

Материалы кафедры ММП факультета ВМК МГУ. Практикум по программированию на языке Python

Семинар 4: openCV и обработка изображений

Денисов Егор (mail:
denisov.official72@gmail.com), Оганов
Александр (tg: @welmud)

Москва, 2025

Ноутбук в основном основан на [ноутбуке 2023 года](#).

часть материалов взята из [ноутбука Ярослава Морозова](#)

```
In [1]: from PIL import Image
import cv2 as cv

import numpy as np
import matplotlib.pyplot as plt
import skimage.io
import skimage.color
import skimage.transform
import scipy.ndimage
import PIL
%matplotlib inline
```

```
In [2]: !wget http://pm1.narvii.com/7354/68e873fd4a92810b38ae2dc639115f4842f80400r1-
!mv 68e873fd4a92810b38ae2dc639115f4842f80400r1-1200-900v2_uhq.jpg cat.jpg
```

```
--2025-09-26 17:41:13-- http://pm1.narvii.com/7354/68e873fd4a92810b38ae2dc63  
9115f4842f80400r1-1200-900v2_uhq.jpg  
Распознаётся pm1.narvii.com (pm1.narvii.com)... 199.232.172.193  
Подключение к pm1.narvii.com (pm1.narvii.com)|199.232.172.193|:80... соединен  
ие установлено.  
HTTP-запрос отправлен. Ожидание ответа... 200 OK  
Длина: 120709 (118K) [image/jpeg]  
Сохранение в: '68e873fd4a92810b38ae2dc639115f4842f80400r1-1200-900v2_uhq.jpg'  
  
68e873fd4a92810b38a 100%[=====] 117,88K ---KB/s за 0,07s
```

```
2025-09-26 17:41:13 (1,65 MB/s) - '68e873fd4a92810b38ae2dc639115f4842f80400r1  
-1200-900v2_uhq.jpg' сохранён [120709/120709]
```

Загрузка изображений

```
In [3]: cat_image = PIL.Image.open('cat.jpg')  
cat_image
```

```
Out[3]:
```



```
In [4]: cat_image.format, cat_image.size, cat_image.mode
```

```
Out[4]: ('JPEG', (1024, 768), 'RGB')
```

```
In [5]: cat_array = np.array(cat_image)  
plt.imshow(cat_array)
```

```
Out[5]: <matplotlib.image.AxesImage at 0x74a1cf39db0>
```



Также можно сразу получить изображение в виде тензора используя библиотеки matplotlib или skimage

```
In [6]: skimage_array = skimage.io.imread('cat.jpg')
np.testing.assert_almost_equal(cat_array, skimage_array)

plt_array = plt.imread('cat.jpg')
np.testing.assert_almost_equal(cat_array, plt_array)
```

```
In [7]: cat_array.shape # HxWxC
```

```
Out[7]: (768, 1024, 3)
```

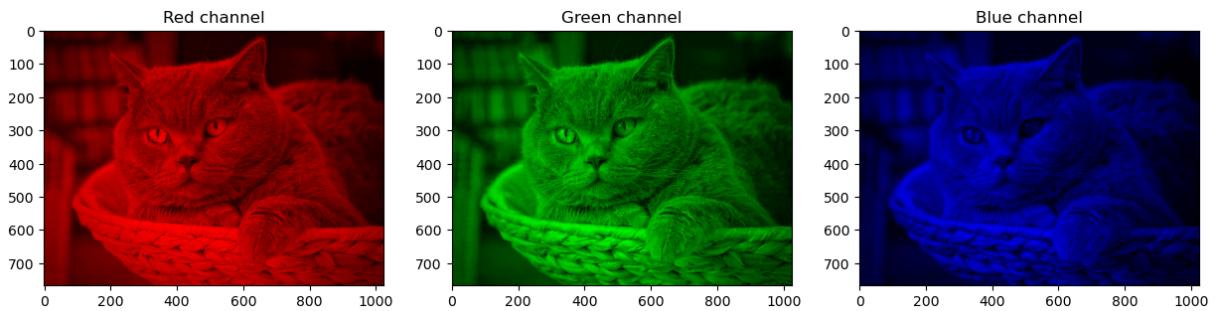
Каждое цветное изображение - трехканальный тензор

Трансформации изображений

```
In [8]: plt.figure(figsize=(15, 10))

titles = ['Red channel', 'Green channel', 'Blue channel']
for i in range(3):
    plt.subplot(1, 3, i + 1)
    zeros = np.zeros_like(cat_array)
    zeros[:, :, i] = cat_array[:, :, i]
    plt.imshow(zeros)
```

```
plt.title(titles[i])  
plt.show()
```

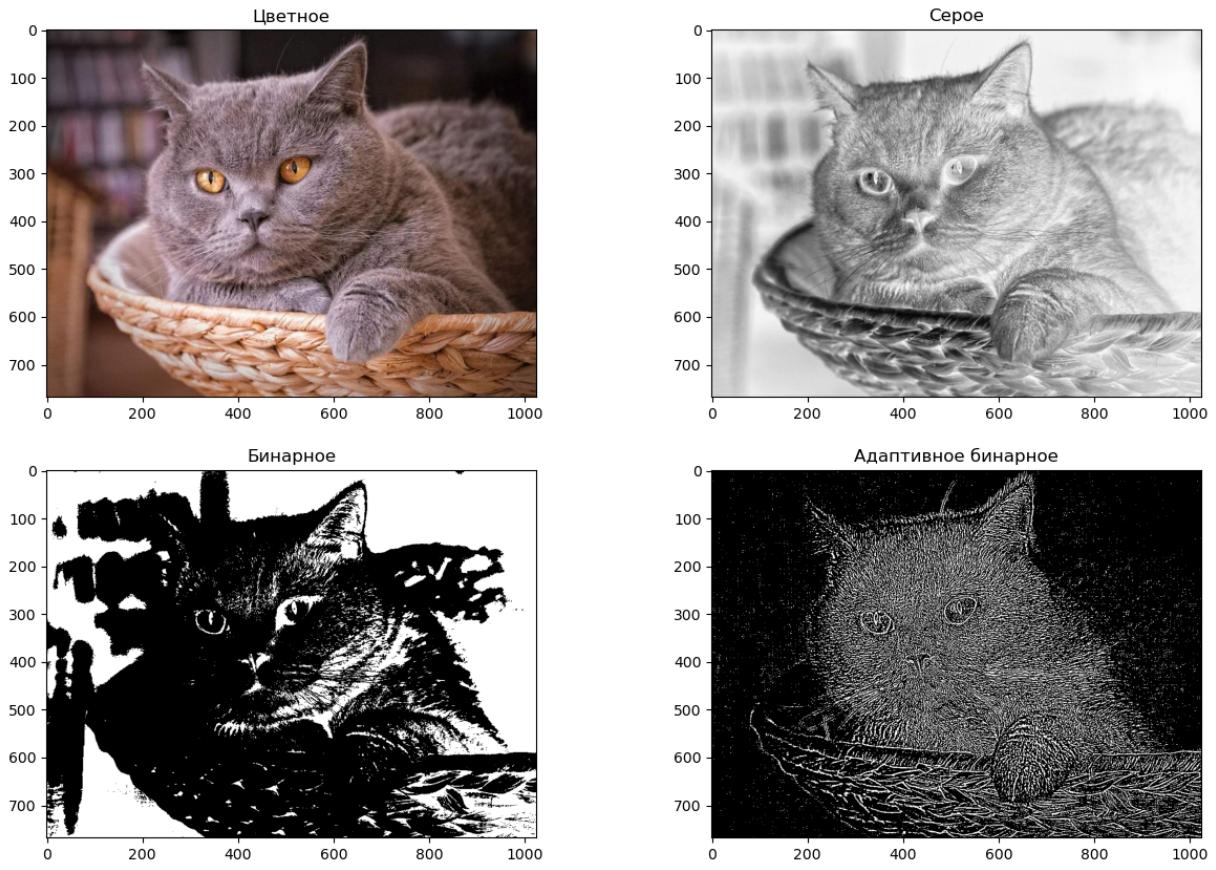


Изображения необязательно могут быть цветными, также они могут быть бинарными или в серых тонах

Формула для получения изображения в серых тонах:

$$Y = 0.299R + 0.587G + 0.114B$$

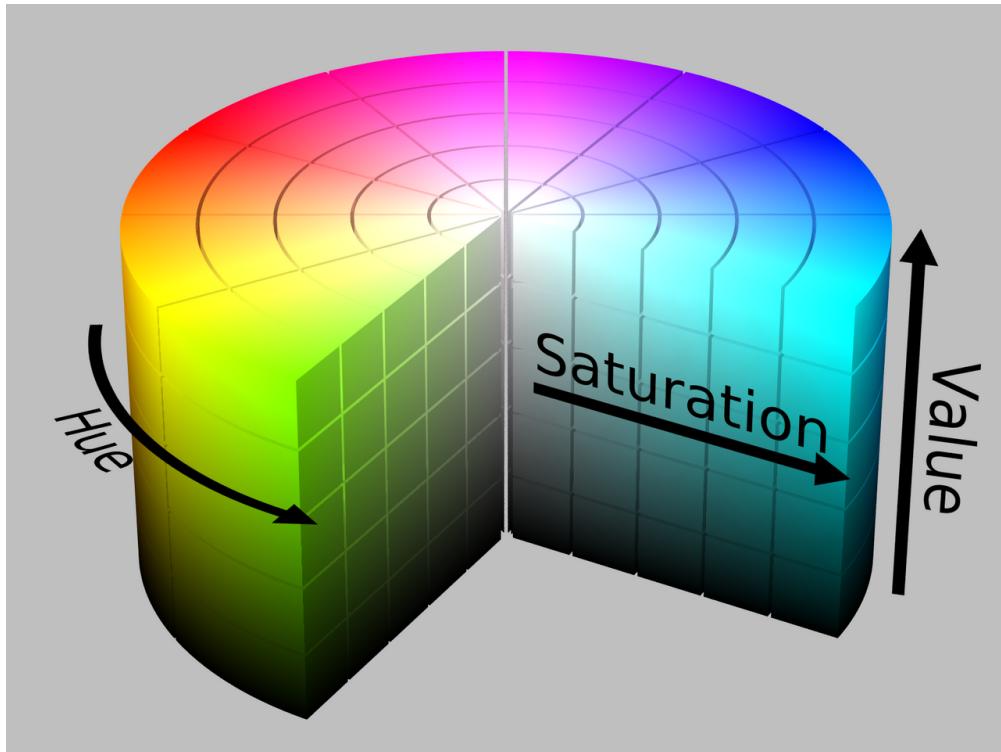
```
In [9]: plt.figure(figsize=(15, 10))  
  
plt.subplot(2, 2, 1)  
plt.title('Цветное')  
plt.imshow(cat_array)  
  
plt.subplot(2, 2, 2)  
plt.title('Серое')  
gray_image = cv.cvtColor(cat_array, cv.COLOR_BGR2GRAY)  
plt.imshow(gray_image, cmap=plt.cm.Greys)  
  
plt.subplot(2, 2, 3)  
thresh, bin_image = cv.threshold(gray_image, 75, 75, cv.THRESH_BINARY)  
plt.title('Бинарное')  
plt.imshow(bin_image, cmap=plt.cm.Greys)  
  
plt.subplot(2, 2, 4)  
adaptive_bin_image = cv.adaptiveThreshold(gray_image, 255, cv.ADAPTIVE_THRESH  
cv.THRESH_BINARY, 5, 5)  
plt.title('Адаптивное бинарное')  
plt.imshow(adaptive_bin_image, cmap=plt.cm.Greys)  
  
plt.show()
```



Бинаризация бывает:

- Глобальная (один порог для всех пикселей на фото)
- Локальная (порог зависит от положения пикселя)
- Адаптивная (порог зависит от положения и от яркости пикселя)

HSV представление



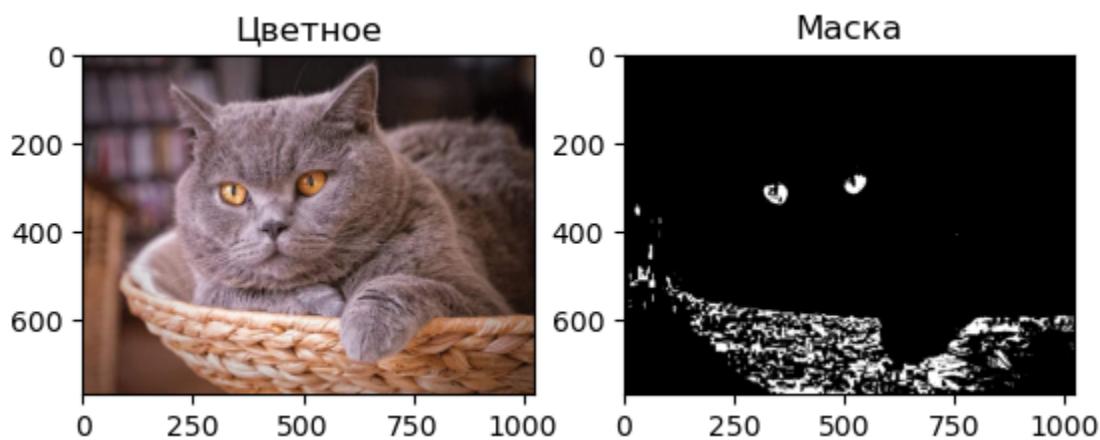
```
In [10]: hsv_cat = cv.cvtColor(cat_array, cv.COLOR_RGB2HSV)

lower_blue = np.array([11, 0, 150])
upper_blue = np.array([16, 250, 255])
mask = cv.inRange(hsv_cat, lower_blue, upper_blue)

plt.subplot(1, 2, 1)
plt.title("Цветное")
plt.imshow(cat_array)

plt.subplot(1, 2, 2)
plt.title("Маска")
plt.imshow(np.uint(mask), cmap='gray')

plt.show()
```



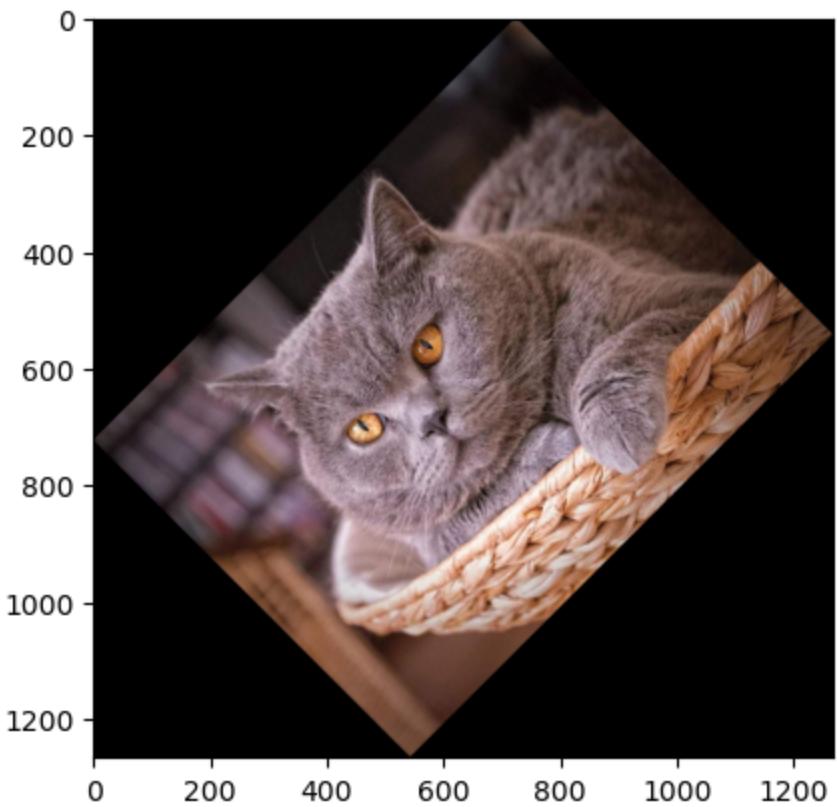
Геометрические преобразования

Изображения можно поворачивать, сдвигать и делать еще много разных интересных вещей...

```
In [11]: shifted_image = scipy.ndimage.shift(cat_array, [150, 150, 0])
plt.imshow(shifted_image)
plt.show()
```



```
In [12]: rotated_image = scipy.ndimage.rotate(cat_array, 45)
plt.imshow(rotated_image)
plt.show()
```



```
In [13]: cat_array.shape, rotated_image.shape
```

```
Out[13]: ((768, 1024, 3), (1267, 1267, 3))
```

```
In [14]: rotated_image = skimage.transform.rotate(cat_array, 45) # with resize=True etc
plt.imshow(rotated_image)
plt.show()
```



```
In [15]: cat_array.shape, rotated_image.shape
```

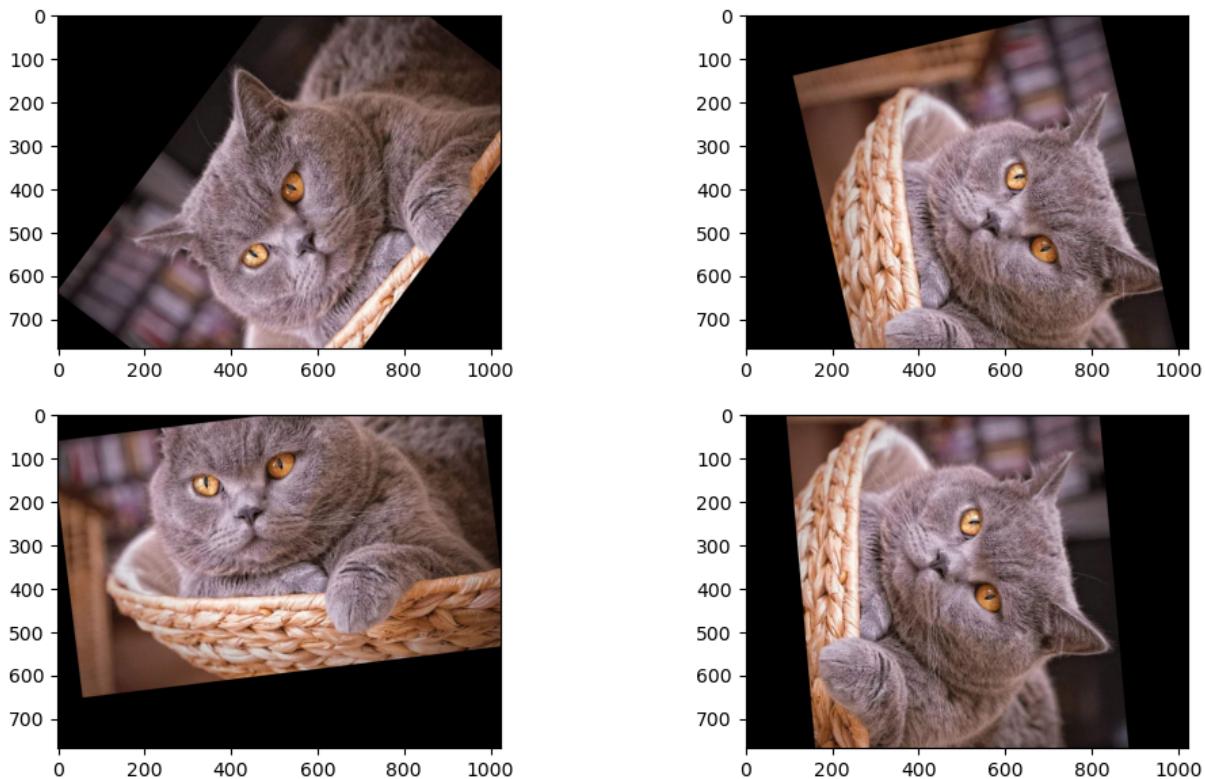
```
Out[15]: ((768, 1024, 3), (768, 1024, 3))
```

```
In [16]: def very_smart_augmentation(image_array, a, b):
    angle = np.random.randint(-b, b, 1)[0]
    shift_x, shift_y = np.random.randint(-a, a, 2)
    img = scipy.ndimage.shift(image_array, [shift_y, shift_x, 0])
    img = skimage.transform.rotate(img, angle)
    return img
```

```
In [17]: assert (very_smart_augmentation(cat_array, 200, 90) != cat_array).any(), "Many pixels have changed"
a = 200
b = 90

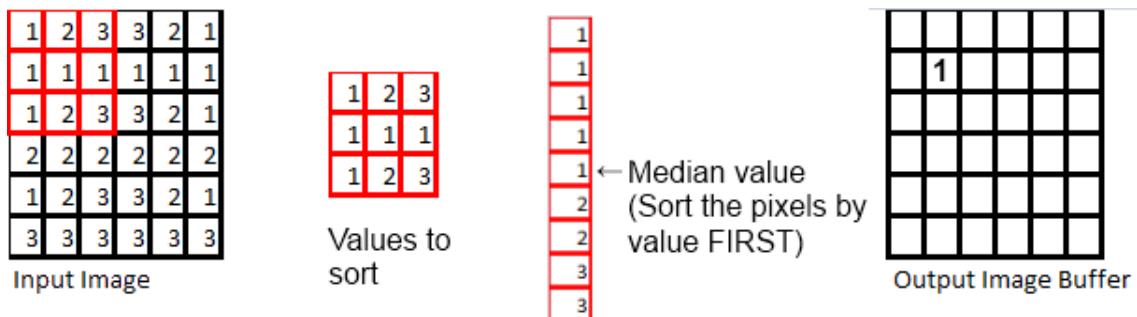
plt.figure(figsize=(12, 7))

for i in range(4):
    plt.subplot(2, 2, i + 1)
    plt.imshow(very_smart_augmentation(cat_array, a, b))
plt.show()
```



Фильтры

(Более подробно про эту сущность вы узнаете в следующем семестре на курсе ОиРИ и DL)



Данный фильтр называется медианным. С его помощью можно удалять шумы с фото

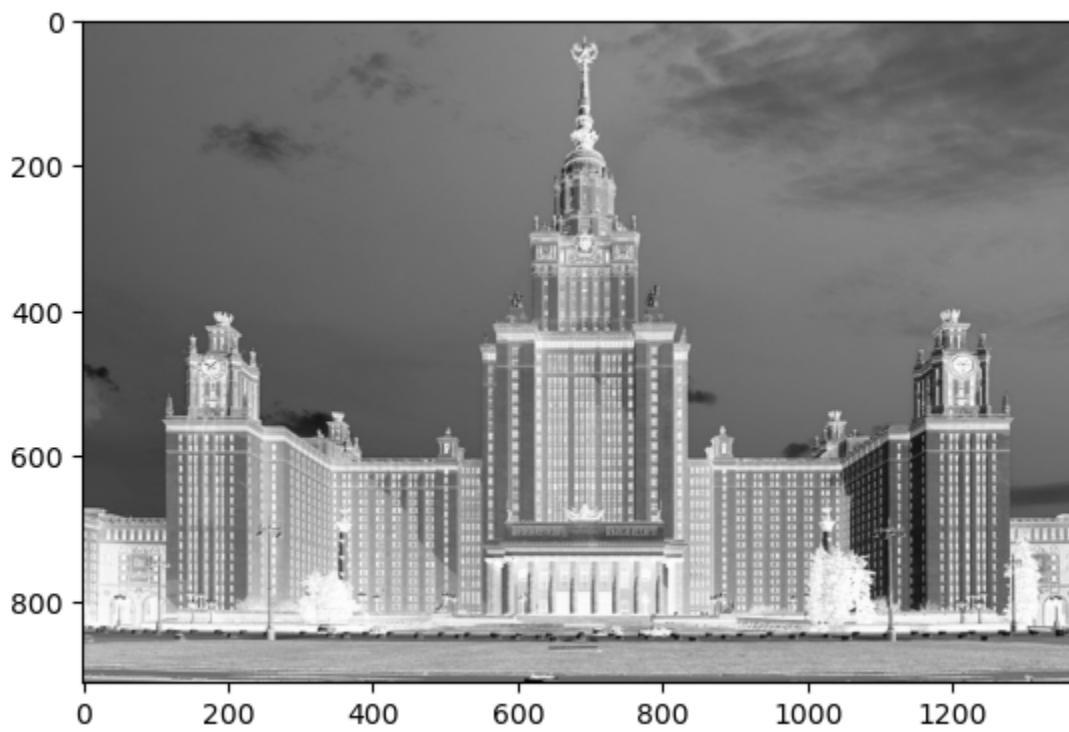
```
In [18]: !wget https://turkmenportal.com/images/uploads/blogs/b1cec01f147f463a30d6ecb03fb74900.jpg msu.jpg
!mv b1cec01f147f463a30d6ecb03fb74900.jpg msu.jpg
msu_array = skimage.io.imread('msu.jpg')
msu_gray = cv.cvtColor(msu_array, cv.COLOR_BGR2GRAY)
```

```
--2025-09-26 17:41:39-- https://turkmenportal.com/images/uploads/blogs/b1cec01f147f463a30d6ecb03fb74900.jpg
Распознаётся turkmenportal.com (turkmenportal.com)... 95.85.126.182
Подключение к turkmenportal.com (turkmenportal.com)|95.85.126.182|:443... соединение установлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
Длина: 241934 (236K) [image/jpeg]
Сохранение в: 'b1cec01f147f463a30d6ecb03fb74900.jpg'

b1cec01f147f463a30d 100%[=====] 236,26K 1,26MB/s    за 0,2s
```

2025-09-26 17:41:40 (1,26 MB/s) - 'b1cec01f147f463a30d6ecb03fb74900.jpg' сохранён [241934/241934]

```
In [19]: plt.imshow(msu_gray, cmap=plt.cm.Greys)
plt.show()
```



Создадим случайный шумовые точки и добавим их в картинку. После чего используем медианный фильтр (scipy.ndimage.median_filter), чтобы исправить изображение любимого вуза

```
In [20]: img_copy = msu_array[400:600, 400:800].copy()
noise_x = np.random.choice(img_copy.shape[1], size=10000)
noise_y = np.random.choice(img_copy.shape[0], size=10000)

plt.figure(figsize=(12, 7))

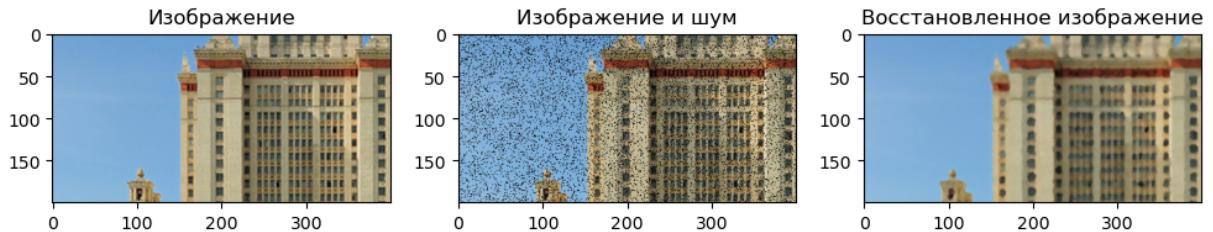
plt.subplot(1, 3, 1)
plt.imshow(img_copy)
plt.title("Изображение")
```

```

img_copy[noise_y, noise_x, :] = 0
plt.subplot(1, 3, 2)
plt.imshow(img_copy)
plt.title("Изображение и шум")

plt.subplot(1, 3, 3)
msu_filtered = cv.medianBlur(img_copy, 5)
plt.imshow(msu_filtered)
plt.title("Восстановленное изображение")
plt.show()

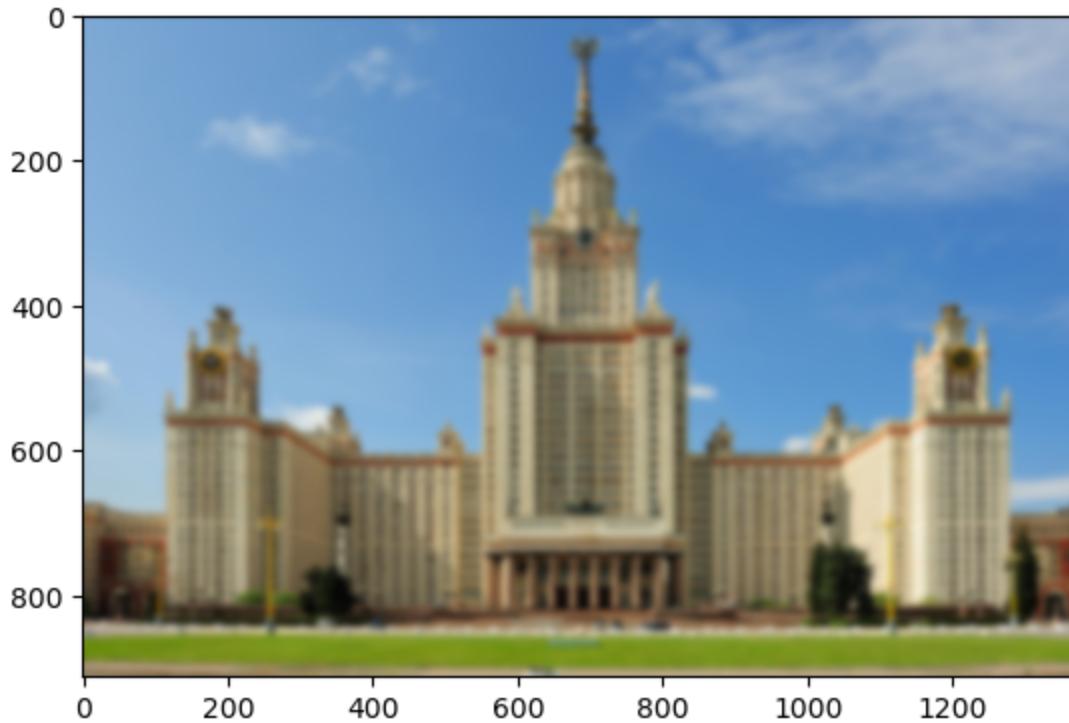
```



Пример применения гауссовского фильтра, позволяющего размыть фото.

$$Y[i, j] = \sum_{l=-k}^k \sum_{m=-k}^k X[i + l, j + m] \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\sqrt{l^2 + m^2}}{2\sigma^2}\right)$$

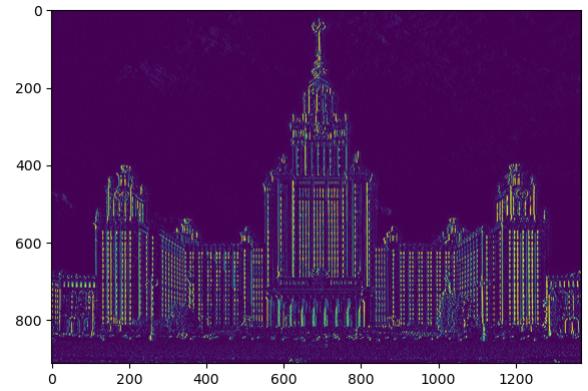
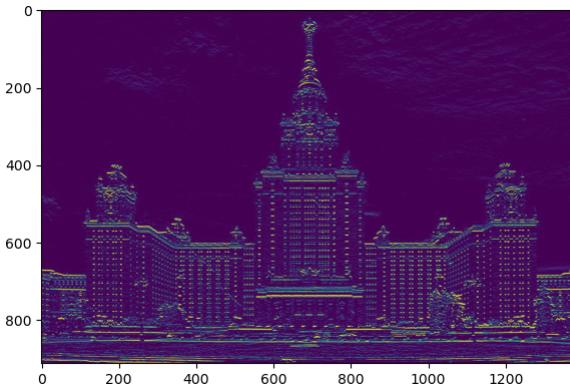
```
In [21]: blured_msu = cv.GaussianBlur(msu_array, (25, 25), 5)
plt.imshow(blured_msu)
plt.show()
```



Выделение контуров изображения с помощью фильтров

```
In [22]: plt.figure(figsize=(15, 10))
plt.subplot(1, 2, 1)
kernel = np.array([[1, 2, 1],
                  [0, 0, 0],
                  [-1, -2, -1]])
msu_horizontal = cv.filter2D(msu_gray, 0, kernel)
plt.imshow(msu_horizontal)

plt.subplot(1, 2, 2)
kernel = np.array([[1, 0, -1],
                  [2, 0, -2],
                  [1, 0, -1]])
msu_vertical = cv.filter2D(msu_gray, 0, kernel)
plt.imshow(msu_vertical)
plt.show()
```



SVD

[Сайт с демонстрацией](#)

Пусть $A \in \mathbb{R}^{m \times n}$, тогда SVD разложение A записывается в следующем виде:

$$A = U\Sigma V^*, \quad U \in \mathbb{R}^{m \times m}, \quad \Sigma \in \mathbb{R}^{m \times n}, \quad V \in \mathbb{R}^{n \times n}$$

$$UU^* = I_m, \quad V^*V = I_n$$

Альтернативная запись:

$$A = \sum_{i=1}^{\text{rank}(A)} \sigma_i u_i v_i^*$$

Какие хорошие свойства SVD вы знаете?

Псевдообратная матрица

$A^\dagger = V\Sigma^{-1}U^*$ или же...

$$A^\dagger = \lim_{\lambda \rightarrow +0} (A^*A + \lambda I)^{-1}A^*$$

Чем-то напоминает решение линейной регрессии...

Какой смысл у псевдообратной матрицы?

$Aw = y$, то $\hat{w} := A^\dagger y$ соответствует $\operatorname{argmin}_w \|Aw - y\|_2$, причем если решений бесконечно много, то \hat{w} имеет наименьшую норму

[Псевдообратная матрица](#)

[Связь с линейной регрессией](#)

Малоранговое приближение

Пусть $A \in \mathbb{R}^{m \times n}$, назовем

$$A_k = U_k \Sigma_k V_k^*, \quad U_k = U[:, :k], \quad \Sigma_k = \Sigma[:, :k], \quad V_k = V[:, :k]$$

Из курса линейной алгебры известно

$$A_k = \operatorname{argmin}_{B \in \mathbb{R}^{m \times n}, \operatorname{rank}(B)=k} \|A - B\|_F$$

Что такое $\|\cdot\|_F$? Чему равно $\|A - A_k\|_F$?

```
In [23]: def trunc_svd(A, k):
    U, S, Vh = np.linalg.svd(A, full_matrices=False)
    return U[:, :k + 1], S[:k + 1], Vh[:k + 1]
```

```
In [24]: k = 20
c = 0
A = np.copy(msu_array)
A_c = A[:, :, c]
_, sing_vals, _ = np.linalg.svd(A_c)

U, S, Vh = trunc_svd(A_c, k)
A_k = U @ np.diag(S) @ Vh

assert abs(np.linalg.norm(A_c - A_k) - np.linalg.norm(sing_vals[k + 1:])) <
```

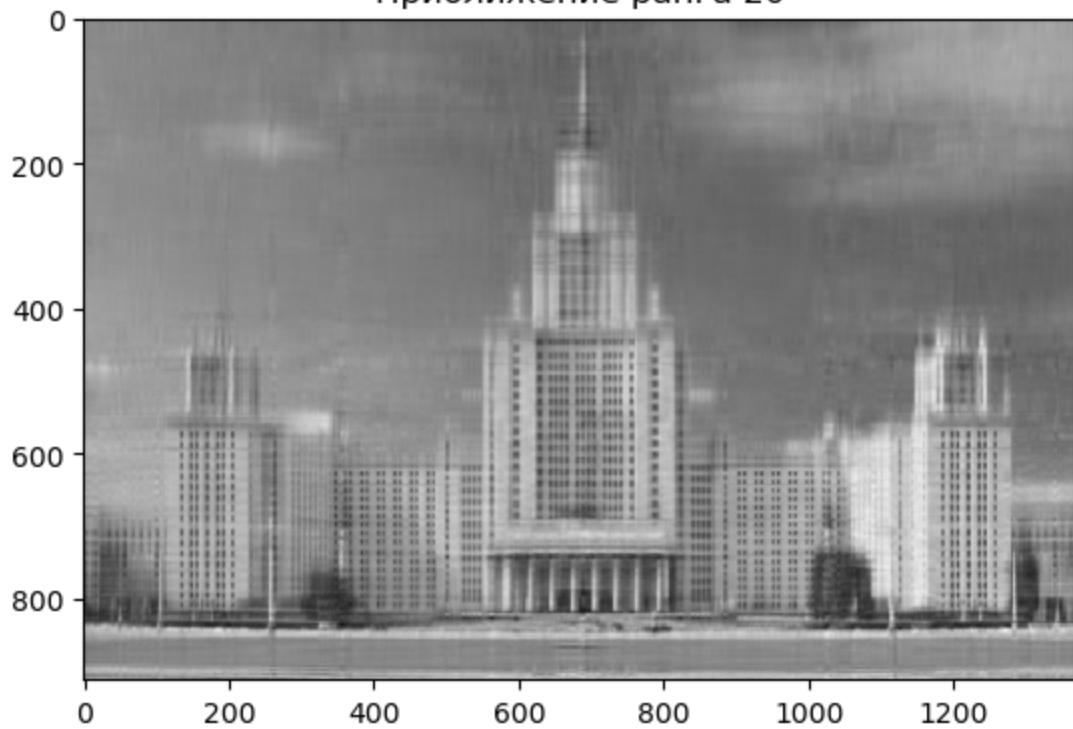
```
In [25]: plt.imshow(A_c, cmap="gray")
plt.title("Исходное изображение")
plt.show()

plt.imshow(A_k, cmap="gray")
plt.title(f"Приближение ранга {k}")
plt.show()
```

Исходное изображение

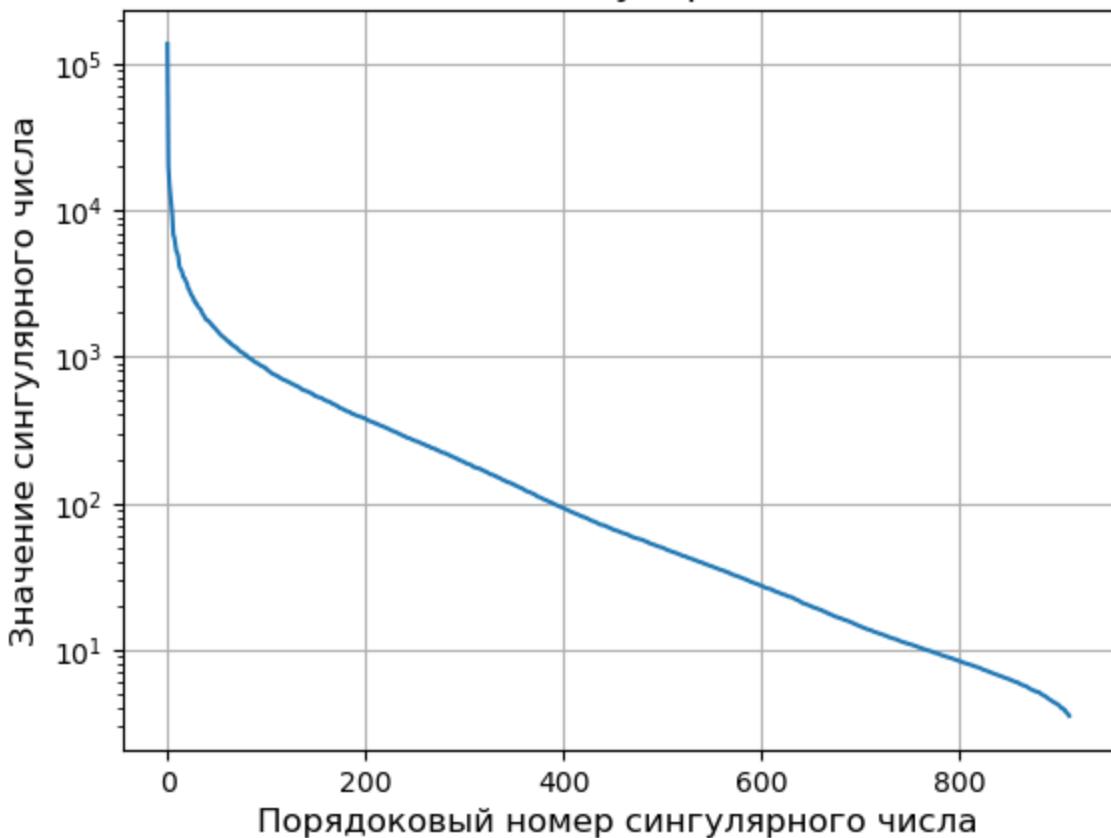


Приближение ранга 20



```
In [26]: plt.title("Значение сингулярных чисел", fontsize=14)
plt.plot(sing_vals)
plt.grid()
plt.xlabel("Порядковый номер сингулярного числа", fontsize=12)
plt.ylabel("Значение сингулярного числа", fontsize=12)
plt.yscale('log')
plt.show()
```

Значение сингулярных чисел



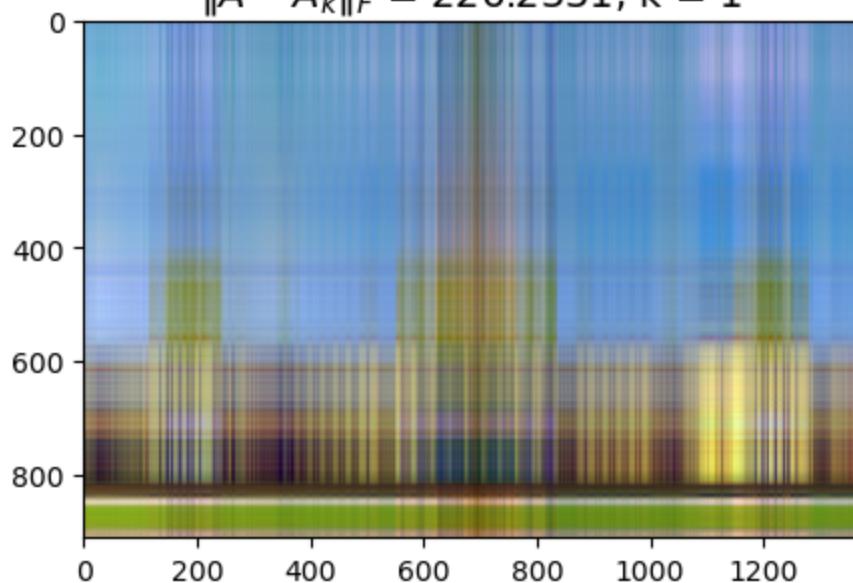
```
In [27]: for i in range(3):
    r = np.linalg.matrix_rank(A[:, :, i])
    print(f"rank(A[:, :, {i}]) = {r}")

k_grid = [1, 2, 5, 10, 20, 50, 100, 200, 400, 912]
A = msu_array.copy() / 255

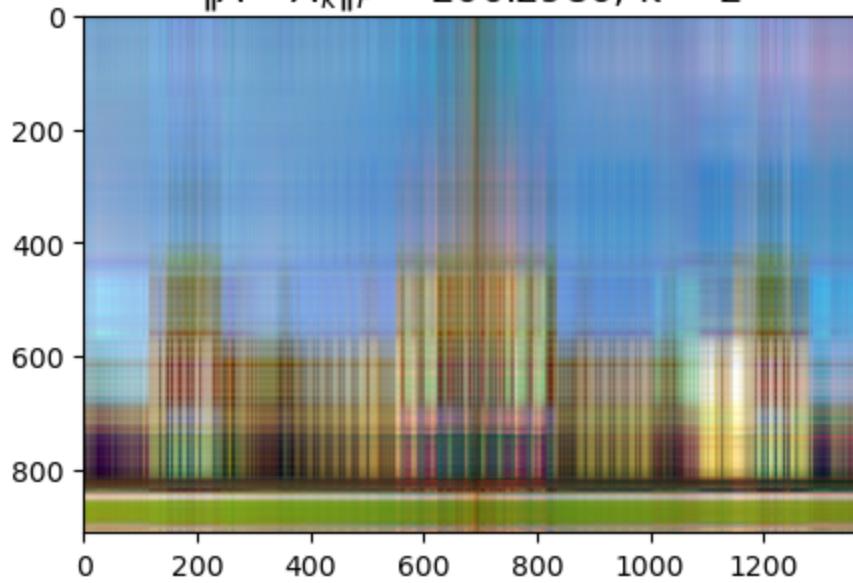
for idx, k in enumerate(k_grid):
    A_k = np.zeros_like(A)
    for i in range(3):
        U, S, Vh = trunc_svd(A[:, :, i], k)
        A_k[:, :, i] = U @ np.diag(S) @ Vh
        A_k = np.clip(A_k, 0, 1)
    plt.figure(figsize=(5, 5))
    plt.imshow(A_k)
    plt.title(fr"\|A - A_k\|_F = {np.linalg.norm(A - A_k):0.4f}, k = {k}")
    plt.show()

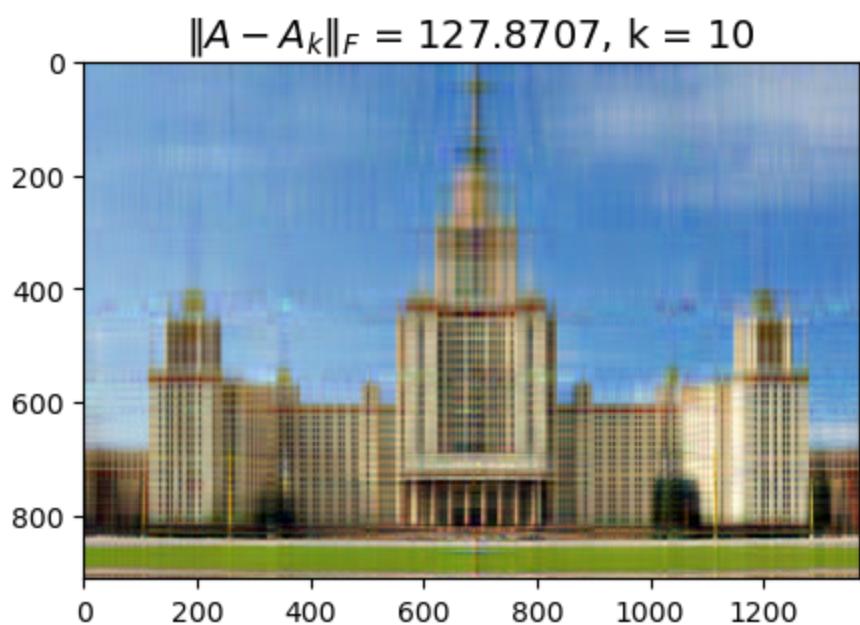
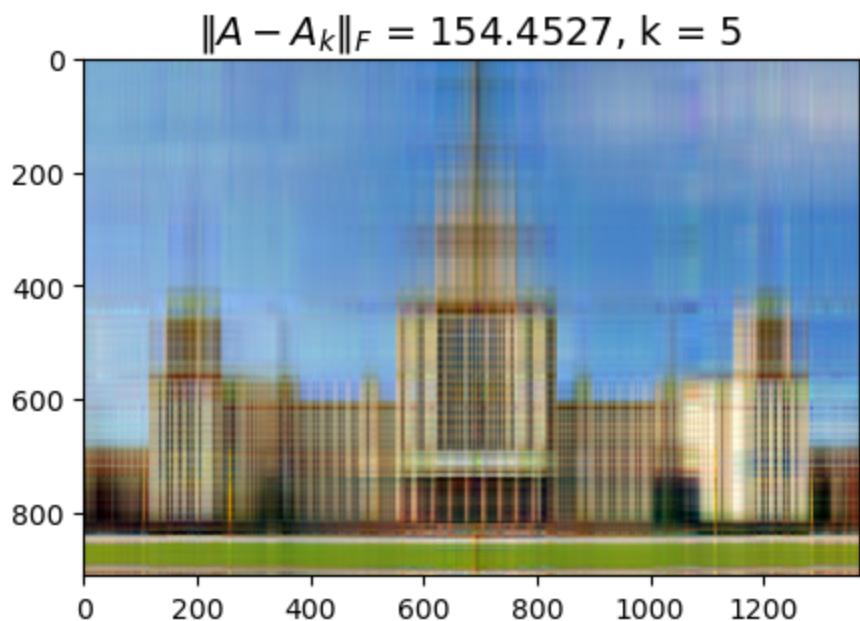
rank(A[:, :, 0]) = 912
rank(A[:, :, 1]) = 912
rank(A[:, :, 2]) = 912
```

$$\|A - A_k\|_F = 226.2331, k = 1$$

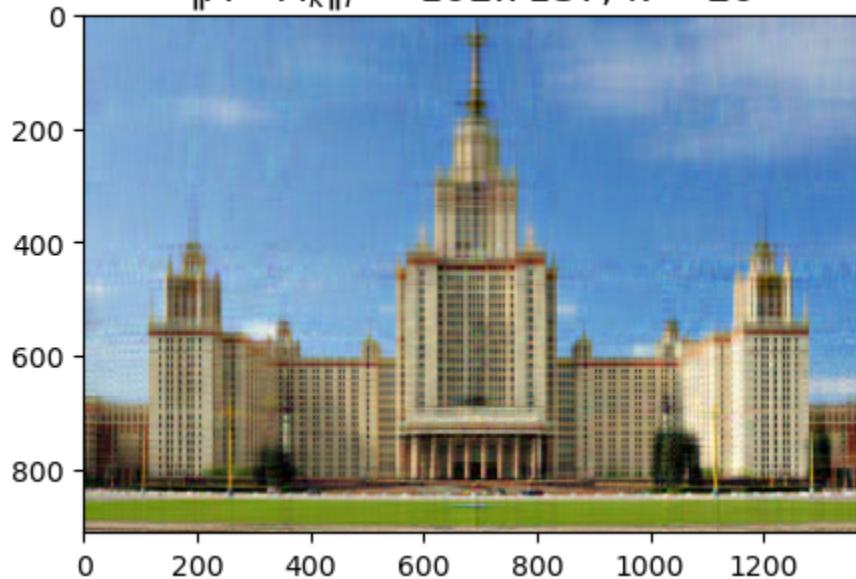


$$\|A - A_k\|_F = 200.2980, k = 2$$





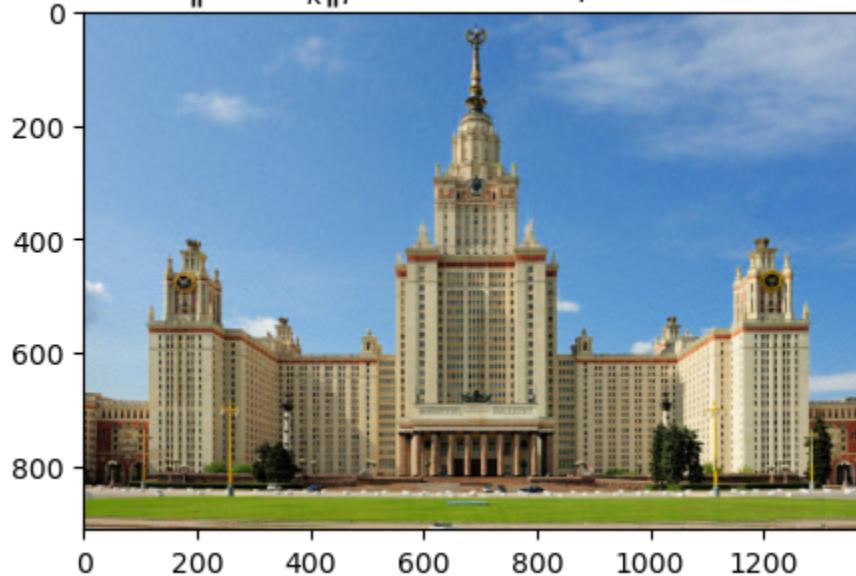
$$\|A - A_k\|_F = 102.7137, k = 20$$



$$\|A - A_k\|_F = 69.0642, k = 50$$

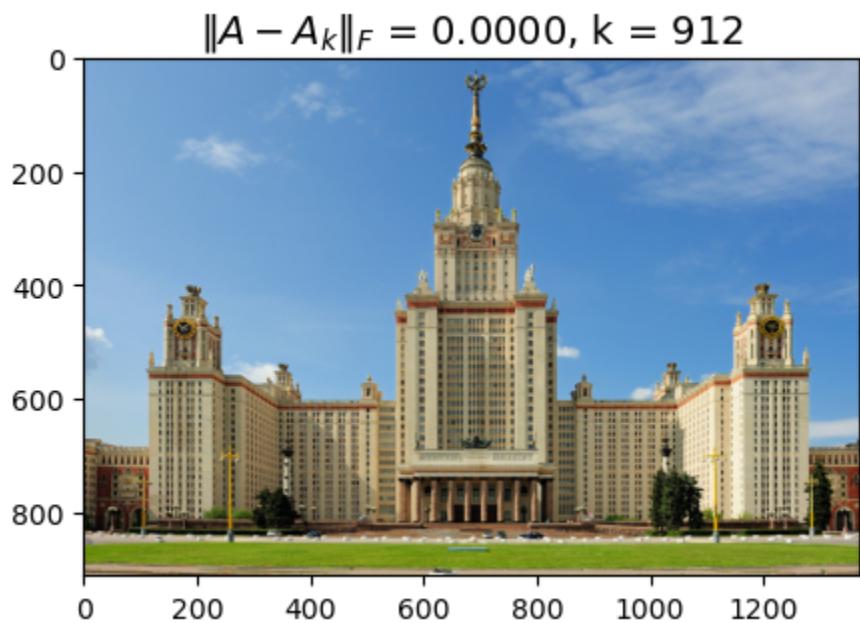
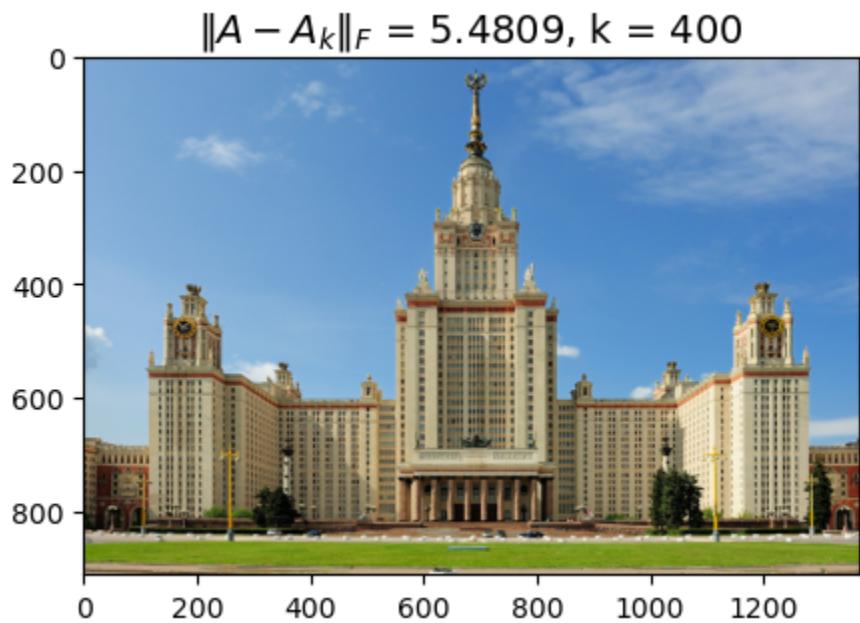


$$\|A - A_k\|_F = 44.1474, k = 100$$



$$\|A - A_k\|_F = 21.5253, k = 200$$





Преобразование Фурье

Формулка дискретного двумерного преобразования Фурье:

$$F(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-i2\pi(\frac{ux}{M} + \frac{vy}{N})}$$

[Пример из opencv](#)

```
In [28]: def fft_transform(img, k, vis=False):
```

```
    rows, cols = img.shape
```

```

crow, ccol = rows // 2, cols // 2

dft = cv.dft(np.float32(img), flags=cv.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)

magnitude_spectrum = 20 * np.log(cv.magnitude(dft_shift[:, :, 0]), dft_shift[:, :, 1])
result = magnitude_spectrum[crow - k:crow + k + 1, ccol - k:ccol + k + 1]

if vis:
    mask = np.zeros((rows, cols, 2), np.uint8)
    mask[crow - k:crow + k + 1, ccol - k:ccol + k + 1] = 1

    print(f"Размерность исходного пространства \t= {img.shape[0]} * {img.shape[1]}")
    print(f"Размерность нового пространства \t= {result.shape[0]} * {result.shape[1]}")

    fshift = dft_shift * mask
    f_ishift = np.fft.ifftshift(fshift)
    img_back = cv.idft(f_ishift)
    img_back = cv.magnitude(img_back[:, :, 0], img_back[:, :, 1])

    plt.figure(figsize=(20, 34))
    plt.subplot(1, 3, 1)
    plt.imshow(img, cmap = 'gray')
    plt.title('Исходное изображение', fontsize=15)
    plt.xticks([])
    plt.yticks([])

    plt.subplot(1, 3, 2)
    plt.imshow(magnitude_spectrum, cmap = 'gray')
    plt.title('Амплитуды спектра', fontsize=15)
    plt.xticks([])
    plt.yticks([])

    plt.subplot(1, 3, 3)
    plt.imshow(img_back, cmap = 'gray')
    plt.title('Восстановленное изображение', fontsize=15)
    plt.xticks([])
    plt.yticks([])

    plt.show()

return result

```

In [29]:

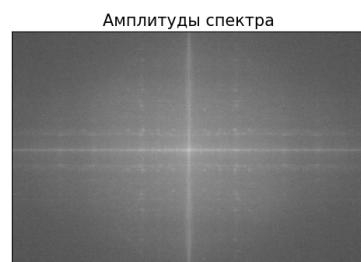
```

k = 120
img = cv.imread('msu.jpg', cv.IMREAD_GRAYSCALE)

_ = fft_transform(img, k, vis=True)

```

Размерность исходного пространства = 1248528
 Размерность нового пространства = 58081



```
In [30]: !wget https://i.pinimg.com/736x/3f/35/f2/3f35f2051a296a05a62d2861196a667a.jpg  
!mv 3f35f2051a296a05a62d2861196a667a.jpg diag.jpg
```

--2025-09-26 17:44:40-- https://i.pinimg.com/736x/3f/35/f2/3f35f2051a296a05a62d2861196a667a.jpg
Распознаётся i.pinimg.com (i.pinimg.com)... 185.5.161.209, 185.5.161.224, 185.5.161.232
Подключение к i.pinimg.com (i.pinimg.com)|185.5.161.209|:443... соединение установлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
Длина: 29150 (28K) [image/jpeg]
Сохранение в: '3f35f2051a296a05a62d2861196a667a.jpg'

3f35f2051a296a05a62 100%[=====] 28,47K -.- KB/s за 0,002s

2025-09-26 17:44:42 (15,3 MB/s) - '3f35f2051a296a05a62d2861196a667a.jpg' сохранён [29150/29150]

In [31]:

```
k = 120
img = cv.imread('diag.jpg', cv.IMREAD_GRAYSCALE)

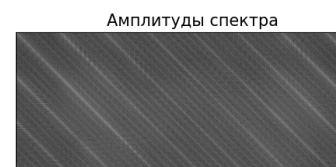
_ = fft_transform(img, k, vis=True)
```

Размерность исходного пространства = 391876
Размерность нового пространства = 58081

Исходное изображение



Амплитуды спектра



Восстановленное изображение



Пример использования преобразования Фурье на датасете Cifar-10

Из чего состоит Cifar-10

```
In [32]: from torchvision.datasets import CIFAR10
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from PIL import ImageOps
from functools import partial
import time
```

```
In [33]: data_train = CIFAR10("./", train=True, download=True)
         data_test = CIFAR10("./", train=False, download=True)
```

```
In [34]: def get_ml_dataset(data, transform=lambda x: x):
    X, y = [], []
    for i in range(len(data)):
        img, target = data[i]
        img_array = np.array(ImageOps.grayscale(img))
        img_array = transform(img_array)
        X.append(img_array)
        y.append(target)
    return np.array(X), np.array(y)
```

```
In [35]: X_train, y_train = get_ml_dataset(data_train)
X_test, y_test = get_ml_dataset(data_test)
X_train.shape, X_test.shape
```

```
Out[35]: ((50000, 32, 32), (10000, 32, 32))
```

```
In [36]: plt.figure(figsize=(10, 10))
for i in range(16):
    plt.subplot(4, 4, i + 1)
    plt.imshow(X_train[i], cmap="gray")
    plt.title(f'y = {y_train[i]}')
    plt.xticks([])
    plt.yticks([])
plt.show()
```



```
In [37]: %%time
clf = LogisticRegression()
clf.fit(X_train.reshape(X_train.shape[0], -1), y_train)
```

CPU times: user 30.9 s, sys: 344 ms, total: 31.3 s
Wall time: 5.46 s

/home/egor/.local/lib/python3.10/site-packages/scikit-learn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
<https://scikit-learn.org/stable/modules/preprocessing.html>
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
n_iter_i = _check_optimize_result()

```
Out[37]: LogisticRegression
```

```
LogisticRegression()
```

```
In [38]: def calc_accuracy(X, y):
    y_pred = clf.predict(X.reshape(X.shape[0], -1))
    return accuracy_score(y_pred, y)
```

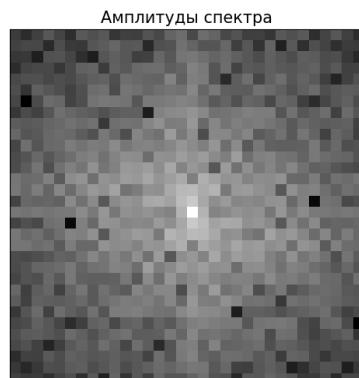
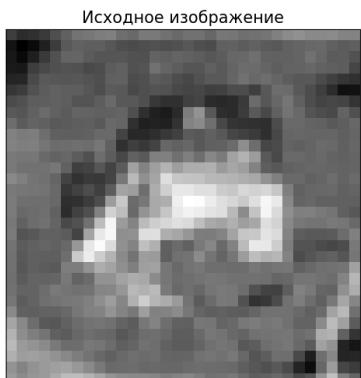
```
In [39]: score_train = calc_accuracy(X_train, y_train)
score_test = calc_accuracy(X_test, y_test)

print(f"train score\t = {score_train:.4f}")
print(f"test score\t = {score_test:.4f}")
```

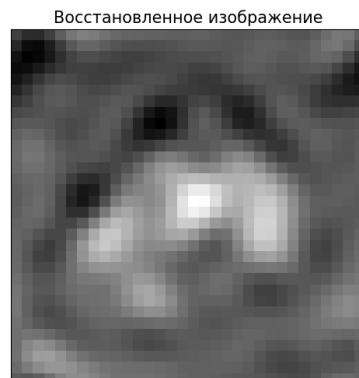
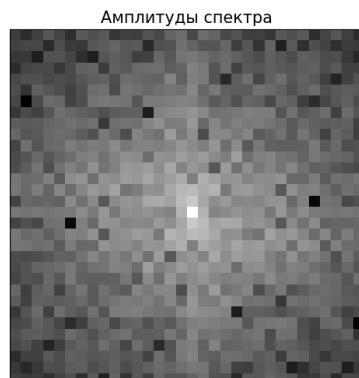
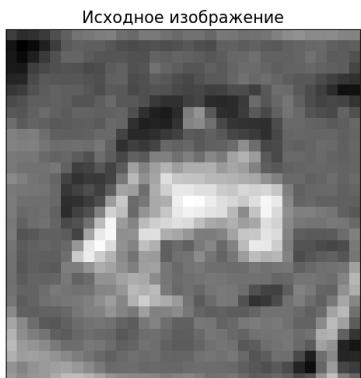
```
train score      = 0.3084
test score      = 0.2882
```

```
In [40]: for k in [1, 5, 10, 14]:
    print(f"=====k = {k}====")
    img = X_train[0]
    _ = fft_transform(img, k, vis=True)
```

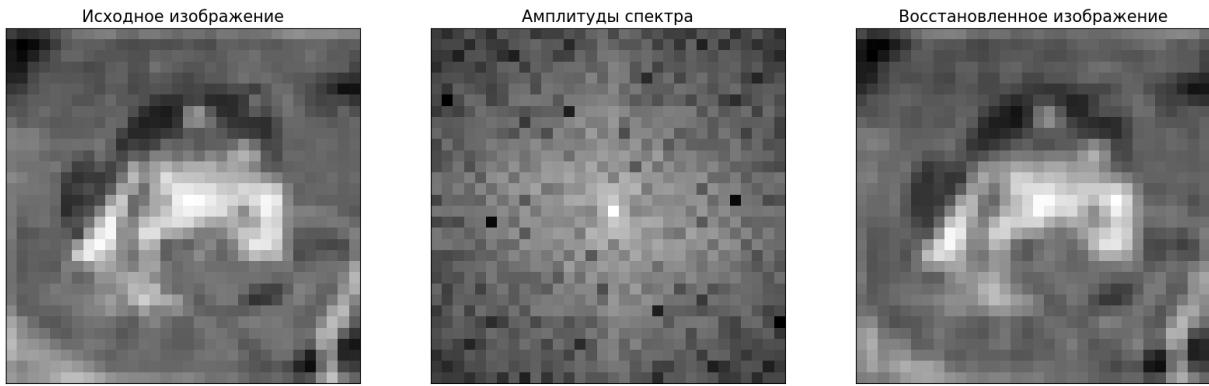
```
=====k = 1=====
Размерность исходного пространства      = 1024
Размерность нового пространства        = 9
```



```
=====k = 5=====
Размерность исходного пространства      = 1024
Размерность нового пространства        = 121
```



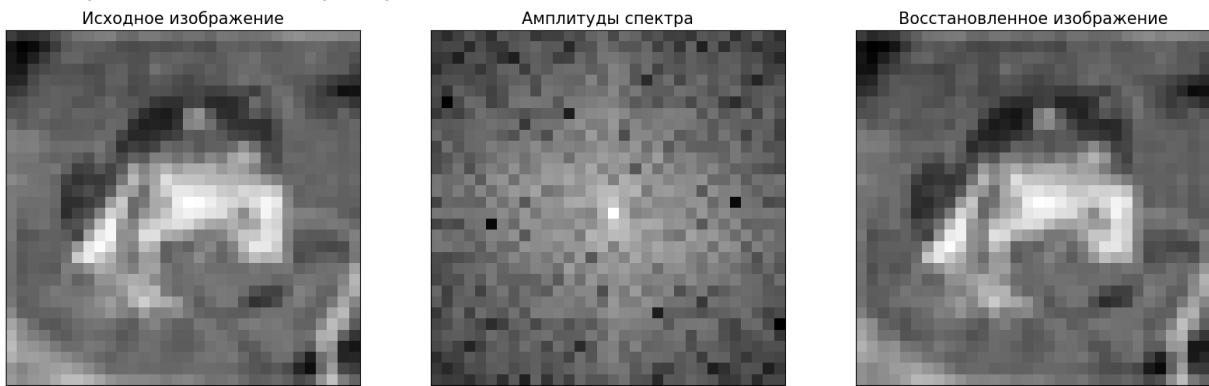
```
=====k = 10=====
Размерность исходного пространства     = 1024
Размерность нового пространства       = 441
```



=====k = 14=====

Размерность исходного пространства = 1024

Размерность нового пространства = 841



```
In [41]: for k in [1, 5, 10, 14]:
    print(f"=====k = {k}=====")
    fft_transform_k = partial(fft_transform, k=k)

    X_train, y_train = get_ml_dataset(data_train, fft_transform_k)
    X_test, y_test = get_ml_dataset(data_test, fft_transform_k)
    print(f"Размерность признаков \t\t= {X_train.shape[1] * X_train.shape[2]}")

    t_start = time.time()
    clf = LogisticRegression()
    clf.fit(X_train.reshape(X_train.shape[0], -1), y_train)
    t = time.time() - t_start
    print(f"Время обучения \t\t= {t}")

    score_train = calc_accuracy(X_train, y_train)
    score_test = calc_accuracy(X_test, y_test)

    print(f"train score\t\t= {score_train:.4f}")
    print(f"test score\t\t= {score_test:.4f}")
```

=====k = 1=====

```
/tmp/ipykernel_11165/1579547970.py:9: RuntimeWarning: divide by zero encountered in log
```

```
    magnitude_spectrum = 20 * np.log(cv.magnitude(dft_shift[:, :, 0], dft_shift[:, :, 1]))
```

Размерность признаков = 9

```
/home/egor/.local/lib/python3.10/site-packages/sklearn/linear_model/_logisti
c.py:469: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion
    n_iter_i = _check_optimize_result(
/tmp/ipykernel_11165/1579547970.py:9: RuntimeWarning: divide by zero encounte
red in log
    magnitude_spectrum = 20 * np.log(cv.magnitude(dft_shift[:, :, 0], dft_shift
[:, :, 1]))
Время обучения          = 0.7859847545623779
train score             = 0.1984
test score              = 0.1942
=====k = 5=====
Размерность признаков      = 121
/home/egor/.local/lib/python3.10/site-packages/sklearn/linear_model/_logisti
c.py:469: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion
    n_iter_i = _check_optimize_result(
/tmp/ipykernel_11165/1579547970.py:9: RuntimeWarning: divide by zero encounte
red in log
    magnitude_spectrum = 20 * np.log(cv.magnitude(dft_shift[:, :, 0], dft_shift
[:, :, 1]))
Время обучения          = 1.1543936729431152
train score             = 0.3092
test score              = 0.3009
=====k = 10=====
Размерность признаков      = 441
/home/egor/.local/lib/python3.10/site-packages/sklearn/linear_model/_logisti
c.py:469: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion
    n_iter_i = _check_optimize_result(
/tmp/ipykernel_11165/1579547970.py:9: RuntimeWarning: divide by zero encounte
red in log
    magnitude_spectrum = 20 * np.log(cv.magnitude(dft_shift[:, :, 0], dft_shift
[:, :, 1]))
```

```

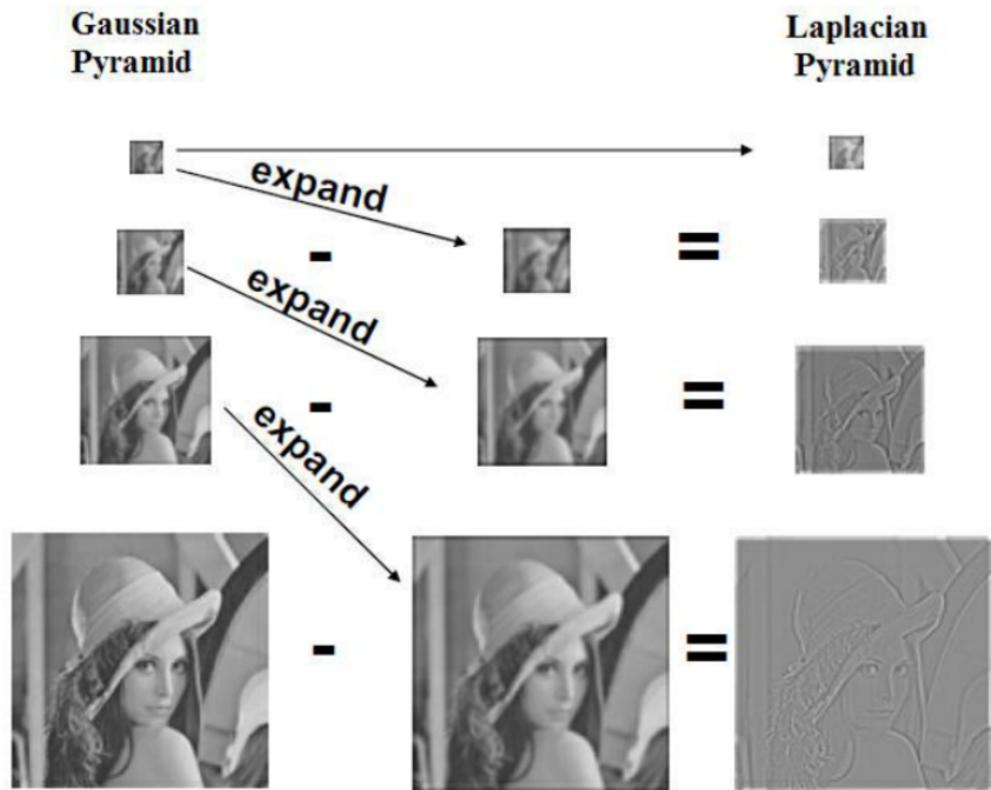
Время обучения          = 2.617600202560425
train score            = 0.3698
test score             = 0.3536
=====k = 14=====
Размерность признаков      = 841
/home/egor/.local/lib/python3.10/site-packages/sklearn/linear_model/_logisti
c.py:469: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

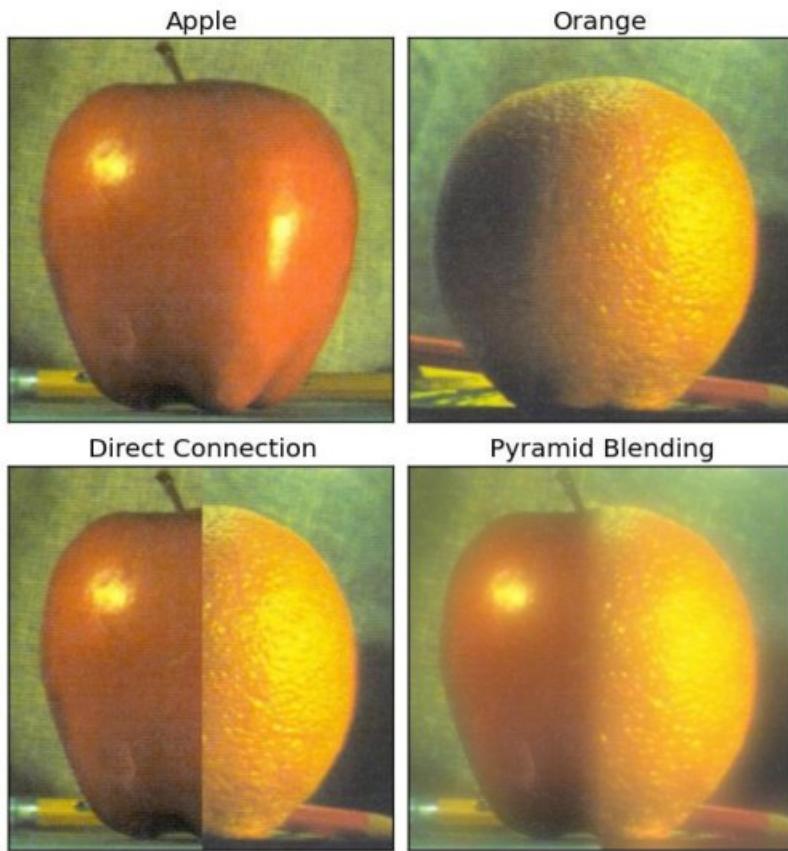
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion
    n_iter_i = _check_optimize_result(
Время обучения          = 3.198789119720459
train score            = 0.3907
test score             = 0.3616

```

Пирамиды изображений

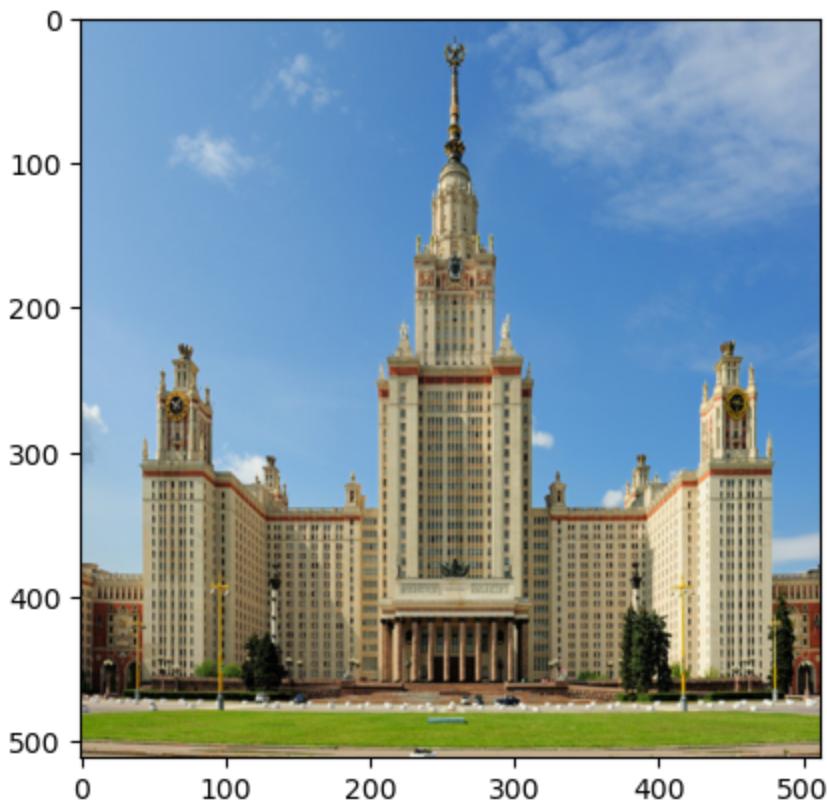
Пример из opencv





```
In [42]: img = Image.open('msu.jpg').resize((512, 512))
img = np.array(img)
plt.imshow(img)
img.shape
```

```
Out[42]: (512, 512, 3)
```



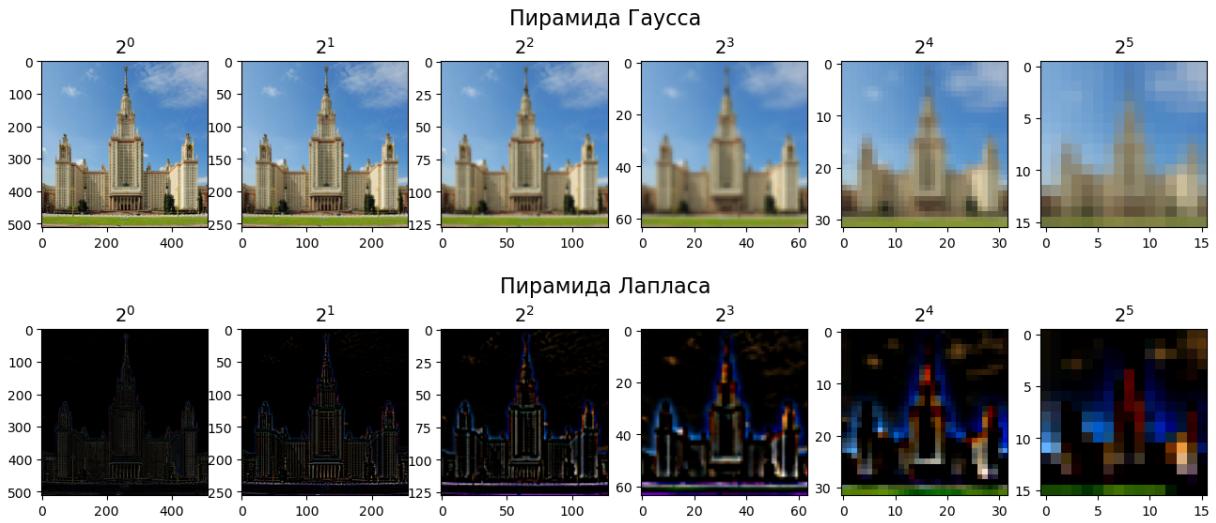
```
In [43]: N = 6
G = img.copy()
gpA = [G]
for i in range(N):
    G = cv.pyrDown(G)
    gpA.append(G)

lpA = [gpA[N - 1]]
for i in range(N, 0, -1):
    GE = cv.pyrUp(gpA[i])
    L = cv.subtract(gpA[i-1], GE)
    lpA.append(L)

fig = plt.figure(figsize=(16, 3))
for i, G in enumerate(gpA[:-1]):
    plt.subplot(1, len(gpA) - 1, i+1)
    plt.imshow(G)
    plt.title(fr"$2^{i}$", fontsize=14)
fig.suptitle("Пирамида Гаусса", fontsize=16)
plt.show()

fig = plt.figure(figsize=(16, 3))
for i, L in enumerate(lpA[1:][::-1]):
    plt.subplot(1, len(lpA) - 1, i + 1)
    plt.imshow(L / np.max(L))
    plt.title(fr"$2^{i}$", fontsize=14)

fig.suptitle("Пирамида Лапласа", fontsize=16)
plt.show()
```



Идеи, которые, казалось бы, являются достаточно древними, находят свое применение в передовых методах. Как пример, текущий SOTA метод в авторегрессионной генерации изображений!

статья

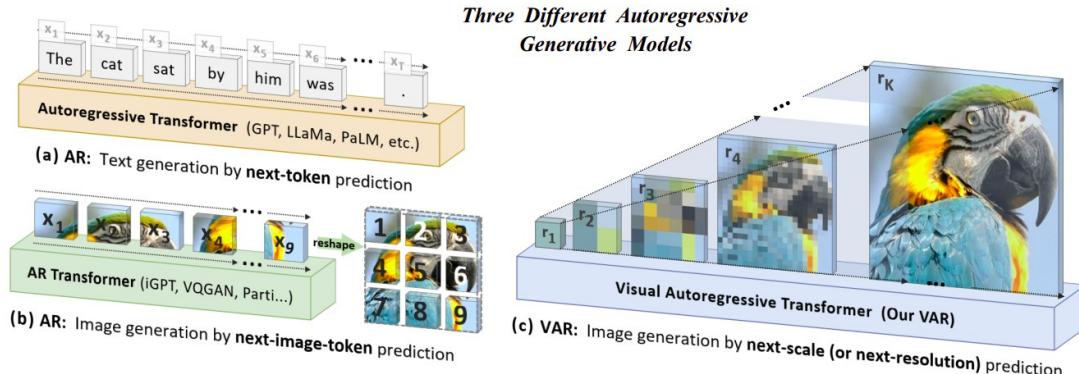


Figure 2: **Standard autoregressive modeling (AR) vs. our proposed visual autoregressive modeling (VAR).** (a) AR applied to language: sequential text token generation from left to right, word by word; (b) AR applied to images: sequential visual token generation in a raster-scan order, from left to right, top to bottom; (c) VAR for images: multi-scale token maps are autoregressively generated from coarse to fine scales (lower to higher resolutions), with parallel token generation within each scale. VAR requires a multi-scale VQVAE to work.

Примеры использования обработки изображений

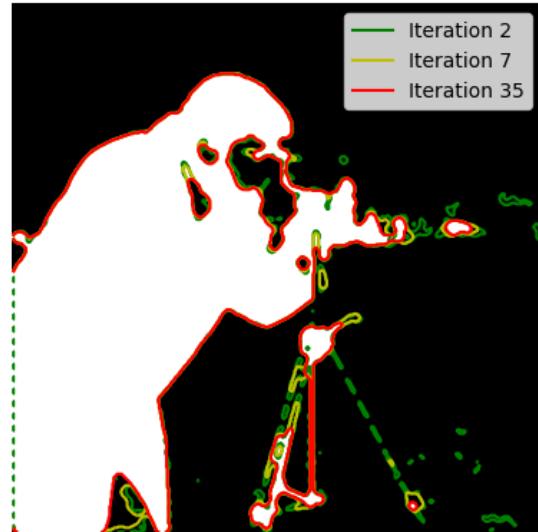
- Предобработка входных изображений (Удаление шума, преобразование цветов)
- Постобработка изображений (Удаление шумовых объектов, отверстий)
- Выделение дополнительных признаков (Выделение важных объектов, областей)
- Аугментация объектов (Преобразование объектов в ходе обучения/применения

модели)

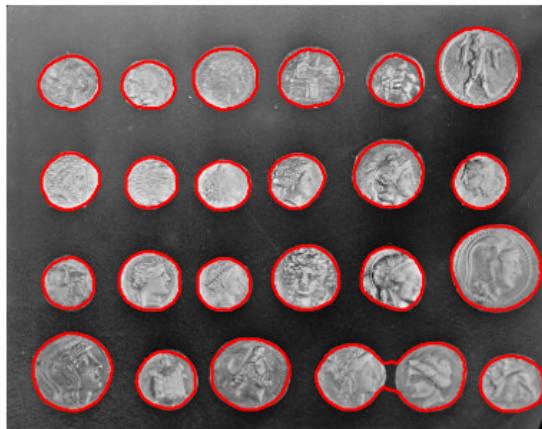
Morphological ACWE segmentation



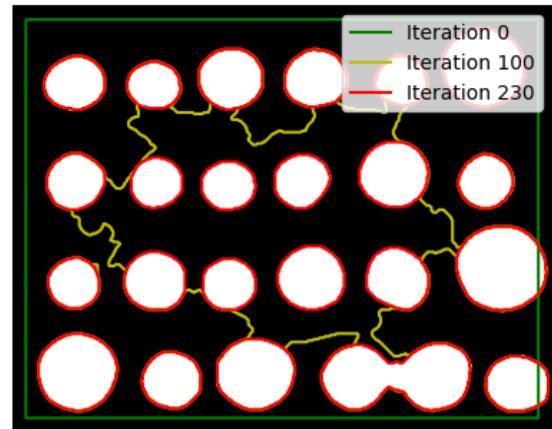
Morphological ACWE evolution



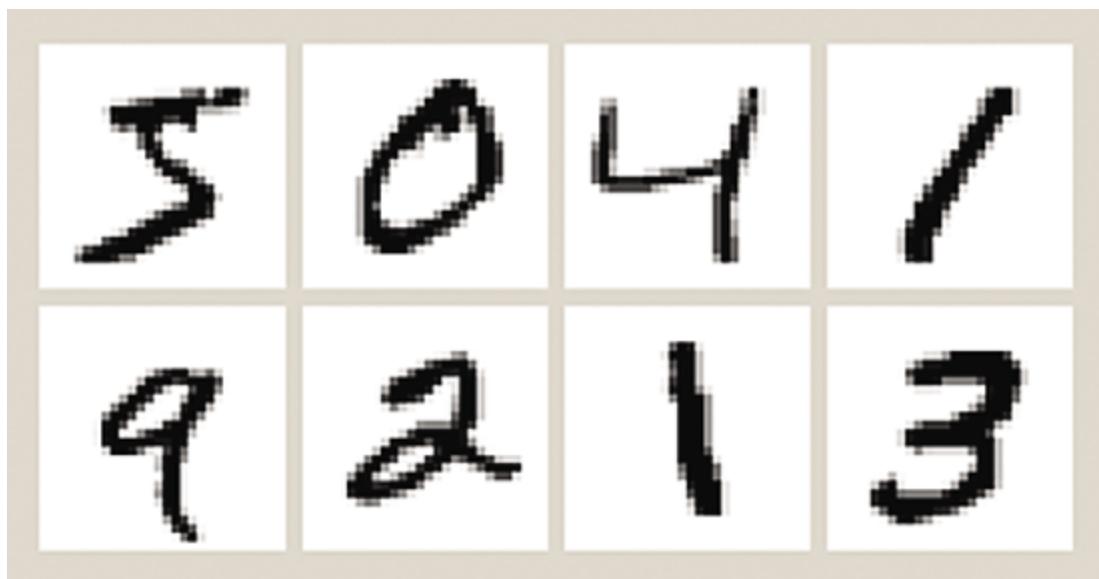
Morphological GAC segmentation



Morphological GAC evolution



Хороший пример использования трансформаций: датасет MNIST



Класс изображения не меняется при:

- сдвигах на 1-10 пикселей
- поворотах на 10-15 градусов в каждую из сторон
- размытии, удалении шумов

Добавив преобразованные объекты в исходную выборку, можно существенно повысить качество

Часто используемые методы аугментации:

- отражение
- сдвиг
- поворот
- масштабирование (увеличение)
- добавление шума, размытие
- изменение яркости, контрастности, цветовой гаммы
- удаление части изображения

In []:

