МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра математического анализа

Нейронные сети в задаче классификации и сжатия изображений

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

студента 2 курса 218 группы

направления ***01.04.02 – Прикладная математика и информатика***

механико-математического факультета

Буймистрюка Льва Александровича

Научный руководитель
профессор, д.ф.-м.н., профессор _____ С.Ф. Лукомский
подпись, дата

Заведующий кафедрой
д.ф.-м.н., профессор _____ Д.В. Прохоров
подпись, дата

Саратов 2020

Введение Автономные роботизированные системы на производствах, автопилотируемые автомобили, продвинутые алгоритмы улучшения и преобразования изображений, поиск любого человека в сети по одной фотографии, голосовые помощники – всё это ещё 10 лет назад казалось далёким будущем и уделом писателей-фантастов. Но благодаря постоянно растущей вычислительной мощи современных ЭВМ и исследованиям в области *нейронных сетей* это становится прекрасным настоящим.

Термин *«нейронная сеть»* появился в середине XX века. В 1943 году Уорреном Мак-Каллоком и Уолтером Питтсом была разработана компьютерная модель нейронной сети на основе математических алгоритмов и теории деятельности головного мозга. Они предложили представить процесс обработки данных как связи между *«нейронами»*, которые принимают на вход, преобразуют и отдают на выход информацию от других нейронов.

С тех пор регулярно публиковались различные исследования на эту тему, но они хоть и были многообещающими, не могли быть в полной мере реализованы на практике из-за технических ограничений и малой вычислительной мощности компьютеров тех лет.

Однако начиная с 2014 года, не в последнюю очередь благодаря инженерам корпорации Google, был сделан огромный шаг вперед: были описаны новые методы обучения нейросетей, их оптимизации и применения к задачам *компьютерного зрения* (*Computer Vision, CV*), задачам *обработки естественного языка* (*Natural Language Processing, NLP*) и прочим. Так же были разработаны программные средства для упрощения разработки нейросетей, например TensorFlow и Keras.

Одним из самых распространенных видов нейронных сетей на сегодняшний день являются *свёрточные нейронные сети* (*Convolutional Neural Network, CNN*), в основе которых лежит идея использования свёрточного линейного фильтра и *рекуррентные нейронные сети* (*Recurrent Neural Network, RNN*).

В данной работе свёрточные нейронные сети будут использованы для решения задач классификации изображений, а рекуррентные нейронные сети – для решения задачи сжатия изображений с потерями.

Основное содержание работы. Работа состоит из 6 глав.

Постановка задачи классификации.

Определение 1.1. (Вероятностная постановка задачи классификации)

Предполагается, что множество пар «объект, класс» $X \times Y$ является вероятностным пространством с неизвестной вероятностной мерой P. Имеется конечная обучающая (тренировочная) выборка наблюдений $X^m = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, сгенерированная согласно вероятностной мере P. Требуется построить алгоритм $a : X \to Y$, способный классифицировать произвольный объект $x \in X$.

Нейросети прямого распространения.

Определение 1.2. Под *нейроном* будем понимать нелинейную функцию $\varphi_j : \mathbb{R} \to \mathbb{R}$, называемую *функцией активации*, которая принимает на вход взвешенную сумму n входных значений и отдает на выход значение этой функции.

Определение 1.3. *Нейронным слоем* размера n называют последовательность, состоящую из n нейронов, которые имеют общую заданную функцию активации φ_j.

Определение 1.4. Нейронным слой размера n называют *полносвязным* (*dense layer*), если каждый j-тый нейрон в нем принимает на свой вход выходы всех нейронов с предыдущего слоя размера m. При этом каждой i-той связи соответствует вес w_{ij}, где $i = \overline{1, m}$, $j = \overline{1, n}$.

Замечание. Если i-тый слой размера n является полносвязным, то выходные значения $o_{i,j}$ всех нейронов этого слоя могут быть представлены в виде:

$$\begin{pmatrix} o_{i,1} \\ o_{i,2} \\ \vdots \\ o_{i,n} \end{pmatrix} = \varphi_i \left(\begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,m} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n,1} & w_{n,2} & \cdots & w_{n,m} \end{pmatrix} \cdot \begin{pmatrix} o_{i-1,1} \\ o_{i-1,2} \\ \vdots \\ o_{i-1,m} \end{pmatrix} \right)$$

где функция активации φ_i применяется к каждому элементу вектора, то есть

$$\varphi_i \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} \varphi_i(a_1) \\ \varphi_i(a_2) \\ \vdots \\ \varphi_i(a_n) \end{pmatrix}$$

Определение 1.5. *Нейросетью* называют совокупность нейронных слоёв, в котором первый слой называется *входным слоем*, последний – *выходным слоем*, а все остальные – *скрытыми слоями*. При этом на входном слое за ненадобностью отсутствует суммирование и функция активации.

Определение 1.6. *Линейной фильтрацией* матрицы A размера $n \times n$ с помощью матрицы *ядра свертки* B размера $m \times m$, где $m < n$ называют преобразование, результатом которой является матрица C, в которой каждый элемент есть сумма поэлементного умножения каждого элемента подматрицы размера $m \times m$ с центром в элементе $a_{ij} \in A$ с матрицей B.

Определение 1.7. *Свёрточным слоем* (*convolution layer*) называют нейронный слой, в котором каждый нейрон связан лишь с частью нейронов предыдущего слоя, причем результатом суммирования таких связей будет эквивалентен результату использования ядра свертки для линейной фильтрации.

Определение 1.8. В случае, когда рассматривается только один канал изображения, понятия *фильтр* и *ядро свёртки* являются взаимозаменяемыми. В случае же, когда принимаются во внимание несколько каналов (например RGB), данные понятия приобретают разный смысл. В этом случае фильтр представляет из себя коллекцию ядер: по ядру на каждый входной канал. Каждое ядро из коллекции является уникальным. Каждый фильтр в свёрточном слое генерирует один и только один выходной канал. Каждое из ядер, входящих в состав фильтра, перемещается вдоль соответствующего ему входного канала, выдавая обработанную версию сигналов. Некоторые ядра могут иметь более «сильные» веса, чем другие, отдавая предпочтение определенным входным каналам. (Например, если ядро, ответственное за красный канал, имеет более «сильные» веса, чем другие ядра, нейронная сеть будет реагировать на красный цвет наиболее выражено).

Алгоритм обучения нейросети выглядит следующим образом:

1. Всем весам w_{ij} присваиваются случайные значения.
2. Для каждой пары (x_i, y_i) вида «объект, класс» из обучающей выборки:
   2.1. Значения x_i подаются на нейроны входного слоя.
   2.2. Вычисляются значения всех нейронов сети.
   2.3. Производится коррекция ошибки.
3. Шаг 2 повторяется, пока результат не станет удовлетворительным.

Определение 1.9. *Точностью* (*accuracy*) работы сети называют отношение правильных результатов к ошибочным.

Численные эксперименты. Благодаря проведенным численным экспериментам, были выявлены лучшие параметры обучения нейросети. Увеличим теперь число эпох и проведем обучение, используя эти параметры:

1. Количество эпох (epochs): 300.
2. Функция темпа обучения ($\mathrm{lr}(k)$): $\mathrm{sawtooth}(0.001,\ 0.0001,\ 300,\ k)$.
3. Искусственное увеличение данных.
4. Количество изображений p в одном пакете (batch size): 64.

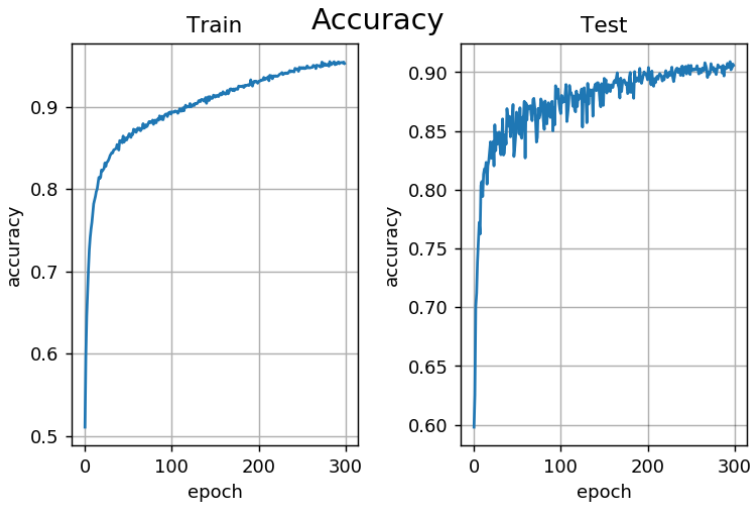

Рисунок 1.1 — Сравнение точности на обучающей (слева) и на тестовой (справа) для лучшей конфигурации

Как видно на Рис. 1.1, нейросеть способна с вероятностью $p = 0.9$ предсказывать, к какому классу относится изображение из тестовой выборки. При этом важно отметить, что те случаи, в которых нейросеть ошибается, в большинстве случаев можно разделить на два класса:

1. Изображение настолько неоднозначное, что даже человек с трудом может сказать, что на нём изображено.
2. Нейросеть путает два схожих класса: например кошек и собак (так как они очень похожи друг на друга, особенно издалека) или птиц и самолетов (так как представители обоих этих классов чаще всего находятся на фоне неба).

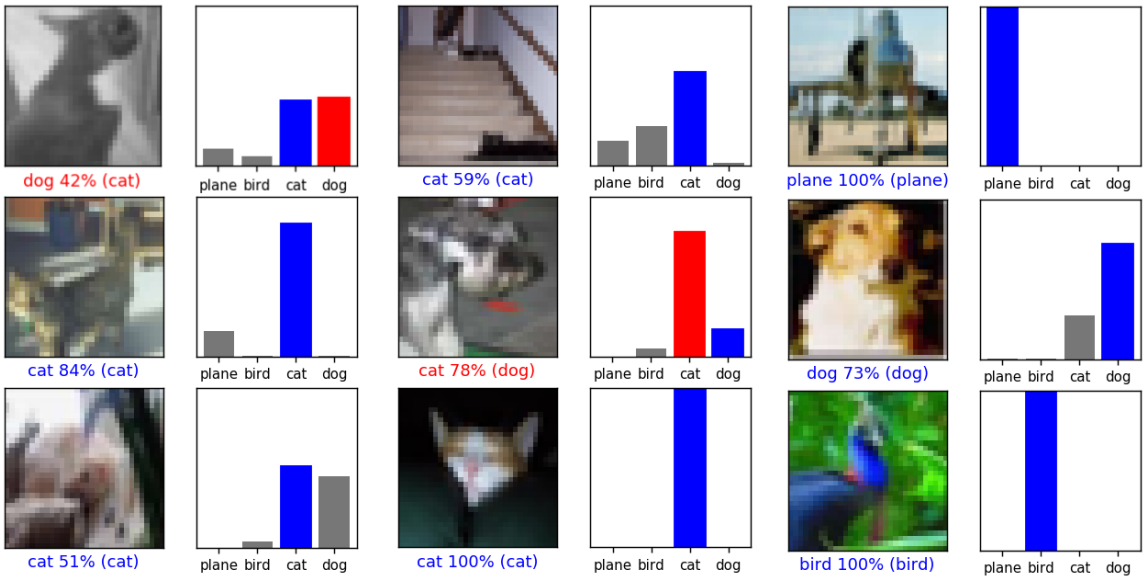

Рисунок 1.2 — Проверка предсказаний нейросети на некоторых изображениях из тестовой выборки. Слева представлено изображение, а справа - значения выходного слоя нейросети (вероятность того, к какому классу относится изображение). Верный класс обозначен синим цветом, а угаданный неверно – красным

Постановка задачи сжатия изображений.

Определение 1.10. *Сжатие данных с потерями* – метод сжатия (компрессии) данных, при использовании которого восстановленные данные отличаются от исходных, но степень отличия не существенна с точки зрения их дальнейшего использования. Этот тип компрессии часто применяется для сжатия аудио и видеоданных, изображений, в интернете (особенно в потоковой передаче данных) и цифровой телефонии.

Определение 1.11. Для оценки потерь при сжатии изображения, оригинальное и восстановленное изображение сравнивается с помощью метрики MS-SSIM:

- SSIM (structure similarity), индекс структурного сходства – метрика сходства двух изображений, которая учитывает не только искажения, вызванные шумом, но и структурные искажения. Эта метрика близка к тому, как человеческий глаз сравнивает два изображения.
  За x и y обозначаются окна, движущиеся по изображениям I и K соответственно.

$$\mathrm{SSIM}_{x,y} = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{x,y} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

  где:

  - μ_x, μ_y – среднее x и y соответственно
  - σ_x, σ_y – дисперсия x и y соответственно
  - $\sigma_{x,y}$ – ковариация x и y
  - $C_1 = (k_1 L)^2$, $C_2 = (k_2 L)^2$ – переменные
  - L – динамический диапазон пикселей. Обычно $L = 2^{\mathrm{bpp}} - 1$
  - $k_1 = 0.01$ и $k_2 = 0.03$ – константы

  SSIM может принимать значения в интервале $[-1, +1]$, при чём значение $+1$ достигается только в случае абсолютно идентичных изображений. Чем больше значение, тем больше структурное сходство между изображениями x и y.
- MS-SSIM (Multi-Scale SSIM), многомасштабный индекс структурного сходства – среднее значение SSIM по всем окнам:

$$\text{MS-SSIM}_{I,K} = \mathrm{avg}_{x,y}\,\mathrm{SSIM}_{x,y}$$

Рекуррентные нейронные сети.

Определение 1.12. Рекуррентная нейронная сеть (Recurrent Neural Network, RNN) – это сеть, вычисление которой происходит несколько раз. Причём на вход следующей итерации подаётся не только новая порция данных, но и сохранённая с предыдущего шага информация. Таким образом, можно разбить последовательность входных данных x_t на небольшие части фиксированного размера $(x_i)_{i=1}^t$, и подавать i-тую часть только на i-том шаге.

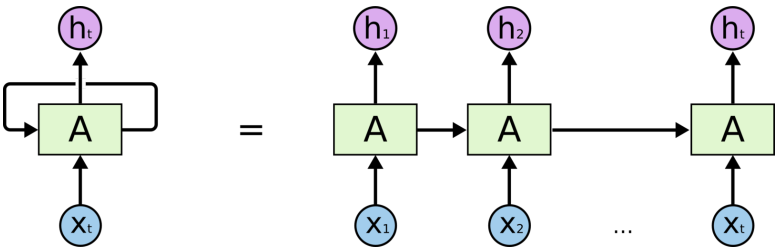

Рисунок 1.3 — Представление итераций рекуррентной нейронной сети

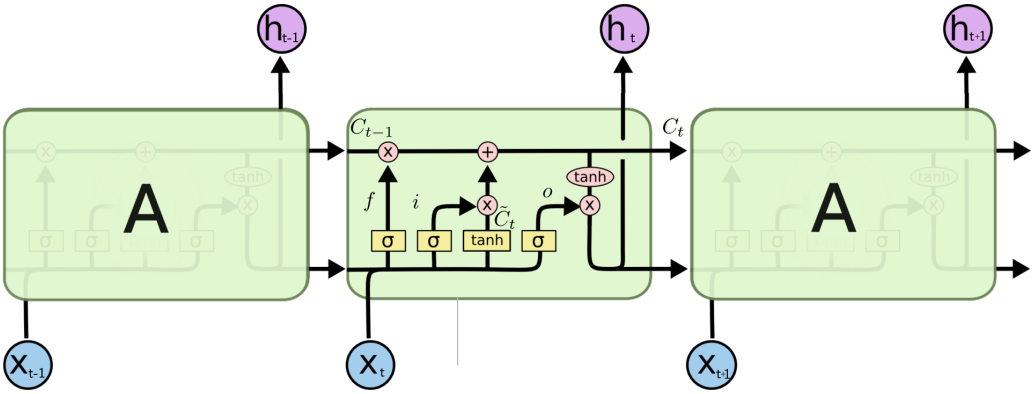

Рисунок 1.4 — Внутреннее представление шага LSTM

То, что RNN напоминают цепочку, говорит о том, что они тесно связаны с последовательностями и списками. RNN – самая естественная архитектура нейронных сетей для работы с данными такого типа. И конечно, их используют для подобных задач. За последние несколько лет RNN с невероятным успехом применили к целому ряду задач: распознавание речи, языковое моделирование, перевод, распознавание изображений и так далее.

Немалая роль в этих успехах принадлежит LSTM (Long Short-Term Memory, Долгая краткосрочная память) – модификации архитектуры RNN.

Наш алгоритм сжатия будет состоять из трёх соединённых последовательно нейросетей: энкодера E, бинарайзера B и декодера D, где E и D - рекуррентные нейросети с LSTM-компонентами. Входное изображение будет кодироваться, а затем трансформироваться в бинарный вид, который может быть сохранён в файл или передан в декодер. Сеть-декодер создаёт приближение оригинального изображения на основе бинарных данных. Процесс можно повторять итеративно, пока ошибка (разница между оригинальным и восстановленным изображением) не станет достаточно близка к нулю.

Каждую итерацию этапа кодирования изображения можно представить в следующем виде:

$$b_t = B(E_t(r_{t-1})), \quad \hat{x}_t = D_t(b_t) + \gamma\, \hat{x}_{t-1}, \quad r_t = |x - \hat{x}_t|$$

где

- Индексы показывают номер шага итерации
- E_t, D_t и B – энкодер, декодер и бинарайзер соответственно
- b_t – бинарное представление
- x – оригинальное изображение
- $\hat{x}_t$ – реконструированное изображение
- $\gamma = \begin{cases} 0 \text{ на первом шаге} \\ 1 \text{ на остальных} \end{cases}$
- r_t – величина ошибки, абсолютная поэлементная разница между x и $\hat{x}_t$
- $r_0 = x$

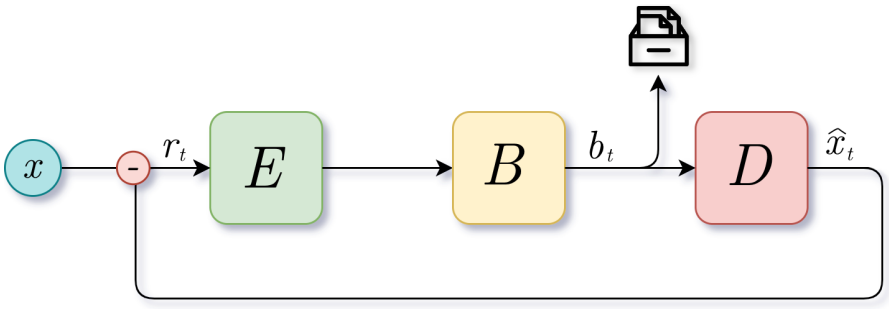

Рисунок 1.5 — Алгоритм кодирования

Каждую итерацию этапа декодирования изображения можно представить в следующем виде:

$$\hat{x} = \sum_t \hat{x}_t = \sum_t D(b_t)$$

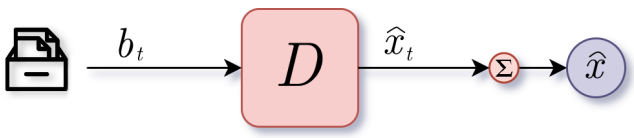

Рисунок 1.6 — Алгоритм декодирования

Численные эксперименты. Для обучения нейросети были использованы следующие параметры:

1. Количество эпох (epochs): 2500.
2. Темп обучения (lr): 0.001.
3. Количество изображений в одном пакете (batch size): 4.
4. Количество итераций на каждом шаге обучения: 16.

Определение 1.13. Для изображения I объемом N байт и размером $H \times W$ пикселей, метрика BPP вычисляется следующим образом:

$$\mathrm{BPP}(I) = \frac{8N}{H \times W}$$

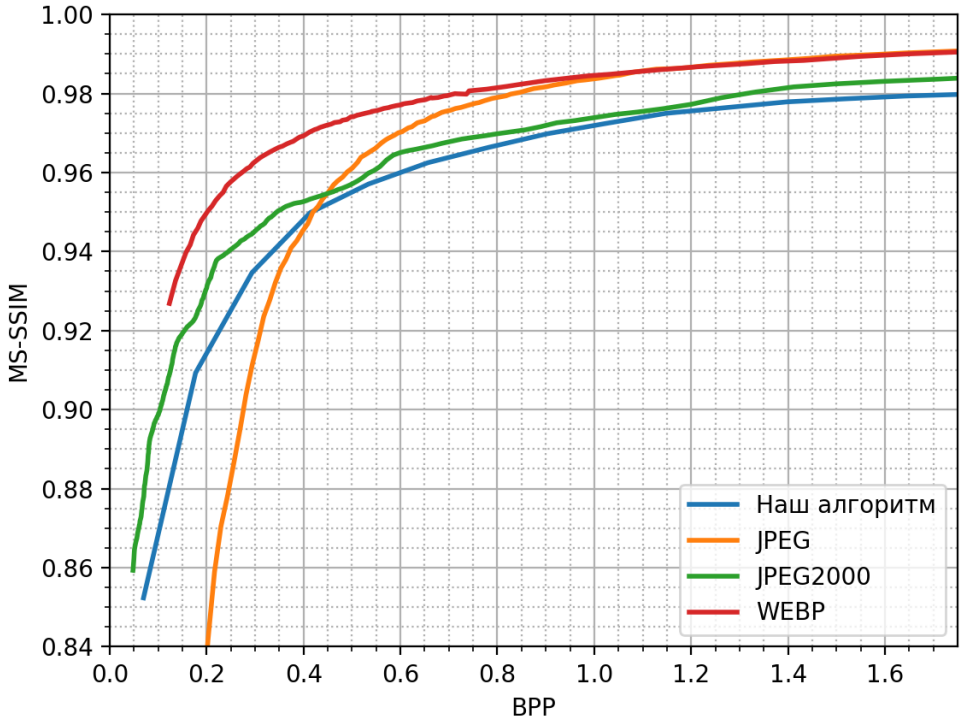

Рисунок 1.7 — Зависимость метрики MS-SSIM от метрики BPP для нескольких алгоритмов сжатия изображений

Из графика видно, что все алгоритмы сжатия начинают сильно терять качество при $\mathrm{BPP} < 0.8$, но хорошо видно, что при малых значениях $\mathrm{BPP} < 0.4$ лучшее качество выдаёт формат WEBP, за ним следует JPEG2000, на третем месте – наш алгоритм, и на последнем – классический JPEG.

Заключение В данной курсовой работе был рассмотрены подразделы машинного обучения, связанные с классификацией и сжатием изображений. Были введены необходимые определения и рассмотрены некоторые распространенные практики: в частности, использование свёрточных и рекуррентных нейросетей для решения этой задачи.

Так же были приведены результаты проведенных численных экспериментов: проведено сравнение различных параметров построения и обучения нейросети, проведены сравнения с уже существующими алгоритмами.

В Приложении 1 приведён исходный код программы, написанной на языке Python 3, который реализует темы, рассматриваемые в главах 1-3.

Алгоритм показывает хорошие результаты на наборе данных CIFAR-10: он может с точностью $p = 0.9$ предсказать, к какому классу относится изображение.

В Приложении 2 приведён исходный код программы, написанной на языке Python 3, который реализует темы, рассматриваемые в главах 4-6.

Алгоритм показывает результаты, лучшие по качеству, чем алгоритм JPEG при BPP < 0.4 и результаты, сравнимые по качеству с алгоритмом JPEG2000 при BPP > 0.4, хоть и уступающие ему.