ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks

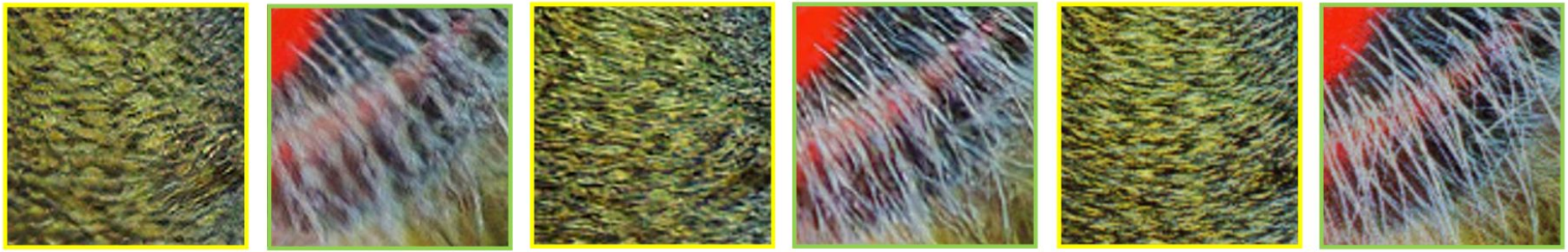
Валентина Герасимова Б21-221

Введение.

Цель: Повышение разрешения изображения (SISR) с фокусом на визуальное качество.

Проблема старых методов (например, SRGAN): артефакты, размытые или искусственные текстуры.

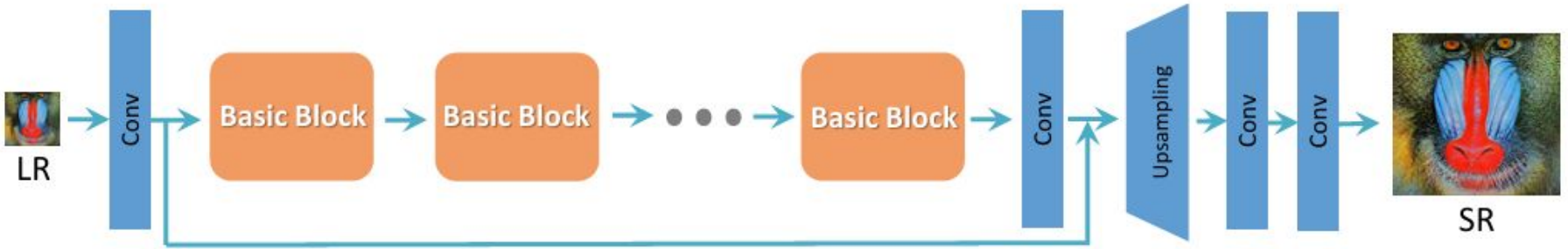
Решение: ESRGAN — улучшенная версия SRGAN с новым блоком, улучшенной функцией потерь и новым дискриминатором.



Слайд 2: Архитектура Генератора

Основной блок: Residual-in-Residual Dense Block (RRDB)

- Удаление Batch Normalization
- Использование Residual Scaling (умножение остаточного сигнала на β)
- Многоуровневая остаточная связь и плотные соединения (Dense Connections)



RRDB

RRDB — это сложный строительный блок, который позволяет нейросети не теряя информацию. Он объединяет две идеи:

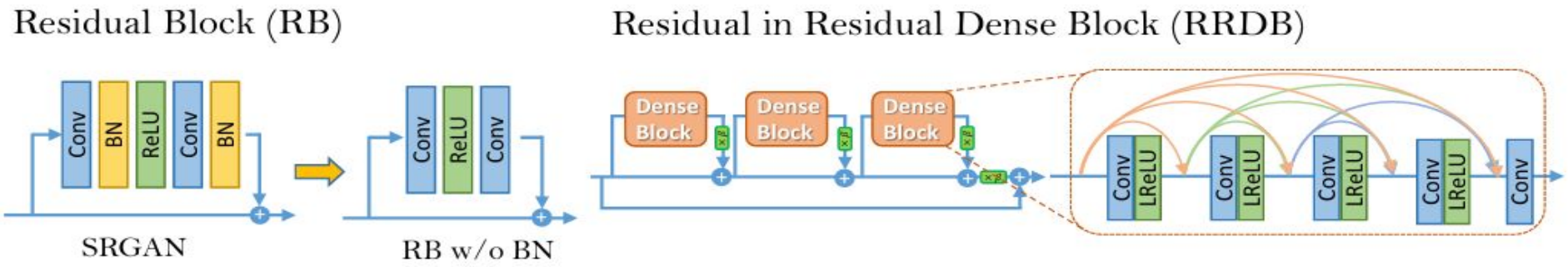
- Остаточные (residual) связи
- Плотные (dense) связи

Остаточная связь означает: "Сохрани вход и добавь его к выходу": Выход = f(вход) + вход

- Помогает избежать "затухания градиента" — сеть не теряет информацию на глубине.
- Делает обучение стабильнее и быстрее.

Плотные связи (Dense Connections): В Dense-блоке — каждый слой передает свои признаки всем последующим.

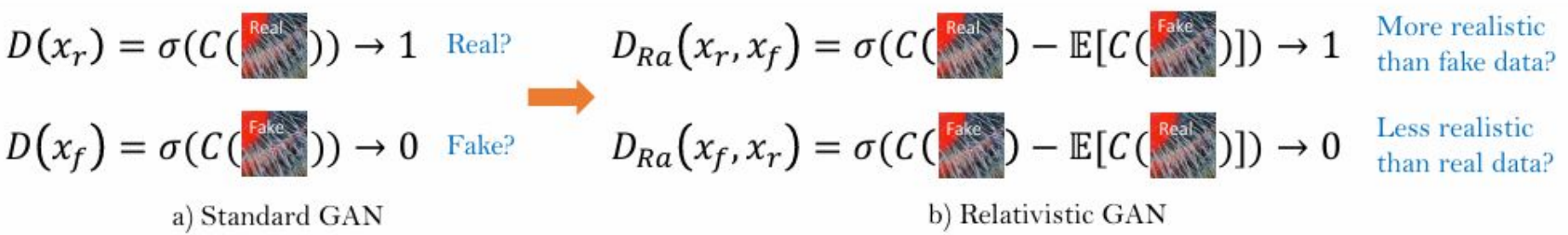
- Улучшается повторное использование признаков
- Сеть становится более "осведомлённой", особенно для сложных текстур



Архитектура Дискриминатора – RaGAN

Обычный дискриминатор (SRGAN): определяет, фейк или реально.

Relativistic Average GAN (RaGAN): определяет, насколько изображение реалистичнее другого.



Запуск модели

Сначала скачиваем модель с гитхаба. Затем запускаем скрипт, прикрепленный там же

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import requests
import time
```

```
!git clone https://github.com/xinntao/ESRGAN
!cd ESRGAN
```

Показать скрытые выходные данные

```
import os

import os.path as osp
import glob
import cv2
import numpy as np
import torch
import ESRGAN.RRDBNet_arch as arch
```

```
model_path = 'ESRGAN/models/RRDB_ESRGAN_x4.pth'  # models/RRDB_ESRGAN_x4.pth OR models/RRDB_PSNR_x4.pth
device = torch.device('cuda')  # if you want to run on CPU, change 'cuda' -> cpu
#device = torch.device('cpu')

test_img_folder = 'ESRGAN/LR/*'

model = arch.RRDBNet(3, 3, 64, 23, gc=32)
model.load_state_dict(torch.load(model_path), strict=True)
model.eval()
model = model.to(device)

print('Model path {:s}. \nTesting...'.format(model_path))

idx = 0
for path in glob.glob(test_img_folder):
    idx += 1
    base = osp.splitext(osp.basename(path))[0]
    print(idx, base)
    # read images
    img = cv2.imread(path, cv2.IMREAD_COLOR)
    img = img * 1.0 / 255
    img = torch.from_numpy(np.transpose(img[:, :, [2, 1, 0]], (2, 0, 1))).float()
    img_LR = img.unsqueeze(0)
    img_LR = img_LR.to(device)

    with torch.no_grad():
        output = model(img_LR).data.squeeze().float().cpu().clamp_(0, 1).numpy()
    output = np.transpose(output[[2, 1, 0], :, :], (1, 2, 0))
    output = (output * 255.0).round()
    cv2.imwrite('ESRGAN/results/{:s}_rlt.png'.format(base), output)
```

Результаты. Запуск модели на кошке из интернета

Оригинальное изображение 255 на 255. Выходное 900 на 900

Запускаем для видео. Сначала извлекаем кадры

```
# === ИЗВЛЕЧЕНИЕ КАДРОВ ИЗ ВИДЕО ===
cap = cv2.VideoCapture(input_video_path)
fps = cap.get(cv2.CAP_PROP_FPS)
w = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
h = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
frame_paths = []

frame_idx = 0
print("Извлечение кадров...")
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break
    frame_path = os.path.join(temp_frames_dir, f"frame_{frame_idx:04d}.png")
    cv2.imwrite(frame_path, frame)
    frame_paths.append(frame_path)
    frame_idx += 1
cap.release()
```

Апскейлим все кадры с помощью модели (как с отдельными картинками)

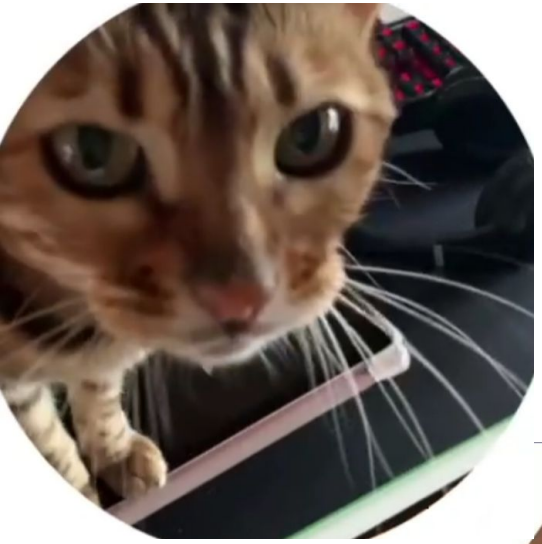
```
# === АПСКЕЙЛ КАДРОВ С ПОМОЩЬЮ ESRGAN ===
print("Апскейл кадров...")
for path in tqdm(frame_paths):
    base = os.path.splitext(os.path.basename(path))[0]
    img = cv2.imread(path, cv2.IMREAD_COLOR)
    img = img * 1.0 / 255
    img = torch.from_numpy(np.transpose(img[:, :, [2, 1, 0]], (2, 0, 1))).float()
    img_LR = img.unsqueeze(0).to(device)

    with torch.no_grad():
        output = model(img_LR).data.squeeze().float().cpu().clamp_(0, 1).numpy()
    output = np.transpose(output[[2, 1, 0], :, :], (1, 2, 0))
    output = (output * 255.0).round().astype(np.uint8)
    cv2.imwrite(os.path.join(results_dir, f"{base}_upscaled.png"), output)
```

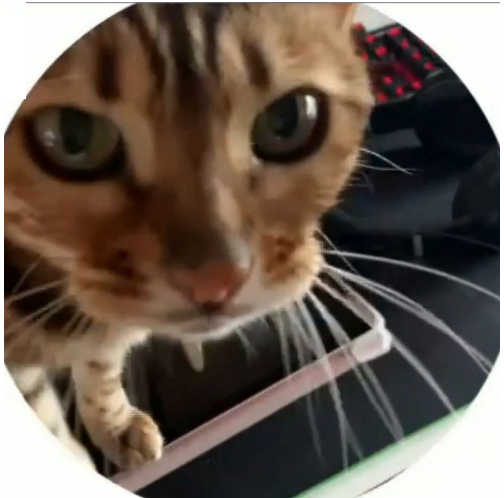
Собираем видео обратно

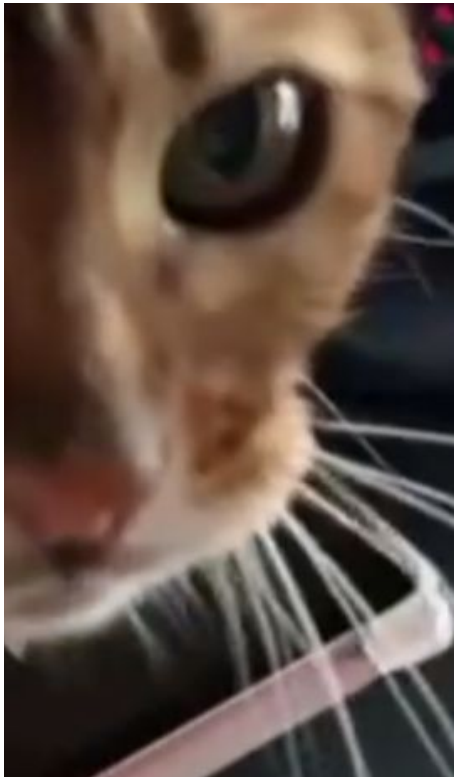
```
# === СОБИРАНИЕ ВИДЕО ОБРАТНО ===
upscaled_frames = sorted(glob.glob(os.path.join(results_dir, '*_upscaled.png')))
sample_frame = cv2.imread(upscaled_frames[0])
height, width, _ = sample_frame.shape

fourcc = cv2.VideoWriter_fourcc(*'mp4v')
out = cv2.VideoWriter(output_video_path, fourcc, fps, (width, height))

print("Сборка видео...")
for frame_path in upscaled_frames:
    frame = cv2.imread(frame_path)
    out.write(frame)
out.release()
```

Результаты для

```
Апскейл кадров...
100%|██████████| 111/111 [3:19:36<00:00, 107.90s/it]
Сборка видео...
✅ Готово! Апскейленное видео сохранено как: cat_murr_and_meow_upscaled.mp4
```







Video