

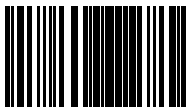
Министерство науки и высшего образования Российской Федерации  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО  
ITMO University

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА  
GRADUATION THESIS

Транскодирование JPEG-изображений при помощи предсказания коэффициентов  
ДКП на основе нейронной сети

**Обучающийся / Student** Подцепко Игорь Сергеевич  
**Факультет/институт/кластер/ Faculty/Institute/Cluster** факультет информационных технологий и программирования  
**Группа/Group** М34381  
**Направление подготовки/ Subject area** 01.03.02 Прикладная математика и информатика  
**Образовательная программа / Educational program** Информатика и программирование 2020  
**Язык реализации ОП / Language of the educational program** Русский  
**Квалификация/ Degree level** Бакалавр  
**Руководитель ВКР/ Thesis supervisor** Беляев Евгений Александрович, PhD, науки, Университет ИТМО, факультет информационных технологий и программирования, доцент (квалификационная категория "ординарный доцент")

Обучающийся/Student

Документ подписан	
Подцепко Игорь Сергеевич	
15.05.2024	

(эл. подпись/ signature)

Подцепко Игорь  
Сергеевич

(Фамилия И.О./ name  
and surname)

Руководитель ВКР/  
Thesis supervisor

Документ подписан	
Беляев Евгений Александрович	
15.05.2024	

(эл. подпись/ signature)

Беляев Евгений  
Александрович

(Фамилия И.О./ name  
and surname)

**Министерство науки и высшего образования Российской Федерации**  
**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ**  
**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**  
**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**  
**ITMO University**

**ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ /**  
**OBJECTIVES FOR A GRADUATION THESIS**

**Обучающийся / Student** Подцепко Игорь Сергеевич  
**Факультет/институт/кластер/ Faculty/Institute/Cluster** факультет информационных технологий и программирования  
**Группа/Group** М34381  
**Направление подготовки/ Subject area** 01.03.02 Прикладная математика и информатика  
**Образовательная программа / Educational program** Информатика и программирование 2020  
**Язык реализации ОП / Language of the educational program** Русский  
**Квалификация/ Degree level** Бакалавр  
**Тема ВКР/ Thesis topic** Транскодирование JPEG-изображений при помощи предсказания коэффициентов ДКП на основе нейронной сети  
**Руководитель ВКР/ Thesis supervisor** Беляев Евгений Александрович, PhD, науки, Университет ИТМО, факультет информационных технологий и программирования, доцент (квалификационная категория "ординарный доцент")

**Характеристика темы ВКР / Description of thesis subject (topic)**

**Тема в области фундаментальных исследований / Subject of fundamental research:** нет / not

**Тема в области прикладных исследований / Subject of applied research:** да / yes

**Основные вопросы, подлежащие разработке / Key issues to be analyzed**

Техническое задание: Разработать и обучить нейронную сеть, которая предсказывает часть коэффициентов дискретного косинусного преобразования (ДКП) из входного JPEG-изображения, и использовать полученные коэффициенты для энтропийного кодирования исходных коэффициентов.

Цель работы: Изучить возможность и потенциал нейронных сетей для реализации этапа внутреннего предсказания коэффициентов ДКП при транскодировании JPEG-изображений.

Задачи работы:

1. Реализовать декодирование JPEG-изображений с удалением (обнулением) части коэффициентов ДКП для подготовки набора данных для обучения нейронной сети;
2. Разработать и обучить нейронную сеть, которая предсказывает (вычисляет) часть коэффициентов ДКП на основании остальных коэффициентов;
3. Реализовать кодирование с предсказанием для сжатия выбранной части коэффициентов ДКП;
4. Оценить степень сжатия, которую удастся достичь благодаря нейросетевому

внутреннему предсказанию.

Рекомендуемые материалы:

- С. Sun, X. Fan and D. Zhao, "Lossless Recompression of JPEG Images Using Transform Domain Intra Prediction," in IEEE Transactions on Image Processing, vol. 32, pp. 88-99, 2023.

**Дата выдачи задания / Assignment issued on: 31.01.2024**

**Срок представления готовой ВКР / Deadline for final edition of the thesis 15.05.2024**

**СОГЛАСОВАНО / AGREED:**

Руководитель ВКР/  
Thesis supervisor

Документ подписан	
Беляев Евгений Александрович	
15.05.2024	

(эл. подпись)

Беляев Евгений  
Александрович

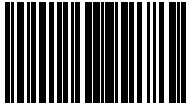
Задание принял к  
исполнению/ Objectives  
assumed BY

Документ подписан	
Подцепко Игорь Сергеевич	
15.05.2024	

(эл. подпись)

Подцепко Игорь  
Сергеевич

Руководитель ОП/ Head  
of educational program

Документ подписан	
Станкевич Андрей Сергеевич	
25.05.2024	

(эл. подпись)

Станкевич  
Андрей  
Сергеевич

**Министерство науки и высшего образования Российской Федерации  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО  
ITMO University**

**АННОТАЦИЯ  
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ  
SUMMARY OF A GRADUATION THESIS**

**Обучающийся / Student** Подцепко Игорь Сергеевич  
**Факультет/институт/кластер/ Faculty/Institute/Cluster** факультет информационных технологий и программирования  
**Группа/Group** М34381  
**Направление подготовки/ Subject area** 01.03.02 Прикладная математика и информатика  
**Образовательная программа / Educational program** Информатика и программирование 2020  
**Язык реализации ОП / Language of the educational program** Русский  
**Квалификация/ Degree level** Бакалавр  
**Тема ВКР/ Thesis topic** Транскодирование JPEG-изображений при помощи предсказания коэффициентов ДКП на основе нейронной сети  
**Руководитель ВКР/ Thesis supervisor** Беляев Евгений Александрович, PhD, науки, Университет ИТМО, факультет информационных технологий и программирования, доцент (квалификационная категория "ординарный доцент")

**ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ  
DESCRIPTION OF THE GRADUATION THESIS**

**Цель исследования / Research goal**

Изучить возможность и потенциал нейронных сетей для реализации этапа внутреннего предсказания коэффициентов ДКП при транскодировании JPEG-изображений.

**Задачи, решаемые в ВКР / Research tasks**

1. Реализовать декодирование JPEG-изображений с удалением (обнулением) части коэффициентов ДКП для подготовки набора данных для обучения нейронной сети; 2. Разработать и обучить нейронную сеть, которая предсказывает (вычисляет) часть коэффициентов ДКП на основании остальных коэффициентов; 3. Реализовать кодирование с предсказанием для сжатия выбранной части коэффициентов ДКП; 4. Оценить степень сжатия, которую удастся достичь благодаря нейросетевому внутреннему предсказанию.

**Краткая характеристика полученных результатов / Short summary of results/findings**

Разработан метод внутреннего предсказания коэффициентов ДКП для транскодирования JPEG-изображений, основанный на применении нейронной сети и позволяющий сократить размер JPEG-файлов из подмножества набора данных ImageNet в среднем на 3,33% при медиане 3,38%. Наилучшее достигнутое сжатие составило 6,28%. Метод успешно комбинируется с утилитой Jpegtran, позволяющей заменять код Хаффмана на арифметическое кодирование, и общая степень сжатия в среднем составляет 13,16%.

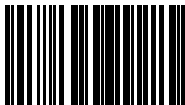
подписан	
Подцепко Игорь Сергеевич	
15.05.2024	

(эл. подпись/ signature)

Подцепко Игорь  
Сергеевич

(Фамилия И.О./ name  
and surname)

Руководитель ВКР/  
Thesis supervisor

Документ подписан	
Беляев Евгений Александрович	
15.05.2024	

(эл. подпись/ signature)

Беляев Евгений  
Александрович

(Фамилия И.О./ name  
and surname)

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	6
1. Постановка задачи о транскодировании JPEG-изображений и обзор существующих решений.....	9
1.1. Алгоритм JPEG.....	9
1.1.1. Преобразование цветового пространства .....	9
1.1.2. Цветовая субдискретизация.....	10
1.1.3. Деление на блоки и дискретное косинусное преобразование .....	11
1.1.4. Квантование.....	11
1.1.5. Кодирование DC-коэффициентов.....	11
1.1.6. Энтропийное кодирование.....	12
1.2. Эффективные методы сжатия изображений .....	13
1.3. Транскодирование JPEG-изображений с потерями.....	14
1.4. Транскодирование JPEG-изображений без потерь .....	15
1.4.1. Общие методы.....	15
1.4.2. Jpegtran.....	16
1.4.3. LLJPEG.....	17
Выводы по главе 1 .....	17
2. Транскодирование JPEG-изображений на основе нейросетевого внутреннего предсказания .....	19
2.1. Мотивация .....	19
2.2. Обзор предлагаемого алгоритма.....	21
2.2.1. Транскодирование .....	21
2.2.2. Трансдекодирование.....	22
2.3. Обнуление коэффициентов .....	23
2.4. Внутреннее предсказание коэффициентов.....	24
2.4.1. AR-CNN.....	24
2.4.2. QE-CNN-P.....	25
2.4.3. Функция потерь .....	25
Выводы по главе 2 .....	25
3. Реализация и тестирование .....	27
3.1. Разработка транскодера JPEG .....	27
3.1.1. Публичное C++ API транскодера .....	27
3.1.2. Интерфейс командной строки .....	28

3.1.3. Детали реализации этапа обнуления коэффициентов .....	29
3.1.4. Детали реализации кодирования коэффициентов .....	29
3.2. Разработка нейронной сети.....	30
3.2.1. Реализация QE-CNN-P .....	30
3.2.2. Подготовка набора данных .....	31
3.2.3. Параметры обучения .....	32
3.2.4. Интерфейс командной строки .....	32
3.3. Тестирование.....	33
3.3.1. Оценка качества работы нейронной сети .....	33
3.3.2. Подготовка набора транскодированных и трансдекодированных изображений.....	34
3.3.3. Детали реализации end-to-end тестирования.....	36
3.3.4. Оценка степени сжатия.....	36
3.3.5. Дальнейшие перспективы исследования.....	38
Выводы по главе 3 .....	39
ЗАКЛЮЧЕНИЕ .....	40
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	41

## ВВЕДЕНИЕ

JPEG — это стандарт сжатия изображений, который был разработан в 1992 году Joint Photographic Experts Group. До настоящего времени он остается наиболее распространенным форматом сжатия изображений из-за своей низкой вычислительной сложности, однако качество сжатия JPEG значительно уступает более современным алгоритмам, таким как WebP, HEIF, AVIF и т. д.

В современном мире цифровых устройств фотографии, сохраненные в формате JPEG, стали неотъемлемой частью социальной жизни. С каждым днем все больше людей используют социальные сети, такие как ВКонтакте и Telegram, для обмена сообщениями и изображениями. Эти веб-сайты хранят огромное число JPEG-изображений на своих серверах. Более того, массовое использование JPEG-изображений приводит к увеличению спроса на локальные и облачные хранилища. Следовательно, исследование способов снижения затрат на хранение JPEG-изображений в облачном хранилище или на персональных компьютерах становится все более важной и актуальной задачей. При этом чаще всего речь идет именно о более эффективном хранении без потерь, так как если пользователь загружает в облачное хранилище изображение — он ожидает получить его в будущем в неизменном виде.

Для решения проблемы хранения больших объемов изображений JPEG транскодер может использоваться, как показано на рисунке 1. При загрузке в облачное хранилище JPEG файл дополнительно сжимается без потерь транскодером, а при запросе от пользователя трансдекодируется в исходное представление.

Настоящая работа рассматривает алгоритм внутреннего предсказания ДКП-коэффициентов с помощью нейронной сети. В процессе работы предлагаемого алгоритма JPEG изображение декодируется до коэффициентов ДКП, после чего часть из них обнуляется. Такое преобразование вносит в изображения определенные искажения, которые могут исправляться специально обученной нейронной сетью. В области коэффициентов ДКП это соответствует появлению на местах обнуленных коэффициентов значений близких к исходным. Далее на места обнуленных коэффициентов записываются ошибки их предсказания, и изображение кодируется энтропийным кодером из алгоритма JPEG. На выходе транскодера получается корректный файл JPEG, но меньшего



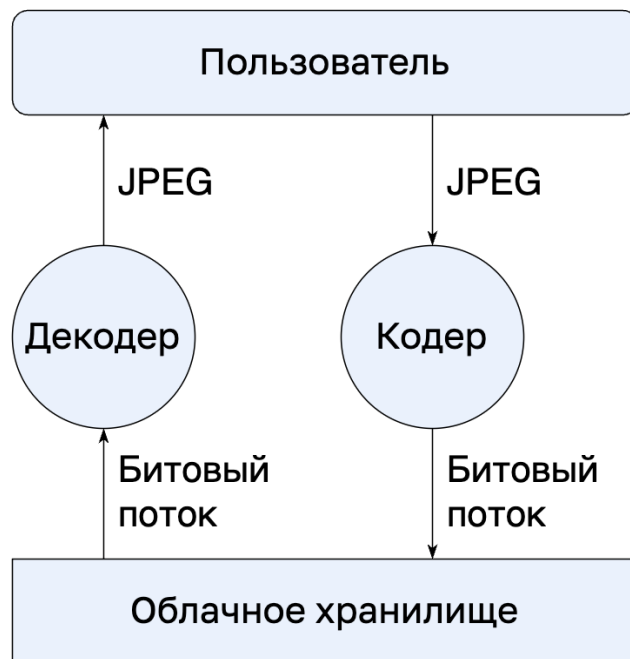


Рисунок 1 – Архитектура внедрения транскодера JPEG

объема. Данный подход позволяет лучше устранять избыточность, возникающую между блоками изображения, так как реализует предсказание не только DC, но и AC-коэффициентов. Метод является совместимым с другими существующими транскодерами JPEG и в среднем позволяет добиться сжатия на 3,33%.

Цель работы: Изучить возможность и потенциал нейронных сетей для реализации этапа внутреннего предсказания коэффициентов ДКП при транскодировании JPEG-изображений.

Задачи работы:

- а) Реализовать декодирование JPEG-изображений с удалением (обнулением) части коэффициентов ДКП для подготовки набора данных для обучения нейронной сети;
- б) Разработать и обучить нейронную сеть, которая предсказывает (вычисляет) часть коэффициентов ДКП на основании остальных коэффициентов;
- в) Реализовать кодирование с предсказанием для сжатия выбранной части коэффициентов ДКП;
- г) Оценить степень сжатия, которую удастся достичь благодаря нейросетевому внутреннему предсказанию.

Настоящая работа имеет следующую структуру. В первой главе рассматриваются существующие алгоритмы сжатия изображений: JPEG, JPEG2000, HEVC, JPEG-LS и другие. Также в этой главе приводится краткая сводка относительно существующих на сегодняшний день подходов к транскодированию JPEG-изображений с потерями и без потерь, формулируются основные подходы к решению данной задачи. Вторая глава содержит мотивацию, краткий обзор и более подробное описание предлагаемого алгоритма транскодирования. В третьей главе приводится детальная информация о том, как было реализовано декодирование JPEG-изображений с обнулением коэффициентов, описываются детали реализации и обучения нейронной сети, а также содержатся результаты экспериментов по оценке степени сжатия предлагаемым алгоритмом и сравнение его с другими подходами.

## ГЛАВА 1. ПОСТАНОВКА ЗАДАЧИ О ТРАНСКОДИРОВАНИИ JPEG-ИЗОБРАЖЕНИЙ И ОБЗОР СУЩЕСТВУЮЩИХ РЕШЕНИЙ

В данной главе приводится постановка задачи транскодирования JPEG-изображений и рассматривается непосредственно алгоритм JPEG. Описывается схема и принцип работы транскодера, основанного на применении методов машинного обучения.

### 1.1. Алгоритм JPEG

При сжатии алгоритмом JPEG [1] изображение проходит несколько этапов преобразования, показанных на рисунке 2. В данном разделе рассматривается каждый из этапов кодирования.

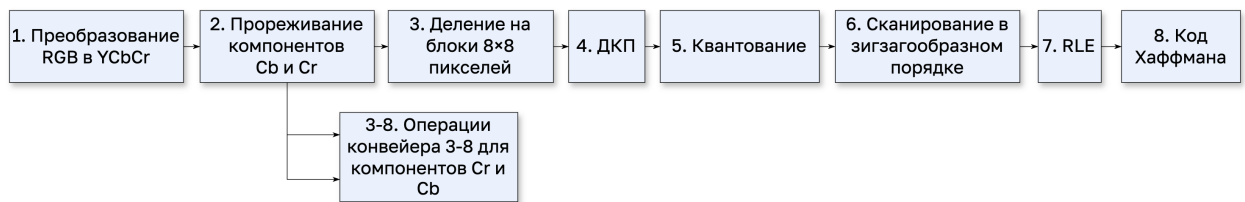


Рисунок 2 – Стадии алгоритма JPEG

#### 1.1.1. Преобразование цветового пространства

На первом этапе работы JPEG изображение переводится из цветового пространства RGB в YCbCr:

$$\begin{cases} Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B - 128 \\ Cb = -0.16874 \cdot R - 0.33126 \cdot G + 0.5 \cdot B \\ Cr = 0.5 \cdot R - 0.41869 \cdot G - 0.08131 \cdot B \end{cases}$$

Компонент Y (luminance) представляет собой яркостную характеристику пикселя и определяет его интенсивность. Он кодирует черно-белую информацию изображения. Компоненты Cb (chroma blue) и Cr (chroma red) вычисляются как разности между цветом пикселя и его яркостью, то есть являются цветовыми компонентами (или каналами). Визуализация данного представления показана на рисунке 3.

Преимущество YCbCr состоит в том, что в силу особенностей зрительной системы человека значимость яркости превышает значимость цвета. Другими словами, наш глаз лучше фиксирует изменения именно в яркостном компоненте. Это говорит о психовизуальной избыточности. По этой причине на



Рисунок 3 – Слева направо: исходное изображение, Y, Cb, Cr

втором шаге алгоритма JPEG выполняется прореживание (субдискретизация) цветowych компонентов.

### 1.1.2. Цветовая субдискретизация

Цветовая субдискретизация — это уменьшение размерностей цветowych каналов, с целью снижения размера итогового цифрового потока. Фактически это означает, что частота выборки для цветowych компонентов оказывается меньше, чем для яркостного компонента.

Существуют различные форматы субдискретизации (рисунок 4). Структура дискретизации изображения описывается соотношением между тремя частями  $n : m_1 : m_2$ . Этими частями являются:

- $n$  — частота дискретизации канала Y (ширина макропикселя);
- $m_1$  — число выборок каналов Cr, Cb в горизонтальном направлении в первой строке;
- $m_2$  — число выборок каналов Cr, Cb в горизонтальном направлении во второй строке;

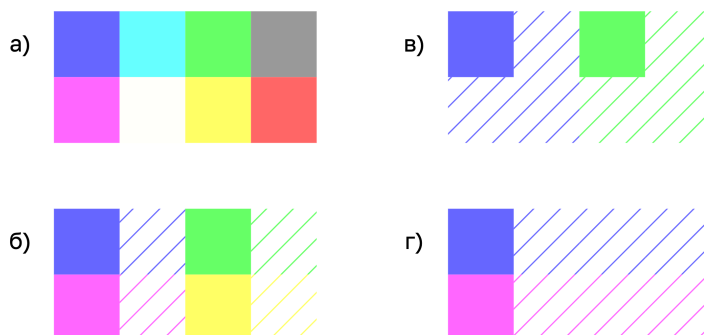


Рисунок 4 – Различные форматы субдискретизации:

а) 4 : 4 : 4; б) 4 : 2 : 2; в) 4 : 2 : 0; г) 4 : 1 : 1

### 1.1.3. Деление на блоки и дискретное косинусное преобразование

После прореживания каждый из компонентов делится на блоки  $8 \times 8$  пикселей, которые кодируются отдельно и независимо друг от друга. К ним применяется двумерное дискретное косинусное преобразование (ДКП) — метод преобразования сигналов (изображений) из пространства времени (пикселей) в пространство частот, использующее косинусные функции для разложения сигнала на сумму синусоид с разными частотами. Вычисляется ДКП по следующей формуле:

$$g(x, y, u, v) = h(x, y, u, v) = a(u)a(v) \cos\left(\frac{(2x+1)u\pi}{2N}\right) \cos\left(\frac{(2y+1)v\pi}{2N}\right),$$

где

$$a(u) = \begin{cases} \frac{1}{\sqrt{N}}, & u = 0, \\ \frac{2}{\sqrt{N}}, & u \neq 0. \end{cases}$$

Коэффициенты, полученные после ДКП, разделяют на:

- DC-коэффициент (или коэффициент постоянной составляющей), находящийся в верхнем левом углу блока — компонент, отвечающий за среднее значение яркости или интенсивности цвета блока;
- AC-коэффициенты — высокочастотные компоненты блока, причем чем больше сумма индексов коэффициента — тем более высокочастотный компонент он описывает.

### 1.1.4. Квантование

Следующим этапом работы алгоритма JPEG является квантование — это поэлементное деление коэффициентов ДКП на матрицы квантования с округлением вниз. Как правило, для каждого компонента изображения используется собственная матрица квантования. Абсолютные значения элементов матриц зависят от параметра качества (quality factor, QF), передаваемого кодеру JPEG, однако обычно более высокочастотные коэффициенты ДКП квантуются сильнее. Более того, на практике многие из них обнуляются.

### 1.1.5. Кодирование DC-коэффициентов

После квантования применяется кодирование с предсказанием: каждый DC-коэффициент (кроме первого) заменяется на разность между квантован-

ным текущим и предыдущим DC-коэффициентами из того же канала. Преимущество такого подхода по сравнению с кодированием самих значений DC-коэффициентов состоит в том, что он позволяет устранять избыточность между соседними блоками, возникающую из-за того, что в реальных изображениях можно наблюдать самоподобие и средние значения яркости/цвета соседних блоков часто близки друг к другу.

### 1.1.6. Энтропийное кодирование

После этапа квантования и внутреннего предсказания DC-коэффициентов применяется кодирование длин серий (run-level encoding, RLE), в процессе которого коэффициенты сканируются в зигзагообразном порядке, показанном на рисунке 5, и последовательности из  $run$  нулей и ненулевого коэффициента со значением  $x$  заменяются на пару  $(run, level)$ , где  $level$  — длина двоичного представления  $x$  в прямом коде.

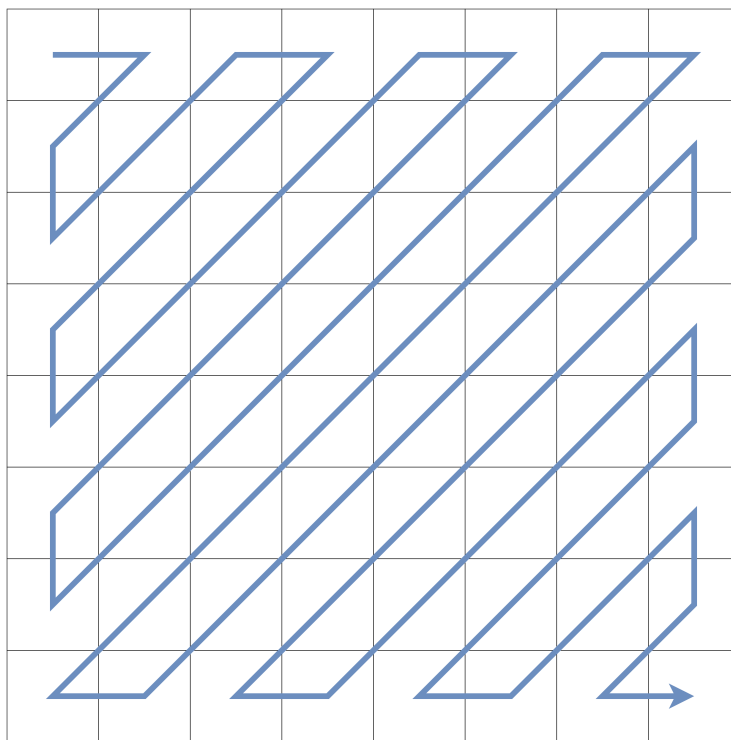


Рисунок 5 – Порядок сканирования коэффициентов ДКП

Множество различных пар  $(run, level)$  представляет собой дискретный ансамбль [2] — некоторое множество  $X$ , в котором каждому элементу  $x \in X$  приписана вероятность  $p(x)$ , определяемая частотой, с которой данная пара встречается в компоненте изображения, что соответствует собственной информации  $I(x) = -\log p(x)$ . Эффективность побуквен-

ного кодирования компонента определяется его энтропией — величиной  $\mathbb{E}(-\log p(x)) = -\sum_{x \in X} p(x) \log p(x)$ . Чем ниже энтропия, тем более эффективно побуквенное кодирование, а оптимальным побуквенным кодом является код Хаффмана, основанный на идее о том, что пары  $(run, level)$  с более высокими вероятностями (содержащие меньшее значение собственной информации) должны иметь более короткие коды. Именно такой способ кодирования применяется в алгоритме JPEG. Каждая пара  $(run, level)$  кодируется кодом Хаффмана, а следом за ней в выходящий битовый поток записывается двоичное представление коэффициента ДКП в прямом коде.

Необходимо отметить, что способ построения таблицы Хаффмана при кодировании алгоритмом JPEG не задан стандартом. При практической реализации применяется один из двух подходов:

- а) Однопроходный, при котором используются универсальные таблицы Хаффмана, содержащие в себе кодовые слова для всех возможных пар  $(run, level)$ ;
- б) Двухпроходный, при котором таблицы Хаффмана строятся для конкретного изображения.

## 1.2. Эффективные методы сжатия изображений

На сегодняшний день в области сжатия изображений есть большие успехи. Так, например, следующие форматы позволяют достичь гораздо большей степени сжатия с потерями при заданном качестве картинки [3]:

- а) JPEG2000 [4] — алгоритм, аналогичный JPEG, но основанный на вейвлет-преобразовании (альтернативном методе преобразования сигналов (изображений) из пространства времени (пикселей) в пространство частот) и арифметическом кодировании;
- б) BPG (Better Portable Graphics) [5] — алгоритм основанный на видеокодеке HEVC (технологии кодирования ключевых кадров). В нем изображение делится на блоки разного, а не фиксированного (в отличие от JPEG) размера. Данный формат поддерживает сжатие с потерями и без, изображение с альфа-каналом (прозрачностью), использование метаданных и анимацию;
- в) WebP (WEB Pictures) [6] — алгоритм, аналогичный по функциональности BPG, но основанный на видеокодеке VP8. Изображение, закодиро-

ванное данным форматом на 25–34% меньше, чем идентичное по качеству JPEG-изображение.

Однако вышеперечисленные форматы не используются так широко, как JPEG, из-за своей высокой вычислительной сложности. Между тем существует ряд алгоритмов сжатия без потерь, например:

- а) JPEG-LS [7] — формат сжатия, представленный Joint Photographic Experts Group в дополнение к JPEG и JPEG2000, но ориентированный именно на сжатие без потерь. Реализует адаптивное предсказание значений пикселей по уже закодированным значениям, а также классификацию контекста, контекстное моделирование ошибки предсказания и её коррекцию. В качестве энтропийного кодирования ошибок предсказания применяются коды Голомба;
- б) CALIC (context-based, adaptive, lossless image codec) [8] — алгоритм, использующий большое число контекстов моделирования для нелинейного предиктора и его адаптации к изменяющейся статистике источника;
- в) FLIF (Free Lossless Image Format) [9] — формат, который по заверениям разработчиков превосходит по эффективности сжатия все перечисленные форматы. Данный метод использует вариацию арифметического кодирования в качестве энтропийного сжатия.

К сожалению, все они являются неэффективными для транскодирования JPEG-изображений без потерь. Связано это, в частности, с округлением значений Y, Cb, Cr и коэффициентов ДКП при кодировании и ограничении значений R, G, B при декодировании, о чем будет упомянуто ниже, в разделе 1.4.1. Несмотря на это, необходимо обратить внимание на то, что все эти подходы:

- а) Реализуют более сложные методы внутреннего предсказания, чем алгоритм JPEG, так как основаны на пространственной корреляции, наблюдаемой между соседними участками изображений;
- б) Используют более эффективные, чем код Хаффмана, подходы для энтропийного кодирования ошибок предсказания.

### **1.3. Транскодирование JPEG-изображений с потерями**

Возможность дополнительного сжатия именно JPEG-изображений выглядит привлекательной, причем это может быть как транскодирование с потерями, так и без. Первую задачу успешно выполняют такие программы как:



- а) TinyPNG [10], достигающий высокой степени сжатия за счет объединения цветов, изменения порядка сканирования и использования особенностей человеческого зрения;
- б) Mozjpeg [11], являющийся эффективным кодером JPEG от компании Mozilla и реализующий возможность транскодирования JPEG, закодированных другим кодером, в более эффективное представление;
- в) Guetzli [12], также является эффективным кодером JPEG, но от компании Google.

Несмотря на успехи в области сжатия с потерями, во многих прикладных сценариях сжатие оно недопустимо. Примером таких задач может быть хранение данных пользователя в облачных хранилищах.

#### **1.4. Транскодирование JPEG-изображений без потерь**

Утилиты, реализующие транскодирование без потерь, также существуют. Наиболее ранней и распространенной является Jpegtran [13, 14], которая, однако, далека от предела возможностей современных транскодеров, таких как LLJPEG [15, 16]. В данном разделе рассмотрены общие подходы к транскодированию изображений и упомянутые программы, выполняющие транскодирование файлов JPEG без потерь.

##### **1.4.1. Общие методы**

Общие положения, лежащие в основе любого транскодера JPEG, были выдвинуты авторами Jpegtran еще в 2005 году [13]. Основаны они на том факте, что при декодировании и повторном кодировании JPEG файла результат может быть отличным от исходного даже при фиксированном значении параметра  $QF$ . Рассмотрим более подробно, откуда возникают неконтролируемые потери.

Первыми этапами при декодировании JPEG-файлов являются применение энтропийного декодера (кода Хаффмана) и обратное квантование. На данных этапах не может возникать никаких дополнительных потерь. Далее выполняется обратное ДКП и на практике эта операция не является обратимой, так как после применения ДКП в алгоритме JPEG коэффициенты округляются до целых чисел. К тому же в зависимости от деталей реализации вычислений, методов округления могут возникать незначительные с точки зрения качества изображения, но не кодирования информации ошибки. Это же относится и к преобразованию цветового пространства.

Так, например, функция преобразования YCbCr в RGB может возвращать дробные, отрицательные значения или значения больше 255, иными словами — некорректные значения пикселей в RGB. Для устранения этих ошибок применяется округление до целого и функция

$$\text{clip}(x) = \begin{cases} 0, & x < 0, \\ 255, & x > 255, \\ x, & \text{otherwise.} \end{cases}$$

Таким образом, для транскодирования JPEG-изображений без потерь можно выполнять модификации алгоритма на этапах, следующих за ДКП — устранять избыточность между коэффициентами преобразования благодаря реализации механизмов внутреннего предсказания, а также дорабатывать этап энтропийного кодирования.

### 1.4.2. Jpegtran

Утилита Jpegtran была разработана в 2005 году и предназначалась в первую очередь для выполнения преобразований над JPEG-изображениями без потерь. Так, например, она поддерживает такие функции, как поворот (*rotate*), отражение (*flip*), транспонирование (*transpose*) и другие. Их выполнение возможно с помощью применения аналогичных преобразований над блоками изображения, таблицами квантования и т.д. Подобные трансформации приводят к изменению размеров файлов JPEG, причем как в большую, так и в меньшую сторону. Происходит это благодаря тому, что после выполнения каких-либо преобразований на вход энтропийному кодеру поступает отличная от исходной последовательность кодовых слов — пар (*run, level*). Фактически, возможность транскодирования JPEG-изображений с целью дополнительного сжатия — это удобный побочный эффект утилиты Jpegtran.

Более современные версии Jpegtran поддерживают также ряд преобразований, направленных именно на дополнительное сжатие (транскодирование) JPEG-изображений без потерь:

- а) Опция `-optimize` позволяет выполнять оптимизацию таблицы Хаффмана под конкретное изображение. Как было упомянуто в разделе 1.1.6, для ускорения кодирования JPEG могут использоваться универсальные таблицы Хаффмана, содержащие кодовые слова для любых возмож-

ных пар (*run*, *level*). Поскольку распределение вероятностей конкретного изображения может отличаться от универсального, для него может быть более эффективный код;

- б) Опция `-arithmetic` позволяет заменить код Хаффмана на арифметическое кодирование.

### 1.4.3. LLJPEG

Разработанный в 2023 году LLJPEG является одним из наиболее эффективных решений задачи о транскодировании файлов JPEG без потерь. Данный метод — это гибридный алгоритм, объединяющий сразу несколько эффективных подходов к транскодированию JPEG-изображений:

- а) При делении на блоки используется адаптивный размер, который итеративно вычисляется для каждого участка изображения. Начиная со значения  $64 \times 64$ , на каждой итерации рассчитывается стоимость кодирования блока и, пока это является эффективным, — блок уменьшается. Минимальный допустимый размер блока —  $4 \times 4$ ;
- б) В качестве метода внутреннего предсказания используется подход аналогичный внутреннему предсказанию видеокодека HEVC, где для вычисления значений в блоке используется наиболее эффективное направление из 35-и, указывающих на блоки слева и сверху. Необходимо отметить, что возможности такого подхода ограничены, так как для предсказания используется только локальная схожесть, наблюдаемая в изображении, но не учитывается подобие удаленных друг от друга участков. Если же учесть нелокальную схожесть пикселей, то гипотетически можно добиться более эффективного предсказания и, следовательно, устранения избыточности.
- в) Код Хаффмана заменяется более эффективным контекстно-адаптивным бинарным арифметическим кодером (context-adaptive binary arithmetic coding, CABAC).

Благодаря объединению вышеперечисленных техник эффективность сжатия LLJPEG достигает порядка 18% для параметра  $QF = 90$ , что на сегодняшний день является наилучшим показателем.

### Выводы по главе 1

В данной главе были подробно рассмотрены основные этапы работы алгоритма JPEG, а также проведен обзор других, более эффективных, методов

сжатия изображений с потерями и без, появившихся уже после широкого распространения JPEG. Далее были рассмотрены методы транскодирования изображений с потерями: TinyPNG, Mozjpeg, Guetzli. И наконец, был проведен анализ литературы на тему того, за счет чего может выполняться транскодирование и как эти подходы реализованы в существующих решениях: Jpegtran и LLJPEG. В результате было показано, что все современные алгоритмы сжатия изображений, позволяющие достигать более высокой степени сжатия, чем JPEG, а также методы транскодирования без потерь основаны на улучшении двух важных этапов: внутреннего предсказания коэффициентов ДКП и энтропийного кодирования. Второй из этих пунктов практически всегда подразумевает использование различных вариаций адаптивного арифметического кодирования, а первый в большинстве эффективных алгоритмов заимствован из видеокодека HEVC.

## ГЛАВА 2. ТРАНСКОДИРОВАНИЕ JPEG-ИЗОБРАЖЕНИЙ НА ОСНОВЕ НЕЙРОСЕТЕВОГО ВНУТРЕННЕГО ПРЕДСКАЗАНИЯ

В данной главе описывается предлагаемый алгоритм транскодирования JPEG-изображений, основанный на нейросетевом внутреннем предсказании.

### 2.1. Мотивация

Как было упомянуто в разделе 1.4.1. Эффективность сжатия зависит от нескольких факторов. С одной стороны, можно оптимизировать энтропийный кодер. Например, строить оптимальные таблицы Хаффмана для конкретного изображения двухпроходным алгоритмом, а не использовать универсальные, либо вовсе отказываться от кода Хаффмана в пользу арифметического кодирования. Кроме того, внутреннее предсказание позволяет уменьшать число ненулевых коэффициентов и их абсолютные значения. Это также положительно сказывается на степени сжатия, так как серии из нулей кодируются эффективно благодаря применению RLE, а меньшим по модулю коэффициентам обычно соответствуют более короткие кодовые слова.

В разделе 1.4 упоминалось, что задача внутреннего предсказания коэффициентов ДКП уже решается с использованием простых конечных алгоритмов, например, в видеокодеках VVC и HEVC, а также в LLJPEG (в последних двух случаях используются одинаковые подходы). Суть данных алгоритмов состоит в поиске зависимостей между кодируемым блоком и одним или несколькими соседними блоками из того же компонента. На рисунке 6 показано, как реализуется внутреннее предсказание в видеокодеке HEVC, а на рисунке 7 — пример применения планарного (planar) режима предсказания. Из приведенного примера видно, что внутреннее предсказание не учитывает нелокальную схожесть разных участков изображения.

В основе настоящей работы лежит идея о том, что для предсказания коэффициентов ДКП можно использовать нейронную сеть. Преимущество данного подхода по сравнению с уже существующими заключается в том, что модели машинного обучения способны искать более сложные взаимосвязи в изображении, например:

- между блоком изображения и его соседями (в любом из направлений);
- между яркостным (Y) и цветовыми (Cr, Cb) компонентами;
- между удаленными друг от друга участками изображения.

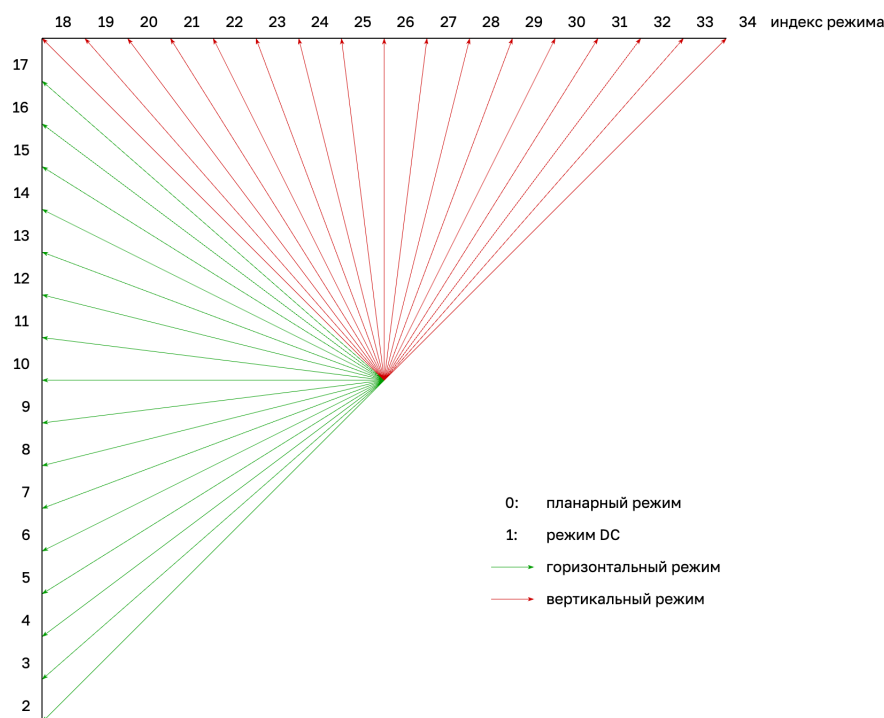


Рисунок 6 – Режимы внутреннего предсказания в видеокодеке HEVC; красные и зеленые линии представляют вертикальные и горизонтальные режимы; режим 0 - это планарный (planar) режим; а режим 1 - режим DC

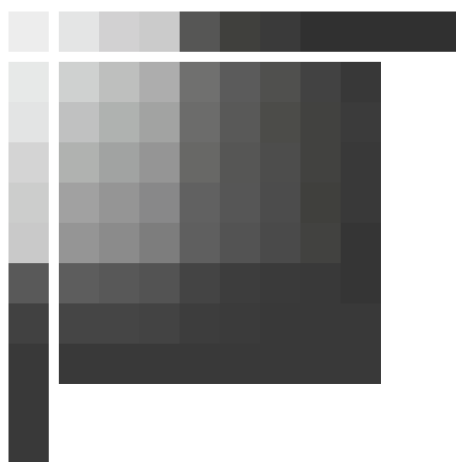


Рисунок 7 – Планарный (planar) режим внутреннего предсказания в видеокодеке HEVC

Таким образом, основную гипотезу можно сформулировать следующим образом: эффективность внутреннего предсказания на основе нейронной сети может превышать эффективность подходов, использующих локальные зависимости между соседними блоками.

## 2.2. Обзор предлагаемого алгоритма

В настоящей работе предлагается новый метод предсказания коэффициентов ДКП и алгоритм транскодирования JPEG-изображений, основанный на данном подходе.

### 2.2.1. Транскодирование

На рисунке 8 представлена общая схема транскодера.

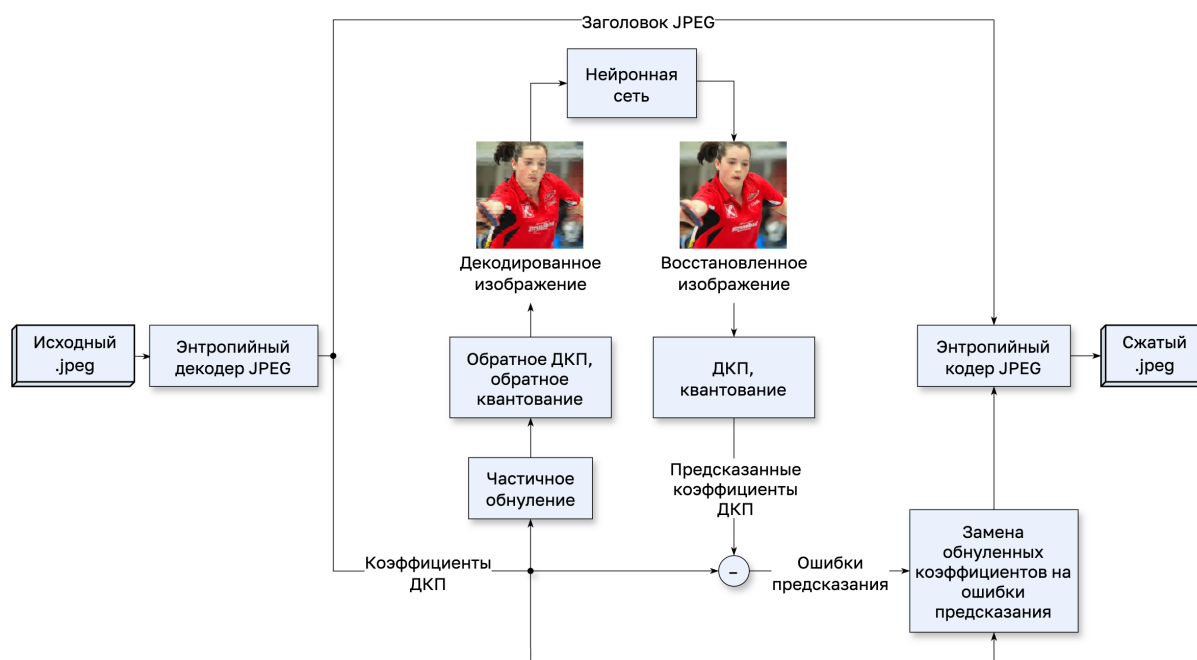


Рисунок 8 – Общая схема работы транскодера

При транскодировании к исходному изображению  $I$  применяется энтропийный декодер JPEG, а затем из него извлекаются квантованные коэффициенты ДКП. Далее они делятся псевдослучайным образом на две группы и коэффициенты первой группы обнуляются. Пример такого преобразования представлен на рисунке 9 (б).

После обнуления блока к оставшимся коэффициентам применяется обратное квантование и обратное ДКП, и изображение переводится в цветовое пространство RGB. Таким образом, фактически выполняется стандартное декодирование JPEG, но изображение оказывается искаженным ( $\tilde{I}$ ). Далее оно подается на вход нейронной сети  $M$ , которая обучена улучшать качество декодированных изображений. В области ДКП это означает, что на местах обнуленных коэффициентов  $x_{i,j}$  первой группы в обработанном моделью изоб-

а)

-23	19	-5	0	0	2	1	0
-7	12	2	-4	-5	0	0	0
14	-1	-5	0	1	0	-1	0
1	0	3	1	-1	0	-2	0
0	-3	-1	0	0	1	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	-1	0	0	0	0	0	0

б)

-23	19	-5	0	0	2	0	0
-7	12	2	-4	0	0	0	0
14	-1	0	0	1	0	-1	0
1	0	3	0	-1	0	0	0
0	0	-1	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

в)

-23	19	-5	0	0	2	1	0
-7	12	2	-4	-4	0	0	0
14	-1	-5	0	1	0	-1	0
1	0	3	0	-1	0	-1	0
0	-4	-1	0	0	1	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	-1	0	0	0	0	0	0

г)

-23	19	-5	0	0	2	0	0
-7	12	2	-4	-1	0	0	0
14	-1	0	0	1	0	-1	0
1	0	3	1	-1	0	-1	0
0	1	-1	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Рисунок 9 – Пример преобразований над блоком: а) исходный блок; б) блок после обнуления части коэффициентов; в) блок на выходе нейронной сети; г) блок с ошибками предсказания на местах исходных коэффициентов

ражении  $I' = M(\tilde{I})$  появляются похожие на них коэффициенты  $x'_{i,j}$  (рисунок 9 (в)). Таким образом, нейронная сеть предсказывает значения коэффициентов первой группы по коэффициентам из второй группы, то есть реализует внутреннее предсказание.

На следующем этапе все коэффициенты кодируются согласно алгоритму JPEG, однако вместо значений коэффициентов первой группы кодируется ошибка их предсказания  $\Delta x_{i,j} = x_{i,j} - x'_{i,j}$ . Если нейронной сети удастся в точности восстановить обнуленный коэффициент ( $x'_{i,j} = x_{i,j}$ ), то вместо него в блоке остается ноль, в противном случае коэффициент заменяется меньшим по модулю значением. Таким образом, достигается дополнительное сжатие JPEG-изображения.

### 2.2.2. Трансдекодирование

Схема трансдекодера представлена на рисунке 10 и аналогична схеме транскодера. При декодировании создается изображение  $\tilde{I}$ , у которого удале-



ны те же самые коэффициенты, что и при кодировании. Оно подается на вход нейронной сети  $M$  и на месте обнуленных коэффициентов появляются значения  $x'$ , к которым остается только добавить поправку  $\Delta x$ , закодированную на местах удаленных коэффициентов. В результате получаются  $x = x' + \Delta x$  — исходные значения коэффициентов ДКП, которые далее кодируются энтропийным кодером JPEG.

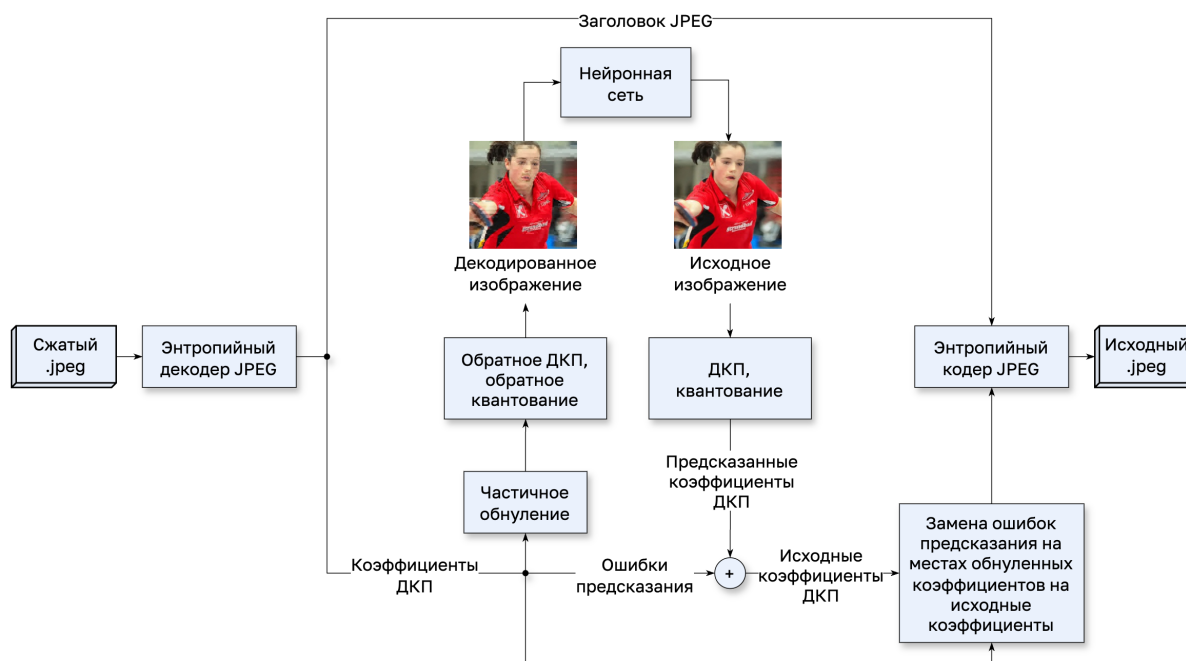


Рисунок 10 – Общая схема работы трансдекодера

### 2.3. Обнуление коэффициентов

Обнуление коэффициентов является одним из ключевых этапов транскодирования. К нему выдвигаются следующие требования:

- Детерминированность.* Список обнуленных коэффициентов должен быть известен при декодировании, а значит, независимо от изображения он должен быть определен;
- Псевдослучайность.* В соседних блоках должны удаляться совершенно разные коэффициенты (настолько, насколько это возможно), так как если наборы обнуляемых коэффициентов существенно отличаются друг от друга, их восстановление может происходить благодаря заимствованию информации из соседних блоков.

Необходимо также отметить тот факт, что стандарт JPEG подразумевает существенное сокращение вклада цветковых компонентов в итоговый объем

файла благодаря прореживанию. В то же время, яркостный компонент кодируется в исходном виде. Следовательно, наибольшее преимущество описанного способа кодирования следует ждать в случае применения его к компоненту  $Y$ . Каналы  $Cb$  и  $Cr$ , напротив, могут быть использованы нейронной сетью для восстановления изображения благодаря тому, что между яркостным и цветовыми компонентами существует локальная пространственная корреляция.

## 2.4. Внутреннее предсказание коэффициентов

Для улучшения качества закодированных изображений обычно используются сверточные нейронные сети. В данном разделе рассматриваются архитектуры моделей машинного обучения, позволяющие решить данную задачу, а также функции потерь, подходящие для обучения этих моделей.

### 2.4.1. AR-CNN

Архитектура AR-CNN (Artifacts Reduction Convolutional Neural Network) [17] призвана устранять артефакты, возникающие на изображении при кодировании его алгоритмом JPEG. Архитектура данной сети содержит три сверточных слоя, между которыми применяется функция-активации  $ReLU$ . Математически ее можно описать следующим образом:

$$ReLU(x) = \max(0, x),$$

$$F_0(Y) = Y,$$

$$F_i(Y) = ReLU(W_i * F_{i-1}(Y) + B_i), \quad i \in \{1, 2\},$$

$$F(Y) = W_3 * F_2(Y) + B_3,$$

где  $W_i$  и  $B_i$  — это веса сверточных фильтров и сдвиги  $i$ -го слоя соответственно,  $F_i$  — значения на выходе сверточных слоев, а « $*$ » означает операцию свертки.

Результаты экспериментов показывают, что данная архитектура действительно хорошо устраняет искажения, вызванные работой JPEG, однако не справляется с более сложными артефактами, появляющимися, например, после работы других методов кодирования изображений и видео. Исследования показывают, что причиной этому является недостаточное количество сверточных слоев.

### 2.4.2. QE-CNN-P

На базе AR-CNN была разработана архитектура QE-CNN-P (Quality Enhancement Convolutional Neural Network) [18], имеющая больше сверточных слоев, но способная искать более сложные искажения и устранять их. Именно данную архитектуру предлагается использовать для реализации внутреннего предсказания в транскодере JPEG-изображений. Она представлена на рисунке 11. Каждый ее слой состоит из свертки, сохраняющей размерность тензора, и функции активации  $PReLU$  с числом параметров равном числу фильтров в свертке. Математически данная архитектура может быть выражена следующим образом:

$$PReLU_i(x) = \max(0, x) + a_i \cdot \min(0, x),$$

$$F_0(Y) = Y,$$

$$F_i(Y) = PReLU_i(W_i * F_{i-1}(Y) + B_i), \quad i \in \{1, 2, 3, 4\},$$

$$F_5(Y) = PReLU_5(W_5 * F_0(Y) + B_5),$$

$$F_i(Y) = PReLU_i(W_i * (F_{i-5}(Y) \oplus F_{i-1}(Y)) + B_i), \quad i \in \{6, 7, 8\},$$

$$F_9(Y) = W_9 * (F_4(Y) \oplus F_8(Y)) + B_9,$$

где  $a_i$  — обучаемый параметр функции активации  $ReLU_i$ , представляющий из себя вектор значений, а « $\oplus$ » означает конкатенацию тензоров.

### 2.4.3. Функция потерь

При работе с изображениями чаще всего используется функция потерь MSE (Mean Squared Error) [17, 18]. Если  $\{I_n\}_{n=1}^N$  — набор исходных изображений,  $\{\tilde{I}_n\}_{n=1}^N$  — множество из тех же изображений но с частично обнуленными ДКП коэффициентами,  $\Theta = \{W_i, B_i, a_i\}$  — обучаемые параметры модели  $M$ , то

$$MSE(\Theta) = \frac{1}{N} \sum_{n=1}^N \left( M(\tilde{I}_n, \Theta) - I_n \right)^2.$$

При обучении требуется минимизировать данную функцию потерь, например, с помощью модификаций алгоритма градиентного спуска.

## Выводы по главе 2

В данной главе был рассмотрен алгоритм транскодирования JPEG-изображений на основе нейронной сети, а именно причины выбора такого ме-

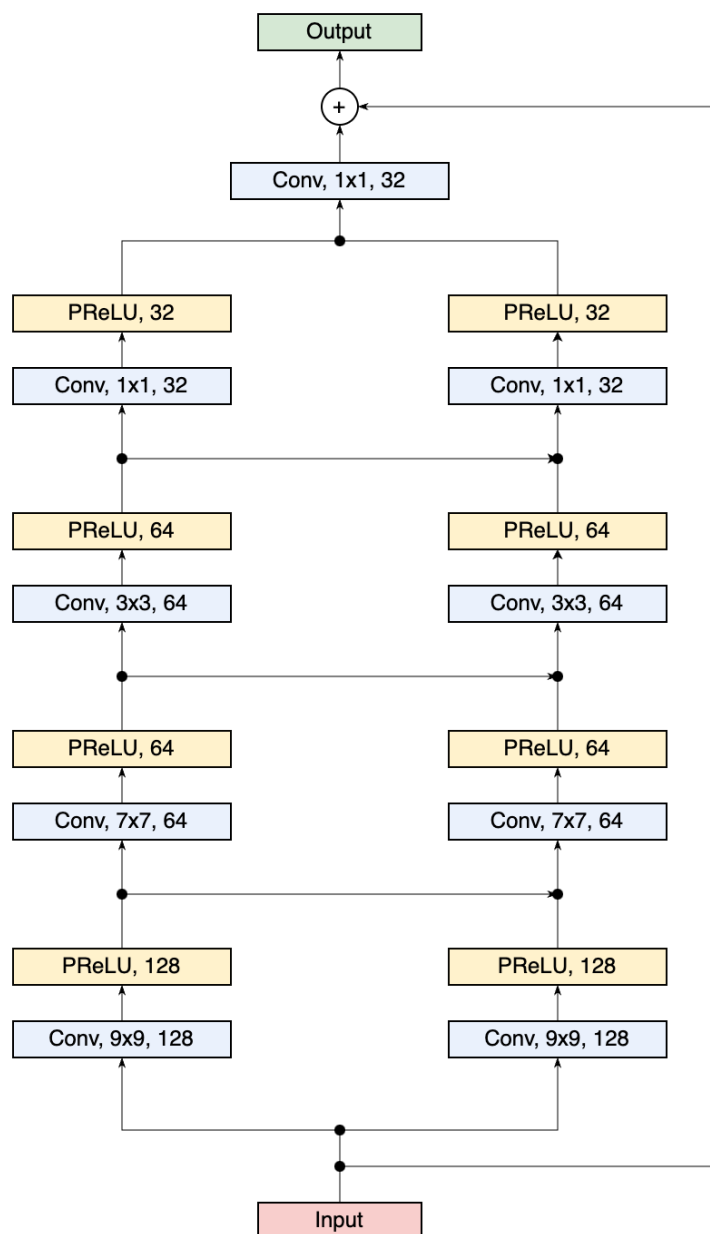


Рисунок 11 – Архитектура предлагаемой нейронной сети

тогда внутреннего предсказания, общая идея алгоритма и все преобразования, которые происходят с изображением в процессе его работы. Отдельное внимание было уделено требованиям к этапу обнуления коэффициентов и архитектуре нейронной сети.

## ГЛАВА 3. РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ

В данной главе описан процесс разработки транскодера, нейронной сети и ее обучения, подготовка набора данных, параметры обучения, а также алгоритм тестирования транскодера, результаты его работы и их анализ.

### 3.1. Разработка транскодера JPEG

Данный раздел содержит описание реализации всех этапов транскодирования изображений, кроме внутреннего предсказания.

#### 3.1.1. Публичное C++ API транскодера

Для обработки JPEG-изображений на C++ был разработан класс `Decoder`, реализующий следующие режимы работы:

- а) `Decoder::Mode::DEFAULT` — стандартное декодирование JPEG в PPM (установлен по умолчанию);
- б) `Decoder::Mode::ZERO_OUT_AND_DECODE` — декодирование JPEG в PPM с частичным обнулением ДКП коэффициентов;
- в) `Decoder::Mode::ENCODE_RESIDUALS` — транскодирование JPEG;
- г) `Decoder::Mode::DECODE_RESIDUALS` — трансдекодирование JPEG.

Публичное API класса `Decoder` содержит следующие методы:

- а) `Decoder & toggle_mode(Mode)` — переключение режима работы;
- б) `Decoder & set_dct_filter(std::size_t)` — настройка числа обнуляемых ДКП-коэффициентов для каждого блока;
- в) `Decoder & set_enhanced_file(const std::string &)` — настройка пути к PPM файлу, содержащему обработанное нейронной сетью в процессе внутреннего предсказания изображение;
- г) `void decode(const BytesList &)` — декодирование потока байтов из файла JPEG (`BytesList` является синонимом для `set::vector<unsigned char>`);
- д) `bool is_color_image() const` — получение информации о том, является ли декодированное изображения цветным;
- е) `std::size_t get_width() const` — получение ширины изображения;
- ж) `std::size_t get_height() const` — получение высоты изображения;

- и) `const ByteList & get_image() const` — получение байтов декодированного изображения;
- к) `const Output & get_transcoded() const` — получение результата транскодирования/трансдекодирования (`Output` — класс, инкапсулирующий в себе поток байтов).

Пример создания и настройки декодера представлен на листинге 1.

Листинг 1 – Пример создания и настройки декодера

```
Decoder decoder;
decoder
    .toggle_mode(Decoder::Mode::ENCODE_RESIDUALS)
    .set_dct_filter(16)
    .set_enhanced_file("enhanced.ppm");
```

### 3.1.2. Интерфейс командной строки

Для удобства работы с декодером был разработан интерфейс командной строки с использованием сторонней библиотеки `args` [19]. Он поддерживает такие опции как `-zero-out-and-decode` (листинг 2), `-encode-residuals` (листинг 3), `-decode-residuals` (листинг 4), соответствующие одноименным режимам работы. Если ни одна из перечисленных опций не передана — запускается обычное декодирование JPEG в PPM (листинг 5).

Листинг 2 – Пример вызова декодера для декодирования JPEG с частичным обнулением коэффициентов ДКП

```
$ ./Decoder --zero-out-and-decode --input "input.jpeg" --output "
    output.ppm" --power 16
```

Листинг 3 – Пример вызова декодера для транскодирования

```
$ ./Decoder --encode-residuals --input "original.jpeg" --output "
    compressed.jpeg" --enhanced "enhanced.ppm" --power 16
```

Листинг 4 – Пример вызова декодера для трансдекодирования

```
$ ./Decoder --decode-residuals --input "compressed.jpeg" --output
    "original.jpeg" --enhanced "enhanced.ppm" --power 16
```

Листинг 5 – Пример вызова декодера для декодирования JPEG

```
$ ./Decoder --input "input.jpeg" --output "output.ppm"
```

### 3.1.3. Детали реализации этапа обнуления коэффициентов

Для обнуления части коэффициентов был реализован класс `utils::DCTCoefficientsFilter`. Объекты данного класса инициализируются следующими параметрами:

- а) `power` — число обнуляемых коэффициентов;
- б) `masks_count` — число различных наборов коэффициентов (значение по умолчанию: 9);
- в) `seed` — зерно для инициализации генератора псевдослучайных чисел (значение по умолчанию: 42).

В конструкторе объекта данного класса создается `masks_count` псевдослучайных битовых масок длины 64 (`std::bitset<64>`), содержащих нули ровно на `power` позициях. Для получения маски используется метод `std::bitset<64> utils::DCTCoefficientsFilter::get_mask()`, который автоматически переходит к следующей по счету маске при каждом обращении. После получения всех масок счетчик сбрасывается. Пример использования данного класса показан на листинге 6.

Листинг 6 – Пример использования класса для обнуления коэффициентов

```
utils::DCTCoefficientsFilter filter(16);
const auto mask = filter.get_mask();
```

### 3.1.4. Детали реализации кодирования коэффициентов

Для транскодирования и трансдекодирования декодеру требуется предоставить RPPM-изображение, содержащее изображение после этапа внутреннего предсказания, то есть обработанное нейронной сетью. Транскодирование каждого блока изображения осуществляется независимо. При обработке блока, начальная точка которого имеет координаты  $(x, y)$ :

- а) Из RPPM-изображения вырезается соответствующий фрагмент (с теми же координатами), переводится в YCbCr и из него выбирается яркостный компонент (листинг 7);

Листинг 7 – Получение фрагмента изображения после внутреннего предсказания

```
std::array<float, 64> image_fragment;
for (std::size_t pixel_x = 0, k = 0; pixel_x < 8; ++pixel_x) {
    for (std::size_t pixel_y = 0; pixel_y < 8; ++pixel_y, ++k) {
        image_fragment[k] = m_enhanced_file->get_yuv(x + pixel_x,
            y + pixel_y).m_luminance;
    }
}
```

б) Далее к вырезанному блоку применяется ДКП (листинг 8);

Листинг 8 – Применение прямого ДКП

```
utils::DiscreteCosineTransform::forward(image_fragment);
```

в) Затем блок квантуется с использованием таблицы для яркостного компонента исходного JPEG-изображения. Реализация данного метода приведена на листинге 9.

Листинг 9 – Квантование блока

```
m_quantization_tables
    .at(component.m_quantization_table_id)
    .forward(image_fragment);
```

Далее после декодирования блока исходного JPEG-изображения, в нем происходит замена обнуляемых коэффициентов на основании маски `mask`, полученной из объекта `utils::DCTCoefficientsFilter`, на разницу между исходными коэффициентами и предсказанными (в режиме `Decoder::Mode::ENCODE_RESIDUALS`) или на сумму закодированной ошибки предсказания и предсказанного коэффициента (для `Decoder::Mode::DECODE_RESIDUALS`). Фрагмент кода, реализующий данные операции представлен на листинге 10.

## 3.2. Разработка нейронной сети

Для разработки и обучения нейронной сети была использована библиотека машинного обучения для Python — PyTorch [20]. В данном разделе описаны детали реализации и обучения QE-CNN-P.

### 3.2.1. Реализация QE-CNN-P

Реализация QE-CNN-P представлена классом `quality_enhancement.models.QECNN`, базовым классом которого



Листинг 10 – Замена коэффициентов ДКП на ошибки предсказания и наоборот

```
for (std::size_t i = 1; i < 64; ++i) {
    if (mask[i]) {
        continue;
    }
    if (IsEncodeResidualsMode()) {
        // Replace DCT coefficient with residual:
        block[i] -= enhanced_block[i];
    }
    else {
        // Recovery of the DCT coefficient based on the residual
        and the predicted value:
        block[i] += enhanced_block[i];
    }
}
```

является `torch.nn.Module`. Слои нейронной сети реализованы с помощью `torch.nn.Sequential` API и классов `torch.nn.Conv2d` (двумерная свертка) и `torch.nn.PReLU` (функция *PReLU*, описанная в разделе 2.4.2).

Для того, чтобы каждый слой модели сохранял размерность входного изображения, использовалась следующая формула расчета отступа  $P$  для свертки

$$P = \left\lfloor \frac{n - 1}{2} \right\rfloor,$$

где  $n$  — размер ядра свертки.

### 3.2.2. Подготовка набора данных

Для обучения нейронной сети было взято 50000 случайных изображений из набора данных ImageNET [21], из которых 5000 было выделено для тестирования модели, 10000 — для валидации и оставшиеся 35000 — для обучения. Изображения были обрезаны до размера  $320 \times 320$ , затем в каждом блоке  $8 \times 8$  пикселей были удалены (заменены нулями) по 16 псевдослучайных АС-коэффициентов, как это было описано в разделе 2.3.

Для упрощения обработки большого числа изображений был реализован скрипт `image_processing.py` на Python. Так в примере, указанном на листинге 11, сначала происходит разметка изображений по их принадлежности к определенной части набора данных (в зависимости от назначения), затем файлы обрезаются до необходимого размера и каждый из них декодируется в

двух режимах — стандартном и с обнулением коэффициентов. Полученные изображения были использованы для обучения модели.

Листинг 11 – Подготовка набора данных для обучения модели

```
$ python3 py/process_images.py --split -i "images/01-original-images" --validation_size 0.2 --test_size 0.1
$ python3 py/process_images.py --crop -i "images/01-original-images" -o "images/02-cropped" --height 320 --width 320
$ python3 py/process_images.py --decode -i "images/02-cropped" -o "images/03-decoded"
$ python3 py/process_images.py --zero_out_and_decode -i "images/02-cropped" -o "images/04-decompressed" --power 16
```

### 3.2.3. Параметры обучения

Обучение производилось на персональном компьютере MacBook Pro 2021 г. (чип Apple M1 Pro) на основании функции ошибки MSE (`torch.nn.MSELoss`) с использованием оптимизатора Adam (`torch.optim.Adam`) с параметром *learning\_rate* =  $1e-4$  на протяжении 100 эпох. После каждой эпохи рассчитывалась метрика MSE на валидационном наборе данных. Наименьшее значение функции ошибки было достигнуто по завершении последней эпохи и составило  $1,405e-3$ .

### 3.2.4. Интерфейс командной строки

Для удобства работы с моделью был реализован интерфейс командной строки. В нем поддерживаются две опции:

- а) `--train` — запуск обучения нейронной сети;
- б) `--enhance` — обработка изображений для их транскодирования.

Для запуска обучения модели требуется настройка следующих параметров (пример показан на листинге 12):

- а) `--limit` — ограничение на размер набора данных;
- б) `--learning_rate` — скорость обучения;
- в) `--checkpoints_folder` — путь к папке, содержащей чекпоинты, то есть файлы, содержащие словари состояний параметров модели и оптимизатора.

Для запуска внутреннего предсказания модели необходимо настроить (листинг 13):

### Листинг 12 – Пример запуска обучения модели

```
$ python3 py/main.py --train --limit 3600 --learning_rate=0.0001
  --checkpoints_folder "py/checkpoints"
```

- а) `--input_wildcard/-I` — шаблон файлов, которые требуется обрабатывать;
- б) `--output_folder/-O` — папка, в которую необходимо сохранить обработанные изображения;
- в) `--checkpoints_folder` — путь к папке, содержащей чекпоинты.

### Листинг 13 – Пример запуска внутреннего предсказания

```
$ python3 py/main.py --enhance -I "images/decompressed/tst*.ppm" -
  O "images/enhanced" --checkpoints_folder "py/checkpoints"
```

## 3.3. Тестирование

### 3.3.1. Оценка качества работы нейронной сети

На рисунке 12 проиллюстрировано то, насколько хорошо обученная модель справляется с искажениями, которые были внесены в результате обнуления коэффициентов. На примере видно, что большая часть видимых искажений устранена. Так, например, на левом изображении практически на всех участках четко видно деление на блоки, а после нейросетевой обработки подобных артефактов уже не остается. Таким образом, визуально заметно, что нейросети удастся приближенно восстанавливать обнуленные ДКП коэффициенты.

Графики на рисунке 13 показывают сравнение распределений ошибок предсказания коэффициентов ДКП (residuals) на позициях обнуленных коэффициентов и самих этих коэффициентов для позиций 6, 7, 12, 13, 21 и 22 (означающих индексы АС-коэффициентов при сканировании в зигзагообразном порядке). Видно, что ошибки предсказания сконцентрированы ближе к нулю, чем сами коэффициенты. Следовательно, число ненулевых кодируемых энтропийным кодером значений сокращается, а их абсолютные значения уменьшаются. На основании этого можно сделать вывод о том, что их энтропия снижается и итоговый объем битового потока должен быть меньше.



Рисунок 12 – Пример работы QE-CNN: слева — изображение после обнуления части ДКП коэффициентов; посередине — изображение на выходе нейронной сети; справа — исходное изображение

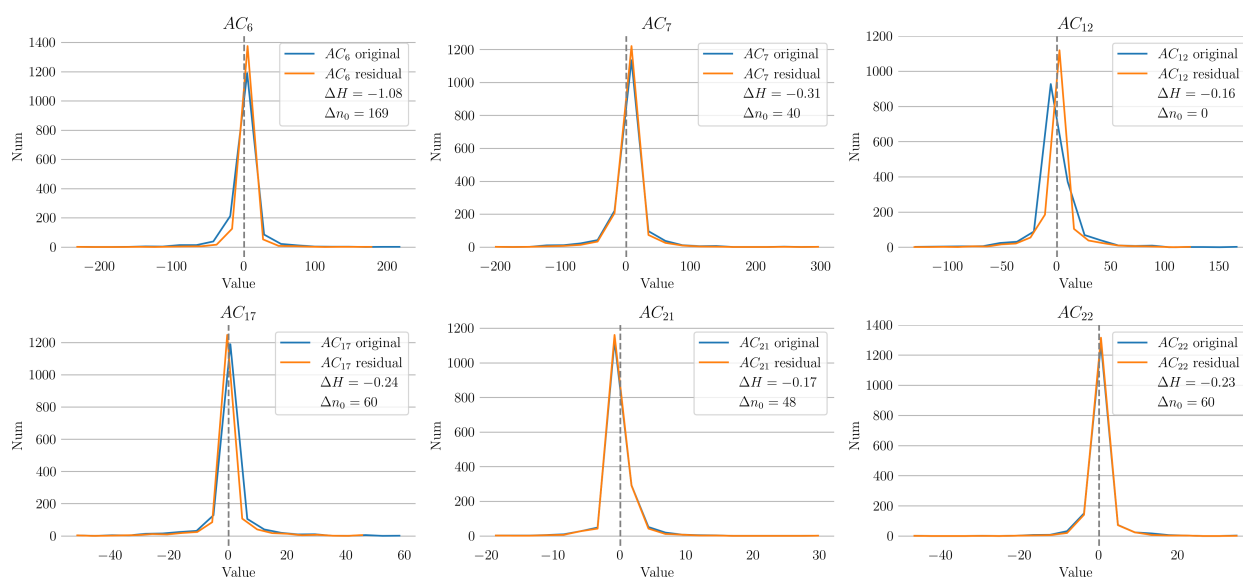


Рисунок 13 – Распределение обнуляемых АС-коэффициентов и ошибок их предсказаний на изображении из тестового набора данных, показанном на рисунке 12 при обнулении 16 коэффициентов,  $\Delta H$  — изменение энтропии,  $\Delta n_0$  — изменение числа нулевых значений коэффициентов

### 3.3.2. Подготовка набора транскодированных и транскодированных изображений

Для выполнения функционального тестирования реализованного транскодера, оценки степени сжатия изображений и анализа возможности интеграции предлагаемого модуля внутреннего предсказания с утилитами Jpegtran и LLJPEG был доработан скрипт `image_processing.py`. В него были добавлены функции транскодирования (листинг 14) и транскодирования (листинг 15) набора изображений, применения Jpegtran к JPEG-изображениям с

целью замены кода Хаффмана на арифметический кодер (листинг 16), функции расчета статистики изображений: средней, медианной и максимальной степеней сжатия (листинг 17), а также опция для запуска end-to-end тестов транскодера (листинг 18).

Листинг 14 – Пример транскодирования набора изображений

```
$ python3 py/process_images.py --encode_residuals -i "images/02-cropped" -o "images/06-transcoded" -e "images/05-enhanced" --power 16
```

Листинг 15 – Пример трансдекодирования набора изображений

```
$ python3 py/process_images.py --decode_residuals -i "images/06-transcoded" -o "images/07-transdecoded" -e "images/05-enhanced" --power 16
```

Листинг 16 – Пример транскодирования набора изображений, основанного на замене кода Хаффмана на арифметический кодер

```
$ python3 py/process_images.py --arithmetic -i "images/02-cropped" -o "images/08-jpegtran-output"
$ python3 py/process_images.py --arithmetic -i "images/06-transcoded" -o "images/09-jpegtran-output-for-transcoded"
```

Листинг 17 – Пример расчета статистики сжатия изображений

```
$ python3 py/process_images.py --statistics -i "images/02-cropped" -o "images/06-transcoded"
$ python3 py/process_images.py --statistics -i "images/02-cropped" -o "images/08-jpegtran-output"
$ python3 py/process_images.py --statistics -i "images/02-cropped" -o "images/09-jpegtran-output-for-transcoded"
```

Листинг 18 – Пример запуска end-to-end тестов транскодера

```
$ python3 py/process_images.py --test_transcoder -i "images/02-cropped" -o "images/07-transdecoded"
Test result: succeeded
```

### 3.3.3. Детали реализации end-to-end тестирования

Функциональное end-to-end тестирование транскодера основано на сравнении файлов до и после транскодирования с точностью до бита. Для этого используется функция стандартной библиотеки Python — `filecmp.cmp`, которая принимает пути к двум файлам и возвращает `True`, если они совпадают.

Тестирование проводилось на том же тестовом наборе данных, на каком оценивалась эффективность работы транскодера, и было завершено без ошибок.

### 3.3.4. Оценка степени сжатия

Оценка качества работы транскодера выполнялась на упомянутом выше тестовом наборе данных. Были рассмотрены следующие варианты транскодирования изображений (результаты экспериментов представлены в таблице 1):

- а) *Предобученная модель* — метод транскодирования, использующий для внутреннего предсказания модель QE-CNN-P обученную для визуального повышения качества изображений [22] и видео, закодированных кодеком HEVC;
- б) *Предложенный 1 (CNN-based)* — метод транскодирования на внутреннего предсказания с помощью обученной в рамках настоящей работы модели машинного обучения;
- в) *Jpegtran*;
- г) *Предложенный 2 (CNN-based + Jpegtran)* — комбинация CNN-based и Jpegtran;
- д) *LLJPEG*;
- е) *Предложенные 3 (CNN-based + LLJPEG)* — комбинация CNN-based и LLJPEG.

Таблица 1 – Сравнение степени сжатия, %

Методы \ Метрики	Среднее	Медиана	$\sigma$	По всему набору данных
Предобученная модель	2,57	2,68	0,87	2,76
Предложенный 1	3,33	3,38	0,54	3,38
Jpegtran	10,05	10,00	3,06	10,12
Предложенный 2	13,16	13,11	2,91	13,18
LLJPEG	16,14	15,49	3,37	15,59
Предложенный 3	15,44	15,08	2,61	15,14

Результаты эксперимента показывают, что обученная в рамках настоящей работы нейронная сеть сжимает тестовый набор изображений на 3,38% (в отличие от предобученной нейронной сети с такой же архитектурой, но решающей иную задачу). На рисунке 14 также можно видеть, что большинство изображений сжаты 1-м предложенным методом на 3-4% и отклонение от этой статистики небольшое (о чем и говорит низкое стандартное отклонение). Лишь на единицах изображений описанный подход к транскодированию оказался неэффективным.

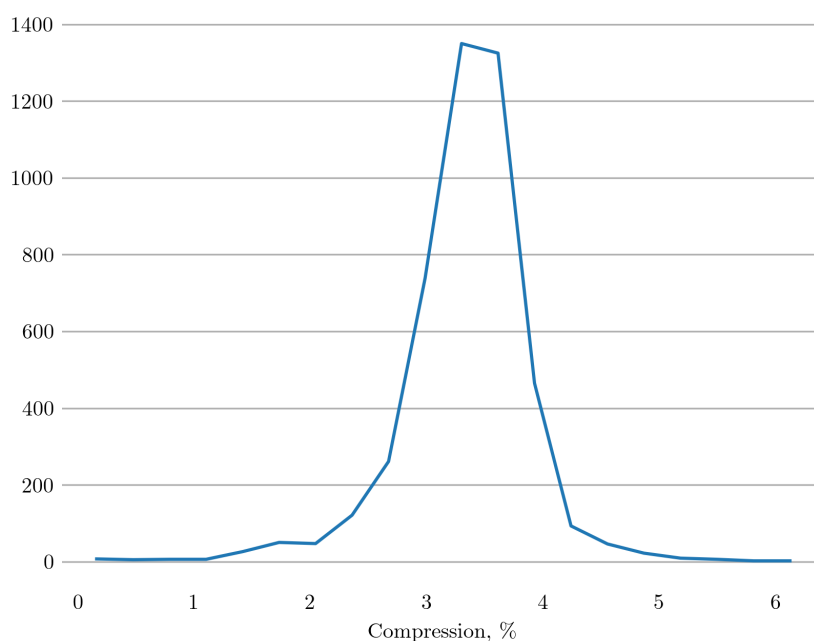


Рисунок 14 – Кривая, показывающая для какого числа изображений из тестового набора удалось достичь определенной степени сжатия

Таким образом, результаты экспериментов показывают, что предлагаемый в настоящей работе метод предсказания коэффициентов ДКП хорошо комбинируется с Jpegtran, позволяющим заменить код Хаффмана на арифметический код. Степени сжатия, наблюдаемые при использовании каждого из этих подходов, фактически суммируются, если использовать методы совместно.

Несмотря на то, что на практике транскодер LLJPEG показывает наилучшие результаты, 2-й предлагаемый метод отстает от него всего на 2,41% (при рассмотрении сжатия всего тестового набора данных в целом). При этом в данной комбинации реализовано только предсказание АС-коэффициентов и

улучшение энтропийного кодера благодаря использованию арифметического кодирования, что необходимо учитывать при сравнении эффективности работы двух данных подходов.

### 3.3.5. Дальнейшие перспективы исследования

Как было упомянуто в предыдущем разделе — рассмотренный в настоящей работе транскодер реализует только часть из известных на сегодняшний день подходов, причем целью исследования являлось изучение конкретного этапа — внутреннего предсказания, и возможности использования нейронной сети на данном этапе. Отдельно требуется отметить, что из-за ограничений на вычислительные ресурсы при подготовке работы:

- а) Не выполнена оптимизация числа обнуляемых коэффициентов — эксперименты проводились для изображений, в которых удалены 16 АС-коэффициентов;
- б) Не проведены эксперименты для разных параметров качества (quality factor, QF);
- в) Нейронная сеть была обучена на небольшом числе изображений и эпох (анализ научных статей на тему кодирования изображений при помощи нейронных сетей [23–25] показывает, что набор данных обычно содержит не менее 90'000 изображений, а обучение проводится на протяжении 1-2 млн. эпох);
- г) Не оптимизированы архитектуры нейронной сети и функции потерь.

Несмотря на все вышеупомянутые ограничения результаты показывают, что нейронные сети можно использовать для предсказания коэффициентов ДКП и что такой метод позволяет добиваться дополнительного сжатия JPEG. Для увеличения степени сжатия в рамках дальнейшей работы над нейросетевым транскодером планируется:

- а) Продолжать обучение нейронной сети (в идеале с использованием больших вычислительных ресурсов);
- б) Оценить степень сжатия для различных гиперпараметров алгоритма; например, иного числа удаляемых коэффициентов, большего числа битовых масок;
- в) Рассмотреть возможность нейросетевого внутреннего предсказания DC, а не АС коэффициентов;
- г) Доработать алгоритм JPEG на остальных этапах:



- 1) Ввести адаптивную величину блока;
- 2) Внедрить в транскoder адаптивное арифметическое кодирование.

### **Выводы по главе 3**

В данной главе были рассмотрены детали реализации транскodера, а именно — C++ API для транскodирования JPEG-изображений, реализация нейронной сети, интерфейсы командной строки для обработки изображений, обучения и запуска модели. Были описаны методики функционального end-to-end тестирования, а также приведены результаты экспериментов по оценке степени сжатия различными алгоритмами транскodирования. Дополнительно была доказана возможность и эффективность интеграции нейросетевого внутреннего предсказания и утилиты Jpegtran.

## ЗАКЛЮЧЕНИЕ

В настоящей работе был проведен анализ тенденций в области сжатия изображений, рассмотрены современные методы кодирования изображений с потерями и без. Было показано, почему эти методы не подходят для дополнительного сжатия JPEG без потерь и рассмотрены существующие подходы к решению данной задачи. При анализе методов были выделены общие положения, определяющие то, на каких этапах возможна реализация транскодирования JPEG без потерь.

Был предложен алгоритм транскодирования без потерь, использующий внутреннее предсказание коэффициентов ДКП на основе нейронной сети, для оптимизации облачного хранения изображений, закодированных в формате JPEG. Эксперименты показывают, что такой подход позволяет сократить в среднем 3,33% (до 6,28%) памяти на изображениях из подмножества набора данных ImageNET.

Несмотря на то, что альтернативные системы транскодирования JPEG-изображений, такие как LLJPEG, позволяют добиться более высокого сжатия, данный подход имеет место быть и нуждается в дальнейшем исследовании, так как в результате его работы получается корректный файл JPEG. Это означает, что данный подход может быть использован одновременно с другими методами транскодирования. К тому же, предлагаемый алгоритм затрагивает исключительно этап внутреннего предсказания, следовательно, он открыт к дальнейшему улучшению, например, за счет отказа от фиксированного размера блока и использования арифметического кодера. Перспективы такого подхода подтверждаются экспериментами по интеграции предлагаемого метода и утилиты Jpegtran.

Результатом работы стал программный код транскодера JPEG-изображений без потерь, обученная нейронная сеть, решающая задачу внутреннего предсказания коэффициентов ДКП, а также ряд скриптов, упрощающих работу с наборами изображений и нейронной сетью.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 *Wallace G. K.* Overview of the JPEG (ISO/CCITT) still image compression standard // Proc. SPIE. — 1990. — June. — Vol. 1244. — P. 220–233.
- 2 *Кудряшов Б. Д.* Теория информации. — Санкт-Петербург : СПбГУ ИТМО, 2010. — 188 с.
- 3 *Mozilla.* Lossy Compressed Image Formats Study [Электронный ресурс]. — 07/2024. — URL: [https://web.archive.org/web/20160311235158/http://people.mozilla.org/~josh/lossy\\_compressed\\_image\\_study\\_july\\_2014](https://web.archive.org/web/20160311235158/http://people.mozilla.org/~josh/lossy_compressed_image_study_july_2014).
- 4 *Christopoulos C., Skodras A., Ebrahimi T.* The jpeg2000 still image coding system: an overview // IEEE Transactions on Consumer Electronics. — 2000. — Vol. 46, no. 4. — P. 1103–1127.
- 5 *Bellard F.* BPG Image Format Website [Электронный ресурс]. — 2024. — URL: <http://bellard.org/bpg/>.
- 6 *Google.* An image format for the Web [Электронный ресурс]. — 2024. — URL: <https://developers.google.com/speed/webp>.
- 7 *Weinberger M. J., Seroussi G., Sapiro G.* The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS // IEEE Trans. Image Process. — 2000. — Aug. — Vol. 9, no. 8. — P. 1309–1324.
- 8 *Wu X., Memon N.* CALIC—A context based adaptive lossless image codec. — 1996. — May.
- 9 *Sneyers J.* FLIF Image Format Website [Электронный ресурс]. — 2024. — URL: <http://flif.info/>.
- 10 *Kyleduo A.* Tinypng Project [Электронный ресурс]. — 2024. — URL: <https://tinypng.com/developers>.
- 11 *Mozilla.* Mozilla Mozjpeg [Электронный ресурс]. — 2024. — URL: <https://github.com/mozilla/mozjpeg>.
- 12 *Guetzli: Perceptually Guided JPEG Encoder / J. Alakuijala [и др.].* — 2017. — arXiv: 1703.04421 [cs.CV].
- 13 *Kulissen H.* Jpegtran Description [Электронный ресурс]. — 2024. — URL: [http://jpegclub.org/articles/Verlustfreie\\_JPEG\\_Drehung.pdf](http://jpegclub.org/articles/Verlustfreie_JPEG_Drehung.pdf).

- 14 *Kulissen H.* Jpegtran Project [Электронный ресурс]. — 2024. — URL: <https://www.npmjs.com/package/jpegtran>.
- 15 *Sun C., Fan X., Zhao D.* Lossless Recompression of JPEG Images Using Transform Domain Intra Prediction // IEEE Transactions on Image Processing. — 2023. — Vol. 32. — P. 88–99.
- 16 *vilab-sct.* LLJPEG [Электронный ресурс]. — 2022. — URL: <https://github.com/vilab-sct/LLJPEG>.
- 17 Compression Artifacts Reduction by a Deep Convolutional Network / C. Dong [и др.]. — 2015. — arXiv: 1504.06993 [cs.CV].
- 18 Enhancing Quality for HEVC Compressed Videos / R. Yang [и др.] // IEEE Transactions on Circuits and Systems for Video Technology. — 2019. — Июль. — Т. 29, № 7. — С. 2039–2054. — ISSN 1558-2205. — DOI: 10.1109/tcsvt.2018.2867568. — URL: <http://dx.doi.org/10.1109/TCSVT.2018.2867568>.
- 19 *Taywee.* args [Электронный ресурс]. — 2024. — URL: <https://github.com/Taywee/args>.
- 20 *Foundation T. L.* PyTorch [Электронный ресурс]. — 2024. — URL: <https://pytorch.org/>.
- 21 ImageNet: A Large-Scale Hierarchical Image Database / J. Deng [et al.] // CVPR09. — 2009.
- 22 *Belyaev E.* AIVideoEnhancement [Электронный ресурс]. — 2024. — URL: <https://gitlab.actcognitive.org/aivideocoding/aivideoenhancement>.
- 23 Context-Based Trit-Plane Coding for Progressive Image Compression / S. Jeon [и др.]. — 2023. — arXiv: 2303.05715 [eess.IV].
- 24 NVTC: Nonlinear Vector Transform Coding / R. Feng [и др.]. — 2023. — arXiv: 2305.16025 [cs.CV].
- 25 Towards Accurate Image Coding: Improved Autoregressive Image Generation with Dynamic Vector Quantization / M. Huang [и др.]. — 2023. — arXiv: 2305.11718 [cs.CV].