определяется формой входного тензора $n_h \times n_w$ и формой ядра свертки $k_h \times k_w$ по формуле 1.9. При этом выходной слой обычно называют картой признаков.

$$(n_h - k_h + 1) \times (n_w - k_w + 1) \tag{1.9}$$

Существуют различные техники контроля размера выходных данных, которые решают различные проблемы в области распознавания изображений. Первая из них – это потеря информации на границах исходного изображения после свертки по причине того, что пиксели по периметру изображения не используются и проблема усугубляется при многократном применении сверток.

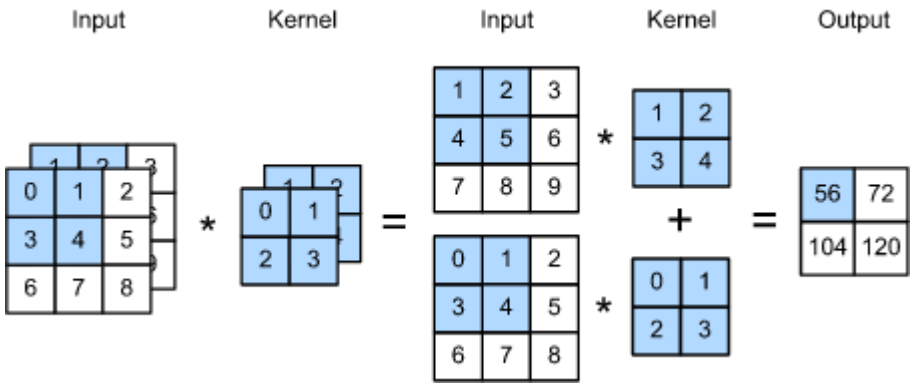

Рисунок 1.2 – Свертка двухканального изображения с помощью двухканального ядра

Решением данной проблемы является добавление дополнительных нулевых пикселей вокруг границы изображения, как показано на рисунке 1.3. Данная техника называется “padding” или “заполнение”. При этом, при добавлении p_h и p_w пикселей по вертикали и горизонтали соответственно, выходная форма будет иметь вид. представленный в формуле 1.9. Если требуется, чтобы форма выхода и входа была одинакова, применяется формула 1.10.

$$(n_h - k_h + p_h + 1) \times (n_w - k_w + p_h + 1) \tag{1.9}$$

$$p_h = k_h - 1,\ p_w = k_w - 1 \tag{1.10}$$

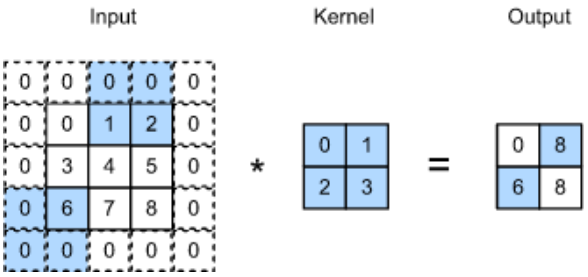

Рисунок 1.3 – Свертка входного изображения, подвергнутого технике “padding” со значением 1

Другая проблема, возникающая в контексте сверточных сетей, это избыточное качество изображений. Для решения этой проблемы с целью повышения скорости вычислений, и, следовательно, снижения дискретизации, используется техника “stride” или “добавление шага”. Пример использования данной техники с шагом 3 по вертикали и 2 по горизонтали с операции свертки

представлено на рисунке 1.4. Выходная форма при добавлении шага s_h и s_w по высоте и ширине соответственно имеет вид, показанный в формуле 1.11. При значениях “padding”, зафиксированных согласно формуле 1.10, выходная форма упрощается до формулы 1.12.

$$\left\lfloor \frac{n_h - k_h + p_h + s_h}{s_h} \right\rfloor x \left\lfloor \frac{n_w - k_w + p_w + s_w}{s_w} \right\rfloor \quad (1.11)$$

$$\left\lfloor \frac{n_h + s_h - 1}{s_h} \right\rfloor x \left\lfloor \frac{n_w + s_w - 1}{s_w} \right\rfloor \quad (1.12)$$

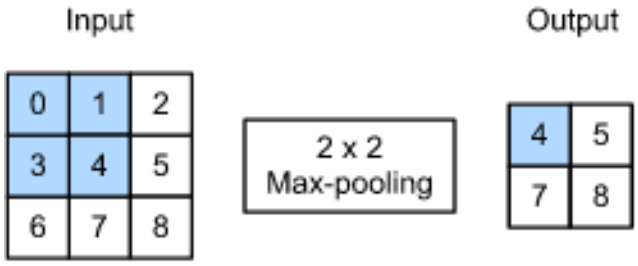

Рисунок 1.4 – Свертка входного изображения, подвергнутого технике “stride” со значением 3 по вертикали и 2 по горизонтали

Следующая техника, именуемая “pooling”, позволяет снизить чувствительность сверточных слоев к местоположению пикселя и пространственно снижает дискретизацию представлений. Ее принцип состоит в следующем: используется окно фиксированной формы, которое скользит по всем областям входных данных в соответствии с его шагом, вычисляя при этом один выходной результат для каждой области. Данное окно, в отличие от ядра свертки, не содержит параметров и является нелинейным преобразованием. Популярным выбором считается max-pooling и average-pooling, которые вычисляют максимум и среднее соответственно. Пример операции max-pooling представлен на рисунке 1.5.

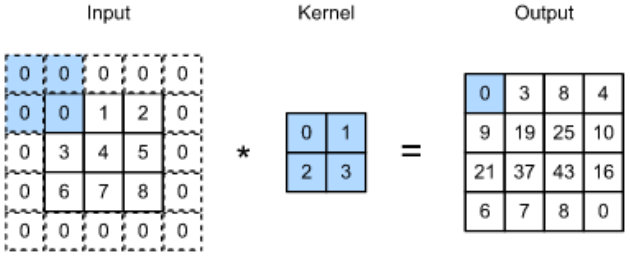

Рисунок 1.5 – Пример операции max-pooling с окном 2x2

1.2.3 Архитектуры сверточных нейронных сетей для задачи классификации изображений

Теперь у нас есть все ингредиенты, необходимые для сборки полнофункциональной сверточной нейронной сети. Одной из первых опубликованных сетей является архитектура LeNet, разработанная для задачи распознавания рукописных цифр на изображениях. Работа была представлена Яном Лекуном, исследователем из AT&T Bell Labs, и стала первым исследованием по успешному обучению CNN с помощью обратного распространения ошибки [10]. Архитектура сети представлена на рисунке 1.6.

В оригинальной версии используется average pooling и сигмоидные функции активации. Архитектура сети состоит из 2-х частей:

1. Сверточные слои с размером ядра свертки $5x5$, pooling слои с размером окна $2x2$ и шагом 2.

2. Три полносвязных слоя с количеством выходных карт признаков: 120, 84, 10. Датасет содержал десять классов, что соответствует последнему слою.

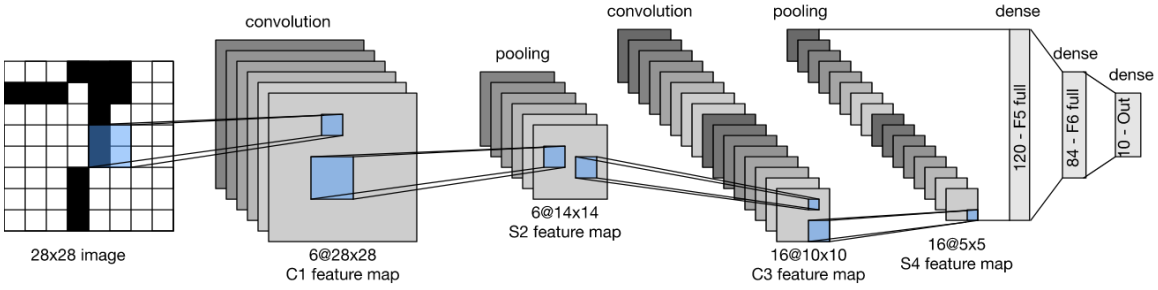

Рисунок 1.6 – Архитектура сети LeNet для входного изображения размерности $28x28$

Хотя LeNet добилась хороших результатов на первых небольших наборах данных, производительность и возможность обучения CNN на более крупных и реалистичных наборах данных еще оставляла желать лучшего. Фактически, на протяжении большей части промежутка времени между началом 1990-х и 2012 годами нейронные сети часто превосходили другие

методы машинного обучения. Первой современной CNN считают архитектуру AlexNet, названную в честь Алекса Крижевского, которая во многом представляет собой эволюционное улучшение по сравнению с LeNet. Она показала отличные результаты в соревновании ImageNet 2012 года [11]. Архитектура сети, показанная на рисунке 1.7, также состоит из двух частей:

1. Сверточные слои с размерами ядра свертки 11*x*11, 5*x*5 и три подряд идущих слоя с ядром свертки 3*x*3, а также с “max-pooling” слои с размером окна 3*x*3 и шагом 2.

2. Три полносвязных слоя с количеством выходных карт признаков: 4096, 84, 10.

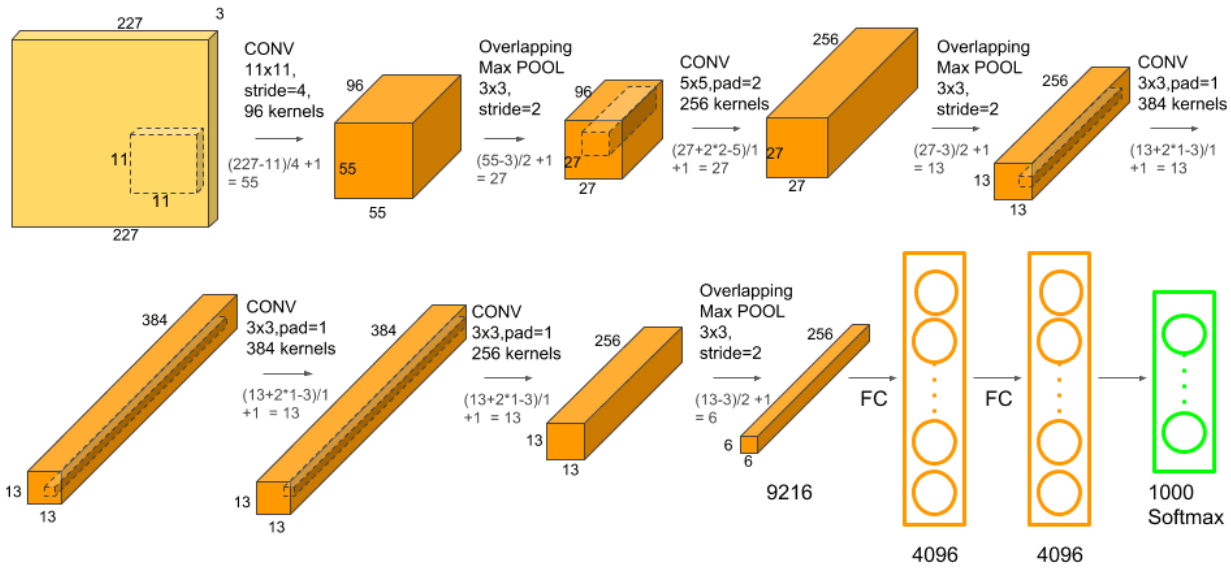

Рисунок 1.7 – Архитектура сети AlexNet для входного изображения размерности 227*x*227

Хотя AlexNet предоставил эмпирические доказательства того, что глубокие CNN могут достичь хороших результатов, он не предоставил общего шаблона, которым могли бы руководствоваться последующие исследователи при разработке новых сетей.

Идея использования блоков впервые возникла у рабочей группы VGG Оксфордского университета в одноименной сети VGG. Данная сеть

противоречит принципам LeNet и AlexNet, однако заложила основы для архитектур Inception и ResNet по части использования повторяющихся блоков. Существуют различные вариации архитектуры [12], каждая из которых представлена на рисунке 1.8 и содержит пять VGG блоков. Для интуитивного понимания архитектуры на рисунке 1.9 представлена версия VGG-16 для входного изображения размерности 224*x*224, рассчитанная на тысячу классов.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Рисунок 1.8 – Вариации архитектуры VGG

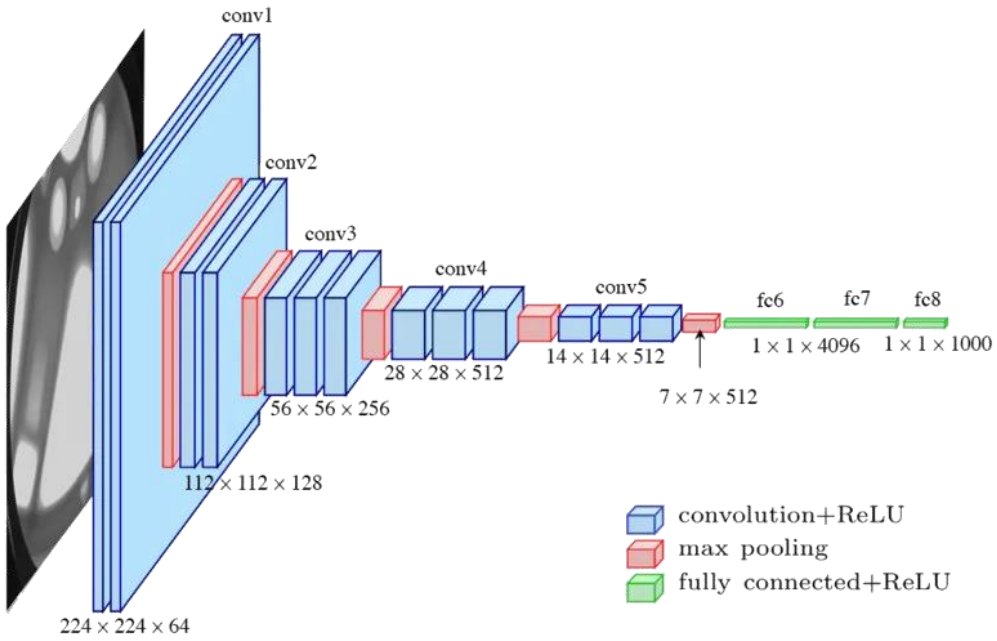

Рисунок 1.9 – Архитектура сети VGG-16 для входного изображения размерности 224*x*224

Основная проблема данной конструкции заключается в том, что полносвязные уровни в конце архитектуры потребляют огромное количество параметров. Например, даже такая простая модель, как VGG-11, требует огромной матрицы, занимающей почти 400 МБ ОЗУ в режиме одинарной точности (Float-32). Это существенное препятствие для вычислений, особенно на мобильных и встроенных устройствах.

Блоки «сеть в сети» (NiN) предлагают альтернативу, способную решить проблему с помощью одной простой стратегии: использовать свертки для добавления локальных нелинейностей по активациям каналов и использовать “global-average-pooling” для интеграции по всем местоположениям на последнем уровне представления. Архитектура NiN, представленная на рисунке 1.10, содержит сверточные модули, которые позволяют сильно повысить эффективность отдельных сверточных слоев посредством их комбинирования в более сложные группы. Стоит также заметить, что NiN совершенно не использует полносвязные слои, что кратно уменьшает количество параметров [13].

Структура сверточного блока включает следующие элементы:

- Три сверточных слоя с количеством выходных каналов 96, ядром свертки 11, 1 и 1 соответственно, а также шагом равным 4 без использования “padding”.
- После каждого сверточного слоя используется ReLU функция активации.

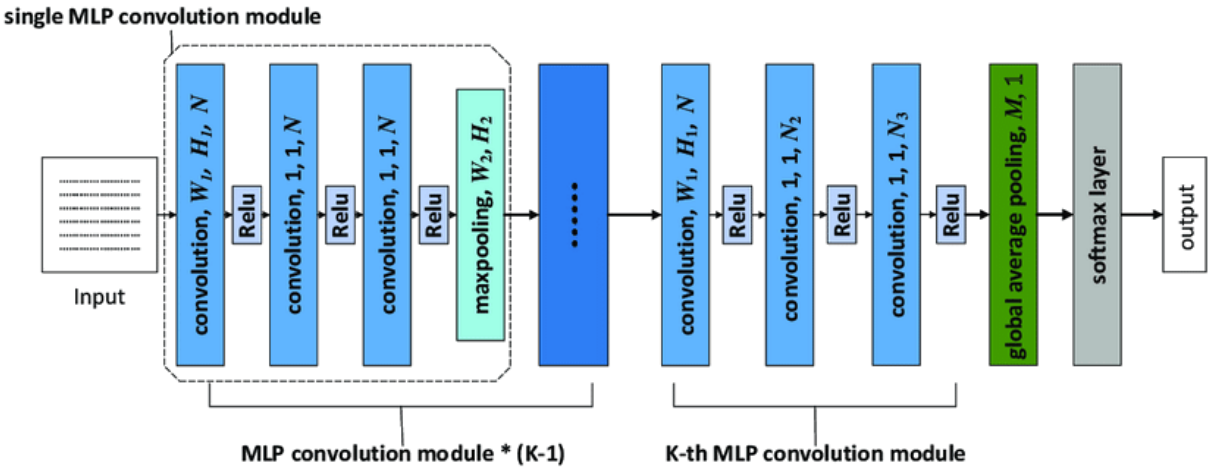

Рисунок 1.10 – Архитектура сети NiN

В 2014 году GoogLeNet выиграл конкурс ImageNet Challenge, используя структуру, сочетающую в себе сильные стороны NiN и повторяющихся блоков VGG. GoogLeNet является одним из первых сверточных сетей, в которой различаются основная часть (приём данных), тело (обработка данных) и голова (прогнозирование) [14]. Он также использует базовый сверточный блок Inception, показанный на рисунке 1.11.

Архитектура сети, представленная на рисунке 1.12, включает девять Inception блоков, сгруппированных на три группы (по 2, 5, 2 блока):

- Основу задают три свертки, которые работают с изображением и извлекают низкоуровневые признаки.
- Телом является набор сверточных блоков.
- В голове сопоставляются полученные признаки с целевой меткой.

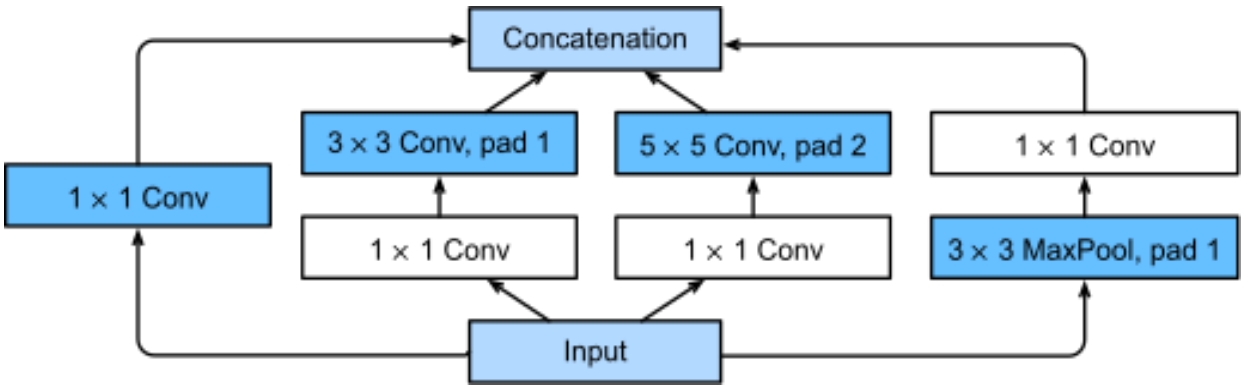

Рисунок 1.11 – Структура блока Inception

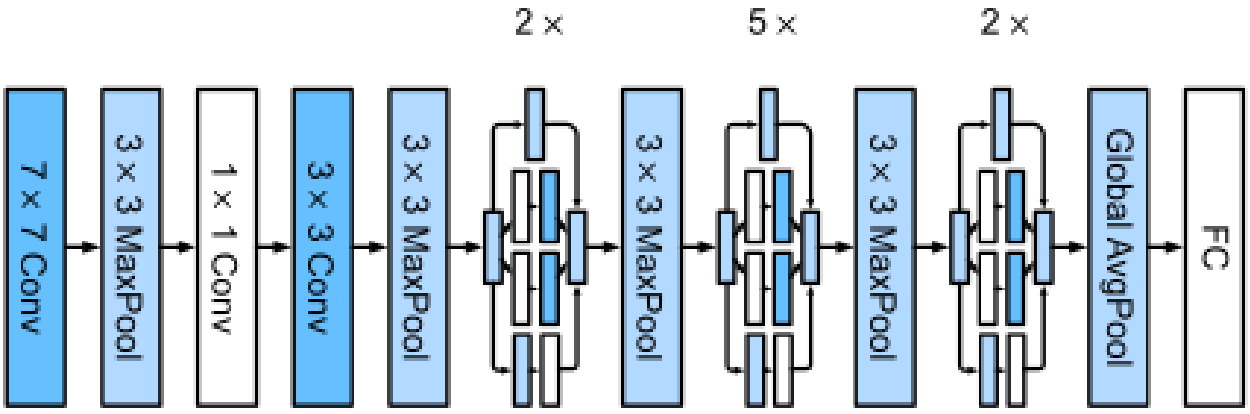

Рисунок 1.12 – Архитектура сети GoogLeNet

1.3 Формулирование требований

Отталкиваясь от предметной области, были сформулированы функциональные требования по отношению к разрабатываемому программному обеспечению:

1. Пользователи должны иметь возможность проектировать архитектуры нейронных сетей из доступных элементов.

2. Система должна включать датасеты, предназначенные для задачи классификации изображений.

3. Элементы интерфейса системы должны включать справочную информацию.

4. Пользователи должны иметь возможность настраивать готовые архитектуры нейронных сетей.

5. Система мониторинга должна отображать информацию о процессе обучения модели, включая метрики точности и функцию потерь.

6. Пользователи должны иметь возможность регулировать гиперпараметры обучения, такие как: функция потерь, оптимизатор обучения, скорость обучения, максимальное количество эпох и метод предварительной инициализации весов.

7. Пользователю должна быть предоставлена возможность скачать и загрузить веса обученной модели.

8. Система должна иметь функции импорта и экспорта элементов нейронной сети.

1.4 Обзор аналогов

Существует несколько интерактивных платформ, которые позволяет пользователям проектировать нейронные сети подобно созданию диаграмм. Рассмотрим наиболее известные из них:

1. Teachable Machine [15] – это веб-приложение от Google, которое позволяет пользователям создавать модели машинного обучения с помощью визуального интерфейса. Приложение ориентировано на простоту использования и не требует навыков программирования. Преимуществом системы является интуитивно понятный интерфейс, позволяющий создавать модели путем простого перетаскивания элементов; поддержка различных типов данных: изображений, звуков, человеческих поз; возможность быстрой проверки и тестирования созданных моделей. Среди недостатков были выделены ограниченные возможности по сравнению с профессиональными инструментами для глубокого обучения, в связи с чем система предназначена только для базового уровня обучения и экспериментов.

2. KNIME (Konstanz Information Miner) [16] – это платформа для анализа данных, которая поддерживает визуальное программирование. Она позволяет создавать потоки данных и модели машинного обучения с помощью

графического интерфейса. Система является мощным инструментом для анализа данных и моделирования, включающего поддержку глубокого обучения через интеграцию с Keras и TensorFlow. Визуальное программирование упрощает процесс создания моделей для пользователей без глубоких знаний программирования. Однако, для освоения платформы требуется длительная подготовка и основной фокус уделяется анализу данных, а не глубокому обучению.

3. IBM Watson Studio [17] предоставляет облачную среду для обработки данных, визуализации и создания моделей машинного обучения, включая глубокое обучение. Платформа включает инструменты для визуального проектирования моделей. Система предоставляет возможность интеграции с мощными облачными сервисами IBM для поддержки крупных проектов, а также визуальные инструменты для проектирования моделей и работы с данными. Тем не менее, во время работы могут потребоваться значительные ресурсы и знания для полной реализации возможностей платформы, которая ориентирована на профессионалов и корпоративное использование.

4. Azure Machine Learning Studio [18] – это облачная платформа от Microsoft для создания моделей машинного обучения с помощью визуального интерфейса. Платформа позволяет пользователям проектировать модели, перетаскивая компоненты на рабочее пространство. Система обладает интуитивно понятным визуальный интерфейсом для проектирования моделей машинного обучения; поддерживает интеграцию с другими сервисами Azure для масштабирования и развертывания моделей, а также будет понятна как для начинающих, так и для опытных пользователей. Однако, платформа может быть дорогой для индивидуальных пользователей и небольших команд.

5. Lobe [19] – это инструмент, приобретенный Microsoft, предназначенный для создания и обучения моделей машинного обучения с использованием визуального интерфейса. Он ориентирован на пользователей без технической подготовки. Преимуществом платформы является простота использования, не требующая навыков программирования и быстрое создание

прототипов и тестирование моделей. К недостаткам можно отнести ограниченные возможности по сравнению с профессиональными инструментами для глубокого обучения и ориентация на базовый уровень обучения и создания простых моделей.

6. Loginom [20] – это аналитическая low-code платформа, обеспечивающая интеграцию, очистку и анализ данных для принятия более эффективных управленческих решений. ПО от компании Loginom company предназначено для анализа и обработки бизнес-данных на базе методов визуального проектирования, является универсальным конструктором с набором готовых компонентов.

Проанализировав множество платформ для работы с нейронными сетями с помощью визуальных элементов, был выявлен дефицит отечественных решений, отвечающих на потребность в интерактивном учебном комплексе, что подтверждает актуальность разработки собственного продукта.

2 ОПРЕДЕЛЕНИЕ СПЕЦИФИКАЦИЙ

2.1 Выбор модели и методологии проектирования

Для разработки учебного комплекса для глубокого обучения была выбрана спиральная модель проектирования. Данная модель сочетается с принципами итеративного и инкрементного подходов, что позволяет гибко реагировать на изменения требований и обеспечивает поэтапную реализацию проекта.

Спиральная модель включает четыре основных фазы, которые повторяются на каждом витке спирали:

- определение целей и спецификаций;
- анализ рисков и прототипирование;
- разработка и тестирование;
- оценка и планирование следующего витка.

На этапе определения целей и спецификаций также формируются основные требования к функциональности и характеристикам учебного комплекса, а также проводится анализ существующих решений. Всего в ходе разработки было пройдено 4 итерации, которые последовательно включали следующие требования:

1. Комплекс должен включать базовый функционал и датасет FashionMNIST для проектирования и обучения простейшего многослойного персептрона. Комплекс должен поддерживать настройку гиперпараметров и мониторинг обучения в среде ClearML.

2. Комплекс должен содержать готовые однопоточные архитектуры LeNet, AlexNet и VGG, протестированные на датасете FashionMNIST. Слои готовых архитектур должны подлежать тонкой настройке.

3. Комплекс должен содержать блочную архитектуру NiN. Комплекс должен иметь возможность располагать архитектуру нейронной сети в свернутом и развернутом виде, редактировать и разворачивать свернутую

архитектуру. Пользователь должен иметь возможность экспортировать и импортировать слои нейронной сети. Обучение нейронной сети должно запускаться в отдельном потоке.

4. Комплекс должен включать многофилиальную архитектуру GoogLeNet. Все элементы должны быть снабжены справочной информацией. В интерфейсе должно отображаться текущее состояние сбора слоев, инициализации и обучения сети.

После определения целей был проведен анализ возможных рисков, которые могут возникнуть при разработке и внедрении комплекса. Среди выявленных рисков были выделены следующие:

- Технические риски, которые заключаются в сложности интеграции различных фреймворков и обеспечение высокой производительности при обработке данных.
- Риски удобства использования заключаются в необходимости создания интуитивного и простого интерфейса для пользователей с разным уровнем подготовки.
- Риски совместимости касаются обеспечения работы комплекса в различных операционных системах.

Для минимизации рисков на этом этапе сначала был создан прототип [21]. Далее в ходе преддипломной практики была реализована первая версия учебного комплекса, которая включала ключевые компоненты системы.

Этап разработки включал в себя реализацию функциональности, определенной на предыдущих этапах. В соответствии с принципами итеративного подхода, разработка велась по циклам, на каждом из которых реализовывались и тестировались отдельные модули и функции системы.

Тестирование проводилось на каждом этапе разработки, чтобы гарантировать корректность и стабильность работы системы. В качестве метода было выбрано модульное тестирование отдельных компонентов, сбор модулей в пайплайн обучения и запуск обучения на подготовленном датасете.

После завершения каждого витка разработки проводилась оценка результатов и планирование следующего цикла. Так как за основу учебного комплекса была взята интерактивная книга по глубокому обучению “Dive into Deep Learning” [22], то анализ соответствия реализованной функциональности требованиям и спецификациям производился в соответствии с результатами, продемонстрированными в упомянутом учебнике. Планирование задач, затрачиваемое время и корректировка графика выполнения бакалаврской работы в свою очередь фиксировались на платформе ClickUp [23]. Помимо этого, данный этап включал выявление и исправление ошибок и недоработок, выявленных на этапе тестирования.

Выбор в пользу спиральной модели проектирования была сделан в связи с возможностью поэтапной реализации проекта, которая позволила постепенно наращивать функциональность и проводить тестирование на каждом этапе, что снизило вероятность появления критических ошибок на финальных стадиях разработки.

2.2 Выбор инструментов разработки

При разработке учебного комплекса был тщательно выбран набор инструментов, обеспечивающий эффективную и удобную разработку, а также поддерживающий ключевые требования проекта. Основными критериями выбора инструментов стали популярность, функциональность, простота использования и активная поддержка со стороны сообщества. Рассмотрим основные инструменты разработки, выбранные для данного проекта:

1. Visual Studio Code был выбран в качестве интегрированной среды разработки благодаря своей популярности, легкости и широким возможностям настройки. Этот редактор поддерживает множество расширений, которые значительно упрощают процесс написания и отладки кода. Расширения "Python" и "Mypy Type Checker" от Microsoft предоставляют полноценную

поддержку языка Python и статическую проверку типов, что особенно полезно при разработке сложных систем.

2. Poetry был выбран в качестве менеджера зависимостей для управления библиотеками и пакетами Python. Он предоставляет удобный интерфейс для установки, обновления и удаления зависимостей, а также обеспечивает совместимость версий пакетов. Poetry также упрощает процесс создания и публикации проектов, что важно для поддержания чистоты и порядка в проекте.

3. Для создания графического пользовательского интерфейса (GUI) выбран фреймворк DearPyGui [24]. Этот инструмент предлагает простой и интуитивно понятный интерфейс для разработки интерактивных приложений с визуальными элементами. DearPyGui поддерживает создание сложных пользовательских интерфейсов с минимальными усилиями, что позволяет сосредоточиться на разработке функционала, а не на технических деталях реализации GUI.

4. Для упрощения процесса обучения моделей глубокого обучения и сокращения избыточного кода был выбран PyTorch Lightning [25]. Эта библиотека предоставляет высокоуровневый интерфейс для PyTorch, который абстрагирует многие детали обучения и валидации моделей. PyTorch Lightning позволяет разработчикам сосредоточиться на архитектуре модели и ее улучшении, не отвлекаясь на рутинные задачи, связанные с процессом обучения.

5. Для версионирования и отслеживания изменений в экспериментах была выбрана система ClearML [26]. Она предоставляет удобный интерфейс для управления экспериментами, отслеживания данных и результатов, а также интеграции с другими инструментами. ClearML облегчает работу с большими объемами данных и множеством экспериментов, что особенно важно в контексте разработки и тестирования моделей глубокого обучения.

6. Для работы с тестовыми данными были интегрированы датасеты из библиотеки torchvision.datasets. Эта библиотека предоставляет доступ к

широкому спектру стандартных наборов данных для машинного обучения, что упрощает процесс обучения и тестирования моделей. torchvision.datasets позволяет быстро и легко загружать и обрабатывать данные, необходимые для обучения нейронных сетей.

7. Для создания и обучения моделей глубокого обучения используются слои из модуля torch.nn библиотеки PyTorch [27]. Этот модуль предоставляет обширный набор инструментов для разработки различных типов нейронных сетей, от простых до сложных архитектур. PyTorch является одним из самых популярных фреймворков для глубокого обучения, что гарантирует доступ к широкой поддержке со стороны сообщества и обилию ресурсов для обучения и решения возникающих проблем.

8. Для аугментации данных используются методы трансформации из библиотеки torchvision.transforms.v2. Эти методы позволяют изменять входные изображения для улучшения обучения моделей, создавая разнообразные варианты данных. Аугментация данных способствует повышению обобщающей способности моделей, что является критически важным для достижения высоких результатов в задачах глубокого обучения.

9. Для поддержки разметки Markdown в графическом пользовательском интерфейсе использовалась библиотека DearPyGui-Markdow [28]. Она позволила улучшить визуальное оформление и структурирование текста внутри приложения, а также обеспечила гибкость в представлении текстовой информации, включая форматирование заголовков, списков, ссылок и выделенного текста.

Выбор вышеописанных инструментов был продиктован их функциональностью, надежностью и удобством использования. Они обеспечивают полноценную поддержку всех этапов разработки системы, начиная от создания и настройки моделей нейронных сетей, заканчивая их обучением и тестированием.

2.3 Определение входных и выходных данных

2.3.1 Входные данные

Для эффективного функционирования учебного комплекса, направленного на обучение глубокому обучению, было определено, какие данные будут вводиться в систему:

1. Обучающие данные в виде изображений для задачи классификации.
2. Метаданные, в виде меток классов для задач классификации.
3. Гиперпараметры обучения, включающие скорость обучения, максимальное количество эпох, размер пакета данных, оптимизатор обучения, функция потерь.
4. Методы предобработки данных, с числе которых масштабирование и конвертация типа данных.
5. Методы аугментации данных, включающие конвертацию данных к одному каналу, вращение, сдвиг, срез и цветовую коррекцию.

2.3.2 Выходные данные

Выходные данные учебного комплекса включают в себя:

1. Обученные веса нейронной сети, сохраненные в виде .pth файла.
2. Метрики качества модели, такие как значение функции потерь и значение точности.
3. Журнал экспериментов, включающий название проекта, задачи, гиперпараметры обучения и графики функции потерь и точности на обучающей и валидационной выборках.

3 ПРОЕКТИРОВАНИЕ

В данной работе использован системный подход к проектированию учебного комплекса. Раздел включает в себя описание диаграммы прецедентов, последовательности, макеты интерфейса и диаграммы классов. Для структурирования и визуализации компонент системы была выбрана нотация UML (Unified Modeling Language).

Диаграмма прецедентов была разработана для идентификации и описания основных сценариев использования системы. Для отображения динамики взаимодействий между объектами системы в ходе выполнения ключевых процессов были созданы диаграммы последовательности. Помимо этого были разработаны макеты интерфейса, которые отражают структуру приложения и расположение основных элементов управления. Диаграммы классов, в свою очередь, были спроектированы для отображения статической структуры системы.

3.1 Диаграмма прецедентов

Диаграмма прецедентов, представленная на рисунке А.1 в приложении А, описывает функциональные возможности учебного комплекса, а также взаимодействие пользователя с системой. Рассмотрим каждый элемент диаграммы и его роль:

1. Выбор датасета предоставляет пользователю возможность выбрать одну из предустановленных датасетов для классификации изображений. После размещения датасета в рабочей области функционал включает:

   а. Настройку параметров аугментации, которая позволяет улучшить обобщающую способность модели. Параметры аугментации включают в себя различные трансформации данных, такие как: поворот кадра, изменение размера, яркости и контраста.

b. Выбор размера пакета, который влияет на скорость и стабильность обучения.

2. Управление обучением сети:

a. Позволяет запустить процесс обучения модели на выбранных данных с заданными параметрами.

b. Позволяет дообучить сеть, возможно, после изменения некоторых гиперпараметров или тонкой настройки слоев сети.

c. Позволяет прервать обучение, если настроенная конфигурация перестала удовлетворять требованиям.

d. Включает настройку гиперпараметров обучения, которая позволяет пользователю задать такие параметры, как скорость обучения, максимальное количество эпох, используемый оптимизатор обучения, функцию потерь, метод предварительной инициализации весов, а также название проекта и задачи для идентификации обучения в системе журналирования.

e. Дает возможность скачать обученные веса модели для дальнейшего использования или анализа.

f. Дает возможность загрузить заранее обученные веса модели, что может ускорить процесс обучения и улучшить результаты.

3. Отображение сообщений в журнале включает параметры собранной и инициализированной архитектуры сети, а также размерность пакета данных, проходящего через каждый слой сети. Переход в систему CLearML расширяет данную функцию, позволяя пользователю мониторить процесс обучения по части метрик модели.

4. Сохранение архитектуры позволяет зафиксировать текущую конфигурацию узлов из рабочей области в файл с расширением .json для последующего использования в другой сессии запуска приложения.

5. Развертывание архитектуры подразделяется на:

a. Выбор готовой архитектуры, предоставляя пользователю возможность выбрать одну из предустановленных нейросетевых архитектур для классификации изображений.

b. Построение архитектуры вручную, позволяя пользователю создать архитектуру модели из базовых слоев самостоятельно.

c. Загрузка архитектуры из файла позволяет загрузить заранее сохраненную конфигурацию узлов из файла с расширением .json.

d. Возможность тонкой настройки слоев, которая позволяет сконфигурировать все доступные параметры каждого слоя предустановленной, составленной вручную или загруженной из файла архитектуры. При этом настройка свернутой или неинициализированной предустановленной архитектуры осуществляется в модальном окне, не поддерживающем функционал добавления слоев.

e. Возможность разворачивания предустановленной архитектуры до базовых элементов, позволяя модифицировать ее, добавляя новые слои.

В качестве основного актора выступает пользователь, который взаимодействует с системой. Пользователь может выбирать датасеты, управлять процессом обучения, настраивать гиперпараметры, сохранять и загружать архитектуры, а также просматривать журнал обучения. Второстепенным актором, с которым взаимодействует пользователь, выступает система мониторинга метрик и результатов обучения ClearML.

Далее были описаны основные взаимодействия, предусмотренные в данной системе:

- Пользователь начинает с выбора датасета, после чего он может настроить параметры аугментации и размер пакета данных.
- Пользователь разворачивает сеть одним из возможных способов: выбрав готовую архитектуру, составив архитектуру вручную, либо же загрузив ее из файла. В последующем пользователь в праве тонко настроить слои архитектуры и даже добавить новые. Причем готовую архитектуру можно развернуть до базовых элементов для детальной настройки и модификаций.
- Пользователь может сохранить архитектуру модели в файл для последующего использования.

- После построения сети пользователь настраивает гиперпараметры обучения.
- Процесс управления обучением условно можно разделить на три сценария:
- Пользователь сразу запускает обучение.
- Если узлы архитектуры были загружены из файла, пользователь может запустить обучение или загрузить предобученные веса модели и дообучить сеть.
- Если метрики качества во время обучения неудовлетворительны или в архитектуре сети были замечены ошибки, пользователь может прервать обучение сети, исправить вручную ошибки и заново запустить обучение модели.
- По завершении обучения пользователь скачивает веса модели.
- Пользователь может просматривать журнал инициализации сети и переходить в систему ClearML для более детального анализа обучения.

В данном подразделе была представлена и проанализирована диаграмма прецедентов системы. Диаграмма отразила основные сценарии использования, такие как выбор датасета, настройка аугментации, запуск и управление обучением, а также просмотр журнала.

3.2 Диаграмма последовательности

Диаграммы последовательности, представленные на рисунках Б.1-Б.2 приложения Б, были спроектированы для визуализации взаимодействия между различными компонентами системы во временном контексте. В данном подразделе будут рассмотрены диаграммы последовательности, включающие такие объекты, как пользовательский интерфейс (UI), блок управления обучением, Pipeline, Trainer и система мониторинга ClearML.

Процесс начинается с выбора пользователем датасета. Пользователь инициирует взаимодействие с UI, выбирая необходимый датасет из

представленного списка. Затем UI передает информацию блоку управления обучением для инициализации. После этого UI отображает пользователю блок датасета и блок управления обучением, позволяя ему настроить параметры и аугментацию датасета, а также гиперпараметры обучения.

Далее пользователь может начать процесс построения архитектуры модели. Для этого он последовательно размещает слои архитектуры через UI, который отображает каждый добавленный слой. Пользователь также имеет возможность настроить параметры каждого слоя. В альтернативном сценарии пользователь может выбрать готовую архитектуру. При этом UI отображает окно настройки архитектуры, где пользователь может внести необходимые изменения в слои. После завершения настройки UI отображает конечную архитектуру.

После завершения настройки архитектуры пользователь запускает процесс обучения. Это инициирует взаимодействие с Pipeline, который отвечает за сборку всех слоев из рабочей области и их инициализацию. Pipeline инициализирует датасет, а затем отображает параметры датасета в журнале пользователя. Следующим шагом Pipeline инициализирует сеть и отображает параметры слоев инициализированной сети в журнале пользователя, а также размерность пакетов данных.

Для отслеживания процесса обучения Pipeline устанавливает соединение с сервером ClearML. После успешного подключения система возвращает ссылку на ClearML, которая отображается в журнале пользователя. Затем Pipeline инициализирует тренера (Trainer), который непосредственно занимается процессом обучения модели. В ходе обучения Trainer вычисляет метрики, такие как функция потерь и точность на тренировочных и валидационных данных. Эти метрики затем отправляются на сервер ClearML для мониторинга.

Таким образом, диаграммы последовательностей демонстрируют, как пользователь, взаимодействуя с интерфейсом, управляет процессом

подготовки данных, настройки и обучения модели, и как различные компоненты системы координируют свои действия для достижения цели.

3.3 Макеты интерфейса

В данном подразделе рассматриваются макеты пользовательского интерфейса, представленные на рисунках В.1-В.4 приложения В. Основное внимание уделялось организации элементов интерфейса, их расположению и функциональному назначению.

Интерфейс включает четыре основных контейнера, которые размещены по бокам рабочего окна. Левый верхний контейнер предназначен для работы с датасетами, левый нижний – для выбора готовых архитектур, правый верхний – для выбора слоев нейронной сети, правый нижний – для выбора функций активации. Центральная часть интерфейса отведена под рабочую область, где пользователь может собирать архитектуру нейронной сети, используя доступные элементы.

Меню интерфейса содержит три основных раздела: "Файл", "Настройки" и "Сообщения". Раздел "Файл" включает опции сохранения и загрузки проекта, а также закрытие программы. В разделе "Настройки" пользователь может включать или отключать логирование, а также очищать всю рабочую область. Раздел "Сообщения" представляет собой ниспадающее окно для просмотра журнала событий и сообщений системы.

Цветовая схема интерфейса была выбрана в соответствии с принципами удобства восприятия и эстетической привлекательности. Фон контейнеров имеет цвет #DAE8FC. Кнопки выполнены в цвете #E8E4DC. Раскрывающиеся заголовки имеют цвет #F5F5F5. Ползунки выполнены в цвете #D6D6D6. Остальная часть интерфейса соответствует светлой цветовой схеме.

Дополнительные функциональные элементы включают всплывающие подсказки и справки. При наведении курсора на параметры отображается краткая информация о них, что помогает пользователю быстро

ориентироваться в настройках. При правом щелчке мыши на кнопки в контейнерах, например, на одну из готовых архитектур, отображается более полная справка, как это показано на рисунке В.2 . Эта справка включает подробное описание и задания, что представляет из себя обучающий аспект учебного комплекса.

Удаление объектов из рабочей области осуществляется посредством активации элемента с последующим нажатием кнопки Del. У каждого блока, инициализированного в рабочей области, есть точки входа и выхода, которые необходимо соединять линией для правильного выстраивания потока данных. Перемещение по рабочей области осуществляется с помощью нажатого колесика мыши. Узлы можно перемещать, удерживая левую клавишу мыши и перетаскивая за заголовок блока.

3.4 Диаграммы классов

В данном подразделе рассматривается диаграмма классов, разработанная с использованием UML.

При проектировании системы был использован шаблон проектирования "фабричный метод", который является порождающим шаблоном проектирования, предоставляющий дочерним подклассам интерфейс для создания экземпляров этих классов. В момент создания наследники могут определить, какой класс создавать. Иными словами, данный шаблон делегирует создание объектов наследникам родительского класса. Это позволяет использовать в коде программы не конкретные классы, а манипулировать абстрактными объектами на более высоком уровне

В результате проектирования классов, отвечающих за интерфейс системы, была получена диаграмма классов, представленная на рисунке Г.1 приложения Г. Детальное описание атрибутов, методов и назначений интерфейсных классов приведено в таблице Г.4 приложения Г.

На следующем шаге были определены ресурсные классы, диаграмма которых представлена на рисунке Г.2 приложения Г. Описание атрибутов и методов соответственно приведено в таблице Г.5 приложения Г.

На завершающем этапе проектирования была составлена диаграмма классов, представленная на рисунке Г.3 приложения Г, отвечающая за процесс сбора данных с рабочей области и последующей инициализации и обучения модели. Подробное описание атрибутов, методов и назначений данных классов приведено в таблице Г.6 приложения Г.

4 КОДИРОВАНИЕ

4.1 Конфигурирование инструментов разработки

При подготовке среды разработки для проекта на операционной системе Arch Linux с использованием пакетного менеджера yay первым делом были актуализированы все версии пакетов, установленных на текущий момент в операционной системе.

Далее, для установки Visual Studio Code с помощью yay была выполнена следующая команда:

yay -S visual-studio-code-bin

Так как системе уже была установлена версия python 3.11, после установки IDE необходимо было только добавить расширения для обеспечения полноценной поддержки Python и статической проверки типов. Для этого в командной строке были выполнены следующие команды:

code --install-extension ms-python.python

code --install-extension ms-python.vscode-pylance

Для управления зависимостями и проектами Python был выбран Poetry. Для установки Poetry была выполнена следующая команда:

curl -sSL https://install.python-poetry.org | python3 -

После установки был создан новый проект под названием dlc с помощью команды: poetry new dlc. Далее в файл pyproject.toml по мере расширения проекта добавлялись библиотеки, устанавливаемые следующей командой:

poetry add new_package

Тогда как актуализация библиотек в соответствии с редактируемым файлом pyproject.toml, представленным в листинге Д.1 приложения Д, осуществлялась посредством команды:

poetry lock

Разберем подробнее, какие библиотеки было необходимо установить для разработки учебного комплекса.

Для упрощения процесса обучения моделей и сокращения избыточного кода была выбрана библиотека PyTorch Lightning. Для ее установки была выполнена следующая команда:

poetry add lightning

Для версионирования и отслеживания изменений в экспериментах была выбрана система ClearML. Для установки ClearML выполнялась следующая команда:

poetry add clearml

Сразу после установки ClearML необходимо было настроить SDK посредством команды clearml-init. Переходя по предложенной ссылке, clearml предлагает войти или зарегистрироваться, после чего необходимо было создать API-ключ и вставить его в командную строку. Наконец конфигурация, представленная в листинге Д.2 приложения Д, сохранилась в файл /home/luzinsan/clearml.conf.

Для работы с аугментацией данных необходимо было установить библиотеку torchvision посредством команды:

poetry add torchvision

Эти шаги позволили создать полноценную среду для дальнейшей комфортной разработки учебного комплекса.

4.2 Описание программной реализации

Всего в проекте было реализовано три пакета: core, nodes и app.

В пакете nodes описаны классы, которые представляют собой различные узлы, используемые в системе. Каждый класс соответствует определенному типу узла и содержит функциональность, связанную с этим типом. Например, класс DataNode представляет узел данных, который может использоваться для загрузки и предобработки данных. Аналогично, классы LayerNode, ModuleNode и TrainParamsNode представляют узлы, связанные со слоями, модулями и параметрами обучения соответственно.

В пакете core содержатся классы, которые являются основными элементами системы. Классы DragSourceContainer, DragSource, InputNodeAttribute, LinkNode, NodeEditor, Node, OutputNodeAttribute и ParamNode представляют различные компоненты и функциональность, связанную с визуальным редактором узлов. Файл utils.py, также входящий в пакет core, содержит вспомогательные функции и утилиты, которые используются в различных частях системы для обеспечения ее работоспособности.

Пакет app является сердцем системы, и в нем содержатся классы, связанные с обработкой данных и созданием датасетов для обучения моделей. Класс DataModule является наследником от PyTorch Lightning и предоставляет интерфейс для загрузки и предобработки данных. Класс Module также является наследником от PyTorch Lightning и представляет собой базовый класс для всех моделей в системе. Класс App является центральным классом системы и инициализирует все слои, модули и другие компоненты, необходимые для построения и обучения моделей глубокого обучения. Класс Pipeline отвечает за сборку слоев в последовательность для обучения моделей и подключение к эксперименту clearml. Здесь же происходит инициализация и запуск обучения моделей.

Предварительная загрузка ресурсов, то есть ссылок на классы, реализующие ту или иную функциональность, осуществляется в конструкторе класса App. Параметр, отвечающий за хранение ресурса, в каждом классе именуется как data. За передачу параметров, которые будут использованы при инициализации любого узла, отвечает параметр params. Непосредственно фабричный метод подгружается из файла, который содержит словарь со всей необходимой информацией для инициализации. В свою очередь параметр default_params используется классом, генерирующим узел, по-своему. Наконец, параметр node_params используется уже непосредственно во время рендеринга узла.

Таким образом, после создания контейнера (класс DragSourceContainer), который будет содержать ресурсы, у него вызывается метод add_drag_source, куда передаются экземпляры класса DragSource. Но перед этим создаётся словарь ресурсов, значениями которого являются экземпляры класса DragSource. Во время рендеринга контейнера ресурсов, т. е. вызова метода submit, вызывается одноименный метод у каждого ресурса, входящего в этот контейнер. Уже в самом методе генерируется элемент кнопки, где запоминаются параметры в методе dpg.drag_payload(), передаваемые при перетаскивании ресурса в рабочую область.

Важным моментом является рендеринг рабочей среды, т. е. закрепление экземпляра класса NodeEditor на главный экран. Данная процедура также осуществляется в методе submit(), где вызывается менеджер dpg.child_window, в котором в аргумент drop_callback передаётся лямбда функция, вызывающая сцепленные функции dpg.get_item_user_data(s).on_drop(s,a,u). Смысловая нагрузка данной конструкции заключается в том, что, при попадании в рабочую область, в ней будет вызван метод on_drop(), в который будет передана вся информация, указанная ранее в dpg.drag_playload(). Следует также обозначить, что сам экземпляр рабочей области создаётся вызовом функции dpg.node_editor, в качестве аргументов callback и delink_callback которого указываются статические методы LinkNode.link_callback и LinkNode.delink_callback.

4.3 Руководство администратора

Под администратором системы в данном случае подразумевается лицо, которое разворачивает учебный комплекс для учащихся, а также пользователь, интегрирующий систему для своих нужд с целью ее модификации.

Для корректной работы учебного комплекса требуется соблюдение следующих системных характеристик:

- Операционная система: Windows 10 и выше; дистрибутивы на основе ядра Linux (например, Ubuntu, Arch Linux) или MacOS.
- Современный процессор с поддержкой многопоточности (например, Intel i5 и выше).
- Оперативная память: минимум 8 ГБ ОЗУ (рекомендуется 16 ГБ и выше).
- Графический процессор (рекомендуется), поддерживающий CUDA (например, NVIDIA GTX 1050 или эквивалентный).
- Минимум 10 ГБ свободного дискового пространства для установки и работы приложения.
- Постоянное подключение к интернету для использования системы мониторинга ClearML

Администратор может модифицировать программу в соответствии со своими потребностями, поскольку программное обеспечение является открытым. Для этого потребуется любая интегрированная среда разработки IDE. Процесс клонирования и установки репозитория выглядит следующим образом:

1. Откройте терминал и выполните команду для клонирования репозитория: git clone https://github.com/Luzinsan/Neural-Network-Constructor
2. Перейдите в папку проекта: cd Neural-Network-Constructor
3. Установите менеджер зависимостей Poetry:
    - Arch Linux: sudo pacman -S poetry
    - Ubuntu: curl -sSL https://install.python-poetry.org | python3 -
    - Для Windows скачайте и установите пакет с сайта [29]
    - macOS: curl -sSL https://install.python-poetry.org | python3 -
4. Находясь в папке проекта, выполните команду для установки всех зависимостей проекта, указанных в файле pyproject.toml: poetry install
5. Выполните команду для инициализации ClearML: clearml-init

6. Пройдите регистрацию и авторизацию на платформе, перейдя по ссылке, выведенной в консоли. Сгенерируйте API ключ для доступа к профилю.

7. Вставьте в консоль в качестве ответа сгенерированный API ключ.

После установки всех зависимостей и инициализации ClearML, администратор может запустить программу для тестирования работоспособности: poetry run python main.py

Если программа запускается корректно, администратор может приступить к модификации кода в соответствии с требуемыми изменениями. Инструкция по установке и дополнительная информация также размещены в репозитории [30] на GitHub в файле README.md.

4.4 Руководство пользователя

В этом руководстве описаны основные функции приложения и процесс работы с ним.

1. Меню:

a) Файл:

- Сохранить – записать в файл с расширением .json все узлы, размещенные в рабочей области.
- Открыть – загрузить в рабочую область узлы из файла с расширением .json.
- Сбросить – удалить все узлы из рабочей области.

b) Настройки:

- Включить/Отключить логирование
- Открыть инструменты для: просмотра метрик, отладки кода, просмотра документации Dearpygui и других. Данные инструменты полезны для администратора и разработчиков.

c) Сообщения – ниспадающее окно для просмотра журнала

2. Область ресурсов:

a) Левый верхний контейнер – список доступных датасетов

b) Левый нижний контейнер – список готовых архитектур

c) Правый верхний контейнер – список слоев

d) Правый нижний контейнер – список функций активации

3. Работа с блоками:

a) Выберите датасет из левого верхнего контейнера. Перетащите кнопку с названием датасета в рабочую область. При этом автоматически генерируется блок для управления обучением, как показано на рисунке Е.1 приложения Е.

b) Выберите слой из правого верхнего контейнера и перетащите его в рабочую область.

c) Подключите выходной узел датасета к входному, как показано на рисунке Е.2 приложения.

d) Чтобы удалить узел, выберите его и нажмите клавишу “Del”

e) Чтобы импортировать узлы, перейдите в меню “Файл” и выберите “Открыть”. В появившемся окне выберите необходимый файл на нажмите “ОК”, как показано на рисунке Е.3 приложения Е.

f) Чтобы экспортировать узлы, перейдите в меню “Файл” и выберите “Сохранить”. В появившемся окне напишите название файла и нажмите на “ОК”.

g) Перемещение узлов осуществляется посредством перетаскивания элементов за заголовок с помощью левой кнопки мыши.

h) Навигация по рабочей области осуществляется с нажатым колесиком мыши.

i) Краткая информация о параметрах показывается при наведении на них, на показано на рисунке Е.4 приложения Е. Полная справка разворачивается при правом щелчке на кнопки, что продемонстрировано на рисунке Е.5 приложения Е.

j) При попытке размещения готовой архитектуры появляется окно конфигурирования модуля, представленное на рисунке Е.6 приложения Е, в

котором можно настроить параметры слоев, но не их количество. Это доступно только при разворачивании модели до простейший элементов. Переключатель “Свернуть модель” выставлен по-умолчанию. Разместив свернутую модель в рабочей области, позже можно беспрепятственно развернуть модель и добавлять дополнительные слои.

4. Запуск обучения:

a) Настройте параметры аугментации изображения, перечисленные в блоке датасета.

b) Настройте гиперпараметры в блоке управления обучением.

c) Нажмите кнопку “Запустить обучение”. Подождите примерно 5 секунд и проверьте журнал, чтобы удостовериться, что сеть построена правильно. Перейдите по ссылке из журнала в рабочее пространство ClearML.

Следуя данному руководству, пользователи смогут эффективно работать с учебным комплексом, проектируя и обучая нейронные сети.

5 ТЕСТИРОВАНИЕ

Тестирование программного обеспечения является важным этапом разработки, направленным на проверку корректности работы системы, выявление и устранение дефектов, а также на обеспечение соответствия системы заявленным требованиям. В данном разделе описываются методология тестирования, процедура его проведения, примеры тестовых сценариев и полученные результаты. Тестирование было выполнено вручную, что позволило глубже понять поведение системы и выявить возможные проблемы на ранних стадиях. Подробное описание методологии, используемых процедур и примеры тестовых сценариев помогут обеспечить репликабельность и эффективность тестирования для последующих итераций разработки и улучшения системы.

5.1 Описание выбранной методологии тестирования

Для тестирования программного обеспечения, разработанного в рамках данного проекта, была выбрана ручная методология тестирования. Данная методология предполагает проведение тестов вручную без использования автоматизированных тестовых инструментов. Выбор в пользу ручного тестирования был обусловлен несколькими ключевыми факторами:

1. Программа обладает сложным и интерактивным интерфейсом, включающим визуальные элементы для проектирования нейронных сетей. Ручное тестирование позволяет тщательно проверить взаимодействие пользователя с интерфейсом и убедиться в правильности отображения и функционирования всех элементов.

2. Поскольку ручное тестирование выполняется человеком, оно позволяет лучше понять и выявить проблемы, с которыми могут столкнуться конечные пользователи. Этот аспект включил в себя проверку удобства

использования, интуитивности интерфейса и общего пользовательского опыта.

3. На текущем этапе разработки, с учетом ограниченности ресурсов и времени, внедрение и поддержка автоматизированных тестов были признаны нецелесообразными.

В рамках ручного тестирования была использована следующая стратегия:

- В рамках функционального тестирования была проведена проверка соответствия системы заявленным функциональным требованиям. Тестировались основные функции приложения, такие как выбор датасета, настройка параметров, инициализация обучения и мониторинг метрик.
- В интерфейсном тестировании проверялась корректность работы пользовательского интерфейса, включая отображение всех визуальных элементов, интерактивные действия и навигация по приложению.
- После внесения изменений в код проводилось повторное тестирование функциональности, чтобы убедиться, что новые изменения не вызвали неожиданных ошибок в ранее работавших модулях.

Использование ручной методологии тестирования обеспечило высокое качество и надежность разработанного программного обеспечения, позволяя быстро выявлять и устранять дефекты, а также улучшать пользовательский опыт.

5.2 Описание процедуры тестирования

Процедура тестирования разработанного программного обеспечения выполнялась вручную. В процессе тестирования особое внимание уделялось проверке корректности работы различных архитектур нейронных сетей, доступных в системе, а также их сопоставлению с теоретическими ожиданиями с ресурса "Dive into Deep Learning" в качестве ориентира для построения и проверки нейронных сетей. Данный ресурс предоставлял

необходимую теоретическую базу по глубокому обучению, что позволяло точно оценить корректность работы реализованных архитектур.

Основной тестовой базой данных был датасет FashionMNIST [31], который широко используется в исследованиях глубокого обучения и хорошо подходит для оценки производительности различных архитектур нейронных сетей.

В процессе тестирования создавались и проверялись архитектуры нейронных сетей, такие как LeNet, AlexNet, VGG, NiN, GoogLeNet и Batch Normalization LeNet. Каждая архитектура была тщательно настроена в системе, включая настройку размера обучаемых изображений и параметров обучения.

Целью тестирования было удостовериться, что каждая из протестированных архитектур функционирует корректно и достигает ожидаемых результатов. Процедура тестирования включала инициализацию датасета, настройку параметров и аугментации, выбор и настройку архитектуры, а также запуск процесса обучения.

На каждом этапе тестирования тщательно проверялись все визуальные и функциональные элементы пользовательского интерфейса, включая корректное отображение слоев сети, размеры пакетов данных и работу системы мониторинга ClearML. Кроме того, проверялись функции удаления и перемещения объектов, а также правильное соединение точек входа и выхода слоев для обеспечения корректного потока данных.

5.3 Описание тестовых сценариев

Тестовые сценарии включали в себя множество аспектов проверки корректности работы системы, начиная с выбора и настройки датасетов и заканчивая обучением различных архитектур нейронных сетей. Каждый тестовый сценарий предполагал определенную последовательность действий, направленную на выявление и устранение возможных ошибок и недочетов.

Первый сценарий начинался с выбора датасета FashionMNIST. Проверялось, корректно ли система загружает и отображает датасет, а также насколько легко пользователю настроить параметры аугментации данных. Важным аспектом было убедиться, что параметры аугментации (такие как вращение, масштабирование и отражение) применяются корректно.

Второй тестовый сценарий включал проверку инициализации различных архитектур нейронных сетей. Каждая архитектура, начиная с простых многослойных перцептронов (MLP) и заканчивая сложными моделями, такими как GoogLeNet и Batch Normalization LeNet, была последовательно настроена и инициализирована. В ходе тестирования проверялось, правильно ли отображаются слои сети в рабочей области, корректно ли соединены точки входа и выхода, и отображаются ли все параметры слоев.

Третий сценарий фокусировался на настройке гиперпараметров для каждой архитектуры. Проверялось, насколько удобно пользователю задавать параметры, такие как количество эпох, размер батча, скорость обучения и параметры регуляризации. Важно было убедиться, что все изменения параметров корректно сохраняются и применяются при запуске процесса обучения.

Четвертый сценарий включал непосредственный запуск процесса обучения. Проверялось, корректно ли инициализируются датасеты и сети, отображаются ли параметры в журнале, и успешно ли устанавливается соединение с сервером ClearML. Во время обучения проверялось, корректно ли вычисляются и отображаются метрики, такие как функция потерь и точность, и соответствуют ли они ожиданиям, основанным на данных ресурса "Dive into Deep Learning".

Пятый сценарий включал проверку системы мониторинга ClearML. Проверялось, правильно ли система передает данные метрик на сервер ClearML, и корректно ли отображаются результаты на платформе. Важно было

убедиться, что метрики обучения и валидации отображаются в реальном времени и позволяют пользователю отслеживать процесс обучения.

5.4 Описание полученных результатов тестирования

Во время тестирования основной упор ставился на корректность обучения различных архитектур и сравнение показателей метрик с результатами, полученными на ресурсе “Dive into Deep Learning”. В связи с этим, на датасете FashionMNIST были выполнены следующие эксперименты и получены результаты:

1. Архитектура LeNet, развернутая сеть которой представлена на рисунке Ж.1 приложения Ж, хоть и считается классической, проявила себя достаточно хорошо на данном датасете. Она продемонстрировала хорошую обобщающую способность и устойчивость к переобучению, что видно на рисунке Ж.2 приложения Ж.

2. Архитектура AlexNet, реализованная в конструкторе, представлена на рисунке Ж.3 приложения Ж. Её обучение заняло намного больше времени, тогда как результаты несильно отличаются от результатов LeNet, что можно посмотреть на рисунке Ж.4 приложения Ж.

3. Архитектура VGG, представленная на рисунке Ж.5 приложения Ж, показала впечатляющие результаты в задаче классификации FashionMNIST. Ее глубокая структура позволила извлечь более сложные признаки из изображений, что привело к высокой точности классификации, что можно увидеть на рисунке Ж.6 приложения Ж.

4. Модель NiN, представленная на рисунке Ж.7 приложения Ж, хоть и имеет более нестандартную структуру, демонстрирует неплохую производительность, что можно понять из рисунке Ж.8 приложения Ж. Ее использование позволяет извлекать более абстрактные признаки из изображений.

5. Архитектура GoogLeNet, реализованная в конструкторе и представленная на рисунке Ж.9 приложения Ж, известная своей глубокой и сложной структурой, показала превосходные результаты обучения, которые можно увидеть на рисунке Ж.10 приложения Ж. Ее использование привело к высокой точности классификации и устойчивости к переобучению.

6. Архитектура LeNet с применением метода Batch Normalization, представленная на рисунке Ж.11 приложения Ж, показала улучшение как по скорости обучения, так и по точности классификации. Этот метод позволяет стабилизировать процесс обучения и ускорить сходимость модели. Обучение модели представлено на рисунке Ж.12 приложения Ж.

В заключение, результаты тестирования архитектур нейронных сетей на датасете FashionMNIST в учебном комплексе соответствуют ожиданиям, представленным в ресурсе "Dive into Deep Learning". Это свидетельствует о том, что разработанная система успешно реализует и воспроизводит концепции и методы глубокого обучения, представленные в академической литературе. Точность классификации и производительность моделей на практике сопоставимы с ожидаемыми результатами, что подтверждает эффективность выбранных архитектур и подходов к обучению.

Заключение

В ходе выполнения выпускной квалификационной работы были выполнены все поставленные цели и задачи, а также закреплены профессиональные компетенции, полученные за весь период обучения.

В результате проведенного анализа требований были выявлены проблемные аспекты, возникающие в процессе освоения сферы глубокого обучения, а также был проведен краткий экскурс в сферу сверточных нейронных сетей для классификации изображений. Помимо этого, в рамках анализа были сформулированы функциональные требования и проведен обзор аналогов, который также показал актуальность разработки в связи с дефицитом русскоязычных учебных комплексов подобного плана.

Далее были определены спецификации системы, включающие выбор спиральной методологии разработки и определение входных и выходных данных. Здесь же был приведен обзор выбранных инструментов разработки, включающий традиционный стек специалиста по работе с данными: Visual Studio Code, Python, Poetry, Pytorch Lightning, и в дополнении, ClearML.

Следующим этапом выступило проектирование, в ходе которого были составлены диаграммы прецедентов, последовательностей и классов, а также визуализированы макеты интерфейса.

На этапе кодирования были реализованы все диаграммы классов. Помимо этого, были описаны руководства администратора и пользователя, а также конфигурирование инструментов разработки, что обеспечило полную документацию по развертыванию системы.

В конце каждой спирали разработки проводилось тестирование программы, в котором проверялись все реализованные модули. Особое внимание уделялось реализованным архитектурам: LeNet, AlexNet, VGG, NiN, GoogLeNet и BN LeNet. Они продемонстрировали корректность работы системы и ее способность обеспечивать стабильное обучение на примере датасета FashionMNIST для классификации изображений.

Также стоит упомянуть, что во время исследования была подготовлена статья по данной тематике: «Прототип программного обеспечения для визуального конструирования нейронных сетей на основе принципов Blueprint» (принята к публикации) [21].

В последствии система может интегрироваться в образовательные и исследовательские учреждения для обучения и тренировки моделей глубокого обучения.

Дальнейшие планы по развитию работы включают расширение функционала системы за счет добавления новых типов слоев и архитектур нейронных сетей, улучшение интерфейса пользователя, а также возможность полноценного тестирования моделей на новых данных из разных ресурсов, интеграцию дополнительных инструментов для автоматизации и оптимизации процесса обучения.

В целом, проведенная работа демонстрирует высокий уровень профессионализма и значительный вклад в развитие инструментов для машинного обучения, предлагая инновационное и удобное решение для конструирования нейронных сетей.

Список использованных источников

1. Top 10 Deep Learning Frameworks [Электронный ресурс]: портал ресурса phoenixnap. URL: https://phoenixnap.com/blog/deep-learning-frameworks (дата обращения: 03.06.2024).

2. About Us [Электронный ресурс]: информационная страница Jupyter. URL: https://jupyter.org/about (дата обращения: 03.06.2024).

3. Сквозные технологии НТИ [Электронный ресурс]: портал НТИ. URL: https://nti2035.ru/technology/ (дата обращения: 03.06.2024).

4. Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions [Электронный ресурс]: журнал National Library of Medicine. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8372231/ (дата обращения: 03.06.2024).

5. On Definition of Deep Learning / Zhang, W.C., Yang, G., Lin, Y., Ji, C. // World Automation Congress, 2018, стр. 1-5 [Электронный ресурс]: электр. журн. URL: https://ieeexplore.ieee.org/document/8430387 (дата обращения: 03.06.2024).

6. Rectified Linear Units Improve Restricted Boltzmann Machines / Vinod Nair, Geoffrey E. Hinton // Department of Computer Science, University of Toronto [Электронный ресурс]: электр. журн. URL: https://www.cs.toronto.edu/~fritz/absps/reluICML.pdf (дата обращения: 03.06.2024).

7. The regression analysis of binary sequences / D. R. Cox // Birkbeck College, University of London [Электронный ресурс]: электр. журн. URL: https://www.nuffield.ox.ac.uk/users/cox/cox48.pdf (дата обращения: 03.06.2024).

8. / Kalman, B. L., Kwasny. Why tanh: choosing a sigmoidal function. Proceedings of the International Joint Conference on Neural Networks. IJCNN: 1992. 578–581 стр.

9. Convolutional Neural Networks / Mario V. Wüthrich, Michael Merz // Springer Actuarial. 2022. 407–424 с. [Электронный ресурс]: электр. журн. URL: https://link.springer.com/chapter/10.1007/978-3-031-12409-9_9 (дата обращения: 03.06.2024).

10. Backpropagation Applied to Handwritten Zip Code Recognition / Y.LeCun, B.Boser, J.S.Denker // AT&T Bell Laboratories [Электронный ресурс]: электр. журн. URL: http://yann.lecun.com/exdb/publis/pdf/lecun-89e.pdf (дата обращения: 03.06.2024).

11. ImageNet Classification with Deep Convolutional / Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton // University of Toronto [Электронный ресурс]: электр. журн. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf (дата обращения: 03.06.2024).

12. Very deep convolutional networks for large-scale image recognition/ Karen Simonyan, Andrew Zisserman // Visual Geometry Group, Department of Engineering Science, University of Oxford. 2015 [Электронный ресурс]: электр. журн. URL: https://arxiv.org/pdf/1409.1556 (дата обращения: 03.06.2024).

13. Network in network / Min Lin, Qiang Chen, Shuicheng Yan [Электронный ресурс]: электр. журн. URL: https://arxiv.org/pdf/1312.4400 (дата обращения: 03.06.2024).

14. Going deeper with convolutions / Christian Szegedy, Wei Liu, Yangqing Jia // Google Inc. [Электронный ресурс]: электр. журн. URL: https://arxiv.org/pdf/1409.4842 (дата обращения: 03.06.2024).

15. Teachable Machine [Электронный ресурс]: официальный сайт. URL: https://teachablemachine.withgoogle.com/ (дата обращения: 03.06.2024).

16. KNIME [Электронный ресурс]: официальный сайт. URL: https://www.knime.com/get-started (дата обращения: 03.06.2024).

17. IBM Watson Studio [Электронный ресурс]: официальный сайт. URL: https://www.ibm.com/products/watson-studio (дата обращения: 03.06.2024).

18. Azure Machine Learning [Электронный ресурс]: официальный сайт. URL: https://azure.microsoft.com/en-us/products/machine-learning (дата обращения: 03.06.2024).

19. Lobe [Электронный ресурс]: официальный сайт. URL: https://www.lobe.ai/ (дата обращения: 03.06.2024).

20. Loginom [Электронный ресур]: официальный сайт. URL: https://loginom.ru/skills (дата обращения: 03.06.2024).

21. Прототип программного обеспечения для визуального конструирования нейронных сетей на основе принципов Blueprint / А.А. Лузинсан // Научная сессия ТУСУР. 2024. Томск: В-Спектр.

22. Dive into Deep Learning [Электронный ресурс]: курс по глубокому обучению. URL: https://d2l.ai/index.html (дата обращения: 03.06.2024).

23. Clickup [Электронный ресурс]: официальный сайт. URL: https://clickup.com (дата обращения: 03.06.2024).

24. Dearpygui [Электронный ресурс]: репозиторий библиотеки. URL: https://github.com/hoffstadt/DearPyGui (дата обращения: 03.06.2024).

25. Pytorch Lightning [Электронный ресурс]: официальная документация. URL: https://lightning.ai/docs/pytorch/stable//index.html (дата обращения: 03.06.2024).

26. ClearML [Электронный ресурс]: официальный сайт. URL: https://clear.ml/ (дата обращения: 03.06.2024).

27. Pytorch [Электронный ресурс]: официальная документация . URL: https://pytorch.org/docs/stable/index.html (дата обращения: 03.06.2024).

28. Dearpygui-markdown [Электронный ресурс]: репозиторий проекта. URL: https://github.com/IvanNazaruk/DearPyGui-Markdown (дата обращения: 03.06.2024).

29. Poetry [Электронный ресурс]: инструкция по установке. URL: https://python-poetry.org/docs/#installation (дата обращения: 03.06.2024).

30. Neural-Network-Constructor [Электронный ресурс]: репозиторий текущего проекта. URL: https://github.com/Luzinsan/Neural-Network-Constructor (дата обращения: 03.06.2024).

31. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms / Han Xiao, Kashif Rasul, Roland Vollgraf [Электронный ресурс]: электр. журнал. URL: https://arxiv.org/pdf/1708.07747 (дата обращения: 03.06.2024).

Приложение А
(справочное)
Диаграмма прецедентов

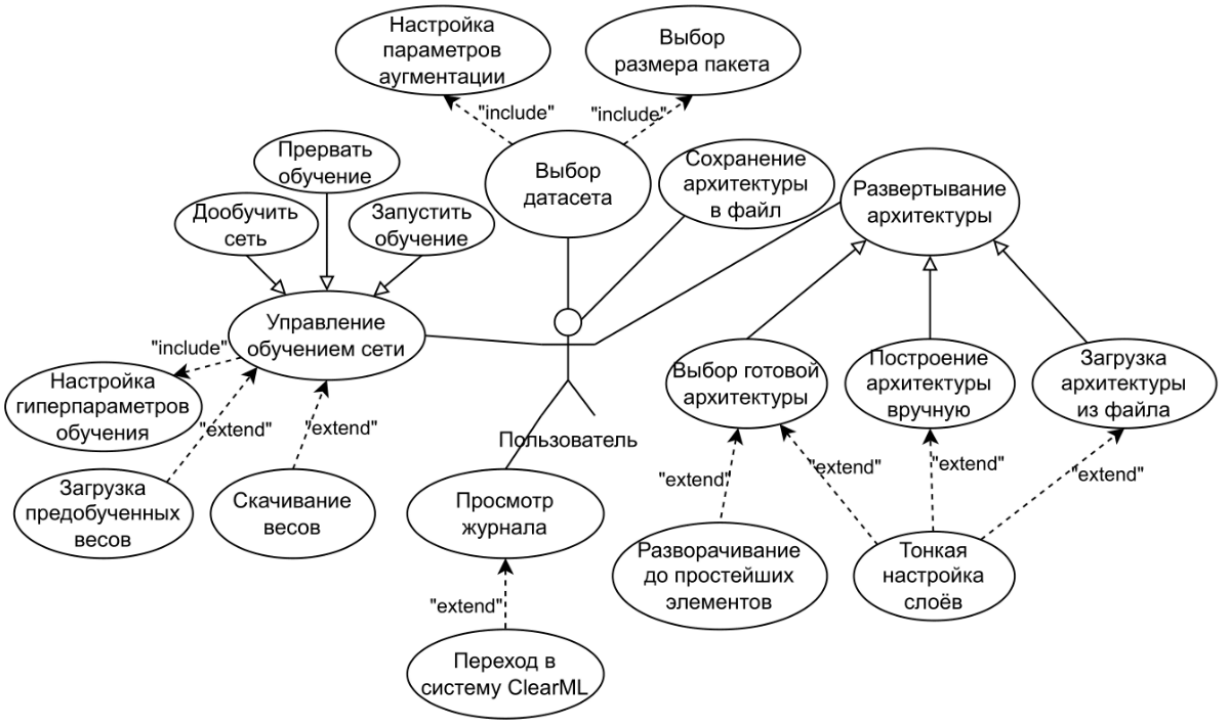

Рисунок А.1 – Диаграмма прецедентов

Приложение Б
(справочное)
Диаграмма последовательности

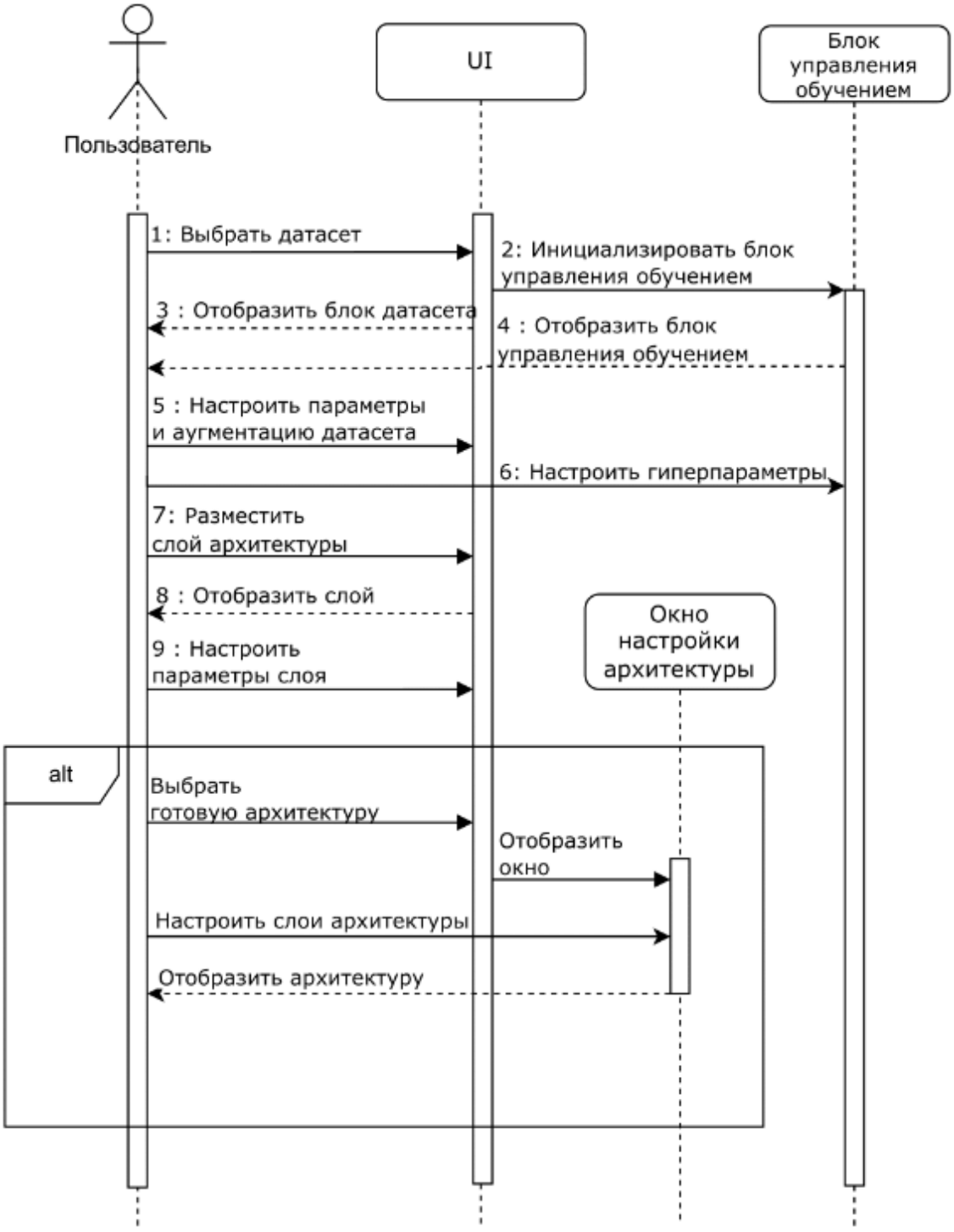

Рисунок Б.1 – Диаграмма последовательности этапа проектирования архитектуры

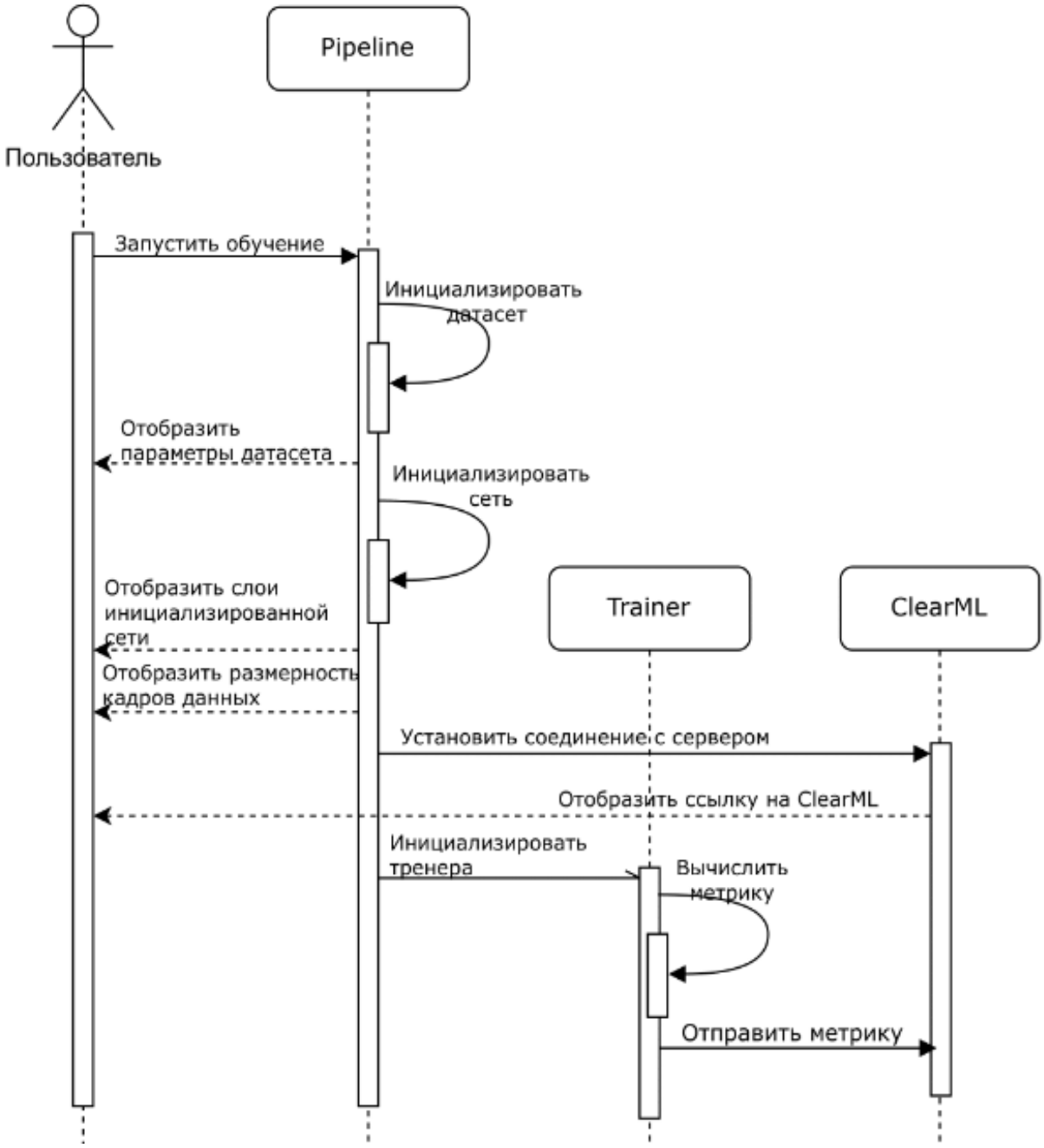

Рисунок Б.2 – Диаграмма последовательности этапа обучения модели

Приложение В
(справочное)
Макеты дизайна

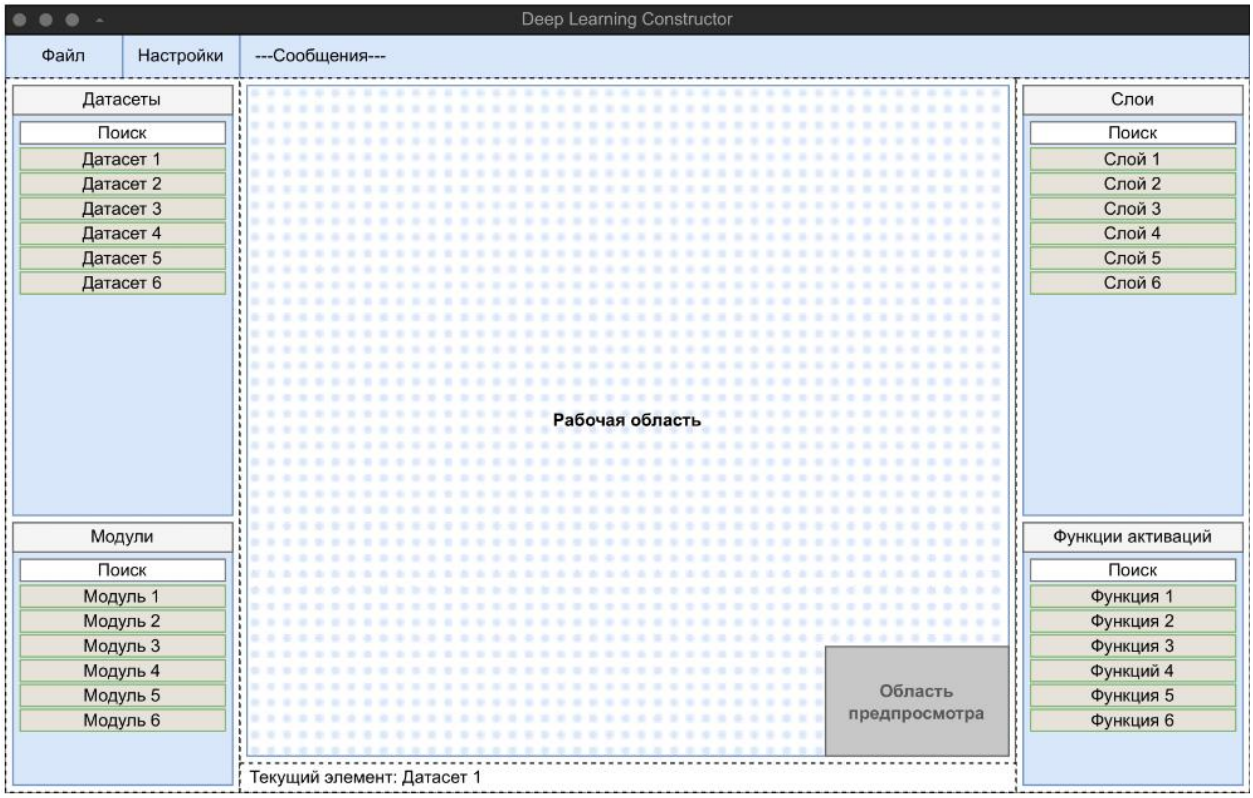

Рисунок В.1 – Макет рабочей области

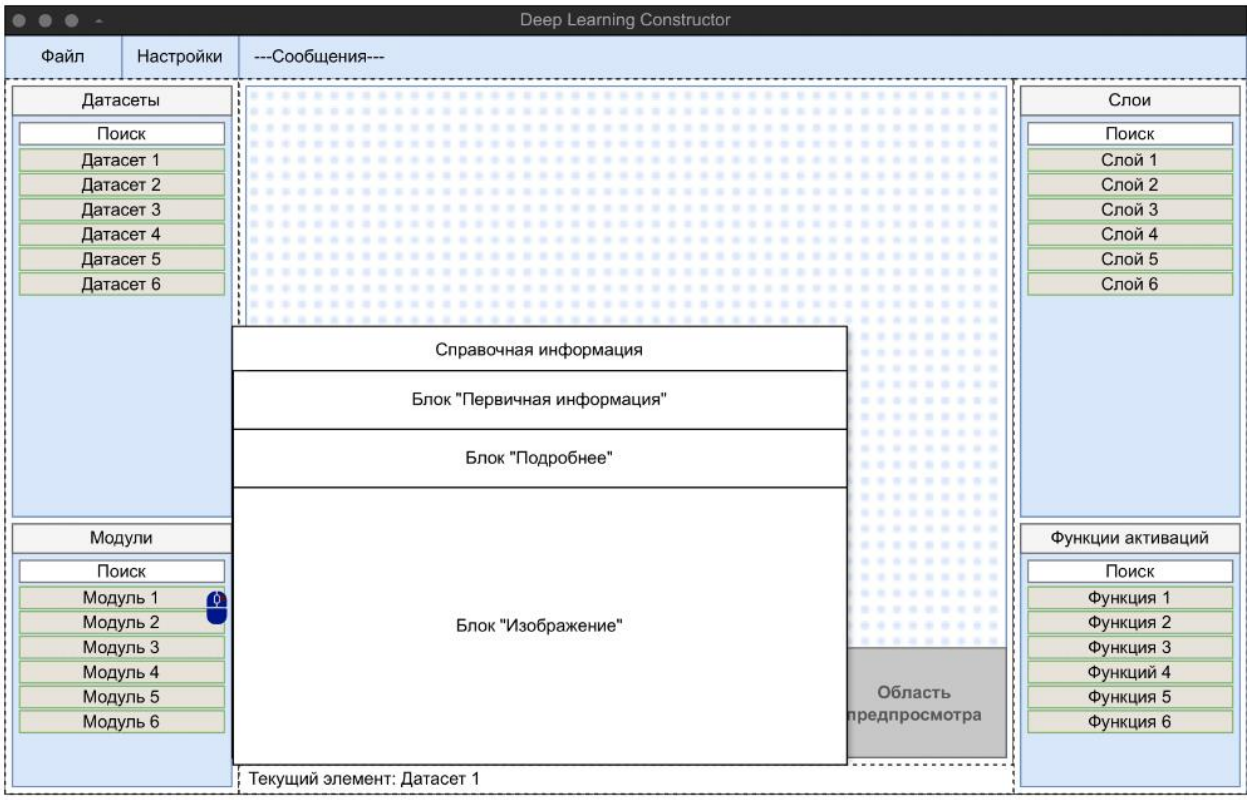

Рисунок В.2 – Макет демонстрации справочной информации

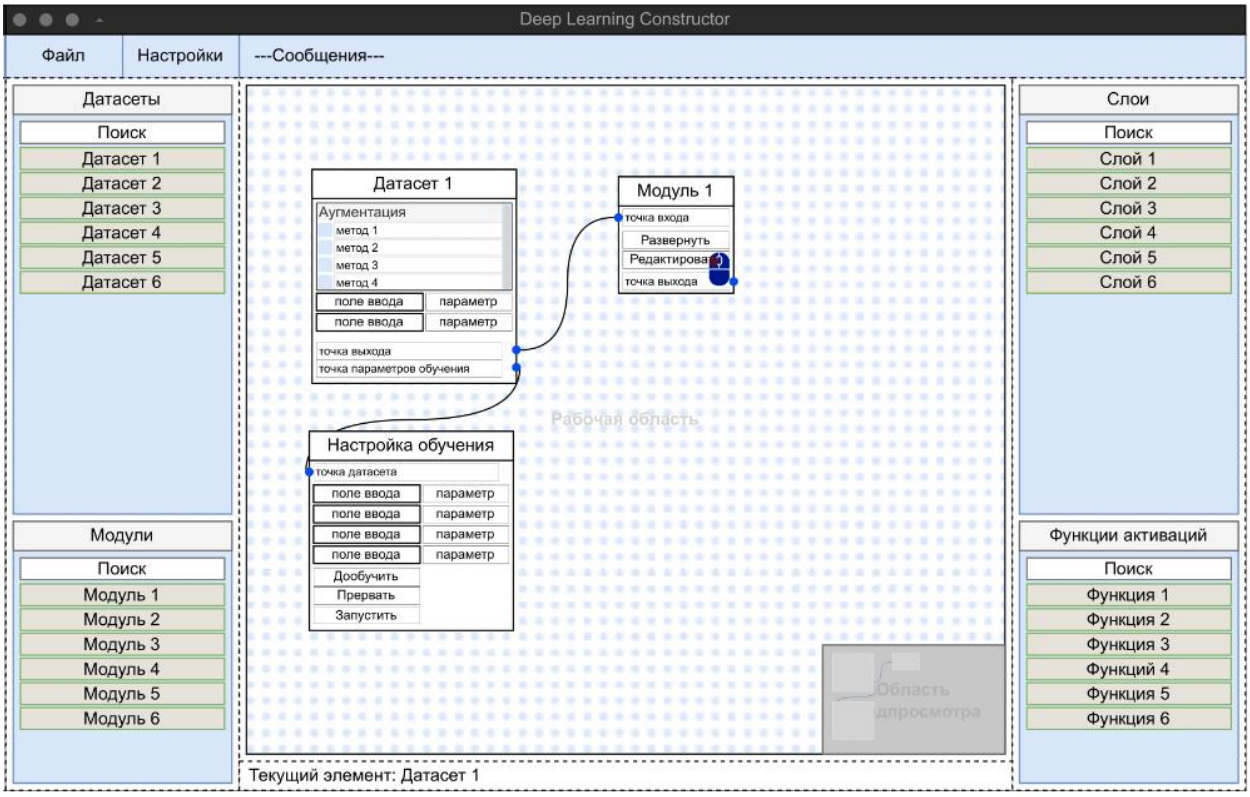

Рисунок В.3 – Макет размещения датасета и предзагруженной архитектуры

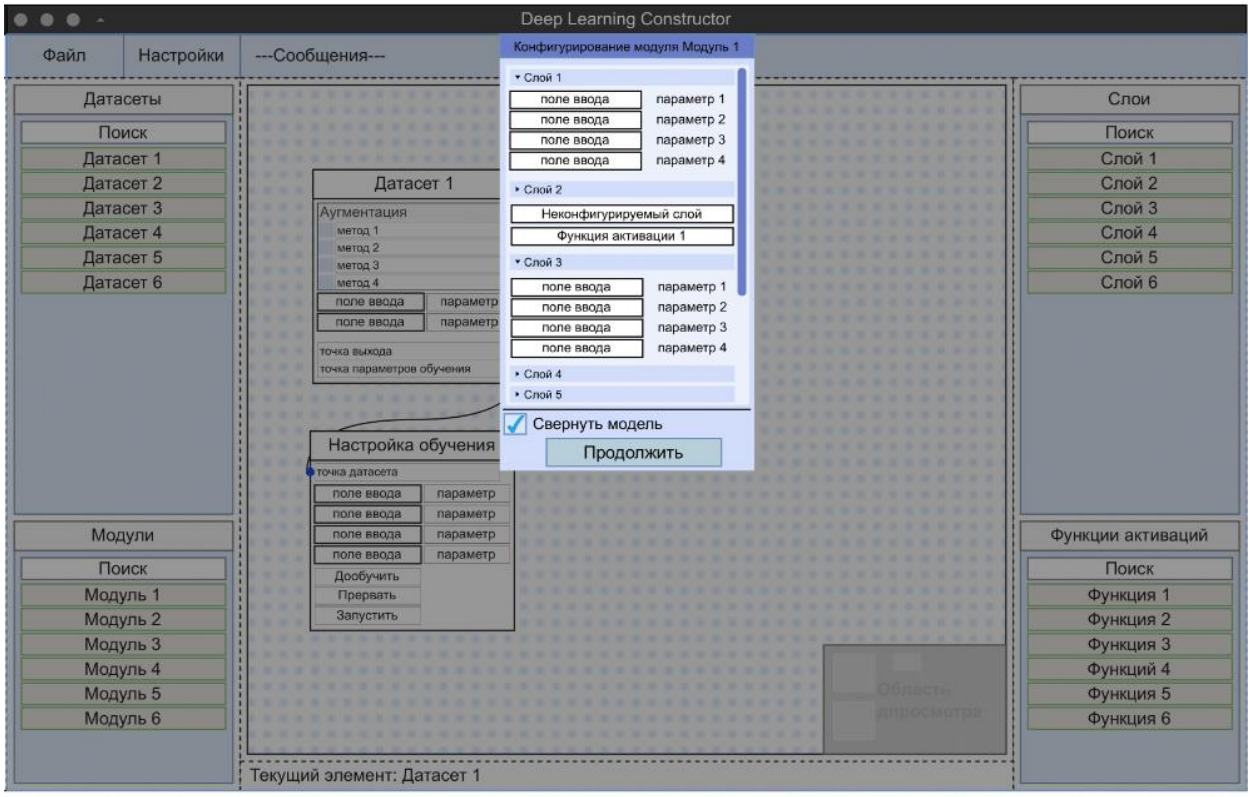

Рисунок В.4 – Макет модального окна для настройки предзагруженной архитектуры

Приложение Г
(справочное)
Диаграммы классов

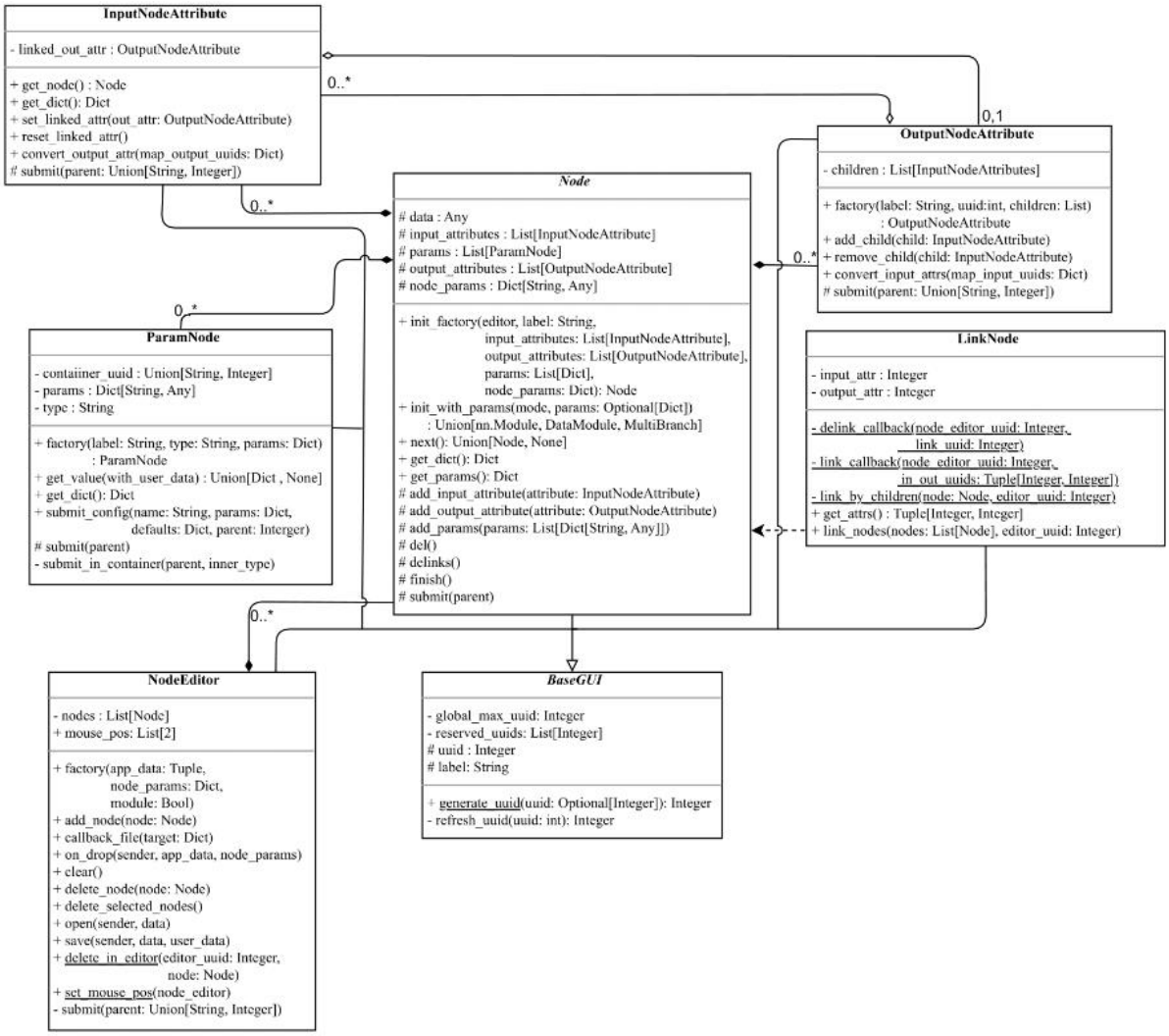

Рисунок Г.1 – Диаграмма интерфейсный классов

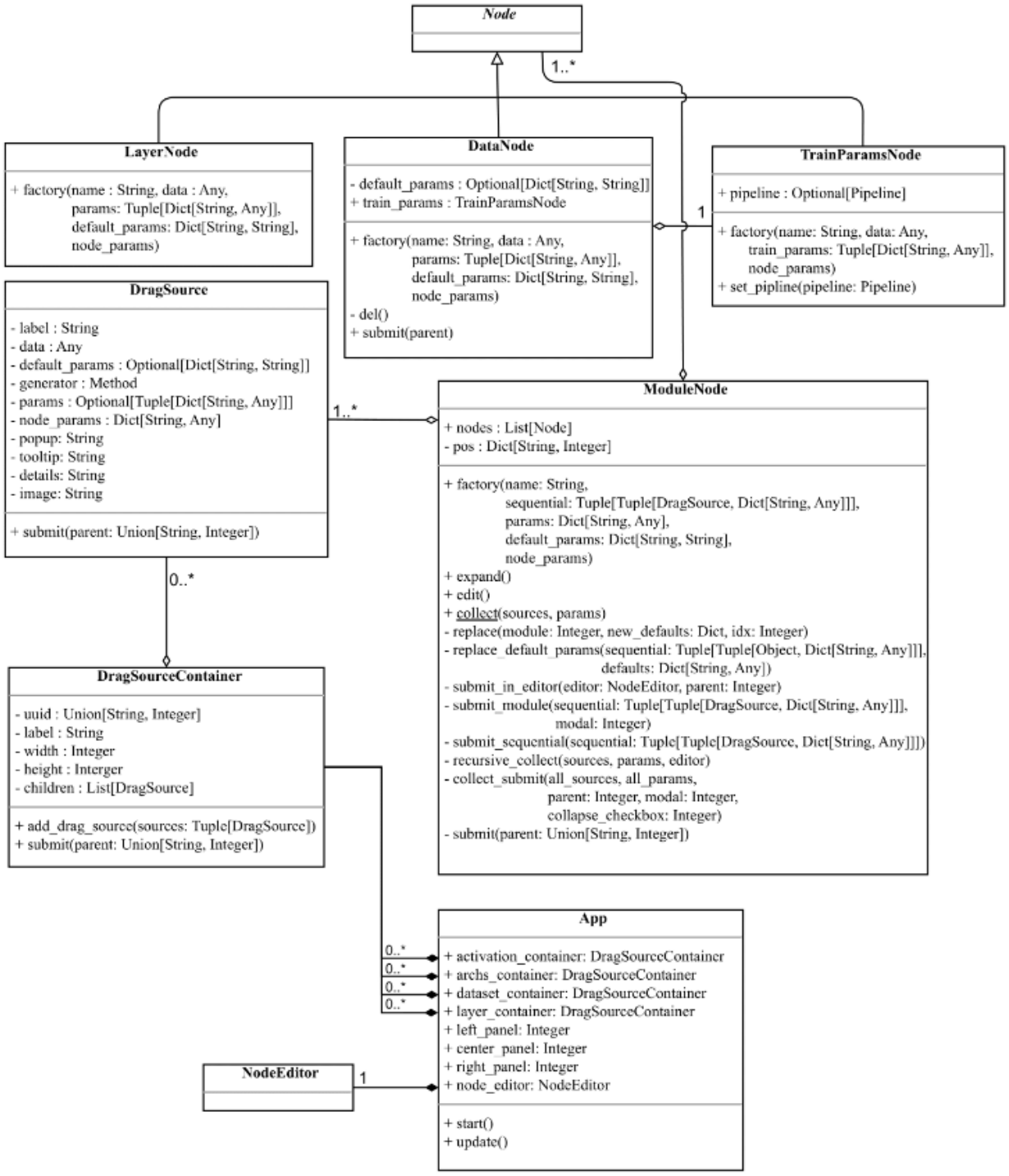

Рисунок Г.2 – Диаграмма ресурсных классов

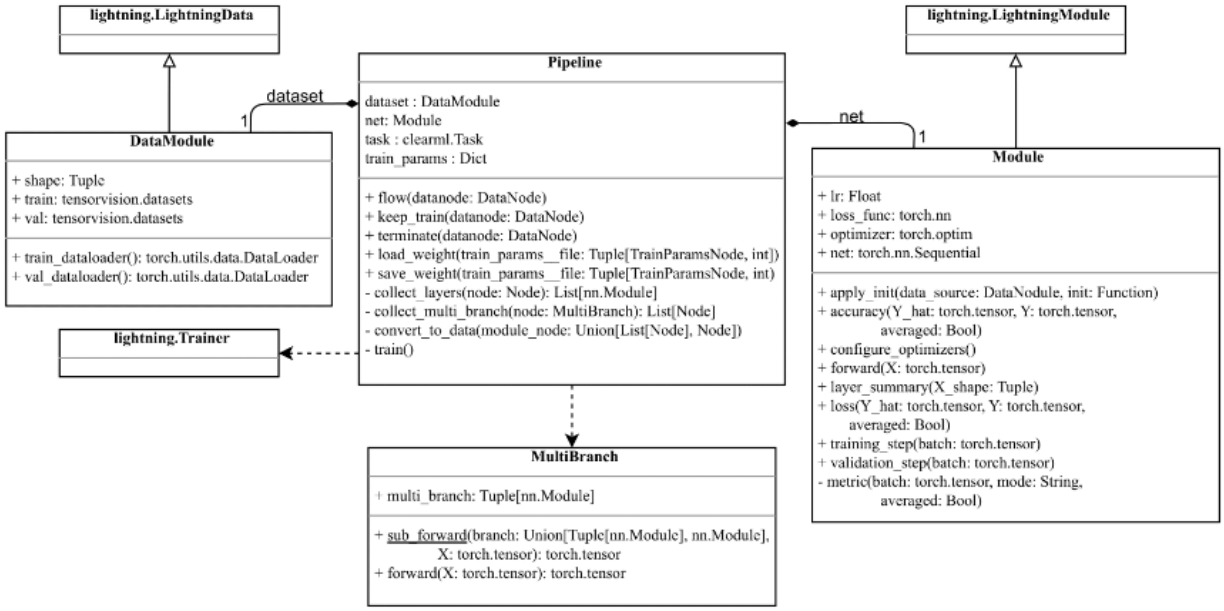

Рисунок Г.3 – Диаграмма классов для процесса обучения и сбора данных

Таблица Г.1 – Описание атрибутов и методов интерфейсных классов

Класс (описание)	Атрибуты	Методы
NodeEditor – класс, представляющий из себя рабочее поле, в котором размещаются все узлы	• nodes – список узлов, инициализированных в рабочем поле; • mouse_pos – координаты в двумерной декартовой системе текущей позиции мышки.	• factory(app_data: Tuple, node_params: Dict, module: Bool) – фабричный метод, инициализирующий и возвращающий экземпляр класса, дочернего к классу Node. • add_node(node: Node) – добавляет узел Node в список узлов nodes; • callback_file(target: Dict) – метод обратного вызова для активации обработчика, указанного в target. На данный момент поддерживается ‘save’ и ‘open’; • on_drop(sender, app_data, node_params) – метод обратного вызова, срабатывающий при сбросе ресурса в рабочее поле. Вызывает генератор, инициализирующий экземпляр дочернего класса к родителю Node; • clear()– удаляет с рабочего поля все узлы; • delete_node(node: Node) – удаляет переданный узел из рабочей области; • delete_selected()– удаляет выделенные в интерфейсе узлы из рабочей области; • open(sender, data) – метод обратного вызова, вызываемый для открытия фала проекта после подтверждения пути в контекстном окне выбора файла; • save(sender, data, user_data) – метод обратного вызова, вызываемый для сохранения проекта после подтверждения пути в контекстном окне выбора файла; • set_mouse_pos(node_editor) – установить значения x и y текущей позиции курсора; • submit(parent) – инициализирует и прикрепляет рабочее поле к главному окну.
BaseGUI – абстрактный класс для всех классов, поддерживающих интерфейс пользователя;инициирует идентификатор и подсчитывает максимальный глобальный идентификатор.	• global_max_uuid – максимальное значение зарезервированного элемента интерфейса; • reserved_uuids – список зарезервированных идентификаторов; • uuid – текущий идентификатор дочернего элемента; • label – название элемента.	• generate_uuid(uuid: Optional[Integer]) – метод, резервирующий идентификатор, который вызывает refresh_uuid. • refresh_uuid(uuid: Integer) – метод, проверяющий совпадение сгенерированного идентификатора со списком зарезервированных uuid.

Продолжение таблицы Г.1

Класс (описание)	Атрибуты	Методы
Node – абстрактный класс, наследуемый от BaseGUI, отвечающий за отображение узла в рабочем поле, а также хранящий информацию, которая будет использована во время обучения сети.	• node_params – настройки узла во время его рендера; • data – информация об узле, используемая во время обучения; • input_attributes – входные точки данных узла типа InputNodeAttribute; • output_attributes – выходные точки данных узла типа OutputNodeAttribute; • params – статические атрибуты узла типа ParamNode, выступающие в качестве параметров узла.	• init_factory(editor, input_attributes, output_attributes, params, node_params) – метод, загружающий список узлов из проекта; отвечает за функцию “Открыть проект”; • init_with_params(mode, params) – инициализирует класс данными для загрузки в сеть; • next() - просматривает выходной узел текущего блока и возвращает связанные блоки; • get_dict() – возвращает информацию о блоке: идентификатор, название, выходные, входные узлы и параметры; • get_params() – возвращает параметры и их текущие значения в виде словаря; • add_input_attribute(attribute) – добавляет экземпляр класса InputNodeAttribute в список входных точек данных; • add_output_attribute(attribute) – добавляет экземпляр класса OutputNodeAttribute в список выходных точек данных; • add_params(params) – добавляет в список статических атрибутов узла экземпляр класса ParamNode; • del() - удаляет текущий узел и отвязывает все привязанные к нему связи; • delinks() - отвязывает все связанные узлы; • finish() – привязывает цветовую схему к узлу и его дочерним элементам; • submit(parent) – инициализирует и прикрепляет к рабочему полю узел и все его дочерние элементы
ParamNode – класс, генерирующий параметры узла.	• container_uuid – идентификатор, непосредственно привязываемый к элементу статического атрибута; • params – параметры генерируемого элемента атрибута • type – тип атрибута, сопоставимый с одним из поддерживаемых в интерфейсе типов	• factory(label, type, params) – фабричный метод, инициализирующий и возвращающий экземпляр класса ParamNode. • submit(parent) – инициализирует родительский атрибут и вызывает внутренний метод submit_in_container; • submit_in_container(parent, inner_type) – инициализирует статический атрибут указанного типа. Поддерживаемые типы: int, float, text, combo, collaps, blank, bool, button, file, path. Тип collaps поддерживает вложенные атрибуты; • get_value(with_user_data) – возвращает текущее значение атрибута, обрабатываемое в зависимости от типа. Параметр with_user_data является флагом для инициализации параметра там, где это требуется; • get_dict() – возвращает параметр и текущее значение в виде словаря.

Окончание таблицы Г.1

Класс (описание)	Атрибуты	Методы
InputNodeAttribute – класс, генерирующий входную точку данных узла.	• linked_out_attr – ссылка на экземпляр выходного атрибута типа OutputNodeAttribute.	• get_node() – возвращает экземпляр класса Node, к которому прикреплен данный входной атрибут; • get_dict() - возвращает текущий входной атрибут, включающий информацию об идентификаторе связанного выходного атрибута в виде словаря; • set_liked_attr(out_attr) – связывает текущий входной атрибут с указанным выходным атрибутом типа OutputNodeAttribute; • reset_linked_attr() – очищает ссылку на выходной атрибут; • convert_output_attr(map_output_uuids) – метод, актуализирующий идентификаторы в случае, если они конфликтуют с зарезервированными uuid; • submit() – инициализирует входной атрибут и прикрепляет к узлу.
OutputNodeAttribute – класс, генерирующий выходную точку данных узла.	• children – список входных атрибутов, которые были соединены с текущим выходным атрибутом.	• factory(label, uuid, children) – фабричный метод, инициализирующий и возвращающий экземпляр класса OutputNodeAttribute. • add_child(child) – добавляет входной атрибут типа InputNodeAttribute в список входных точек данных; • remove_child(child) – удаляет указанный входной атрибут из списка входных точек данных; • convert_input_attrs(map_input_uuid) – метод, актуализирующий идентификаторы в случае, если они конфликтуют с зарезервированными uuid; • submit(parent) – инициализирует выходной атрибут и прикрепляет к узлу.
LinkNode – класс, связывающий выходную и входную точки данных узла. Необходим при инициализации сети перед обучением.	• input_attr – входной атрибут, прикрепленный к данной связи; • output_attr – выходной атрибут, прикрепленный к данной связи.	• link_callback(node_editor_uuid, in_out_attrs) – инициализирует связь и соединяет выходной и входной атрибуты двух узлов; • delink_callback(node_editor_uuid, link_uuid) – удаляет связь между двумя атрибутами узлов; • get_attrs() – возвращает пару входной/выходной идентификаторы атрибутов; • link_nodes(nodes, editor_uuid) – связывает указанные узлы последовательно.

Таблица Г.2 – Описание атрибутов и методов ресурсных классов

Класс (описание)	Атрибуты	Методы
LayerNode – наследник класса Node, отвечающий за функционирование обучаемых компонент модели.	–	• factory(name, data, params, default_params) – фабричный метод, инициализирующий и возвращающий экземпляр класса LayerNode.
DataNode – наследник класса Node, выступающий в качестве датасета, на котором будет происходить обучение.	• default_params – словарь параметров обучения, устанавливаемых по-умолчанию для конкретного датасета; • train_params – экземпляр класса TrainParamsNode, который закреплен для текущего датасета.	• factory(name, data, params, default_params) – фабричный метод, инициализирующий и возвращающий экземпляр класса DataNode. • submit(parent) – инициализирует узел с данными и узел TrainParamsNode, соединяя при этом связью.
TrainParamsNode – автоматически генерируется при появлении узла DataNode. Предназначен для регулирования параметров обучения и дополнительного функционала.	• pipeline – экземпляр класса Pipeline, за которым закрепляются параметры обучения.	• factory(name, data, params, default_params) – фабричный метод, инициализирующий и возвращающий экземпляр класса TrainParamsNode. • set_pipeline(pipeline) – устанавливает экземпляр класса Pipeline.
DragSourceContainer – класс-контейнер, содержащий экземпляры DragSource, отображаемый в области ресурсов в интерфейсе.	• label – заголовок контейнера с ресурсами; • width – ширина контейнера; • height – высота контейнера; • children – ресурсы типа DragSource контейнера.	• submit(parent) – инициализирует и закрепляет контейнер с ресурсами DragSource в интерфейсе пользователя; • add_drag_source(sources) – добавляет новый ресурс в контейнер.

Продолжение таблицы Г.2

Класс (описание)	Атрибуты	Методы
ModuleNode – класс, аггрегирующий узлы Node, что позволяет проектировать глубокие модульные структуры нейронных сетей.	• pos – относительная позиция узла, инкрементируемая по мере размещения модулей; • nodes – обработанная последовательность узлов сети;	• factory(name, data, params, default_params) – фабричный метод, инициализирующий и возвращающий экземпляр класса ModuleNode. • expand() – метод, располагающий модуль в развернутом до простейших элементов виде; • edit() – удаляет выбранный свернутый модуль и открывает окно конфигурирования модуля; • collect() – метод сбора слоев • replace() – метод применения конфигурации из окна в инициализируемую последовательность; • replace_default_params() – метод замены параметров по умолчанию на параметры, указываемые при инициализации архитектуры; • submit_in_editor() – метод размещения архитектуры в рабочей области; • recursive_collect() – метод рекурсивного сбора многомодульной архитектуры, в том числе многофилиальной; • submit(parent) – вызывает метод submit_module() и связывает все узлы последовательно; • submit_module() – обрабатывает последовательность ресурсов sequential: вызывает replace_default_params(), инициализирует узлы Node и закрепляет в рабочую область; • replace_default_params(source, defaults) – заменяет параметры элементов последовательности на параметры по-умолчанию, указанные при передаче ресурса.

Окончание таблицы Г.2

Класс (описание)	Атрибуты	Методы
DragSource – наследник класса BaseGUI; класс-генератор, инициализирующий узел или сеть после перетаскивания в рабочее поле.	• data – ссылка на класс, выступающая в качестве генератора датасета или слоя; • default_params – значения атрибутов по-умолчанию; • generator – ссылка на фабричный метод; • params – атрибуты узла; • node_params – параметры узла в целом$ • popup – справка, оформленная в markdown; • tooltip – краткая информация, отображаемая при наведении на элемент; • details – информация, отображаемая в разделе “Подробнее”; • image – путь до изображения, отображаемого в справке.	• submit(parent) – инициализирует кнопку и загрузочный ресурс, используемый в дальнейшем для вызова фабричного метода.
App - основной класс, инициализирующий пользовательский интерфейс.	• activation_container – экземпляр класса DragSourceContainer, который хранит экземпляры DragSource с функциями активаций; • archs_container – хранит шаблоны архитектур; • dataset_container – хранит информацию для инициализации датасетов; • layer_container – хранит информацию для инициализации слоев; • left_panel – идентификатор левой части интерфейса; • center_panel – идентификатор центральной части интерфейса; • right_panel – идентификатор правой части интерфейса; • node_editor – экземпляр класса рабочей области.	• start() – метод, инициализирующий весь интерфейс; • update() – метод, обновляющий интерфейс.

Таблица Г.3 – Описание атрибутов и методов классов, отвечающих за процесс сбора данных с рабочей области и последующей инициализации и обучения модели

Класс (описание)	Атрибуты	Методы
MultiBranch - класс для работы с многофилиальными сетями.	• multi_branch – кортеж из модулей	• sub_forward(branch, X) - метод рекурсивного сбора ветвей и их инициализации; • forward() - метод прямого прохода по элементам из multi_branch.

Окончание таблицы Г.3

Класс (описание)	Атрибуты	Методы
DataModule – класс инициализации датасета	• shape – итоговая размерность датасета; • train – экземпляр класса с примененной аугментацией и выделенной тренировочной частью; • val – экземпляр класса с примененной аугментацией и выделенной валидационной частью.	• train_dataloader() – возвращает экземпляр класса torch.DataLoader для тренировочной части; • val_dataloader() – возвращает экземпляр класса torch.DataLoader для валидационной части.
Module - класс, представляющий собой сеть нейронной сети	• lr – скорость обучения; • loss_func – функция потерь; • optimizer – оптимизатор обучения; • net – список слоев сети.	• apply_init(data_source, init) – предварительно инициализирует веса сети; • accuracy(Y_hat, Y) – вычисляет точность между предсказанным и фактическим тензором; • configure_optimizers() – возвращает метод оптимизации обучения; • forward(X) – метод прямого прохождения по сети; • layer_summary() – метод отображения размерности кадра, проходящего через каждый слой сети; • loss(Y_hat, Y) – метод вычисления значения функции потерь; • training_step(batch) – тренировочный шаг для пакета данных; • validation_step(batch) – валидационный шаг для пакета данных. • metric(batch, mode, average) – вычисление указанной в параметре ‘mode’ метрики на пакете данных.
Pipeline – основной класс, инициализирующий сеть	• dataset – экземпляр класса DataModule; • net – список инициализированных слоев в экземплярах класса Module; • task – экземпляр класса ClearML для мониторинга обучения; • train_params – экземпляр класса clearml.Task	• flow(datanode) – метод инициализации экземпляра класса Pipeline для старта обучения; • keep_train(datanode) – метод, вызываемый для продолжения обучения; • terminate() – метод прерывания обучения; • load_weight(train_params__file) – метод загрузки весов из файла в сеть; • save_weight(train_params__file) – метод сохранения весов; • collect_layers(node) – метод сбора слоев с рабочей области; • collect_multi_branch() – метод сбора многофилиальных сетей; • convert_to_data(module_node) – метод конвертации в подходящий формат для поддержки многофилиальных сетей.

Приложение Д
(справочное)
Конфигурация проекта

Листинг Д.1 – Конфигурация проекта dlc в файле pyproject.toml

```
[tool.poetry]
name = "dlc"
version = "0.1.0"
description = ""
authors = ["luzinsan  <luzinsan.a.430-2@e.tusur.ru>"]
readme = "README.md"
package-mode = false

[tool.poetry.dependencies]
python = "^3.11"
dearpygui = "^1.11.1"
torch = "^2.3.0"
torchvision = "^0.18.0"
lightning = "^2.2.4"
clearml = "^1.16.0"
dearpygui-ext = "^0.9.5"
environs = "^11.0.0"
matplotlib-inline = "^0.1.7"
matplotlib = "^3.9.0"
ipython = "^8.24.0"
tensorboardx = "^2.6.2.2"
mistletoe = "^1.3.0"
gdown = "^5.2.0"
scipy = "^1.13.1"
graphviz = "^0.20.3"

[tool.poetry.group.dev.dependencies]
pytest-pdb = "^0.3.1"

[[tool.mypy.overrides]]
module = "dearpygui.*"
ignore_missing_imports = true

[build-system]
requires = ["poetry-core"]
build-backend = "poetry.core.masonry.api"
```

Листинг Д.2 – Конфигурация SDK ClearML

```
api {
  api_server: https://api.clear.ml
  web_server: https://app.clear.ml
  files_server: https://files.clear.ml
  credentials {"access_key": "КЛЮЧ_ДОСТУПА",
               "secret_key": "ЗАКРЫТЫЙ_КЛЮЧ"}
}
sdk {
  # ClearML - default SDK configuration
  storage {
    cache {
      default_base_dir: "~/.clearml/cache"
    }
    direct_access: [{ url: "file://*" }]
  }
  metrics {
    file_history_size: 100
    matplotlib_untitled_history_size: 100
    images {
      format: JPEG
      quality: 87
      subsampling: 0
    }
    tensorboard_single_series_per_graph: false
  }
  network {
    file_upload_retries: 3
    metrics {
      file_upload_threads: 4
      file_upload_starvation_warning_sec: 120
    }
    iteration {
      max_retries_on_server_error: 5
      retry_backoff_factor_sec: 10
    }
  }
  }
  apply_environment: false
  apply_files: false
}
```

Приложение Е
(справочное)
Реализация проекта

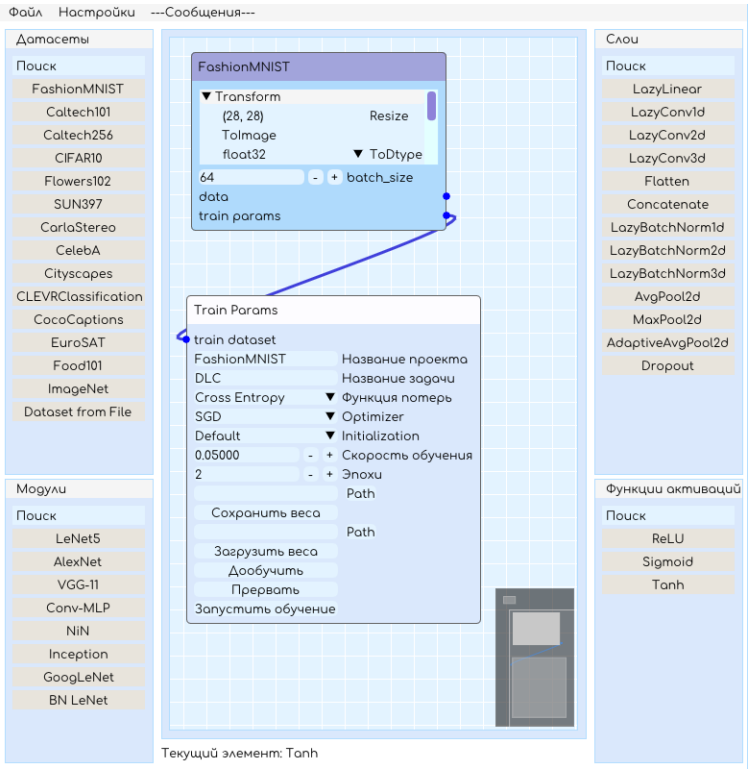

Рисунок Е.1 – Размещение датасета и инициализация тренировочного блока в рабочей области

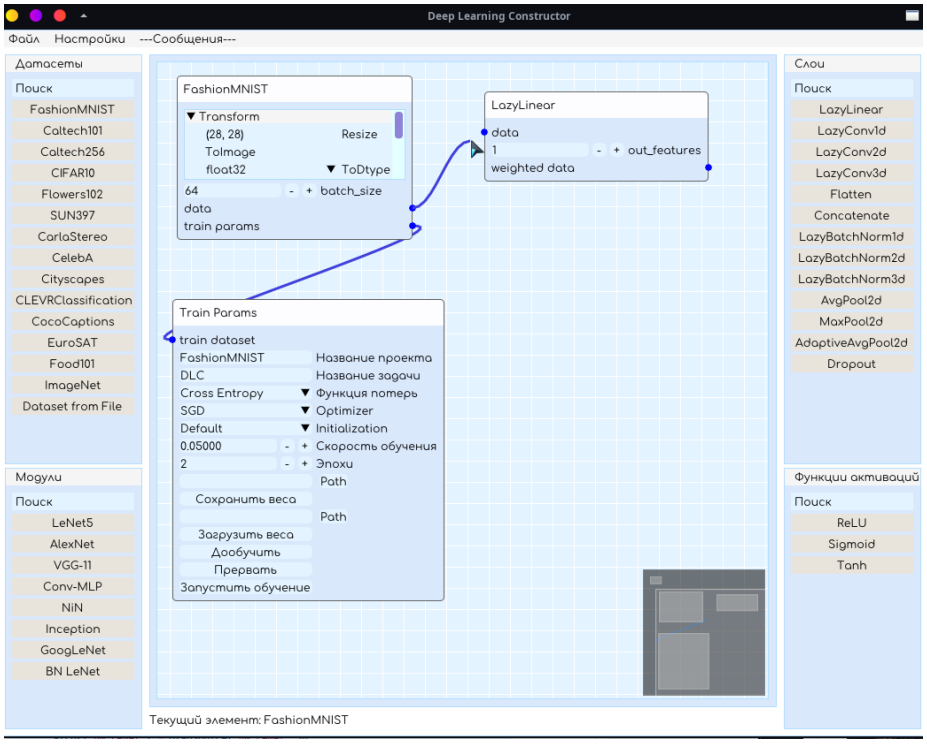

Рисунок Е.2 – Связывание двух узлов

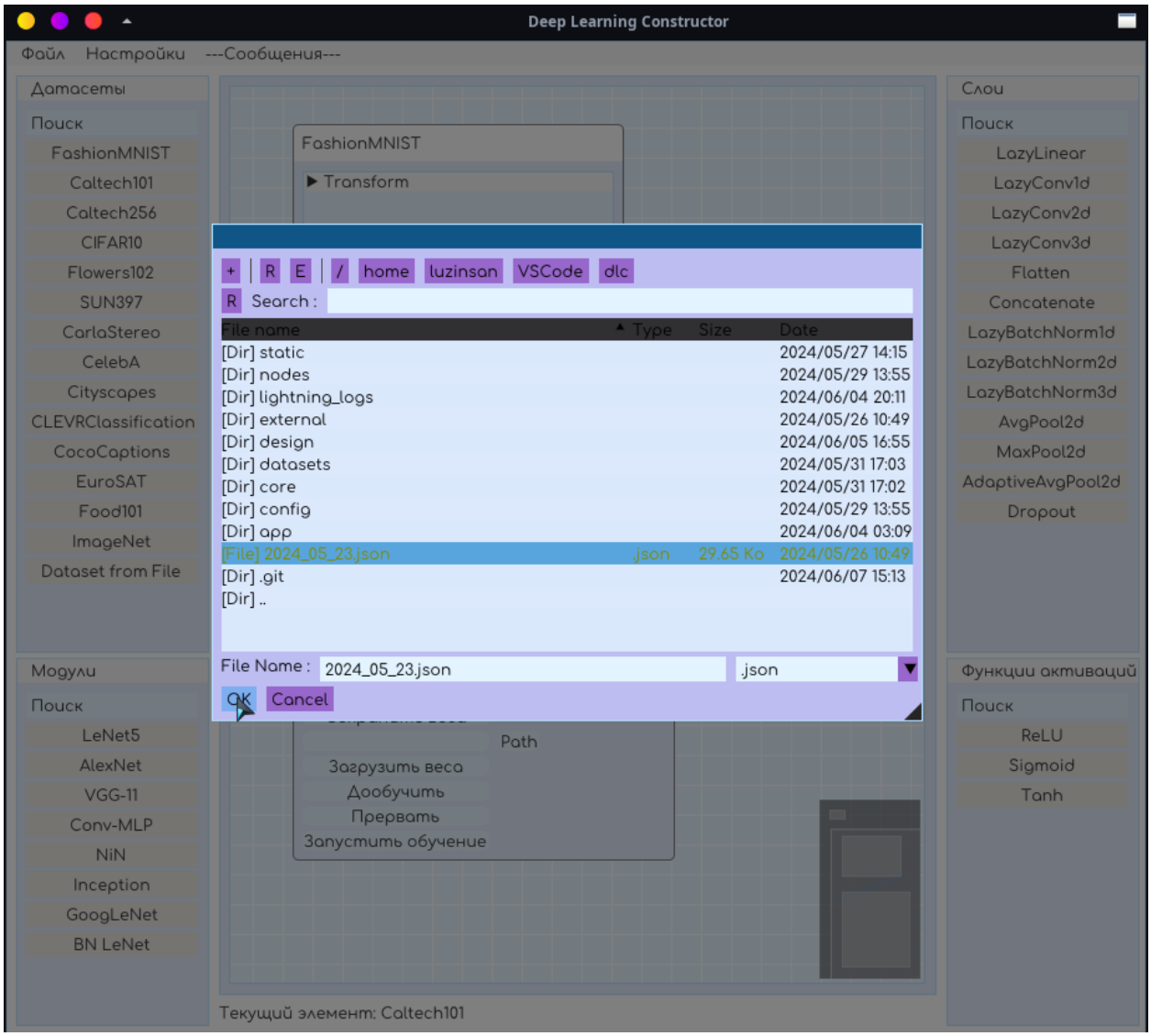

Рисунок Е.3 – Контекстное окно для открытия проекта

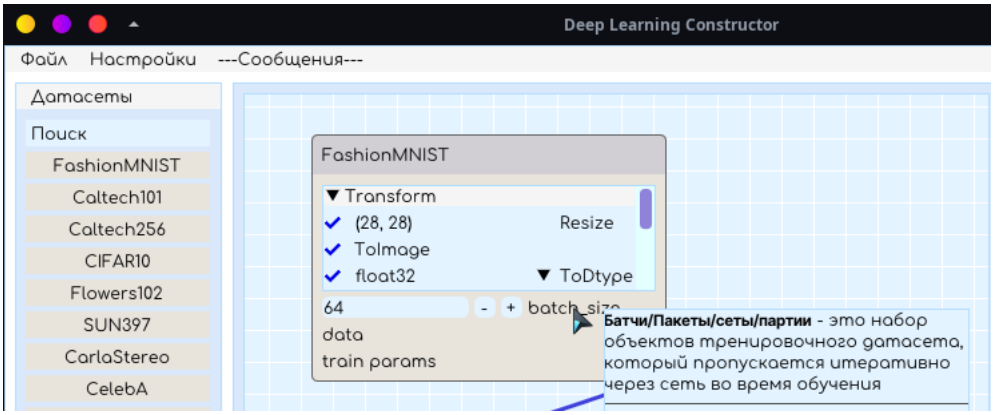

Рисунок Е.4 – Всплывающая подсказка при наведении на параметр

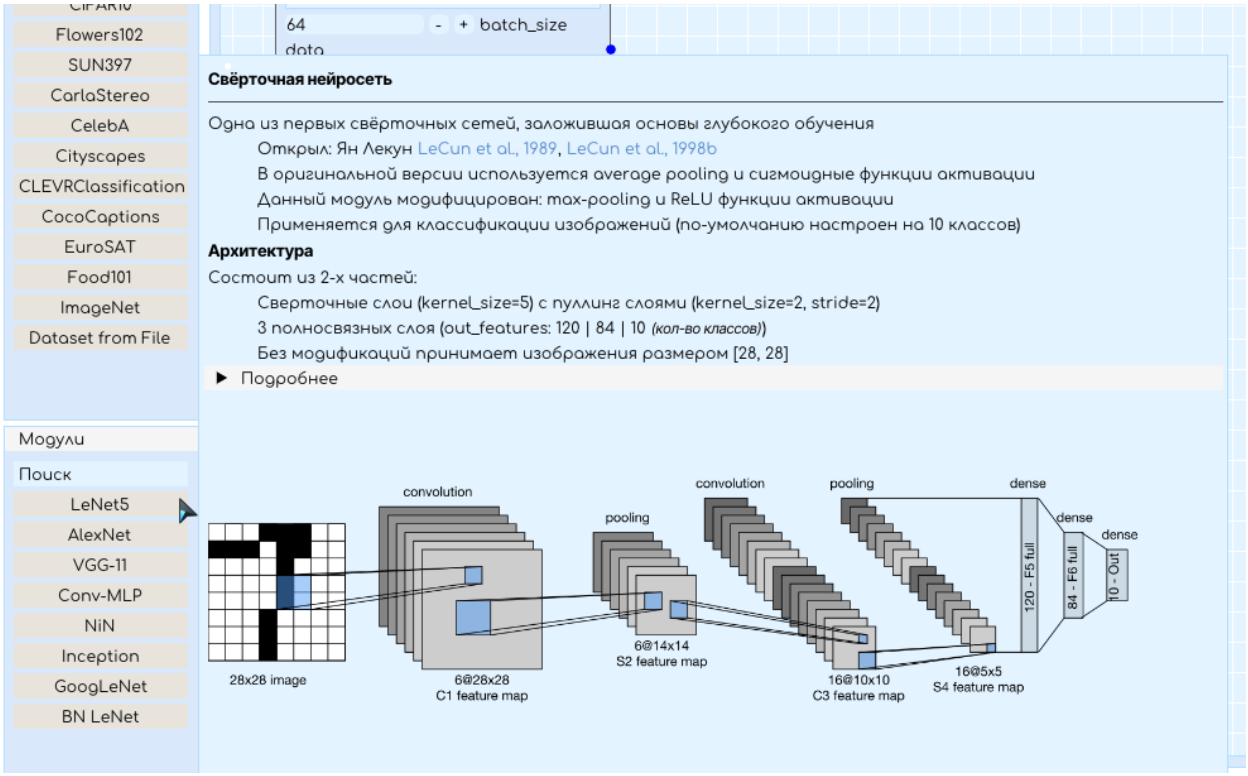

Рисунок Е.5 – Справочная информация по архитектура LeNet5

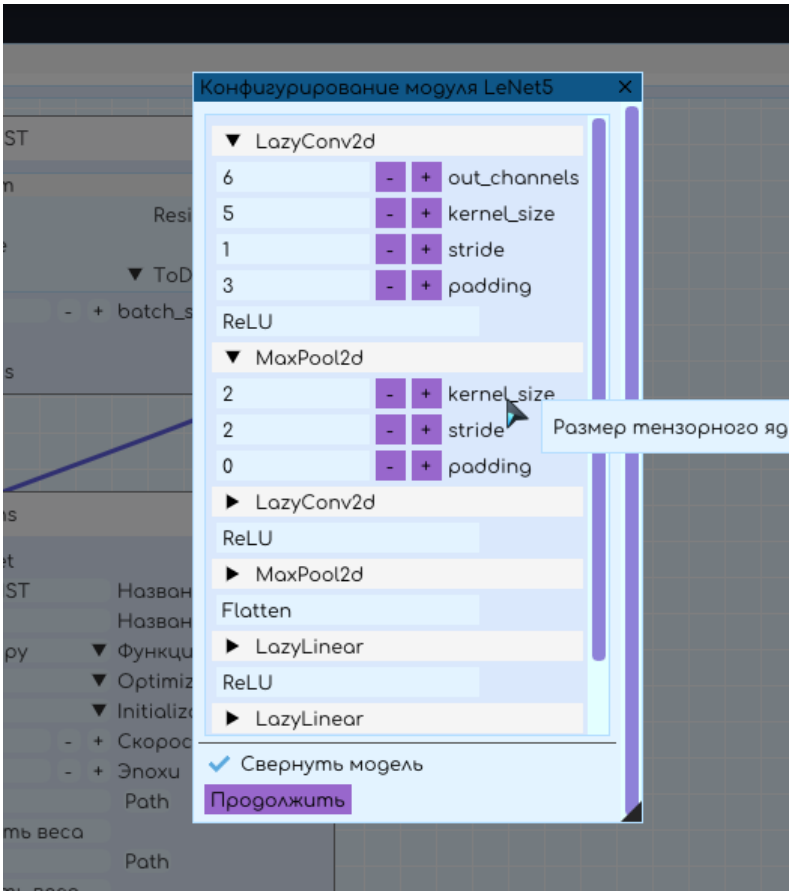

Рисунок Е.6 – Окно конфигурации архитектуры LeNet5

Приложение Ж
(справочное)
Тестирование архитектур

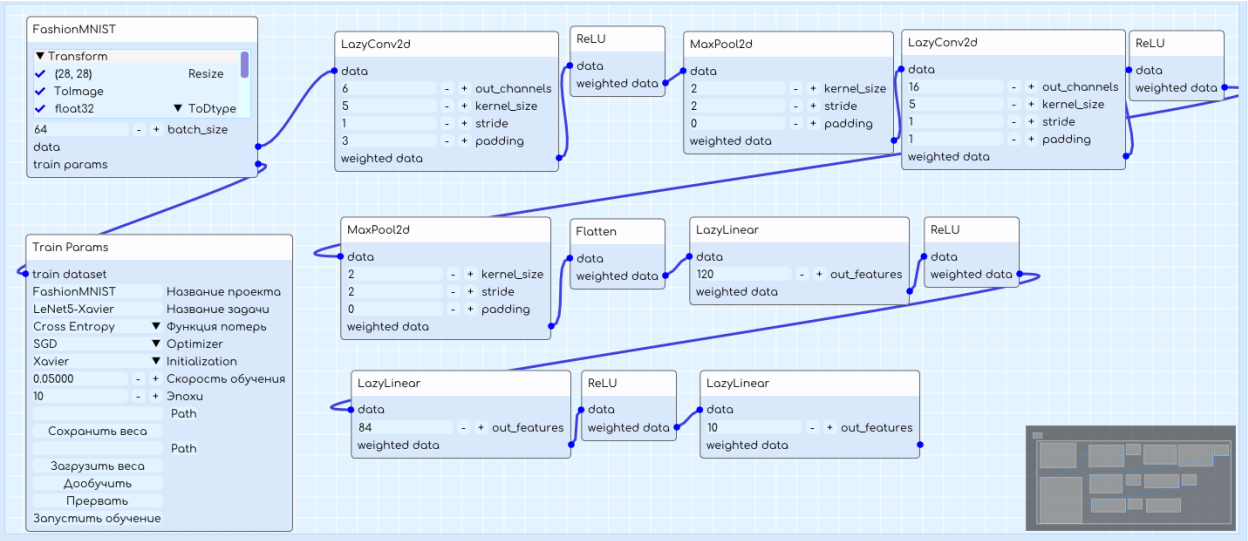

Рисунок Ж.1 – Обучение архитектуры LeNet на датасете FashionMNIST

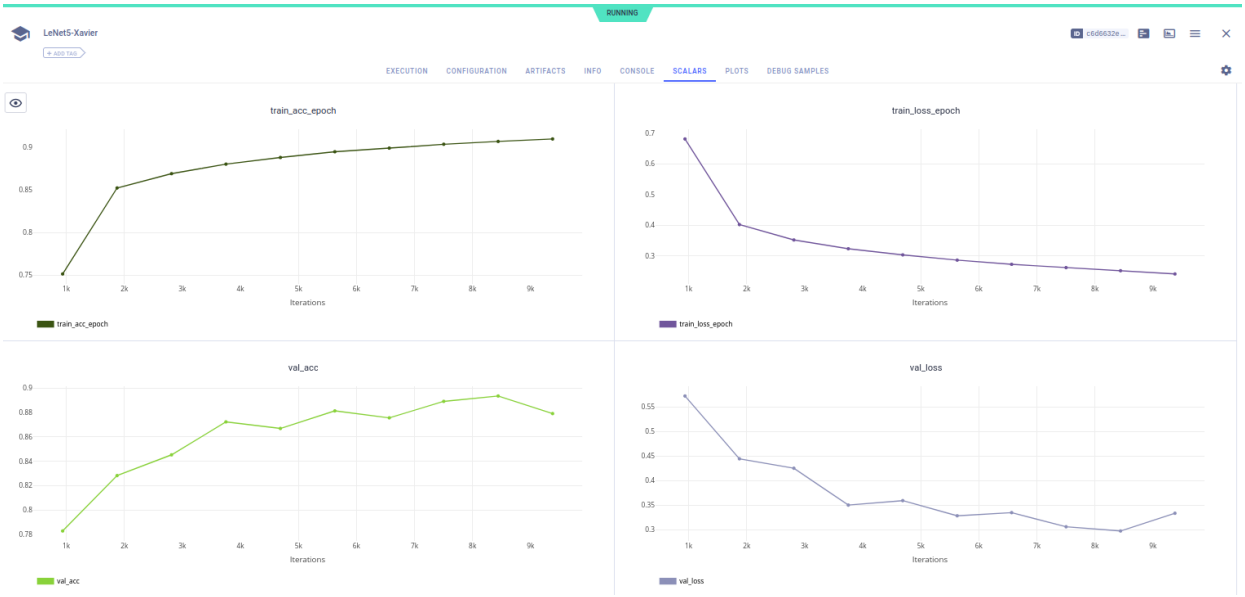

Рисунок Ж.2 – Мониторинг метрик по время обучения сети LeNet

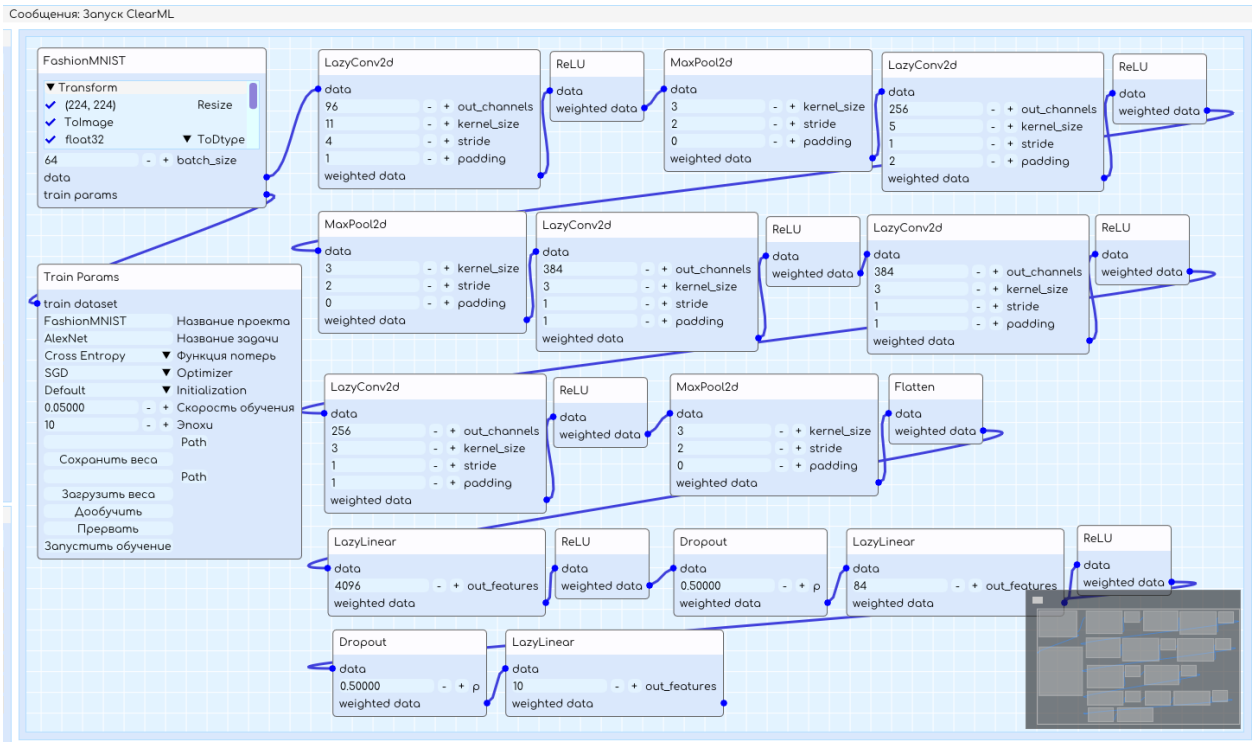

Рисунок Ж.3 – Обучение архитектуры AlexNet на датасете FashionMNIST

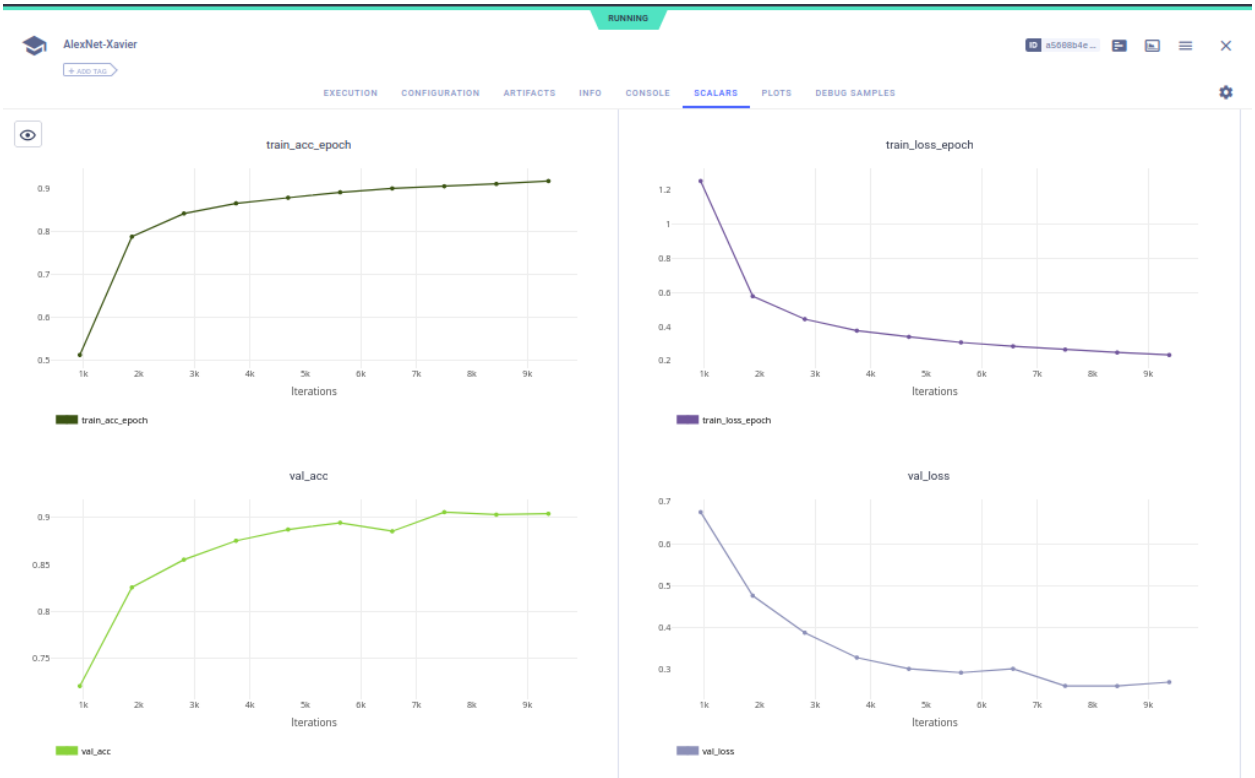

Рисунок Ж.4 – Мониторинг метрик по время обучения сети AlexNet

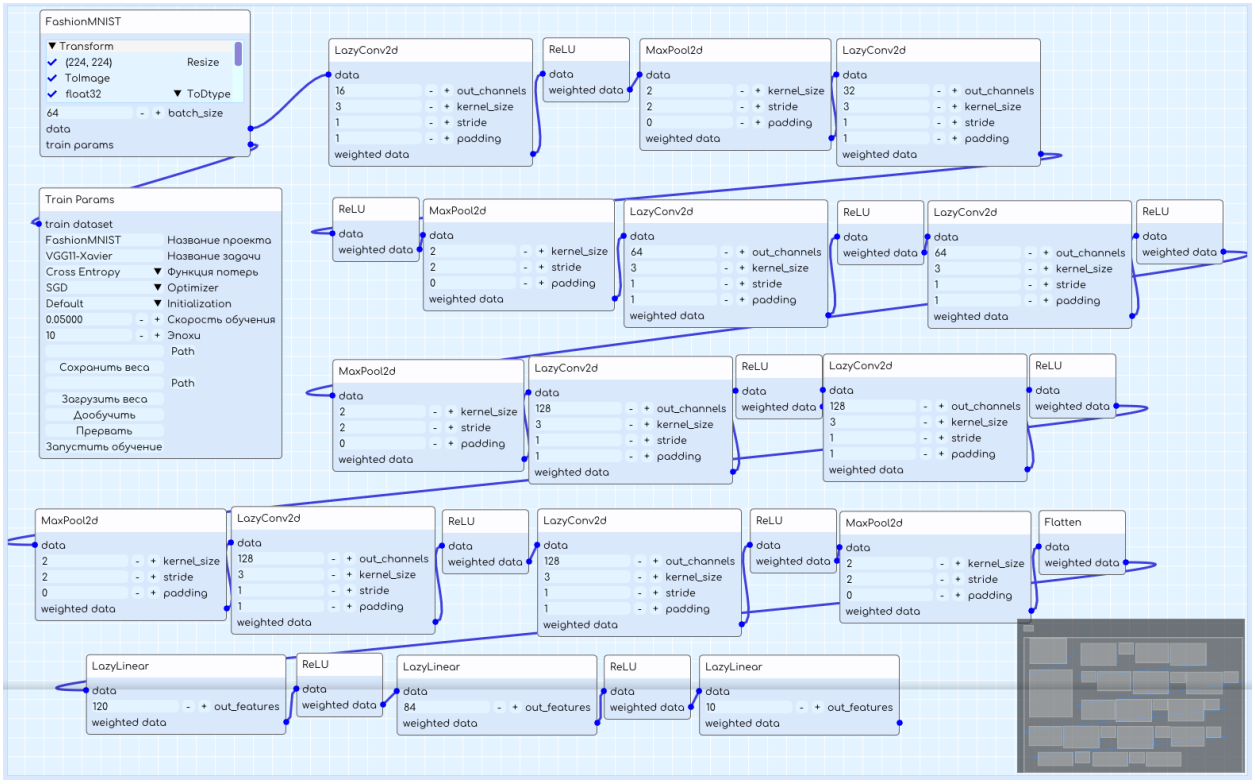

Рисунок Ж.5 – Обучение архитектуры VGG-11 на датасете FashionMNIST

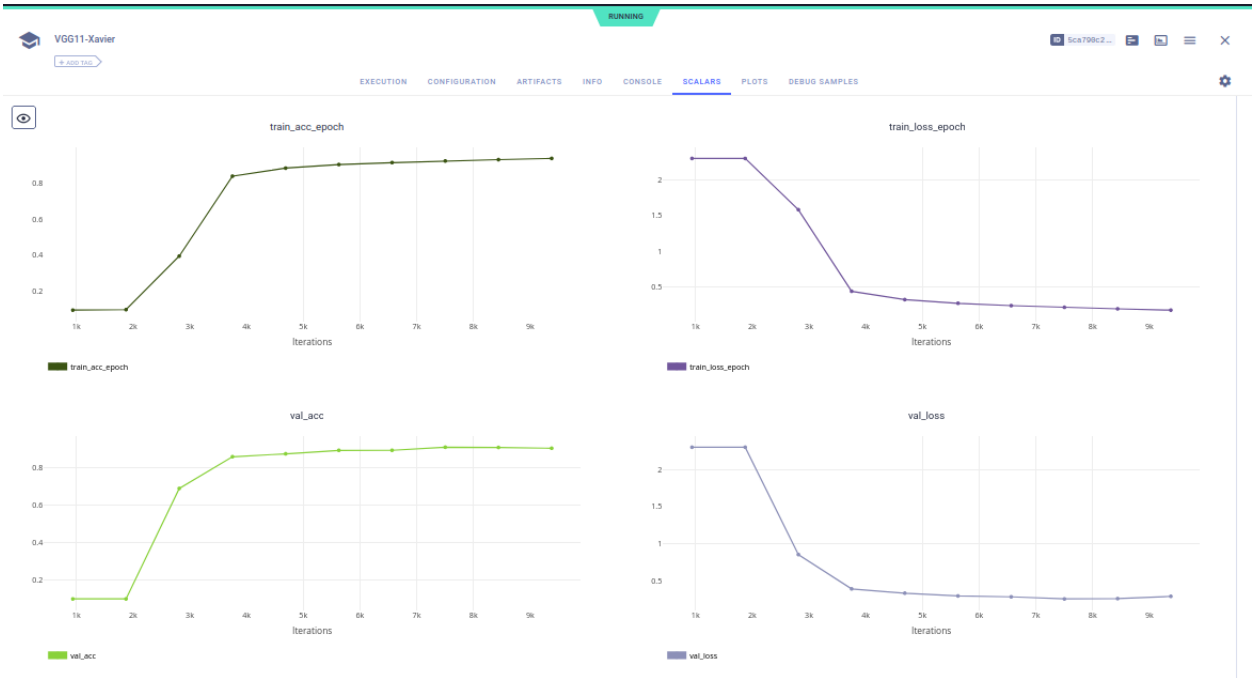

Рисунок Ж.6 – Мониторинг метрик по время обучения сети VGG-11

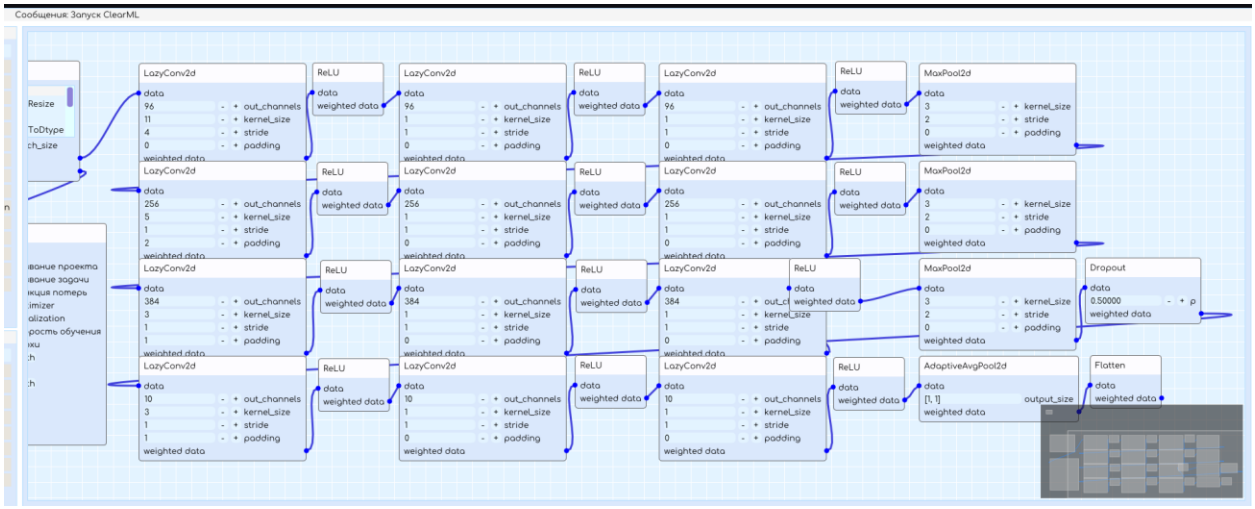

Рисунок Ж.7 – Обучение архитектуры NiN на датасете FashionMNIST

Рисунок Ж.8 – Мониторинг метрик по время обучения сети NiN

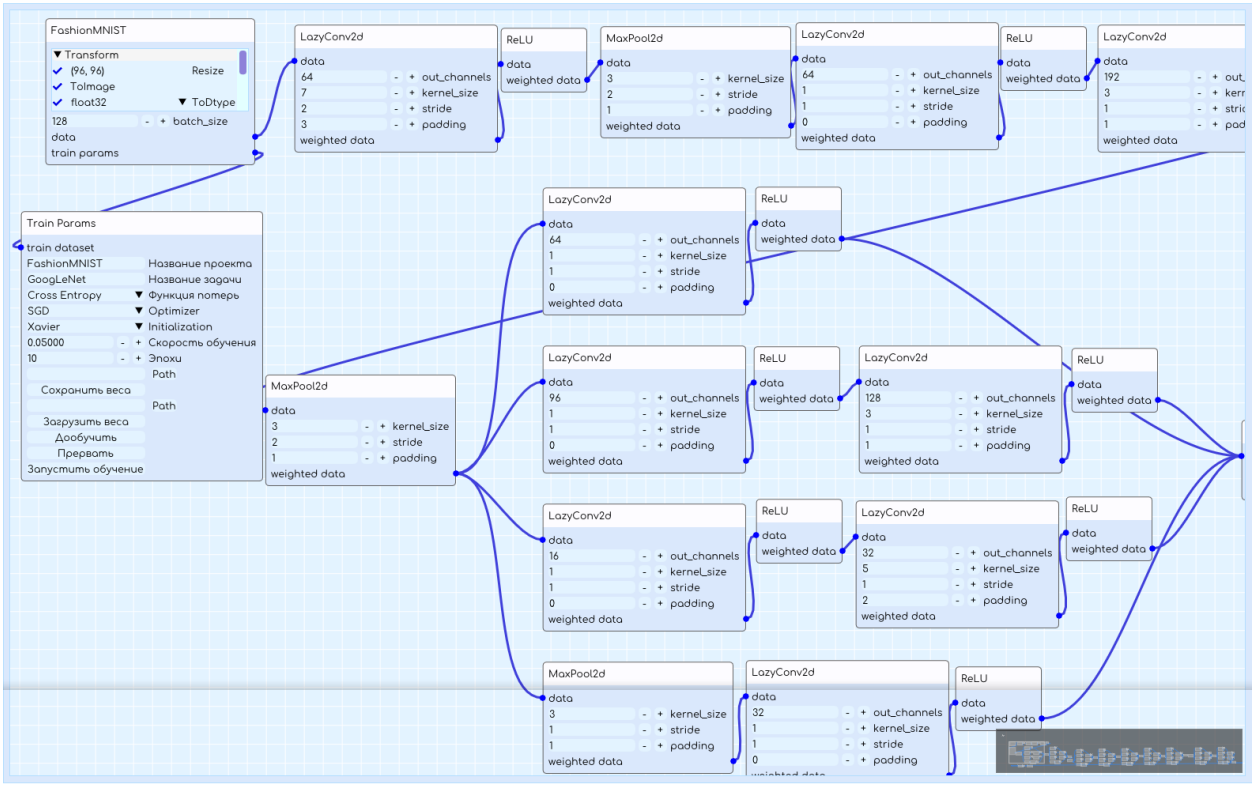

Рисунок Ж.9 – Обучение архитектуры GoogLeNet на датасете FashionMNIST

Рисунок Ж.10 – Мониторинг метрик по время обучения сети GoogLeNet

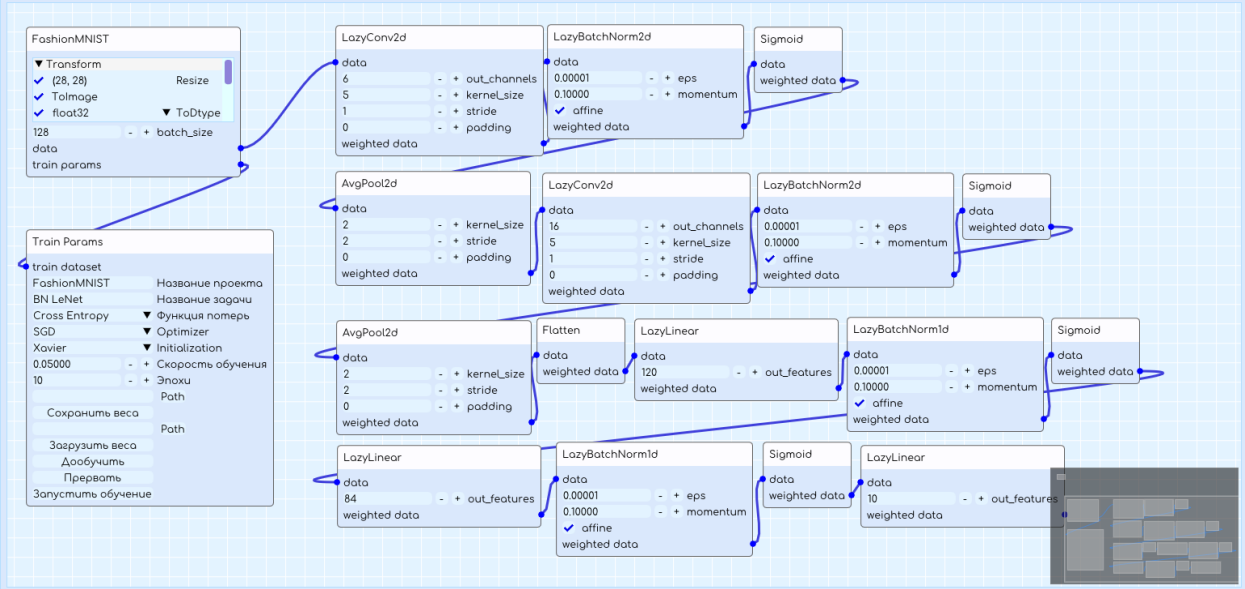

Рисунок Ж.11 – Обучение архитектуры BN LeNet на датасете FashionMNIST

Рисунок Ж.12 – Мониторинг метрик по время обучения сети BN LeNet