Федеральное государственное бюджетное образовательное учреждение

высшего образования

«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ

РОССИЙСКОЙ ФЕДЕРАЦИИ»

Департамент анализа данных и машинного обучения

Курсовая работа

на тему:

«Машинное обучение в задачах сжатия информации»

Выполнил:
Студент группы ПМ18-3
Факультета информационных технологий и анализа больших данных
Купернюк Н. А.

Научный руководитель:
канд. физ.-мат. наук, доцент Каверина В. К.

Москва 2020

Оглавление

Введение

Сжатие информации широко используется во всевозможных областях информационных технологий. Существует большое количество определённых алгоритмов для сжатия, которые показывают хороший результат лишь в той области, для которой они были созданы. Например, JPEG для фотографий или алгоритм Хаффмана для векторных изображений. Но все эти методы не универсальны, поэтому поиск новых технологий сжатия информации на сегодняшний день все ещё остаётся актуальной задачей.

Учитывая современные темпы роста объемов разнообразной информации, хранящихся на физических носителях, передаваемых как по Интернету, так и по локальным сетям, необходимо передавать, хранить и сжимать данные с максимально возможной эффективностью.

Машинное обучение все глубже проникает во все сферы современных технологий. Большинство корпораций связанных не только с IT, но и с самыми разнообразными сферами деятельности, всё чаще используют технологии машинного обучения в своей инфраструктуре. Эта тема невероятна актуальна на сегодняшний день.

Некоторые способы сжатия информации с применением машинного обучения и будут рассмотрены в этой работе.

Основные понятия

Для демонстрации методы сжатия информации будут применены к изображениям. Это может помочь лучше определить качество сжатия, так как существует довольно много различных метрик применимых к изображениям. Так же сжатие на изображениях позволяет определить приемлемость сжатия «на глаз», так как незначительные изменения в качестве изображения будут видны человеческому глазу. Для измерения качества сжатия в этой работе будет применена метрика PSNR.

PSNR (пиковое отношение сигнала к шуму) — это технический термин, обозначающий соотношение между максимально возможным значением сигнала и мощностью шума, искажающего значение сигнала. Поскольку многие сигналы имеют широкий динамический диапазон, PSNR измеряется в логарифмической шкале в децибелах. PSNR чаще всего используется для измерения уровня искажения при сжатии изображения. Его определение производится через среднеквадратичную ошибку (MSE).

В случае двух монохромных изображений I и K одинакового размера m×n, одно из которых считается приближением другого, MSE и PSNR высчитываются так:

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}|I(i,j) - K(i,j)|^2$$

$$PSNR = 10 * \log_{10}\frac{MAX_I^2}{MSE}$$

где MAX_I — это максимальное значение, принимаемое пикселем изображения. Например, если пиксели имеют разрядность 8 бит, $MAX_I = 255$.

Для цветных изображений с тремя компонентами в цветовом пространстве RGB (красный цвет, зелёный и синий) на пиксель применяется такое же определение PSNR, но среднеквадратичная ошибка теперь считается по всем трем компонентам и

делится на произведение всех размерностей изображения – ширина × высота × количество цветовых каналов. Чем больше значение PSNR, тем лучше качество зашумленного изображения относительно оригинала.

Уменьшение размера файла является основной целью сжатия, так что необходимо рассматривать эту метрику в работе. Будут рассмотрены значения для изображений в формате PNG и JPG, так как это, пожалуй, самые популярные на сегодняшний день форматы для хранения и передачи изображений.

Описание исследуемых данных

В этой работе методы машинного обучения будут отрабатываться на фотографиях цветов, распространяемых Google по свободной лицензии CC-BY 2.0. Данные были выбраны потому, что содержат много (3600) изображений, собранных вместе в удобном для использования формате. Эти данные представляют собой высококонтрастные яркие фотографии, но в невысоком разрешении, что поможет лучше определить качество полученных в результате сжатия изображений и поможет определить успешность применяемых методов в «тяжёлых» условиях.

Рисунок 1. Примеры изображений из датасета

Создание и обучение моделей

1) K-means

Первым рассмотренным методом будет кластеризация методом k-means. Из обрабатываемого изображения берутся все пиксели, представляющие из себя совокупность трёх цветовых компонент RGB. Затем они разбиваются на определённое количество кластеров. И для всех пикселей изображения, принадлежащих к одному кластеру, выбирается один цвет, определённый, опять же, тремя цветовыми компонентами. Результатом окажется изображение с уменьшенным количеством цветов, что позволит использовать меньше памяти для хранения изображения. Настраиваемыми параметрами будут количество кластеров, на которые будут разбиваться все цвета исходного изображения, а также метод взятия одного конечного цвета для кластера: взять среднее, медианное или кластерные центры. Проверим этот метод для различных способов выбора итогового цвета и различного числа кластеров.

	Средние цвета		Медианные цвета		Цвета-центры кластеров		Память, кб	
Количество цветов	**PSNR**	**TIME**	**PSNR**	**TIME**	**PSNR**	**TIME**	**PNG**	**JPG**
5	25,147	1,269	24,849	1,34	25,147	1,533	12,776	14,179
10	27,544	3,366	27,41	3,071	27,546	3,272	22,302	14,626
15	29,141	5,789	28,966	4,887	29,11	5,923	28,614	14,369
20	30,074	7,906	29,951	7,582	30,082	10,253	33,936	13,993
25	30,761	11,538	30,655	12,432	30,747	9,843	38,985	13,944
30	31,35	13,383	31,207	14,182	31,347	13,043	42,018	14,006
35	31,835	16,56	31,733	13,062	31,848	17,1	47,594	13,986
40	32,265	16,976	32,151	18,895	32,267	18,863	51,931	13,872
45	32,635	18,07	32,509	23,353	32,644	19,49	54,294	13,912
50	32,999	25,338	32,889	24,409	32,995	20,718	56,402	13,71
55	33,291	24,569	33,19	21,932	33,288	24,383	57,979	13,801
60	33,574	26,987	33,472	28,799	33,588	24,905	61,58	13,612

Таблица 1. Сравнение параметров метода K-Means

Заметно, что лучшим образом себя показал способ, при котором итоговым цветом для кластера становится средний показатель цветовых компонент по кластеру. Однако итоговое количество цветов выбрать не так просто: при увеличении количества кластеров увеличивается качество, но также растёт и время выполнения программы.

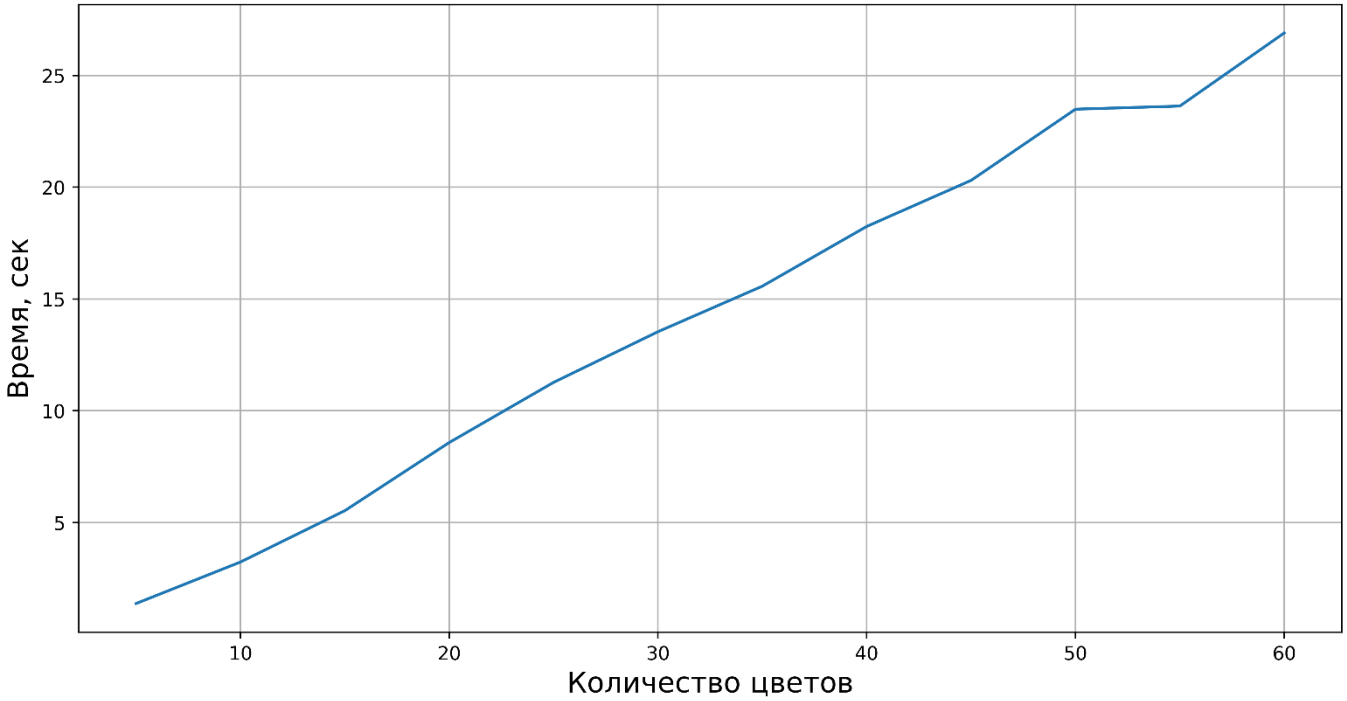

Рисунок 2. Время сжатия изображения методом K-Means с разным количеством цветов

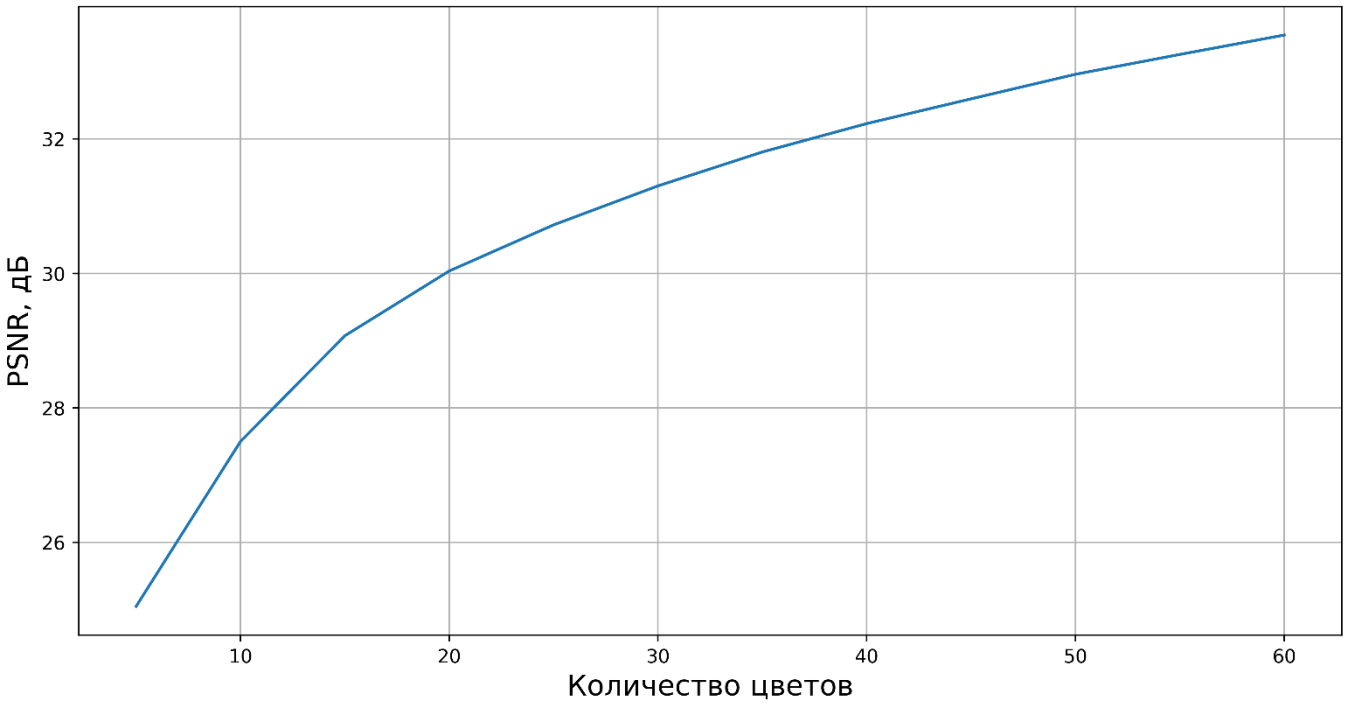

Рисунок 3. Результат сжатия изображения методом K-Means с разным количеством цветов

В некоторых источниках указано, что типичные значения PSNR для сжатия изображений лежат в пределах от 30 до 40 дБ. Тогда метод k-means будет оптимально настраивать на поиск 20 кластеров в пикселях изображения, так как именно такое количество позволяет получить PSNR > 30. Приведём пример для 25 кластеров.

Оригинальное изображение

Изображение с 25 цветами

Рисунок 4. Сравнение оригинального изображения и обработанного

Хотя качество изображения и ухудшилось, оно все ещё остаётся на приемлемом уровне. Однако, заметим, что для формата JPG такой способ сжатия не работает, ведь

в большинстве случаев размер сжатого файла получился больше размера исходного файла, что противоречит самой идее сжатия информации. В то же время сохранение в формате PNG позволяет достичь хороших результатов и существенно уменьшить размер изображения.

Такой результат скорее всего связан с тем, что формат JPG сам по себе является эффективным методом сжатия и кодирования данных. В нём изображение переводится из цветового пространства RGB в пространство YCbCr, а каждая компонента затем сжимается по отдельности. Скорее всего при изменении количества цветов меньшая вариативность изображения помешала сжать данные с помощью алгоритмов JPG так же хорошо, как на оригинальных изображениях.

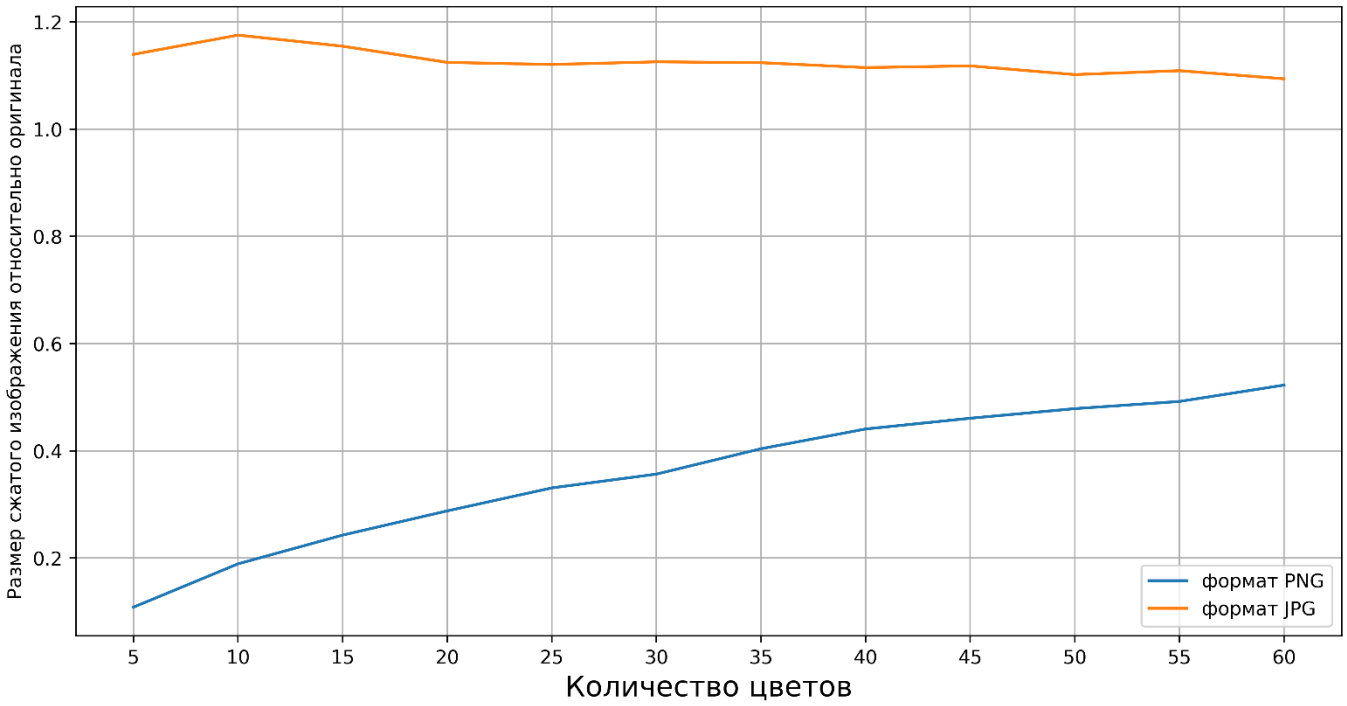

Рисунок 5. Сравнение сжатия изображений в формате PNG и JPG

2) PCA

Метод главных компонент – один из методов предобработки данных для понижения размерности. Он позволяет убрать избыточные измерения и сохранить

только важные. Метод позволяет уменьшить дисперсию признаков, что поможет снизить размер обработанного файла.

Из матрицы объекты-признаки F размерностью l×n выбираются только m признаков, так что новая матрица объектов-признаков G имеет размерность l×m. При этом создаётся матрица линейного преобразования новых признаков в старые, U размерностью n×m так, чтобы $GU\hat{}T \approx F$. Если $m \leq rk\, F$, то минимум $\|GU^T - F\|^2$ достигается, когда столбцы U – это собственные значения $\lambda_1, \lambda_2, \ldots, \lambda_m$, а матрица $G = FU$.

Рассмотрим реализации метода для разных количеств главных компонент, от 1 до 100.

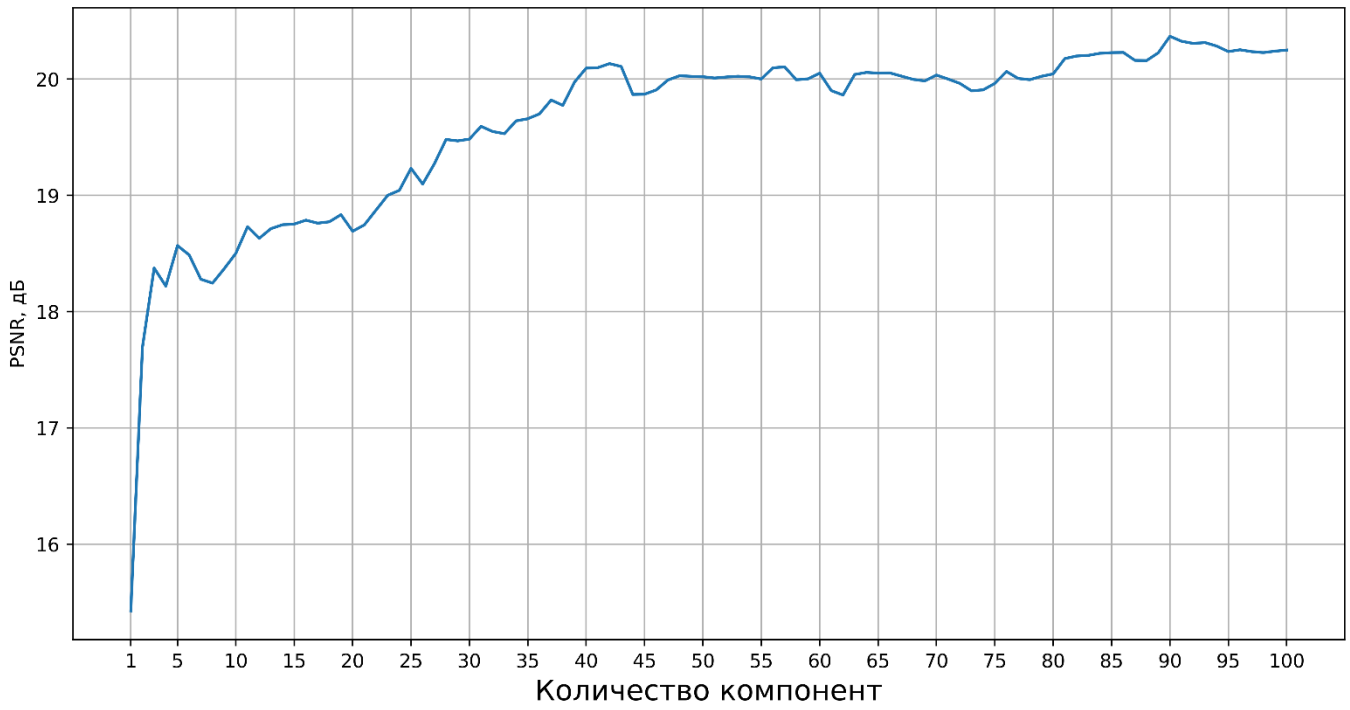

Рисунок 6. Зависимость PSNR от количества компонент

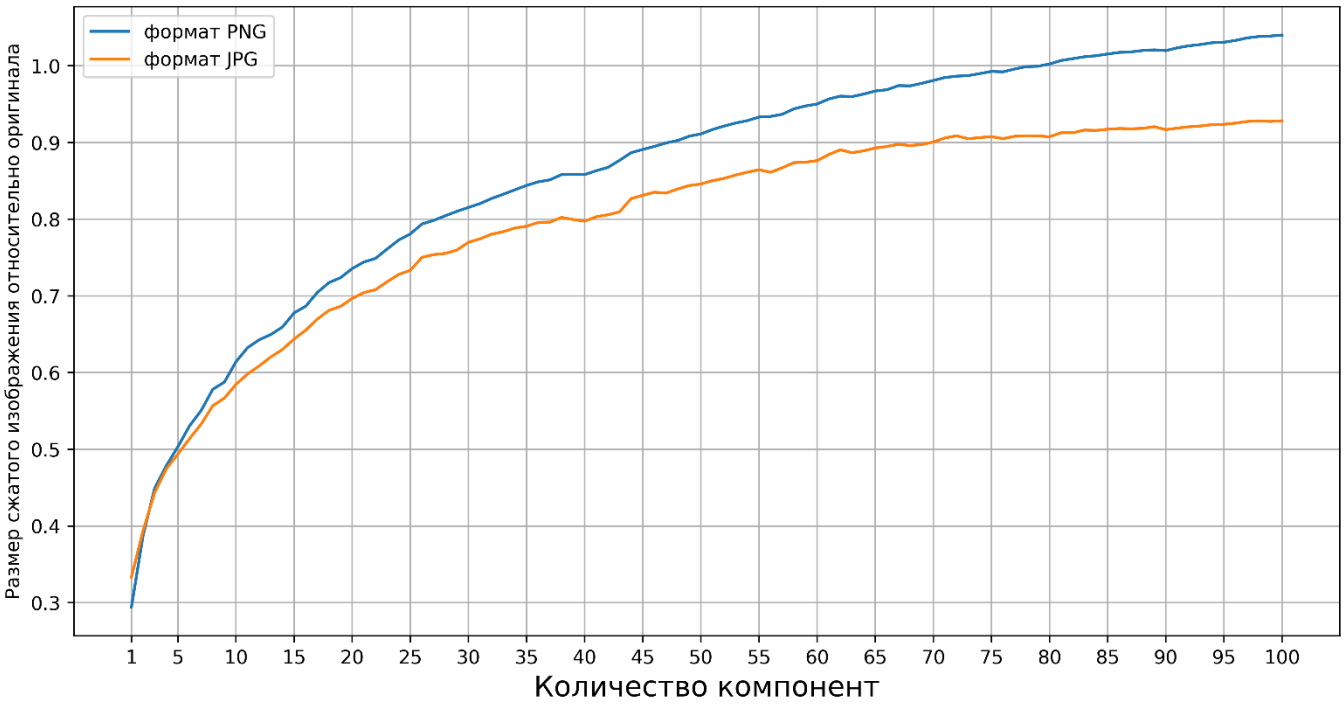

Рисунок 7. Зависимость размера файла от количества компонент

Заметно, что качество получаемого изображения хуже, чем у предыдущего метода k-means. Так же заметим, что не имеет смысл брать больше 80 главных компонент, так как в таком случае размер сжатого файла в расширении PNG получается больше, чем у оригинального файла. Рассмотрим пример для 45 компонент.

Рисунок 8. Сравнение обработанного изображения и оригинального с числом компонент равным 45

Заметно, что качество изображения получилось довольно неплохим, но восстановленная картинка получилась очень зашумленной. Сравним также изображения с другим числом компонент.

Рисунок 9. Сравнение обработанного изображения и оригинального с числом компонент равным 100

Изображение восстановилось гораздо лучше, но все ещё остаётся сильно зашумленным. Возможно, именно из-за этого размер файла получается высоким в формате PNG, ведь количество цветов на обработанном изображении получается больше, чем на оригинале.

3) Автоэнкодер

Автоэнкодер – это особый тип нейронной сети, которая обучается копировать свои входные данные на выходные. Как следствие, архитектура этой нейронной сети предусматривает равенство размерностей входного и выходного слоя, а так же, как минимум один скрытый слой, размерности меньше, чем входной и выходной. Автоэнкодер учится сжимать данные, сводя к минимуму ошибку восстановления. Эту нейросеть можно разделить на 2 части: энкодер и декодер. Энкодер принимает на вход исходные данные и сжимает их, а декодер восстанавливает первоначальные

данные из сжатых. Рассмотрим архитектуру нейронной сети с размерностью 16×16×3 = 768 нейронов во входном и выходном слоях и 64 нейрона во входном слое.

Коэффициент сжатия у такой нейронной сети равен $\frac{768}{64} = 12$ при идеальных условиях. Для обработки реальных фотографий будем разбивать их на фрагменты 16×16 пикселей. Недостающие фрагменты будем дополнять черным цветом. Если фотография разбивается на m×n таких фрагментов, то её можно представить в виде массива размерностью m*n×16×16×3, а в нейросеть будет передаваться массив размерностью m*n×768. Результат кодировки одного изображения – массив размером m*n×64.

Нейросеть будет реализована с помощью библиотеки tensorflow. TensorFlow – это открытая программная библиотека для машинного обучения, разработанная и опубликованная Google для решения задач построения и тренировки нейронных сетей. Она проста в освоении и использовании, поэтому предпочтение было отдано именно ей.

Вычисления производились в Google Colab – это бесплатный облачный сервис, в основе которого лежит уже известный Jupyter Notebook. Он предоставляет все необходимые инструменты прямо в браузере, а также даёт доступ к мощным CPU и GPU, что бывает необходимо при работе с нейронными сетями.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 768)               0
_________________________________________________________________
dense (Dense)                (None, 64)                49216
=================================================================
Total params: 49,216
Trainable params: 49,216
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 10. Структура кодирующей части нейросети

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 768)               49920
_________________________________________________________________
reshape (Reshape)            (None, 16, 16, 3)         0
=================================================================
Total params: 49,920
Trainable params: 49,920
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 11. Структура декодирующей части нейросети

Обучим нейросеть на наборе из 500 случайных картинок из тренировочного набора с 20 эпохами обучения. Размер валидационной выборки возьмём за 0.3. Будем сводить к минимуму среднеквадратичную ошибку (MSE) между изображениями, так как это приведёт к увеличению PSNR. Обучение заняло примерно 6 минут.

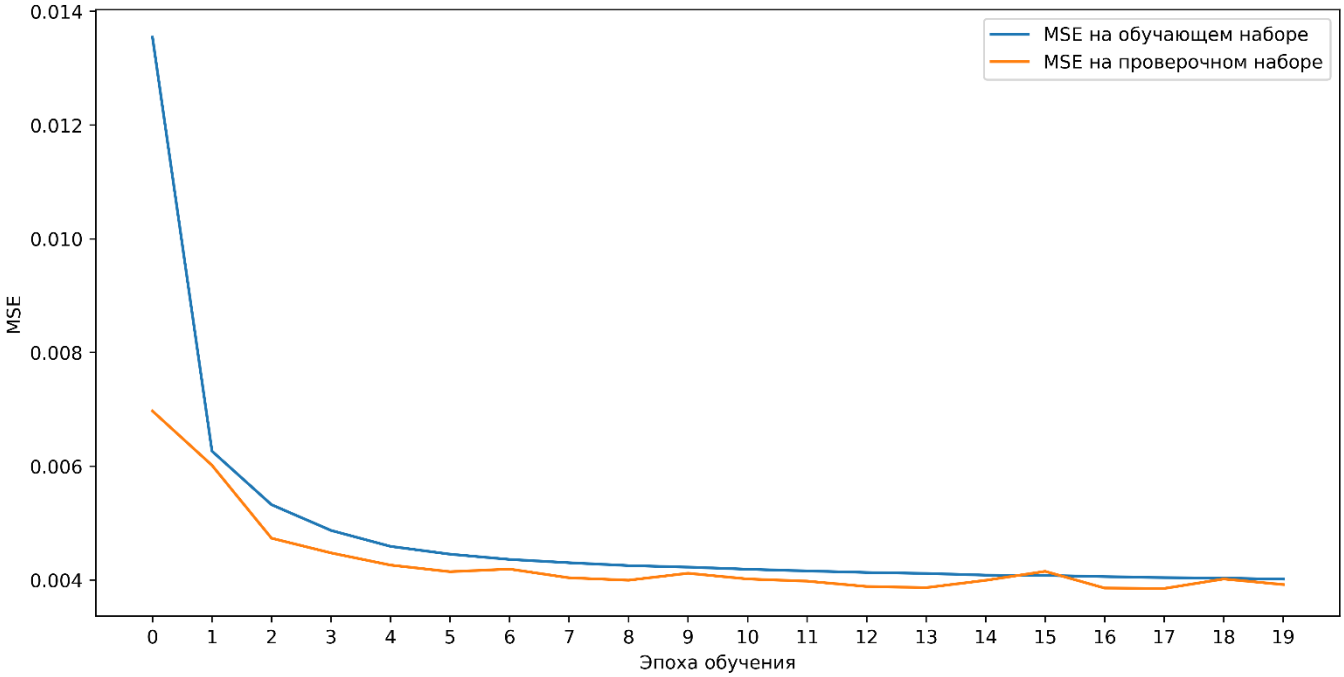

Рисунок 12. Зависимость ошибки MSE нейросети от эпохи обучения

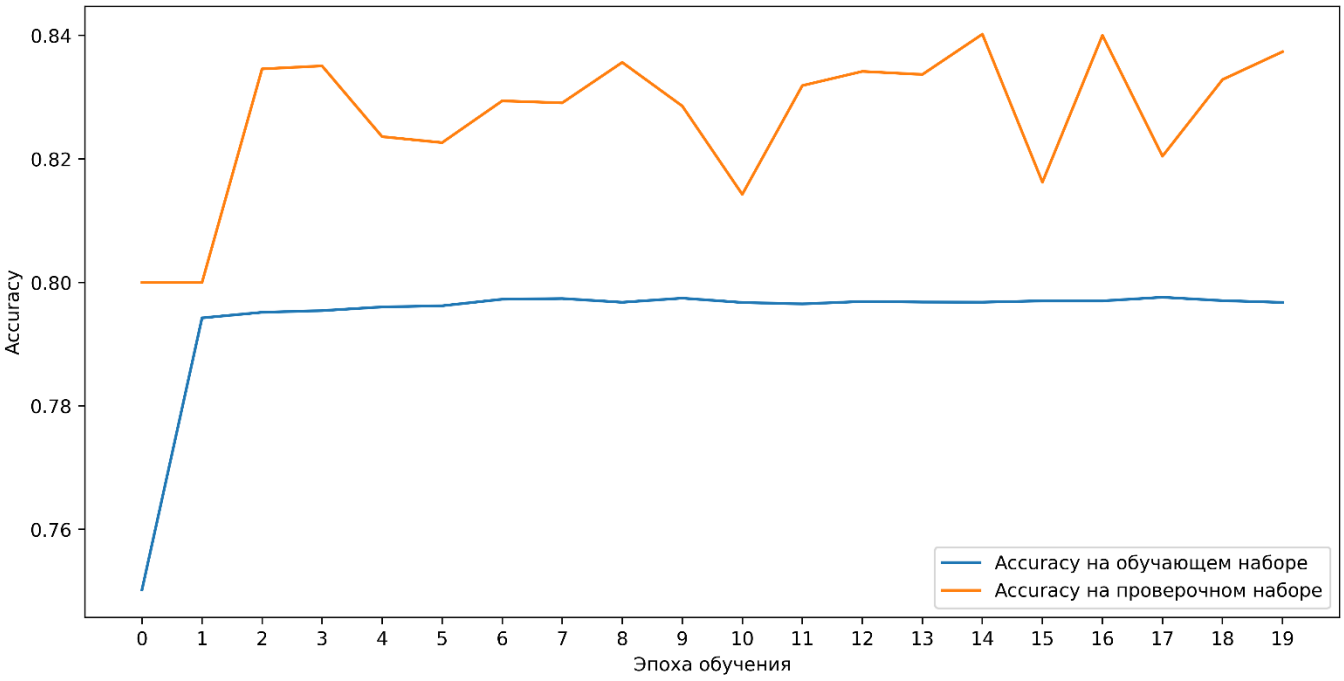

Рисунок 13. Зависимость точности нейросети от эпохи обучения

Выведем PSNR из среднеквадратичной ошибки:

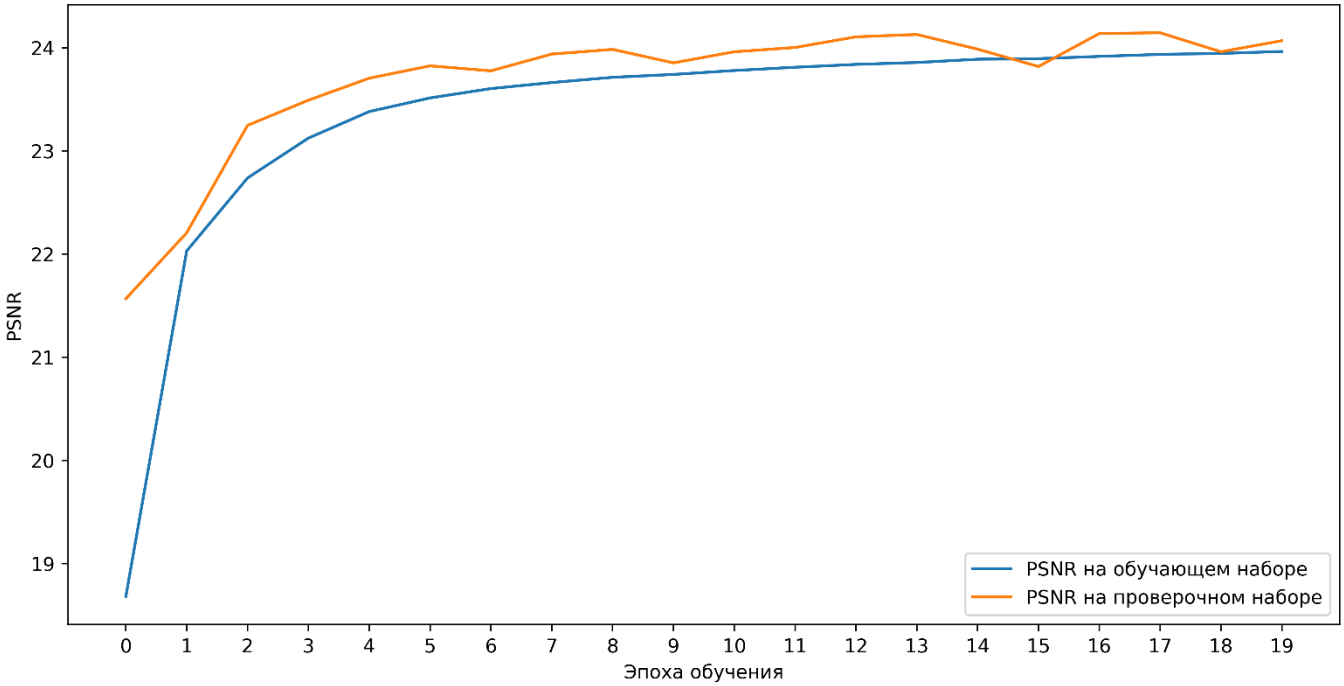

Рисунок 14. Зависимость PSNR от эпохи обучения

И просмотрим итоговый результат для случайного изображения:

Рисунок 15. Сравнение изображений, сжатых нейросетью

Заметно, что качество изображения ухудшилось, появилось много зашумленных и размытых участков. Так же видно разбиение исходного изображения на квадраты 16×16 пикселей, особенно на местах с мягким цветовым градиентом.

Сравнение моделей

Для сравнения моделей будем использовать набор из 100 случайных изображений из основного набора. Сравним размер сжатых файлов, а также средний PSNR по всем изображениям.

	MEAN_PSNR	MEMORY_JPG	MEMORY_PNG
Kmeans(25)	29,937	1,068	0,392
Kmeans(45)	31,476	1,057	0,526
PCA	23,843	0,803	0,885
NN	22,977	0,479	0,122

Таблица 2. Сравнение методов

Как мы видим, лучше всего себя показывает уменьшение цветов изображения методом K-Means, но также этот метод запрашивает много памяти для хранения изображений в формате JPG, следовательно использовать его для сжатия таких изображений просто бессмысленно. В то же время для формата PNG этот метод показывает неплохой результат сжатия и сжатые файлы заменяют всего 0.392 от объема необработанных данных. Но стоит учитывать, что K-Means запрашивает больше времени для обработки по сравнению с другими методами. Метод PCA показывает результат заметно хуже, но сжимает данные для обоих форматов изображений. Однако, степень сжатия оставляет желать лучшего по сравнению с K-Means, обработанные данные имеют размер примерно 80% по сравнению с оригинальными. Нейронные сети показывают качество хуже всех, однако отрыв по PSNR не очень большой; в то же время, они позволяют сжимать файлы гораздо лучше, чем остальные методы. Для формата JPG данные удалось сжать примерно в 2 раза, а для PNG сжатые данные составляют всего 12% от оригинальных. Возможно, результат PSNR для нейросети можно улучшить, если обучить модель на большем количестве данных. Этот способ, пожалуй, является самым перспективным из рассмотренных.

Заключение

В данной работе были рассмотрены несколько методов по сжатию информации с помощью машинного обучения. Некоторые из них показали весьма неплохой результат. Метод k-средних позволил достичь хорошего качества сжатых данных при том, что они занимают всего чуть больше 50% от объема необработанной информации. Нейросеть показала не такой хороший результат для восстановленных изображений, однако сжатые данные занимают намного меньше компьютерной памяти по сравнению с остальными моделями. Такую модель наверняка можно улучшить, если обучить её на большем количестве данных или изменить некоторые параметры нейросети. Метод главных компонент показал себя не самым лучшим образом: его качество ненамного лучше, чем у нейросети, а размер сжатой информации иногда превосходит таковой у k-means.

Список литературы

1. И. А. Лёзин, А. В. Соловьёв Сжатие изображения с использованием многослойного персептрона, Самарский университет, 2016.
2. *Коэльо Л.П., Ричарт В.* Построение систем машинного обучения на языке Python. 2016. 302 с.
3. *Hastie T., Tibshirani R., Friedman J.* The Elements of Statistical Learning. Springer, 2014.
4. Р. А. Воронкин Использование нейросетевого подхода к решению задачи сжатия изображений с возможностью осуществления генетического поиска, СГУ, 2014.
5. https://www.coursera.org/learn/vvedenie-mashinnoe-obuchenie/home/welcome
6. https://www.tensorflow.org/