АЛГОРИТМЫ ПОВЫШЕНИЯ КАЧЕСТВА ИЗОБРАЖЕНИЙ С ПРИМЕНЕНИЕМ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ

Сачивко Н.С.

Белорусский государственный университет информатики и радиоэлектроники, Минск, Республика Беларусь

Калугина М.А. – канд. физ.-мат. наук, доцент

Аннотация. В рамках данной статьи рассмотрена проблема обработки изображений низкого качества. Предложено два алгоритма повышения качества изображения: алгоритм увеличения размера изображения и алгоритм снижения степени его размытости. Описаны устройства моделей используемых нейросетей, применяемые инструменты, приведены результаты работы предложенных алгоритмов.

Ключевые слова. Машинное обучение, глубокое обучение, нейросети, нейросетевые модели, распознавание текста, увеличение изображений, повышение качества изображений, устранение размытия, повышение читабельности текста.

Сегодня цифровая фотография является неотъемлемой частью нашей жизни, однако далеко не все фотографии получаются удачными: возможности цифровых фото- и видеокамер не безграничны, нередко текст на изображениях оказывается нечитаемым за счёт их низкого разрешения, иногда изображение получается размытым по той или иной причине. Обычно плохое качество изображения является скорее неприятной особенностью, чем нерешаемой проблемой, ведь зачастую, скажем, некачественная портретная фотография может быть сделана повторно. Однако так происходит не всегда. Например, в криминалистической фотографии, для автоматической фиксации нарушений правил дорожного движения, в целях оцифровки медицинских карт, для регистрации происшествий на дорогах (например, с помощью автомобильных видеорегистраторов) качество полученных цифровых изображений может оказаться критически важным, особенно когда изображение объекта представлено лишь в одном экземпляре.

Один из подходов к решению проблемы наличия изображений низкого качества – повышение их качества. Например, можно увеличить их разрешение, снизить степень размытости, устранить шум. Однако возникает проблема, связанная со сложностью данных операций: на самом деле, задача восстановления исходного изображения из размытого или из изображения с низким разрешением имеет слишком большое количество решений [1], то есть получить точно исходное изображение из “испорченного” на практике невозможно. Именно поэтому предлагается использовать нейросети и методы глубокого обучения для решения указанных задач.

Далее будем считать, что исходное изображение представлено в формате RGB и имеет размер по крайней мере 30×30 пикселей. Рассмотрим два алгоритма: увеличения размера изображения и уменьшения степени его размытости.

Для помощи в решении поставленных задач взят язык программирования Python, библиотеки Pillow, PyTorch 2.1, Torchvision 0.16 и Matplotlib. В качестве аппаратного обеспечения используется домашний компьютер.

Важной особенностью библиотеки Torchvision является способ обработки изображений: встроенная функция to_tensor преобразует изображение в тензор, при этом стандартный

диапазон яркости цветов от 0 до 255 приводится к диапазону $[0.0; 1.0]$. Этим и обусловлен выбор последнего слоя нейронных сетей: наиболее подходящим является слой активаций Sigmoid, ведь именно он обеспечивает корректный диапазон значений.

Функция активации Sigmoid определяется следующим образом:

$$s(x) = \frac{1}{1 + e^{-x}}. \quad (1)$$

Также будет применяться функция активации ReLU:

$$r(x) = \max(0, x). \quad (2)$$

Первая задача, которую необходимо решить – это увеличение размера изображения. Для получения результата было принято решение использовать комбинацию свёрточной нейронной сети (Convolutional Neural Network или CNN) и такой же сети, работающей в обратном процессе (Deconvolutional Neural Network или DNN).

Поскольку исходное изображение представлено в формате RGB, количество каналов на входе и на выходе нейросети принимается равным трём.

Путём проведения экспериментов было установлено, что модель со следующими слоями является наиболее оптимальной (стрелкой указывается переход от исходного количества каналов к количеству каналов на выходе слоя):

Deconvolution: $3 \to 75$, ядро 6×6, шаг 2.

ReLU.

Convolution: $75 \to 75$, ядро 4×4, шаг 1.

ReLU.

Convolution: $75 \to 75$, ядро 4×4, шаг 1.

ReLU.

Deconvolution: $75 \to 3$, ядро 6×6, шаг 2.

Sigmoid.

Такая модель обеспечивает увеличение размера изображения в 4 раза.

В ходе проведения экспериментов изменялись следующие параметры модели:

Количество скрытых слоёв;

Количество каналов на входе и выходе скрытых слоёв;

Размеры ядра для слоёв свёртки и “деконволюции” (то есть слоёв, которые работают в обратном процессе по отношению к слоям свёртки, англ. “deconvolution”).

Также были осуществлены попытки использовать слои, увеличивающие размер изображения в несколько раз по заранее определённому алгоритму, вместо слоёв “деконволюции”: например, с применением алгоритма “ближайшего соседа” (nearest neighbor) [2]. Данный вариант архитектуры является допустимой альтернативой, однако экспериментальным путём установлено, что такая модель требует большего количества вычислительной мощности для обучения. Другими словами, обучение данной модели займёт большее количество времени по сравнению с моделью со слоями “деконволюции” при прочих равных условиях.

Для оценки результатов поставленных экспериментов использовались такие метрики, как скорость сходимости (снижения ошибки за одну эпоху обучения), точность обученной модели, скорость обучения модели (исчисляется на единицу времени).

Стоит отметить, что размеры ядра свёртки и шага для слоёв “деконволюции” выбраны неслучайно: с целью избежать эффекта “шахматной доски”. Если бы размер ядра свёртки не делился без остатка на шаг “деконволюции”, то в результате этого процесса мог бы возникнуть “неравномерный нахлёст” (англ. “uneven overlap”) фрагментов, который и привёл бы к возникновению указанного эффекта [3]. Такой эффект заметен невооружённым глазом, как представлено на рисунке 1.

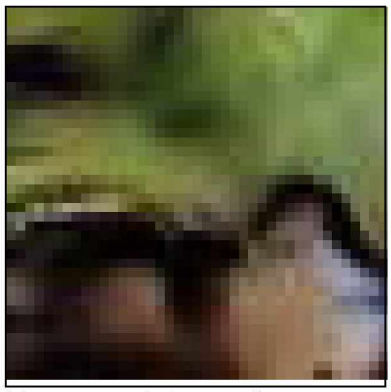

Рисунок 1 – Пример эффекта “шахматной доски” [4]

Изначально для обучения модели для увеличения размера изображения планировалось использовать набор изображений, содержащих части страниц общедоступных документов [5] – пример такого изображения представлен на рисунке 2.

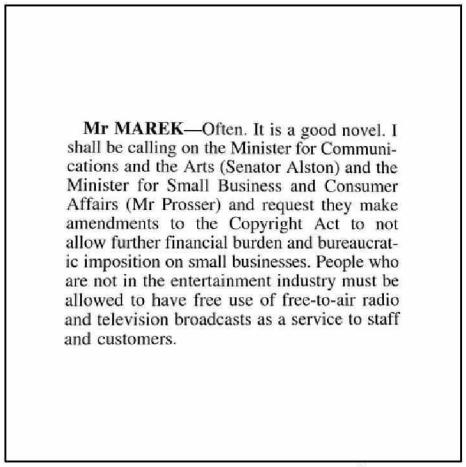

Рисунок 2 – Пример исходного изображения (BMVC)

Такой набор данных, безусловно, хорош в случае, когда модель должна обрабатывать только документы. Но у такой модели есть один существенный недостаток: она преобразует все изображения, в том числе цветные, в чёрно-белые. По этой причине было решено использовать другой набор данных.

Например, существует набор синтетически сгенерированных символов [6] – они также являются чёрно-белыми, но все изображения в наборе содержат всего два цвета: либо полностью белый, либо полностью чёрный, что позволяет перекрасить их. Пример исходного символа из данного набора представлен на рисунке 3.

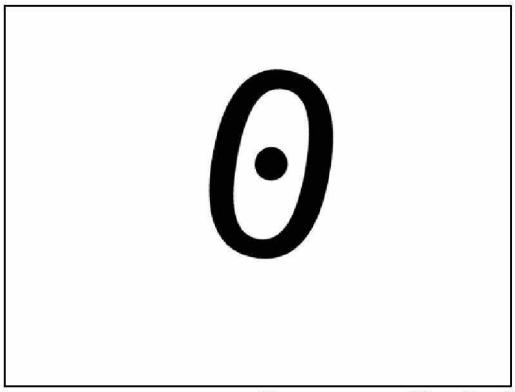

Рисунок 3 – Пример исходного изображения (Synthetic Character Set)

С указанным набором данных были проделаны следующие операции:
Все изображения обрезаны так, чтобы по краям не оставалось белого фона.

Чёрный цвет в 20% случаев заменён на случайный.

Полученные изображения объединены в коллекции, причём в 20% случаев белый фон заменён случайным цветом, а в 50% случаев итоговое изображение повёрнуто на случайный угол.

Итоговые коллекции обрезаны до одинакового размера.

Примеры изображений из полученного набора (назовём его “исходный набор”) представлены на рисунке 4.

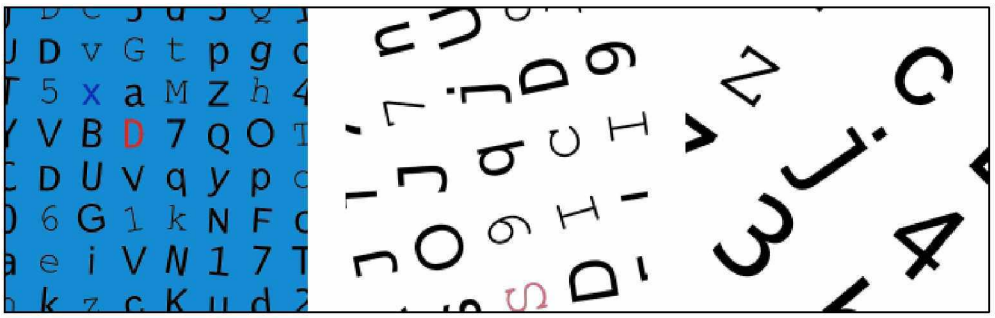

Рисунок 4 – Пример изображений исходного набора

Изображения из исходного набора были уменьшены в 4 раза с использованием следующих методов:

BOX;
HAMMING;
LANCZOS;
BICUBIC;
BILINEAR.

Примеры изображений из полученного набора (назовём его “итоговый набор”) представлены на рисунке 5.

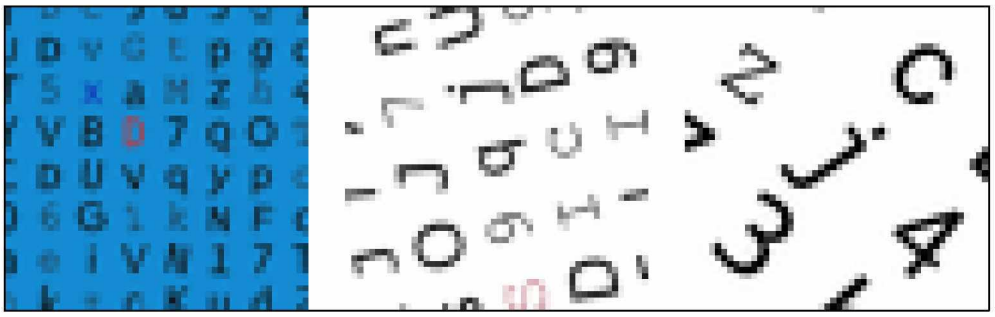

Рисунок 5 – Пример изображений итогового набора

Для обучения созданной модели изображения из исходного и итогового наборов были собраны в пары, 70% таких пар отобраны случайным образом непосредственно для обучения модели, остальные 30% используются для валидации результатов.

Стоит отметить, что данные не были нормализованы. Хоть процесс нормализации и позволял достигнуть меньшей ошибки модели, но это зачастую приводило к искажению цветов изображения в процессе обработки, что в итоге негативно сказывалось на восприятии человеком результата.

Модель для увеличения размера изображения обучалась из начального состояния, которое определялось случайным образом с помощью метода инициализации Ксавье (Xavier или Glorot initialization) [7].

Для обучения использовался модифицированный алгоритм стохастического градиентного спуска, именованный Adam. В ходе тестирования установлено, что именно данный алгоритм обеспечивает хорошую сходимость. В качестве меры ошибки была выбрана среднеквадратичная ошибка (MSELoss), которая определяется следующим образом:

$$l(X,Y) = \frac{\sum_{i=1}^{N}(X_i - Y_i)^2}{N}, \tag{3}$$

где $N = |X| = |Y|$; X – исходный набор данных, Y – итоговый набор данных.

Модель обучалась в течение 600 эпох (одна эпоха означает, что все данные из созданного набора были использованы для обучения ровно один раз). График зависимости среднеквадратичной ошибки (оранжевым – ошибка на данных для валидации, синим – на тренировочных данных) от эпохи представлен на рисунке 6.

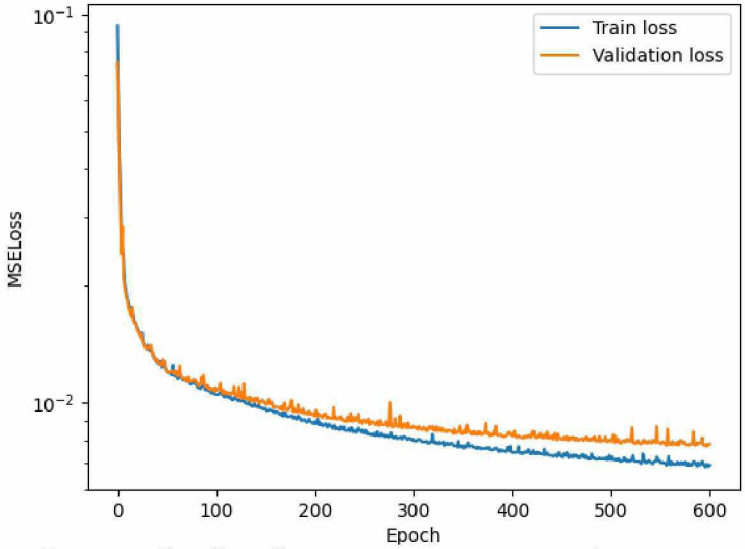

Рисунок 6 – Ошибка на валидационных (оранжевым)
и тренировочных (синим) данных,логарифмическая шкала

Пример обработки уменьшенных изображений представлен на рисунке 7, более детальное сравнение – на рисунке 8.

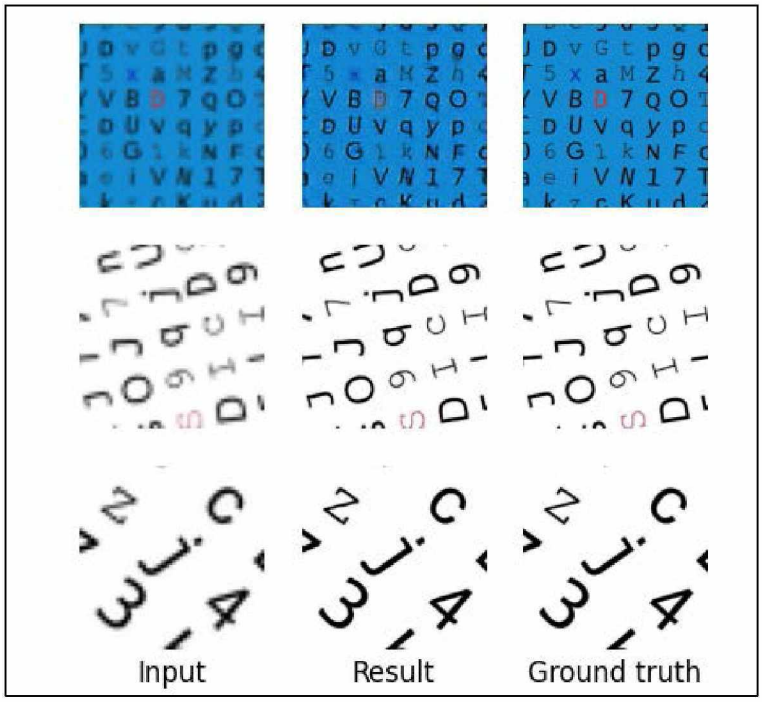

Рисунок 7 – Пример обработки уменьшенных изображений: слева – изображение итогового набора, посередине – результат его обработки, справа – соответствующее изображение исходного набора

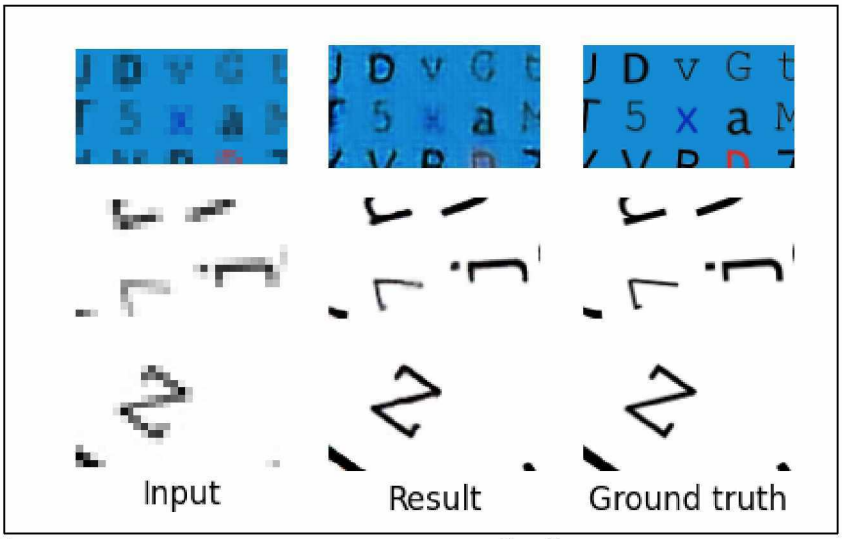

Рисунок 8 – Детальное сравнение результатов обработки уменьшенных изображений: слева – изображение итогового набора, посередине – результат его обработки, справа – соответствующее изображение исходного набора

Видно, что модель корректно выполняет свою функцию: текст становится более чётким при увеличении размера изображения. К сожалению, при работе с цветным фоном результат несколько хуже, чем при работе с белым фоном. Вероятно, это можно исправить путём увеличения объёма набора данных для тренировки, добавлением большего количества цветных изображений и обучением модели в течение большего количества эпох.

Для решения задачи снижения степени размытости изображения используется свёрточная нейронная сеть (Convolutional Neural Network или CNN). Как и ранее, исходное изображение представлено в формате RGB, в связи с чем количество каналов на входе и на выходе нейросети принимается равным трём.

Архитектура данной нейросети схожа с архитектурой ранее рассмотренной сети, но имеет существенные отличия (стрелкой указывается переход от исходного количества каналов к количеству каналов на выходе слоя):

ReflectionPad: по 4 пикселя снизу, сверху, слева и справа.
Convolution: $3 \to 75$, ядро 4×4, шаг 1.
ReLU.
Convolution: $75 \to 75$, ядро 2×2, шаг 1.
ReLU.
Convolution: $75 \to 75$, ядро 1×1, шаг 1.
ReLU.
Convolution: $75 \to 75$, ядро 2×2, шаг 1.
ReLU.
Convolution: $75 \to 3$, ядро 4×4, шаг 1.
Sigmoid.

В архитектуре данной сети используется новый слой: ReflectionPad. Необходимость его применения объясняется особенностью слоя свёртки: фактически для каждого канала применяется фильтр, в результате чего значения результирующей матрицы являются суммами произведений элементов фильтра на элементы исходной матрицы. Нетрудно установить, что в случае, когда используется размер фильтра больший, чем 1×1, результирующая матрица имеет меньший размер, чем исходная.

С целью сохранения исходного размера изображения оно искусственно расширяется на 4 пикселя во все стороны, причём расширяется неслучайно, а путём отражения частей изображения, расположенных на его границе. Для наглядности на рисунке 9 приведён пример использования ReflectionPad на 2 пикселя во все стороны для изображения размером 4×4 пикселя.

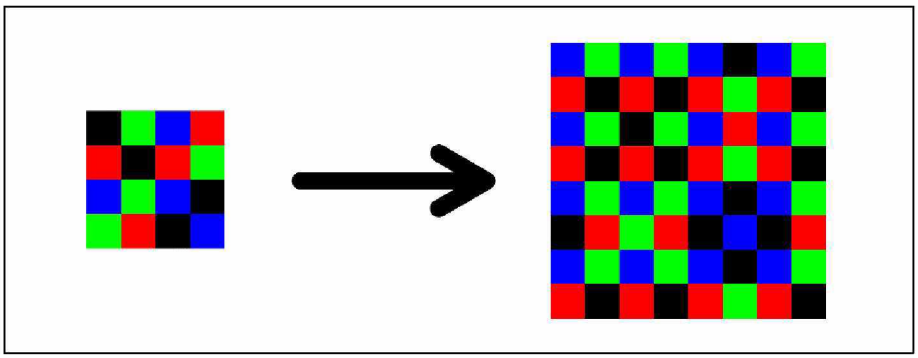

Рисунок 9 – Пример использования ReflectionPad

В качестве набора данных для обучения модели для снижения степени размытости изображений было решено использовать некоторую часть из набора данных с British Machine Vision Conference (BMVC) [4], а именно сгенерированные функции рассеяния точки.

Функция рассеяния точки (англ. Point Spread Function или PSF) представляет собой изображение, которое было бы получено при наблюдении точечного источника света в некоторых условиях. Именно эти условия и можно описать с помощью PSF. В частности, такие функции удобно применять при моделировании размытия.

Итак, из набора BMVC случайным образом были выбраны функции рассеяния точки. Пример выбранных функций представлен на рисунке 10.

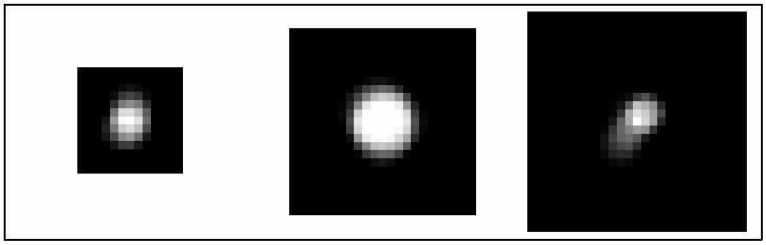

Рисунок 10 – Пример используемых PSF

Для применения PSF достаточно конвертировать рассмотренные представления функции в виде изображения в матрицу, состоящую из нормированных яркостей пикселей, и применить к исходному изображению фильтр, ядром которого выступает указанная матрица.

В качестве исходного был использован набор, аналогичный тому, что использовался для обучения модели для увеличения изображения (см. рисунок 4). К каждому изображению из исходного набора была применена соответствующая ему случайным образом выбранная функция рассеяния точки. Примеры изображений из полученного набора (назовём его “итоговый набор”) представлены на рисунке 11.

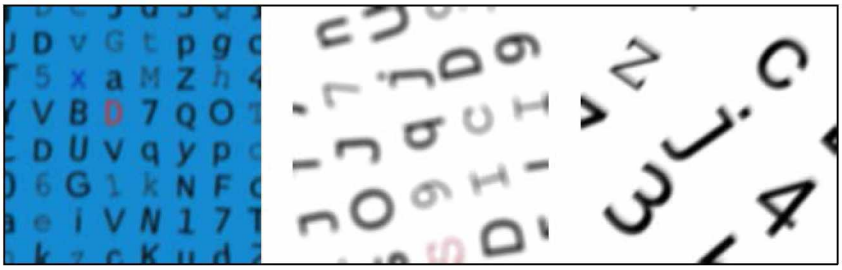

Рисунок 11 – Пример изображений итогового набора

Для обучения созданной модели изображения из исходного и итогового наборов были собраны в пары, 70% таких пар отобраны случайным образом непосредственно для обучения модели, остальные 30% используются для валидации результатов. Как и в случае с моделью для увеличения размера изображения, данные не были нормализованы.

Как и в ранее рассмотренном случае, модель для снижения степени размытости изображений обучалась из начального состояния, которое определялось случайным образом с помощью метода инициализации Ксавье (Xavier или Glorot initialization) [6].

В качестве меры ошибки была выбрана среднеквадратичная ошибка (MSELoss). Для обучения использовался тот же модифицированный алгоритм стохастического градиентного спуска: Adam. Аналогично предыдущей модели данный алгоритм показал хорошую скорость сходимости и позволил снизить итоговую ошибку.

Модель обучалась на протяжении 1000 эпох. График зависимости среднеквадратичной ошибки (оранжевым – ошибка на данных для валидации, синим – на тренировочных данных) от эпохи представлен на рисунке 12.

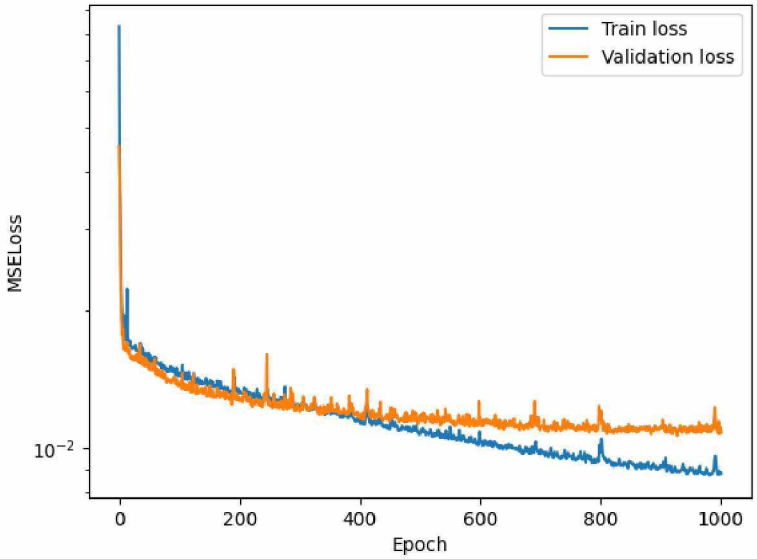

Рисунок 12 – Ошибка на валидационных (оранжевым)
и тренировочных (синим) данных, логарифмическая шкала

Видно, что ошибка на данных для валидации на последних эпохах практически не уменьшалась, в то время как ошибка на тренировочных данных продолжала падать. Это может свидетельствовать о переобучении модели. Впрочем, нельзя сказать, что ошибка на валидационных данных начала расти, да и, как будет видно далее, нельзя утверждать о плохом в том или ином смысле поведении модели на таких данных.

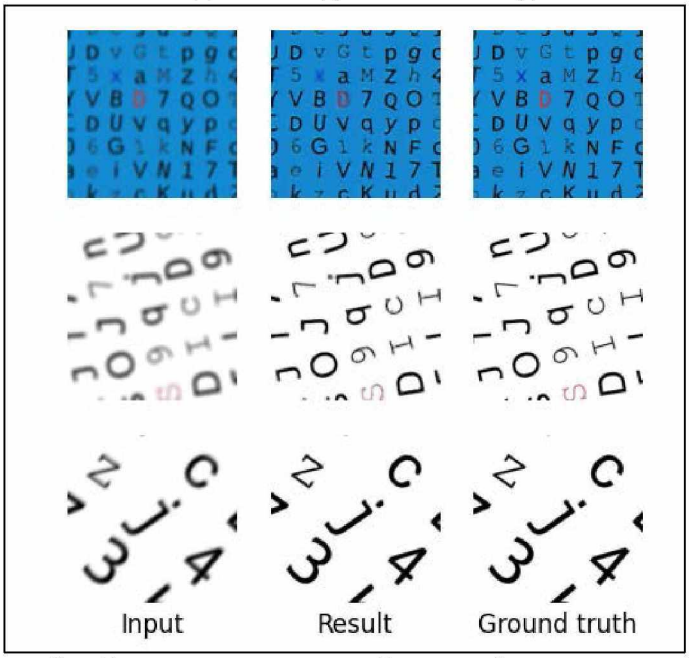

Рисунок 13 – Пример обработки размытых изображений: слева – изображение итогового набора, посередине – результат его обработки, справа – соответствующее изображение исходного набора

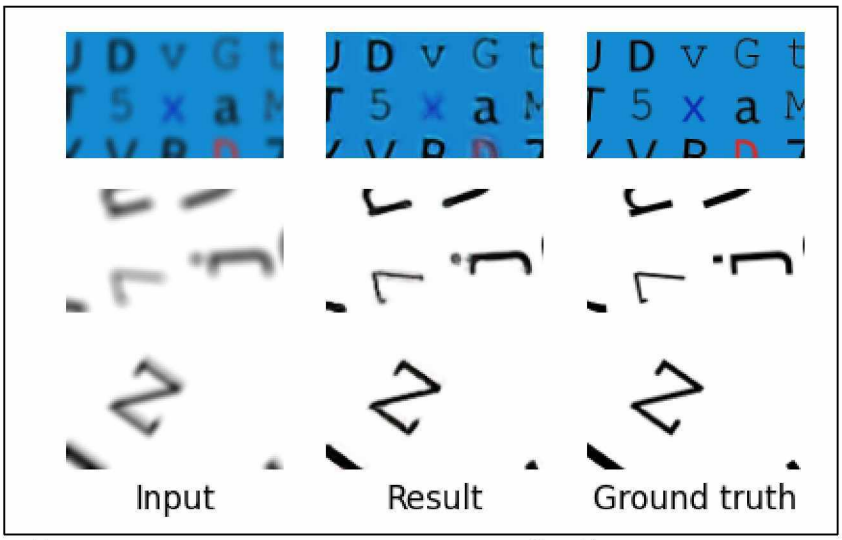

Рисунок 14 – Детальное сравнение результатов обработки размытых изображений: слева – изображение итогового набора, посередине – результат его обработки, справа – соответствующее изображение исходного набора

Полученные модели можно применять независимо друг от друга, однако зачастую низкое качество изображения подразумевает именно комбинацию размытости и низкого разрешения. Вопрос рациональности объединения двух соответствующих функций в одной модели требует более детального исследования, однако, основываясь на полученных результатах, более предпочтительным порядком действий в случае обработки размытого изображения низкого разрешения зачастую окажется сначала увеличение размера изображения, а затем снижение степени его размытости.

Предложенные алгоритмы успешно решают задачи увеличения размера изображения в 4 раза и снижение степени размытости текста. Примечательно, что рассмотренные нейросети могут обрабатывать изображения практически любых размеров (за исключением самых маленьких). Достигнутые результаты теоретически можно улучшить, например, путём модификации архитектуры нейронных сетей, изменения используемых наборов данных, использования профессионального оборудования для быстрого обучения более глубоких сетей, пред- или постобработки изображений с применением иных методов.

Список использованных источников:

1. Learning to super-resolve blurry face and text images / X. Xu [et al]. // Proceedings of the IEEE international conference on computer vision. – 2017. – P. 251-260.
2. Matthews, E. Nearest Neighbor Classication for Classical Image Upsampling / E. Matthews, N. Prate // arXiv preprint arXiv:2403.19611. – 2024.
3. Odena, A. Deconvolution and checkerboard artifacts / A. Odena, V. Dumoulin, C. Olah // Distill. – 2016. – Т. 1. – №. 10. – С. e3.
4. Donahue, J. Adversarial feature learning / J. Donahue, P. Krähenbühl, T. Darrell // arXiv preprint arXiv:1605.09782. – 2016.
5. BMVC OCR test data [Электронный ресурс]. – Режим доступа: https://www.fit.vutbr.cz/~ihradis/CNN-Deblur/BMVC_OCR_test_data.tar.gz. – Дата доступа: 25.11.2023.
6. Synthetic Character Set [Электронный ресурс]. – Режим доступа: https://www.kaggle.com/datasets/shreyasubbu/synthetic-character-set. – Дата доступа: 10.12.2023.
7. Glorot, X. Understanding the difficulty of training deep feedforward neural networks / X. Glorot, Y. Bengio // Proceedings of the thirteenth international conference on artificial intelligence and statistics. – JMLR Workshop and Conference Proceedings, 2010. – P. 249-256.