

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

УДК 004.032.26

**КРОЩЕНКО**  
Александр Александрович

**МЕТОДЫ ОБУЧЕНИЯ ГЛУБОКИХ НЕЙРОННЫХ СЕТЕЙ  
ДЛЯ ЗАДАЧ КОМПЬЮТЕРНОГО ЗРЕНИЯ**

Диссертация на соискание ученой степени  
кандидата технических наук  
по специальности 05.13.17 – Теоретические основы информатики

Научный руководитель  
доктор технических наук, профессор  
Головко В. А.

**Минск 2023**

# СОДЕРЖАНИЕ

Перечень сокращений и обозначений . . . . .	4
Введение . . . . .	6
Общая характеристика работы . . . . .	7
ГЛАВА 1 Проблемы обучения глубоких нейронных сетей . . . . .	11
1.1 Общие сведения . . . . .	11
1.2 Проблемы обучения глубоких моделей . . . . .	16
1.3 Методы обучения . . . . .	18
1.4 Критерии минимизации при обучении . . . . .	24
1.5 Выводы . . . . .	26
ГЛАВА 2 Метод неконтролируемого предобучения . . . . .	28
2.1 Обучение RBM . . . . .	28
2.2 Общее описание предлагаемого метода . . . . .	33
2.3 Вывод правил предобучения . . . . .	35
2.3.1 Критерий MSE . . . . .	35
2.3.2 Критерий СЕ . . . . .	40
2.4 Предобучение сверточных слоев . . . . .	46
2.5 Метод редуцирования параметров . . . . .	49
2.6 Выводы . . . . .	51
ГЛАВА 3 Экспериментальное обоснование метода неконтролируемого предобучения . . . . .	53
3.1 Сжатие данных . . . . .	53
3.2 Распознавание образов . . . . .	55
3.2.1 Критерий оценки результатов классификации . . . . .	56
3.2.2 Описание выборок . . . . .	57
3.2.3 Параметры вычислительного эксперимента и результаты . . . . .	58
3.3 Редуцирование параметров НС . . . . .	60
3.4 Выводы . . . . .	63
ГЛАВА 4 Разработка систем компьютерного зрения на основе глубоких нейронных сетей . . . . .	64
4.1 Автоматический контроль качества нанесения маркировки продукции . . . . .	64
4.1.1 Постановка задачи и обзор существующих решений . . . . .	64
4.1.2 Предлагаемый подход . . . . .	68
4.1.3 Обучающие выборки: формирование и основные характеристики . . . . .	73
4.1.4 Результаты обучения и тестирования . . . . .	74

4.2	Обнаружение солнечных панелей на аэрофотоснимках . . . . .	75
4.2.1	Постановка задачи и обзор существующих решений	75
4.2.2	Предлагаемый подход . . . . .	77
4.2.3	Обучающая выборка . . . . .	79
4.2.4	Оценка качества обученных моделей . . . . .	80
4.2.5	Результаты обучения и тестирования . . . . .	82
4.3	Выводы . . . . .	84
	Заключение . . . . .	85
	Список использованных источников . . . . .	87
	Библиографический список . . . . .	87
	Список публикаций соискателя ученой степени . . . . .	93
	ПРИЛОЖЕНИЕ А Исходный код реализации процедур предобучения и редукции . . . . .	98
	ПРИЛОЖЕНИЕ Б Документы об использовании результатов диссертационной работы . . . . .	109

# ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

*GNC* – глубокая нейронная сеть

*ИНС* – искусственная нейронная сеть

*НС* – нейронная сеть

*СНС* – сверточная нейронная сеть

*AP* (англ. *Average Precision metric*) – метрика, используемая для оценки алгоритмов детекции объектов на изображениях, применяется для одноклассовых случаев

*CD* (англ. *Contrastive Divergence*) – контрастное расхождение

*CE* (англ. *Cross-Entropy error*) – кросс-энтропийная функция ошибок

*COCO* (англ. *Common Objects in Context*) – выборка изображений с разметкой, применяемая для решения задач обнаружения, сегментации и аннотирования объектов на изображениях

*C-RBM* (англ. *Classic Restricted Boltzmann Machine training* – классический метод обучения ограниченной машины Больцмана.

*DNN* (англ. *Deep Neural Network*) – глубокая нейронная сеть

*IoU* (англ. *Intersection over Union, мера Жаккара*) – метрика, применяемая для оценки эквивалентности двух множеств, в задачах компьютерного зрения применяется для оценки методов детекции (обнаружения) объектов как величина, описывающая степень перекрытия двух прямоугольных областей, используемых для задания эталонной и предсказываемой моделью локализации объекта.

*HREBA* (англ. *Hybrid Reconstruction Error-Based Approach*) – гибридный вариант предобучения ГНС, сочетающий в себе использование классического (C-RBM) и предлагаемого в данной работе (REBA) методов обучения ограниченной машины Больцмана.

*mAP* (англ. *mean-Average Precision metric*) – метрика, применяемая для оценки качества алгоритмов детекции объектов на изображениях, применяется для многоклассовых случаев детекции

*MNIST* (англ. *Mixed National Institute of Standards and Technology database*) – смешанная база Национального института стандартов и технологий

*MSE* (англ. *Mean Squared Error*) – среднеквадратичная ошибка

*Faster-RCNN* (англ. *Faster Region-based Convolutional Neural Network*) – архитектура глубокой сверточной нейронной сети, применяемая для решения задач детекции (обнаружения) объектов на изображениях

*PCA* (англ. *Principal Component Analysis*) – метод главных компонент

*RBM* (англ. *Restricted Boltzmann Machine*) – ограниченная машина Больцмана

*REBA* (англ. *Reconstruction Error-Based Approach*) – метод обучения ограниченной машины Больцмана, базирующийся на минимизации ошибки восстановления образов на видимом и скрытом слоях

*ReLU* (англ. *Rectified Linear Unit*) – исправленный линейный элемент

*ResNet-50* (англ. *Residual Neural Network*) – архитектура глубокой нейронной сети, которая характеризуется передачей сигналов между отдельными слоями модели, применяется в качестве классификатора

*SVM* (англ. *Support Vector Machines*) – метод опорных векторов

*SVHN* (англ. *The Street View House Numbers*) – выборка изображений номеров домов

# ВВЕДЕНИЕ

Практические приложения компьютерного зрения с каждым годом становятся все более разнообразными. В современном мире компьютерное зрение используется повсеместно – в сложных производственных и медицинских системах, в интеллектуальных системах интернета вещей и развлекательных приложениях для мобильных устройств.

В качестве основы при разработке таких систем все чаще находят применение глубокие нейросетевые модели. Данные модели показывают впечатляющие результаты при решении самых разнообразных задач компьютерного зрения – распознавания, детекции и сегментации объектов на фото- и видеоизображениях, получения аннотаций для фотографий и генерации изображений по текстовому описанию. Глубокие нейронные сети, применяемые для решения подобных задач, содержат миллионы настраиваемых параметров и, для некоторых архитектур, десятки слоев нейронных элементов.

Обучение подобных «тяжелых» моделей является нетривиальной задачей. Оно часто сопряжено с риском переобучения, результатом которого является отличная приспособленность сети к данным из обучающей выборки, но плохая обобщающая способность, то есть неэффективность модели для данных, не использовавшихся при обучении. Чаще всего переобученность возникает при применении малой обучающей выборки. Другой проблемой является ресурсоемкость процесса обучения таких моделей, даже при применении современных технических средств.

Проблемы обучения глубоких нейронных сетей активно изучаются в зарубежных научных школах. В нашей стране такие исследования также проводятся. Однако, нужно отметить, что такие исследования часто носят эмпирический характер, поэтому разработка строгих математически обоснованных методов обучения остается важной задачей теории нейронных сетей.

В диссертационной работе разработаны методы неконтролируемого предобучения глубоких нейронных сетей и редукции параметров моделей. Предложены алгоритмы для решения практических задач теории компьютерного зрения – обнаружения (детекции) и распознавания объектов на изображениях, контроля качества нанесения маркировки на единицы продукции на поточных производственных линиях. Предложенные методы и алгоритмы позволяют улучшить работу интеллектуальных систем, использующих полносвязные и сверточные нейросетевые модели.

# ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

## Связь работы с научными программами (проектами), темами

Тема диссертации соответствует приоритетному направлению научно-технической деятельности согласно пункту 1 перечня приоритетных направлений научной, научно-технической и инновационной деятельности на 2021-2025 годы (Указ Президента Республики Беларусь от 07 мая 2020 г. № 156).

Исследования по теме диссертационной работы проводились в рамках научных программ:

1. НИР МОРБ «Алгоритмы интеллектуального анализа и обработки больших объемов данных на основе нейронных сетей глубокого доверия» (ГБ 15/203, № госрегистрации 20150743),

2. ГПНИ «Информатика и космос, научное обеспечение безопасности и защиты от чрезвычайных ситуаций» по заданию «Нейросетевые методы обработки комплексной информации и принятия решений на основе интеллектуальных многоагентных систем» (№ госрегистрации 20140547),

3. ГПНИ «Информатика и космос, научное обеспечение безопасности и защиты от чрезвычайных ситуаций» по заданию «Методы и алгоритмы интеллектуальной обработки и анализа большого объема данных на основе нейронных сетей глубокого доверия» (задание 1.6.05, № госрегистрации 20163595),

4. НИР «Методы и алгоритмы построения интеллектуальных систем анализа и обработки данных», этап «Разработка гибридных интеллектуальных систем на основе нейросимволического подхода»,

5. НИР БРФФИ «Модели и исследование 3-Д оцифровки на основе фактических данных и анализа гетерогенных данных» (№ Ф22КИ-046 от 05.11.2021 г.).

## Цель, задачи, объект и предмет исследования

*Целью исследования является разработка эффективных методов и алгоритмов для обучения глубоких нейронных сетей, используемых для решения задач компьютерного зрения, включающих распознавание маркировки продукта на конвейерной линии и обнаружение солнечных панелей на аэрофотоснимках.*

Указанная цель определяет следующие *задачи исследования*:

1) Разработать метод неконтролируемого предобучения глубоких нейронных сетей, позволяющий повысить эффективность обучения моделей;

- 2) Разработать алгоритм редуцирования параметров глубоких нейронных сетей, позволяющий упростить структуру моделей;
- 3) Провести сравнительный анализ эффективности разработанных метода обучения и алгоритма редуцирования;
- 4) Разработать нейросетевые системы распознавания маркировки продукта на конвейерной линии и обнаружения солнечных панелей на аэрофотоснимках.

*Объектом исследования* являются нейросетевые системы компьютерного зрения. *Предметом исследования* выступают методы и алгоритмы обучения глубоких нейронных сетей и их применение к задачам компьютерного зрения.

### **Научная новизна**

Научная новизна состоит в выявлении и доказательстве эквивалентности максимизации функции правдоподобия распределения входных данных, минимизации кросс-энтропийной функции ошибки и суммарной квадратичной ошибки при использовании линейных нейронов в пространстве синаптических связей ограниченной машины Больцмана.

Разработан метод обучения ограниченной машины Больцмана на основе доказанных эквивалентностей, применение которого для предобучения ГНС позволяет расширить класс обучаемых моделей и повысить обобщающую способность ГНС.

Разработан алгоритм сокращения параметров нейросетевой модели, основывающийся на неконтролируемом предобучении сети, что позволяет уменьшить количество настраиваемых параметров полносвязных и сверточных нейронных сетей без потери обобщающей способности.

Разработаны нейросетевые системы компьютерного зрения (система распознавания маркировки на производственной линии, система обнаружения солнечных панелей на аэрофотоснимках), которые основываются на применении предложенного метода для выполнения предобучения соответствующих нейросетевых моделей.

### **Положения, выносимые на защиту**

1. Метод обучения ограниченной машины Больцмана, базирующийся на эквивалентности максимизации функции правдоподобия распределения входных данных в пространстве синаптических связей и минимизации суммарной квадратичной ошибки сети при использовании линейных нейронов, что позволяет расширить класс обучаемых моделей и повысить обобщающую способность глубоких нейронных сетей.

2. Алгоритм редуцирования параметров глубокой нейронной сети, базирующийся на неконтролируемом предобучении сети, что позволяет упростить ее архитектуру (путем сокращения числа настраиваемых параметров модели) без потери обобщающей способности.

3. Нейросетевые системы компьютерного зрения, базирующиеся на использовании предобученных моделей для решения задач классификации и детекции – система распознавания (в реальном режиме времени) маркировки продукта на производственной линии и система обнаружения солнечных панелей на аэрофотоснимках (включая снимки низкого разрешения) с точностью до 87,46%.

### **Личный вклад соискателя ученой степени в результаты диссертации**

Основные положения диссертации получены автором лично. Из публикаций, сделанных в соавторстве, в содержание диссертации включены только те результаты, которые были получены соискателем лично. Соавтором основных публикаций автора является научный руководитель д.т.н., профессор В.А. Головко, который осуществлял определение целей и постановку задач исследований, выбор методов исследований, принимал участие в планировании работ и обсуждении результатов.

### **Апробация диссертации и информация об использовании ее результатов**

Основные положения и результаты диссертационной работы докладывались и обсуждались на следующих международных конференциях: «International Conference on Neural Networks and Artificial Intelligence» (Брест, 2014); «Информационное, программное и техническое обеспечение систем управления организационно-технологическими комплексами» (Луцк, 2015); «8th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications» (Варшава, 2015); «International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T)» (2018); «International Conference on Pattern Recognition and Information Processing (PRIP)» (2019); «Вычислительные методы, модели и образовательные технологии» (Брест, 2013, 2014, 2015, 2016, 2019); «Современные проблемы математики и вычислительной техники» (Брест, 2015, 2019, 2021); «Открытые семантические технологии проектирования интеллектуальных систем» (Минск, 2018, 2019, 2020, 2021, 2022, 2023); «11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems:

Technology and Applications (IDAACS)» (Краков, 2021).

## **Опубликованность результатов диссертации**

По теме диссертационной работы опубликовано 28 печатных работ (**9,48** авторских листа). Из них 11 статей в научных журналах (5,35 авторских листа), 5 статей в сборниках материалов научных конференций, включенных в системы международного цитирования (Scopus, Web of Science и IEEE Xplore digital, 0,48 авторских листа) и 12 статей в сборниках материалов научных конференций (**3,65** авторских листа).

## **Структура и объем диссертации**

Диссертация состоит из введения, общей характеристики работы, четырех глав с краткими выводами по каждой главе, заключения, библиографического списка, списка публикаций автора и приложений.

В *первой главе* рассмотрено краткое введение в теорию искусственных нейронных сетей, дана классификация их типов. Определено понятие глубокой нейронной сети. Рассмотрены задачи обучения глубоких нейронных сетей и существующие методы обучения. *Вторая глава* посвящена рассмотрению разработанного метода обучения ограниченной машины Больцмана и метода предобучения глубоких нейронных сетей на его основе. Также в этой главе предлагается алгоритм редуцирования параметров нейросетевых моделей. В *третьей главе* приведены экспериментальные результаты, обосновывающие полученные теоретические результаты. В *четвертой главе* рассмотрено практическое применение разработанных методов в интеллектуальных системах компьютерного зрения.

Общий объем диссертации составляет 111 страниц, из которых 86 страниц основного текста, **43 рисунок на 33 страницах**, 14 таблиц на 8 страницах, библиография из 115 источников, включая 28 публикаций автора, приложения на 14 страницах.

# ГЛАВА 1

## ПРОБЛЕМЫ ОБУЧЕНИЯ ГЛУБОКИХ НЕЙРОННЫХ СЕТЕЙ

В данной главе вводится теоретическая основа для понимания основных концепций теории обучения глубоких нейронных сетей. Приводится классификация основных типов слоев и архитектур моделей. Описывается историческая ретроспектива развития теории обучения глубоких архитектур, а также основные проблемы, которые возникали и возникают при выполнении обучения.

Осуществляется обзор существующих подходов для обучения глубоких нейронных сетей. Приводится описание методов обучения с использованием стохастического градиентного спуска и активационных функций ReLU и ее вариантов. Описываются основные проблемы, присущие подобному типу обучения.

Особое внимание уделяется методам неконтролируемого предобучения, для реализации которых не требуется наличие обучающей выборки большого размера. Главной целью, достигаемой при выполнении предобучения, является формирование «хорошей» начальной инициализации параметров глубокой нейронной сети. Подобная инициализация гарантирует сходимость нейросетевой модели в процессе дальнейшего дообучения на этапе «тонкой настройки».

Также приводятся алгоритмы предобучения с использованием RBM и автоэнкодерных моделей, формируемых из слоев ГНС и описывается этап «тонкой настройки» нейронной сети, цель которого состоит в дообучении модели для решения определенной задачи.

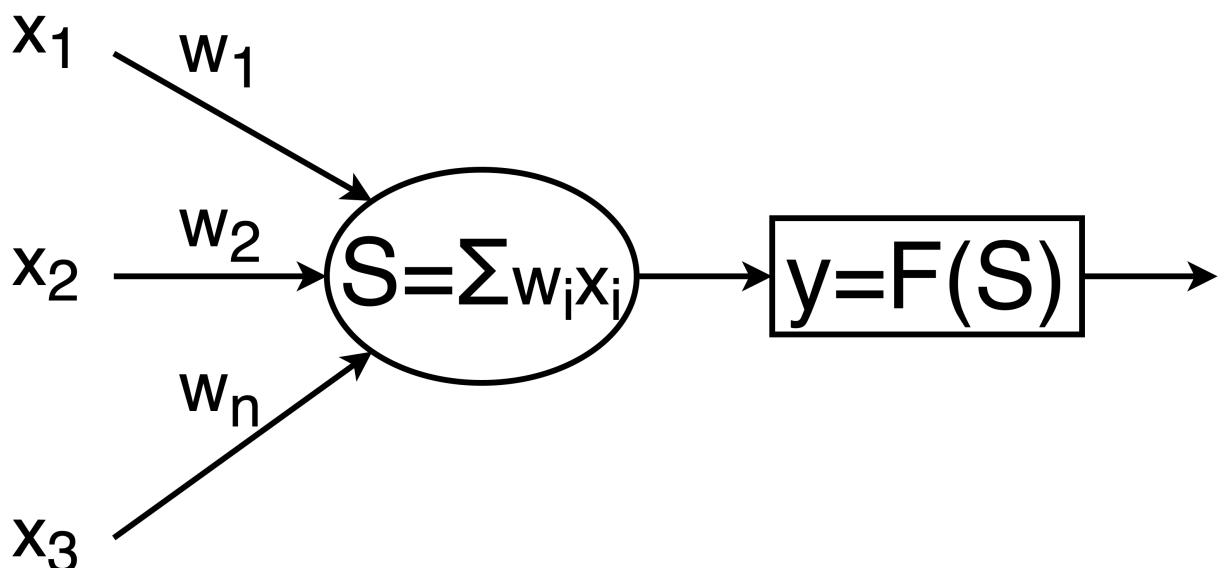
### 1.1 Общие сведения

Вопросы разработки и применения нейросетевых моделей часто рассматриваются в контексте более широкой области машинного обучения, являющейся по сути одним из направлений современной науки об искусственном интеллекте. При этом само машинное обучение сформировалось на стыке нескольких научных дисциплин, включающих вычислительную математику, теорию вероятности и статистику, линейную алгебру и теорию оптимизации. С помощью алгоритмов машинного обучения создаются модели, которые используют предшествующий опыт для повышения эф-

фективности компьютера в решении определенных классов задач [1]. Под понятием предшествующего опыта здесь понимается совокупность статистических данных, собранных при наблюдении и описании некоторого процесса.

Являясь моделями машинного обучения, нейронные сети также способны к решению различных типов задач после реализации определенного алгоритма настройки параметров (обучения). Само свойство обучаемости приближает искусственные нейросети к их естественному прототипу, а именно к человеческому мозгу, где обучение является важным процессом в формировании психики [2].

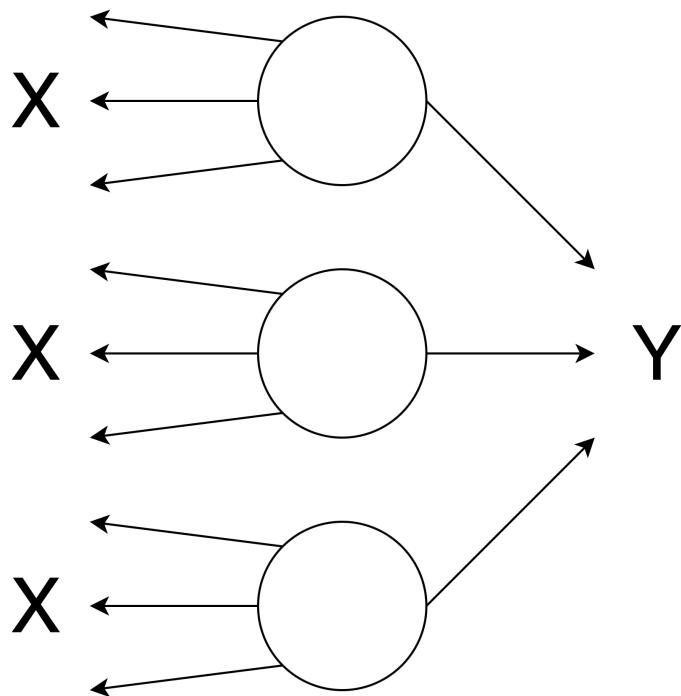
В основе теории искусственных нейронных сетей лежит биологически инспирированная модель искусственного нейрона, представляющая собой вычислительный элемент, выполняющий нелинейное преобразование входных данных (рис. 1.1). Данное преобразование заключается в вычислении некоторой функции активации (линейной или нелинейной)  $F$  от взвешенной суммы  $S$  компонент вектора входных данных  $X$ . Параметры  $W$  называются весовыми коэффициентами нейрона и настраиваются в процессе обучения.



**Рисунок 1.1. – Модель искусственного нейрона**

Данные простейшие вычислительные элементы объединяют в группы, называемые слоями, по аналогии с биологическими нейронными слоями. Каждый нейрон слоя, обладая независимым набором весовых коэффициентов, получает на вход одинаковый набор данных  $X$  и выполняет его преобразование, формируя отдельную компоненту выходного вектора  $Y$  или паттерна выходной активности слоя 1.2. Весовые коэффициенты слоя

для удобства объединяются в матрицы, которые называются матрицами весовых коэффициентов слоя.



**Рисунок 1.2. – Слой нейронных элементов**

Наконец, слои нейронных элементов объединяются в последовательности, формируя общую архитектуру нейронной сети (последовательность слоев, их тип, количество нейронов в каждом слое, функцию активации и т.д.). Итоговая модель способна выполнять аппроксимацию любой непрерывной функции с любой точностью [3] после выполнения обучения. В процессе этого данные из некоторой выборки подаются на вход сети с целью определения градиентов изменения параметров нейронных элементов. Данная процедура выполняется итеративно до тех пор, пока не будет достигнут ответ сети, полученный с желаемой точностью или пока весовые коэффициенты не войдут в некоторое стационарное состояние.

Искусственные нейронные сети могут быть классифицированы по различным признакам (например, по характеру обучения, по архитектуре и т.д.). В литературе чаще всего используется классификация по архитектурному признаку. Согласно ей, выделяют [4]:

- 1) Сети персептронного типа (многослойные персептроны [5], рекуррентные ([6], [7]), рециркуляционные [8], сверточные ([9], [10])), глубокие);
- 2) Самоорганизующиеся НС (НС Кохонена [11], сети адаптивного резонанса [12]);
- 3) Релаксационные НС (сети Хопфилда [13], Хэмминга [14] и двунаправленная ассоциативная память [15]);

- 4) Гибридные НС (сети встречного распространения [16], RBF-сети [17], нечеткие сети [18]);
- 5) Нейронные иммунные сети [19].

В данной работе объектом исследования являются модели первой группы, а именно, глубокие нейронные сети, которые фактически являются развитием полносвязных сетей (персепtronов) и сверточных нейронных сетей.

В настоящий момент сверточные нейронные сети и глубокие архитектуры на их основе являются доминирующими в области компьютерного зрения [20]. В основном это связано с тем, что в подобных задачах используются данные с высокой корреляцией между соседними элементами (в видео и фото в качестве этих значений выступают массивы пикселей) и сверточные слои особенно эффективны при обработке именно таких данных [21].

Сверточные модели могут включать как чисто сверточные слои, так и некоторое количество полносвязных слоев (например, для моделей классификации, где последние слои выполняют роль классификаторов признаков, полученных на первых сверточных). При этом подобные модели также называются сверточными.

Несмотря на лидирующее положение сверточных сетей, чисто полносвязные архитектуры также используются для некоторых задач компьютерного зрения (например, при решении задачи трехмерной реконструкции [22]). Далее в работе будет показано, что сверточный слой может быть представлен с помощью разряженного полносвязного слоя. Таким образом, исследование и обучение полносвязных архитектур остается актуальной задачей.

Строгое определения понятия глубокой нейронной сети не существует, но ее можно определить как искусственную нейронную сеть с архитектурой персептронного типа, имеющую более двух скрытых слоев нейронных элементов.

В общем случае глубокая нейронная сеть (deep neural network – DNN) представляет собой нейросетевую модель с множеством слоев нейронных элементов. Рассмотрим вариант полносвязной нейросетевой модели – персептрана (рисунок 1.3). Известно, что подобная сеть осуществляет глубокое иерархическое преобразование входного пространства образов [23].

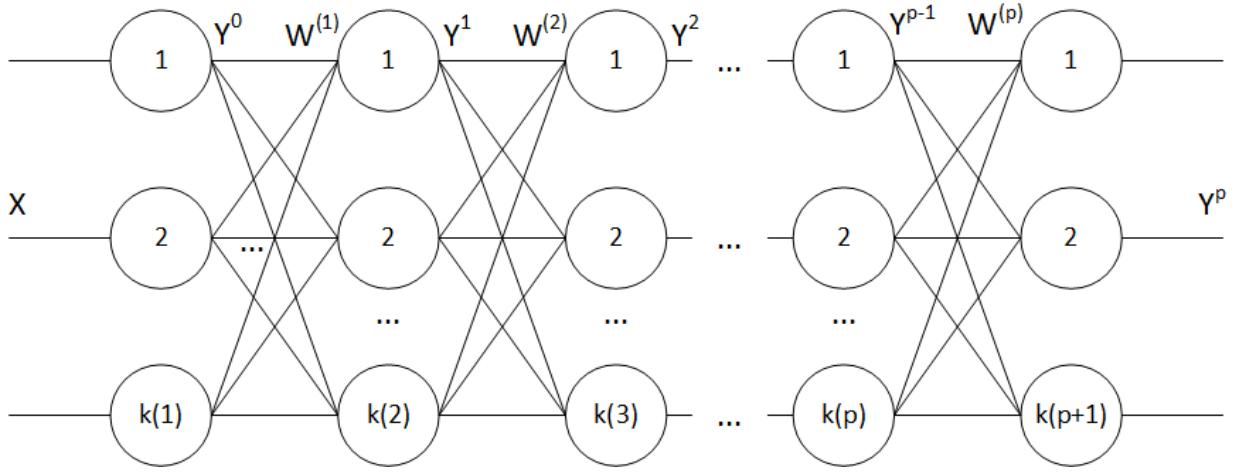


Рисунок 1.3. – Глубокая нейронная сеть

Первый скрытый слой выделяет низкоуровневое пространство признаков входных данных, второй слой определяет пространство признаков более высокого уровня абстракции и т.д. [24].

Выходное значение  $j$ -го нейрона  $k$ -го слоя определяется следующим образом:

$$y_j^k = F(S_j^k), \quad (1.1)$$

$$S_j^k = \sum_{i=1} w_{ij}^k y_i^{k-1} + T_j^k, \quad (1.2)$$

где  $F(\cdot)$  – функция активации нейронного элемента  $k$ -го слоя,  $S_j^k$  – взвешенная сумма  $j$ -го нейрона  $k$ -слоя,  $w_{ij}^k$  – весовой коэффициент между  $i$ -ым нейроном  $(k-1)$ -го слоя и  $j$ -м нейроном  $k$ -го слоя,  $T_j^k$  – пороговое значение  $j$ -го нейрона  $k$ -го слоя.

Для первого (распределительного) слоя

$$y_j^0 = x_j. \quad (1.3)$$

В матричном виде выходной вектор  $k$ -го слоя

$$Y^k = F(S^k) = F((Y^{k-1})^T W^k + T^k), \quad (1.4)$$

где  $W^k$  – матрица весовых коэффициентов  $k$ -го слоя,  $Y^{k-1}$  – выходной вектор-столбец  $(k-1)$ -го слоя,  $T^k$  – вектор пороговых значений нейронов  $k$ -го слоя.

Если глубокая нейронная сеть используется для классификации образов, то выходные значения сети часто определяются на основе функции активации **softmax** [4]:

$$y_j^F = \text{softmax}(S_j) = \frac{e^{S_j}}{\sum_i e^{S_i}} \quad (1.5)$$

В этом случае значения, получаемые на последнем слое нейронной сети, отражают вероятность принадлежности данного образа к определенному классу.

## 1.2 Проблемы обучения глубоких моделей

Интенсивность исследований в области нейронных сетей никогда не выражалась однородным течением. В основном причинами приостановок в исследованиях этих моделей являлось отсутствие доступных на определенный момент времени эффективных методов их обучения.

Начиная с первых моделей ИНС, описанных в работе У. Мак-Каллока и У. Питтса [25] в 1943 году, искусственным нейронным сетям были посвящены концептуальные и важные работы того времени (например, [26], [27]). В 1959 году Ф. Розенблatt предложил модель персептрона [28]. Однако, после опубликования монографии М. Минского и С. Пайперта «Персептроны» в 1969 году [29] крупные исследования в этой области были приостановлены. В этой работе был проведен анализ персептронной модели и выявлены ее основные ограничения, в том числе касающиеся невозможности эффективного решения существующими на тот момент моделями некоторых задач (например, задач инвариантного представления образов).

В конце 1970-х годов произошел очередной всплеск работ, посвященных нейросетевым моделям (например, [30], [31]). В 1986 году Д. Румельхартом, Дж. Хинтоном и Р. Вильямсом в работе [32] был предложен метод обратного распространения ошибки для обучения многослойных моделей. Результаты, изложенные в данной работе, открыли теоретические возможности для обучения моделей с большим количеством слоев. Однако, до середины 2000-х годов сети с количеством слоев три и более широко не применялись, так как считалось, что использование глубоких моделей не дает существенного прироста в эффективности по сравнению с другими подходами. Неэффективность была связана с проблемой «исчезающего» градиента. Она проявляется в том, что при обучении глубокой нейронной сети методом обратного распространения ошибки значения градиентов весовых коэффициентов первых слоев сети быстро становятся близкими к нулю, в результате весовые коэффициенты таких слоев практически не изменяются [23]. Это существенно замедляет процесс обучения, делая невозможным практическое применение подобных моделей по сравнению с другими су-

ществовавшими на тот момент времени методами (например, [33]).

Тем не менее, проводя аналогии с многоуровневыми нейронными архитектурами в человеческом мозге (в частности, строением центральной зрительной системы), было понятно, что подобный тип искусственных нейронных сетей обладает полезными свойствами, позволяя формировать многоуровневую иерархию признаков [34].

В 2006 году Джоном Хинтоном был предложен подход к обучению глубоких моделей, который ознаменовал начало новой эпохи в развитии теоретических работ в данной области [35].

В предложенном методе использовался «жадный» алгоритм послойного предобучения глубоких нейронных сетей (greedy layer-wise algorithm), основанный на применении ограниченных машин Больцмана, последовательно формируемых и обучаемых из слоев глубокой нейронной сети.

С этой работы фактически начался новый подъем в исследованиях нейронных сетей, который длится до настоящего времени. За последние годы в области компьютерного зрения благодаря применению глубоких нейронных сетей удалось значительно улучшить результаты для различных задач, включая классификацию, детекцию, сегментацию объектов на изображении, генерацию текстовых описаний к изображениям, генерацию изображений по текстовым описаниям и т.д.

В течение следующих лет сместился основной акцент в исследованиях – вместо послойного обучения стали применяться методы, позволяющие обучать глубокие нейронные сети, начиная с произвольной начальной инициализации, без использования предобучения (например, стохастический градиентный спуск с использованием функции активации ReLU и ее вариантов). Однако, применение таких подходов не снижает требований к объему обучающей выборки, которая должна быть сравнима с числом настраиваемых параметров модели, поскольку большие модели, обучаемые на малых выборках, с большей вероятностью будут переобучаться. Результатом этого является отличная приспособленность модели к обучающей выборке, но плохая обобщающая способность, т.е. неэффективность сети на тестовых данных. Таким образом, объем используемой обучающей выборки для глубоких нейронных сетей остается по-прежнему критичным фактором для обучения. Даже наличие подходящей по объему выборки не гарантирует успешное завершение обучения, так как подготовка больших моделей может повлечь неприемлемые аппаратные издержки ([36], [37], [38]).

### 1.3 Методы обучения

Рассмотрим основные методы, применяемые для обучения глубоких нейронных сетей.

Существуют два основных подхода к обучению глубоких нейронных сетей, оба из которых используют предобучение:

1) Обучение с использованием неконтролируемого предобучения (I тип);

2) Обучение с использованием предобучения на большой обучающей выборке и любого оптимизирующего метода для настройки параметров нейронной сети (II тип).

Дадим краткое описание предобучения II типа, а затем рассмотрим более подробно I тип.

Предобучение II типа представляет собой обучение модели с использованием метода обратного распространения ошибки [39], некоторого оптимизирующего метода и активационных функций ReLU и ее вариантов в качестве функций активации нейронов [20], а затем дообучение полученной модели для новой выборки (*transfer learning*).

Среди основных оптимизирующих методов, применяемых для предобучения II типа, выделяют:

1) SGD (стохастический градиентный спуск). В данном методе корректировка настраиваемых параметров и.н.с. выполняется в направлении максимального уменьшения функции стоимости, то есть в направлении, противоположном вектору градиента функции потерь [40];

2) Метод Нестерова. Обучение методом стохастического градиентного спуска не редко происходит очень медленно. Импульсный метод позволяет ускорить обучение, особенно в условиях высокой кривизны, небольших, но устойчивых градиентов или зашумленных градиентов. В импульсном методе вычисляется экспоненциально затухающее скользящее среднее прошлых градиентов и продолжается движение в этом направлении. Метод Нестерова является вариантом импульсного алгоритма, в котором градиент вычисляется после применения текущей скорости [41];

3) AdaGrad: данный метод по отдельности адаптирует скорости обучения всех настраиваемых параметров и.н.с., умножая их на коэффициент, обратно пропорциональный квадратному корню из суммы всех прошлых значений квадрата градиента [42];

4) RMSProp. Данный метод является модификацией AdaGrad, которая позволяет улучшить его поведение в невыпуклом случае путем изменения способа агрегирования градиента на экспоненциально взвешенное

скользящее среднее. Использование экспоненциально взвешенного скользящего среднего гарантирует повышение скорости сходимости после обнаружения выпуклой впадины, как если бы внутри этой впадины алгоритм AdaGrad был инициализирован заново [41];

5) Adam. Данный метод можно рассматривать как комбинацию RMSProp и AdaGrad [43]. Помимо усредненного первого момента, данный метод использует усредненное значение вторых моментов градиентов.

II тип фактически представляет собой обучение с нуля, который осуществляется на больших выборках данных с использованием подходящих аппаратных средств. Предобученные таким образом модели могут быть использованы для решения других задач. Таким образом, современный инженер получает в свое распоряжение репозиторий готовых предобученных моделей, которые могут быть дообучены на небольших индивидуальных выборках данных.

Рассмотрим подробно реализацию предобучения I типа.

Исследование методов предобучения I типа остается актуальной и перспективной задачей, поскольку позволяет существенно сократить объем применяемой обучающей выборки (для малых обучающих выборок позволяет преодолеть переобучение [20]) и жесткие требования к используемой аппаратной части.

В общем случае процесс обучения глубоких нейронных сетей с послойным предобучением (предобучением I типа) можно описать последовательностью из двух этапов [35, 44, 24, 45]:

- 1) Предобучение нейронной сети методом послойного обучения, начиная с первого слоя (pre-training). Такое обучение осуществляется неконтролируемо;
- 2) Настройка синаптических связей всей сети (fine-tuning) при помощи алгоритма обратного распространения ошибки или алгоритма «бодрствования и сна» (wake-sleep algorithm) [46].

**Первый этап** является «ядром» всего подхода. Выделяют два основных подхода к реализации первого этапа (рисунок 1.4). В обоих этих методах формирование вспомогательных архитектур (RBM, автоэнкодеров) осуществляется из структуры исходной сети.

## Подходы к предобучению DNN

RBM

Autoencoder

Рисунок 1.4. – Методы предварительного обучения глубоких нейронных сетей

Первый подход основывается на представлении каждого слоя нейронной сети в виде ограниченной машины Больцмана (RBM). При использовании второго, автоэнкодерного, подхода, каждый слой представляется автоассоциативной нейронной сетью.

Рассмотрим каждый из этих подходов подробнее.

При предобучении ГНС в соответствии с **первым подходом** осуществляется последовательное конструирование RBM-сетей из параметров скрытых слоев ГНС и их обучение.

Для понимания подхода рассмотрим модель RBM.

Ограниченная машина Больцмана состоит из двух слоев стохастических бинарных нейронных элементов, которые соединены между собой двунаправленными симметричными связями (рисунок 1.5). Входной слой нейронных элементов называется видимым (слой  $X$ ), а выходной – скрытым (слой  $Y$ ). Ограниченная машина Больцмана может генерировать любое дискретное распределение, если используется достаточное количество нейронов скрытого слоя [23].

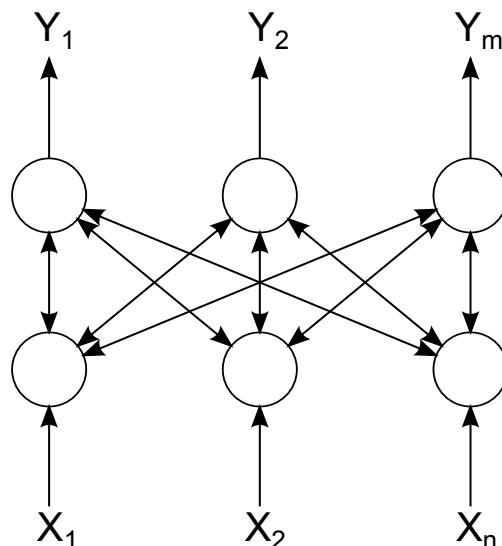


Рисунок 1.5. – Ограниченнная машина Больцмана

В данной модели состояния видимых и скрытых нейронов меняются

в соответствии с вероятностной версией сигмоидной функции активации:

$$p(y_j|x) = \frac{1}{1 + e^{-S_j}}, \quad S_j = \sum_i^n w_{ij}x_i + T_j \quad (1.6)$$

$$p(x_i|y) = \frac{1}{1 + e^{-S_i}}, \quad S_i = \sum_j^m w_{ij}y_j + T_i \quad (1.7)$$

где  $S_i, S_j$  – взвешенные суммы видимого и скрытого слоев,  $T_i, T_j$  – пороговые элементы видимого и скрытого слоев соответственно.

Состояния видимых и скрытых нейронных элементов принимаются независимыми:

$$P(x|y) = \prod_{i=1}^n P(x_i|y)$$

$$P(y|x) = \prod_{j=1}^m P(y_j|x)$$

Таким образом, состояния всех нейронных элементов ограниченной машины Больцмана определяются через распределение вероятностей. В RBM нейроны скрытого слоя являются детекторами признаков, которые сохраняют закономерности входных данных. Основная задача обучения состоит в воспроизведении распределения входных данных на основе состояний нейронов скрытого слоя как можно точнее. Для достижения этого применяется процедура CD- $k$ . В главе 2 будет подробно описана данная процедура и приведен вывод классических правил обучения RBM.

При предобучении ГНС на базе RBM вначале создается RBM-модель из первого скрытого слоя ГНС. Для данной RBM входные данные поступают на видимый слой нейронных элементов и используя CD- $k$  процедуру вычисляются состояния скрытых  $P(y|x)$  и видимых нейронов  $P(x|y)$ . В процессе выполнения данной процедуры (не более 100 эпох) изменяются весовые коэффициенты и пороговые значения сети RBM, которые затем фиксируются. Следующим берется второй скрытый слой глубокой нейронной сети и конструируется новая RBM. Входными данными для нее являются данные с предыдущего слоя (выходные данные, полученные первой RBM). Выполняется обучение новой модели и процесс продолжается для всех последующих слоев нейронной сети, за исключением, возможно, самого последнего (включение последнего слоя в процесс предобучения зависит от решаемой задачи – например, для задачи классификации предобучение

последнего слоя не требуется, так как он должен обучаться с учителем), как показано на рисунке 1.6.

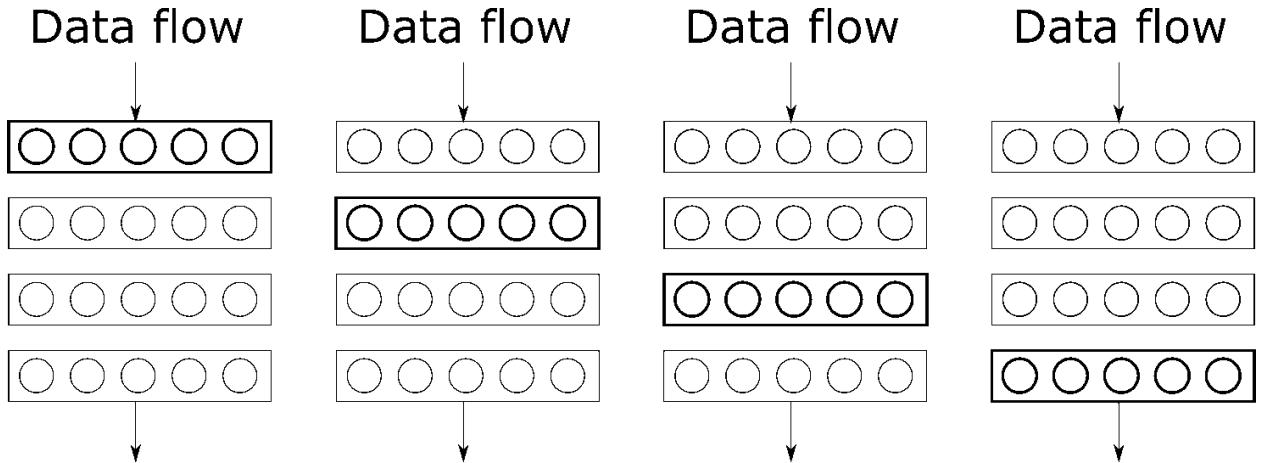


Рисунок 1.6. – «Жадный» алгоритм послойного обучения

При использовании **второго подхода** [47], вначале обучается первый слой ГНС как автоассоциативная нейронная сеть с целью минимизации суммарной ошибки реконструкции данных, затем обучается второй слой ГНС и так далее. Для обучения каждого слоя можно использовать алгоритм обратного распространения ошибки.

Рассмотрим персепtron с тремя скрытыми слоями (рисунок 1.7). Тогда в соответствии с автоэнкодерным методом, прежде всего, берутся первые два слоя нейронной сети (1 и 2) и на базе их конструируется автоассоциативная (рециркуляционная) нейронная сеть (1-2-1), то есть добавляется восстановливающий слой (рисунок 1.8).

Затем, происходит обучение такой сети, например, при помощи алгоритма обратного распространения ошибки с целью минимизации ошибки реконструкции данных. Продолжительность обучения обычно не превышает 100 эпох.

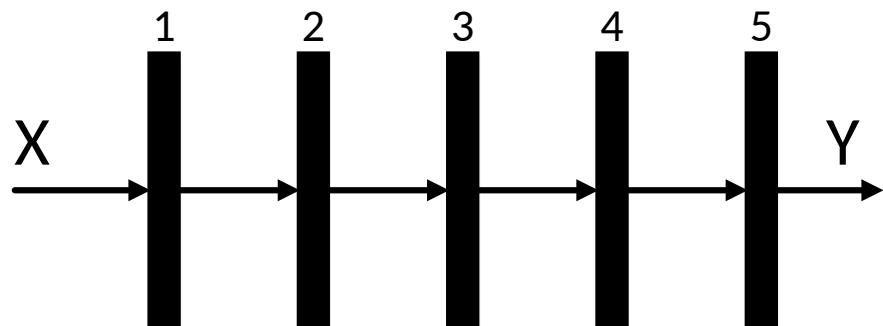
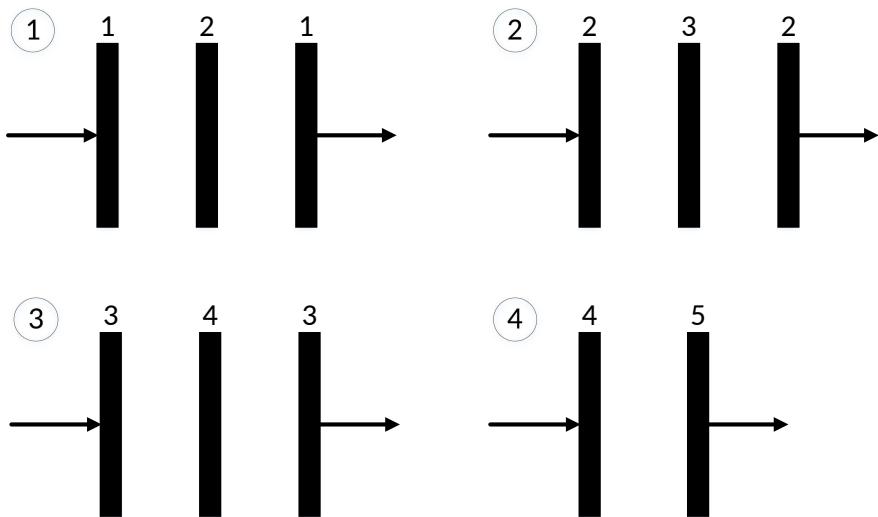


Рисунок 1.7. – Персепtron с тремя скрытыми слоями



**Рисунок 1.8. – Автоэнкодерный метод обучения**

После этого отбрасывается восстанавливающий слой (последний слой) автоассоциативной сети, фиксируются веса скрытого слоя, и конструируется автоассоциативная сеть из следующих двух слоев нейронной сети (2-3-2), которая обучается на основе данных, поступающих с предыдущего (2-го слоя). Процесс продолжается до последнего или предпоследнего слоя, как это схематично изображено на рисунке 1.8.

Данный процесс можно представить в виде следующего алгоритма:

- 1) Конструируется автоассоциативная сеть с входным слоем  $X$ , скрытым  $Y$  и выходным слоем  $X$ .
- 2) Обучается автоассоциативная сеть, например при помощи алгоритма обратного распространения ошибки (как правило не более 100 эпох) и фиксируются синаптические связи первого слоя  $W_1$ .
- 3) Берется следующий слой глубокой нейронной сети и снова формируется автоассоциативная сеть аналогичным образом.
- 4) Используя настроенные синаптические связи предыдущего слоя  $W_1$ , подаются входные данные на вторую автоассоциативную сеть и она обучается аналогичным образом. В результате получаются весовые коэффициенты второго слоя  $W_2$ .
- 5) Процесс продолжается до последнего или предпоследнего слоя нейронной сети.

Для обоих подходов (RBM и автоэнкодерный), с помощью реализации такого неконтролируемого предобучения можно получить приемлемую начальную инициализацию настраиваемых параметров глубокой нейронной сети.

На **втором этапе**, иначе называемом этапом «тонкой настройки» нейронной сети или fine-tuning, выполняется дообучение всей ГНС неко-

торым традиционным подходом (например, алгоритмом обратного распространения ошибки или «бодрствования и сна»).

Критически важным для этого этапа является начальная инициализация весов и порогов. Если такая инициализация выполнена в ходе предобучения, то сеть начинает процесс «тонкой настройки» с «хороших» начальных значений, что обуславливает быструю сходимость.

Если же предобучение отсутствует и используется традиционная «поверхностная» схема обучения со случайной инициализацией параметров, обучение сети может быстро остановиться и, как результат, приемлемая обобщающая способность не будет достигнута.

В дальнейшем под предобучением будем понимать предобучение I типа, проводимое с использованием RBM-сетей, если не указано иного.

#### 1.4 Критерии минимизации при обучении

При обучении ГНС (например, на этапе «тонкой настройки», если сеть предобучалась) важно определить критерий минимизации. Наиболее известными и используемыми в настоящий момент функциями ошибок являются функция среднеквадратичной ошибки (MSE):

$$E_{MSE} = \frac{1}{2} \sum_{j=1}^n (y_j - t_j)^2 \quad (1.8)$$

и кросс-энтропийная функция (CE) – случай мультиклассовой классификации:

$$E_{CE_{mult}} = - \sum_{j=1}^n t_j \log(y_j) \quad (1.9)$$

где  $n$  – количество нейронных элементов в последнем слое,  $y_j$  – выход нейрона  $j$ ,  $t_j$  – эталонное значение этого нейрона, множитель  $\frac{1}{2}$  введен для удобства вычислений.

Формула 1.8 справедлива для одного обучающего примера. Для  $L$  обучающих примеров она запишется в следующем виде:

$$E_{MSE} = \frac{1}{2L} \sum_{i=1}^L \sum_{j=1}^n (y_j^i - t_j^i)^2 \quad (1.10)$$

При обучении автоэнкодеров вместо формулы 1.9 для мультиклассового случая применяется формула, обобщающая двухклассовый случай ([48]):

$$E_{CE} = - \sum_{j=1}^n (t_j \log(y_j) + (1 - t_j) \log(1 - y_j)) \quad (1.11)$$

В соответствии с методом обратного распространения ошибки, для настройки весовых коэффициентов и порогов применяются следующие правила:

$$w_{i-1,i} = w_{i-1,i} - \alpha \frac{\partial E}{\partial w_{i-1,i}} \quad (1.12)$$

$$T_i = T_i - \alpha \frac{\partial E}{\partial T_i} \quad (1.13)$$

где  $w_{i-1,i}$  – весовые коэффициенты для  $i$ -го обрабатывающего слоя нейронной сети,  $i$  изменяется от 1 до  $N$ ,  $N$  – количество обрабатывающих слоев нейронной сети,  $\alpha$  – скорость обучения, а соответствующие градиенты находятся по формулам:

$$\frac{\partial E}{\partial w_{i-1,i}} = \frac{\partial E}{\partial S_i} y_{i-1} \quad (1.14)$$

$$\frac{\partial E}{\partial T_i} = \frac{\partial E}{\partial S_i} \quad (1.15)$$

где  $\frac{\partial E}{\partial S_i}$  – частная производная по взвешенной сумме для  $i$ -го слоя НС.

Частные производные по взвешенным суммам  $S_i$  получаются в соответствии со следующими выражениями:

$$\frac{\partial E}{\partial S_i} = (y_i - t_i) F'(S_i) \quad (1.16)$$

$$\frac{\partial E}{\partial S_{i-1}} = \left( \sum_i \frac{\partial E}{\partial S_i} w_{i-1,i} \right) F'(S_{i-1}) \quad (1.17)$$

где  $F$  – функция активации нейронных элементов.

Формула 1.16 используется для последнего слоя НС, а формула 1.17 – для других слоев, кроме последнего.

Формула 1.16 актуальна для случая, если используется целевая функция  $E_{MSE}$ . Для случая  $E_{CE}$  она будет иметь следующий вид:

$$\frac{\partial E}{\partial S_i} = \frac{y_i - t_i}{y_i(1 - y_i)} F'(S_i) \quad (1.18)$$

В случае использования логистической функции, формула 1.18 упрощается:

$$\frac{\partial E}{\partial S_i} = y_i - t_i \quad (1.19)$$

Формулы 1.14, 1.15 и 1.17 справедливы как для случая  $E_{MSE}$ , так и для  $E_{CE}$ .

Для  $E_{CE_{mult}}$  эти формулы справедливы для функции активации **softmax** на последнем слое нейронной сети. Покажем это.

Имеем

$$\frac{\partial E_{CE_{mult}}}{\partial S_i} = \sum_{k=1}^n \frac{\partial E_{CE_{mult}}}{\partial y_k} \frac{\partial y_k}{\partial S_i} = \frac{\partial E_{CE_{mult}}}{\partial y_i} \frac{\partial y_i}{\partial S_i} + \sum_{k \neq i} \frac{\partial E_{CE_{mult}}}{\partial y_k} \frac{\partial y_k}{\partial S_i} \quad (1.20)$$

Очевидно,

$$\frac{\partial E_{CE_{mult}}}{\partial y_i} = -\frac{t_i}{y_i} \quad (1.21)$$

Учитывая, что

$$y_i = \frac{e^{S_i}}{\sum_{k=1}^n e^{S_k}} \quad (1.22)$$

где  $i$  – индекс последнего слоя сети,  $S_i$  – соответствующая взвешенная сумма, найдем значения частной производной  $\frac{\partial y_i}{\partial S_k}$ .

1) Случай  $i = k$

$$\begin{aligned} \frac{\partial y_i}{\partial S_i} &= \frac{e^{S_i} \sum e^{S_k} - e^{2S_i}}{(\sum e^{S_k})^2} = \\ &= \frac{e^{S_i} \sum e^{S_k}}{(\sum e^{S_k})^2} - \frac{e^{2S_i}}{(\sum e^{S_k})^2} = y_i - y_i^2 = y_i(1 - y_i) \end{aligned} \quad (1.23)$$

2)  $i \neq k$

$$\frac{\partial y_k}{\partial S_i} = -e^{S_k} \left( \sum e^{S_i} \right)^{-2} e^{S_i} = -y_k y_i \quad (1.24)$$

Подставляя полученные выражения 1.23, 1.24 и 1.21 в 1.20, получим

$$\frac{\partial E_{CE_{mult}}}{\partial S_i} = -t_i(1 - y_i) + \sum_{k \neq i} t_k y_i = -t_i + y_i \sum_{k=1}^n t_k = y_i - t_i \quad (1.25)$$

## 1.5 Выводы

1) Дано описание основных понятий теории искусственных нейронных сетей, приведена классификация моделей ИНС.

2) Сформулирована проблема обучения глубоких нейронных сетей, обоснована ее актуальность. Приведены предпосылки развития методов обучения глубоких нейронных сетей.

3) Рассмотрены существующие методы обучения глубоких нейронных сетей.

4) Описаны основные этапы обучения ГНС с использованием неконтролируемого предобучения, включающие в себя предобучение ГНС и «тонкую настройку» модели.

5) Описаны основные подходы к осуществлению неконтролируемого предобучения ГНС, базирующиеся на использовании ограниченных машин Больцмана и автоассоциативных нейронных сетей.

# ГЛАВА 2

## МЕТОД НЕКОНТРОЛИРУЕМОГО ПРЕДОБУЧЕНИЯ

В данной главе в контексте проблемы обучения ГНС приводится вывод классических правил обучения ограниченной машины Больцмана (restricted Boltzmann Machine), применяемой на этапе неконтролируемого предобучения ГНС.

Предложен альтернативный подход к обучению ограниченной машины Больцмана ([1–A, 4–A, 5–A]), базирующийся на минимизации ошибок восстановления образов на его видимом и скрытом слоях, используя итерации сэмплирования Гиббса. В качестве критериев минимизации используются MSE (mean squared error – среднеквадратичная ошибка) и CE (cross-entropy loss – кросс-энтропийная функция потерь).

Для предлагаемого подхода рассмотрен вывод правил обучения RBM. Показано, что полученные правила обобщают классические правила обучения, полученные в работах Дж. Хинтона.

Предлагается подход к редуцированию размерности нейросетевых моделей за счет уменьшения количества настраиваемых параметров и количества используемых нейронов после выполнения процедуры предобучения с использованием сетей RBM.

Для основных случаев приводятся правила обучения CRBM, которые могут быть использованы для предобучения глубоких сверточных нейронных сетей. Также показано, что сверточный слой может быть представлен в виде полносвязного слоя.

### 2.1 Обучение RBM

В главе 1 было описано, что одним из методов обучения ГНС является метод предобучения, основывающийся на использовании ограниченных машин Больцмана.

Также, ранее было упомянуто, что основная задача обучения ограниченной машины Больцмана состоит в воспроизведении распределения входных данных на основе состояний нейронов скрытого слоя как можно точнее. Это эквивалентно максимизации функции правдоподобия путем модификации синаптических связей нейронной сети. Покажем основные этапы вывода правил обучения для RBM.

Вероятность нахождения видимого и скрытого нейрона в состоянии  $(x, y)$  определяется на основе распределения Гиббса [49]:

$$P(x, y) = \frac{e^{-E(x,y)}}{Z}$$

где  $E(x, y)$  – энергия системы в состоянии  $(x, y)$ ,  $Z$  – параметр, который определяет условие нормализации вероятностей, то есть, условие, при котором сумма вероятностей  $P(x, y)$  равняется единице. Данный параметр определяется следующим образом:

$$Z = \sum_{x,y} e^{-E(x,y)}$$

Вероятность нахождения видимых нейронов в определенном состоянии равняется сумме вероятностей конфигураций  $P(x, y)$  по состояниям скрытых нейронов:

$$P(x) = \sum_y P(x, y) = \sum_y \frac{e^{-E(x,y)}}{Z} = \frac{\sum_y e^{-E(x,y)}}{\sum_{x,y} e^{-E(x,y)}}$$

Для нахождения правил модификации параметров модели необходимо максимизировать вероятность воспроизведения состояний видимых нейронов  $P(x)$  ограниченной машиной Больцмана. Для того, чтобы определить максимум функции правдоподобия распределения данных  $P(x)$  будем использовать метод градиентного подъема в пространстве весовых коэффициентов и пороговых значений сети, где в качестве целевой функции используем функцию логарифмического правдоподобия:

$$\ln P(x) = \ln \sum_y e^{-E(x,y)} - \ln \sum_{x,y} e^{-E(x,y)}$$

Тогда градиент равен

$$\frac{\partial \ln P(x)}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \ln \sum_y e^{-E(x,y)} - \frac{\partial}{\partial w_{ij}} \ln \sum_{x,y} e^{-E(x,y)}$$

Преобразуя последнее выражение, получим

$$\frac{\partial \ln P(x)}{\partial w_{ij}} = -\frac{1}{\sum_y e^{-E(x,y)}} \sum_y e^{-E(x,y)} \frac{\partial E(x, y)}{\partial w_{ij}} + \frac{1}{\sum_{x,y} e^{-E(x,y)}} \sum_{x,y} e^{-E(x,y)} \frac{\partial E(x, y)}{\partial w_{ij}}$$

Так как

$$P(x, y) = P(y|x)P(x)$$

то

$$P(y|x) = \frac{P(x, y)}{P(x)} = \frac{\frac{1}{Z} e^{-E(x, y)}}{\frac{1}{Z} \sum_y e^{-E(x, y)}} = \frac{e^{-E(x, y)}}{\sum_y e^{-E(x, y)}}$$

В результате можно получить следующее выражение:

$$\frac{\partial \ln P(x)}{\partial w_{ij}} = - \sum_y P(y|x) \frac{\partial E(x, y)}{\partial w_{ij}} + \sum_{x,y} P(x, y) \frac{\partial E(x, y)}{\partial w_{ij}} \quad (2.1)$$

В данном выражении первое слагаемое определяет позитивную fazу работы машины Больцмана, когда сеть работает на основе образов из обучающей выборки. Второе слагаемое характеризует негативную fazу функционирования, когда сеть работает в свободном режиме независимо от окружающей среды.

Если рассматривать энергетическую функцию сети RBM, то в данном случае задача обучения состоит в том, чтобы на основе входных данных найти конфигурацию нейронов скрытого слоя с минимальной энергией. В результате на обучающем множестве сеть будет иметь меньшую энергию по сравнению с другими состояниями. Функция энергии бинарного состояния  $(x, y)$  определяется аналогично энергетической функции сети Хопфилда:

$$E(x, y) = - \sum_i x_i T_i - \sum_j y_j T_j - \sum_{i,j} x_i y_j w_{ij} \quad (2.2)$$

В этом случае

$$\frac{\partial E(x, y)}{\partial w_{ij}} = -x_i y_j$$

Подставляя полученное значение в формулу 2.1, получим

$$\frac{\partial \ln P(x)}{\partial w_{ij}} = \sum_y P(y|x) x_i y_j - \sum_{x,y} P(x, y) x_i y_j$$

Так как математическое ожидание входных данных равняется:

$$E(x) = \sum_i x_i P_i \quad (2.3)$$

то

$$\frac{\partial \ln P(x)}{\partial w_{ij}} = E[x_i y_j]_{\text{data}} - E[x_i y_j]_{\text{model}} \quad (2.4)$$

Рассуждая аналогичным образом, находим

$$\frac{\partial \ln P(x)}{\partial T_i} = - \sum_y P(y|x) \frac{\partial E(x,y)}{\partial T_i} + \sum_{x,y} P(x,y) \frac{\partial E(x,y)}{\partial T_i}$$

и

$$\frac{\partial \ln P(x)}{\partial T_j} = - \sum_y P(y|x) \frac{\partial E(x,y)}{\partial T_j} + \sum_{x,y} P(x,y) \frac{\partial E(x,y)}{\partial T_j}$$

Так как

$$\frac{\partial E(x,y)}{\partial T_i} = -x_i$$

и

$$\frac{\partial E(x,y)}{\partial T_j} = -y_j$$

то, учитывая формулу 2.3, получим градиенты для пороговых значений:

$$\begin{aligned} \frac{\partial \ln P(x)}{\partial T_i} &= E[x_i]_{\text{data}} - E[x_i]_{\text{model}} \\ \frac{\partial \ln P(x)}{\partial T_j} &= E[y_j]_{\text{data}} - E[y_j]_{\text{model}} \end{aligned} \quad (2.5)$$

Как следует из последних выражений, первое слагаемое характеризует работу сети на основе данных из обучающей выборки, а второе слагаемое характеризует работу сети на основе данных модели (данные генерируемые сетью), то есть в свободном режиме независимо от окружающей среды. Так как вычисление математического ожидания в формулах 2.4 и 2.5 является сложной задачей, Хинтон предложил использовать аппроксимацию данных слагаемых, получаемую применением процедуры, которую он назвал контрастным расхождением (Contrastive Divergence (CD)) [35].

Такая аппроксимация основывается на дискретизаторе Гиббса (Gibbs sampling). В этом случае первые слагаемые в выражениях 2.4 и 2.5 характеризуют распределение данных в момент времени  $t = 0$ , а вторые слагаемые характеризуют реконструированные или генерируемые моделью данные в

момент времени  $t = k$ . Исходя из этого, CD- $k$  процедура может быть представлена следующим образом:

$$x(0) \rightarrow y(0) \rightarrow x(1) \rightarrow y(1) \rightarrow \dots \rightarrow x(k) \rightarrow y(k) \quad (2.6)$$

В результате можно сформулировать следующие правила для обучения сети RBM. В случае применения CD-1 (случай  $k = 1$ ) и, учитывая, что в соответствии с методом градиентного подъема

$$w_{ij}(t+1) = w_{ij}(t) + \alpha \frac{\partial \ln P(x)}{\partial w_{ij}(t)}$$

Можно получить, что

$$\begin{aligned} w_{ij}(t+1) &= w_{ij}(t) + \alpha(x_i(0)y_j(0) - x_i(1)y_j(1)) \\ T_i(t+1) &= T_i(t) + \alpha(x_i(0) - x_i(1)) \\ T_j(t+1) &= T_j(t) + \alpha(y_j(0) - y_j(1)). \end{aligned}$$

Аналогичным образом, для алгоритма CD- $k$

$$\begin{aligned} w_{ij}(t+1) &= w_{ij}(t) + \alpha(x_i(0)y_j(0) - x_i(k)y_j(k)) \\ T_i(t+1) &= T_i(t) + \alpha(x_i(0) - x_i(k)) \\ T_j(t+1) &= T_j(t) + \alpha(y_j(0) - y_j(k)). \end{aligned}$$

Из последних выражений видно, что в процессе обучения ограниченной машины Больцмана минимизируется разница между оригиналыми данными и данными, генерируемыми моделью.

Схожим образом можно получить правила обучения для группового случая (например, при использовании mini-batch оптимизации, случай CD-1):

$$\begin{aligned} w_{ij}(t+1) &= w_{ij}(t) + \frac{\alpha}{L} \sum_{l=1}^L (x_i^l(0)y_j^l(0) - x_i^l(1)y_j^l(1)) \\ T_i(t+1) &= T_i(t) + \frac{\alpha}{L} \sum_{l=1}^L (x_i^l(0) - x_i^l(1)) \\ T_j(t+1) &= T_j(t) + \frac{\alpha}{L} \sum_{l=1}^L (y_j^l(0) - y_j^l(1)). \end{aligned}$$

где  $L$  – количество образов в группе.

Суммируя вышесказанное, приведем полный алгоритм обучения RBM для случая CD-1 (алгоритм 1).

---

### Алгоритм: Процедура обучения RBM

---

**Исходные данные:**  $\mathbf{x}(0)$  – вектор-образ из обучающей выборки,  
 $\alpha$  - скорость обучения

**Результат:** матрица весовых коэффициентов  $\mathbf{W}$ , вектор порогов  
нейронов видимого слоя  $\mathbf{V}$ , вектор порогов нейронов  
скрытого слоя  $\mathbf{H}$

**foreach** нейрона скрытого слоя  $j$  **do**

  Вычислить  $P(y_j(0) = 1|x_i(0))$  (для биномиальных нейронов  
  sigmoid( $\sum_i w_{ij}x_i(0) + H_j$ ));  
  Генерировать  $y_j(0) \in \{0, 1\}$  из  $P(y_j(0)|x_i(0))$ ;

**foreach** нейрона видимого слоя  $i$  **do**

  Вычислить  $P(x_i(1) = 1|y_j(0))$  (для биномиальных нейронов  
  sigmoid( $\sum_j w_{ij}y_j(0) + V_i$ ));  
  Генерировать  $x_i(1) \in \{0, 1\}$  из  $P(x_i(1)|y_j(0))$ ;

**foreach** скрытых нейронов  $j$  **do**

  Вычислить  $P(y_j(1) = 1|x_i(1))$  (для биномиальных нейронов  
  sigmoid( $\sum_i w_{ij}x_i(1) + H_j$ ));

$\mathbf{W} \leftarrow \mathbf{W} + \alpha(\mathbf{x}(0)\mathbf{y}(0)^T - \mathbf{x}(1)\mathbf{P}(\mathbf{y}(1) = 1|\mathbf{x}(1))^T);$

$\mathbf{V} \leftarrow \mathbf{V} + \alpha(\mathbf{x}(0) - \mathbf{x}(1));$

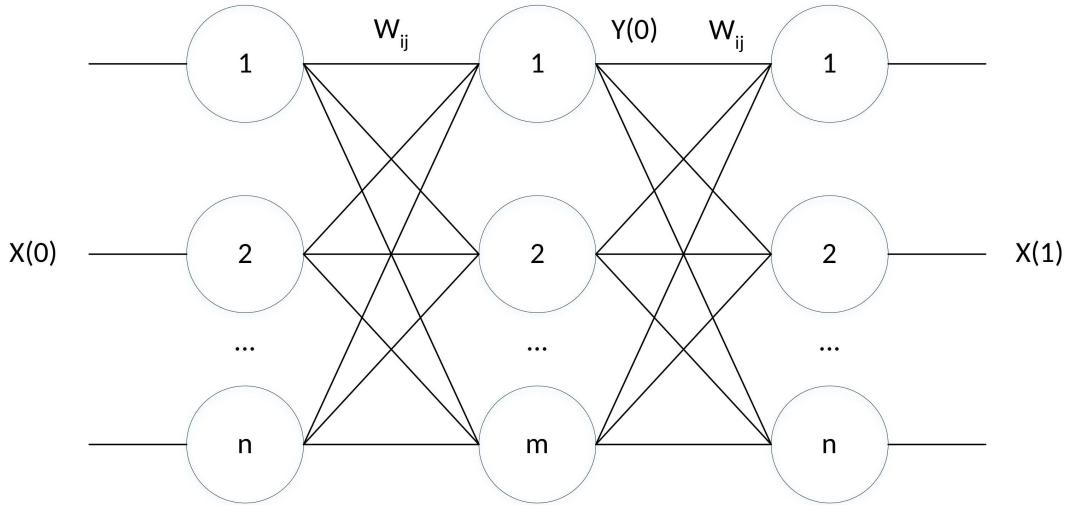
$\mathbf{H} \leftarrow \mathbf{H} + \alpha(\mathbf{y}(0) - \mathbf{P}(\mathbf{y}(1) = 1|\mathbf{x}(1)))$ ;

---

## 2.2 Общее описание предлагаемого метода

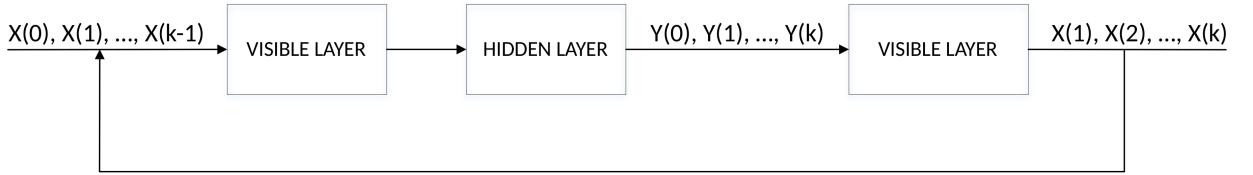
По сравнению с традиционным, предложенным Джекфри Хинтоном, методом обучения RBM [35], который основывается на применении энергетической функции (energy-based method) и фактически оперирует линейным представлением нейронных элементов, предлагаемый нами метод позволяет учитывать их нелинейную природу.

Рассмотрим ограниченную машину Больцмана, которую будем представлять в виде трех слоев нейронных элементов [50]: видимого, скрытого и видимого (рисунок 2.1). В приведенном представлении матрица весовых коэффициентов  $W_{ji}$  для второго слоя получается транспонированием матрицы  $W_{ij}$  первого слоя. Фактически, при выполнении обучения, производится однократное изменение ее элементов.



**Рисунок 2.1. – Развёрнутое представление RBM**

Представим сэмплирование Гиббса, используя развёрнутое представление RBM (рисунок 2.2).



**Рисунок 2.2. – Сэмплирование Гиббса**

Сэмплирование Гиббса заключается в следующей процедуре. Пусть  $x(0)$  – входной вектор, который поступает на видимый слой в момент времени  $t = 0$ . Тогда выходные значения нейронов скрытого слоя будут определяться следующим образом:

$$y_j(0) = F(S_j(0)), \quad (2.7)$$

$$S_j(0) = \sum_i w_{ij} x_i(0) + T_j. \quad (2.8)$$

Инверсный (последний) слой реконструирует входной вектор на основе данных со скрытого слоя и текущего значения настраиваемых параметров (порогов видимого слоя и матрицы весов). В результате получается восстановленный вектор  $x(1)$  в момент времени  $t = 1$ :

$$x_i(1) = F(S_i(1)), \quad (2.9)$$

$$S_i(1) = \sum_j w_{ij} y_j(0) + T_i. \quad (2.10)$$

Затем вектор  $x(1)$  поступает на видимый слой, и вычисляются выходные значения нейронов скрытого слоя:

$$y_j(1) = F(S_j(1)), \quad (2.11)$$

$$S_j(1) = \sum_i w_{ij}x_i(1) + T_j. \quad (2.12)$$

Продолжая данный процесс, можно получить на шаге  $k$  следующие выражения:

$$\begin{aligned} y_j(k) &= F(S_j(k)), \quad S_j(k) = \sum_i w_{ij}x_i(k) + T_j. \\ x_i(k) &= F(S_i(k)), \quad S_i(k) = \sum_j w_{ij}y_j(k-1) + T_i. \end{aligned}$$

## 2.3 Вывод правил предобучения

Хинтоном была предложена энергетическая модель, базирующаяся на идее максимизации функции правдоподобия распределения входных данных  $P(x)$ . Вывод классических правил обучения был приведен в главе I. Основываясь на идее использования ограниченной машины Больцмана в качестве вспомогательной модели для проведения предобучения, нами было предложено использование двух разных критериев для ее обучения [4–A]. Первый критерий основывается на минимизации среднеквадратичной ошибки (MSE), а второй – на минимизации кросс-энтропийной функции ошибки.

Покажем, что применение различных критериев минимизации позволяет, тем не менее, получить одинаковые правила обучения.

### 2.3.1 Критерий MSE

В случае использования в качестве критерия обучения MSE основной целью обучения ограниченной машины Больцмана является минимизация суммарной среднеквадратичной ошибки реконструкции данных на скрытом и видимом (восстанавливающем) слое, которая в случае CD-k определяется следующим образом:

$$E_s(k) = \frac{1}{2L} \left( \sum_{l=1}^L \sum_{j=1}^m \sum_{p=1}^k (y_j^l(p) - y_j^l(p-1))^2 + \sum_{l=1}^L \sum_{i=1}^n \sum_{p=1}^k (x_i^l(p) - x_i^l(p-1))^2 \right)$$

где  $k$  определяет параметр процедуры сэмплирования Гиббса,  $n$  – количество нейронов на видимом слое,  $m$  – количество нейронов на скрытом слое,  $L$  – размерность обучающей выборки.

В случае CD-1 суммарная среднеквадратичная ошибка

$$E_s(1) = \frac{1}{2L} \left( \sum_{l=1}^L \sum_{j=1}^m (y_j^l(1) - y_j^l(0))^2 + \sum_{l=1}^L \sum_{i=1}^n (x_i^l(1) - x_i^l(0))^2 \right) \quad (2.13)$$

Как следует из приведенных выше выражений ошибка состоит из двух частей: ошибки восстановления информации на видимом и скрытом слоях, т.е. может быть представлена в следующем виде:

$$E_s(k) = E_h(k) + E_v(k) \quad (2.14)$$

где

$$E_h(k) = \frac{1}{2L} \sum_{l=1}^L \sum_{j=1}^m \sum_{p=1}^k (y_j^l(p) - y_j^l(p-1))^2 \quad (2.15)$$

$$E_v(k) = \frac{1}{2L} \sum_{l=1}^L \sum_{i=1}^n \sum_{p=1}^k (x_i^l(p) - x_i^l(p-1))^2 \quad (2.16)$$

Найдем правила обучения, соответствующие критерию 2.14 и докажем их эквивалентность классическим правилам обучения RBM при выполнении некоторых специальных условий.

**Теорема 1.** Максимизация функции правдоподобия распределения данных  $P(x)$  в пространстве синаптических связей ограниченной машины Больцмана эквивалентна минимизации суммарной квадратичной ошибки сети в том же пространстве при использовании линейных нейронов.

**Доказательство:** Рассмотрим последовательное обучение RBM, когда модификация синаптических связей происходит после подачи каждого входного образа на сеть (онлайн-обучение). В соответствии с методом градиентного спуска для минимизации суммарной квадратичной ошибки сети, синаптические связи должны изменяться следующим образом:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha \frac{\partial E}{\partial w_{ij}(t)}, \quad (2.17)$$

$$T_i(t+1) = T_i(t) - \alpha \frac{\partial E}{\partial T_i(t)}, \quad (2.18)$$

$$T_j(t+1) = T_j(t) - \alpha \frac{\partial E}{\partial T_j(t)}, \quad (2.19)$$

В случае CD-k квадратичная ошибка  $E$  для одного образа:

$$E = \frac{1}{2} \sum_{j=1}^m \sum_{p=1}^k (y_j(p) - y_j(p-1))^2 + \frac{1}{2} \sum_{i=1}^n \sum_{p=1}^k (x_i(p) - x_i(p-1))^2$$

Тогда

$$\begin{aligned} \frac{\partial E}{\partial w_{ij}} &= \frac{\partial E}{\partial y_j(p)} \frac{\partial y_j(p)}{\partial S_j(p)} \frac{\partial S_j(p)}{\partial w_{ij}} + \frac{\partial E}{\partial x_i(p)} \frac{\partial x_i(p)}{\partial S_i(p)} \frac{\partial S_i(p)}{\partial w_{ij}} = \\ &= \sum_{p=1}^k (y_j(p) - y_j(p-1)) x_i(p) F'(S_j(p)) + \\ &\quad + \sum_{p=1}^k (x_i(p) - x_i(p-1)) y_j(p-1) F'(S_i(p)) \end{aligned}$$

Если ограниченная машина Больцмана использует линейные нейроны с линейной функцией активации, то

$$\frac{\partial S_i(p)}{\partial w_{ij}} = F'(S_i(p)) = \frac{\partial S_j(p)}{\partial w_{ij}} = F'(S_j(p)) = 1,$$

Тогда

$$\frac{\partial E}{\partial w_{ij}} = \sum_{p=1}^k (y_j(p)x_i(p) - y_j(p-1)x_i(p-1)) = y_j(k)x_i(k) - y_j(0)x_i(0),$$

В результате можно получить CD-k правило обучения RBM:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha(x_i(0)y_j(0) - x_i(k)y_j(k)),$$

Аналогичным образом для пороговых значений:

$$\begin{aligned} T_j(t+1) &= T_j(t) + \alpha(y_j(0) - y_j(k)), \\ T_i(t+1) &= T_i(t) + \alpha(x_i(0) - x_i(k)) \end{aligned}$$

Как видно последние выражения совпадают с классическим правилом обучения ограниченной машины Больцмана для CD-k. Отсюда следует,

что для линейной RBM максимизация функции правдоподобия распределения данных  $P(x)$  эквивалентна минимизации суммарной квадратичной ошибки сети. Теорема доказана.

**Следствие 1.1.** Линейная ограниченная машина Больцмана с точки зрения обучения эквивалентна автоассоциативной нейронной сети при использовании в ней при обучении сэмплирования Гиббса.

**Следствие 1.2.** Для нелинейной ограниченной машины Больцмана правило модификации синаптических связей в случае CD-k будет следующим:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha \left( \sum_{p=1}^k (y_j(p) - y_j(p-1))x_i(p)F'(S_j(p)) + (x_i(p) - x_i(p-1))y_j(p-1)F'(S_i(p)) \right)$$

$$\begin{aligned} T_i(t+1) &= T_i(t) - \alpha \left( \sum_{p=1}^k (x_i(p) - x_i(p-1))F'(S_i(p)) \right), \\ T_j(t+1) &= T_j(t) - \alpha \left( \sum_{p=1}^k (y_j(p) - y_j(p-1))F'(S_j(p)) \right), \end{aligned}$$

**Следствие 1.3.** Для нелинейной ограниченной машины Больцмана правило модификации синаптических связей в случае CD-1 будет следующим:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha((y_j(1) - y_j(0))F'(S_j(1))x_i(1) + (x_i(1) - x_i(0))F'(S_i(1))y_j(0)),$$

$$T_i(t+1) = T_i(t) - \alpha(x_i(1) - x_i(0))F'(S_i(1)),$$

$$T_j(t+1) = T_j(t) - \alpha(y_j(1) - y_j(0))F'(S_j(1)).$$

При использовании группового обучения (batch learning), метод градиентного спуска примет следующий вид:

$$w_{ij}(t+1) = w_{ij}(t) - \frac{\alpha}{L} \frac{\partial E_s}{\partial w_{ij}(t)} \quad (2.20)$$

$$T_i(t+1) = T_i(t) - \frac{\alpha}{L} \frac{\partial E_s}{\partial T_i(t)} \quad (2.21)$$

$$T_j(t+1) = T_j(t) - \frac{\alpha}{L} \frac{\partial E_s}{\partial T_j(t)} \quad (2.22)$$

где  $L$  определяет размер мини-батча.

**Теорема 2.** При использовании CD-k для нелинейной ограниченной машины Больцмана в случае группового обучения правило модификации синаптических связей определяется на основе следующих выражений:

$$w_{ij}(t+1) = w_{ij}(t) - \frac{\alpha}{L} \left( \sum_{l=1}^L \sum_{p=1}^k (y_j^l(p) - y_j^l(p-1)) x_i^l(p) F'(S_j^l(p)) + (x_i^l(p) - x_i^l(p-1)) y_j^l(p-1) F'(S_i^l(p)) \right),$$

$$T_i(t+1) = T_i(t) - \frac{\alpha}{L} \left( \sum_{l=1}^L \sum_{p=1}^k (x_i^l(p) - x_i^l(p-1)) F'(S_i^l(p)) \right),$$

$$T_j(t+1) = T_j(t) - \frac{\alpha}{L} \left( \sum_{l=1}^L \sum_{p=1}^k (y_j^l(p) - y_j^l(p-1)) F'(S_j^l(p)) \right)$$

Процесс доказательства данной теоремы является аналогичным доказательству теоремы 1.

**Следствие 2.1.** При использовании CD-1 для нелинейной ограниченной машины Больцмана в случае группового обучения правило модификации синаптических связей определяется на основе следующих выражений:

$$w_{ij}(t+1) = w_{ij}(t) - \frac{\alpha}{L} \left( \sum_{l=1}^L (y_j^l(1) - y_j^l(0)) x_i^l(1) F'(S_j^l(1)) + (x_i^l(1) - x_i^l(0)) y_j^l(0) F'(S_i^l(1)) \right),$$

$$T_i(t+1) = T_i(t) - \frac{\alpha}{L} \left( \sum_{l=1}^L (x_i^l(1) - x_i^l(0)) F'(S_i^l(1)) \right),$$

$$T_j(t+1) = T_j(t) - \frac{\alpha}{L} \left( \sum_{l=1}^L (y_j^l(1) - y_j^l(0)) F'(S_j^l(1)) \right)$$

**Следствие 2.2.** При использовании CD-k для линейной ограниченной машины Больцмана в случае группового обучения правило модификации синаптических связей определяется на основе следующих выражений:

$$w_{ij}(t+1) = w_{ij}(t) + \frac{\alpha}{L} \sum_{l=1}^L (x_i^l(0)y_j^l(0) - x_i^l(k)y_j^l(k)),$$

$$T_i(t+1) = T_i(t) + \frac{\alpha}{L} \sum_{l=1}^L (x_i^l(0) - x_i^l(k)),$$

$$T_j(t+1) = T_j(t) + \frac{\alpha}{L} \sum_{l=1}^L (y_j^l(0) - y_j^l(k))$$

**Следствие 2.3.** При использовании CD-1 для линейной ограниченной машины Больцмана в случае группового обучения правило модификации синаптических связей определяется на основе следующих выражений:

$$w_{ij}(t+1) = w_{ij}(t) + \frac{\alpha}{L} \sum_{l=1}^L (x_i^l(0)y_j^l(0) - x_i^l(1)y_j^l(1)),$$

$$T_i(t+1) = T_i(t) + \frac{\alpha}{L} \sum_{l=1}^L (x_i^l(0) - x_i^l(1)),$$

$$T_j(t+1) = T_j(t) + \frac{\alpha}{L} \sum_{l=1}^L (y_j^l(0) - y_j^l(1))$$

Таким образом, получены правила обучения для ограниченной машины Больцмана, которые базируются на минимизации квадратичной ошибки восстановления информации на видимом и скрытом слоях. Предложенный метод позволяет учитывать нелинейную природу нейронных элементов. Показано, что классические выражения для обучения ограниченной машины являются частным случаем предложенного метода. Доказана теорема об эквивалентности максимизации функции правдоподобия распределения входных данных  $P(x)$  и минимизации суммарной квадратичной ошибки сети в одном и том же пространстве синаптических связей для линейной ограниченной машины Больцмана.

### 2.3.2 Критерий СЕ

В случае CD-k кросс-энтропийная функция ошибки для видимого слоя определяется следующим образом:

$$CE_v(k) = -\frac{1}{L} \sum_{l=1}^L \sum_{p=1}^k \sum_{i=1}^n x_i^l(p-1) \log(x_i^l(p)) + (1 - x_i^l(p-1)) \log(1 - x_i^l(p))$$

где  $L$  – размер обучающей выборки,  $k$  – параметр процедуры сэмплирования,  $n$  – количество нейронов на видимом слое.

Аналогично для скрытого слоя:

$$CE_h(k) = -\frac{1}{L} \sum_{l=1}^L \sum_{p=1}^k \sum_{j=1}^m y_j^l(p-1) \log(y_j^l(p)) + (1 - y_j^l(p-1)) \log(1 - y_j^l(p))$$

где  $m$  – количество нейронов на скрытом слое.

Общая функция определяется как сумма кросс-энтропийных функций ошибки видимого и скрытого слоев:

$$CE_s(k) = CE_h(k) + CE_v(k) \quad (2.23)$$

Докажем следующую теорему.

**Теорема 3.** Максимизация функции правдоподобия распределения входных данных  $P(x)$  эквивалентна минимизации кросс-энтропийной целевой функции  $CE_s(k)$  в одном и том же пространстве синаптических весов ограниченной машины Больцмана (случай  $k = 1$ ).

**Доказательство.** В случае CD-1 кросс-энтропийная функция для одного обучающего примера будет иметь следующий вид:

$$\begin{aligned} CE(1) &= - \sum_{i=1}^n (x_i(0) \log(x_i(1)) + (1 - x_i(0)) \log(1 - x_i(1))) - \\ &- \sum_{j=1}^m (y_j(0) \log(y_j(1)) + (1 - y_j(0)) \log(1 - y_j(1))) = CE_v(1) + CE_h(1) \end{aligned} \quad (2.24)$$

Найдем частные производные кросс-энтропийной функции по весовым элементам. Получим

$$\frac{\partial CE(1)}{\partial w_{ij}} = \frac{\partial CE_v(1)}{\partial w_{ij}} + \frac{\partial CE_h(1)}{\partial w_{ij}}$$

Тогда

$$\begin{aligned} \frac{\partial CE_v(1)}{\partial w_{ij}} &= -\frac{x_i(0)}{x_i(1)} x_i(1)(1 - x_i(1)) y_j(0) + \frac{1 - x_i(0)}{1 - x_i(1)} x_i(1)(1 - x_i(1)) y_j(0) = \\ &= -x_i(0)(1 - x_i(1)) y_j(0) + (1 - x_i(0)) x_i(1) y_j(0) = \\ &= -x_i(0) y_j(0) + x_i(0) x_i(1) y_j(0) + x_i(1) y_j(0) - x_i(0) x_i(1) y_j(0) = \\ &= -x_i(0) y_j(0) + x_i(1) y_j(0) \end{aligned}$$

и

$$\begin{aligned}
\frac{\partial CE_h(1)}{\partial w_{ij}} &= -y_j(0)(1 - y_j(1))x_i(1) + (1 - y_j(0))y_j(1)x_i(1) = \\
&= -y_j(0)x_i(0) + y_j(0)y_j(1)x_i(1) + y_j(1)x_i(1) - y_j(0)y_j(1)x_i(1) = \\
&= -y_j(0)x_i(1) + y_j(1)x_i(1).
\end{aligned}$$

Окончательно получим

$$\begin{aligned}
\frac{\partial CE(1)}{\partial w_{ij}} &= \frac{\partial CE_v(1)}{\partial w_{ij}} + \frac{\partial CE_h(1)}{\partial w_{ij}} = \\
&= -x_i(0)y_j(0) + x_i(1)y_j(0) - y_j(0)x_i(1) + y_j(1)x_i(1) = x_i(1)y_j(1) - x_i(0)y_j(0).
\end{aligned}$$

Аналогично, для пороговых элементов имеем:

$$\begin{aligned}
\frac{\partial CE(1)}{\partial T_i} &= \frac{\partial CE_v(1)}{\partial T_i} = -\frac{x_i(0)}{x_i(1)}x_i(1)(1 - x_i(1)) + \frac{1 - x_i(0)}{1 - x_i(1)}x_i(1)(1 - x_i(1)) = \\
&= -x_i(0) + x_i(0)x_i(1) + x_i(1) - x_i(0)x_i(1) = x_i(1) - x_i(0)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial CE(1)}{\partial T_j} &= \frac{\partial CE_h(1)}{\partial T_j} = -y_j(0)(1 - y_j(1)) + (1 - y_j(0))y_j(1) = \\
&= -y_j(0) + y_j(0)y_j(1) + y_j(1) - y_j(0)y_j(1) = y_j(1) - y_j(0).
\end{aligned}$$

Теорема доказана.

Рассматривая более общий случай CD- $k$ , можно доказать следующую теорему.

**Теорема 4.** Максимизация функции правдоподобия распределения входных данных  $P(x)$  эквивалентна минимизации кросс-энтропийной целевой функции  $CE_s(k)$  в одном и том же пространстве синаптических весов ограниченной машины Больцмана (случай произвольного  $k$ ).

**Доказательство.** В случае CD- $k$  кросс-энтропийная функция для одного обучающего примера будет иметь следующий вид:

$$\begin{aligned}
CE_s(k) = & - \sum_{p=1}^k \sum_{i=1}^n (x_i(p-1) \log(x_i(p)) + (1-x_i(p-1)) \log(1-x_i(p))) - \\
& - \sum_{p=1}^k \sum_{j=1}^m (y_j(p-1) \log(y_j(p)) + (1-y_j(p-1)) \log(1-y_j(p))) = CE_v(k) + CE_h(k)
\end{aligned} \tag{2.25}$$

Как и в случае CD-1 находим соответствующие частные производные по весовым и пороговым элементам. Имеем:

$$\frac{\partial CE_s(k)}{\partial w_{ij}} = \frac{\partial CE_v(k)}{\partial w_{ij}} + \frac{\partial CE_h(k)}{\partial w_{ij}}$$

Причем

$$\begin{aligned}
\frac{\partial CE_v(k)}{\partial w_{ij}} = & \\
= & - \sum_{p=1}^k (x_i(p-1)(1-x_i(p))y_j(p-1) - (1-x_i(p-1))x_i(p)y_j(p-1)) = \\
= & - \sum_{p=1}^k (x_i(p-1)y_j(p-1) - x_i(p-1)x_i(p)y_j(p-1) - \\
& - x_i(p)y_j(p-1) + x_i(p-1)x_i(p)y_j(p-1)) = \\
= & \sum_{p=1}^k (x_i(p)y_j(p-1) - x_i(p-1)y_j(p-1))
\end{aligned}$$

И

$$\begin{aligned}
\frac{\partial CE_h(k)}{\partial w_{ij}} &= \\
&= - \sum_{p=1}^k (y_j(p-1)(1-y_j(p))x_i(p) - (1-y_j(p-1))y_j(p)x_i(p)) = \\
&= - \sum_{p=1}^k (y_j(p-1)x_i(p) - y_j(p-1)y_j(p)x_i(p) - \\
&\quad - y_j(p)x_i(p) + y_j(p-1)y_j(p)x_i(p)) = \\
&= \sum_{p=1}^k (x_i(p)y_j(p) - x_i(p)y_j(p-1)).
\end{aligned}$$

А значит

$$\begin{aligned}
\frac{\partial CE_s(k)}{\partial w_{ij}} &= \sum_{p=1}^k (y_j(p)x_i(p) - x_i(p-1)y_j(p-1)) = \\
&= y_j(1)x_i(1) - x_i(0)y_j(0) + x_i(2)y_j(2) - x_i(1)y_j(1) + \dots + \\
&\quad + x_i(k)y_j(k) - x_i(k-1)y_j(k-1) = \\
&= x_i(k)y_j(k) - x_i(0)y_j(0).
\end{aligned}$$

Аналогично для пороговых элементов:

$$\begin{aligned}
\frac{\partial CE_s(k)}{\partial T_i} &= \frac{\partial CE_v(k)}{\partial T_i} = \\
&= - \sum_{p=1}^k (x_i(p-1)(1-x_i(p)) - (1-x_i(p-1))x_i(p)) = \\
&= \sum_{p=1}^k (x_i(p) - x_i(p-1)) = x_i(1) - x_i(0) + x_i(2) - x_i(1) + \dots + x_i(k) - x_i(k-1) = \\
&= x_i(k) - x_i(0).
\end{aligned}$$

$$\begin{aligned}
\frac{\partial CE_s(k)}{\partial T_j} &= \frac{\partial CE_h(k)}{\partial T_j} = \\
&= - \sum_{p=1}^k (y_j(p-1)(1-y_j(p)) - (1-y_j(p-1))y_j(p)) = \\
&= \sum_{p=1}^k (y_j(p) - y_j(p-1)) = y_j(1) - y_j(0) + y_j(2) - y_j(1) + \dots + y_j(k) - y_j(k-1) = \\
&= y_j(k) - y_j(0).
\end{aligned}$$

Теорема доказана.

Из доказанных теорем 3 и 4 следует, что правила обучения ограниченной машины Больцмана могут быть получены более простым путем, чем при использовании традиционного подхода, основанного на применении функции энергии. Произведя минимизацию кросс-энтропийной функции и применяя простые итерации сэмплирования Гиббса, мы получили классические линейные правила обучения RBM.

Таким образом, на основании доказанных теорем 1-4 можно сформулировать общую теорему.

**Теорема 5.** Максимизация функции правдоподобия распределения входных данных  $P(x)$  эквивалентна минимизации кросс-энтропийной функции и специальному случаю минимизации среднеквадратичной ошибки в одном и том же пространстве синаптических весов ограниченной машины Больцмана:

$$\max(\ln P(x)) = \min(CE_s) = \min(E_s) \quad (2.26)$$

Теорема 5 обобщает предыдущие результаты. Как следует из этой теоремы, использование различных критериев обучения приводит к тем же самим правилам. Поэтому сущность неконтролируемого обучения для ограниченной машины Больцмана одинакова, даже если используются различные целевые функции. Максимизация функции правдоподобия и минимизация кросс-энтропийной функции ошибки приводят к линейному представлению нейронных элементов в терминах минимизации функции MSE. Необходимо отметить, что, применяя критерий MSE, мы учитываем также нелинейное представление нейронных элементов. Таким образом, очевидное преимущество в использовании MSE в качестве функции ошибок перед применением функции максимального правдоподобия и кросс-энтропийной функции в том, что при использовании MSE могут быть получены как линейные, так и нелинейные правила обучения.

В дальнейшем для идентификации предлагаемого подхода будем использовать аббревиатуру REBA (Reconstruction Error-Based Approach). Классический метод будем называть C-RBM (Classic Restricted Boltzmann Machine Pretraining).

## 2.4 Предобучение сверточных слоев

Выполнение предобучения сверточных слоев приобретает особую актуальность при решении задач компьютерного зрения в силу эффективности таких слоев при обработке визуальной информации, представленной с помощью отдельных изображений и видео.

Отметим, что преобразования, выполняемые над сверточными слоями при обучении, идентичны преобразованиям для полносвязных слоев. Отличие заключается в операции «развертки» (deconvolution), которая используется на этапе предобучения при реконструкции активности видимых нейронов и на этапе тонкой настройки при вычислении ошибок на скрытых слоях сети. Эта операция применяется в сверточных нейронных сетях для повышения частоты дискретизации. Рассмотрим визуальное представление операции свертки для случая карт 3Х3 и 2Х2 (рис. 2.3).

$$\begin{array}{|c|c|c|} \hline X_1 & X_2 & X_3 \\ \hline X_4 & X_5 & X_6 \\ \hline X_7 & X_8 & X_9 \\ \hline \end{array} * \begin{array}{|c|c|} \hline y_1 & y_2 \\ \hline y_3 & y_4 \\ \hline \end{array} = \begin{array}{|c|c|} \hline Z_1 & Z_2 \\ \hline Z_3 & Z_4 \\ \hline \end{array}$$

Рисунок 2.3. – Операция свертки

Сверточный слой с одним ядром может быть представлен в виде разряженного слоя, а операция «развертки» – слоем, полученным применением симметричного преобразования относительно разряженного. При этом полученная сеть представляет собой автоэнкодер с симметричными связями относительно скрытого слоя (рис. 2.4).

На основании данного представления могут быть получены формулы для выполнения «развертки»:

$$x'_1 = z_1 y_1$$

$$x'_2 = z_1 y_2 + z_2 y_1$$

$$x'_3 = z_2 y_2$$

$$x'_4 = z_1 y_3 + z_3 y_1$$

$$x'_5 = z_1y_4 + z_2y_3 + z_3y_2 + z_4y_1$$

$$x'_6 = z_2y_4 + z_4y_2$$

$$x'_7 = z_3y_3$$

$$x'_8 = z_3y_4 + z_4y_3$$

$$x'_9 = z_4y_4$$

Сама операция «развертки» представлена на рис. 2.5.

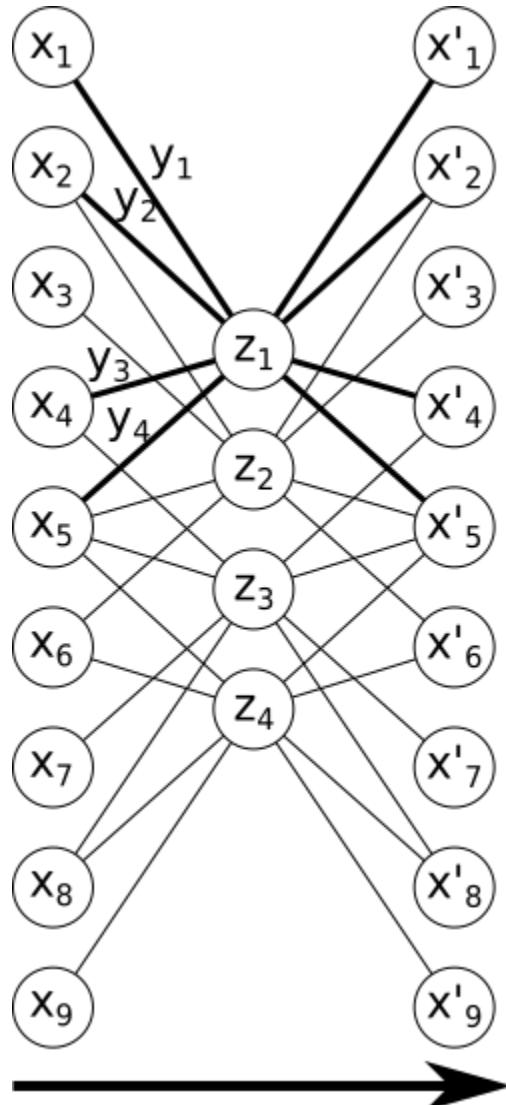


Рисунок 2.4. – Разряженный автоэнкодер операции свертки

0	0	0	0
0	$Z_1$	$Z_2$	0
0	$Z_3$	$Z_4$	0
0	0	0	0

$$\ast \begin{bmatrix} y_4 & y_3 \\ y_2 & y_1 \end{bmatrix} = \begin{bmatrix} x'_1 & x'_2 & x'_3 \\ x'_4 & x'_5 & x'_6 \\ x'_7 & x'_8 & x'_9 \end{bmatrix}$$

Рисунок 2.5. – Операция «развертки»

Введя операции матричного поворота на 180 градусов и заполнения нулями границ матрицы (padding), можно получить следующие общие формулы для выполнения «развертки» (рис. 2.6).

$$\text{padding}\left(\begin{array}{|c|c|} \hline Z_1 & Z_2 \\ \hline Z_3 & Z_4 \\ \hline \end{array}\right), 1) * \text{rot180}\left(\begin{array}{|c|c|} \hline Y_1 & Y_2 \\ \hline Y_3 & Y_4 \\ \hline \end{array}\right) = \begin{array}{|c|c|c|} \hline X'_1 & X'_2 & X'_3 \\ \hline X'_4 & X'_5 & X'_6 \\ \hline X'_7 & X'_8 & X'_9 \\ \hline \end{array}$$

Рисунок 2.6. – Формула «развертки»

Введем условное обозначение для операции «развертки» –  $\circledast$ .

Применяя полученную формулу для выполнения «развертки», можно осуществлять процедуру CD на сверточных слоях ГНС, которые в этом случае рассматриваются как сверточные ограниченные машины Больцмана (CRBM).

Рассмотрим формулы для получения необходимых карт признаков для расчета градиента изменения параметров слоя в случае использования процедуры CD-1:

$$y_j(0) = F(x_i(0) * w_{ij} + T_j)$$

$$x_i(1) = F(y_j(0) \circledast w_{ij} + T_i)$$

$$y_j(1) = F(x_i(1) * w_{ij} + T_j)$$

где  $w_{ij}$  –  $i$ -тый компонент  $j$ -того ядра свертки,  $x_i(0)$  –  $i$ -тый компонент входной карты признаков,  $T_j$  – пороговый элемент  $j$ -того нейрона скрытого слоя,  $y_j(0)$  –  $j$ -тый компонент выходной карты признаков,  $x_i(1)$  –  $i$ -тый компонент реконструированной входной карты признаков,  $T_i$  – пороговый элемент  $i$ -того нейрона видимого слоя,  $y_j(1)$  –  $j$ -тый компонент реконструированной выходной карты признаков.

Классические правила обучения RBM могут быть легко переформулированы для случая CRBM (CD-1, последовательное обучение):

$$w_{ij}(t+1) = w_{ij}(t) + \alpha(x_i(0) * y_j(0) - x_i(1) * y_j(1))$$

$$T_i(t+1) = T_i(t) + \alpha(x_i(0) - x_i(1))$$

$$T_j(t+1) = T_j(t) + \alpha(y_j(0) - y_j(1))$$

В соответствии с предлагаемым методом, правила обучения могут быть переформулированы следующим образом (CD-1, последовательное обучение):

$$w_{ij}(t+1) = w_{ij}(t) - \alpha((y_j(1) - y_j(0))F'(S_j(1)) * x_i(1) + (x_i(1) - x_i(0))F'(S_i(1)) * y_j(0)),$$

$$T_i(t+1) = T_i(t) - \alpha(x_i(1) - x_i(0))F'(S_i(1)),$$

$$T_j(t+1) = T_j(t) - \alpha(y_j(1) - y_j(0))F'(S_j(1)).$$

Аналогичным образом могут быть получены правила для CD- $k$  случая процедуры обучения.

Таким образом, для глубокой сверточной нейронной сети, содержащей слои разных типов (полносвязные и сверточные) возможно сочетание нескольких вариантов обучения – с представлением в виде CRBM (для сверточных слоев) и в виде RBM (для полносвязных).

## 2.5 Метод редуцирования параметров

Известно, что полносвязные нейронные сети обладают определенной степенью избыточности. Полносвязный слой в сравнении со сверточным содержит большее количество настраиваемых параметров, однако в задачах компьютерного зрения сверточные нейронные сети показывают существенно лучшие результаты по обобщающей способности, чем полносвязные. Таким образом, очевидно, что в полносвязных сетях при большем количестве настраиваемых параметров, они используются менее оптимально. Можно предположить, что указанные «избыточные» параметры могут быть исключены без существенного ухудшения эффективности работы модели. Важный вопрос, возникающий при выполнении такой операции редуцирования, касается самого алгоритма отсеивания малоинформационных параметров. К настоящему моменту предложено несколько работ, в которых авторами выполняется уменьшение размерности нейросетевых архитектур (например, [51], [52]).

Рассмотрим подход для редуцирования связей полносвязной нейронной сети, основанный на использовании предобучения. Первый и четвертый этапы данной процедуры аналогичны этапам выполнения предобучения типа I, описанного в главе I. В ходе выполнения дополнительных этапов 2 и 3 формируются разряженные связи между входными и выходными нейронными элементами слоя и уменьшается его размерность за счет удаления

части нейронных элементов, которые не используются при «тонкой настройке» и дальнейшей эксплуатации нейросетевой модели (рисунок 2.7):

1) Неконтролируемое предобучение НС с использованием «жадного» алгоритма, начиная с первого слоя. Параметры каждого слоя, представленные весовыми и пороговыми коэффициентами, настраиваются в соответствии с правилами обучения ограниченной машины Больцмана;

2) «Обнуление» весовых коэффициентов слоев нейронной сети, абсолютные значения которых не превышают некоторый заданный порог  $t > 0$ . Иначе говоря, параметры со значениями, попадающими в интервал  $[-t, t]$ , не изменяются при дальнейшем обучении;

3) Архитектурная реконфигурация нейронной сети, в ходе которой удаляются нейроны, не участвующие в формировании выходной активности сети (нейроны, имеющие нулевые векторы весовых коэффициентов или, в случае использования сверточных слоев, нейроны, имеющие нулевые ядра свертки). Реконфигурация выполняется в соответствии со следующими правилами:

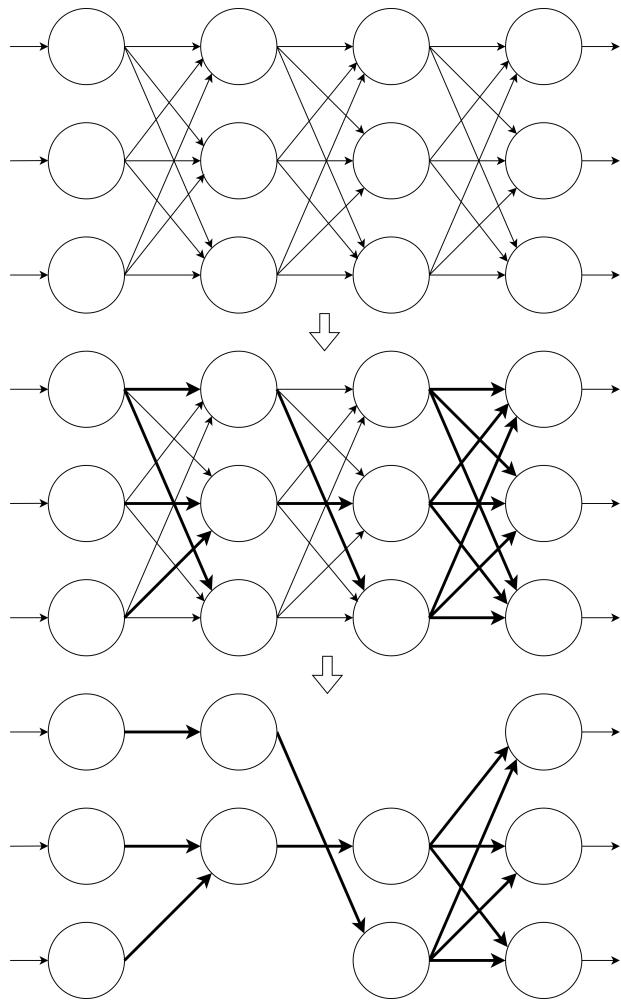
для каждого  $i$ -того слоя НС, кроме первого и последнего:

- если вектор-столбец  $j$  матрицы весовых коэффициентов  $W_i$  нулевой, то удалить  $j$ -тый вектор-столбец из  $W_i$  и удалить  $j$ -тую вектор-строку из  $W_{i+1}$ ;
- если вектор-строка  $k$  матрицы весовых коэффициентов  $W_i$  нулевая, то удалить  $k$ -тую вектор-строку из  $W_i$  и удалить  $k$ -тый вектор-столбец из матрицы  $W_{i-1}$ ;

4) «Тонкая настройка» параметров слоев полученной НС с упрощенной архитектурой, например, методом обратного распространения ошибки.

Этап 2 может дополнительно включать уплотнение для разреженной матрицы параметров в целях достижения более компактного представления весовых коэффициентов.

Таким образом, в процессе реализации данного алгоритма осуществляется отсеивание весовых коэффициентов, значениями которых можно пренебречь в соответствии с некоторым заданным порогом.



**Рисунок 2.7. – Метод редуцирования весовых коэффициентов**

## 2.6 Выводы

- 1) Предложен альтернативный подход к обучению ограниченной машины Больцмана, базирующийся на идее минимизации среднеквадратичной ошибки восстановления образов на скрытом и видимом слоях.
- 2) Доказана теорема об эквивалентности максимизации функции правдоподобия распределения входных данных  $P(x)$  и минимизации суммарной квадратичной ошибки восстановления образов на скрытых и видимых слоях в одном и том же пространстве синаптических связей ограниченной машины Больцмана при использовании линейных нейронов.
- 3) Доказана теорема об эквивалентности максимизации функции правдоподобия распределения входных данных  $P(x)$  минимизации кросс-энтропийной целевой функции  $CE_s(k)$  в одном и том же пространстве синаптических весов ограниченной машины Больцмана (случаи CD-1 и CD-k).
- 4) Сформулирована обобщающая теорема об эквивалентности максимизации функции правдоподобия распределения входных данных  $P(x)$ , ми-

нимизации кросс-энтропийной функции, минимизации суммарной квадратичной ошибки (специальный случай) в одном и том же пространстве синаптических весов ограниченной машины Больцмана.

- 5) Приведены правила обучения для различных случаев линейной и нелинейной машины Больцмана (RBM).
- 6) Приведены правила для обучения сверточных слоев ГНС с использованием модели CRBM.
- 7) Предложен метод редуцирования параметров глубокой полносвязанной нейронной сети, основывающийся на использовании процедуры неконтролируемого предобучения слоев.

# ГЛАВА 3

## ЭКСПЕРИМЕНТАЛЬНОЕ ОБОСНОВАНИЕ МЕТОДА НЕКОНТРОЛИРУЕМОГО ПРЕДОБУЧЕНИЯ

В данной главе приводится практическое обоснование теоретических результатов, описанных в главе №2, а именно, результаты сравнительного анализа методов предобучения и оценки редуцирования весовых коэффициентов нейросетевых моделей.

Представленные результаты получены на известных выборках компьютерного зрения, используемых для оценки эффективности выбранных архитектур и подходов:

- 1) выборка MNIST, представляющая собой набор из 60.000 черно-белых изображений рукописных цифр размерностью 28X28 пикселей, представленных в виде одномерных векторов данных [53];
- 2) выборка CIFAR-10, представляющая собой набор из 60.000 цветных изображений объектов 10 различных классов размерностью 32X32 пикселя [54];
- 3) выборка CIFAR-100, представляющая собой набор из 60.000 цветных изображений объектов 100 различных классов размерность 32X32 пикселя [54].

Приводятся результаты экспериментальных исследований, полученных для задач сжатия данных, распознавания образов и редуцирования параметров ГНС.

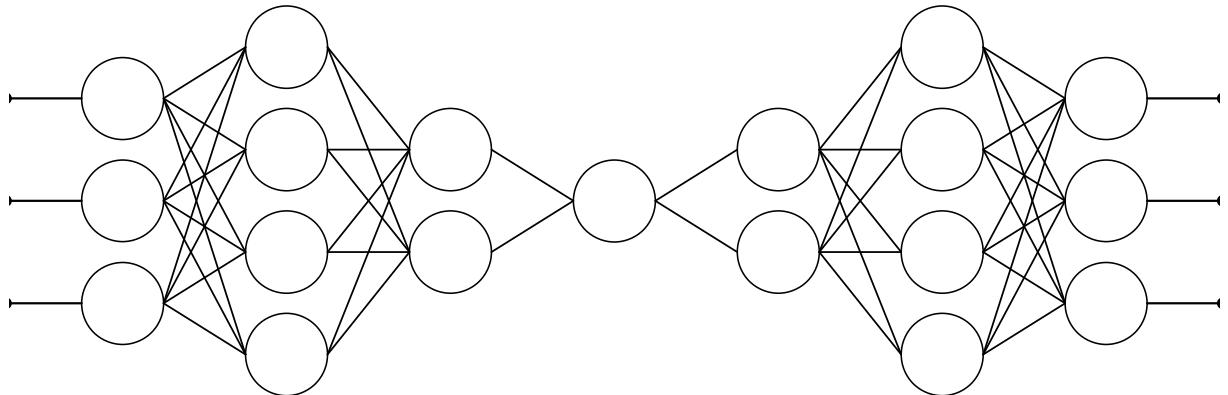
### 3.1 Сжатие данных

Для оценки эффективности предлагаемого подхода предобучения глубокой нейронной сети, были проведены вычислительные эксперименты на искусственном наборе данных  $x$ , лежащем на одномерном многообразии (спиральная петля), погруженном в трехмерное пространство [55] и генерируемом равномерно распределенным параметром  $t \in [-1, 1]$ :

$$\begin{cases} x_1 = \sin(\pi t) + \mu \\ x_2 = \cos(\pi t) + \mu \\ x_3 = t + \mu \end{cases}$$

где  $\mu$  – Гауссов шум со средним 0 и стандартным отклонением 0.05.

Для проверки предложенного метода, нами был обучен семислойный автоэнкодер, используя выборку из 1000 примеров. Глубокий автоэнкодер показан на рисунке 3.1. Мы использовали логистические функции активации для всех слоев нейронной сети, за исключением среднего слоя («узкого места»). На этом слое использовалась линейная функция активации.



**Рисунок 3.1. – Глубокий автоэнкодер**

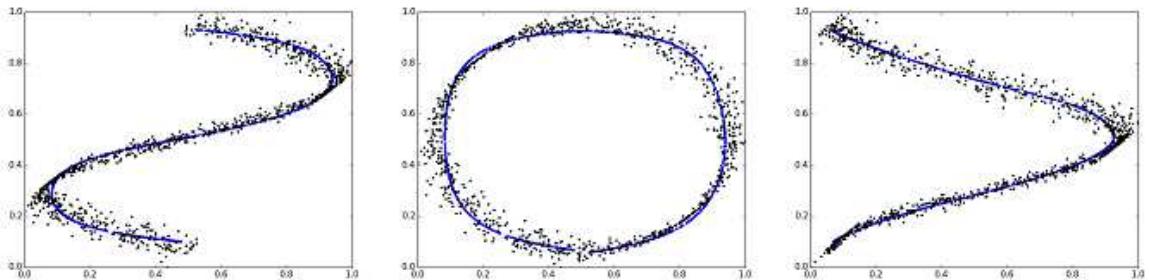
Результаты обучения показаны в таблице 3.1. Здесь MSE – среднеквадратичная ошибка на выборке обучения, MS – среднеквадратичная ошибка на выборке тестирования для проверки обобщающей способности сети. Количество примеров в выборке тестирования – 1000. Скорость обучения  $\alpha$  – 0.1 для классического метода обучения и 0.5 для REBA для всех экспериментов.

Таблица 3.1. – Сравнение методов предобучения (сжатие)

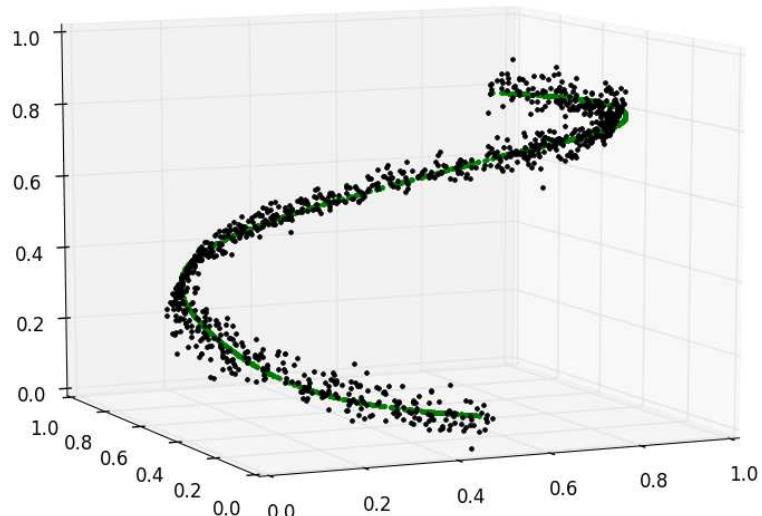
Метод	CD-k	MSE	MS
C-RBM	1	0,699	0,886
	5	<b>0,710</b>	<b>0,932</b>
	10	0,689	0,916
	15	<b>0,688</b>	<b>0,873</b>
REBA	1	<b>0,673</b>	<b>0,851</b>
	5	0,719	0,966
	10	<b>0,677</b>	<b>0,907</b>
	15	0,700	0,895

Число эпох предобучения – 10. Число эпох «тонкой настройки» нейронной сети – 1000. Из полученных результатов видно, что использование метода предобучения REBA позволило улучшить обобщающую способность глубокого автоэнкодера для случаев CD-1 и CD-10. На рисунках 3.2 и 3.3 изображены оригинальные данные, на которых производилось

обучение, и восстановленные из одного нелинейного компонента, используя данные тестирования. Как можно видеть, автоэнкодер восстанавливает данные из одного нелинейного компонента с хорошей точностью.



**Рисунок 3.2.** – 2D-изображения оригинальных и реконструированных данных



**Рисунок 3.3.** – 3D-изображение оригинальных и реконструированных данных

### 3.2 Распознавание образов

С практической точки зрения более полезным является проверка разработанных методов на более сложных выборках.

Задача распознавания образов является одной из основных в области компьютерного зрения. В настоящий момент «золотым стандартом» выборок, применяемых для оценки эффективности моделей выступают MNIST, CIFAR-10 и CIFAR-100.

Выполнение экспериментов с использованием выборок большого размера занимает достаточно продолжительное время. Эта проблема была решена проведением расчетов с использованием видеокарт nVidia, поддерживающих технологию CUDA (нами использовались ускорители A100 или V100, доступные при использовании сервиса Google Colab [56]). Существуют другие способы ускорения расчетов, производимых при обучении нейронной сети, например, использование вычислительных кластеров [57].

Помимо классического и предложенного методов предобучения нами был протестирован подход, при котором первый слой нейросетевой модели предобучается классическим методом, а все прочие слои, кроме последнего классифицирующего – с использованием предлагаемого подхода REBA. Будем обозначать такой вариант предобучения как гибридный (HREBA).

### 3.2.1 Критерий оценки результатов классификации

Для оценки качества решения задачи классификации нами применялся следующий подход. Вначале определялся  $k$ -тый нейрон, выходное значение которого было максимальным для заданного образа  $s$  (данное число соответствует метке класса, получаемого моделью):

$$k_s = \arg \max_j y_j^s \quad (3.1)$$

Затем получившееся значение сравнивалось с эталонными значениями и количество совпадений суммировалось для всех образов:

$$S = \sum_{s=1}^L I[k_s = e_s] \quad (3.2)$$

где  $I[]$  – индикаторная функция,  $L$  – общее количество образов из оцениваемого множества,  $e_s$  – эталонное значение, метка класса, соответствующая  $s$ -тому образу.

Значения индикаторной функции могут быть получены по следующей формуле:

$$I[x] = \begin{cases} 1, & \text{x is True} \\ 0, & \text{x is False} \end{cases}$$

Таким образом, общая эффективность на тестовой выборке может быть получена по следующей формуле:

$$\text{Efficiency} = \frac{S}{L} * 100\% \quad (3.3)$$

### 3.2.2 Описание выборок

Выборка MNIST (Mixed National Institute of Standards and Technology database) является классической при тестировании систем распознавания образов, а также широко используемой для обучения и тестирования алгоритмов машинного обучения. Она сформирована как подмножество более крупной оригинальной базы NIST [53]. Кроме этого, над оригинальными изображениями были выполнены определенные преобразования (изменение размера и центрирование).

База MNIST состоит из 60000 образов для обучения и 10000 образов для тестирования. Каждый образ представляет собой изображение цифры размером 28X28 пикселей в градациях серого цвета. На рис. 3.4 изображен фрагмент базы изображений MNIST с наиболее труднораспознаваемыми цифрами.

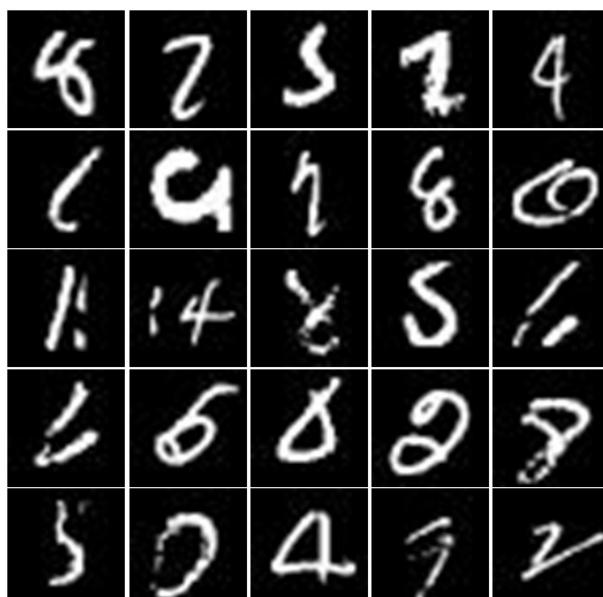
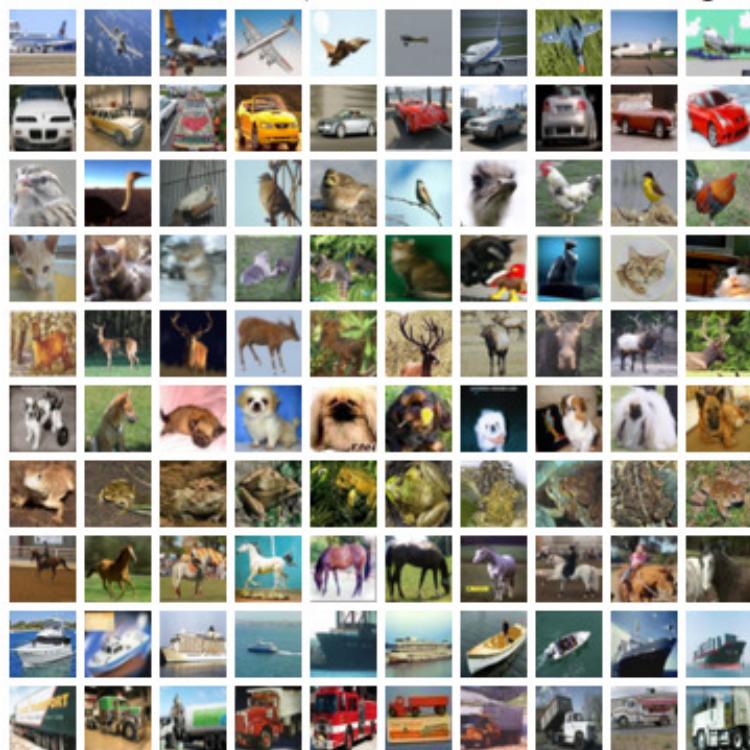


Рисунок 3.4. – Фрагмент базы изображений MNIST

Выборка CIFAR-10 [54] является подмножеством выборки Tiny Images [58] и включает в себя 60.000 цветных изображений технических средств и живых существ, принадлежащий 10 различным классам (рис. 3.5) по 6.000 изображений на каждый класс. Каждое изображение имеет размер 32X32 пикселя.

Выборка CIFAR-100 идентична выборке CIFAR-10 по параметрам изображений, которые в нее включены, но отличается количеством представленных классов изображений (в этой выборке общее количество классов 100). Общая размерность выборки составляет также 60.000 изображений, по 600 изображений на каждый класс.



**Рисунок 3.5. – Фрагмент базы изображений CIFAR-10**

Выборки CIFAR-10 и CIFAR-100 разделены авторами на обучающую и контрольную подвыборки (объемом 50.000 и 10.000 изображений соответственно).

### 3.2.3 Параметры вычислительного эксперимента и результаты

В качестве целевой модели для решения задачи распознавания изображений из выборки MNIST была выбрана сверточная нейронная сеть с параметрами, представленными в таблице 3.2.

Таблица 3.2. – MNIST: основные параметры используемой модели

Параметр	Значение
Архитектура	40X5X5 – 40X5X5 – 640X320 – 320X160 – 160X10
Функция активации	ReLU
Функция активации на последнем слое	Softmax
Начальная инициализация параметров	Нормальное распределение

Первые (сверточные) слои обозначены условно как  $K \times S \times S$ , где

$K$  обозначает количество ядер свертки в соответствующем слое, а  $S \times S$  – размерность ядра свертки.

Как видно из представленной выше таблицы, нами использовалась архитектура с 5 обрабатывающими слоями, 2 из которых сверточные, а 3 – полно связные. При этом общее число параметров модели составило 299.170.

Для того, чтобы создать методам максимально равноправные условия, мы использовали функцию ReLU на всех слоях сети, за исключением последнего слоя, на котором применялась softmax-функция. Использование ReLU позволило варианту обучения без предобучения начать процесс и завершить его с приемлемым значением эффективности.

В таблице 3.3 приведены основные используемые нами параметры обучения.

Таблица 3.3. – MNIST: основные параметры обучения

Этап	Параметр	Значение
Предобучение	Скорость обучения	0,000125
	Размер мини-батча	128
	Моментный параметр	[0,5; 0,9]
	Количество эпох обучения	30
Обучение	Скорость обучения	0,001
	Размер мини-батча	128
	Моментный параметр	0,9
	Количество эпох обучения	50

Эксперименты проводились с одной и той же начальной инициализацией параметров для всех методов серией в 10 попыток, получаемые результаты затем усреднялись. В ходе эксперимента нами сравнивалось четыре основных варианты обучения:

- 1) BP – обучение без предобучения;
- 2) REBA – обучение с предобучением (предлагаемый подход);
- 3) HREBA – обучение с предобучением (гибридный подход);
- 4) C-RBM – обучение с классическим методом предобучения).

В результате были получены показатели эффективности для выше перечисленных методов, представленные в таблице 3.4.

Как видно из представленных результатов, лучший средний показатель был достигнут гибридным методом HREBA, сочетающим в себе предобучение классическим и предложенным подходом, при этом максимальная эффективность была достигнута этим же методом и составила **99.53 %**.

Таблица 3.4. – MNIST: результаты обучения

Метод обучения	Эффективность, %
BP	99.367
REBA	99.371
HREBA	<b>99.458</b>
C-RBM	99.447

При проведении экспериментов с выборками CIFAR-10 и CIFAR-100 использовалась модель с архитектурой, представленной в таблице

Параметр	Значение
Архитектура	64X5X5 – 32X5X5 – 800X128 – 128X10
Функция активации	ReLU - Tanh - ReLU
Функция активации на последнем слое	Softmax
Начальная инициализация параметров	Нормальное распределение
Общее число параметров модели	159.914

### 3.3 Редуцирование параметров НС

Продемонстрируем эффективность предложенного подхода на примере редуцирования различных архитектур полносвязных нейронных сетей, применяемых для классификации изображений из выборок MNIST [53], CIFAR10 и CIFAR100 [54]. Данные выборки являются классическими для проверки эффективности моделей машинного обучения. Нами были проведены серии экспериментов, включающих различные используемые выборки, архитектуры и варианты предобучения. В рамках одной выборки и архитектуры НС текущая инициализация параметров сохранялась для возможности сравнения эффективности различных вариантов предобучающей процедуры. Ниже для рассматриваемых выборок приведены основные параметры, включающие скорость обучения, размер мини-батча, моментный параметр и количество эпох для предобучения и дообучения моделей (табл. 3.5).

В результате вычислительного эксперимента были получены результаты для различных выборок, архитектур НС и значений параметра реду-

Таблица 3.5. – Основные параметры обучения

Этап	Параметр	Значение
Обучение	Скорость обучения	0.05-0.1
	Размер мини-батча	100
	Моментный параметр	0.9
	Количество эпох обучения	50-100
Предобучение	Скорость обучения	0.05-0.2
	Размер мини-батча	32-100
	Моментный параметр	[0.5, 0.9]
	Количество эпох обучения	10

цирования  $t$  (табл. 3.6-3.10).

Таблица 3.6. – 784-800-800-10, MNIST

Тип	Эффективность, %, C-RBM / REBA	Количество параметров, C-RBM / REBA	Редуцировано параметров, %, C-RBM / REBA
без редуц.	<b>98.63</b> / 98.33	1276810 / 1276810	0/0
$t=0.2$	<b>98.61</b> / 98.27	<b>233760</b> / 279635	<b>81.69</b> / 78.1
$t=0.5$	98.03 / <b>98.05</b>	<b>32524</b> / 32817	<b>97.45</b> / 97.43
$t=0.8$	<b>97.1</b> / 96.48	17061 / <b>12217</b>	98.66 / <b>99.04</b>

Таблица 3.7. – 784-1600-1600-800-800-10, MNIST

Тип	Эффективность, %, C-RBM / REBA	Количество параметров, C-RBM / REBA	Редуцировано параметров, %, C-RBM / REBA
wr	<b>98.76</b> / 98.37	5747210 / 5747210	0/0
$t=0.2$	98.51 / <b>98.55</b>	<b>710734</b> / 781103	<b>87.63</b> / 86.41
$t=0.5$	98.01 / <b>98.03</b>	54709 / <b>43867</b>	99.05 / <b>99.24</b>
$t=0.8$	<b>96.9</b> / 93.08	25385 / <b>14914</b>	99.56 / <b>99.74</b>

Таблица 3.8. – 3072-1024-512-256-128-64-10, CIFAR10

Тип	Эффективность, %, C-RBM / REBA	Количество параметров, C-RBM / REBA	Редуцировано параметров, %, C-RBM / REBA
wr	<b>58.56</b> / 55.85	3844682 / 3844682	0/0
$t=0.2$	<b>58.69</b> / 54.37	<b>409211</b> / <b>227072</b>	89.36 / <b>94.09</b>
$t=0.5$	<b>42.08</b> / 41.2	29033 / <b>11320</b>	99.24 / <b>99.71</b>
$t=0.8$	<b>23.02</b> / 10.0	10058 / <b>4886</b>	99.74 / <b>99.87</b>

Как видно из приведенных результатов, исследуемые архитектуры в целом сохраняют обобщающую способность, будучи редуцированными

Таблица 3.9. – 3072-512-256-128-64-10, CIFAR10

Тип	Эффективность, %, C-RBM / REBA	Количество параметров, C-RBM / REBA	Редуцировано параметров, %, C-RBM / REBA
wr	<b>57.28</b> / 53.69	1746506 / 1746506	0/0
t=0.2	<b>56.83</b> / 41.72	220037 / <b>126846</b>	87.40 / <b>92.73</b>
t=0.5	<b>45.29</b> / 44.93	20431 / <b>11383</b>	98.83 / <b>99.35</b>
t=0.8	10.0 / 10.0	8599 / 3797	99.51 / 99.78

Таблица 3.10. – 3072-3072-1024-512-256-128-64-100, CIFAR100

Тип	Эффективность, %, C-RBM / REBA	Количество параметров, C-RBM / REBA	Редуцировано параметров, %, C-RBM / REBA
wr	20.84 / <b>21.63</b>	13290788 / 13290788	0/0
t=0.2	20.77 / <b>21.01</b>	1304525 / <b>703319</b>	90.18 / <b>94.71</b>
t=0.5	<b>13.4</b> / 1.0	49847 / <b>24636</b>	99.62 / <b>99.81</b>
t=0.8	<b>2.67</b> / 1.0	21329 / <b>16977</b>	99.84 / <b>99.87</b>

более чем на 80 процентов и даже при большей степени редуцированности демонстрируют неплохие свойства.

Также можно заметить, что чем больше настраиваемых параметров в модели, тем эффективней осуществляется редуцирование. Однако с увеличением параметра редуцирования эффективность исходной сети постепенно снижается, так как редуцированию начинают подвергаться параметры, наличие которых влияет на итоговый результат.

Полученные результаты обосновывают возможность предобучения глубокой нейронной сети с использованием неконтролируемой процедуры без получения эффекта переобучения и снижения эффективности модели, так как в процессе предобучения фактически снижается влияние определенных параметров модели на итоговую выходную активность сети. Такие параметры имеют “паразитический” характер и фактически являются фактором переобучения модели. На этапе дообучения модели они не модифицируются и могут быть удалены после этапа предобучения.

Помимо редуцирования параметров, было выполнено так называемое архитектурное редуцирование (шаг 3 алгоритма), на котором производилось упрощение структуры слоев моделей за счет исключения нейронов с нулевыми векторами связей.

В следующих таблицах (3.11, 3.12) приведены результаты архитектурного редуцирования для двух различных архитектур и обучающей выборки MNIST. Для предобучения использовался подход REBA.

Из приведенных таблиц можно заметить, что для моделей с большим

Таблица 3.11. – 784-800-800-10, MNIST

Параметр	Исходная	Редуцированная
t=0.2	784-800-800-10	784-800-556-10
t=0.5	-//-	784-710-422-10
t=0.9	-//-	784-91-114-10

Таблица 3.12. – 784-1600-1600-800-800-10, MNIST

Параметр	Исходная	Редуцированная
t=0.2	784-1600-1600-800-800-10	784-889-192-686-221-10
t=0.5	-//-	784-464-157-567-182-10
t=0.9	-//-	784-17-101-118-50-10

количеством слоев и нейронов в каждом слое вероятность архитектурного редуцирования выше и оно начинает происходить уже для сравнительно небольших значений параметра редуцирования  $t$ , в том время как для более «поверхностных» моделей оно существеннее проявляется только для больших значений параметра. Стоит отметить, что архитектуры, которые сохраняют почти все нейроны (например, 3.11, архитектура при  $t=0.2$ ), все же при этом теряют для каждого нейрона основную часть связей (сохранение нейрона в данном случае будет иметь место, даже при сохранении одной связи нейрона с предыдущим слоем).

### 3.4 Выводы

- 1) Проведен сравнительный анализ методов предобучения ГНС для задачи сжатия данных с использованием автоэнкодерной модели НС.
- 2) Проведен сравнительный анализ методов предобучения ГНС для задачи распознавания образов с использованием выборок MNIST, CIFAR-10 и CIFAR-100. Предложена модификация метода предобучения.
- 3) Проведены экспериментальные исследования редуцирования параметров глубокой нейронной сети с использованием различных методов предобучения и архитектур ГНС.

# ГЛАВА 4

## РАЗРАБОТКА СИСТЕМ КОМПЬЮТЕРНОГО ЗРЕНИЯ НА ОСНОВЕ ГЛУБОКИХ НЕЙРОННЫХ СЕТЕЙ

Научные и практические результаты, полученные в рамках данной диссертационной работы, внедрены и используются на следующих предприятиях Республики Беларусь:

- ОАО «Савушкин продукт» (г. Брест) при разработке системы автоматического контроля качества нанесения маркировки продукции;
- «Intelligent Semantic Systems» (г. Минск) при разработке модуля компьютерного зрения для осуществления распознавания лиц пользователей и оценки их эмоционального состояния;
- в учебном процессе учреждения образования «Брестский государственный технический университет».

Сведения об использовании результатов диссертационной работы отражены в соответствующих актах внедрения, приведенных в приложении Б к диссертации.

### **4.1 Автоматический контроль качества нанесения маркировки продукции**

#### **4.1.1 Постановка задачи и обзор существующих решений**

В основе конкурентности любого производства лежит выпуск разнообразной и качественной продукции. Качество является определяющим критерием при выборе клиентом того или иного продукта, а разнообразие позволяет охватить различные группы потенциальных покупателей. Системы, которые автоматизируют процесс проверки качества и при этом поддерживают многообразие продукции, выпускаемой на большом предприятии, имеют особую ценность. При этом процесс проверки качества осуществляется не только для самого продукта, но и для той упаковки и маркировки, которую видит покупатель. Так как упаковка формирует первоначальное впечатление о товаре, ее качество является одной из причин, которая дает покупателю основание для покупки. Маркировка как элемент упаковки, гарантирует покупателю сохранность продукции в течение указанного срока при соблюдении условий хранения.

В последнее время методы искусственного интеллекта в целом и машинного обучения в частности широко используются в промышленных системах, устанавливаемых на предприятиях для контроля за процессом производства. Интеллектуальные подсистемы позволяют упростить многие рутинные операции, проводимые для поддержания качества готовой продукции. Например, контроль за правильностью нанесения маркировки ранее производился исключительно оператором-человеком. Сейчас, с развитием теории компьютерного зрения, получившей значительный рывок благодаря постоянно развивающейся области обучения глубоких нейронных сетей, становится особенно актуальной разработка систем, позволяющих осуществлять рутинные операции быстрее и регулярнее, чем это мог бы сделать человек.

Поставленная перед нами задача заключалась в разработке системы проверки корректности нанесения маркировки на продукцию, производимую ОАО «Савушкин продукт». Важным аспектом является то, что распознавание должно выполняться в реальном времени, основываясь на данных, поступающих с камеры, установленной над производственной линией.

Помимо буквенно-цифрового кода (рис. 4.1), начиная с недавнего времени, продукция может выпускаться с вариантами маркировки, включающей код Data Matrix (рис. 4.2) [59]. Данный тип маркировки является удобным и емким представлением специальных и общих данных о продукте.



**Рисунок 4.1. – Продукт с буквенно-цифровой маркировкой**

Основываясь на общей постановке задачи, были выделены следующие подзадачи, которые должна решать система:

- 1) обнаружение и распознавание типа маркировки;
- 2) распознавание маркировки;
- 3) выявление возможных проблем с маркировкой.



Рисунок 4.2. – Продукт с кодом Data Matrix

Перечислим основные возникающие на производстве проблемы с маркировкой [26–А]:

- 1) **отсутствие чернил:** в случае поступления на конвейер продуктов без маркировки, система должна сделать вывод об отсутствии чернил, дополнительно запросив проверку, обратившись к принтеру;
- 2) **сдвиг камеры:** если от нейросетевых модулей не поступают данные о результатах распознавания, но системе известно, что движение по конвейеру началось, то она должна сделать вывод о сдвиге камеры;
- 3) **ошибочная маркировка:** маркировка была обнаружена и распознана, но не совпала с эталонным представлением. В этом случае должен быть сделан вывод о том, что маркировка неверная;
- 4) **нечитаемая маркировка:** в случае, если маркировка получается смазанной и не может быть распознана, необходимо остановить конвейер и сообщить об ошибке оператору.

Для 1,3 и 4 проблемы необходимо выполнить отсев продуктов, которые имеют “проблемную” маркировку. Возникновение этих проблем предполагает полную остановку движения конвейера и сообщение оператору о возникшей проблеме.

Следует отметить, что в процессе решения задачи распознавания маркировки возникают следующие проблемы, влияющие на процесс решения и подбор алгоритмов:

– **Высокая скорость видеопотока.** Так как скорость видеопотока составляет 76 кадров в секунду, то время обработки каждого кадра составляет около 13 миллисекунд. Следует отметить, что этого времени недостаточно для запуска сложной нейросетевой архитектуры.

– **Невозможность корректной прямой детекции цифр.** Помимо цифр, содержащихся непосредственно в маркировке, в кадр могут попадать цифры, нанесенные на другие объекты, например, на сам конвейер или его части. Помимо этого, следует отметить, что изображение попадает на

нейронную сеть с уменьшенным разрешением, что приводит к сложности распознавания очень мелких объектов (цифр).

– **Возможность изменения ориентации маркировки.** В процессе движения товара по производственной линии возможен поворот маркировки на произвольный угол, что приводит к существенному ухудшению распознавания.

Таким образом, резюмируя все вышеперечисленные задачи и возникающие проблемы, перечислим основные требования, которым должна удовлетворять разрабатываемая система:

– **Высокая скорость работы.** Конвейер движется очень быстро, поэтому обнаружение брака должно осуществляться с минимальными задержками;

– **Автономность.** Система должна минимизировать участие оператора в процессе контроля качества;

– **Универсальность.** Система должна настраиваться на распознавание маркировки любой продукции;

– **Адаптируемость.** Система должна работать при любых условиях, возникающих на производстве (например, недостаточность освещения, ошибки персонала и т.д.).

Несмотря на существующий интерес к автономизации производственных процессов и те неоспоримые преимущества, которые влечет ее внедрение, задачи, подобные описанной, решаются в большом числе случаев с участием человека. Оператор просто периодически выборочно проверяет часть продукции. У такого подхода есть недостатки:

– контроль проходит малая часть продукции, таким образом есть вероятность, что дефектная маркировка будет пропущена;

– скорость реакции человека на возникающую нештатную ситуацию может быть недостаточной;

– человек может не заметить небольшое расхождение проверяемой маркировки с эталонной;

– работа по ручной проверке является монотонной.

Существующие разработки базируются на аппаратных решениях, например, на использовании специальных датчиков [60].

Такие решения осуществляют распознавание маркировки, но имеют ряд важных недостатков:

– Нестабильное качество распознавания, зависящее от условий, при которых производится съемка (в частности, от освещенности). Так как производственная линия движется быстро, то необходимые условия для качественного распознавания чаще всего не соблюдаются;

- Необходимость покупки специализированного программного обеспечения для настройки датчиков.

Таким образом, подобные решения создают дополнительные сложности при эксплуатации, которые проявляются в необходимости помимо выборочного ручного контроля качества продукции осуществлять также контроль за функционированием самой системы распознавания.

#### 4.1.2 Предлагаемый подход

Предлагаемый подход состоит в использовании конвейерной структуры из отдельных нейросетевых модулей, каждый из которых решает собственную подзадачу распознавания маркировки.

Задачей данного конвейера является обнаружение маркировки, определение ее типа и ее распознавание.

Остановимся далее подробнее на архитектуре системы.

Осуществляя декомпозицию решаемой задачи можно выделить следующие подзадачи:

- 1) Оценка положения товара;
- 2) Детекция товара и маркировки;
- 3) Определение типа маркировки;
- 4) Поворот маркировки для горизонтальной ориентации;
- 5) Распознавание маркировки (осуществляется по-разному, в зависимости от типа);
- 6) «Сборка» маркировки: формирование выходной информации (даты производства товара, номера в партии и т.д.);
- 7) Проверка распознанной маркировки (определение корректности распознанных данных в соответствии с заранее заданным шаблоном).

Применение предпоследнего этапа в большей степени относится к одному из типов маркировки – алфавитно-цифровому.

Помимо указанных, в процессе распознавания маркировки решаются дополнительные задачи, связанные с корректностью нанесения такой маркировки.

- 1) определение отсутствия маркировки на продукте;
- 2) определение наличия искажений маркировки, возникающих в процессе печати, отсутствия ее частей и т.д.

Указанные задачи решаются в процессе выполнения основных этапов распознавания.

Проблемы, указанные в постановке задачи, могут быть решены архитектурно.

1) Высокая скорость видеопотока: решается пропуском незначащих кадров, в которых товар находится не в середине кадра. Это позволяет увеличить интервал времени, необходимый для обработки изображения нейросетью. Таким образом, необходимо оценить значимость кадра. Это можно сделать более простой моделью с малым временем обработки;

2) Невозможность корректной прямой детекции цифр: решается осуществлением декомпозиции задачи обнаружения на отдельные подзадачи. Например, в начале обнаруживается товар, затем маркировка на товаре и, наконец, отдельные цифры;

3) Возможность изменения ориентации маркировки: решается оценкой угла поворота маркировки для представления ее в максимально горизонтальном положении.

Наличие нескольких подзадач предполагает использование группы моделей, каждая из которых выполняет свою часть работы.

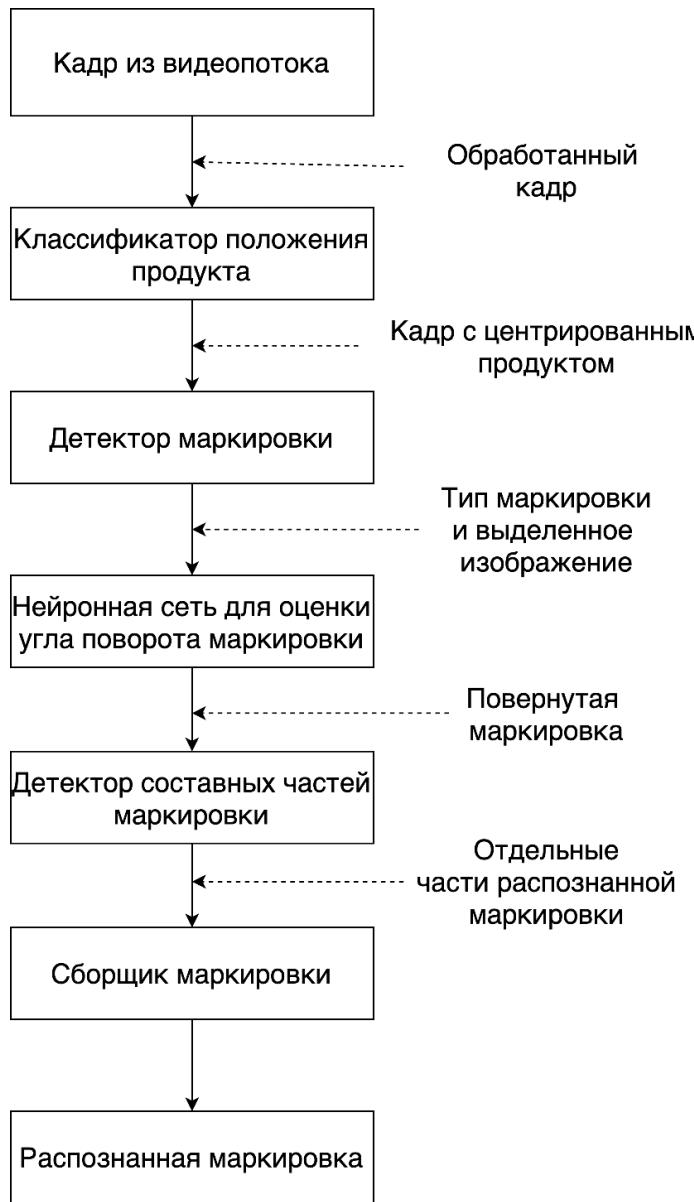
Следует отметить, что в предлагаемой нейросетевой системе практически для каждой подзадачи используется собственная нейросетевая модель. Это позволяет легко модифицировать систему, улучшать отдельные модули, изменять их, а также добавлять новые.

Архитектура готового модуля распознавания представлена на рис. 4.3.

Опишем основные применяемые модули и их роль в общей архитектуре.

**Классификатор положения продукта.** Первая модуль реализован на основе сверточного классификатора, который определяет значимость кадра для последующего анализа. При этом наиболее значимым является кадр, в котором товар находится ближе всего к центру кадра (рис. 4.4). Было определено четыре основных класса позиций, по мере удаленности продукта от центра кадра. 1 класс описывает минимальную удаленность. Только кадры этого класса участвуют в последующих этапах обработки и анализа. Классы 2 и 3 описывают среднюю и максимальную удаленность. Класс 4 используется для случая отсутствия продукта в кадре (пустая линия).

Архитектура применяемого классификатора представлена на рис. 4.5. Он состоит из 5 слоев и имеет 4 выходных нейрона по числу классов, определяющих положение товара в кадре. На всех слоях используется функция активации ReLU за исключением 3-го и последнего слоев. Они используют линейную и softmax-функции активации соответственно. Также применяется max pooling после первого и второго сверточного слоев с параметром stride, равным 2.



**Рисунок 4.3. – Структура модуля распознавания маркировки**

В случае, если кадр был отнесен к классу 1, он передается далее на следующую модель.

**Модуль детекции маркировки.** В составе данного модуля используется нейросетевой детектор, осуществляющий поиск товара и маркировки в кадре. Здесь в качестве архитектуры была выбрана сеть SSD [61] на базе классификатора MobileNet v1 [62].

Независимое обнаружение товара и маркировки позволяет идентифицировать ситуацию с отсутствующей маркировкой автоматически. Для этого достаточно проверить логическое условие отсутствия маркировки при наличии самого товара в кадре. При выявлении дефекта, система оповещает об этом оператора.

Следует отметить, что данный детектор применяется для обнаружения разных типов маркировки. Так как модель SSD может использоваться



Рисунок 4.4. – Примеры изображений разных классов по степени удаленности объекта от центра кадра

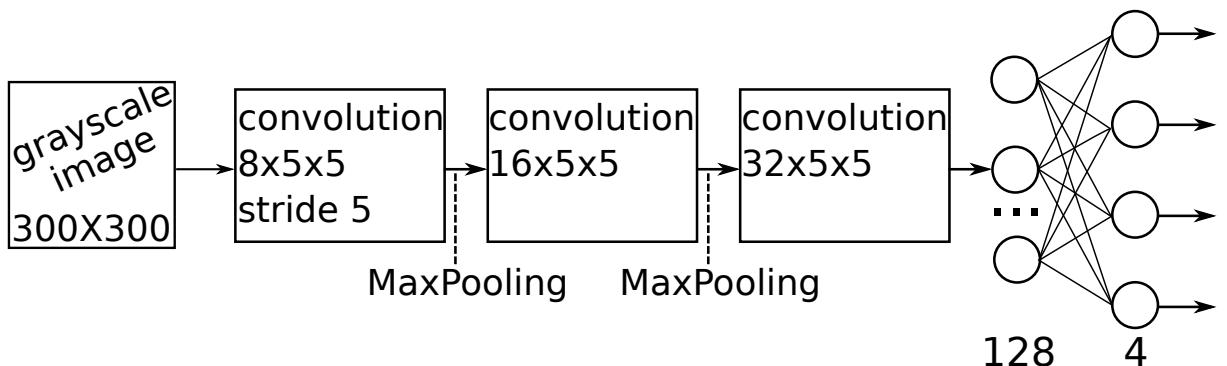


Рисунок 4.5. – Структура классификатора для оценки положения бутылки  
[26–A]

для обнаружения объектов разных классов, было принято решение использовать ее и для определения типа маркировок, которое не требует каких-либо специальных процедур анализа.

В итоге, если дефект маркировки не был обнаружен, выполняется передача изображения маркировки (в оригинальном размере) и информации о ее типе на следующую модель.

**Модуль оценки угла поворота.** В реализации третьего модуля используется регрессор, применяемый для оценки угла поворота маркировки. Данная модель возвращает угол, на который должна быть повернута маркировка для достижения горизонтальной ориентации изображения. Такое преобразование позволяет улучшить качество последующего распознавания цифр. После поворота выполняется передача изображения маркировки на модель для анализа соответствующего типа маркировки (в нашей реализации это модели для анализа кода Data Matrix или цифровой метки).

**Детектор составных частей.** Для анализа цифрового кода также используется детектор SSD-MobileNet v1, который обнаруживает отдельные цифры в маркировке. Для анализа кода Data Matrix в текущей реализации используется не нейросетевая модель. Применимость нейронной сети для подобного анализа может быть предметом дальнейших исследований.

Далее осуществляется «сборка» распознанной маркировки и обработка результата.

Принцип работы нейросетевого компонента на примере буквенно-цифровой маркировки продемонстрирован на рис. 4.6.

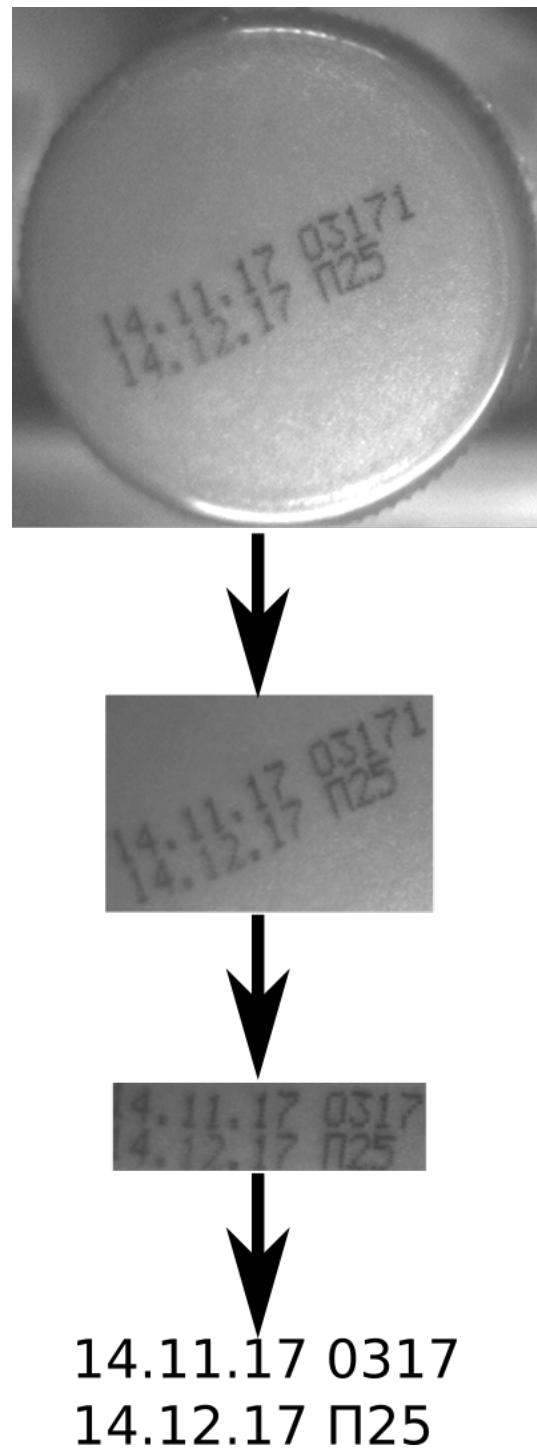


Рисунок 4.6. – Принцип работы системы

#### 4.1.3 Обучающие выборки: формирование и основные характеристики

В процессе подготовки нейросетевых моделей нами использовались различные обучающие выборки:

- Выборка для обучения классификатора
- Выборка для обучения детектора маркировок и товаров (определения типа маркировки)
- Выборка для обучения регрессора угла поворота
- Выборка для обучения детектора цифр

**Выборка для обучения классификатора.** Для создания выборки использовалась модель Faster R-CNN [63] (на базе предобученного классификатора ResNet50 [64]). Данная модель обладает лучшими показателями эффективности, чем SSD-MobileNet, но уступает ей в скорости. Она использовалась для автоматической разметки имеющихся данных (главным образом видеофайлов производственного процесса) по степени удаленности товара от центра кадра. В качестве метрики применялось евклидово расстояние от центра продукта до центра кадра. Таким образом были сформированы четыре класса изображений, которые использовались для последующего обучения сверточного классификатора. Общий объем выборки составил 6189 изображений, 1303 из которых составили контрольную выборку.

**Выборка для обучения детектора маркировок и товаров (определения типа маркировки).** Для формирования этой выборки мы использовали размеченные вручную изображения из выборки для обучения классификатора и дополнительные изображения сгенерированных кодов Data Matrix. Общий объем выборки составил 815 изображений, 163 из которых составили контрольную выборку.

**Выборка для обучения регрессора угла поворота.** При создании данной выборки использовались изображения маркировок, повернутые под произвольными углами. Общий объем выборки составил 59385 изображений, 11877 из которых составили контрольную выборку. В качестве основы брались изображения маркировок, полученные из выборки для обучения детектора маркировок и товаров.

**Выборка для обучения детектора цифр.** Для создания этой выборки применялся датасет номеров домов SVHN [65], а также размеченные цифровые маркировки. Использовался вариант выборки SVHN, включающий 33402 изображения в обучающей части и 13068 в контрольной (рис. 4.7). Объем выборки размеченных цифровых маркировок составил

419 изображений.



Рисунок 4.7. – Пример изображений из выборки SVHN

#### 4.1.4 Результаты обучения и тестирования

После обучения классификатора 1 итоговая точность распознавания составила **93.27%**.

Оба детектора (продуктов/маркировок и отдельных цифр) обучались на основе предобученных моделей (использовалось предобучение II типа).

Применение SSD-модели позволяет достичь эффективности детекции в **99% (mAP = 0.99)** для обнаружения товара и маркировки и **92% (mAP = 0.92)** для отдельных цифр. Кроме этого, скорость обработки позволяет успешно обнаруживать маркировку в видеопотоке со скоростью 76 кадров в секунду. Результаты эффективности распознавания отдельных цифр представлены в таблице 4.1.

Таблица 4.1. – Эффективность обнаружения отдельных классов цифр

Class label	AP
0	0.9218
1.	0.9107
2.	0.9354
3.	0.9286
4.	0.9265
5.	0.9137
6.	0.9274
7.	0.9167
8.	0.9646
9.	0.8975
<b>mAP</b>	<b>0.92429</b>

Результаты обнаружения детектора маркировок и детектора отдельных цифр изображены на рис. 4.8 и 4.9.

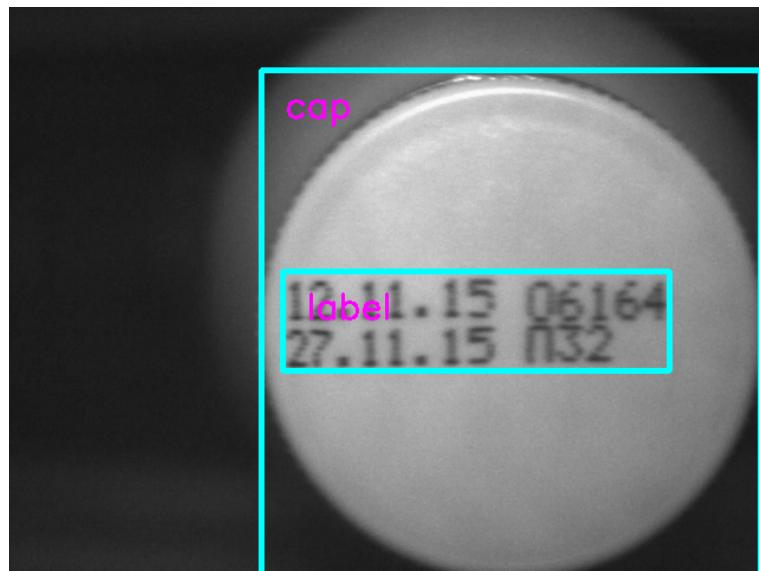


Рисунок 4.8. – Обнаруженный продукт и маркировка

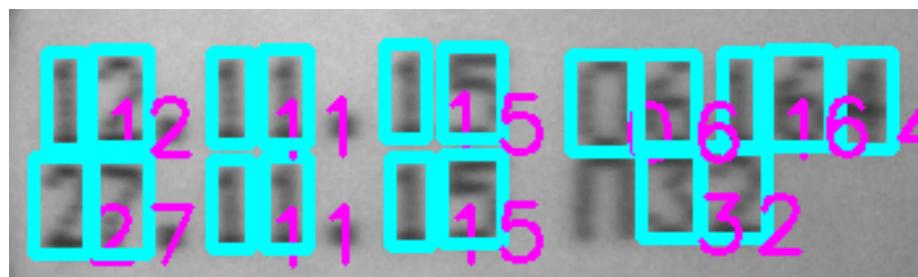


Рисунок 4.9. – Обнаруженные цифры в маркировке

## 4.2 Обнаружение солнечных панелей на аэрофотоснимках

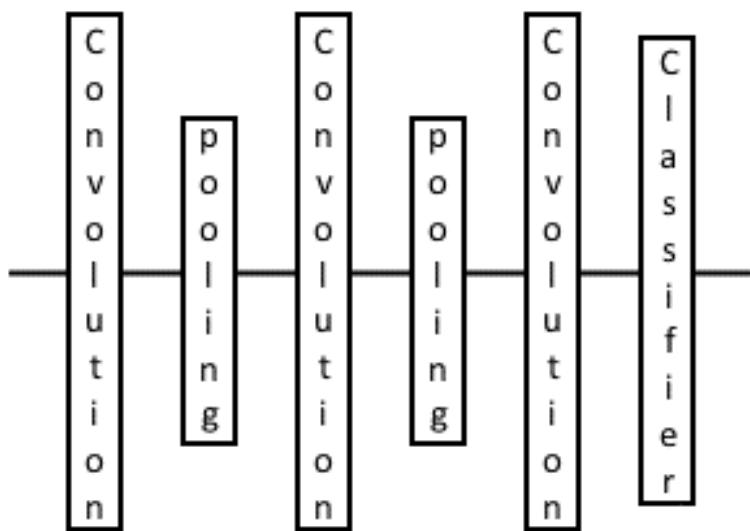
### 4.2.1 Постановка задачи и обзор существующих решений

По причине увеличивающейся доли использования солнечной энергии, спрос на фотоэлектрические элементы в мире постоянно растет. С начала 2010-х годов индустрия производства и поддержки солнечных панелей переживает стремительный рост. Так, по состоянию на 2021 год только в США в данной отрасли работает более 10000 компаний [66].

В связи с растущей популярностью этой технологии проблемы, связанные с обслуживанием солнечных панелей, также становятся актуальными. Многие сервисные компании заинтересованы в получении информации о потенциальных клиентах. Таким образом, анализ фотоснимков с целью обнаружения солнечных панелей и накопления статистической информации является актуальной задачей.

Нами решалась задача обнаружения солнечных панелей на изображении с получением точных координат местонахождения с использованием глубоких сверточных нейронных сетей.

Сверточные нейронные сети базируются на моделях неокогнитрона, предложенной К. Фуксисом [9], и сетях с разделяемыми весами. Первая классическая модель сверточной нейронной сети была разработана ЛеКуном и получила название LeNet-5 [67]. Сверточные нейросети интегрировали три концепции, называемые локальным рецептивным полем, разделяемыми весовыми коэффициентами и пространственной подвыборкой. Используя локальное рецептивное поле, нейронные элементы первого сверточного слоя могут извлекать простейшие признаки, такие как края, углы и т.д. Сверточная нейронная сеть представляет собой комбинацию сверточных и подвыборочных (pooling) слоев (рис. 4.10), благодаря чему осуществляет нелинейное иерархическое преобразование входной информации. Последний блок сверточной сети представляет собой многослойный персептрон, SVM или любой другой классификатор.



**Рисунок 4.10. – Общий вид сверточной нейронной сети**

В статье Дж. Малофа [68] был описан подход, базирующийся на использовании SVM для автоматического обнаружения солнечных панелей, используя фотографии высокого разрешения, полученные с помощью спутника. В этом подходе сначала применяется операция предварительного скрининга, которая идентифицирует регионы, которые затем обрабатываются с целью выделения признаков. В качестве выходной информации, получаемой с помощью данной модели, выступает список регионов и доверительные значения, показывающие, насколько вероятно наличие солнечной панели в заданной области. Общая эффективность подхода опре-

деления наличия панелей в [68] составляет порядка 94%. Однако, нужно отметить, что данный метод дает лишь приблизительную информацию о местонахождении панели. Оценки точности локализации панелей в работе не приводится.

В [69] используется подход, базирующийся на применении деревьев решений. Достигнутый показатель локализации панелей составил 90% в случае использования определенных параметров алгоритма, а сам метод состоит из четырех стадий. В работе в качестве обучающей выборки используется Aerial Imagery Dataset, включающий изображения разрешением до 5000 X 5000 пикселей [70].

#### 4.2.2 Предлагаемый подход

Предложенный нами нейросетевой алгоритм для обнаружения солнечных панелей отличается тем, что в качестве базовой модели используется сверточная нейронная сеть, которая показывает лучшие результаты при решении задач классификации и детекции объектов на изображениях. Другое важное отличие заключается в использовании фотографий низкого разрешения для обучения нашей модели (например, фото из Google Maps, нередко неудовлетворительного качества, см. рис. 4.11).

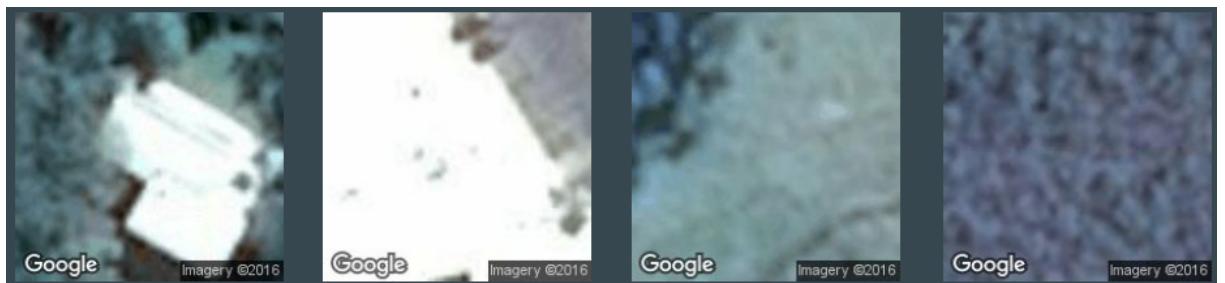


Рисунок 4.11. – Примеры изображений низкого качества из обучающей выборки

Предлагаемый алгоритм обнаружения панелей включает два основных этапа, на каждом из которых используется предобученная глубокая нейронная сеть:

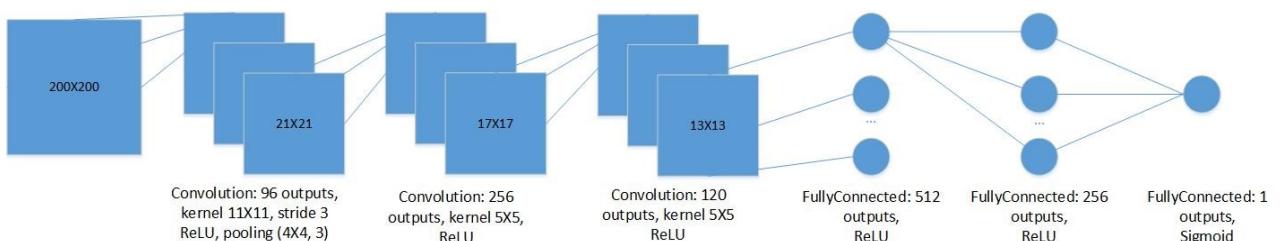
- 1) Оценка наличия солнечной панели на аэрофотоснимке
- 2) Локализация солнечной панели

Использование двухэтапности в решении поставленной задачи позволяет существенно ускорить обработку изображений, большую часть которых занимают объекты, отличные от искомых (солнечных панелей), т.к. нейросетевые модели, используемые на этапе локализации, как правило, более ресурсоемки. Таким образом, модель первого этапа выполняет роль

фильтра для последующей обработки. Тем самым предлагаемый алгоритм можно применять для последовательной обработки больших изображений.

**Оценка наличия солнечной панели.** Для первого этапа в качестве базовой модели для решения поставленной задачи мы использовали сверточную нейронную сеть с архитектурой, представленной на рис. 4.12 [35].

Представленная нейронная сеть состоит из шести слоев. Первые три слоя являются сверточными и выполняют выделение высокогуровневых признаков. Последние три слоя представляют собой полно связные слои нейронных элементов, решающие задачу классификации. Для всех слоев, кроме последнего, используется функция активации ReLU 4.1.



**Рисунок 4.12.** – Архитектура используемой сверточной нейронной сети

$$f(x) = \max(0, x) \quad (4.1)$$

В последнем полно связном слое используется сигмоидная функция активации 4.2.

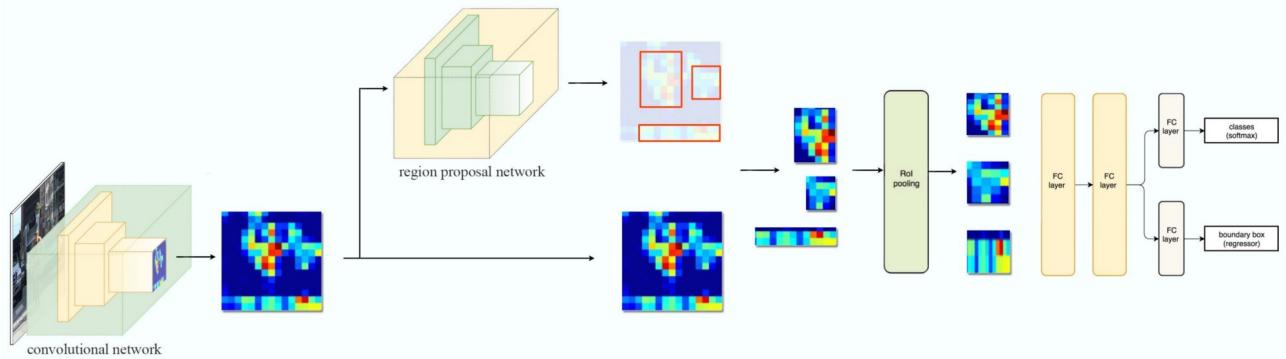
$$f(x) = \frac{1}{1 + e^{-x}} \quad (4.2)$$

Дополнительно, на первом сверточном слое мы используем шаг, равный 3, с целью уменьшения размерности получаемых со сверточного слоя карт признаков.

Изображения из обучающей выборки подавались на первый слой сверточной сети и передавались вплоть до последнего слоя. Последний полно связный слой содержит только один нейрон, который возвращает вероятность наличия солнечной панели на изображении. Для обучения сверточной нейронной сети использовался метод стохастического градиентного спуска.

**Локализация солнечной панели.** Для решения задачи локализации нами применялась глубокая нейронная сеть Faster-RCNN (Faster Region-based Convolutional Neural Network), базирующаяся на классификаторе ResNet-50 (4.13). Данная модель состоит из двух частей. Первая часть – это классификатор ResNet-50, предобученный на выборке COCO

[71]. Вторая часть – это детектор, который представлен сверточной нейронной сетью, генерирующей координаты прямоугольных областей, содержащих в себе искомые объекты, и метки класса для каждой области.



**Рисунок 4.13. – Глубокая сверточная сеть Faster-RCNN**

Подобная архитектура обеспечивает получение высоких показателей эффективности при решении задач детекции объектов в системах, где общее время, затраченное на обработку и вывод результатов, некритично.

#### 4.2.3 Обучающая выборка

В качестве исходных данных, используемых для формирования обучающей выборки для решения задачи обнаружения солнечных панелей, нами использовались цветные изображения Google Maps с разрешением 200 X 200 пикселей (рис. 4.14).



**Рисунок 4.14. – Примеры изображений из выборки Google Maps**

Для обучения модели первого этапа использовалась выборка из 3347 фотографий (где 1643 изображения содержали солнечные панели, а 1704 –

не содержали), при этом 80% исходной выборки формировали обучающую выборку, а 20% - тестовую. Для обучения модели второго этапа использовалась выборка из 1000 изображений, 800 из которых было отнесено к обучающей, а 200 – к тестовой выборкам.

Подготовка обучающей выборки состояла в ручном переборе изображений и отнесении их к двум группам изображений – содержащих солнечные панели и, соответственно, не содержащих. При подготовке выборки для локализации панелей также использовался ручной перебор изображений с определением для каждого характеристики прямоугольных областей, включающих солнечные панели (длина, ширина, координаты левого верхнего угла). Интересующих областей на изображении может быть несколько (для случая множественных панелей).

Выделение прямоугольных областей для некоторых изображений может не выглядеть целесообразным, например, случаи, когда панели расположены под углом к горизонтальной оси снимка и имеют вытянутую форму (рис. 4.15). Однако, для таких фотографий удается получить приемлемые результаты локализации на тестовых данных, несмотря на то, что большая часть области включает фоновое изображение.



Рисунок 4.15. – Пример изображения с преобладающим фоном

#### 4.2.4 Оценка качества обученных моделей

Для оценки качества применялись две основные метрики.

Первая метрика использовалась для оценки классификатора. В режиме тестирования модели нами применялось простое преобразование выходных данных – пороговая функция вида:

$$b_s = I[y_s > 0, 5]$$

где  $y_s$  представляет реальное выходное значение CNN-сети, возвращаемое для  $s$ -го входного изображения выборки,  $b_s$  – бинаризованная форма, полученная вычислением индикаторной функции вида:

$$I[x] = \begin{cases} 1, & \text{x is True} \\ 0, & \text{x is False} \end{cases}$$

Метрика оценки рассчитывалась по формулам:

$$A = \frac{S}{L} * 100\%$$

$$S = \sum_{s=1}^L I[b_s = e_s]$$

где  $e_s$  – эталонное значение (метка),  $L$  – объем выборки. Таким образом, оценивается доля верных ответов, возвращаемых нейронной сетью.

Для оценки эффективности модели локализации использовалась метрика mAP.

Метрика mAP является стандартом де-факто метрик, используемых для оценки качества моделей, применяемых для детекции. Эта метрика используется вместе со своими модификациями, вычисленными для различных значений порога IoU. Так как для рассматриваемой задачи существует только один класс объектов, метрика mAP совпадает с AP.

Как известно, точность вычисляется по формуле:

$$P = \frac{TP}{TP + FP}$$

где  $TP$  и  $FP$  обозначают соответственно число истинно-положительных и ложно-положительных результатов детекции, и, соответственно,  $P$  определяет долю корректных детекций в общем числе детекций, полученных моделью.

Значение  $TP$  определяет общее количество прямоугольных областей, для которых величина  $IoU$ , вычисленная относительно истинных областей (*Ground-true box*), больше некоторого заданного порога (чаще всего выбирается порог 0,5). Таким образом, если величина  $IoU$  для такой спрогнозированной области превысила 0,5, то детекция рассматривается как истинно-положительная. Если детекций для данной истинной области

несколько, то выбирается одна детекция с самым большим значением  $IoU$ , а остальные рассматриваются как  $FP$ .

Усредненное значение по всем изображениям выборки дает величину АР:

$$AP = \frac{1}{L} \sum_{i=1}^L \frac{TP_i}{TP_i + FP_i}$$

где  $L$  – число изображений в выборке.

#### 4.2.5 Результаты обучения и тестирования

Сверточная сеть для определения наличия солнечных панелей обучалась в течение 70 эпох, используя следующие параметры:

- скорость обучения – 0,001;
- моментный параметр – 0,9;
- weight-decay – 0,0005;
- размер мини-батча – 20.
- вероятность применения dropout для полносвязных слоев – 0,5

Точность классификации для данной задачи, составила порядка **87,46%** (см. таблица 4.2).

Таблица 4.2. – Матрица ошибок

	Спрогнозировано «No»	Спрогнозировано «Yes»	Точность, %
Ожидалось «No»	325	32	91
Ожидалось «Yes»	52	261	83
Итого	377	293	87

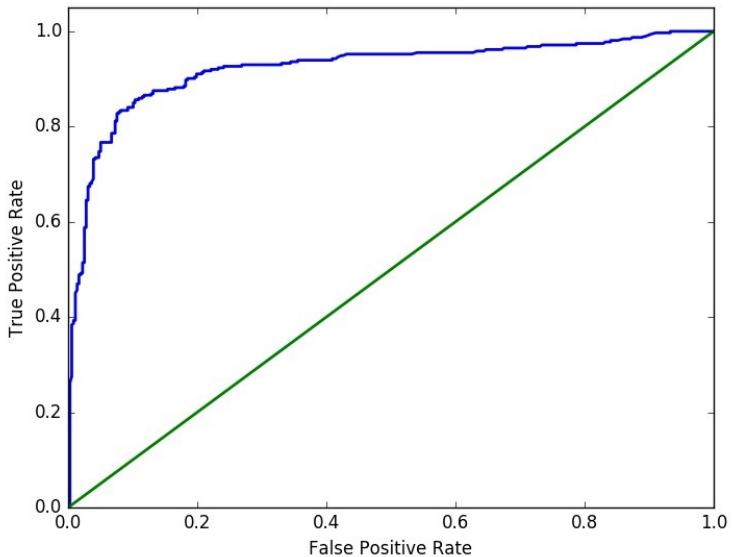
На рис. 4.16 изображена ROC-кривая, построенная для обученного классификатора.

Другие характеристики полученного классификатора:

- полнота = 0,8339,
- специфичность = 0,9104,
- точность = 0,8907,
- F-мера = 0,8614.

Сеть для локализации объектов обучалась в течение 5000 итераций (под итерацией здесь понимается настройка параметров для одного случайного изображения из обучающей выборки), используя следующие параметры:

- скорость обучения – 0,0003;

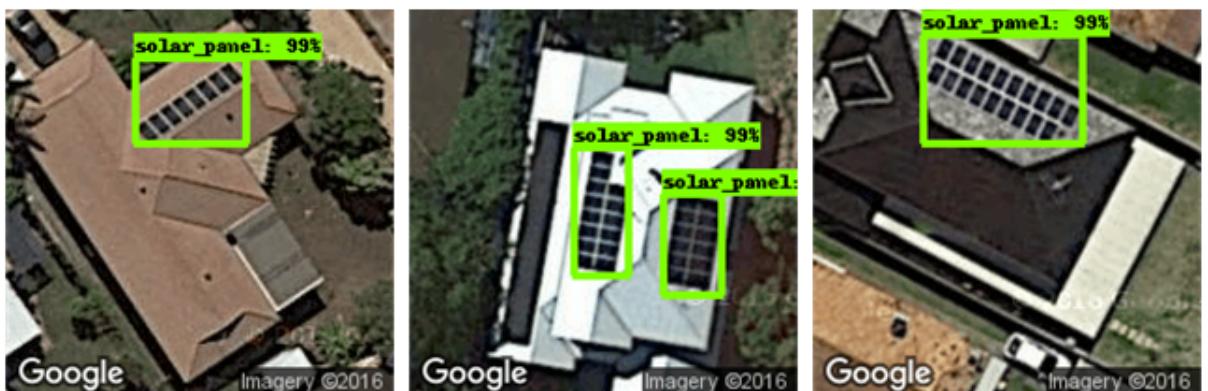


**Рисунок 4.16.** – ROC-кривая для обученного бинарного классификатора,  
**AUC = 0,92**

– моментный параметр – 0,9;

После выполнения обучения, нами проводилось исследование обобщающей способности сети с использованием тестовой выборки из 200 изображений. Полученный результат составил  **$AP=0,9299$** .

На рис. 4.17 изображены некоторые результаты детекции солнечных панелей на изображениях низкого разрешения.



**Рисунок 4.17.** – Результаты детекции для изображений из тестовой выборки

Как можно видеть, предложенный метод осуществляет достаточно точное обнаружение. На рис. 4.18 изображены результаты работы метода для произвольных изображений (не из тестовой выборки).



Рисунок 4.18. – Результаты детекции для произвольных изображений

### 4.3 Выводы

1) Разработана система автоматического контроля качества нанесения маркировки продукции. Достоинствами предлагаемого решения является модульная структура, позволяющая осуществлять независимую поддержку каждого функционального модуля. Разработанные нейросетевые модули детекции и распознавания позволили осуществлять обнаружение продукта и его маркировки с эффективностью в **99%** ( $mAP = 0.99$ ) и **92%** ( $mAP = 0.92$ ) для отдельных символов маркировки.

2) Разработана система определения наличия и детекции солнечных панелей на аэрофотоснимках. Достоинством разработанной системы является возможность работы системы с фотографиями, имеющими низкое разрешение. Использование предобученных сверточных нейронных сетей позволило достичь точности 87,46 % в определении наличия солнечных панелей на фото.

3) Разработана нейросетевая система детекции и распознавания лиц на фото и видеоизображениях, которая основывается на использовании предобученных сверточных нейронных сетей и позволяет расширять базу знаний интеллектуальной системы.

# ЗАКЛЮЧЕНИЕ

## Основные научные результаты диссертации

1. Выявлена и доказана эквивалентность максимизации функции правдоподобия распределения входных данных, минимизации суммарной квадратичной ошибки сети при использовании линейных нейронов и минимизации кросс-энтропийной функции ошибки сети в пространстве синаптических связей ограниченной машины Больцмана. Из полученных теоретических результатов следует, что природа неконтролируемого обучения в RBM-сети является идентичной при использовании различных целевых функций ([2–A], [3–A], [4–A], [5–A], [10–A], [12–A], [13–A]);
2. Разработан метод неконтролируемого предобучения глубоких нейронных сетей, базирующийся на минимизации квадратичной ошибки сети в скрытом и видимом слоях RBM-машины, что позволяет учитывать нелинейную природу нейронных элементов. Разработанный метод применен для обучения глубоких полносвязных и сверточных архитектур нейронных сетей и протестирован на выборках MNIST, CIFAR-10, CIFAR-100. Показано, что предложенный метод обладает большей эффективностью, чем классический ([1–A], [2–A], [3–A], [4–A], [5–A], [10–A], [12–A], [13–A], [17–A], [18–A], [19–A], [20–A], [21–A]);
3. Предложен алгоритм редуцирования параметров глубокой нейронной сети, базирующийся на неконтролируемом предобучении сети и позволяющий сократить количество настраиваемых параметров сети и упростить ее архитектуру без потери обобщающей способности. Проведены вычислительные эксперименты, доказывающие эффективность предложенного метода ([11–A], [16–A], [28–A]);
4. Разработана нейросетевая система распознавания маркировки продукта на производственной линии, базирующаяся на интеграции различных сверточных нейронных сетей. Предложенный конвейер представляет собой цепочку взаимодействующих нейросетевых моделей, детекторов и классификаторов, каждая из которых решает отдельную задачу обнаружения и распознавания определенного типа маркировки либо ее части. Подобная архитектура позволила осуществлять распознавание различных типов маркировок с возможностью изменения и добавления новых типов благодаря простой модульной структуре. Типы нейросетей, используемых для построения конвейера, позволяют осуществлять обработку изображения в реальном времени ([6–A], [7–A], [8–A], [22–A], [23–A], [24–A], [25–A], [26–A], [27–A]);

5. Разработана нейросетевая система обнаружения солнечных панелей на аэрофотоснимках, базирующаяся на применении предобученных сверточных нейронных сетей и позволяющая обнаруживать солнечные панели с точностью 87,46% с возможностью использования фотографий низкого разрешения ([9–A], [14–A], [15–A]).

### **Рекомендации по практическому использованию результатов**

Результаты исследований использованы для практической реализации интеллектуальной системы видеонаблюдения реального времени. Научные и практические результаты диссертационной работы использованы в НИР, а также в ряде прикладных систем.

Предложенный метод предобучения глубоких нейронных сетей применялся при обучении нейросетевых компонентов для нейросетевой системы распознавания маркировки продукта на производственной линии ОАО «Савушкин продукт».

Разработанные алгоритмы обнаружения и локализации солнечных панелей на аэрофотоснимках, методы предобучения нейросетевых моделей и инструментальные средства внедрены и используются в компании ООО «Intelligent Semantic Systems» при разработке интеллектуальной диалоговой системы в составе подсистемы компьютерного зрения.

Кроме того, научные и практические результаты диссертационной работы используются в учебном процессе учреждения образования «Брестский государственный технический университет».

Разработка проводилась с использованием свободного программного обеспечения и может быть использована в различных отечественных проектах без необходимости приобретения дорогостоящих программных средств.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

## Библиографический список

1. Mitchell, T. M. Machine learning / T. M. Mitchell. — McGraw-hill New York, 1997. — Vol. 1.
2. Kandel, E. R. In Search of Memory: The Emergence of a New Science of Mind / E. R. Kandel. — New York: W. W. Norton & Company, 2006.
3. Cybenko, George V. Approximation by superpositions of a sigmoidal function / George V. Cybenko // Mathematics of Control, Signals and Systems. — 1989. — Vol. 2. — P. 303–314.
4. Головко, В. А. Нейросетевые технологии обработки данных / В. А. Головко, В. В. Краснопрошин. — Минск: БГУ, 2017.
5. Ivakhnenko, A. G. Cybernetics and Forecasting Techniques / A. G. Ivakhnenko, V. G. Lapa. Modern analytic and computational methods in science and mathematics. — American Elsevier Publishing Company, 1967. <https://books.google.by/books?id=z7hPAQAAIAAJ>.
6. Elman, Jeffrey L. Finding structure in time / Jeffrey L. Elman // Cogn. Sci. — 1990. — Vol. 14. — P. 179–211.
7. Jordan, Michael I. Serial order: A parallel distributed processing approach / Michael I. Jordan // Advances in psychology. — 1997. — Vol. 121. — P. 471–495.
8. Hinton, Geoffrey E. Learning representations by recirculation / Geoffrey E. Hinton, James L. McClelland // NIPS. — 1987.
9. Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position / K. Fukushima // Biological Cybernetics. — Vol. 36. — 1980. — P. 193–202.
10. Handwritten digit recognition with a back-propagation network / Yann LeCun, Bernhard E. Boser, John S. Denker [et al] // NIPS. — 1989.
11. Kohonen, Teuvo. Self-organizing maps / Teuvo Kohonen. — Berlin: Springer, 2001.
12. Grossberg, Stephen. Competitive learning: From interactive activation to adaptive resonance / Stephen Grossberg // Cogn. Sci. — 1987. — Vol. 11. — P. 23–63.
13. Hopfield, J. J. Neurons with graded response have collective computational properties like those of two-state neurons. / J. J. Hopfield // Proceedings of the National Academy of Sciences. — 1984. — Vol. 81, № 10. — P. 3088–3092.

14. Lippmann, R. An introduction to computing with neural nets / R. Lippmann // IEEE ASSP Magazine. — 1987. — Vol. 4, № 2. — P. 4–22.
15. Kosko, Bart. Bidirectional associative memories / Bart Kosko // IEEE Trans. Syst. Man Cybern. — 1988. — Vol. 18. — P. 49–60.
16. Hecht-Nielsen, Robert. Counterpropagation networks / Robert Hecht-Nielsen // Appl. Opt. — 1987. — Vol. 26, № 23. — P. 4979–4984.
17. Broomhead, David. Radial basis functions, multi-variable functional interpolation and adaptive networks / David Broomhead, David Lowe // ROYAL SIGNALS AND RADAR ESTABLISHMENT MALVERN (UNITED KINGDOM). — 1988. — Vol. RSRE-MEMO-4148.
18. Jang, Jyh-Shing. Neuro-fuzzy and soft computing-a computational approach to learning and machine intelligence [book review] / Jyh-Shing Jang, Chuen-Tsai Sun, Eiji Mizutani // Automatic Control, IEEE Transactions on. — 1997. — Vol. 42. — P. 1482 – 1484.
19. Neural Network and Artificial Immune Systems for Malware and Network Intrusion Detection / Vladimir Golovko, Sergei Bezobrazov, Pavel Kachurka, Leonid Vaitsekhovich // Advances in Machine Learning II: Dedicated to the Memory of Professor Ryszard S.Michalski / Ed. by Jacek Koronacki, Zbigniew W. Raś, Sławomir T. Wierzchoń, Janusz Kacprzyk. — Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. — P. 485–513.
20. LeCun, Y. Deep learning / Y. LeCun, Y. Bengio, G. Hinton // Nature. — 2015. — № 521 (7553). — P. 436–444.
21. An introductory review of deep learning for prediction models with big data / Frank Emmert-Streib, Zhen Yang, Han Feng [et al] // Frontiers in Artificial Intelligence. — 2020. — Vol. 3.
22. Nerf: Representing scenes as neural radiance fields for view synthesis / B. Mildenhall, P. P. Srinivasan, M. Tancik [et al] // ECCV. — 2020.
23. Bengio, Y. Learning deep architectures for ai / Y. Bengio // Foundations and Trends in Machine Learning. — 2009. — № 2(1). — P. 1–127.
24. Hinton, G. Reducing the dimensionality of data with neural networks / G. Hinton, R. Salakhutdinov // Science. — 2006. — № 313 (5786). — P. 504–507.
25. McCulloch, Warren. A logical calculus of ideas immanent in nervous activity / Warren McCulloch, Walter Pitts // Bulletin of Mathematical Biophysics. — 1943. — Vol. 5. — P. 127–147.
26. Hebb, Donald O. The organization of behavior: A neuropsychological theory / Donald O. Hebb. — New York: Wiley, 1949.

27. Widrow, Bernard. Adaptive switching circuits / Bernard Widrow, Marcian E. Hoff // 1960 IRE WESCON Convention Record, Part 4. — New York: IRE, 1960. — P. 96–104.
28. Rosenblatt, F. Principles of Neurodynamics / F. Rosenblatt. — Spartan Books, 1959.
29. Minsky, Marvin. Perceptrons: An Introduction to Computational Geometry / Marvin Minsky, Seymour Papert. — Cambridge, MA, USA: MIT Press, 1969.
30. Grossberg, Stephen. Adaptive pattern classification and universal recoding: II. feedback, expectation, olfaction, illusions / Stephen Grossberg // Biol. Cybern. — 1976. — Vol. 23, № 4. — P. 187–202.
31. Kohonen, Teuvo. Associative memory : a system-theoretical approach / Teuvo Kohonen / Teuvo. Kohonen. — Springer-Verlag Berlin ; New York, 1977. — P. 176.
32. Rumelhart, David E. Learning representations by back-propagating errors / David E Rumelhart, Geoffrey E Hinton, Ronald J Williams // nature. — 1986. — Vol. 323, № 6088. — P. 533–536.
33. Cortes, Corinna. Support-vector networks / Corinna Cortes, Vladimir Vapnik // Mach. Learn. — 1995. — Vol. 20, № 3. — P. 273–297.
34. Behnke, Sven. Hierarchical Neural Networks for Image Interpretation / Sven Behnke. Lecture Notes in Computer Science. — Springer Berlin, Heidelberg, 2003.
35. Hinton, G. E. A fast learning algorithm for deep belief nets / G. E. Hinton, S. Osindero, Y. Teh // Neural Computation. — 2006. — № 18. — P. 1527–1554.
36. Taori, Rohan. Alpaca: A strong, replicable instruction-following model. — Режим доступа: <https://crfm.stanford.edu/2023/03/13/alpaca.html>. — 2023. — Дата доступа: 05.03.2023.
37. Taori, Rohan. Stanford alpaca: An instruction-following llama model. — [https://github.com/tatsu-lab/stanford\\_alpaca](https://github.com/tatsu-lab/stanford_alpaca). — 2023. — Дата доступа: 05.03.2023.
38. Zhang, Susan. Opt: Open pre-trained transformer language models. — 2022.
39. Glorot, X. Deep sparse rectifier networks / X. Glorot, A. Bordes, Y. & Bengio // Proceedings of the 14th International Conference on Artificial Intelligence and Statistics. — Vol. 15. — 2011. — P. 315–323.

40. Хайкин, С. Нейронные сети: полный курс, 2-е издание / С. Хайкин. — Москва: Издательский дом «Вильямс», 2006. — 1104 с.
41. Гудфеллоу, Я. Глубокое обучение / Я. Гудфеллоу, И. Бенджио, А. Курвилль. — Москва : ДМК Пресс, 2017. — 652 с.
42. Duchi, J. Adaptive subgradient methods for online learning and stochastic optimization / J. Duchi, E. Hazan, Y. Singer // Journal of Machine Learning Research. — 2011.
43. Kingma, D. Adam: A method for stochastic optimization / D. Kingma, J. Ba // 3rd International Conference for Learning Representations. — 2014.
44. Hinton, G. Training products of experts by minimizing contrastive divergence / G. Hinton // Neural Computation. — 2002. — № 14. — P. 1771–1800.
45. Hinton, G. E. A practical guide to training restricted boltzmann machines: Tech. Rep. 2010-000 / G. E. Hinton: University of Toronto, 2010.
46. The “wake-sleep” algorithm for unsupervised neural networks / Geoffrey E. Hinton, Peter Dayan, Brendan J. Frey, Radford M. Neal // Science. — 1995. — Vol. 268, № 5214. — P. 1158–1161.
47. Greedy layer-wise training of deep networks / Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle // Advances in neural information processing systems. — MA: MIT Press, Cambridge, 2007. — P. 153–160.
48. Using different cost functions to train stacked auto-encoders / Telmo Amaral, Luis Silva, Luis Alexandre [et al] // 12th Mexican International Conference on Artificial Intelligence. — 2013. — P. 114–120.
49. Gibbs, J.W. Elementary Principles in Statistical Mechanics: Developed with Especial Reference to the Rational Foundations of Thermodynamics / J.W. Gibbs. — C. Scribner's sons, 1902.
50. Neural network model for transient ischemic attacks diagnostics / V. Golovko, H. Vaitsekhovich, E. Apanal, A. Mastykin // Optical Memory and Neural Networks (Information Optics). — 2012. — № 21(3). — P. 166–176.
51. Mostafa, Hesham. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. — 2019.
52. Han, Song. Learning both weights and connections for efficient neural networks. — 2015.
53. LeCun, Y. The mnist database of handwritten digits. — Режим доступа: <http://yann.lecun.com/exdb/mnist/>. — Дата доступа: 2023-03-05.
54. Krizhevsky, Alex. Learning multiple layers of features from tiny images / Alex Krizhevsky. — 2009. — P. 32–33.

55. Scholz, M. Nonlinear principal component analysis: neural network models and applications / M. Scholz, M. Fraunholz, J. Selbig. — Springer, 2008. — P. 44–67.
56. Perry, C. Colab's 'pay as you go' offers more access to powerful nvidia compute for machine learning. — Режим доступа: <https://blog.tensorflow.org/2022/09/colabs-pay-as-you-go-offers-more-access-to-powerful-nvidia-compute-for-machine-learning.html>. — Дата доступа: 2023-03-05.
57. Turchenko, V. Parallel batch pattern training algorithm for deep neural network / V. Turchenko, Golovko V. // High Performance Computing & Simulation (HPCS). — IEEE, 2014. — P. 697–702.
58. Torralba, Antonio. 80 million tiny images: A large data set for nonparametric object and scene recognition / Antonio Torralba, Rob Fergus, William T. Freeman // IEEE Transactions on Pattern Analysis and Machine Intelligence. — 2008. — Vol. 30, № 11. — P. 1958–1970.
59. Milk products – electronic sign. — Режим доступа: <https://datamark.by/markirovka-molochnoy-produktsii>. — Дата доступа: 05.03.2023.
60. Omron documentation. — Режим доступа: <https://robotics.omron.com/browse-documents>. — Дата доступа: 05.03.2023.
61. Liu, W. Ssd: Single shot multibox detector. — Режим доступа: <https://arxiv.org/abs/1512.02325>. — 2015. — Дата доступа: 05.03.2023.
62. Howard, A. Mobilenets: Efficient convolutional neural networks for mobile vision applications. — Режим доступа: <https://arxiv.org/abs/1704.04861>. — 2017. — Дата доступа: 05.03.2023.
63. Ren, S. Faster r-cnn: Towards real-time object detection with region proposal networks. — Режим доступа: <https://arxiv.org/abs/1506.01497>. — 2016. — Дата доступа: 05.03.2023.
64. He, K. Deep residual learning for image recognition. — Режим доступа: <https://arxiv.org/abs/1512.03385>. — 2015. — Дата доступа: 05.03.2023.
65. Reading digits in natural images with unsupervised feature learning / Y. Netzer, T. Wang, A. Coates [et al] // NIPS Workshop on Deep Learning and Unsupervised Feature Learning. — 2011.
66. Solar industry data // seia.org. — Режим доступа: <https://www.seia.org/solar-industry-research-data>. — 2020. — Дата доступа: 05.03.2023.
67. Gradient-based learning applied to document recognition / Y. LeCun, L. Bottou, Bengio Y., P. Haffner // Proceedings of the IEEE. — Vol. 86. — 1998. — P. 2278–2324.

68. Automatic solar photovoltaic panel detection in satellite imagery / J. Malof, R. Hou, L. Collins [et al] // International Conference on Renewable Energy Research and Applications ICRERA' 2015. — 2015.
69. Automatic detection of solar photovoltaic arrays in high resolution aerial imagery / J. Malof, K. Bradbury, L. Collins, R. Newell // Applied Energy. — Vol. 183. — 2016. — P. 229–240.
70. Bradbury, K. Distributed solar photovoltaic array location and extend data set for remote sensing object identification. — Режим доступа: <http://dx.doi.org/10.6084/m9.figshare.3385780.v1>. — 2016. — Дата доступа: 05.03.2023.
71. Lin, T. Microsoft coco: Common objects in context. — Режим доступа: <https://arxiv.org/pdf/1405.0312.pdf>. — 2015. — Дата доступа: 05.03.2023.
72. Why does unsupervised pre-training help deep learning? / D. Erhan, Y. Bengio, A. Courville [et al] // Journal of Machine Learning Research. — 2010. — № 11. — P. 625–660.
73. G., Hinton. Greedy layer-wise algorithm / Hinton G. // Journal of Machine Learning Research. — 2009. — № 1. — P. 1–40.
74. Ciresan, D. Multi-column deep neural networks for image classification / D. Ciresan, U. Meier, J. Schmidhuber // IEEE Conference on Computer Vision and Pattern Recognition. — IEEE, 2012. — P. 3642–3649.
75. Golik, P. Cross-entropy vs. squared error training: a theoretical and experimental comparison / P. Golik, P. Doetsch, H. Ney // Interspeech. — Lyon, France: 2013. — P. 1756–1760.
76. Zhou, P. Learning criteria for training neural network classifiers / P. Zhou, Austin J. // Neural Computing & Applications. — 1998. — № 7(4). — P. 334–342.
77. Fisher, R. The use of multiple measurements in taxonomic problems / R. Fisher // Annals of Eugenics. — 1936. — № 7. — P. 179–188.
78. Головко, В.А. От многослойных персепtronов к нейронным сетям глубокого доверия: парадигмы обучения и применение / В.А. Головко // Лекции по Нейроинформатике. — НИЯУ МИФИ, 2015. — С. 47–84.
79. Krizhevsky, A. Imagenet classification with deep convolutional neural networks / A. Krizhevsky, I. Sutskever, G. Hinton // In Proc. Advances in Neural Information Processing Systems. — Vol. 25. — Curran Associates, Inc., 2012. — P. 1090–1098.
80. Rosenblatt, Frank. Principles of neurodynamics: perceptrons and the theory of brain mechanisms / Frank Rosenblatt. — Washington: Spartan Books, 1962.

81. B., Widrow. Adaptive switching circuits / Widrow B., Hoff M. // In 1960 IRE WESCON Convention Record. — 1960. — P. 96–104.
82. Gs1 datamatrix – electronic sign. — Режим доступа: <https://datamark.by/voprosy-otvety/gs1-datamatrix>. — Дата доступа: 05.03.2023.
83. Schroff, F. Facenet: A unified embedding for face recognition and clustering / F. Schroff, D. Kalenichenko, J. Philbin // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). — 2015. — P. 815–823.
84. Deep residual learning for image recognition / K. He, X. Zhang, S. Ren, J. Sun // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). — 2016. — P. 770–778.
85. King, D. E. Dlib-ml: A machine learning toolkit / D. E. King // Journal of Machine Learning Research. — Vol. 10. — 2009. — P. 1755–1758.
86. Joint face detection and alignment using multitask cascaded convolutional networks / K. Zhang, Z. Zhang, Z. Li, Y. Qiao // in IEEE Signal Processing Letters. — Vol. 23. — 2016. — P. 1499–1503.

## **Список публикаций соискателя ученой степени**

### **Статьи в рецензируемых научных журналах**

- 1–А. Golovko, V. Learning Technique for Deep Belief Neural Networks / V. Golovko, A. Kroshchanka, U. Rubanau, S. Jankowski // Springer. — 2014. — Vol. 440. Communication in Computer and Information Science — P. 136–146.
- 2–А. Головко, В. А. Персептроны и нейронные сети глубокого доверия: обучение и применение / В. А. Головко, А. А. Крощенко // Вестник Брестского государственного технического университета. — Брест, 2014. — Т. 5. — С. 2–12.
- 3–А. Головко, В. А. Метод обучения нейронной сети глубокого доверия и применение для визуализации данных / В. А. Головко, А. А. Крощенко // Комп’ютерно-інтегровані технології: освіта, наука, виробництво. — Луцьк, 2015. — Вып. 19. — С. 6–12.
- 4–А. Golovko, V. The nature of unsupervised learning in deep neural networks: A new understanding and novel approach / V. Golovko, A. Kroshchanka, D. Treadwell // Optical Memory and Neural Networks. — Pleiades Publishing, 2016. — P. 127–141.
- 5–А. Головко, В. А. Теория глубокого обучения: конвенциальный и новый подход / В. А. Головко, А. А. Крощенко, М. В. Хацкевич // Вестник

Брестского государственного технического университета. — Брест, 2016. — № 5. — С. 7–16.

6–А. Головко, В. А. Интеграция искусственных нейронных сетей с базами знаний / В. А. Головко, В. В. Голенков, В. П. Иващенко, В. В. Таберко, Д. С. Шаток, А. А. Крощенко, М. В. Ковалёв // Онтология проектирования. — EBSCO Publishing, 2018. — Т. 8. — № 3(29). — С. 366–386.

7–А. Крощенко, А. А. Реализация нейросетевой системы распознавания маркировки продукции / А. А. Крощенко, В. А. Головко // Вестник Брестского государственного технического университета. — Брест, 2019. — № 5. — С. 9–12.

8–А. Golovko, V. Brands and caps labeling recognition in images using deep learning / V. Golovko, A. Kroshchanka, E. Mikhno // International Conference on Pattern Recognition and Information Processing. — Springer, Cham, 2019. — P. 35–51.

9–А. Golovko, V. Deep convolutional neural network for detection of solar panels / V. Golovko, A. Kroshchanka, E. Mikhno, M. Komar, A. Sachenko // Data-Centric Business and Applications. — Springer, Cham, 2021. — P. 371–389.

10–А. Golovko, V. A. Deep Neural Networks: Selected Aspects of Learning and Application / V. A. Golovko, A. A. Kroshchanka, E. V. Mikhno // Pattern Recognition and Image Analysis. — Pleiades Publishing, 2021. — Vol. 31. — № 1. — P. 132–143.

11–А. Kroshchanka, A. A. Method for Reducing Neural-Network Models of Computer Vision / A. A. Kroshchanka, V. A. Golovko, M. Chodyka // Pattern Recognition and Image Analysis. — Pleiades Publishing, 2022. — Vol. 32. — № 2. — P. 294–300.

**Статьи в сборниках материалов научных конференций,  
включенных в системы международного цитирования  
(Scopus, Web of Science и IEEE Xplore digital)**

12—A. Golovko, V. A New Technique for Restricted Boltzmann Machine Learning / A. Kroshchanka, V. Turchenko, S. Jankowski, D. Treadwell // Proceedings of the 8th IEEE International Conference IDAACS-2015, Warsaw 24-26 September 2015. — Warsaw, 2015. — P. 182–186.

13—A. Golovko, V. Theoretical Notes on Unsupervised Learning in Deep Neural Networks / V. Golovko, A. Kroshchanka // Proceedings of the 8th International Joint Conference on Computational Intelligence (IJCCI 2016). — SCITEPRESS, 2016. — P. 91–96.

14—A. Golovko, V. Convolutional neural network based solar photovoltaic panel detection in satellite photos / V. Golovko, S. Bezobrazov, A. Kroshchanka, A. Sachenko, M. Komar, A. Karachka // 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS). — IEEE, 2017. — Vol. 1. — P. 14–19.

15—A. Golovko, V. Development of solar panels detector / V. Golovko, A. Kroshchanka, S. Bezobrazov, A. Sachenko, M. Komar, O. Novosad // International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T). — IEEE, 2018. — P. 761–764.

16—A. Kroshchanka, A. The Reduction of Fully Connected Neural Network Parameters Using the Pre-training Technique / A. Kroshchanka, V. Golovko // 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS). — IEEE, 2021. — Vol. 2. — P. 937–941.

**Статьи в сборниках материалов научных конференций**

17—А. Крощенко, А. А. Методы глубокого обучения нейронных сетей / А.А. Крощенко // Материалы конференции «Вычислительные методы, модели и образовательные технологии». — Брест : БрГУ, 2013. — С. 21–22.

18—А. Головко, В. А. Об одном методе обучения нейронных сетей глубокого доверия / В. А. Головко, А. А. Крощенко // Вычислительные методы, модели и образовательные технологии : сборник материалов международной научно-практической конференции. — Брест : БрГУ, 2014. — С. 98–99.

19—А. Крощенко, А. А. Применение нейронных сетей глубокого доверия в интеллектуальном анализе данных / А. А. Крощенко // сборник материалов IX Республиканской научной конференции молодых ученых и студентов «Современные проблемы математики и вычислительной техники». — Брест : БрГТУ, 2015. — С. 12–14.

20—А. Головко, В. А. Применение нейронных сетей глубокого доверия в интеллектуальном анализе данных / В. А. Головко, А. А. Крощенко // Вычислительные методы, модели и образовательные технологии : сборник материалов международной научно-практической конференции. — Брест : БрГУ, 2015.— С. 97–98.

21—А. Крощенко, А. А. Применение глубокой нейронной сети для решения задачи распознавания образов / А. А. Крощенко // Вычислительные методы, модели и образовательные технологии : сборник материалов международной научно-практической конференции. — Брест : БрГУ, 2016.— С. 132–133.

22—A. Golovko, V. A. Integration of artificial neural networks and knowledge bases / V. A. Golovko, A. A. Kroshchanka, V. V. Golenkov, V. P. Ivashenko, M. V. Kovalev, V. V. Taberko, D. S. Ivaniuk // Open Semantic Technologies for Intelligent Systems (OSTIS-2018). — Minsk : BSUIR, 2018. — Р. 133–146.

23—A. Golovko, V. Principles of decision-making systems building based on the integration of neural networks and semantic models / V. Golovko, A. Kroshchanka, V. Ivashenko, M. Kovalev, V. Taberko, D. Ivaniuk // Open Semantic Technologies for Intelligent Systems (OSTIS-2019). — Minsk : BSUIR, 2019. — Р. 91–102.

24—А. Головко, В. А. Обнаружение и распознавание маркировки продукции с помощью нейросетевых алгоритмов / В. А. Головко, А. А. Крощенко // Вычислительные методы, модели и образовательные технологии : сборник материалов VII международной научно-практической конференции. — Брест : БрГУ, 2019. — С. 3–6.

25—A. Golovko, V. Implementation of an intelligent decision support system to accompany the manufacturing process / V. Golovko, A. Kroshchanka, M. Kovalev, V. Taberko, D. Ivaniuk // Open Semantic Technologies for Intelligent Systems (OSTIS-2020). — Minsk : BSUIR, 2020. — Р. 175–182.

26—A. Golovko, V. Neuro-Symbolic Artificial Intelligence: Application for Control the Quality of Product Labeling / V. Golovko, A. Kroshchanka, M. Kovalev, V. Taberko, D. Ivaniuk // International Conference on Open Semantic Technologies for Intelligent Systems. — Springer, Cham, 2020.— Р. 81–101.

27—A. Kroshchanka, A. Neural network component of the product marking recognition system on the production line / A. Kroshchanka, D. Ivaniuk // Open Semantic Technologies for Intelligent Systems (OSTIS-2021). — Minsk : BSUIR, 2021. - P. 219–224.

28—A. Kroshchanka, A. Reduction of neural network models in intelligent computer systems of a new generation / A. Kroshchanka // Open Semantic Technologies for Intelligent Systems (OSTIS-2023). — Minsk : BSUIR, 2023. - P. 127–132.

ПРИЛОЖЕНИЕ А

**ИСХОДНЫЙ КОД РЕАЛИЗАЦИИ ПРОЦЕДУР  
ПРЕДОБУЧЕНИЯ И РЕДУКЦИИ**

```
import torch
import torch.nn as nn
from common_types import InitTypes

class UnifiedClassifier(nn.Module):
    def __init__(self, layers_config):
        super().__init__()
        self.layers = nn.ModuleList()
        self.a_functions = []
        print(layers_config["architecture"])
        self.layers_config = layers_config["architecture"]
        layer_index = 0
        for layer in self.layers_config:
            if len(layer[0]) == 2:
                new_layer = nn.Linear(*layer[0])
            if len(layer[0]) == 3:
                new_layer = nn.Conv2d(*layer[0])
            self.layers.append(new_layer)
            self.a_functions.append(layer[1][-1])
            layer_index += 1

    def forward(self, x):
        for layer, afunc, layer_config in zip(self.layers, self.a_functions, self.layers_config):
            x = afunc(layer(x))
            if len(layer_config) == 3:
                post_processing_actions = layer_config[2]
                for action in post_processing_actions:
                    x = action(x)
        return x

class RBM(nn.Module):
    def __init__(self, n_vis, n_hid, a_func, init_type,
                 without_sampling):
        super(RBM, self).__init__()
        if init_type == InitTypes.Kaiming:
            W = torch.empty(n_vis, n_hid)
            v = torch.empty(1, n_vis)
            h = torch.empty(1, n_hid)
            torch.nn.init.kaiming_normal_(W, mode='fan_out',
```

```

        nonlinearity="relu")
    torch.nn.init.kaiming_normal_(v, mode='fan_out',
        nonlinearity="relu")
    torch.nn.init.kaiming_normal_(h, mode='fan_out',
        nonlinearity="relu")
    self.W = nn.Parameter(W)
    self.v = nn.Parameter(v)
    self.h = nn.Parameter(h)
elif init_type == InitTypes.SimpleNormal:
    self.W = nn.Parameter(0.1 * torch.randn(n_vis,
        n_hid))
    self.v = nn.Parameter(torch.zeros(1, n_vis))
    self.h = nn.Parameter(-1 * torch.ones(1, n_hid))
elif init_type == InitTypes.SimpleUniform:
    self.W = nn.Parameter(0.02 * torch.rand(n_vis,
        n_hid) - 0.01)
    self.v = nn.Parameter(0.02 * torch.rand(1, n_vis) -
        0.01)
    self.h = nn.Parameter(0.02 * torch.rand(1, n_hid) -
        0.01)
self.a_func = a_func
self.without_sampling = without_sampling

def visible_to_hidden(self, v):
    weighted_sum = torch.mm(v, self.W) + self.h
    output = self.a_func[1](weighted_sum)
    return output, weighted_sum

def hidden_to_visible(self, h):
    weighted_sum = torch.mm(h, self.W.t()) + self.v
    output = self.a_func[0](weighted_sum)
    return output, weighted_sum

def forward(self, v0):
    h0, h0_ws = self.visible_to_hidden(v0)
    if self.without_sampling:
        h_sampled = h0
    else:
        if self.a_func[1] == torch.sigmoid:
            h_sampled = 1. * (h0 > torch.rand(h0.shape))
        elif self.a_func[1] == torch.relu:
            h0_std = torch.std(h0, dim=0, unbiased=False)
            h_sampled = h0 + torch.normal(0, h0_std)
        else:
            h_sampled = h0
    v1, v1_ws = self.hidden_to_visible(h_sampled)
    h1, h1_ws = self.visible_to_hidden(v1)

```

```

        return v0, v1, v1_ws, h0, h0_ws, h1, h1_ws

    def __str__(self) -> str:
        return "vis: " + str(self.v.shape) + \
               "hid: " + str(self.h.shape)

class CRBM(nn.Module):
    def __init__(self, n_vis_channels, n_hid_channels,
                 kernel_size, a_func, init_type, without_sampling):
        super(CRBM, self).__init__()
        if init_type == InitTypes.Kaiming:
            W = torch.empty(n_hid_channels, n_vis_channels,
                            kernel_size, kernel_size)
            torch.nn.init.kaiming_normal_(W, nonlinearity="relu")
            self.W = nn.Parameter(W)
        elif init_type == InitTypes.SimpleNormal:
            self.W = nn.Parameter(0.01 * torch.randn(
                n_hid_channels, n_vis_channels, kernel_size,
                kernel_size))
        elif init_type == InitTypes.SimpleUniform:
            self.W = nn.Parameter(0.02 * torch.randn(
                n_hid_channels, n_vis_channels, kernel_size,
                kernel_size) - 0.01)
        self.a_func = a_func
        self.without_sampling = without_sampling

    def visible_to_hidden(self, v):
        weighted_sum = torch.convolution(v, self.W, None,
                                         stride=[1, 1], padding=[0, 0], dilation=[1, 1],
                                         transposed=False, output_padding=[0, 0], groups=1)
        output = self.a_func[1](weighted_sum)
        return output, weighted_sum

    def hidden_to_visible(self, h):
        weighted_sum = torch.conv_transpose2d(h, self.W, None,
                                              stride=[1, 1], padding=[0, 0], output_padding=[0, 0],
                                              groups=1,
                                              dilation=[1, 1])
        output = self.a_func[0](weighted_sum)
        return output, weighted_sum

    def forward(self, v0):
        h0, h0_ws = self.visible_to_hidden(v0)
        if self.without_sampling:
            h_sampled = h0
        else:

```

```

        if self.a_func[1] == torch.sigmoid:
            h_sampled = 1. * (h0 > torch.rand(h0.shape))
        elif self.a_func[1] == torch.relu:
            h0_std = torch.std(h0, dim=0, unbiased=False)
            h_sampled = h0 + torch.normal(0, h0_std)
        else:
            h_sampled = h0
    v1, v1_ws = self.hidden_to_visible(h_sampled)
    h1, h1_ws = self.visible_to_hidden(v1)
    return v0, v1, v1_ws, h0, h0_ws, h1, h1_ws

def __str__(self) -> str:
    return "vis: " + str(self.v.shape) + \
           "hid: " + str(self.h.shape)

import torch
from torch import nn
import utilities as utl
import config
from models import RBM, CRBM

class RBMStack:
    def __init__(self, layers_config, device, init_type,
                 without_sampling):
        self.rbm_stack = []
        self.layers = layers_config["architecture"]
        self.input_dim = layers_config["input_dim"]
        self.device = device
        for i in range(0, len(self.layers) - 1):
            layer_params = self.layers[i][0]
            layer_activation_function = self.layers[i][1]
            rbm_constructor = RBM if len(layer_params) == 2
            else CRBM
            rbm = rbm_constructor(*layer_params,
                                  layer_activation_function, init_type,
                                  without_sampling)
            self.rbm_stack.append(rbm.to(self.device))

    def _prepare_train_set(self, train_set, batch_size,
                          input_dim, layer_index=0):
        batches_count = len(train_set) / batch_size
        if isinstance(input_dim, tuple):
            resulted_array = torch.zeros((train_set.shape[0], *
                                         input_dim))
        else:
            resulted_array = torch.zeros((train_set.shape[0],
                                         input_dim))

```

```

i = 0
while i < batches_count:
    _slice = slice(i * batch_size, (i + 1) * batch_size)
    inputs = train_set[_slice]
    if layer_index == 0:
        resulted_array[_slice] = inputs
    else:
        inputs = inputs.to(self.device)
        v0, v1, v1_ws, h0, h0_ws, h1, h1_ws = self.
            rbm_stack[layer_index - 1](inputs)
        resulted_array[_slice] = h0
    i += 1
if len(self.layers[layer_index - 1]) == 3:
    post_processing_actions = self.layers[layer_index -
        1][2]
for action in post_processing_actions:
    if not isinstance(action, torch.nn.Dropout):
        resulted_array = action(resulted_array)
return resulted_array

def train(self, train_set, pretrain_type):
    layers_losses = {}
    train_set = self._prepare_train_set(train_set, config.
        pretraining_batch_size, self.input_dim)
    batches_count = len(train_set) / config.
        pretraining_batch_size
    layer_index = 0
    with torch.no_grad():
        for rbm, layer in zip(self.rbm_stack, self.layers):
            output_shape = utl.
                train_rbm_with_custom_dataset(
                    train_set, self.device, rbm, pretrain_type,
                    batches_count)
            layer_index += 1
            train_set = self._prepare_train_set(train_set,
                config.pretraining_batch_size, output_shape
                [1:],
                layer_index
            )
    return layers_losses

def torch_model_init_from_weights(self, torch_model):
    with torch.no_grad():
        for i in range(0, len(torch_model.layers) - 1):
            if len(torch_model.layers_config[i][0]) == 2:
                torch_model.layers[i].weight.data = self.

```

```

        rbm_stack[i].W.T
        torch_model.layers[i].bias.data = torch.
            reshape(self.rbm_stack[i].h, (len(self.
                rbm_stack[i].h[0]),))
    if len(torch_model.layers_config[i][0]) == 3:
        torch_model.layers[i].weight.data = self.
            rbm_stack[i].W

def do_reduction(self, layers_config):
    with torch.no_grad():
        for i in range(0, len(self.layers) - 1):
            mask = torch.abs(self.rbm_stack[i].W) > config.
                reduction_param
            self.rbm_stack[i].W *= mask.double()
    condition = None
    for i in range(0, len(self.layers) - 1):
        if condition is not None:
            self.rbm_stack[i].W = nn.Parameter(self.
                rbm_stack[i].W[condition])
        condition = torch.abs(self.rbm_stack[i].W).sum(
            dim=0) != 0
        self.rbm_stack[i].W = nn.Parameter(self.
            rbm_stack[i].W[:, condition])
        self.rbm_stack[i].h = nn.Parameter(self.
            rbm_stack[i].h[:, condition])
    layers_config["architecture"][i][0] = tuple(
        self.rbm_stack[i].W.shape)
    previous_neurons_count = layers_config["
        architecture"][-2][0][1]
    end_layer_neurons_count = layers_config["
        architecture"][-1][0][1]
    layers_config["architecture"][-1][0] = (previous_neurons_count,
        end_layer_neurons_count)

import random
import config
import torchvision.datasets as datasets
import torch
from rbm_stack import RBMStack
from models import UnifiedClassifier
from torch import nn
import torch.optim as optim
import data_config
import torch.nn.functional as F
from common_types import PretrainingType, DatasetType,
    Statistics

```

```

from models import RBM


def get_random_seeds(count):
    seeds = []
    for i in range(0, count):
        seeds.append(random.randint(0, config.max_random_seed))
    return seeds


def get_dataset_constructor(dataset_type: DatasetType):
    dataset_selector = {
        DatasetType.MNIST: datasets.MNIST,
        DatasetType.CIFAR10: datasets.CIFAR10,
        DatasetType.CIFAR100: datasets.CIFAR100
    }
    return dataset_selector[dataset_type]


def train_rbm_with_custom_dataset(train_set, device, rbm,
                                  pretrain_type, batches_count):
    train_func = train_rbm if isinstance(rbm, RBM) else
        train_crbm
    losses, output_shape = train_func(rbm, device,
                                      batches_count, train_set, pretrain_type)
    return output_shape


def train_rbm(rbm, device, batches_count, train_set,
             pretrain_type):
    delta_weights = torch.zeros(rbm.W.shape).to(device)
    delta_v_thresholds = torch.zeros(rbm.v.shape).to(device)
    delta_h_thresholds = torch.zeros(rbm.h.shape).to(device)
    losses = []
    act_func = rbm.a_func
    for epoch in range(config.pretraining_epochs):
        loss = 0.
        i = 0
        momentum = config.momentum_beg if epoch < config.
            momentum_change_epoch else config.momentum_end
        while i < batches_count:
            inputs = train_set[i * config.
                pretraining_batch_size:(i + 1) * config.
                pretraining_batch_size].to(device)
            v0, v1, v1_ws, h0, h0_ws, h1, h1_ws = rbm(inputs)
            if pretrain_type == PretrainingType.RBMClassic:
                der_v, der_h = 1, 1

```

```

        rate = config.pretraining_rate
    elif pretrain_type == PretrainingType.REBA:
        der_v = (act_func[0](v1_ws + 0.00001) -
                  act_func[0](v1_ws - 0.00001)) / 0.00002
        der_h = (act_func[1](h1_ws + 0.00001) -
                  act_func[1](h1_ws - 0.00001)) / 0.00002
        rate = config.pretraining_rate_reba
    if config.with_adaptive_rate:
        b_h = h0 * ((v1 * v0).sum() + 1) - h1 * (1 +
            v1 * v1).sum())
        b_v = v0 * (1 + (h0 * h0).sum()) - v1 * (1 +
            h0 * h1).sum())
        rate = (((v0-v1) * b_v).sum() + ((h0-h1) * b_h)
            .sum()) / ((b_h*b_h).sum() + (b_v*b_v).sum())
    part_v = (v1 - v0) * der_v
    part_h = (h1 - h0) * der_h
    delta_weights = delta_weights * momentum + rate /
        config.pretraining_batch_size * (
            torch.mm(part_v.T, h0) + torch.mm(v1.T,
                part_h))
    delta_v_thresholds = delta_v_thresholds * momentum
        + rate / config.pretraining_batch_size * part_v.
        sum(0)
    delta_h_thresholds = delta_h_thresholds * momentum
        + rate / config.pretraining_batch_size * part_h.
        sum(0)
    rbm.W -= delta_weights
    rbm.v -= delta_v_thresholds
    rbm.h -= delta_h_thresholds
    part_loss = ((v1 - v0) ** 2).sum()
    loss += part_loss.item()
    i += 1
    print(loss)
    losses.append(loss)
return losses, h0.shape

def train_crpbm(rbm, device, batches_count, train_set,
    pretrain_type):
    delta_weights = torch.zeros(rbm.W.shape).to(device)
    losses = []
    act_func = rbm.a_func
    for epoch in range(config.pretraining_epochs):
        loss = 0.0
        i = 0
        momentum = config.momentum_beg if epoch < config.

```

```

        momentum_change_epoch else config.momentum_end
    while i < batches_count:
        inputs = train_set[i * config.
            pretraining_batch_size:(i + 1) * config.
            pretraining_batch_size].to(device)
        v0, v1, v1_ws, h0, h0_ws, h1, h1_ws = rbm(inputs)
        if pretrain_type == PretrainingType.RBMClassic:
            der_v, der_h = 1, 1
            rate = config.pretraining_rate
        if pretrain_type == PretrainingType.REBA:
            der_v = (act_func[0](v1_ws+0.00001) - act_func
                [0](v1_ws-0.00001)) / 0.00002
            der_h = (act_func[1](h1_ws+0.00001) - act_func
                [1](h1_ws-0.00001)) / 0.00002
            rate = config.pretraining_rate_reba
        part_v = (v1 - v0) * der_v
        part_h = (h1 - h0) * der_h
        first_convolution_part = torch.convolution(
            torch.permute(part_v, (1, 0, 2, 3)),
            torch.permute(h0, (1, 0, 2, 3)), None, stride
            =[1,1], padding=[0,0], dilation=[1,1],
            transposed=False, output_padding=[0,0],
            groups=1)
        second_convolution_part = torch.convolution(
            torch.permute(v1, (1, 0, 2, 3)),
            torch.permute(part_h, (1, 0, 2, 3)), None,
            stride=[1,1], padding=[0,0], dilation=[1,1],
            transposed=False, output_padding=[0,0],
            groups=1)
        common_conv_expr = first_convolution_part +
            second_convolution_part

        delta_weights = delta_weights * momentum + rate /
            config.pretraining_batch_size * torch.permute(
            common_conv_expr, (1, 0, 2, 3))
        rbm.W -= delta_weights
        loss += ((v1 - v0) ** 2).sum()
        i += 1
        print(loss.item())
        losses.append(loss.item())
    return losses, h0.shape

def train_torch_model(model, train_loader, test_loader,
    optimizer, criterion, device):
    best_total_accuracy = 0
    losses = []

```

```

for epoch in range(config.finetuning_epochs):
    running_loss = 0.0
    for i, data in enumerate(train_loader, 0):
        inputs, labels = data[0].to(device), data[1].to(
            device)

        optimizer.zero_grad()

        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
    if epoch % config.test_every_epochs == 0:
        current_accuracy = test_torch_model(model,
                                              test_loader, device)
        if current_accuracy > best_total_accuracy:
            best_total_accuracy = current_accuracy
    losses.append(running_loss)
    print(running_loss)
return best_total_accuracy, losses

def test_torch_model(model, test_loader, device):
    correct_answers = 0
    with torch.no_grad():
        for data in test_loader:
            images, labels = data[0].to(device), data[1].to(
                device)
            outputs = model(images)
            _, predictions = torch.max(outputs, 1)
            correct_answers += (predictions == labels).sum()
    return 100 * float(correct_answers) / len(test_loader.
                                                dataset)

def run_experiment(layers_config, pretrain_type, meta_data,
device, init_type, without_sampling):
    rbm_stack = RBMStack(layers_config, device, init_type,
                         without_sampling)
    layers_losses = None
    train_loader = meta_data[2]
    train_set = data_config.get_tensor_dataset_from_loader(
        train_loader)
    if pretrain_type != PretrainingType.Without:
        layers_losses = rbm_stack.train(train_set,

```

```
    pretrain_type)
if config.with_reduction:
    rbm_stack.do_reduction(layers_config)

classifier = UnifiedClassifier(layers_config).to(device)
rbm_stack.torch_model_init_from_weights(classifier)

criterion = nn.CrossEntropyLoss(reduction="sum")
optimizer = optim.SGD(classifier.parameters(), lr=config.
    finetune_rate, momentum=config.finetuning_momentum,
    weight_decay=1e-6)

test_loader = meta_data[3]
best_total_acc, losses = train_torch_model(classifier,
    train_loader, test_loader, optimizer, criterion, device)

return Statistics.get_train_statistics(layers_losses,
    best_total_acc), losses
```

# ПРИЛОЖЕНИЕ Б

## ДОКУМЕНТЫ ОБ ИСПОЛЬЗОВАНИИ РЕЗУЛЬТАТОВ ДИССЕРТАЦИОННОЙ РАБОТЫ

Адкрытае акцыянернае  
таварыства «Савушкін прадукт»  
(ААТ «Савушкін прадукт»)  
вул. Янкі Купалы, 118, 224028, г. Брэст,  
тэл. +375 (162) 47 06 24, факс 47 10 80,  
e-mail: info@savushkin.by  
URL: <https://www.savushkin.by/>  
р/р ВY04 BELB 3012 1500 7502 8022 6000  
у ААТ «Банк БелЗЭБ», код BELBBY2X  
УНП 200030514, АКПА 05882743



Открытое акционерное  
общество «Савушкін продукт»  
(ОАО «Савушкін продукт»)  
ул. Янки Купалы, 118, 224028, г. Брест,  
тел. +375 (162) 47 06 24, факс 47 10 80,  
e-mail: info@savushkin.by  
URL: <https://www.savushkin.by/>  
р/сч ВY04 BELB 3012 1500 7502 8022  
6000  
в ОАО «Банк БелВЭБ», код BELBBY2X  
УНП 200030514, ОКПО 05882743

17.05.2023 №  
На № от

Учреждение образования "Брестский  
государственный технический  
университет"

### Акт внедрения результатов диссертационного исследования

Мы, нижеподписавшиеся, директор по ИТ ОАО «Савушкін Продукт» Таберко В.В. и начальник БПР ОАО «Савушкін продукт» Иванюк Д.С. с одной стороны, и соискатель Крощенко А.А., с другой стороны, составили настоящий АКТ ВНЕДРЕНИЯ результатов диссертационного исследования на тему «Методы обучения глубоких нейронных сетей для задач компьютерного зрения» (научн. руководитель д.т.н., проф. Головко В.А.).

Разработанная Крощенко А.А. нейросетевая интеллектуальная система, а также соответствующие методы обучения и инструментальные средства внедрены и используются предприятием для контроля качества маркировки продукции на поточной линии.

Интеллектуальная система включает в себя комплекс модулей, формирующий конвейер обработки входного изображения из видеопотока. Разработанные программные модули базируются на применении нейросетевых классификаторов и детекторов (5 моделей). Кроме этого, система включает в себя средства обучения и управления работой моделей.

Достоинством указанной системы, методов обучения и средств в отличие от применявшимся на предприятии ранее, является их гибкость, позволяющая осуществлять дообучение и интеграцию новых модулей для распознавания различных типов маркировок, а также высокие показатели скорости работы и качества распознавания (99 % при обнаружении отдельных товаров и маркировки, 92% при обнаружении отдельных цифр для алфавитно-цифрового типа маркировки).

Соискатель

  
А.А. Крощенко

Начальник бюро перспективных  
разработок

  
Д.С. Иванюк

Директор по ИТ

  
В.В. Таберко



МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ  
"БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ"

УТВЕРЖДАЮ

Проректор по учебной работе



Базенков Т.Н.  
20 г.

М.П.

УТВЕРЖДАЮ

Проректор по научной работе



Рубанов В.С.  
20 г.

М.П.

АКТ ВНЕДРЕНИЯ

результатов НИР в учебный процесс

Мы, нижеподписавшиеся, начальник НИЧ Макарук Д.Г., научный руководитель д.т.н., профессор Головко В.А., исполнитель аспирант кафедры ИИТ 2-го года обучения Крощенко А.А. с одной стороны и начальник учебно-методического отдела Сторожук Н.Ю., декан ФЭИС к.ф.-м.н., доцент Ракецкий В.М. и заведующий кафедрой ИИТ д.т.н., профессор Головко В.А. с другой стороны составили настоящий *Акт внедрения* результатов научно-исследовательской работы «Нейросетевые методы обработки комплексной информации и принятия решений на основе интеллектуальных многоагентных систем», выполненной в рамках задания ГРНИ «Информатика и космос, научное обеспечение безопасности и защиты от чрезвычайных ситуаций 1.16», № ГР 20140547, научный руководитель – д.т.н., профессор Головко В.А.

Основные результаты работы:

- разработан альтернативный подход обучения нейронных сетей глубокого доверия, построенной на основе ограниченных машин Больцмана
- для подкрепления теоретических результатов проведены вычислительные эксперименты по исследованию широко известных выборок обучения (MNIST, CIFAR-10, NORB).

Указанная работа внедрена в учебный процесс в 2014-2015 уч. г. на кафедре интеллектуальных информационных технологий в качестве лекционного и лабораторного курсов для студентов специальностей 1-40 03 01 «Искусственный интеллект» и 1-53 01 02 «Автоматизированные системы обработки информации»

Наименование объекта внедрения и характеристика его преимущества: на простой теоретической базе разработан альтернативный подход к обучению нейронных сетей глубокого доверия, позволяющий активизировать решение совокупности актуальных научно-исследовательских задач.

Учебно-методический эффект:

- освоение современных и востребованных теоретических знаний, нацеленных на решение актуальных научно-технических задач;
- активизация научно-исследовательской инициативы студентов.

Начальник НИЧ

Научный руководитель

/ Д.Г. Макарук

Исполнитель

/ В.А. Головко

/ А.А. Крощенко

Начальник учебного отдела

Декан факультета

/ Н.Ю. Сторожук

Заведующий кафедрой

/ В.М. Ракецкий



**Semantic**  
intelligent semantic systems

**АКТ ВНЕДРЕНИЯ**  
**результатов диссертационного исследования**

г. Минск

31.03.2023 г.

Мы, нижеподписавшиеся, директор ООО «Интелиджент семантик системс» Грюневальд Т. и технический руководитель проектов ООО «Интелиджент семантик системс» Ковалев М.В. с одной стороны, и представитель БрГТУ, соискатель Кроценко А.А. с другой стороны, составили настоящий АКТ ВНЕДРЕНИЯ результатов диссертационной работы на тему «Методы обучения глубоких нейронных сетей для задач компьютерного зрения» (научн. руководитель д.т.н., проф. Головко В.А.).

Разработанные алгоритмы классификации и локализации солнечных панелей на аэрофотоснимках, методы обучения нейросетевых моделей и инструментальные средства внедрены и используются на предприятии ООО «Интелиджент семантик системс» при разработке интеллектуальной диалоговой программной системы в составе подсистемы компьютерного зрения.

Применение разработанных соискателем алгоритмов, методов обучения и инструментальных средств обеспечило возможность повышения точности классификации объектов на изображениях, которая при одинаковых условиях оказалась выше, чем при использовании традиционных подходов.

Директор ООО  
«Интелиджент семантик системс»

  
T. Грюневальд

Соискатель

  
A.A. Кроценко

Технический руководитель проектов

  
M.B. Ковалев