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)

СИСТЕМА КОНСТРУИРОВАНИЯ НЕЙРОННЫХ СЕТЕЙ

Отчет
по результатам
производственной практики: преддипломная практика

Обучающийся гр. 430-2

_____ А.А. Лузинсан
(подпись) (И.О. Фамилия)

_____
(дата)

Руководитель практики от профильной организации:
Ген. директор ООО «Девинсайд»
(должность, ученая степень, звание)

_____ И.Д. Тикшаев
(подпись) (И.О. Фамилия)

_____
(оценка)

М.П. _____
(дата)

Руководитель практики от Университета:
Профессор каф. АСУ ТУСУР, к.юр.н.,PhD

_____ С.М. Левин
(подпись) (И.О. Фамилия)

_____ _____
(оценка) (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
к.т.н., доцент
Романенко В.В.
_____
(подпись)
05.02.2024

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

на производственную практику: преддипломная практика
студенту гр. 430-2 факультета систем управления
Лузинсан Анастасии Александровне

1. Тема практики: Система конструирования нейронных сетей.
2. Цель практики: подготовка к выполнению выпускной квалификационной работы по автоматизации процесса проектирования архитектур нейронных сетей в академических и исследовательских целях.
3. Сроки прохождения практики: 05.02.2024 – 11.05.2024

Совместный рабочий график (план) проведения практики

№ п/п	Перечень заданий	Сроки выполнения
1	Прохождение техминимума и тех. безопасности на рабочем месте.	05.02.2024
2	Ознакомление со структурой предприятия, видами деятельности, процессами организации управления деятельностью предприятия	12.02.2024
3	Изучение целей и функций автоматизации технологических процессов, автоматизированных систем управления, используемых средств вычислительной техники в деятельности предприятия	19.02.2024
4	Изучение и освоение различных пакетов программ, применяемых на предприятии	26.02.2024
5	Постановка задачи автоматизации (описание предметной области, разработка требований к программному продукту)	11.03.2024
6	Обзор аналогов для решения задачи	18.03.2024
7	Проектирование системы	01.04.2024
8	Реализация системы	29.04.2024
9	Написание отчета по практике и создание презентации с основными результатами	06.05.2024

Дата выдачи: 05.02.2024 г.

Руководитель практики от университета

Профессор каф. АСУ, к.юр.н., PhD _____ С.М. Левин
(должность) (Подпись) (Ф.И.О.)

Согласовано:

Руководитель практики от профильной организации

Ген. Директор ООО «Девинсайд» _____ Тикшаев И.Д.
(должность) (Подпись) (Ф.И.О.)

М.П.

Задание принял к исполнению 05.02.2024 г.

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

Оглавление

Введение

Производственная практика: «Преддипломная практика» проходила в компании ООО «Девинсайд».

Для исследований в ходе практики была выбрана тема «Система конструирования нейронных сетей».

В условиях экспоненциального роста сферы искусственного интеллекта и машинного обучения, разработка интуитивных и доступных инструментов для конструирования нейронных сетей является немаловажной задачей. Существует потребность у начинающих исследователей и учащихся в понятном и бесплатном инструменте, основанном на принципах blueprint, который сделает процесс создания, обучения и применения глубоких нейронных сетей намного проще. Это способствует ускоренному пониманию принципов проектирования архитектур глубоких нейронных сетей, а также их применения в различных областях.

В условиях ограниченного доступа к определенным ресурсам в России, разработанная система представляет собой альтернативу уже существующим системам. Однако, вопреки последним, является проектом с открытым исходным кодом, что позволяет модернизировать систему под свои нужды.

Кроме того, система применима в научных исследованиях, где требуется быстрая разработка и тестирование новых архитектур нейронных сетей. Это позволит исследователям сосредоточиться на разработке новых идей, а не тратить время на реализацию базовых функций.

Целью практики является подготовка к выполнению выпускной квалификационной работы по автоматизации процесса проектирования архитектур нейронных сетей в академических и исследовательских целях.

Основные задачи:

- ознакомление со структурой компании, видами деятельности, процессами организации управления деятельностью компании;

- изучение целей и функций автоматизации технологических процессов, автоматизированных систем управления, используемых средств вычислительной техники в действиях данной компании;
- изучение и освоение различных пакетов программ, применяемых в компании;
- постановка задачи автоматизации (описание предметной области, разработка требований к программному продукту);
- обзор аналогов для решения задачи;
- проектирование системы;
- реализация системы.

Объектом исследования является научная деятельность специалиста по глубокому обучению.

Предмет исследования – методы, технологии и системы проектирования нейронных сетей.

В ходе исследования осуществлялся тематический поиск и критический анализ научно-технической информации, выполнено моделирование системы с помощью UML-диаграмм, спроектирован графический интерфейс, а также разработано программное обеспечение в виде десктопного приложения.

1 ОБЩАЯ ХАРАКТЕРИСТИКА ПРЕДПРИЯТИЯ ООО «ДЕВИНСАЙД»

1.1 Общая характеристика предприятия

Общество с ограниченной ответственностью «Девинсайд» расположено в Томской области, г. Томск.

Компания занимается разработкой компьютерного программного обеспечения, консультированием, управлением компьютерным оборудованием и другими смежными услугами.

Основная деятельность компании связана с разработкой аналитической системы "Tenderchad", специализированной на оценке релевантности тендеров для софтверных компаний. Эта система значительно оптимизирует процесс поиска и оценки тендерных заявок, обеспечивая более эффективную работу и экономию времени для клиентов.

Кроме того, компания активно занимается разработкой и внедрением моделей машинного обучения. Это позволяет "Девинсайд" создавать инновационные интеллектуальные системы, способные адаптироваться к изменяющимся условиям рынка и предсказывать будущие тренды в сфере компьютерных технологий. Внедрение таких технологий позволяет компании оставаться на передовой позиции в индустрии и предлагать клиентам самые передовые решения.

1.2 Научная деятельность специалиста по глубокому обучению

Наука о данных — это раздел информатики, изучающий проблемы анализа, обработки и представления данных в цифровой форме. Она объединяет методы по обработке данных в условиях больших объёмов и высокого уровня параллелизма, статистические методы, методы интеллектуального анализа данных и приложения искусственного интеллекта для работы с данными, а также методы проектирования и разработки баз данных [1]. В данной сфере деятельности выделяют три основные роли:

инженер данных, аналитик данных и специалист по науке о данных.

Чтобы вести бизнес, основываясь на данных, компании собирают информацию с сайтов, приложений, камер видеонаблюдения и датчиков на производстве и отдают их специалистам для анализа. Эти данные помогают прогнозировать спрос, изучать поведение клиентов, планировать эффективные рекламные кампании. Но прежде чем работать с информацией, её нужно собрать, грамотно структурировать, где-то сохранить, а потом доставить до аналитика. Для этого нужна инфраструктура: хранилища, серверные мощности для анализа, инструменты для сбора, очистки и доставки данных.

Инженер данных — это специалист, который занимается построением и обслуживанием инфраструктуры для работы с данными, а также их предварительной обработкой. Инженер не участвует в анализе, но обеспечивает аналитиков нужными мощностями, инструментами и подготовленными данными [2].

Аналитик данных проводит статистический анализ, чтобы ответить на вопросы или решить проблемы. Для этого он получает данные, выявляет закономерности и формирует отчёты, которые помогают руководителям проекта или бизнеса принимать стратегические решения.

Специалист по науке о данных создаёт инструменты для решения задач бизнеса. Для этого он использует навыки анализа данных и построения моделей машинного и глубокого обучения. Специалист работает на стыке нескольких областей знания: статистики, программирования, машинного и глубокого обучения. Поэтому он не только умеет анализировать и визуализировать данные, но и строить модели на их основе, заниматься прогнозированием и оптимизацией бизнес-процессов [3].

Однако основной целевой аудиторией данной системы являются исследователи в области глубокого обучения. Эти специалисты отвечают за проведение исследований и анализ данных для разработки новых технологий, продуктов и процессов. Таким образом, как правило, они отвечают за

разработку новых алгоритмов и моделей, которые могут быть использованы для решения сложных задач.

В связи с этим становится ясной необходимость в инструменте, который бы позволял упростить процесс проведения исследований во время разработки новых алгоритмов.

2 ПОСТАНОВКА ЗАДАЧИ

Целью разработки системы является создание инструмента, автоматизирующего процесс проектирования и тестирования архитектур моделей глубокого обучения.

Цели системы:

1. Разработать интуитивно понятный интерфейс для построения, обучения и тестирования моделей глубокого обучения.

2. Реализовать механизмы мониторинга процесса обучения моделей с возможностью визуализации результатов.

3. Обеспечить пользователей системы инструментами для тонкой настройки параметров моделей, включая гиперпараметры.

4. Предоставить возможность анализа результатов обучения с помощью статистических методов и визуализации данных.

Описание функциональности включали в себя следущие моменты:

1. Пользователи должны иметь возможность создавать и настраивать архитектуры нейронных сетей с помощью графического интерфейса или редактирования сценариев кода.

2. Система должна отображать информацию о процессе обучения моделей, включая метрики точности, функции потерь и другие параметры.

3. Пользователи должны иметь возможность регулировать параметры обучения, такие как тип регуляризатора, функция потерь, оптимизатор, скорость обучения и другие гиперпараметры. Помимо этого, пользователю должна быть предоставлена возможность скачать и загрузить веса обученной модели.

4. Система должна предоставлять инструменты для сравнения моделей, анализа статистических показателей и визуализации данных обучения, чтобы помочь пользователям принимать информированные решения о выборе оптимальной модели.

Задание также подразумевало наличие следующих требований и

ограничений:

1. Интерфейс должен быть интуитивно понятным и удобным для пользователей всех уровней квалификации.
2. Система должна обеспечивать эффективное использование вычислительных ресурсов при обучении моделей с большими объемами данных.
3. Система должна быть стабильной и надежной, обеспечивая сохранность данных и предотвращение потерь в случае сбоев.
4. Необходимо следить за соблюдением авторских прав и лицензионных соглашений при использовании сторонних библиотек и инструментов.
5. Необходимо соблюдать принципы безопасности и конфиденциальности данных пользователей.
6. Необходимо учитывать требования к совместимости существующих систем и технологий в области глубокого обучения.

3 ОБЗОР АНАЛОГОВ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

В связи со стремительным развитием сферы машинного и глубокого обучения, существует уже немало инструментов, обеспечивающих достижение поставленных целей. Однако, с недавних пор, данные сервисы перестали быть доступны на территории России. Тем не менее, для поддержания полной картины, далее будут перечислены все аналоги.

Azure Machine Learning — сервис, предоставляющий облачные вычислители для создание критически важных для бизнеса крупномасштабных моделей машинного обучения. Он позволяет настраивать и развертывать базовые модели из OpenAI, Hugging Face и Meta с помощью каталога моделей [4].

Loginom – 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. Делает продвинутую аналитику доступной конечным пользователям без привлечения IT-специалистов, позволяя автоматизировать бизнес-процессы и кратно ускорить работу с данными [5].

IBM Watson Studio позволяет специалистам по обработке и анализу данных, разработчикам и аналитикам создавать, запускать модели ИИ и управлять ими, а также оптимизировать принятие решений в любом месте IBM Cloud Pak для данных. Существует возможность объединения команд и платформ с открытым исходным кодом, таких как PyTorch, TensorFlow, scikit-learn и инструментами IBM с ее экосистемой для обработки и анализа данных на основе кода и визуальной обработки данных [6].

Сравнение функциональных возможностей систем представлено в таблице 3.1.

Таблица 3.1 — Сравнительный анализ функциональных возможностей

Система	Функции				
	Понятный интерфейс	Дополнительный образовательный материал	Журналирование обучения	Кастомизация моделей	Доступен в России
Azure Machine Learning	-	+	+	+	-
Loginom	+	+	+	-	+
IBM Watson Studio	+	+	+	+	-

Таким образом, рассмотренные аналоги либо представляют из себя хорошие решения для анализа данных, но с недостаточным функционалом по кастомизации моделей, либо не доступны в России.

Эти факты подтверждают необходимость разработки собственного продукта.

4 ПРОЕКТИРОВАНИЕ СИСТЕМЫ

Проектирование системы играет ключевую роль в разработке программного обеспечения, обеспечивая эффективную и гибкую архитектуру системы.

Поскольку в разделе 2 были описаны общие требования системы, далее было выполнено проектирование классов с помощью UML, что в дальнейшем обеспечит понятность и поддерживаемость кода, а также облегчит добавление новых функций и модификацию существующих. Кроме того, было уделено внимание выбору подходящих паттернов проектирования, чтобы обеспечить гибкость, масштабируемость и удобство использования системы.

4.1 Проектирование пользовательского интерфейса

В этом подразделе мы рассмотрим каждый класс в отдельности, опишем его функциональность, свойства и методы, а также обсудим его взаимодействие с другими компонентами системы.

В результате проектирования классов, отвечающих за интерфейс системы, была получена диаграмма классов, представленная на рисунке 4.1

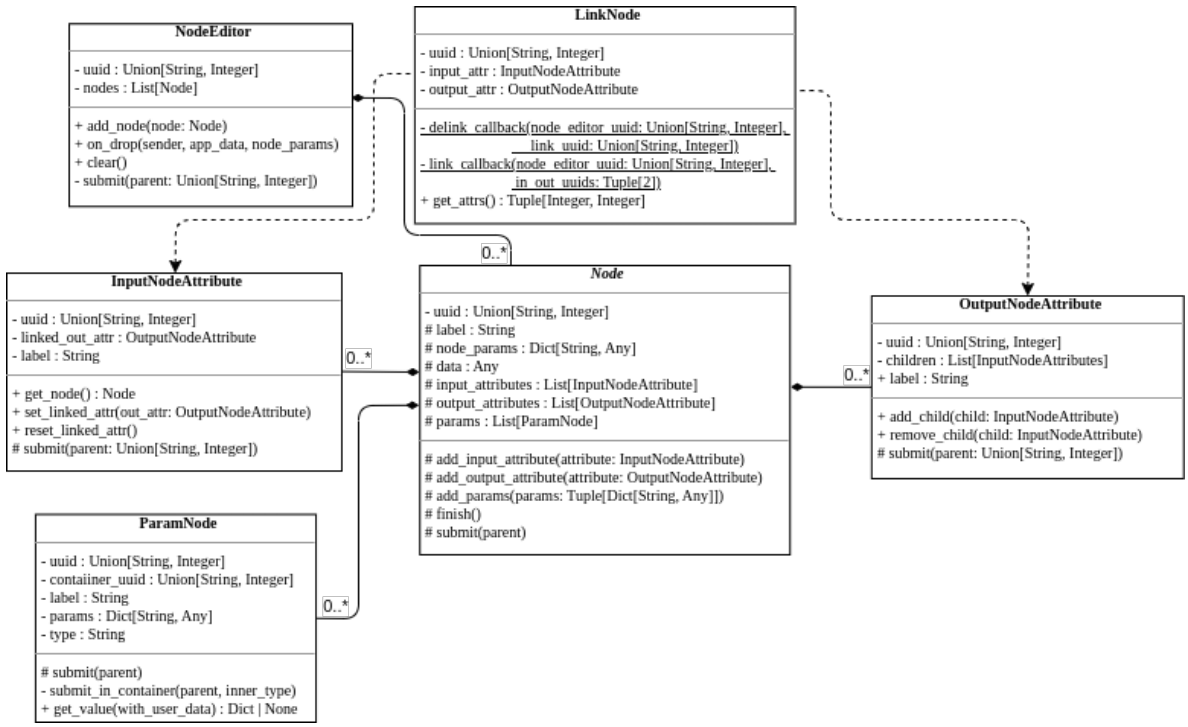

Рисунок 4.1 — Диаграмма классов уровня интерфейса системы

Во всех классах дублируется атрибут uuid, предназначенный для сохранения идентификатора элемента интерфейса. Поэтому далее будет опущено описание этого атрибута.

Более детальное описание атрибутов, методов и назначений интерфейсных классов приведено в таблице 4.1.

Таблица 4.1 — Описание классов пользовательского интерфейса

Класс (описание)	Атрибуты	Методы
NodeEditor — класс, представляющий из себя рабочее поле, в котором размещаются все узлы	• nodes — список узлов, инициализированных в рабочем поле.	• add_node(node: Node) — добавляет узел Node в список узлов nodes; • 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; • submit(parent) — инициализирует и прикрепляет рабочее поле к главному окну.
Node — абстрактный класс, отвечающий за отображение узла в рабочем поле, а также хранящий информацию, которая будет использована во время обучения сети.	• label — заголовок, отображаемый у узла; • 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.	• 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; • finish() — привязывает цветовую схему к узлу и его дочерним элементам; • submit(parent) — инициализирует и прикрепляет к рабочему полю узел и все его дочерние элементы;

Окончание таблицы 4.1

Класс (описание)	Атрибуты	Методы
ParamNode — класс, генерирующий параметры узла.	• container_uuid — идентификатор, непосредственно привязываемый к элементу статического атрибута; • label — подпись параметра; • params — параметры генерируемого элемента атрибута • type — тип атрибута, сопоставимый с одним из поддерживаемых в интерфейсе типов	• 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.
InputNodeAttribute — класс, генерирующий входную точку данных узла.	• label — подпись входного атрибута; • linked_out_attr — ссылка на экземпляр выходного атрибута типа OutputNodeAttribute	• get_node() — возвращает экземпляр класса Node, к которому прикреплен данный данный входной атрибут; • set_liked_attr(out_attr) — связывает текущий входной атрибут с указанным выходным атрибутом типа OutputNodeAttribute; • reset_linked_attr() — очищает ссылку на выходной атрибут; • submit() — инициализирует входной атрибут и прикрепляет к узлу.
OutputNodeAttribute — класс, генерирующий выходную точку данных узла.	• label — подпись выходного атрибута; • children — список входных атрибутов, которые были соединены с текущим выходным атрибутом.	• 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; • 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.

4.2 Проектирование ресурсных классов

На следующем шаге были определены классы ресурсов, диаграмма которых представлена на рисунке 4.2.

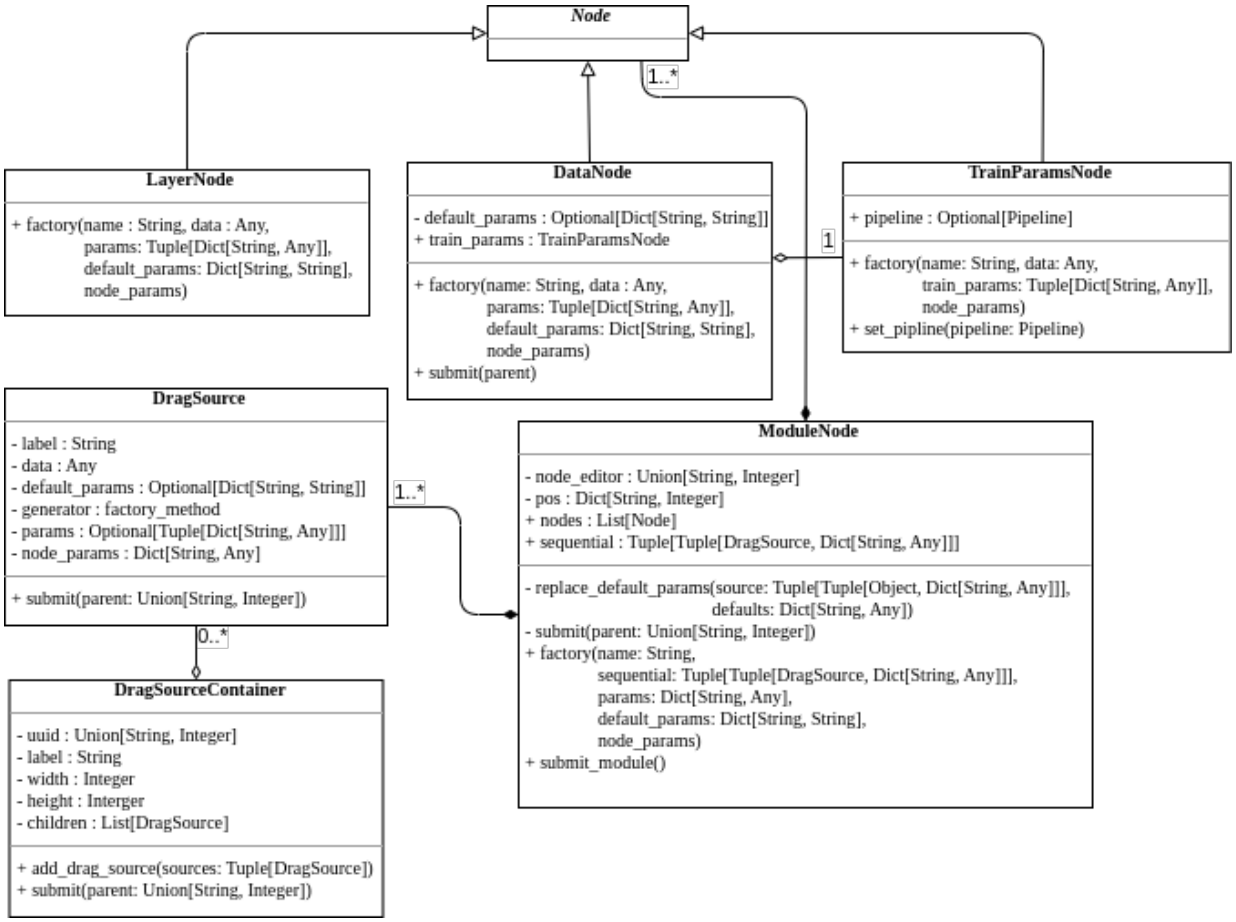

Рисунок 4.2 — Диаграмма классов уровня ресурсов системы

Во время проектирования ресурсных классов было принято решение использовать фабричный паттерн проектирования. Фабричный метод (англ. Factory Method), или виртуальный конструктор (англ. Virtual Constructor) — это порождающий шаблон проектирования, предоставляющий подклассам (дочерним классам, субклассам) интерфейс для создания экземпляров некоторого класса. 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 [7].

Описание атрибутов и методов соответственно приведено в таблице 4.2.

Таблица 4.2 — 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 — наследний класса Node, автоматически генерируемый 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.
ModuleNode — класс, аггрегирующий узлы Node, что позволяет проектировать глубокие модульные структуры нейронных сетей.	• node_editor — идентификатор рабочего поля, в котором будет разворачиваться модуль; • pos — относительная позиция узла, инкрементируемая по мере размещения модулей; • nodes — обработанная последовательность узлов сети; • sequential — необработанная последовательность ресурсов.	• factory(name, data, params, default_params) — фабричный метод, инициализирующий и возвращающий экземпляр класса ModuleNode. • 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 4.2

Класс (описание)	Атрибуты	Методы
DragSource — класс-генератор, инициализирующий узел или сеть после перетаскивания в рабочее поле.	• label — подпись кнопки ресурса; • 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.	• submit(parent) — инициализирует кнопку и загрузочный ресурс, используемый в дальнейшем для вызова фабричного метода.
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.

5 РАЗРАБОТКА СИСТЕМЫ

5.1 Предварительная настройка проекта

При разработке системы важно обеспечить правильную настройку рабочего окружения, которая позволит эффективно выполнять задачи разработки. В данном подразделе описывается процесс разворачивания проекта в операционной системе ядра Linux с использованием текстового редактора Visual Studio Code, менеджера зависимостей Poetry, фреймворка DearPyGui, PyTorchLightning, а также системы версионирования ClearML.

В качестве интегрированной среды разработки (IDE) выбран Visual Studio Code, который является популярным инструментом для разработки программного обеспечения. После установки Visual Studio Code были скачаны расширения "Python" и "Mypy Type Checker" от Microsoft, чтобы обеспечить поддержку Python и удобное редактирование кода.

Для управления зависимостями Python проекта был выбран менеджер Poetry. После установки Poetry был создан новый проект путем выполнения команды poetry new dlc. Затем были добавлены необходимые зависимости посредством редактирования файла pyproject.toml с помощью команды poetry add <package_name>.

В качестве фреймворка для создания графического пользовательского интерфейса (GUI) используется DearPyGui. Для его использования необходимо было установить библиотеку с помощью менеджера зависимостей Poetry. DearPyGui предоставляет простой и интуитивно понятный интерфейс для создания интерактивных приложений.

В процессе разработки системы был внедрен PyTorch Lightning для сокращения избыточного кода и упрощения процесса обучения моделей. Использование данной библиотеки позволило абстрагировать множество деталей, связанных с обучением моделей, и сосредоточиться на ключевых аспектах проектирования архитектуры системы. Этот высокоуровневый интерфейс предоставил удобные методы для определения структуры модели,

процесса обучения и валидации, а также обработки результатов.

Для версионирования и отслеживания изменений в экспериментах используется система ClearML. После установки ClearML на систему необходимо зарегистрироваться на платформе clearml.ai и получить API ключ. Далее API ключ используется для инициализации ClearML в проекте. ClearML предоставляет удобный интерфейс для отслеживания экспериментов, управления данными и многое другое. Для обеспечения надежной и эффективной работы разработчика необходимо также настроить сервер для системы версионирования ClearML. Для этого следует убедиться, что clearml-server установлен и запущен на системе. После запуска сервера его можно настроить для хранения и отслеживания экспериментов системы.

Кроме того, для работы с тестовыми данными были интегрированы датасеты из torchvision.datasets. Эти датасеты предоставляют удобный способ доступа к широкому спектру стандартных наборов данных для обучения и тестирования моделей машинного обучения.

Для создания и обучения моделей глубокого обучения были использованы слои из модуля torch.nn. PyTorch предоставляет обширный набор инструментов и возможностей для разработки и обучения различных типов нейронных сетей.

Для аугментации данных, то есть изменения их входных изображений с целью улучшения обучения моделей, были использованы методы трансформации из библиотеки torchvision.transforms. Эти трансформации позволяют создавать разнообразные варианты входных данных для улучшения обобщающей способности моделей.

5.2 Реализации классов

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. Эти классы разбиты на отдельные файлы в соответствии с принципом разделения ответственности, что облегчает поддержку и понимание кода.

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 содержит 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.

5.3 Механизм передачи ресурсов

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 указывается в параметре node_generator. 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, требующие параметры node_generator, data, params, default_params и node_params. Во время при 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.

6 РЕЗУЛЬТАТЫ РАБОТЫ

6.1 Интегрированные датасеты

В рамках системы конструирования нейронных сетей пользователю предоставляется широкий выбор поддерживаемых датасетов, что обеспечивает гибкость и универсальность в проектировании моделей машинного обучения. Среди доступных датасетов имеются как широко используемые стандартные наборы данных, такие как FashionMNIST, CIFAR10 и ImageNet, так и более специализированные наборы, включая Caltech101, Caltech256, Cityscapes и EuroSAT.

Пользователь может выбирать подходящий датасет в зависимости от конкретной задачи или области применения своей модели. Например, для задач классификации изображений пользователь может использовать CIFAR10, Flowers102 или Food101, в то время как для сегментации изображений подойдут Cityscapes или CocoCaptions. Кроме того, для пользователей, у которых есть собственные данные, предусмотрена возможность использовать свои датасеты, загружаемые из файла. Список доступных датасетов с выбранным набором FashionMNIST представлен на рисунке 6.1.

В дополнение к широкому выбору поддерживаемых датасетов, система предоставляет возможности для аугментации данных, что позволяет улучшить обобщающую способность моделей и сделать их более устойчивыми к различным условиям входных данных. Пользователям доступны различные методы аугментации, такие как AutoAugment, RandomIoUCrop, ElasticTransform, Grayscale, RandomCrop, RandomVerticalFlip, RandomHorizontalFlip и другие.

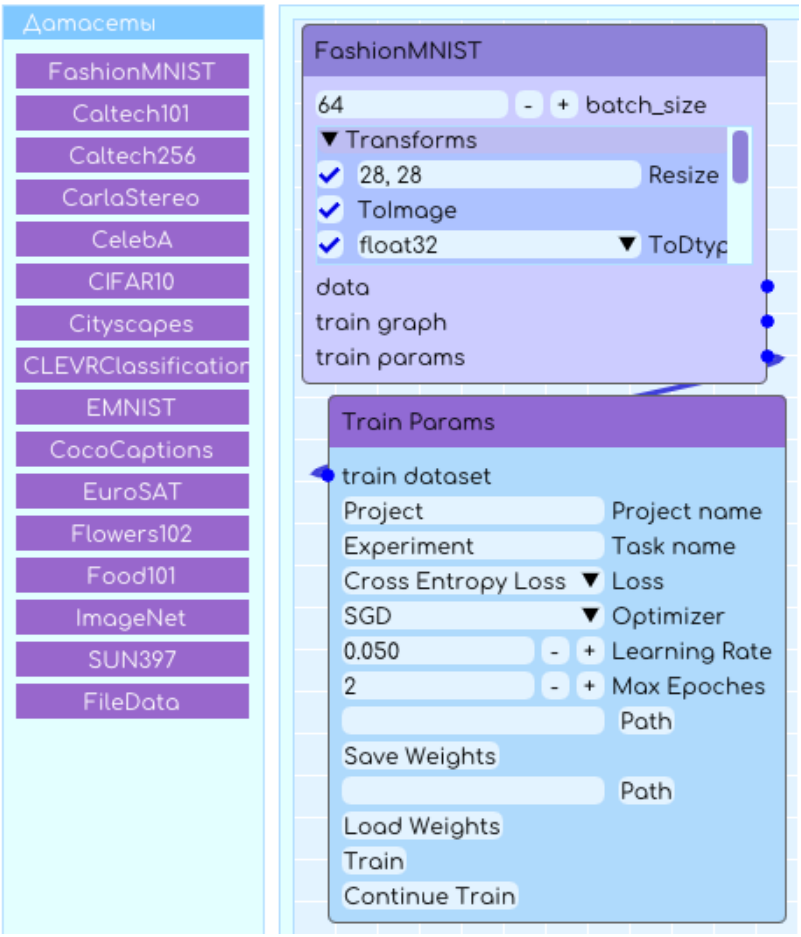

Рисунок 6.1 — Доступные датасеты и часть рабочей области

Аугментация данных происходит на этапе предобработки, прежде чем данные будут переданы в модель для обучения. Например, методы RandomCrop и RandomVerticalFlip позволяют создавать случайные обрезки и отражения изображений, что помогает увеличить разнообразие тренировочных данных и сделать модель более устойчивой к различным расположениям объектов на изображениях. Методы AutoAugment и ElasticTransform предоставляют возможность автоматической аугментации данных, основанной на оптимизации определенных критериев, таких как точность классификации или сегментации.

Кроме того, к обязательным методам аугментации данных относятся Resize, ToImage и ToDtype, которые обеспечивают единообразный формат данных для обучения моделей и улучшают производительность обработки данных. Эти методы являются важной частью процесса предобработки данных и обеспечивают согласованность и стабильность входных данных для модели.

6.2 Настройка обучения

В системе предоставляется возможность тонкой настройки процесса обучения моделей, что включает в себя выбор оптимальной функции потерь и оптимизатора, а также регулировку параметров обучения для достижения лучших результатов.

Функция потерь (Loss Function) является ключевым компонентом в оценке качества модели. Она представляет собой математическое выражение, которое измеряет разницу между предсказанными значениями модели и фактическими данными. В системе доступны различные функции потерь, каждая из которых подходит для определенных типов задач. Например:

- Mean Squared Error (MSE) вычисляет среднеквадратичное отклонение между предсказанными и фактическими значениями и широко используется в задачах регрессии.
- Cross Entropy Loss часто применяется в задачах классификации и вычисляет разницу между распределением вероятностей, предсказанных моделью, и фактическими метками классов.
- Mean Absolute Error (MAE) является средней абсолютной ошибкой, измеряющей среднее абсолютное отклонение предсказанных значений от фактических.
- Root Mean Squared Error (RMSE) представляет из себя квадратный корень из среднеквадратичной ошибки, обеспечивающий измерение среднеквадратичного отклонения.
- R2 Score (R-квадрат) — это коэффициент детерминации, который оценивает пропорцию вариации зависимой переменной, которая объясняется моделью.
- F1 Score используется в задачах бинарной классификации и является гармоническим средним между precision и recall.

Оптимизатор, предоставленный пользователям, отвечает за обновление весов модели с целью минимизации функции потерь в процессе обучения. В

системе доступны различные оптимизаторы, каждый из которых имеет свои особенности и преимущества:

- Stochastic Gradient Descent (SGD) является классическим оптимизатором, который обновляет веса модели в направлении антиградиента функции потерь.
- Adam — это адаптивный метод оптимизации, который корректирует скорость обучения для каждого параметра на основе оценки первого и второго моментов градиентов.
- Adadelta — также адаптивный метод оптимизации, который регулирует скорость обучения на основе накопленной истории градиентов.
- Adamax является вариацией метода Adam, которая использует бесконечную норму вместо второго момента градиентов.

Кроме выбора функции потерь и оптимизатора, пользователи также могут настраивать параметры обучения, такие как learning rate (скорость обучения) и количество эпох обучения (max epochs), чтобы достичь оптимальной производительности модели. Кроме того, система позволяет настраивать параметры для инициализации эксперимента в ClearML, такие как название проекта и задачи, для более удобного управления обучением и анализа результатов.

6.3 Интегрированные модули

В системе реализовано множество слоев, которые представляют собой ключевые компоненты нейронных сетей и играют важную роль в их функционировании. Среди этих слоев можно выделить следующие:

- LazyLinear — слой, который представляет линейное преобразование, выполняющееся над входными данными с использованием весов и смещения.
- LazyBatchNorm1d, LazyBatchNorm2d, LazyBatchNorm3d — представляют собой слои пакетной нормализации, которые выполняют нормализацию активаций на каждом батче данных.

- LazyConv1d, LazyConv2d, LazyConv3d — сверточные слои, которые применяют операцию свертки к входным данным с использованием ядра свертки.

- Flatten - слой выполняющий операцию "сплющивания", преобразуя многомерные данные в одномерный вектор.

- AvgPool2d, MaxPool2d — слои, представляющие собой слои пулинга, которые выполняют операцию усреднения или выбора максимального значения на определенных областях входных данных.

- Dropout — слой, который применяет операцию исключения (выключения) случайных нейронов во время обучения, что помогает предотвратить переобучение модели.

- ReLU, Softmax, Tanh, GELU — слои представляют собой функции активации, которые применяются к выходам нейронов для введения нелинейности в модель. Функция ReLU (Rectified Linear Unit) вычисляется по формуле ReLU(x)=max(0,x), Softmax преобразует выходы в вероятностные распределения, Tanh преобразует выходы в диапазон от -1 до 1, а GELU (Gaussian Error Linear Unit) представляет собой аппроксимацию нормального распределения и широко используется в современных архитектурах глубоких нейронных сетей.

Эти слои представляют собой основные строительные блоки нейронных сетей и позволяют конструировать разнообразные архитектуры моделей для решения различных задач глубокого обучения. Список доступных слоев и примеры отображения некоторых слоев представлены на рисунке 6.2.

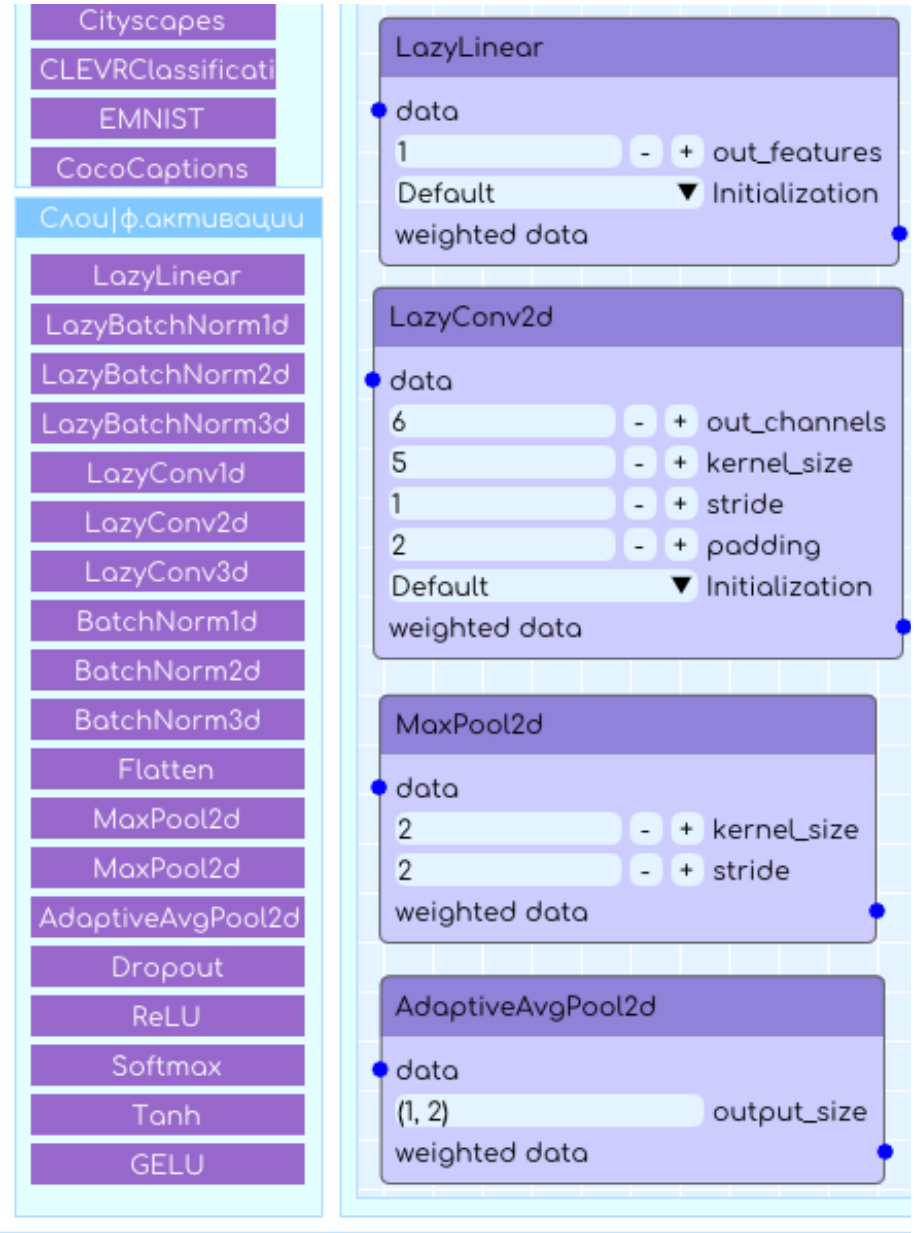

Рисунок 6.2 — Список доступных слоев

Пример реализации многослойного перцептрона с одним скрытым слоем на 256 нейронов представлен на рисунке 6.3. Помимо этого, мониторинг обучения сети с помощью ClearMl представлен на рисунке 6.4.

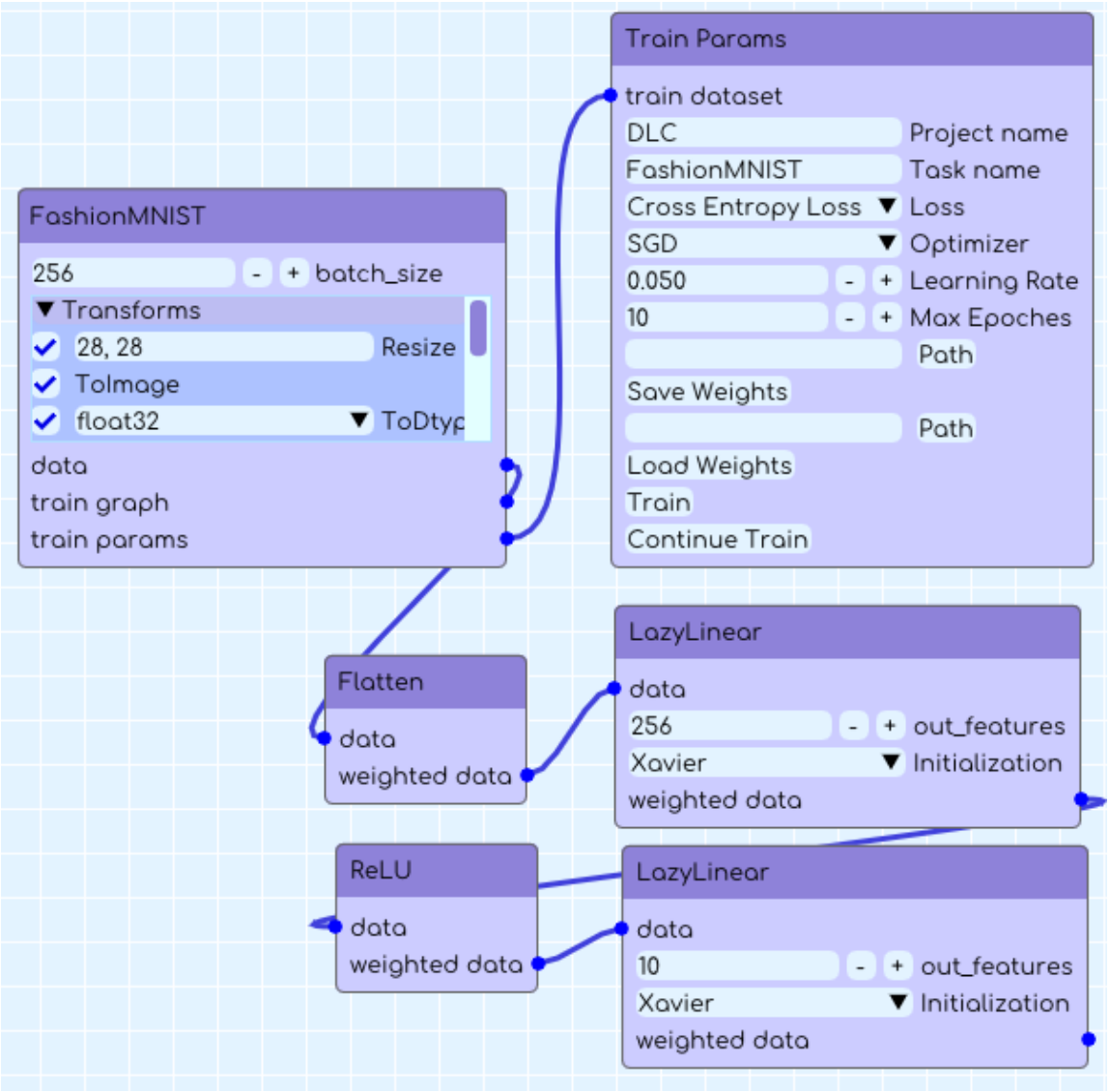

Рисунок 6.3 — Пример реализации многослойного перцептрона

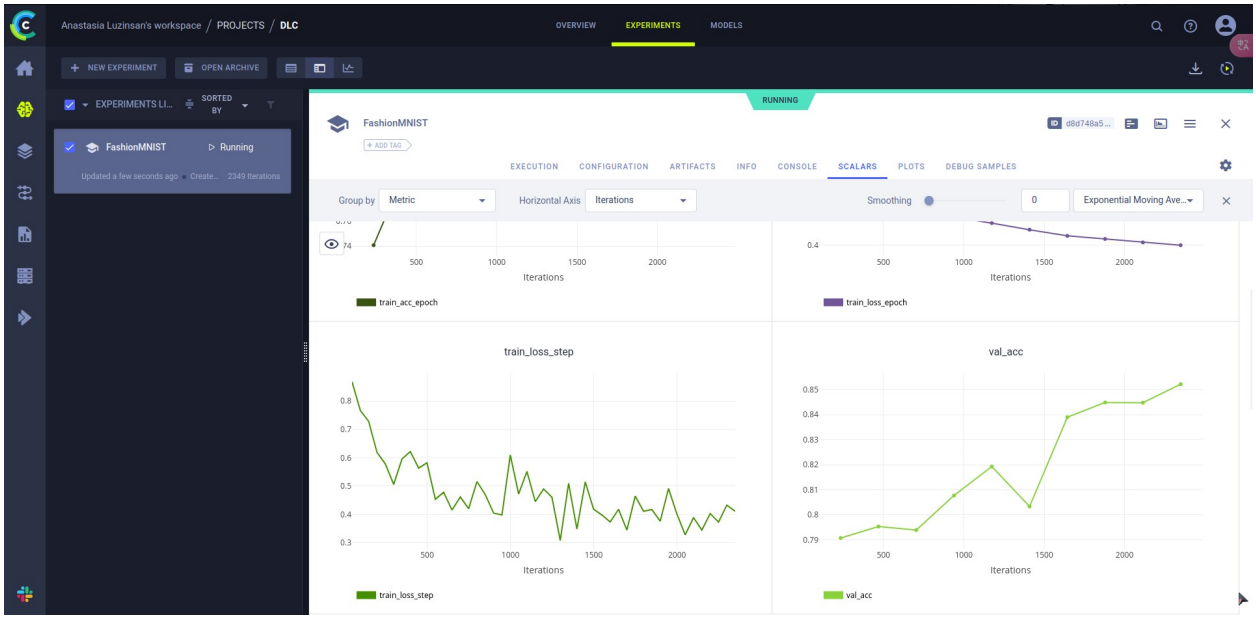

Рисунок 6.4 — Мониторинг обучения сети в ClearMl

Как можно видеть из рисунка 6.3, для каждого узла доступна настройка параметров, что позволяет пользователю адаптировать поведение узлов в соответствии с требованиями и ограничениями конкретной задачи.

Например, для сверточных слоев доступны параметры, такие как размер ядра свертки (kernel_size), шаг (stride) и дополнение (padding). Размер ядра свертки определяет размер фильтра, применяемого к входным данным, в то время как шаг управляет смещением окна при выполнении свертки, и дополнение позволяет контролировать размер выходных данных путем добавления нулевых значений по краям входных данных. Пулинговые слои также поддерживают настройку различных параметров, таких как размер окна пулинга (kernel_size) и шаг (stride). Размер окна пулинга определяет размер области, по которой выполняется операция пулинга, а шаг указывает на величину смещения окна при применении этой операции. Для полносвязных слоев пользователь может настраивать количество выходных нейронов (out_features) и выбирать функцию активации, которая будет применена к выходам нейронов. Следовательно, настройка параметров таких слоев позволяет пользователю контролировать сложность модели и ее способность к обобщению. Следует также упомянуть, что слои нормализации поддерживают настройку различных параметров, включая размер пакета (batch_size) и эпсилон (eps). Размер пакета определяет количество данных в пакете, на котором выполняется нормализация, а эпсилон добавляет небольшое значение к дисперсии для стабилизации вычислений.

Таким образом, настройка параметров для различных типов слоев предоставляет пользователям возможность гибкого проектирования сетей, соответствующих специфическим требованиям задач глубокого обучения.

6.4 Интегрированные архитектуры сетей

На последнем этапе реализации были внедрены готовые архитектуры нейронных сетей, таких как LeNet, VGG, AlexNet, NiN, NiN Net, GoogLeNet,

ResNet, ResNeXt и DenseNet. Эти архитектуры широко используются в области глубокого обучения и представляют собой эффективные модели для различных задач, включая классификацию изображений, обнаружение объектов и сегментацию.

- LeNet [8], разработанная Яном ЛеКуном в 1998 году, является одной из первых сверточных нейронных сетей, применяемых для распознавания рукописных символов. Эта архитектура включает в себя последовательность сверточных слоев, слоев подвыборки и полносвязных слоев, что делает ее эффективной для классификации изображений.
- VGG (Visual Geometry Group) [9] — еще одна популярная архитектура нейронной сети, разработанная исследователями из Visual Geometry Group в Университете Оксфорда. VGG отличается своей глубокой структурой, состоящей из последовательности сверточных слоев с малыми фильтрами, что позволяет ей успешно применяться для классификации изображений.
- AlexNet [10] является знаменитой архитектурой, которая сделала значительный вклад в развитие области глубокого обучения. Разработанная Алексеем Крижевским и его коллегами в 2012 году, она стала первой нейронной сетью, выигравшей соревнование ImageNet по классификации изображений с большим отрывом.
- NiN (Network in Network) [11] представляет собой архитектурный подход, введенный в работе "Network in Network" Миньцю Лином. Он предлагает заменить обычные сверточные слои на блоки, называемые "многослойными перцептронами в сети", что позволяет лучше моделировать нелинейные зависимости в данных.
- GoogLeNet [12], или Inception, представляет собой глубокую архитектуру нейронной сети, разработанную исследователями из Google. Его ключевая особенность заключается в использовании "модулей Inception", которые позволяют сети изучать и использовать признаки различных масштабов и аспектов изображений.
- ResNet (Residual Network) [13] — это архитектура нейронной сети,

которая внесла значительный вклад в решение проблемы затухающего градиента. Основная идея ResNet заключается в использовании "блоков-остатков", которые позволяют сети более эффективно обучаться на глубоких архитектурах.

- ResNeXt является дальнейшим развитием архитектуры ResNet, которая представляет собой сеть, состоящую из множества параллельных путей. Этот подход позволяет эффективно использовать ресурсы и улучшить производительность модели.

- DenseNet (Densely Connected Convolutional Network) представляет собой архитектуру нейронной сети, в которой каждый слой связан с каждым другим внутри блока. Это позволяет эффективно использовать признаки из предыдущих слоев и обеспечивает более плотные связи между слоями.

Реализация сети LeNet представлена на рисунке 6.5.

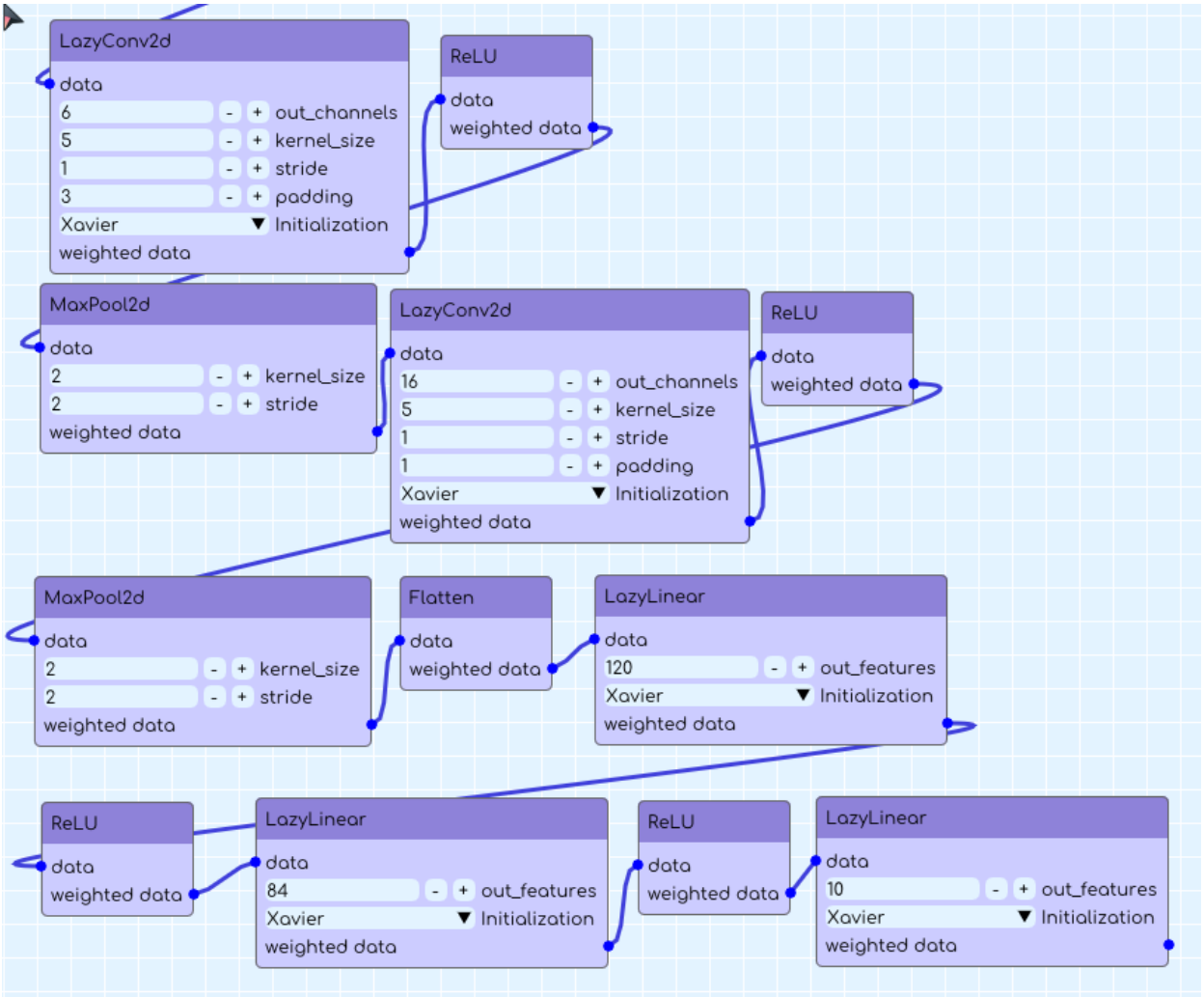

Рисунок 6.5 — Реализация сети LeNet

Внедрение данных архитектур нейронных сетей в систему позволяет пользователю эффективно использовать передовые методы глубокого обучения для решения различных задач в области компьютерного зрения и обработки изображений.

Заключение

В результате производственной практики, осуществленной в ООО "Девинсайд" и направленной на преддипломную работу, был достигнут значительный прогресс в изучении и понимании систем конструирования нейронных сетей. Цель практики, заключавшаяся в подготовке к выпускной квалификационной работе, была достигнута благодаря осуществлению ряда задач, охватывающих различные аспекты исследований в области глубокого обучения.

В ходе практики были рассмотрены и изучены структура компании, ее виды деятельности и процессы управления, что позволило получить полное представление о рабочей среде и условиях работы. Исследование целей и функций автоматизации технологических процессов, а также освоение различных пакетов программ, применяемых в компании, способствовали расширению знаний и навыков в области автоматизации и информационных технологий.

Особое внимание было уделено постановке задачи автоматизации, включающей в себя описание предметной области и разработку требований к программному продукту. Проведенный обзор аналогов для решения поставленной задачи позволил ознакомиться с существующими решениями на рынке и выделить основные преимущества и недостатки.

Наконец, реализация системы конструирования нейронных сетей была завершена, что позволило протестировать и оценить эффективность разработанных решений.

Список использованных источников

1. Peter Naur: Concise Survey of Computer Methods [Электронный ресурс]: электрон. книга. URL: http://www.naur.com/Conc.Surv.html (дата обращения: 01.05.2024).

2. Data Engineer [Электронный ресурс]: блог Практикума. URL: https://practicum.yandex.ru/blog/professiya-data-engineer/ (дата обращения: 01.05.2024).

3. Data Scientist [Электронный ресурс]: блог Практикума. URL: https://practicum.yandex.ru/blog/kto-takoy-data-scientist/ (дата обращения: 01.05.2024).

4. Azure Machine Learning [Электронный ресурс]: официальный сайт Azure: https://azure.microsoft.com/en-us/products/machine-learning#Features (дата обращения: 02.05.2024).

5. Loginom [Электронный ресурс]: официальный сайт Loginom. URL: https://loginom.ru/ (дата обращения: 02.05.2024).

6. Watson Studio [Электронный ресурс]: официальный сайт Watson Studio. URL: https://www.ibm.com/products/watson-studio (дата обращения: 02.05.2024).

7. Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. Приемы объектно–ориентированного проектирования. Паттерны проектирования = Design Patterns: Elements of Reusable Object-Oriented Software. — СПб.: «Питер», 2007, 366 с. (дата обращения: 03.05.2024).

8. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324 [Электронный ресурс]: электрон. журн. URL: http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf (дата обращения: 04.05.2024).

9. Krizhevsky, A., Sutskever, I., & Hinton, G.E. (2012). ImageNet classification with deep convolutional neural networks. Advances in Neural

Information Processing Systems [Электронный ресурс]: электрон. журн. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf (дата обращения: 04.05.2024).

10. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition [Электронный ресурс]: электрон. журн. URL: https://arxiv.org/abs/1409.1556 (дата обращения: 05.05.2024).

11. Lin, M., Chen, Q., & Yan, S. (2013). Network in network [Электронный ресурс]: электрон. журн. URL: https://arxiv.org/abs/1312.4400 (дата обращения: 05.05.2024).

12. Szegedy, C., Liu, W., Jia, Y., Sermanet, P, etc. Going deeper with convolutions. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition [Электронный ресурс]: электрон. журн. URL: https://www.researchgate.net/publication/265787949_Going_Deeper_with_Convolutions (дата обращения: 05.05.2024).

13. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition [Электронный ресурс]: электрон. журн. URL: https://ieeexplore.ieee.org/document/7780459 (дата обращения: 05.05.2024).