

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное бюджетное образовательное учреждение  
высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)»**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**по курсу**

**«Data Science»**

**Применение алгоритмов глубокого обучения к задачам шумопонижения  
изображений**

Слушатель

Лаврентьев Василий Юрьевич

Москва, 2023

## Содержание

Введение.....	1
1 Аналитическая часть.....	3
1.1 Постановка задачи .....	3
1.2 Обзор использованных архитектур.....	5
1.2.1 Автоэнкодеры.....	5
1.2.2 dnCNN .....	6
1.2.3 Unet .....	6
2 Практическая часть.....	9
2.1 Методология .....	9
2.2 Автоэнкодер.....	11
2.3 dnCNN .....	12
2.4 Unet .....	13
2.5 Реализация второго этапа.....	14
2.6 Построение модели реального шума .....	17
Заключение .....	28
Библиографический список .....	29

## Введение

Одной из особенностью устройств цифровой регистрации изображений является тот факт, что из-за несовершенства систем формирования изображения, средств регистрации и передачи, цифровые изображения содержат не только полезную информацию (собственно, изображение), но некоторые случайные искажения, вносимые, как самим светочувствительным сенсором, так и электронными компонентами устройств. Такие искажения сигнала носят названия шума – дефекта изображения, проявляющегося в виде случайным образом расположенных точек, имеющих размеры, близкие к размерам пикселя и интенсивность, отличающуюся от интенсивности полезного сигнала.

В ряде случаев, при хороших условиях съемки и отсутствии каких-либо особых требований к качеству изображения подобный шум не вносит скольнибудь значимых изменений в итоговое изображение и, чаще всего, просто игнорируется, однако при плохих условиях съемки, а также в случаях, когда к получаемому результату предъявляются определенные требования шум на получаемом изображении может приводить к серьезному ухудшению качества последнего и потребуются применение мер для снижения его уровня.

Таким образом, очистка от шума является одной из основных задач цифровой обработки изображений. Актуальность этой задачи в современных условиях повсеместного распространения цифровых фотокамер, смартфонов, камер видеонаблюдения и других аналогичных устройств возрастает многократно. Особую актуальность задача принимает в случаях практического применения компьютерного зрения, сегментации, распознавания образов и классификации, т.к. загрязнение исходного изображения шумом напрямую влияет на качество работы алгоритмов и получаемые результаты.

Целью данной работы является обзор применения ряда архитектур нейронных сетей к задачам снижения шумов на изображении и их сравнительный анализ; автором работы также предложен новый подход,

модифицирующий алгоритм на этапе формирования зашумленных изображений и позволяющий получить картину шума, близкую к реальной.

# 1 Аналитическая часть

## 1.1 Постановка задачи

В данной работе рассматривается задача очистки цифровых изображений от шумов с применением алгоритмов, основанных на использовании нейронных сетей. К настоящему времени существует множество методов, реализующих различные архитектуры нейронных сетей и позволяющих в той или иной мере решить данную задачу. Помимо этого, существуют классические методы шумоподавления, такие как линейное усреднение по соседним пикселям, медианный фильтр, вейвлет-преобразования и др.

В рамках данной работы будут рассмотрены несколько архитектур: автоэнкодер, DNCNN, Unet.

В качестве исходных данных использован набор изображений Imagenet [15]; в целях ускорения процесса обучения данный датасет использован не в полном объеме: для обучающей и тестовой выборок взяты 2.500 изображений (2000 – обучающая, 500 – тест), для валидационной выборки – 1000. Все изображения приведены к размеру (256,256,3) с сохранением пропорций, значения интенсивности в каждом канале приведены к диапазону [0, 1].

Формирование зашумленных изображений производилось путем добавления к исходным изображениям гауссова шума (среднее значение – 0, стандартное отклонение – 1). В целях формирования более реальной картины, конкретный уровень шума, добавляемый к каждому из изображений, определялся дополнительным множителем, выбираемым случайным образом из диапазона [0.1, 0.3].

Для оценки качества работы алгоритмов применялись следующие показатели: пиковое отношение сигнал-шум (PSNR, peak signal to noise ratio):

$$PSNR = 10 \log_{10} \frac{MAX_I^2}{MSE} \quad (1)$$

где MSE – среднеквадратичное отклонение,

и индекс структурного сходства (SSIM, structural similarity index measure):

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}, \quad (2)$$

где

$\mu_x, \mu_y$  – средние значения первого и второго изображения соответственно,

$\sigma_x, \sigma_y$  – среднеквадратичные отклонения,

$\sigma_{xy}$  – ковариация,

$c_1, c_2$  – поправочные коэффициенты, определяемые как

$$c_1 = (0.01L)^2, c_2 = (0.03L)^2, \quad (3)$$

где  $L$  – динамический диапазон изображения.

Для расчета данных показателей будем использовать методы `psnr()` и `ssim()` из библиотеки `tensorflow`.

В качестве эталонного использовалось исходное незашумленное изображение. Финальные показатели рассчитывались как усредненные значения по всей валидационной выборке.

В качестве функций потерь были опробованы:  $MAE$ ,  $MSE$  и комплексный показатель, основанный на [2]  $SSIM\_L2$ , рассчитываемый как

$$SSIM\_L2 = (1 - SSIM) + MSE \quad (4)$$

Обучение моделей, построенных в ходе выполнения работы производилось на ПК с процессором AMD Atlon X6, 32 Гб ОЗУ и графическим ускорителем Nvidia RTX 3060 в среде Jupyter Notebook. Часть экспериментов проводилась с использованием ресурсов Kaggle.

## 1.2 Обзор использованных архитектур

### 1.2.1 Автоэнкодеры

Автоэнкодер – нейронная сеть прямого распространения, которая восстанавливает входной сигнал на выходе (рис.1).

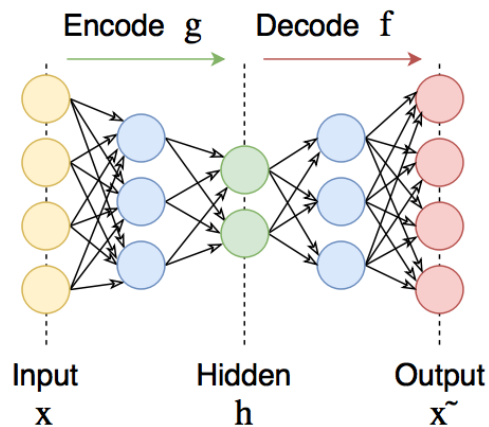


Рисунок 1 - Автоэнкодер

Автоэнкодеры сжимают входные данные для представления их в latent-space (скрытое пространство), а затем восстанавливают из этого представления output (выходные данные). Цель — получить на выходном слое отклик, наиболее близкий к входному. Автоэнкодеры конструируются таким образом, чтобы не иметь возможность точно скопировать вход на выходе. Обычно их ограничивают в размерности латентного (скрытого) пространства. Входной сигнал восстанавливается с ошибками из-за потерь при кодировании, но, чтобы их минимизировать, сеть вынуждена учиться отбирать наиболее важные признаки и отбрасывать несущественные; именно эта особенность является ключевой для применения архитектуры автоэнкодеров к задачам шумоподавления изображений: на вход подается зашумленное изображение, на выходе сеть пытается построить изображение, очищенное от шума.

### 1.2.2 dnCNN

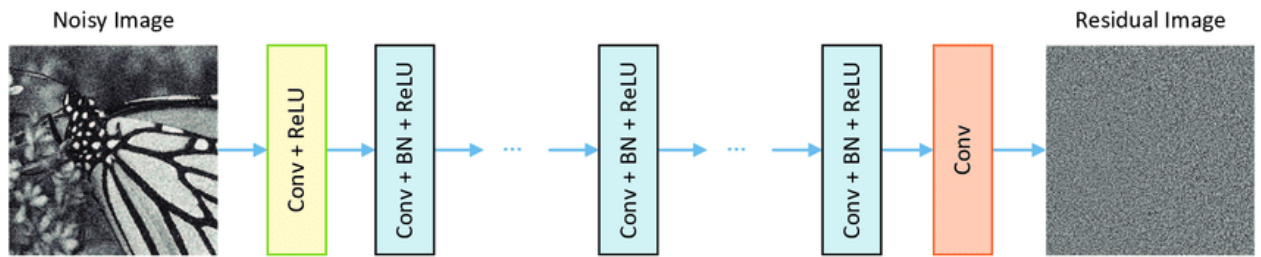


Рисунок 2 - Оригинальная архитектура dnCNN

Предложенная в [5] архитектура dnCNN представляет собой глубокую сверточную сеть, имеющую на входе Conv+Relu слой и N блоков Conv+BN+Relu далее (рис. 2). Последний сверточный слой сети представляет собой Conv блок с размерностью 3 для соответствия входному изображению. На выходе сеть формирует «residual image» (по сути шумовая карта), которое на выходе сети вычитается из входного сигнала и в итоге дает изображение, очищенное от шума. В оригинальной работе каждый из блоков содержал Conv слой с 64 фильтрами и ядром 3x3. Входной слой также имел Conv 64x3x3. Функция активации – Relu, функция потерь – MSE. Инициализация весов - Xavier initialization [14].

### 1.2.3 Unet

U-Net - это сверточная нейронная сеть, разработанная для сегментации изображений. Она была представлена в статье "U-Net: Convolutional Networks for Biomedical Image Segmentation" в 2015 году [11] и была изначально предназначена для сегментации медицинских изображений. U-Net позволяет точно выделять объекты на изображении и является основой для многих приложений в области компьютерного зрения и медицины.

Сеть содержит сверточную (слева) и разверточную части (справа), поэтому архитектура похожа на букву U, что и отражено в названии. На каждом шаге количество каналов признаков удваивается.



Сверточная часть похожа на обычную свёрточную сеть, содержит два подряд свёрточных слоя  $3 \times 3$ , после которых идет слой ReLU и пулинг с функцией максимума  $2 \times 2$  с шагом 2.

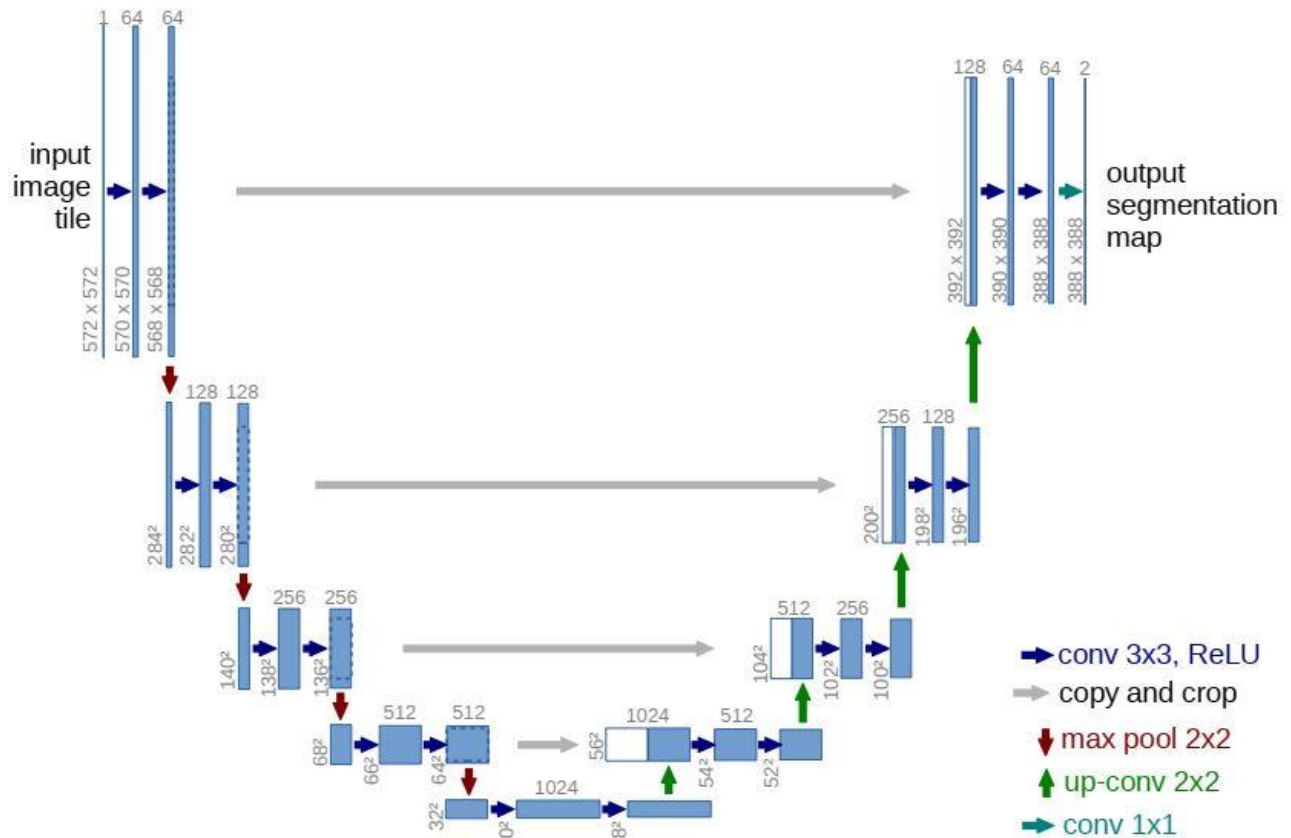


Рисунок 3 - Архитектура Unet

Каждый шаг разверточной части содержит слой, обратный пулинг, который расширяет карту признаков, после которого следует свертка  $2 \times 2$ , которая уменьшает количество каналов признаков. После идет конкатенация с картой признаков из сжимающего пути и две свертки  $3 \times 3$ , после каждой из которой идет ReLU. На последнем слое свертка  $1 \times 1$  используется для приведения каждого 64-компонентного вектора признаков до требуемого количества классов.

Всего сеть имеет 23 свёрточных слоя.

Несмотря на то, что архитектура изначально предназначалась для задач сегментации, существует множество ее реализаций применительно и к другим

задачам обработки изображений (снижение шумов, восстановление, повышение разрешения и др.), как в качестве самостоятельной сети, так и в виде блока генератора в GAN.

## 2 Практическая часть

### 2.1 Методология

Работа состояла из трех этапов. На первом этапе проводились эксперименты с целью подбора наиболее оптимальных параметров для каждой из сетей – тестовое обучение с числом эпох 30 и уменьшенными размерами обучающей/тестовой выборки (1600 обучение, 400 тест). Были опробованы функции активации: Relu, LeakyRelu, Gelu [3]; оценка результата производилась по полученным итоговым показателям PSNR, SSIM; также оценивался ход обучения (стабильность процесса обучения, наличие/отсутствие скачков функции потерь, контроль переобучения) и величина ошибки на обучающей/тестовой выборке по завершении обучения. Также на этом этапе проводились эксперименты с модификацией архитектуры в части выбора числа слоев, количества фильтров и т.п. Число слоев сети, число нейронов в слое, количество фильтров в сверточных слоях подбирались эмпирически. Выбранные таким образом конкретные реализации архитектуры и функции активации использовались далее для трех тестовых запусков (30 эпох, выборка 1600 обучение, 400 тест) с тремя различными функциями потерь – MAE, MSE, SSIM\_L2. Полученные результаты по каждой из построенных архитектур сведены в таблицу 1, как можно видеть во всех трех случаях самое высокое значение SSIM было получено с применением комплексной функции потерь SSIM\_L2. Некоторое несоответствие между оценками по PSNR и SSIM связано с особенностями самих показателей – PSNR оценивает соотношение «сигнал-шум» и в некоторых случаях может дать более высокую оценку изображению с более низким уровнем шума, но меньшей детализацией. SSIM же оценивает «степень похожести» и потеря мелких деталей на изображении, равно как и сдвиг цветового баланса» может заметно снизить оценку, при том, что для этого же изображения PSNR такие потери может проигнорировать и дать в результате более высокое значение. Таким образом, оценка результата производилась,

опираясь на значение именно SSIM при учете общего визуального восприятия изображения, а также динамики процесса обучения.

Таблица 1. Результаты тестов первого этапа

Архитектура	Loss function	Результат после 30 эпох			Примечание
		loss	PSNR	SSIM	
AE	Adamax, mae	loss: 0.0474 - val_loss: 0.0552	22.99	0.62	
	Adamax, mse	loss: 0.0045 - val_loss: 0.0050	23.53	0.64	
	Adamax, ssim+l2	loss: 0.3076 - val_loss: 0.3303	23.03	0.67	
dnCNN	Adamax, mae	loss: 0.0427 - val_loss: 0.0480	21.93	0.67	
	Adamax, mse	loss: 0.0035 - val_loss: 0.0060	24.34	0.69	
	Adamax, ssim+l2	loss: 0.4657 - val_loss: 0.4881	22.85	0.69	40 эпох
res dnCNN	Adamax, mae	loss: 0.0391 - val_loss: 0.0398	24,23	0,69	
	Adamax, mse	loss: 0.0034 - val_loss: 0.0033	25,42	0,69	
	Adamax, ssim+l2	loss: 0.2991 - val_loss: 0.3009	24.17	0.70	40 эпох
Unet	Adamax, mae	loss: 0.0525 - val_loss: 0.0517	23.10	0.62	40 эпох
	Adamax, mse	loss: 0.0056 - val_loss: 0.0053	23.19	0.61	
	Adamax, ssim+l2	loss: 1.3919 - val_loss: 1.4290	21.32	0.66	45 эпох, lr=0,001

Изображения, полученные с использованием каждой из архитектур в варианте, показавшем лучшую оценку по SSIM приведены на рис 4.

Второй этап заключался в обучении каждой из сетей с параметрами, подобранными по результатам выполнения первого этапа. Обучение производилось на большем наборе данных, число эпох – 30 (для некоторых архитектур понадобилось большее число эпох). Показатели PSNR, SSIM рассчитывались по валидационной выборке. Полученные результаты приведены в таблице 2.

Третий этап. Формирование финального результата - сеть, показавшая лучший результат обучена на двух моделях шума – гауссовом шуме и предложенной в данной работе модели реального шума (далее в 2.7). Оценка результата производилась на валидационной выборке с расчетом PSNR, SSIM, а также на реальных зашумленных изображениях.

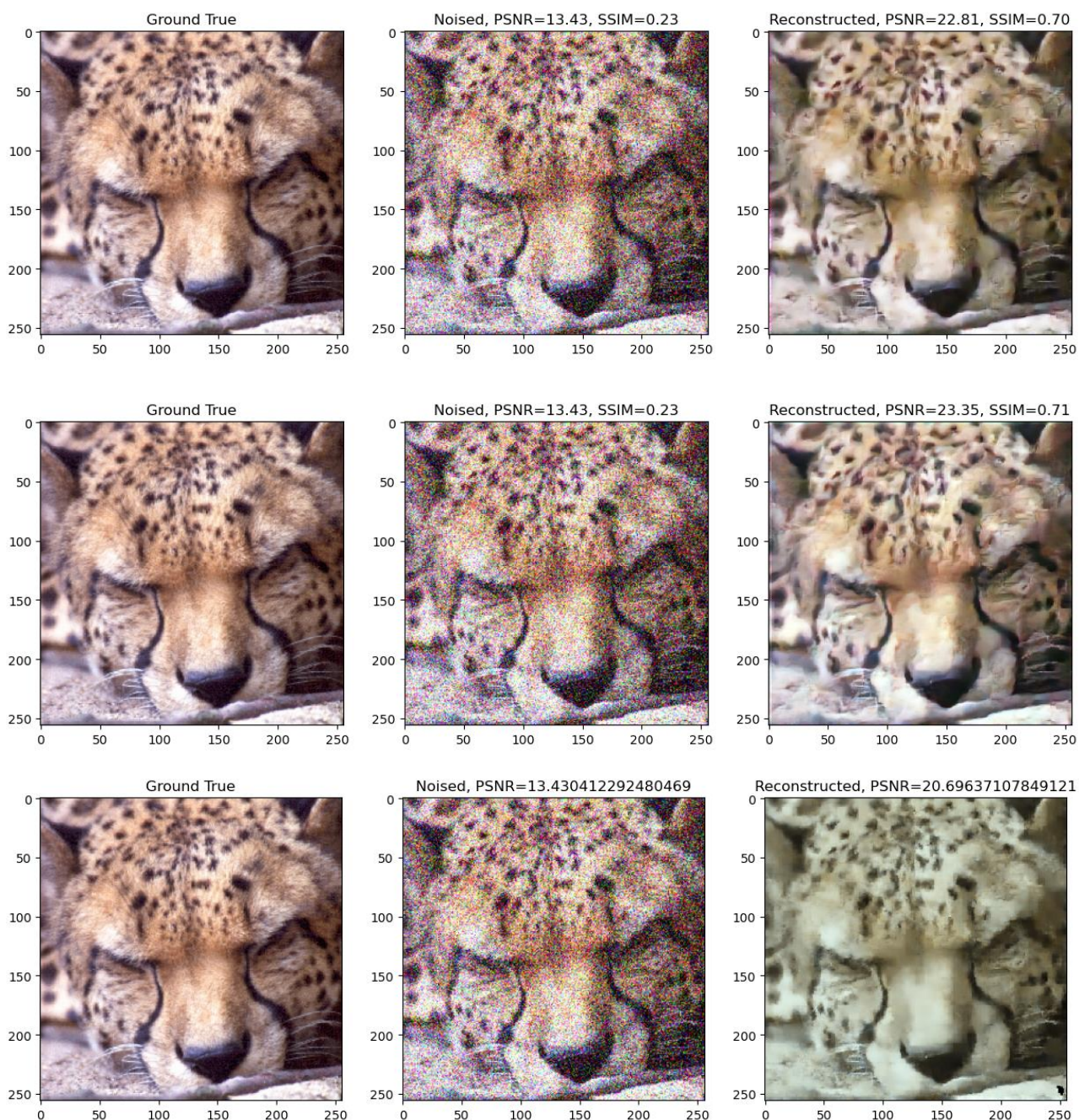


Рисунок 4 - Визуальные результаты первого этапа (сверху вниз - автоэнкодер, res dnCNN, Unet)

## 2.2 Автоэнкодер

Исходный код приведен в файле ae.ipynb

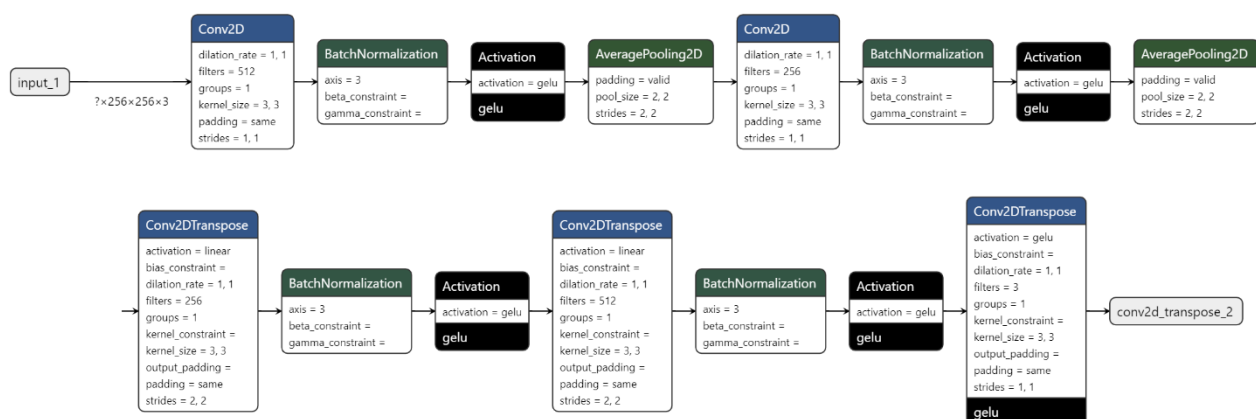


Рисунок 5 - Реализованный автоэнкодер

Реализованный вариант архитектуры изображен на рис 5. Построенная сеть имеет по два сверточных слоя в энкодере и декодере, слои BatchNormalization и слои AveragePooling2D в энкодере. В декодере пулингслои не использованы, а повышение дискретизации выполнено непосредственно в сверточных слоях через задание параметра stride=2. Функция активации – Gelu. Наилучший результат на первом этапе -PSNR 23.03, SSIM 0.67 после 30 эпох обучения.

## 2.3 dnCNN

Исходный код приведен в файлах dncnn.ipynb, dncnn\_res.ipynb.

Реализован вариант архитектуры из оригинальной статьи, число conv блоков (без учета входного и выходного слоев) – 16, инициализация весов Xavier не использовалась, функция потерь – SSIM\_L2. Наилучший результат первого этапа – PSNR 22.85, SSIM 0.69.

Помимо оригинальной архитектуры в рамках работы была предложена модификация – в каждом из conv блоков добавлены short connection от входа



блока на его выход. Остальные слои, функция активации и пр. оставлены без изменений. Схема архитектуры приведена на рис 6.

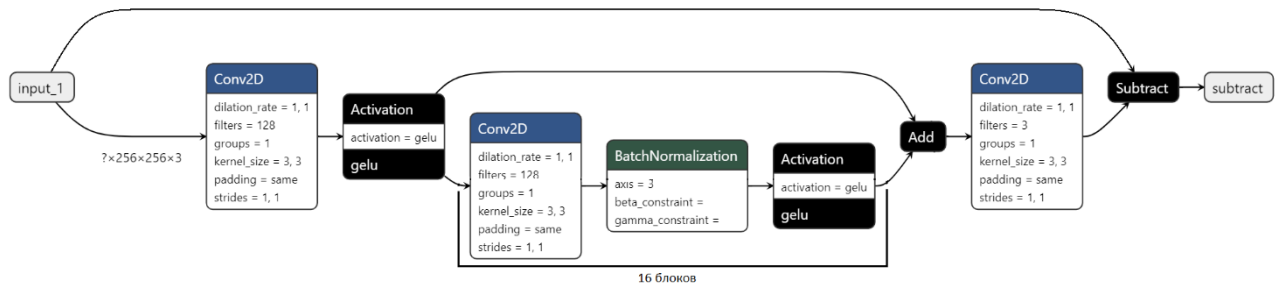


Рисунок 6 – Модифицированный вариант dnCNN

Такое решение позволило несколько улучшить результат, показанный сетью, но при этом пришлось увеличить число эпох обучения до 40: PSNR 23.17, SSIM 0.70

## 2.4 Unet

Исходный код приведен в файле unet.ipynb

Реализован вариант оригинальной архитектуры с некоторыми изменениями: помимо соединений внутри между соответствующими блоками up и down добавлено соединение (-) от входа на выход; эксперименты показали, что такая модификация показала чуть лучшие результаты – PSNR 21.32, SSIM 0.66. Здесь стоит отметить, что из всех трех рассмотренных сетей Unet в данной задаче показывал наихудший результат, вне зависимости от примененных функций активации и функций потерь; максимальный показатель SSIM, который удалось получить на Unet – 0.66, что несколько меньше, чем показал автоэнкодер и заметно меньше, чем удалось получить на модифицированной dnCNN.

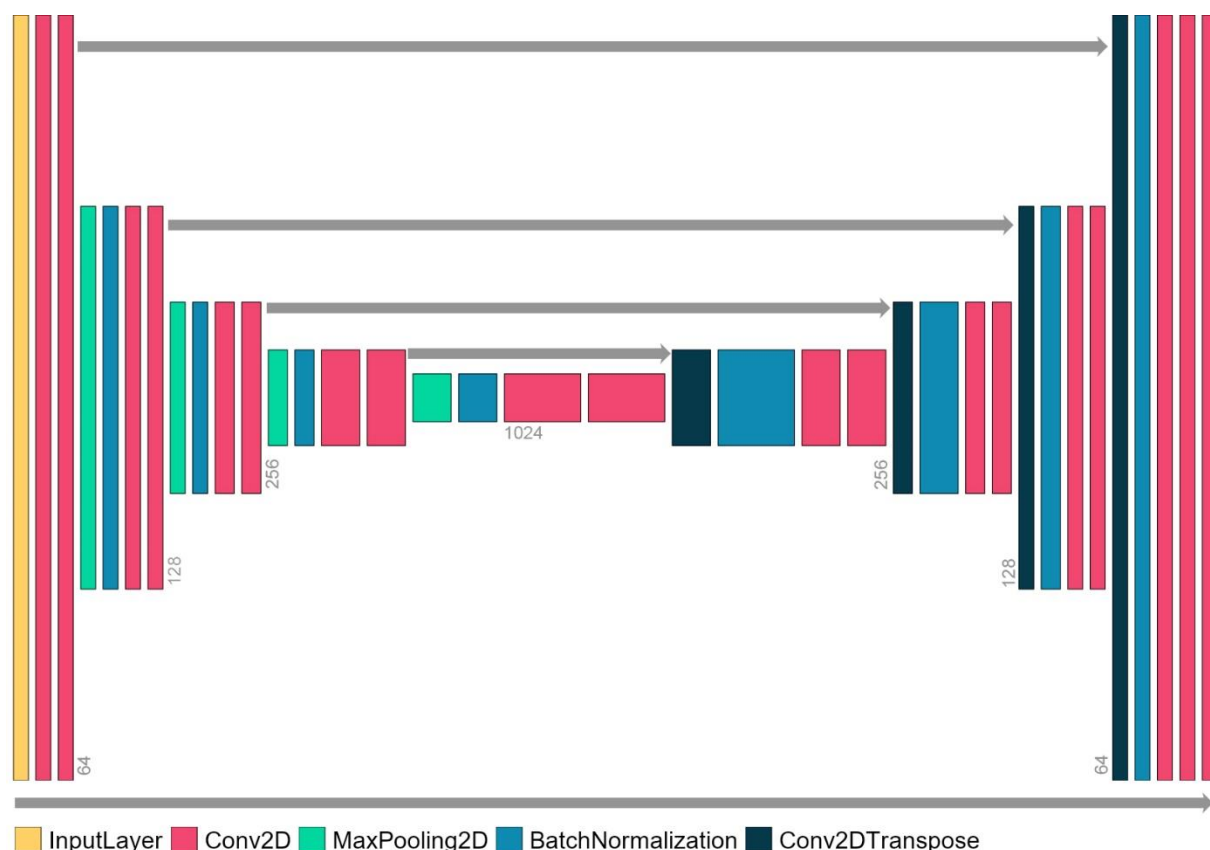


Рисунок 7 – Реализация Unet

## 2.5 Реализация второго этапа

Варианты реализаций, показавшие наилучшие результаты по итогам первого этапа работы обучены на большем объеме исходных данных – 2500 изображений обучение и тест (2000 и 500 соответственно) и 1000 изображений - валидация. Полученные по итогу результаты приведены в таблице 2. Графики процесса обучения приведены на рис.10. Примеры изображений – на рис 8, 9.

Таблица 2. Результаты тестов второго этапа

Архитектура	Число эпох	Лосс на конец обучения	обучение/тест		валидация	
			PSNR	SSIM	PSNR	SSIM
AE	30	loss: 0.3464 - val_loss: 0.3493	20,18	0,65	20,18	0,63
res dnCnn	40	loss: 0.2930 - val_loss: 0.2914	24,39	0,71	23,51	0,66
unet	45	loss: 0.6286 - val_loss: 0.6450	13,05	0,4	15,01	0,57



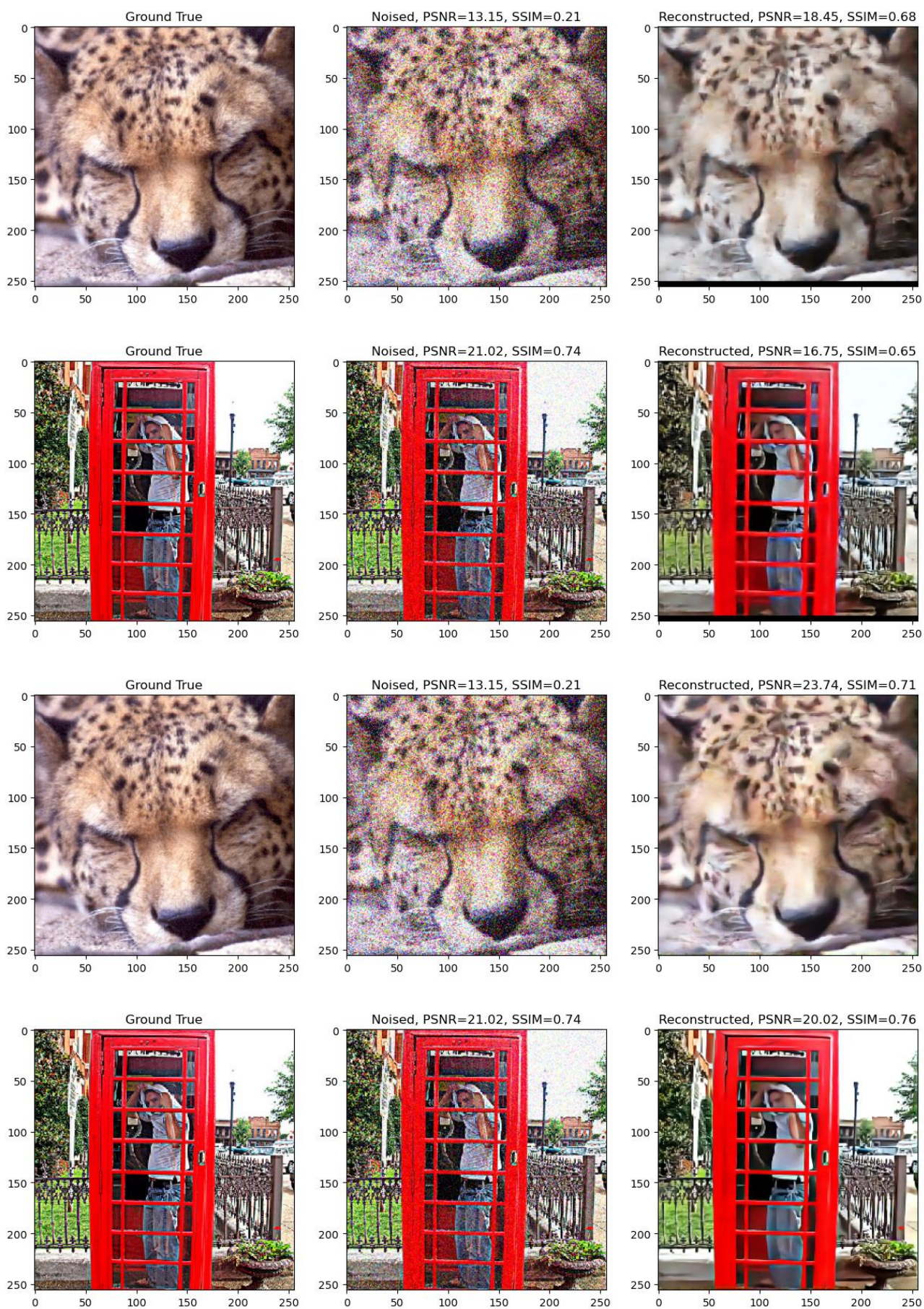


Рисунок 8 - вверху – автоэнкодер (train/val), внизу – Unet (train/val)



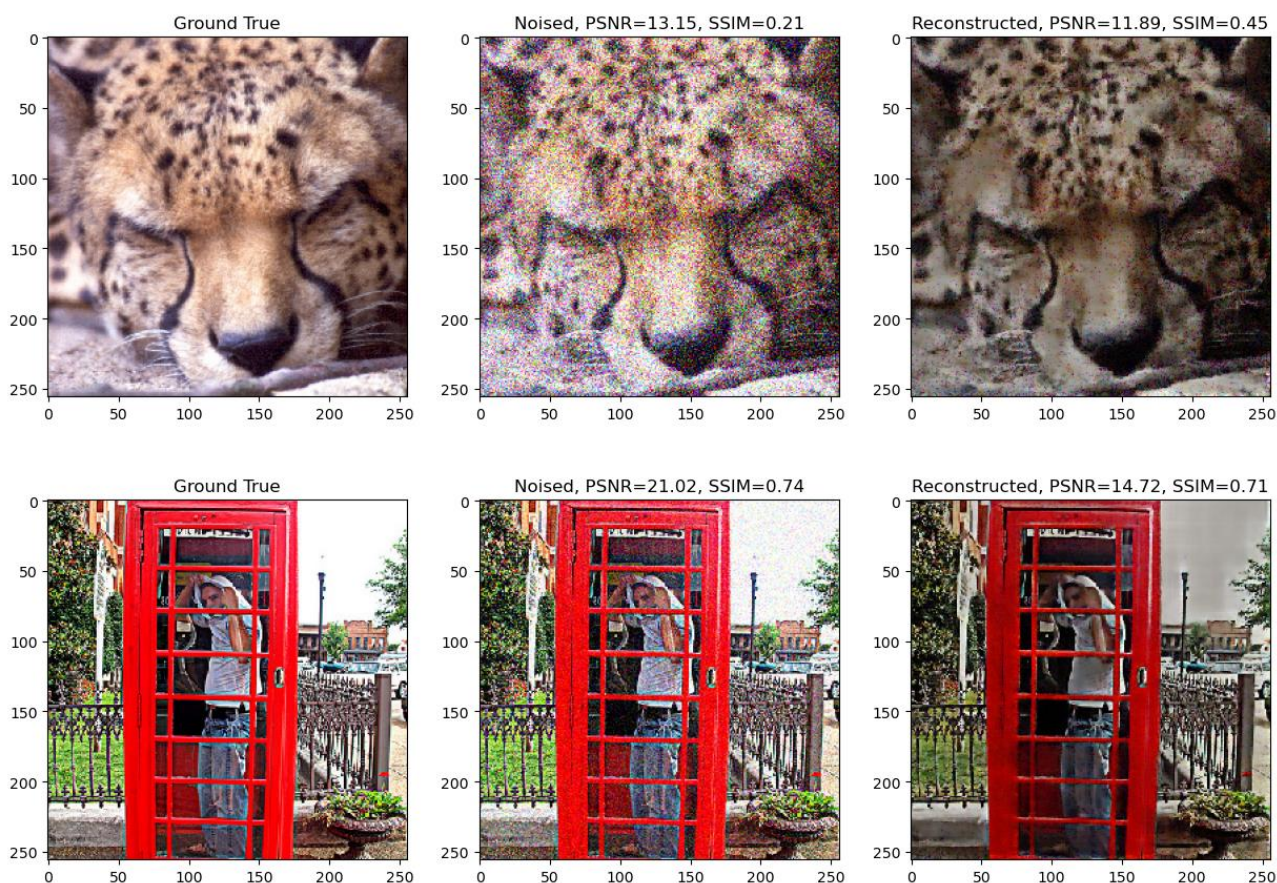


Рисунок 9 - Модификация dnCNN – res dnCNN, обучение/тест сверху, валидация – внизу

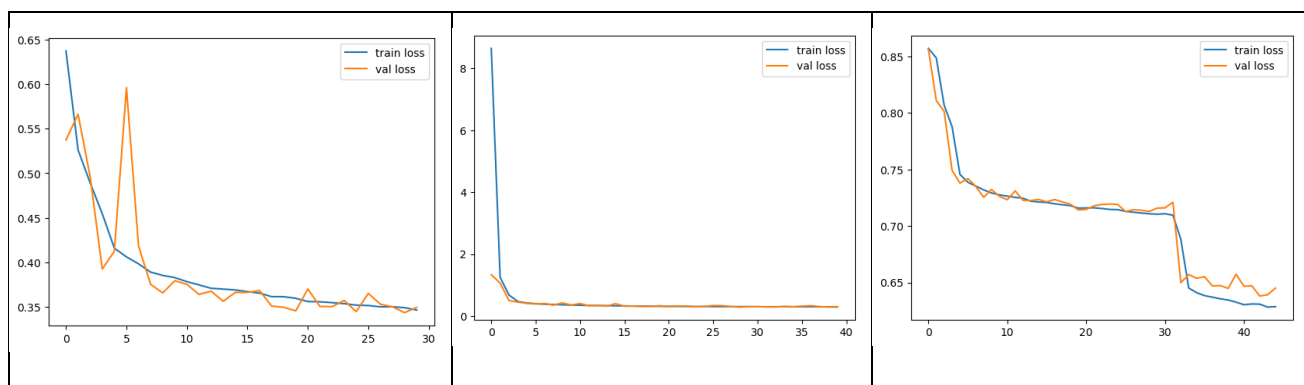


Рисунок 10 – Динамика обучения второго этапа

По итогам второго этапа можно сделать вывод, что наилучшим из рассмотренных вариантов оказалась модификация dnCNN с добавленными short connection во внутренних conv блоках. Данная реализация показала PSNR 23.51

и SSIM 0.66, рассчитанные как среднее по валидационной выборке. При этом стоит отметить, что оценки, полученные на валидации несколько хуже оценок, данных по обучающей и тестовой выборкам. Визуальная оценка полученных восстановленных изображений (рис) показывает заметные локальные размытия изображений и, как следствие, снижение детальности. Учитывая, что признаков переобучения по графикам не выявляется, можно сделать вывод, что обобщающие способности построенных реализаций хуже ожидаемых. Однако, здесь стоит сделать оговорку, что уровень добавленного шума при формировании зашумленного набора данных был выбран достаточно высоким и, как можно видеть из примеров изображений, значительная часть мелких деталей скрыта за шумами.

## **2.6 Построение модели реального шума**

Исходный код – файл `real_noise.ipynb`.

Известно, что алгоритмы шумопонижения, построенные с применением искусственно внесенного шума, при применении их к реальным зашумленным изображениям, как правило, показывает результат хуже ожидаемых; это связано с тем, что реальный шум отличается от искусственно созданного в значительной степени. Принимая во внимание этот факт, в данной работе делается попытка построения модели реального шума и применение этой модели к обучению сети.

### **2.6.1 Подготовка исходных данных**

Исходные данные для построения модели представляют собой 500 цифровых изображений типа «dark», полученных в рамках выполнения данной работы с применением цифровой камеры Canon EOS R. Съемка производилась со снятым объективом и закрытой крышкой сериями по 100 кадров. Между сериями делался (минимально) часовой перерыв для снижения вероятности влияния нагрева сенсора камеры на картину шума. Температура в помещении,

где производилась съемка находилась в пределах 20-25 градусов Цельсия. Время экспозиции каждого кадра составляло 1/30 секунды. Выбор именно такого значения обусловлен тем, что в реальных условиях заметный шум на снимках возникает, по причине повышения величины ISO при плохом освещении, а величина 1/30 секунды есть максимально рекомендуемое значение при съемке с рук (выдержки длиннее, вероятнее всего, дадут смаз на фото, выдержки короче – еще более усугубят недостаток света; при этом недостаток экспозиции как раз и компенсируется повышением ISO). Съемка каждой серии производилась при отдельно выбранном значении ISO; таким образом итоговый набор изображений содержит пять поднаборов по 100 кадров со значениями ISO: 400, 640, 1000, 1600, 3200. Для исключения вероятности внесения дополнительных шумов и артефактов при преобразовании в jpeg сохранение производилось в формате RAW.

### 2.6.2 Анализ данных

На этапе подготовки данных был собран датасет из 500 RAW-изображений размером 4498x6742. Дальнейший анализ и обработка данных производились в Jupyter Notebook.

Для чтения raw изображений и преобразования их в числовой массив данных была использована библиотека rawru. Ввиду значительного объема сохранение данных было реализовано в виде отображаемых в память структур numpy memmap. Подготовленный таким образом массив данных для удобства обработки и сокращения объема занимаемой памяти далее был разделен на три массива по числу каналов цветного изображения; в итоге были получены три массива с размерностями (15162758000, 1), и вся дальнейшая работа выполнялась уже с поканальными данными.

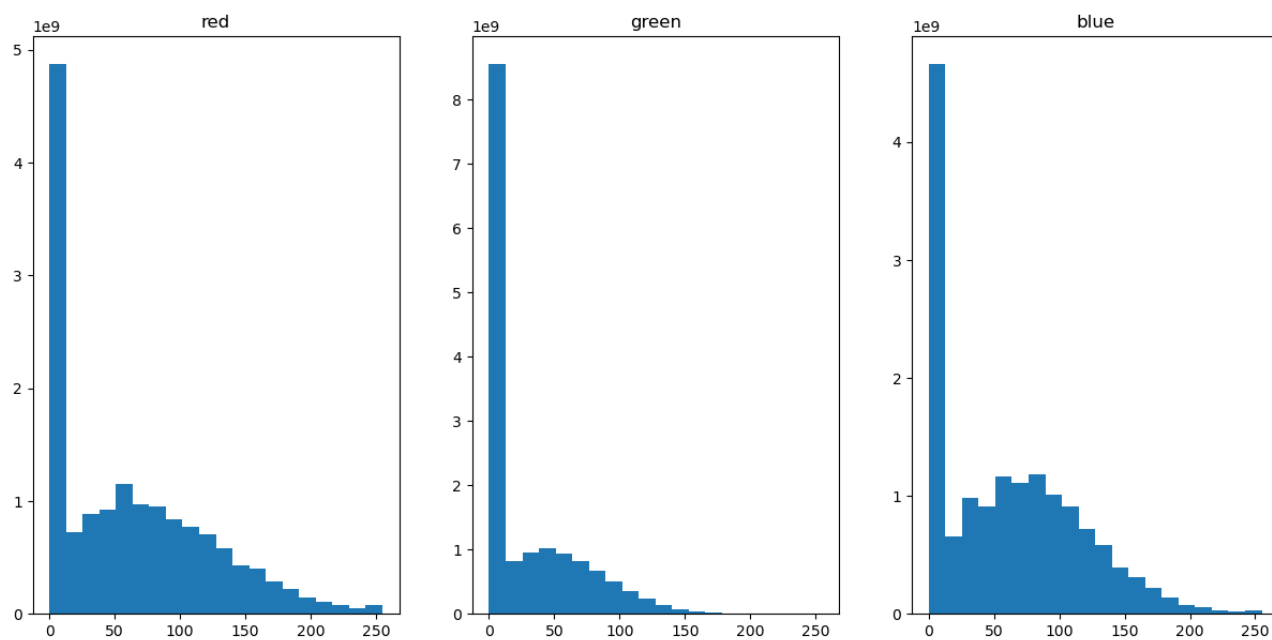


Рисунок 11 - Поканальные гистограммы распределения шума

В целях оценки характера распределения и анализа на наличие выбросов были построены гистограммы (рис. 11) и диаграммы «ящик с усами» (рис. 12).

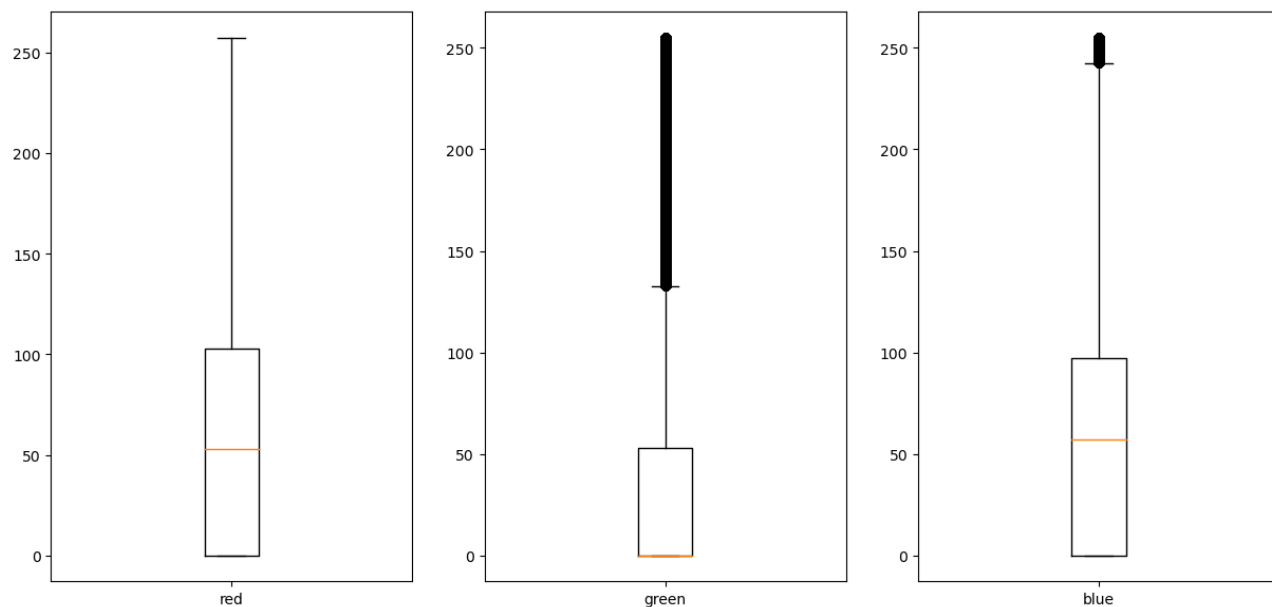


Рисунок 12 - Поканальные boxplot распределения шума

По построенной гистограмме очевидно, что распределение не является нормальным. Анализ диаграммы boxplot показал отсутствие выбросов в данных в нижней границе и наличие отдельных выбросов в области высоких значений

интенсивности; используя правило межквантильных интервалов было произведено удаление выбросов, выходящих за пределы верхнего диапазона  $Q3 + 1.5(Q3 - Q1)$ , где  $Q1, Q3$  – первый и третий квантиль соответственно.

### 2.6.3 Построения модели и генератора

Подготовленные и очищенные от выбросов исходные данные были использованы для построения эмпирической функции распределения. Идея заключалась в следующем: имея набор значений интенсивности шума получить эмпирическую модель распределения и на ее основе построить генератор, который можно будет использовать для генерации шума при формировании обучающей выборки; поскольку в основе модели лежат реальные данные, можно сделать предположение о том, что значения, выдаваемые генератором, будут соответствовать реальному распределению шума.

Для реализации идеи из имеющихся исходных данные необходимо получить частоты встречаемости каждого из значений интенсивности, далее рассчитать вектор значений и вектор вероятностей и подать их на вход метода `scipy.stats.rv_discrete`. Результатом будет созданный объект `rv_discrete`, который, при помощи вызова метода `rvs()` можно использовать в качестве генератора. На рис. 13 приведены полученные в ходе построения модели графики распределения вероятностей по уровням интенсивности шумового сигнала.

В качестве проверки построенным генератором были смоделированы 10.000 образцов и используя их как исходные данные были проделаны все те же шаги, что и при построении генератора. Полученные в результате графики распределения вероятностей интенсивности приведены на рис 14; как можно видеть, полученные модельные результаты практически не отличаются от исходного шума.

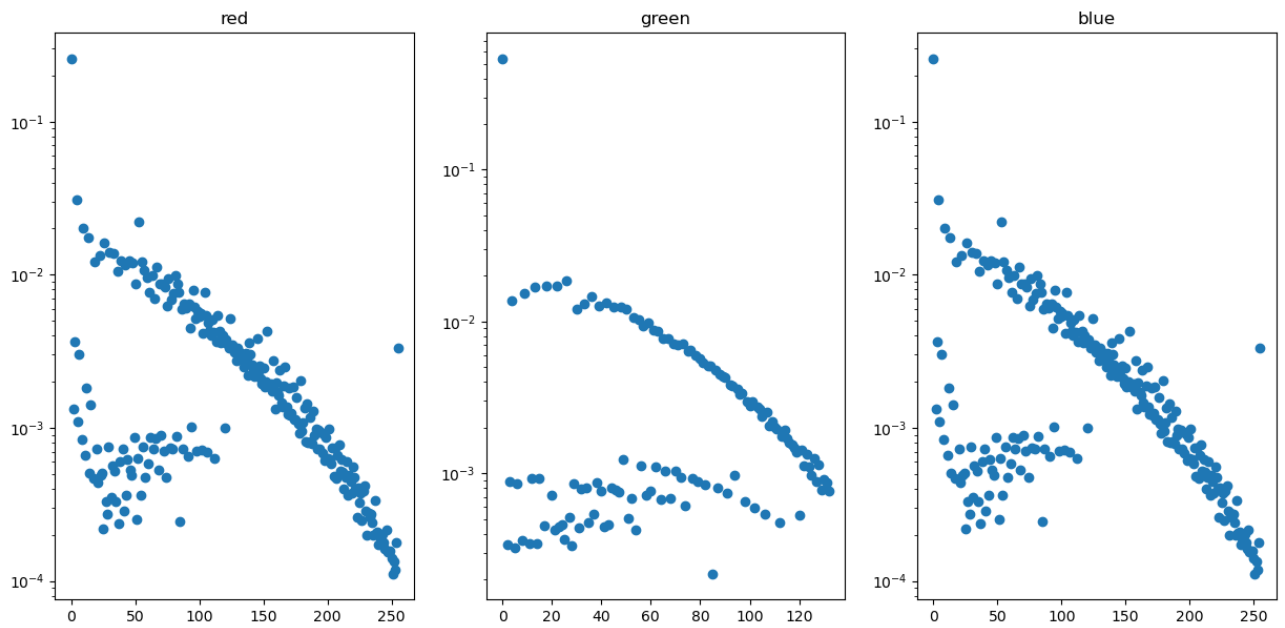


Рисунок 13 - Картина распределения шума - диаграмма  
вероятность/интенсивность

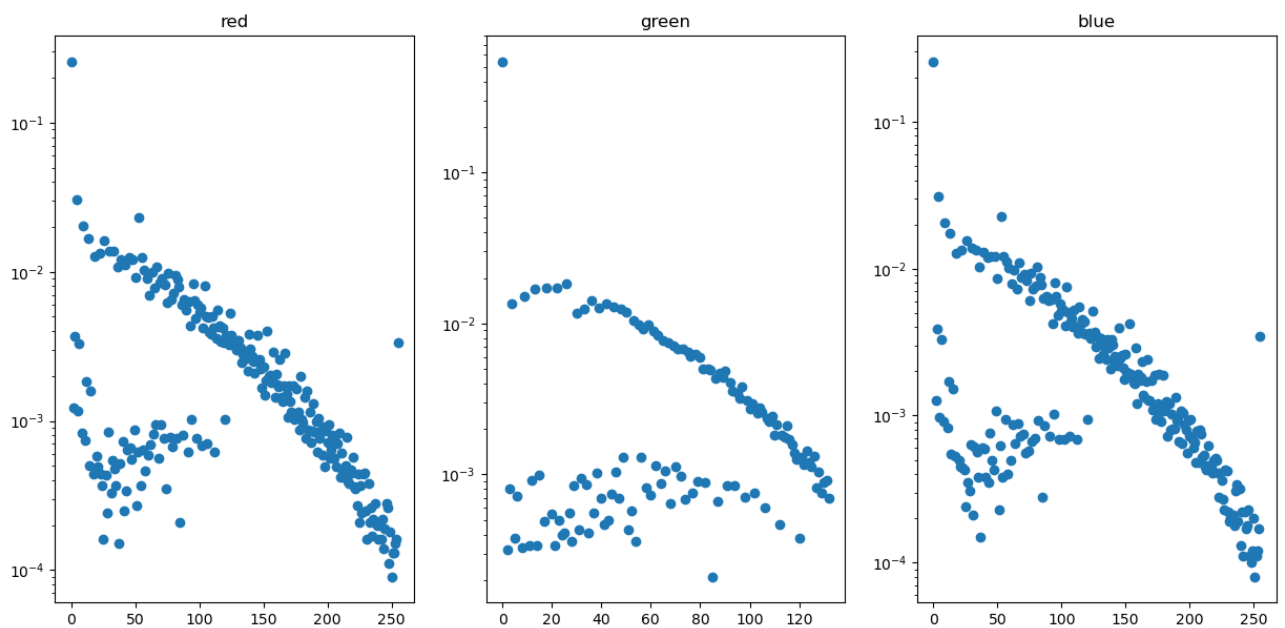


Рисунок 14 - Диаграмма вероятность/интенсивность, построенная для  
смоделированных данных.

Далее, используя построенную модель были сформированы наборы зашумленных изображений с добавочным коэффициентом, равным 0.5. Подбор конкретного значения осуществлялся эмпирически, ориентируясь на



получаемый уровень шумов, близкий к таковому для реальных фото, сделанных в плохих условиях освещения и с высоким ISO (рис. 15).

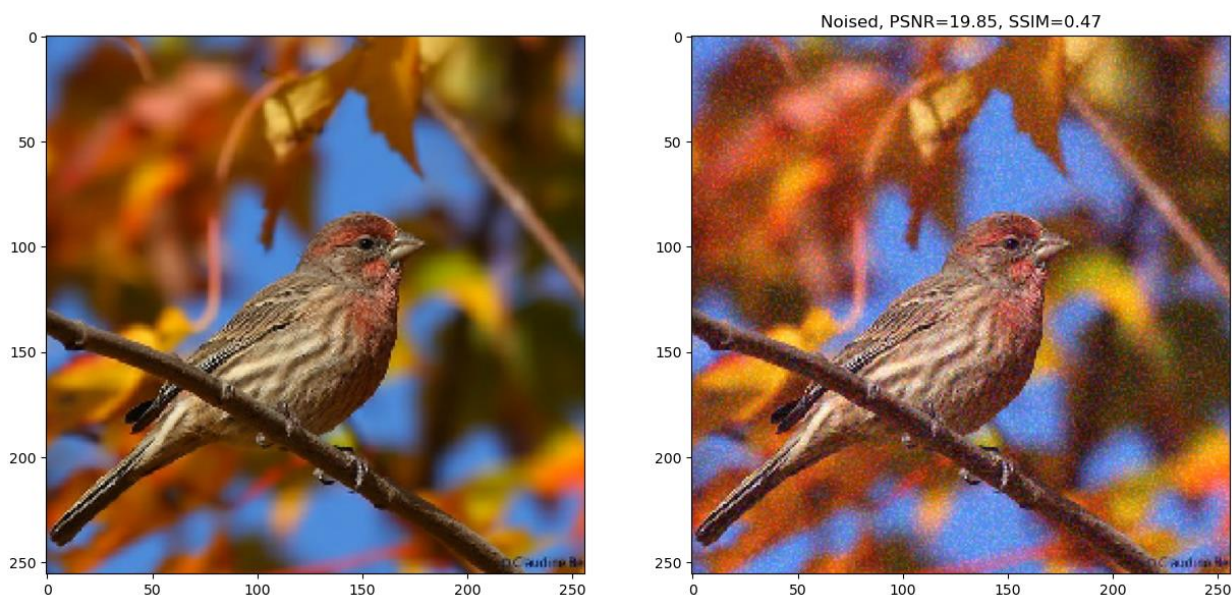


Рисунок 15 - Исходное изображение и изображение, зашумленное с применением построенной модели

#### 2.6.4 Обучение сети на модели реального шума

Исходный код – файл `dncnn_res_final_real_noise050.ipynb`.

Сеть, показавшая наилучший результат по итогам второго этапа работы – модификация `dnCNN` - была обучена на датасете, подготовленном с применением построенной модели реального шума: функция потерь `SSIM_L2`, число эпох – 40, `learning rate` – 0.01, обучающая/тестовая выборки составили 2000 и 500 изображений соответственно, валидационная выборка – 1000. График динамики обучения показан на рис.16 Ошибка на конец обучения составила 0.0830 на обучении и 0.0839 на валидации.

Далее на вход обученной сети подавались обучающая/тестовая и валидационная выборки; показатели `PSNR` и `SSIM` рассчитывались как среднее по всей выборке. Получены следующие значения: на обучении/тесте – `PSNR`



30.78, SSIM 0.917, на валидации – PSNR 30.66, SSIM 0.915. Примеры изображений приведены на рис 17; как видно, построенная сеть прекрасно справилась с задачей – очищенные от шума изображения практически идентичны исходным; об этом же говорит и показатель SSIM, очень близкий к максимально возможному значению, равному 1.0 (значение показателя 1.035 означает полную идентичность двух изображений).

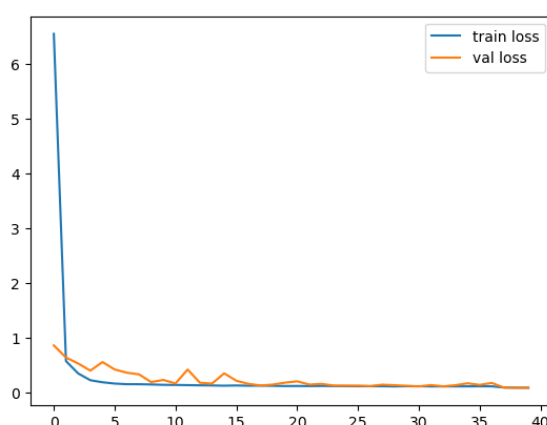


Рисунок 16 - Кривая обучения (train/val)

Финальным этапом валидации построенной сети стала проверка на изображениях, снятым в плохих условиях, с высокими значениями ISO и содержащих реальный шум. Для этого были подготовлены два набора изображений: первый – набор изображений, сделанных той же камерой, шум которой лег в основу построения модели; и второй – изображения, снятые другими камерами.

Первый набор данных был отснят в рамках выполнения данной работы при слабом освещении и со значениями ISO, равными 1600, 3200 и 10000. Сохраненные в RAW изображения поданы на вход сети. Учитывая, что размер изображений превышает размер, на котором обучалась сеть (256x256) предварительно изображения дробились на части требуемого размера и в таком виде подавались на вход сети. Полученные с выхода, соответственно, собирались обратно в цельное изображение.

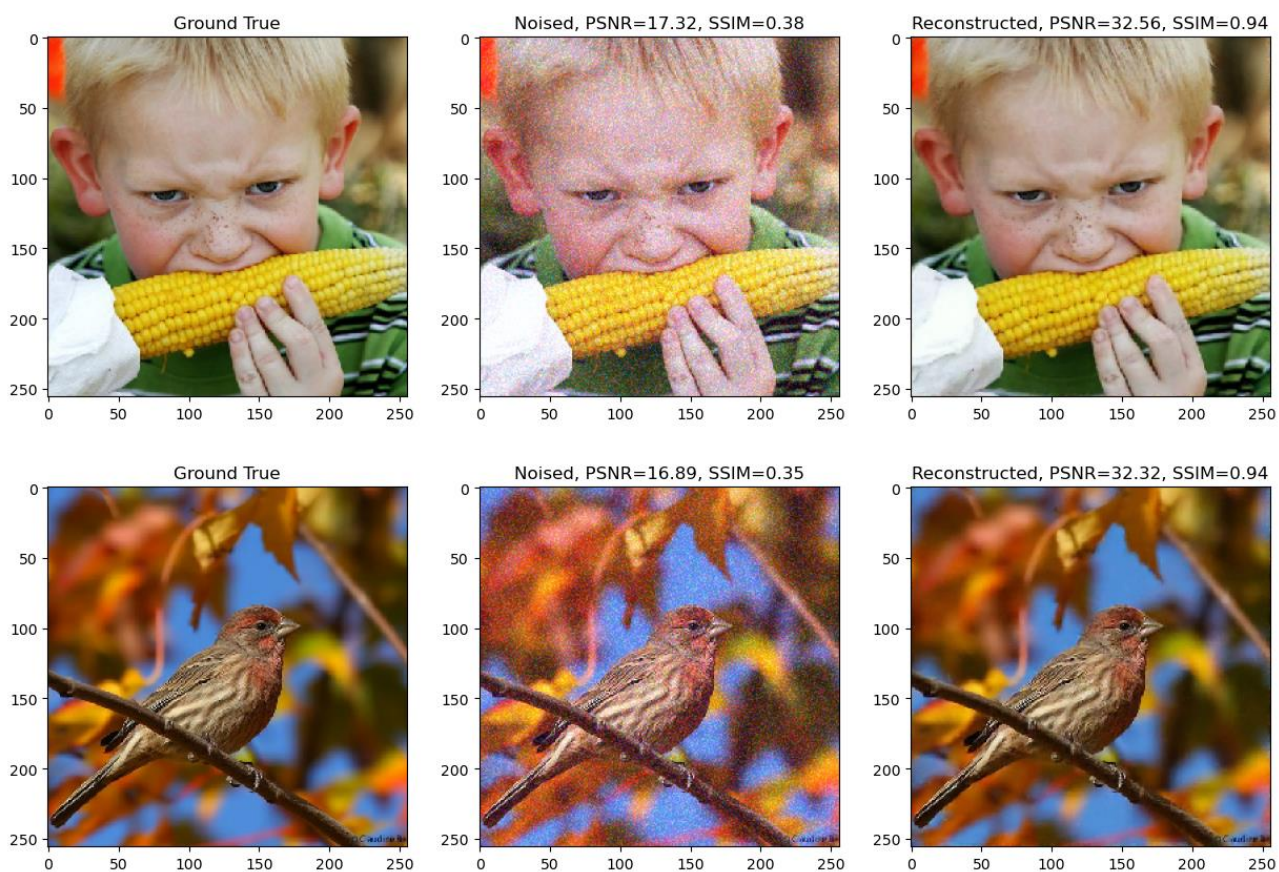


Рисунок 17 - Исходные, зашумленные и восстановленные изображения  
финального этапа (валидация – вверху, обучение/тест – внизу)



Рисунок 18 - Удаление шумов реального изображения (Canon EOS R, ISO 3200)



Второй набор – изображения из датасета Renoir [16], представляющего собой реальные снимки, сделанные в условиях плохого освещения на три модели камер – зеркальная Canon T3i, компакт Canon S90 и смартфон Xiaomi Mi3.

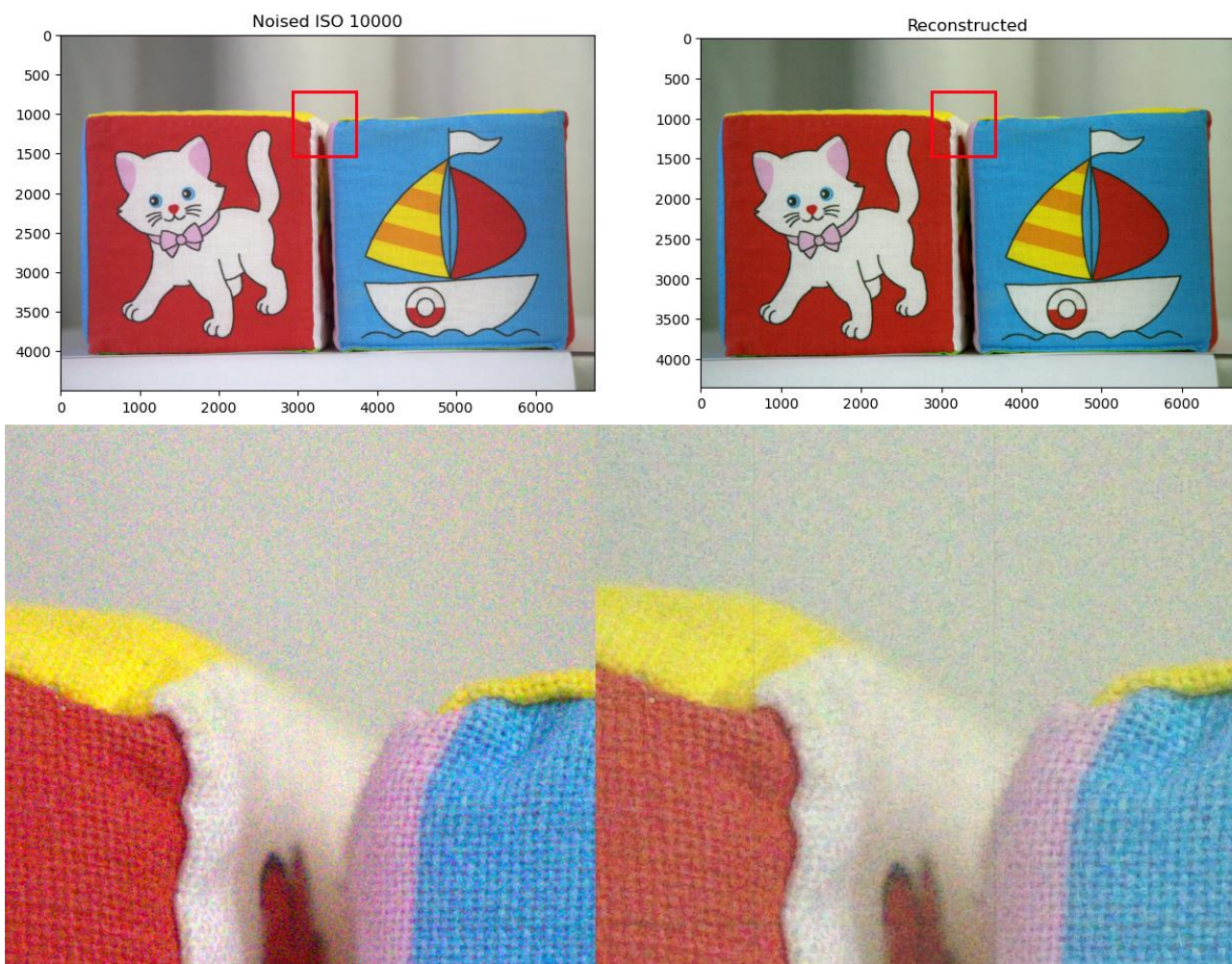


Рисунок 19 - Удаление шумов реального изображения (Canon EOS R,ISO 10000)

Как можно видеть, сеть на изображениях (рис. 18, 19), снятых на ту же камеру, шум с которой являлся основой для построения модели показала заметно лучшие результаты, чем на «чужих» изображениях, что ожидаемо и объяснимо – картина шума есть характеристика сенсора и электронных компонентов обвязки, и каждая его модель (а, тем более, у разных производителей) будет иметь свои особенности. Тем не менее, даже при таких условиях построенная

сеть смогла выполнить заметное на глаз снижение шумов на изображениях, полученных с Xiaomi Mi3, обучаясь на шумовой картине сенсора от Canon (рис. 21).

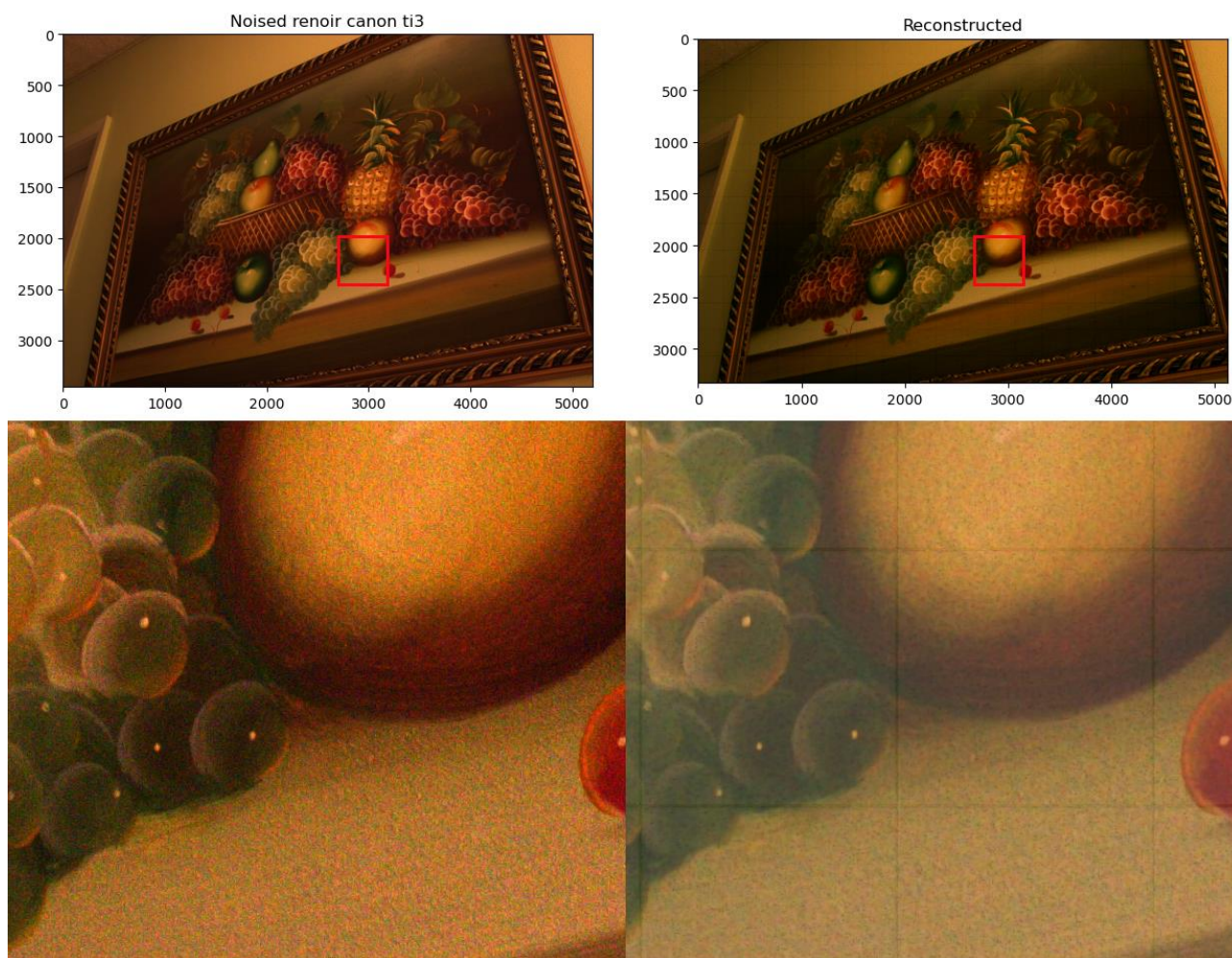


Рисунок 20 - Удаление шумов реального изображения (Canon Ti3)

Численная оценка показателей PSNR и SSIM не осуществлялась ввиду того, что при работе с реальными изображениями у нас нет возможности иметь чистую «эталонную» версию шумного кадра, а наличие референса является неотъемлемым условием алгоритма расчета данных показателей. Даже если предположить, что снимки делаются в лабораторных условиях и есть возможность отснять одну и ту же сцену с различными условиями освещения



полученные «референс» кадры использовать как эталон при расчетах будет нельзя, т.к. между отдельными кадрами всегда будут отличия, в том числе сдвиги, независимо от того, насколько жестко закреплена камера; соответственно, и рассчитанные показатели будут не совсем корректны.



Рисунок 21 - Удаление шумов реального изображения (Xiaomi Mi3)

## Заключение

Данная работа была посвящена применению глубоких нейронных сетей к задачам шумопонижения изображений. В процессе работы были рассмотрены и реализованы несколько архитектур, сравнены результаты, полученные на каждой из них; наилучший результат с PSNR 23.51 и SSIM 0.66 на валидационной выборке показала модифицированная версия dnCNN (оригинальная dnCNN из [6] с добавленными short connection на внутренних conv блоках).

Важной частью работы было построение модели шума, основываясь на реальных данных, собранных в процессе ее выполнения; построенная таким образом модель показала картину шума, практически идентичную реальной, а нейронная сеть, обученная на этой модели показала способность практически полностью восстанавливать исходное изображение без какого-либо заметного снижения детальности – показатель SSIM достиг значения 0.915 на валидации, что является очень высоким значением.

Хорошие результаты также были получены и при применении обученной на построенной модели шума сети к реальным зашумленным изображениям, сделанным различными моделями цифровых камер.

## Библиографический список

- 1 D. Hendrycks, K. Gimpel, Gaussian Error Linear Units (GELUs) 2016
- 2 H. Zhao, O. Gallo, I. Frosio and J. Kautz, "Loss Functions for Image Restoration With Neural Networks," in IEEE Transactions on Computational Imaging, vol. 3, no. 1, pp. 47-57, March 2017, doi: 10.1109/TCI.2016.2644865.
- 3 Hendrycks, Dan, Gimpel, Kevin. Gaussian Error Linear Units (GELUs). 2016. eprint arXiv:1606.08415
- 4 Hu, Xiaodan, et al. "RUNet: A Robust UNet Architecture for Image Super-Resolution." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2019
- 5 Javier Gurrola-Ramos, Oscar Dalmau and Teresa E. Alarcón. A Residual Dense U-Net Neural Network for Image Denoising, IEEE Access, vol. 9, pp. 31742-31754, 2021
- 6 Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, Lei Zhang. - Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising
- 7 K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in IEEE International Conference on Computer Vision, 2015, pp. 1026–1034.
- 8 Kai Zhang, Wangmeng Zuo, Senior Member, IEEE, and Lei Zhang, Fellow, IEEE. FFDNet: Toward a Fast and Flexible Solution for CNN based Image Denoising. 2018. arXiv:1710.04026v2
- 9 Komatsu, R.; Gonsalves, T. Comparing U-Net Based Models for Denoising Color Images. AI 2020, 1, 465-486. <https://doi.org/10.3390/ai1040029>
- 10 Nicolas Anderson, Mikael Antero Paavola, Johnny Sognnes. On the use of Denoising Autoencoders and Deep Convolutional Adversarial Networks for Automated Removal of Date Stamps, University of Agder, 2019

- 11 O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. - Springer International Publishing, Cham, 2015. pp. 234-241
- 12 Paavani Dua. Image Denoising Using a U-net. Stanford University, 2019
- 13 Rina Komatsu and Tad Gonsalves. Comparing U-Net Based Models for Denoising Color Images. AI. 2020
- 14 Xavier Glorot and Bengio, Y. Understanding the Difficulty of Training Deep Feedforward Neural Networks. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, 9, 249-256, 2010.  
<http://proceedings.mlr.press/v9/glorot10a.html>
- 15 Imagenet dataset. <https://www.kaggle.com/datasets/lijiyu/imagenet>
- 16 RENOIR - A dataset of digital camera images corrupted by real low-light noise. <https://adrianbarburesearch.blogspot.com/p/image-denoising.html>