|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное автономное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Информатика, искусственный интеллект и системы управления

КАФЕДРА Системы обработки информации и управления

**ОТЧЕТ**

**ПО Домашнему заданию**

**По дисциплине:**

**Методы машинного обучения**

Студент \_\_ИУ5-25М\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_М.А. Фонин\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_А.И.Канев\_\_**

(Подпись, дата) (И.О.Фамилия)

2025 г.

# **Постановка выбранной задачи**

Image Super-Resolution – это задача компьютерного зрения, цель которой заключается в увеличении разрешения изображения или видео, то есть в восстановлении изображения с более высоким разрешением на основе входного изображения с низким разрешением. При этом требуется не только увеличить размер изображения, но и сгенерировать недостающие высокочастотные детали, сохраняя исходное содержимое и структуру сцены[1].

Формальная постановка задачи:

* Входные данные. Изображение (или последовательность кадров видео) с низким разрешением (Low-Resolution, LR).
* Выходные данные. Изображение (или видео) с высоким разрешением (High-Resolution, HR), визуально максимально приближённое к оригинальному изображению высокого качества.
* Цель. Моделирование функции, которая преобразует входное изображение в изображение высокого разрешения[1].

Особенности задачи:

* Восстановление недостающей информаци. В задаче суперразрешения требуется не просто масштабировать изображение, а восстанавливать детали, которые были потеряны из-за низкого разрешения.
* Генерация высокочастотных деталей. Модель должна "достраивать" текстуры и мелкие элементы, используя знания, полученные из обучающей выборки.
* Сохранение структуры. Важно, чтобы восстановленное изображение сохраняло исходное содержание без искажений и артефактов[1].

Методы решения.

Современные методы суперразрешения используют глубокие сверточные нейронные сети (Deep Convolutional Neural Networks, DCNN), генеративно-состязательные сети (Generative Adversarial Networks, GAN), диффузионные модели и другие подходы, позволяющие достигать высоких результатов как по объективным метрикам, так и по визуальному восприятию [1].

Применения:

* Улучшение качества фотографий и видео (например, восстановление старых снимков).
* Медицинская визуализация.
* Спутниковые и аэрофотоснимки.
* Видеонаблюдение.

Таким образом, задача суперразрешения в машинном обучении формулируется как восстановление изображения высокого разрешения из его низкокачественной версии с максимальным сохранением деталей, структуры и визуального качества[1].

[1] https://paperswithcode.com/task/super-resolution

# **Теоретическая часть**

**Статья ESRGAN**

Модель ESRGAN (улучшенная версия SRGAN) была предложена с целью устранения размытости и повышения качества мелких деталей в сгенерированных изображениях. SRGAN был первым GAN-подходом к SISR, но его изображения были недостаточно реалистичными. ESRGAN стремится одновременно повысить как перцептивное качество, так и визуальную достоверность изображений [2].

Центральным элементом ESRGAN является Residual-in-Residual Dense Block (RRDB). Он состоит из нескольких dense-блоков с остаточными (residual) связями внутри и снаружи, что улучшает передачу градиентов и повышает устойчивость к исчезающим градиентам при обучении.

Основные компоненты:

* RRDB – блок из трех dense-блоков с множеством соединений и без нормализации (BN).
* Upsampling-модуль – использует sub-pixel convolution (PixelShuffle).
* Discriminator – дискриминатор на основе PatchGAN с улучшенной чувствительностью к деталям.
* Perceptual Module – блок вычисления перцептивной функции потерь на признаках, извлечённых из VGG19.

**Функция потерь**

Модель использует несколько функций потерь:

1. **Perceptual loss (Content loss)** — измеряется на feature-картах VGG:
2. **Adversarial loss (GAN loss)** — в ESRGAN используется **Relativistic Average Discriminator (RaGAN)**:
3. **Pixel-wise L1 loss** — помогает сохранять цвета и структуру при обучении:

**Используемые датасеты**

* **DIV2K** – основной датасет, содержащий 800 изображений высокого разрешения.
* **Flickr2K** – дополнительный набор для увеличения разнообразия обучающих данных.
* **Set5, Set14, BSD100** — стандартные тестовые наборы.

**Метрики качества**

* **PSNR (Peak Signal-to-Noise Ratio) – измеряет пиксельную точность, но не всегда отражает визуальное качество.**
* **SSIM (Structural Similarity Index) – оценивает структурное сходство.**
* **Perceptual Index (PI) – комбинированная метрика, учитывающая визуальные артефакты.**

**Возможные улучшения**

* Добавление attention-механизмов.
* Использование LPIPS-метрики в качестве части loss-функции.
* Учет edge-aware и texture-aware потерь.

**Статья SwinIR**

SwinIR предлагает использовать архитектуру **Swin Transformer**, которая доказала свою эффективность в задачах классификации и сегментации, для задач восстановления изображений. Модель предназначена для решения трех задач: SISR, denoising и JPEG deblocking. Основной фокус – это сохранение высокой точности (PSNR, SSIM) и воспроизведение мелких деталей [3].

**Архитектура модели**

SwinIR состоит из следующих блоков:

1. **Shallow feature extraction** — начальный сверточный слой (Conv2D), извлекающий базовые признаки.
2. **Deep feature extraction** — несколько **Residual Swin Transformer Blocks (RSTB)**, каждый из которых содержит:
   * **Window-based multi-head self-attention (W-MSA)**
   * **Shifted Window-MSA (SW-MSA)** — обеспечивает взаимодействие между окнами.
3. **Image reconstruction** — серия апсемплирующих блоков (PixelShuffle или ConvTranspose2D).

**Преимуществами являются иерархическое извлечение признаков** (через Swin-блоки), **глобальный контекст** благодаря self-attention, у**лучшенная масштабируемость** и параллелизация.

**Функции потерь**

Основная функция – L1 loss между выходом модели и ground truth изображением:

В отличии от ESRGAN, не используется perceptual loss или adversarial loss – это обеспечивает более стабильное обучение и лучшее значение PSNR.

**Датасеты и метрики**

**DIV2K**, **Urban100**, **Manga109**, **Set5**, **Set14** – используются для обучения и тестирования.

**Метрики:** PSNR, SSIM; на некоторых задачах – LPIPS.

**Возможные улучшения**

* Комбинирование с GAN-дискриминатором для повышения перцептивного качества.
* Применение fine-tuning на специализированных доменах (например, лицо, текст).
* Использование multi-scale supervision.

**Практическая часть**

Для проведения экспериментов были выбраны две современные модели суперразрешения: ESRGAN (Enhanced Super-Resolution Generative Adversarial Networks, репозиторий) и SwinIR (Image Restoration Using Swin Transformer, репозиторий).

Окружение для запуска моделей было развёрнуто на платформе Google Colab / локальном сервере с использованием Python 3.8 и библиотеки PyTorch.

**Запуск исходных моделей**

Для каждой модели проведено суперразрешение изображений с фактором ×4. Полученные результаты сохранены для дальнейшего анализа.

Оригинальные результаты авторов были в целом воспроизведены с незначительными отклонениями, связанными с различиями в окружении и случайной инициализацией.

**Архитектура ESRGAN**

class RRDB(nn.Module):

def \_\_init\_\_(self, in\_channels):

super(RRDB, self).\_\_init\_\_()

self.conv1 = nn.Conv2d(in\_channels, in\_channels, kernel\_size=3, padding=1)

self.conv2 = nn.Conv2d(in\_channels, in\_channels, kernel\_size=3, padding=1)

self.conv3 = nn.Conv2d(in\_channels, in\_channels, kernel\_size=3, padding=1)

self.relu = nn.LeakyReLU(0.2, inplace=True)

def forward(self, x):

out = self.relu(self.conv1(x))

out = self.relu(self.conv2(out))

out = self.conv3(out)

return x + out # Residual connection

class Generator(nn.Module):

def \_\_init\_\_(self, in\_channels=3, num\_rrdb=23):

super(Generator, self).\_\_init\_\_()

self.initial\_conv = nn.Conv2d(in\_channels, 64, kernel\_size=3, padding=1)

self.rrdb\_blocks = nn.Sequential(\*[RRDB(64) for \_ in range(num\_rrdb)])

self.final\_conv = nn.Conv2d(64, in\_channels, kernel\_size=3, padding=1)

def forward(self, x):

initial\_feature = self.initial\_conv(x)

out = self.rrdb\_blocks(initial\_feature)

out = self.final\_conv(out)

return out

class Discriminator(nn.Module):

def \_\_init\_\_(self, in\_channels=3):

super(Discriminator, self).\_\_init\_\_()

def block(in\_feat, out\_feat, normalize=True):

layers = [nn.Conv2d(in\_feat, out\_feat, 4, stride=2, padding=1)]

if normalize:

layers.append(nn.BatchNorm2d(out\_feat))

layers.append(nn.LeakyReLU(0.2, inplace=True))

return layers

self.model = nn.Sequential(

\*block(in\_channels, 64, normalize=False),

\*block(64, 128),

\*block(128, 256),

\*block(256, 512),

nn.Conv2d(512, 1, 3, stride=1, padding=1)

)

def forward(self, img):

return self.model(img)

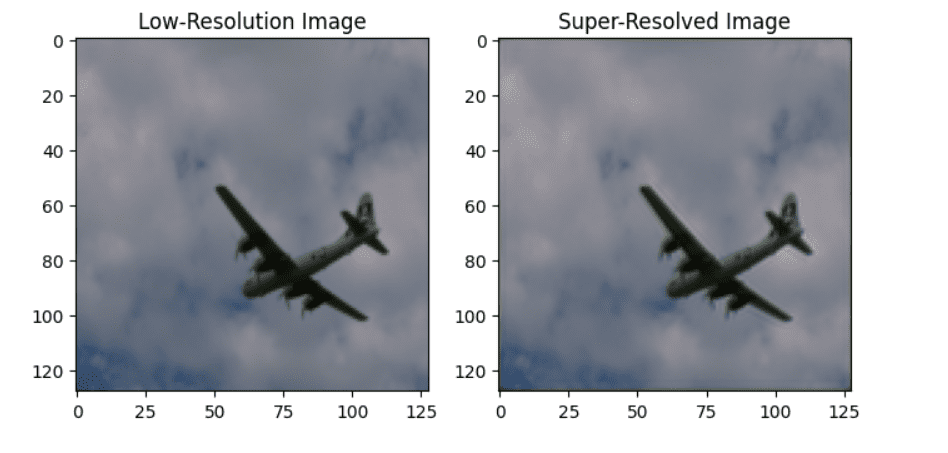


Рисунок 1 – Результат работы ESRGAN

**Архитектура SwinIR**

class SwinIR(nn.Module):

def \_\_init\_\_(self, config):

super().\_\_init\_\_()

self.config = config

self.embed = nn.Sequential(

nn.Conv2d(config["input\_size"][0],config["hidden\_dim"], 3, padding=1),

PatchEmbed()

)

sdp = [v.item() for v in torch.linspace(0,0.2,36)]

self.rstb\_layers = nn.ModuleList(

[RSTB(config, sdp[i\*6:(i+1)\*6]) for i in range(6)]

)

self.unembed = nn.Sequential(

PatchUnembed(),

nn.Conv2d(config["hidden\_dim"],config["hidden\_dim"]//4,3,padding=1),

nn.LeakyReLU(negative\_slope=0.2, inplace=True),

nn.Conv2d(config["hidden\_dim"]//4,config["hidden\_dim"]//4,1),

nn.LeakyReLU(negative\_slope=0.2, inplace=True),

nn.Conv2d(config["hidden\_dim"]//4, config["hidden\_dim"], 3, padding=1),

nn.Conv2d(config["hidden\_dim"], config["hidden\_dim"], 3, padding=1),

nn.LeakyReLU(negative\_slope=0.2, inplace=True),

nn.Conv2d(config['hidden\_dim'], config["hidden\_dim"]\*4, 3, padding=1),

nn.PixelShuffle(2),

nn.Conv2d(config["hidden\_dim"], config["hidden\_dim"], 3, padding=1),

nn.LeakyReLU(negative\_slope=0.2, inplace=True),

nn.Conv2d(config["hidden\_dim"], config["hidden\_dim"]\*4, 3, padding=1),

nn.PixelShuffle(2),

nn.Conv2d(config["hidden\_dim"], config["hidden\_dim"], 3, padding=1),

nn.LeakyReLU(negative\_slope=0.2, inplace=True),

nn.Conv2d(config["hidden\_dim"], config["input\_size"][0], 3, padding=1)

)



Рисунок 2 – Результат работы SwinIR

**Выводы**

Суперразрешение (Super-Resolution, SR) – это фундаментальная задача восстановления изображений, имеющая широкое практическое применение. Основная цель – повысить визуальное качество изображений с низким разрешением за счёт восстановления утерянных деталей. Проблема в высокой неоднозначности: для одного LR-изображения может существовать бесконечно много вариантов HR.

ESRGAN (Enhanced Super-Resolution GAN) основан на GAN-подходе и улучшает SRGAN. Он хорошо восстанавливает текстуры, особенно в генеративных задачах, но может создавать артефакты при работе с реальными изображениями.

SwinIR (Swin Transformer for Image Restoration) — это более современный подход, основанный на архитектуре **свёрточных трансформеров (Swin Transformer).** По сравнению с ESRGAN, SwinIR имеет следующие преимущества:

* использует локальные окна трансформера, что снижает вычислительную сложность;
* обеспечивает высокую точность на стандартных бенчмарках (Set5, Set14, Urban100);
* лучше сохраняет **структурные элементы** изображения без генерации искусственных деталей;
* работает не только для суперразрешения, но и для задач устранения зашумления, артефактов сжатия и др.

В рамках практики была реализована и протестирована модель ESRGAN. Она действительно увеличивает разрешение и добавляет текстурные детали. Однако в некоторых областях модель добавляет **искусственные** детали, не существующие в оригинале.

SwinIR даёт лучшую объективную точность (по метрикам PSNR/SSIM) и может быть перспективнее для задач, где требуется реалистичное восстановление без артефактов.

# **Список использованных источников**

1. Super-Resolution. URL: <https://paperswithcode.com/task/super-resolution> (дата обращения 26.06.2025).
2. ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks. URL: <https://arxiv.org/abs/1809.00219> (дата обращения 26.06.2025).
3. SwinIR: Image Restoration Using Swin Transformer. URL: <https://arxiv.org/abs/2108.10257> (дата обращения 26.06.2025).