

Разведочный анализ данных (EDA)

Восстановление цвета фотографий при помощи средств машинного обучения

Название программы:

Цифровое моделирование и суперкомпьютерные технологии

Название команды:

"Supertos Industries"

Оглавление

Введение	3
Набор фотографий Прокудина-Горского.....	4
Датасет FLICKR30K	9
Итоговый Датасет	16
Датасет с размерами 256x256 пикселей.....	16
Датасет размером 128x128 пикселей	19
Заключение	22
Приложение	23
Импорты и константы.....	23
Построение распределение размеров фотографий.....	23
Отсев фотографий малого размера	24
Отображение распределения среднего значения в канале и его стандартного отклонения	24
Отбор фотографий по стандартному отклонению в каналах	26
Удаление фотографий после отбора.....	26
Распределения средних значений разных каналов	27
Коэффициент корреляции между средними значениями канала	28
Распределения среднеквадратичных отклонений.....	29
Корреляции среднеквадратичных распределений разных каналов	29
Нормировка размеров	31
Гистограммы количества по разным характеристикам каналов для итоговых датасетов	31
Отделение L канала	32
Добавление шума к фотографии	33

Введение

Для решения задачи колоризации изображений и удаления шумов с помощью нейронных сетей было необходимо собрать датасет из изображений. Для первой нейронной сети необходимы пары цветных и соответствующих им черно-белых изображений, для второй – пары черно-белых изображений с шумом и без шума.

Ниже приведен разведочный анализ данных изображений датасета. Сам датасет был составлен из набора фотографий Flickr30k и фотографий, сделанных Прокудиным-Горским. Сначала описан анализ составляющих его наборов, а затем датасета в целом, и методы его обработки.

После создания полного набора цветных фотографий, было произведено отделение L канала в формате Lab с последующим его сохранением в отдельном изображении, для создания соответствующего набора черно-белых фотографий. Данные черно-белые фотографии вместе с цветными используются для обучения с учителем нейронной сети для задач колоризации, а также с помощью них был создан набор черно белых фотографий с шумом для обучения нейросети удаляющей шумы.

Весь используемый Python код для обработки данных приведен в разделе «Приложение», так же он представлен в соответствующим разделе проекта на GitHub.

Набор фотографий Прокудина-Горского

Набор данных собран из коллекции оцифрованных фотографий в формате JPG, сделанных Сергеем Михайловичем Прокудиным-Горским в 1900-1910х годах и расположенных на сайте архива конгресса США (<https://www.loc.gov/collections/prokudin-gorskii/>). Часть фотографий на данном ресурсе являются черно-белыми и не подходят для обучения с учителем, а часть имеет плохое качество.

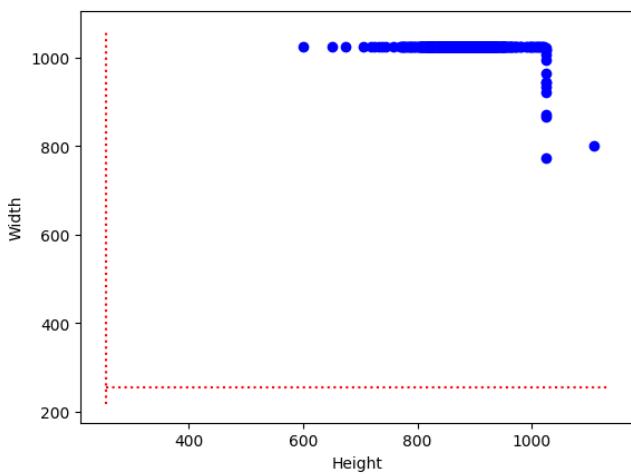


Так же в наборе фотографий имеются повторы с разным качеством изображений:

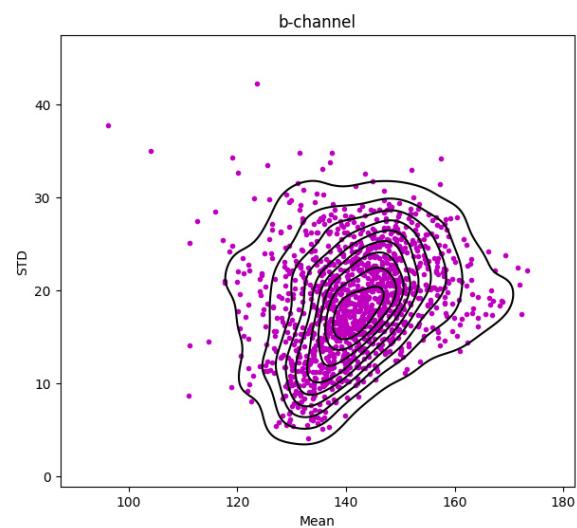
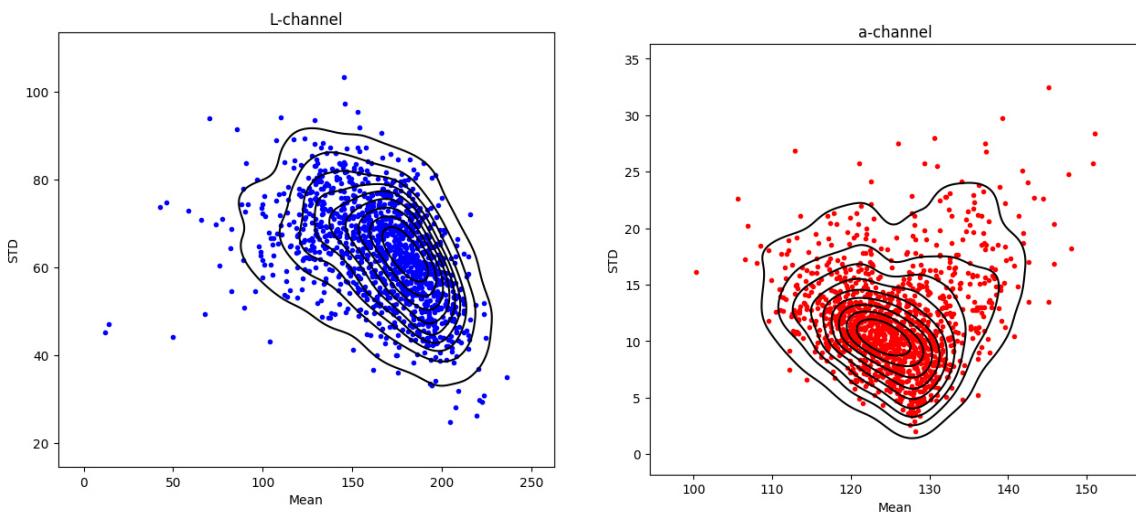


Подобного рода дефекты и повторы были удалены вручную.

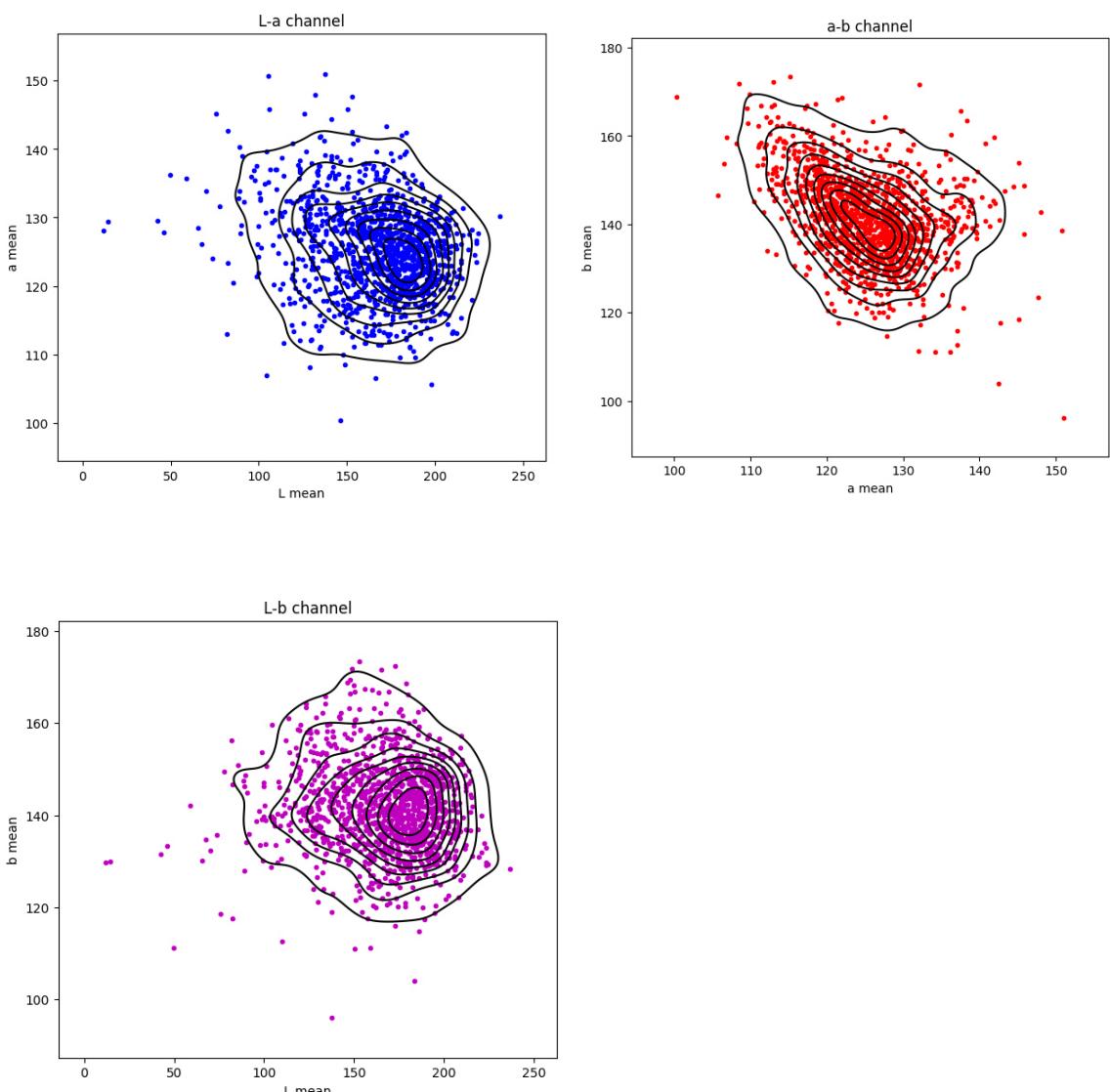
Ниже приведено распределение размеров изображений, где вертикальная ось отображает ширину в пикселях, а горизонтальная – высоту. Красным пунктиром отмечена граница размеров изображений имеющий по любому из направлений меньше 256 пикселей, такие изображения не войдут в итоговый датасет, чтобы избежать экстраполяции цветов. Как видно из графика, в данном наборе фотографий таких изображений нет.



Ниже приведено распределение фотографий по цветовым каналам, где по горизонтальной оси откладывается среднее арифметическое значение канала для всех пикселей одного изображения, а по вертикальной оси среднеквадратичное отклонение значений канала в пикселях от среднего для одного изображения. Чёрным выделены линии плотности.

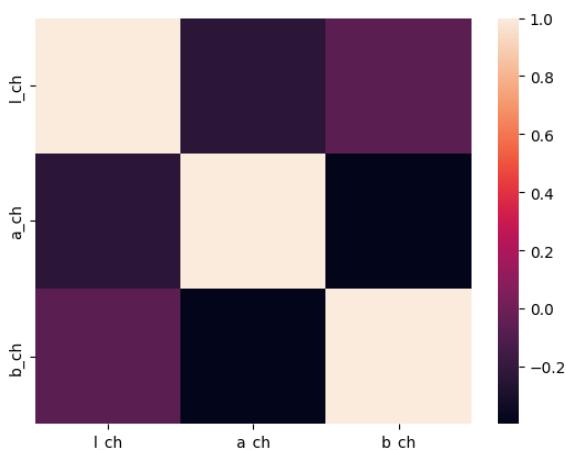


Ниже приведены распределения среднего арифметического значения одного канала относительно другого для каждого изображения.

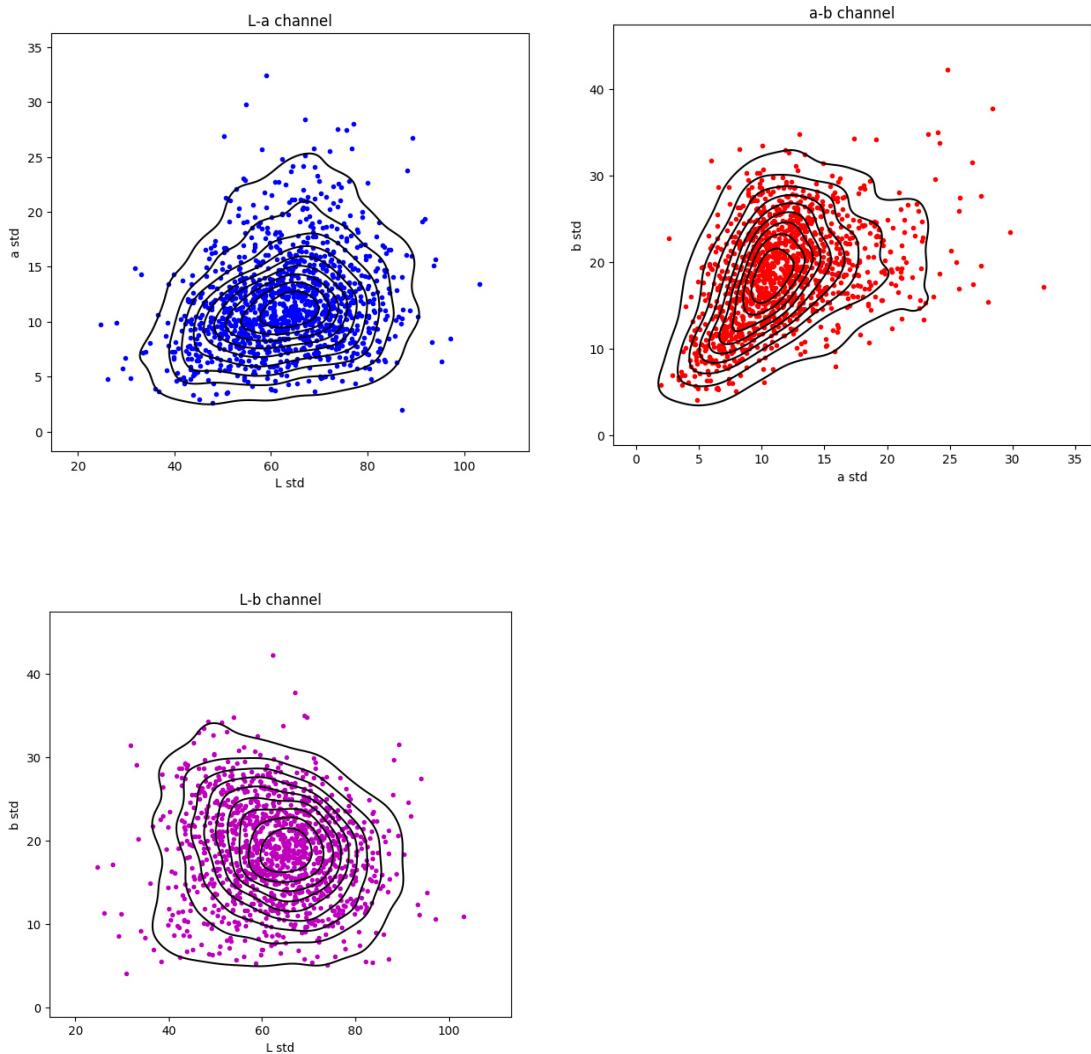


Ниже приведена матрица корреляций этих значений и тепловая карта корреляций. Сильной корреляции нет.

	L канал	а канал	б канал
L канал	1	-0.237196	-0.067729
а канал	-0.237196	1	-0.397433
б канал	-0.067729	-0.397433	1

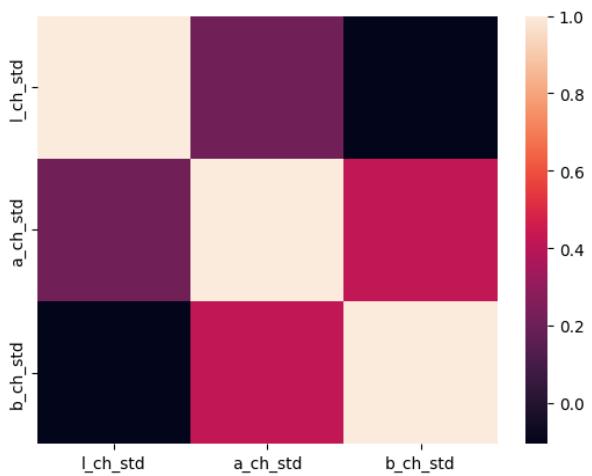


Ниже приведены распределения значения стандартного отклонения цвета одного канала относительно другого для каждого изображения.



Ниже приведена матрица корреляций этих значений и тепловая карта корреляций. Значения корреляции среднеквадратичных отклонений каналов a и b наибольшие, но все равно достаточно малы.

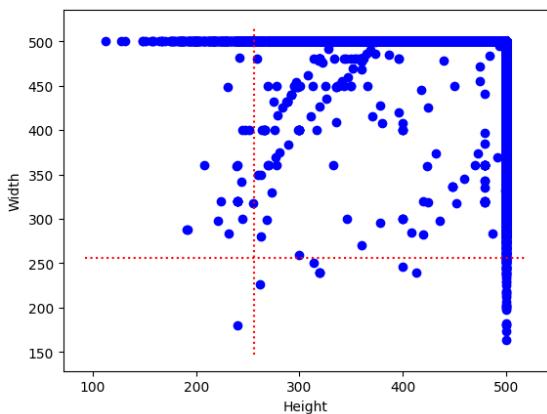
	L std	a std	b std
L std	1	0.214279	-0.105692
a std	0.214279	1	0.420147
b std	-0.105692	0.420147	1



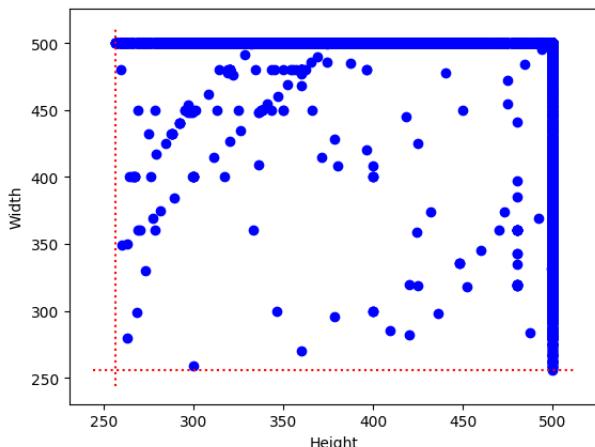
Датасет FLICKR30K

До обработки содержит 31783 фотографии в формате JPG.

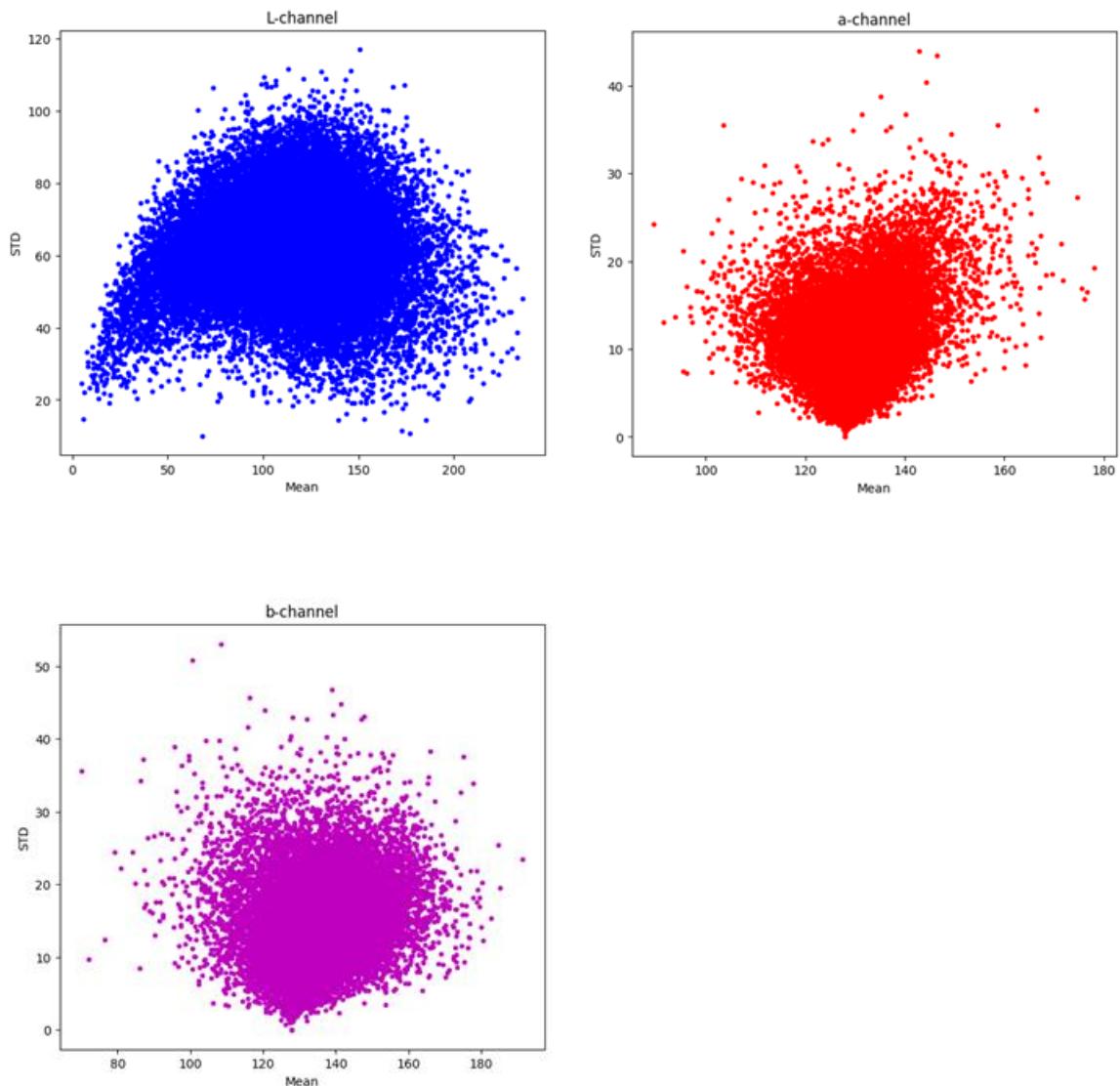
Ниже приведено распределение размеров изображений, где вертикальная ось отображает ширину в пикселях, а горизонтальная – высоту. Красным пунктиром отмечена граница размеров изображений имеющий по любому из направлений меньше 256 пикселей, такие изображения не войдут в итоговый датасет, чтобы избежать экстраполяции цветов. Есть несколько таких изображений.



Отсеивание изображений происходит с помощью Python скрипта, после отсеивания размеров изображений остается 31461 штука.



Ниже приведено распределение фотографий по цветовым каналам, где по горизонтальной оси откладывается среднее арифметическое значение канала для всех пикселей одного изображения, а по вертикальной оси среднеквадратичное отклонение значений канала в пикселях от среднего для одного изображения.



В данном датасете есть некоторые фотографии с низкой цветовой контрастностью, с «засветами» и с использованием фильтров обесцвечивающими окружение.



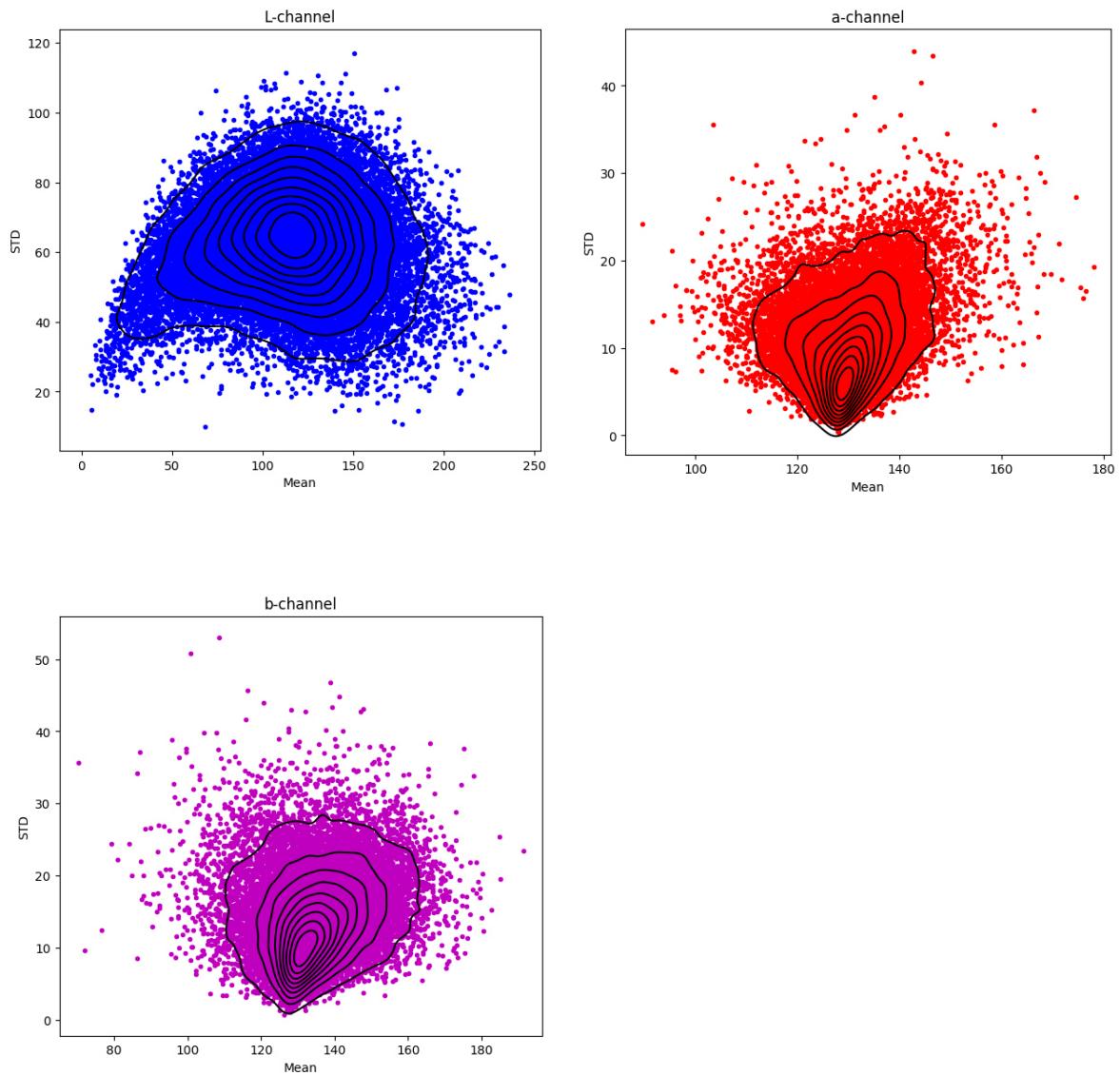


Данные фотографии можно отсеять, введя отбор по среднеквадратичному отклонению значений цветового канала среди пикселей относительно среднего значения цветового канала по пикселям для каждого изображения. При $\text{std} < 1$ выдается 14 фотографий, несколько из которых являются цветными (как правило это изображения в снегу).

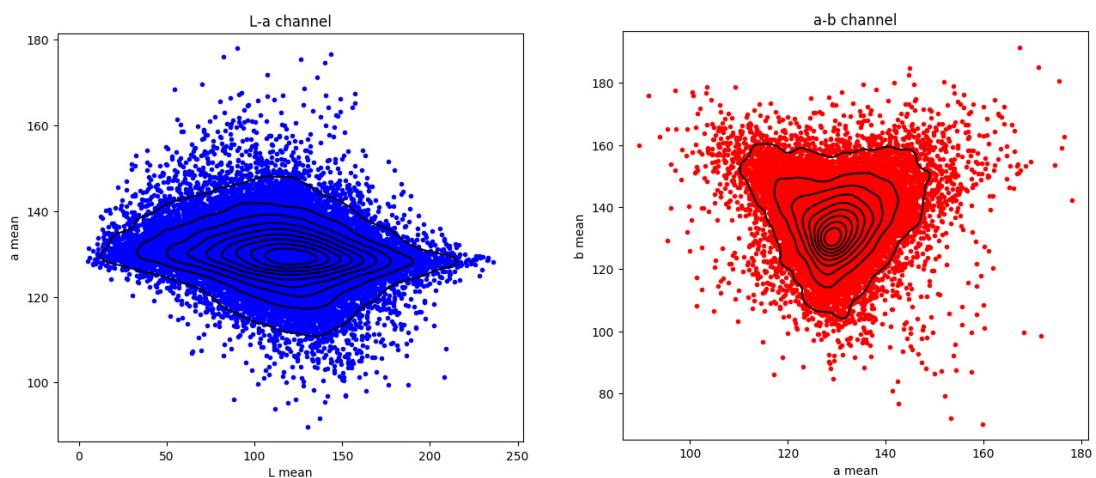


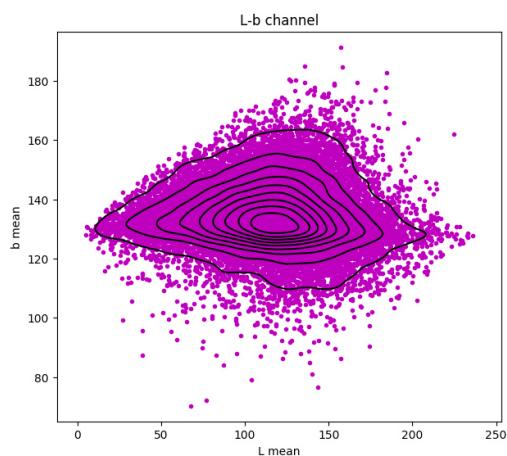
Вручную будет проведен отбор этих фотографий, после которого останется 6 фотографий, которые будут удалены из датасета.

Ниже приведено распределение фотографий по цветовым каналам, после удаления фотографий. Черным выделены линии плотности. На них видно, что нижние «хвосты» распределений на а и в канале были немного обрезаны.



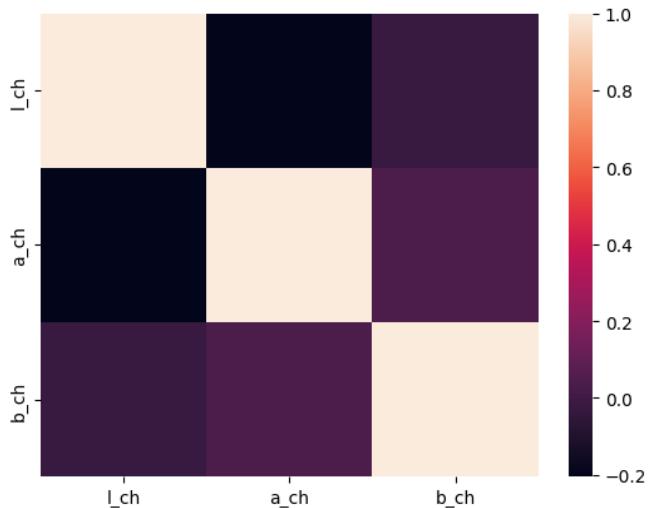
Ниже приведены распределения среднего арифметического значения одного канала относительно другого для каждого изображения.



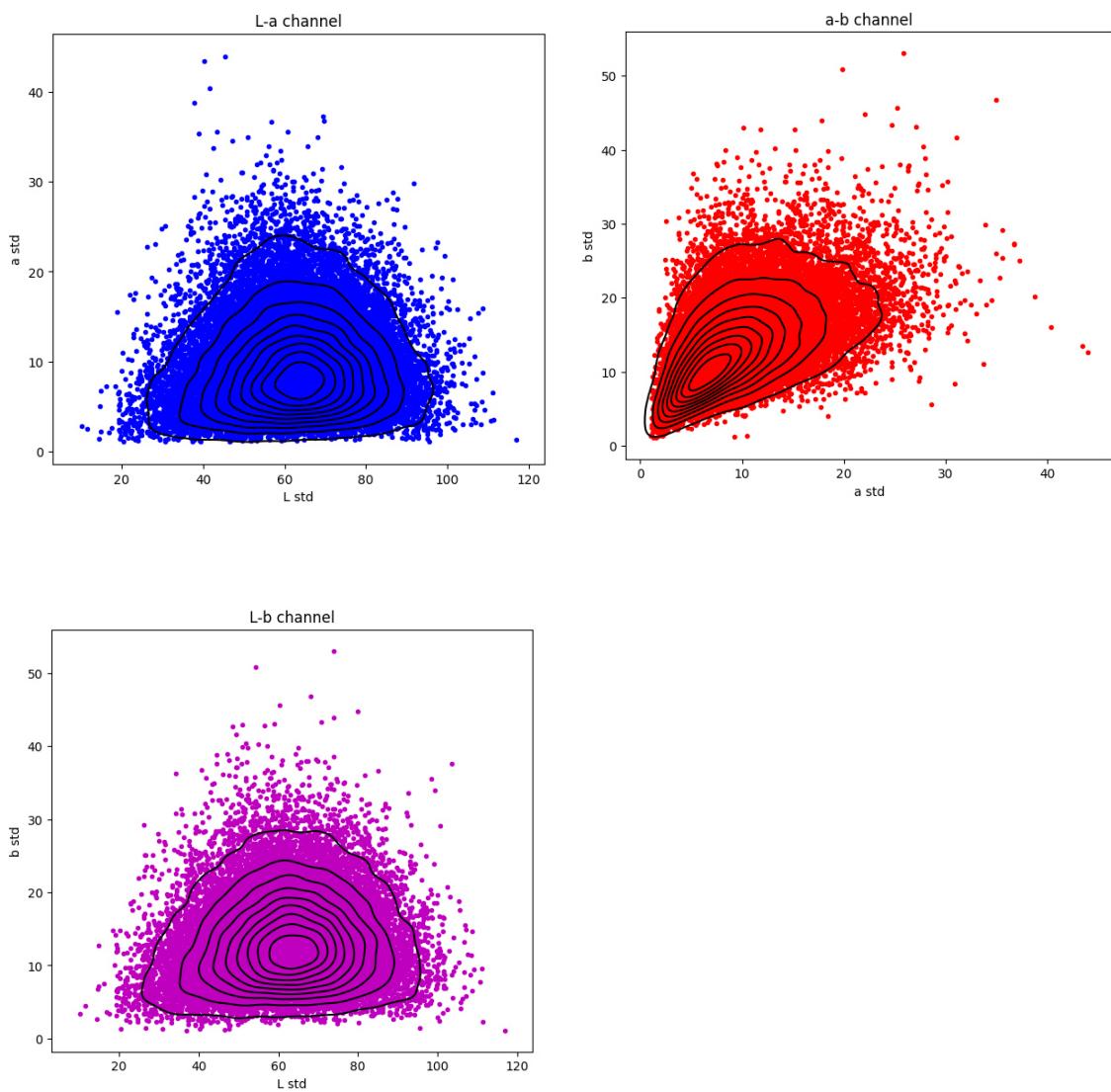


Ниже приведена матрица корреляций этих значений и тепловая карта корреляций. Сильной корреляции нет.

	L канал	a канал	b канал
L канал	1	-0.204174	-0.024990
a канал	-0.204174	1	0.039535
b канал	-0.024990	0.039535	1

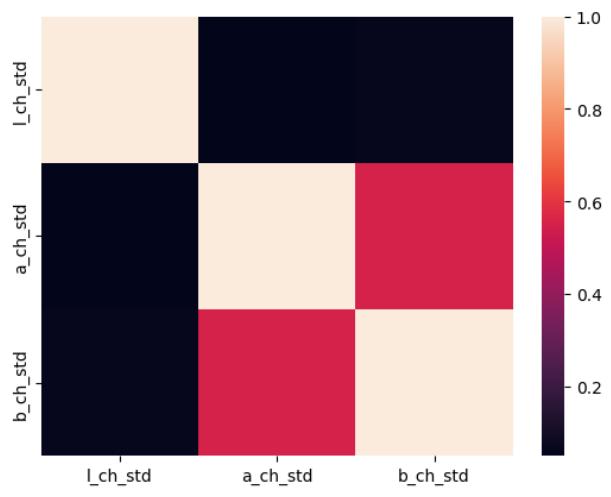


Ниже приведены распределения значения стандартного отклонения цвета одного канала относительно другого для каждого изображения.



Ниже приведена матрица корреляций этих значений и тепловая карта корреляций. Значения корреляции среднеквадратичных отклонений каналов *a* и *b* наибольшие, но все равно достаточно малы.

	L std	a std	b std
L std	1	0.050021	0.063020
a std	0.050021	1	0.553947
b std	0.063020	0.553947	1

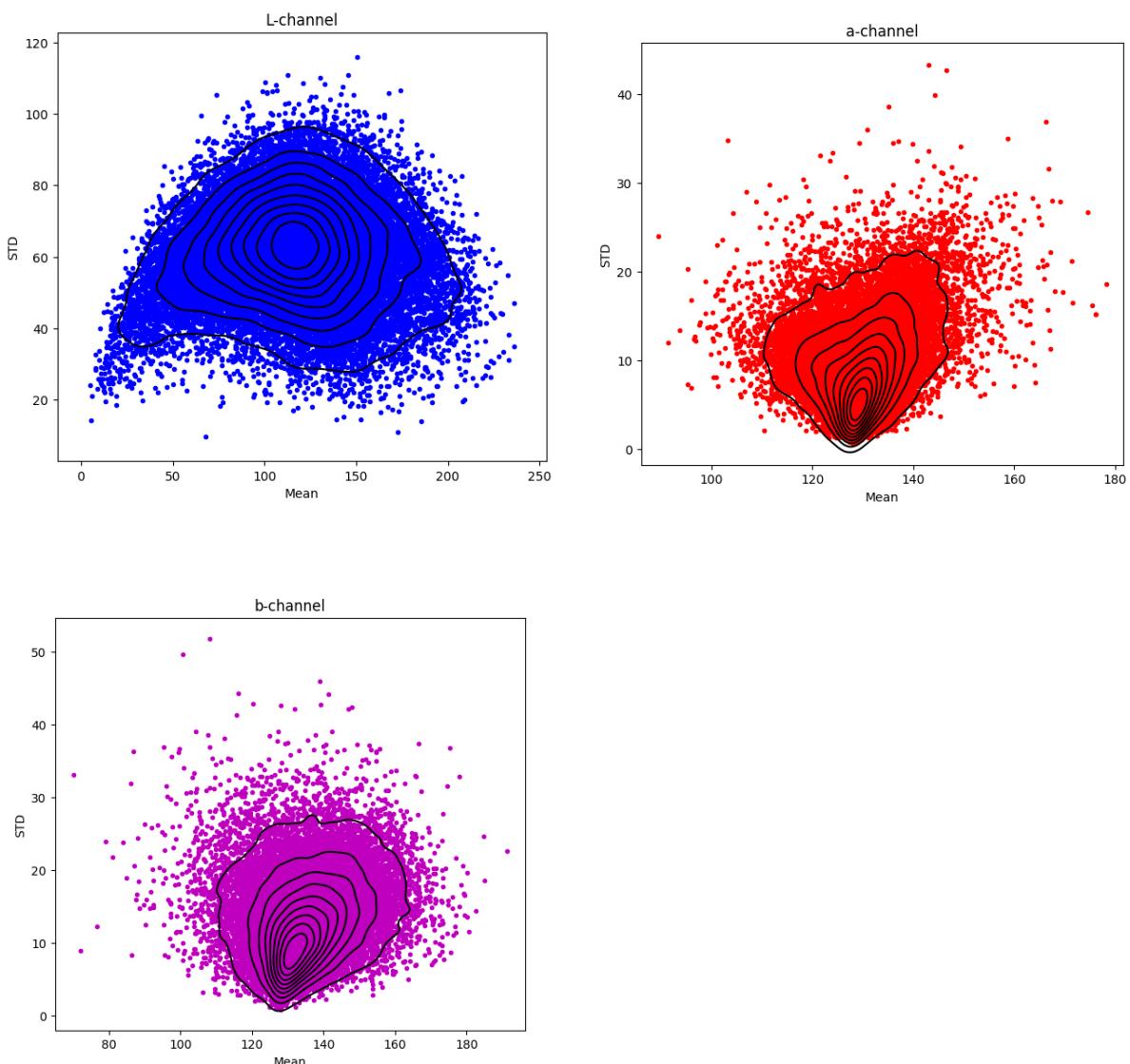


Итоговый Датасет

Итоговый датасет получается путем слияния двух предыдущих наборов фотографий и нормировки их размерностей. Размерностями выбрано 256x256 пикселей и 128x128 пикселей. Для последней размерности есть обученная нейронная сеть с архитектурой U-net AE-CNN для колоризации.

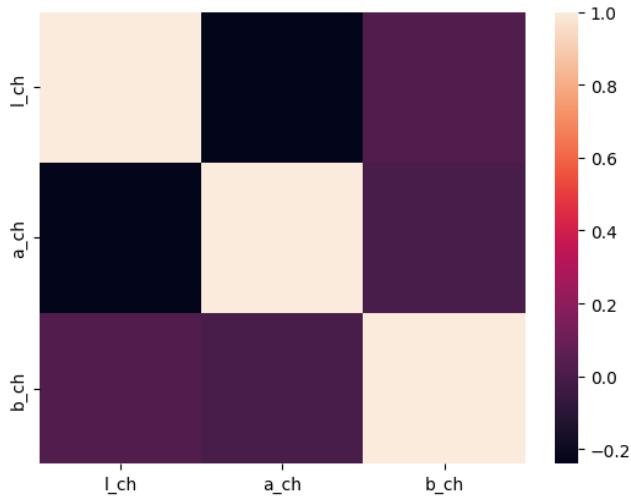
Датасет с размерами 256x256 пикселей

Ниже приведено распределение фотографий по цветовым каналам, где по горизонтальной оси откладывается среднее арифметическое значение канала для всех пикселей одного изображения, а по вертикальной оси среднеквадратичное отклонение значений канала в пикселях от среднего для одного изображения. Чёрным выделены линии плотности.



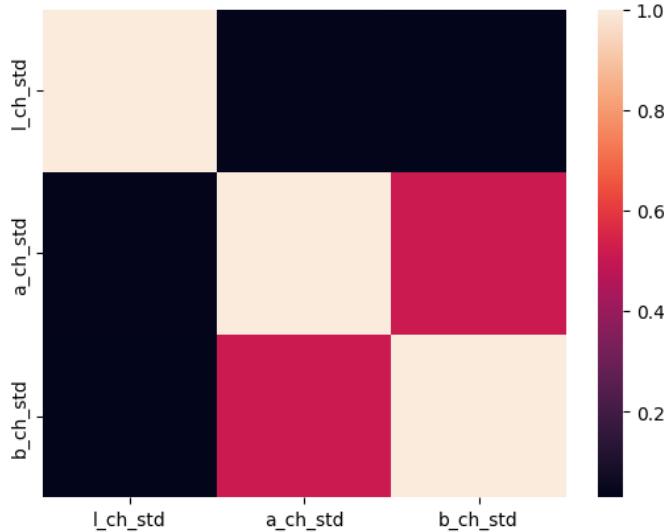
Ниже приведена матрица корреляций распределения среднего арифметического значения одного канала относительно другого и тепловая карта корреляций. Сильной корреляции нет.

	L канал	а канал	б канал
L канал	1	-0.239314	0.018946
а канал	-0.239314	1	-0.001842
б канал	0.018946	-0.001842	1

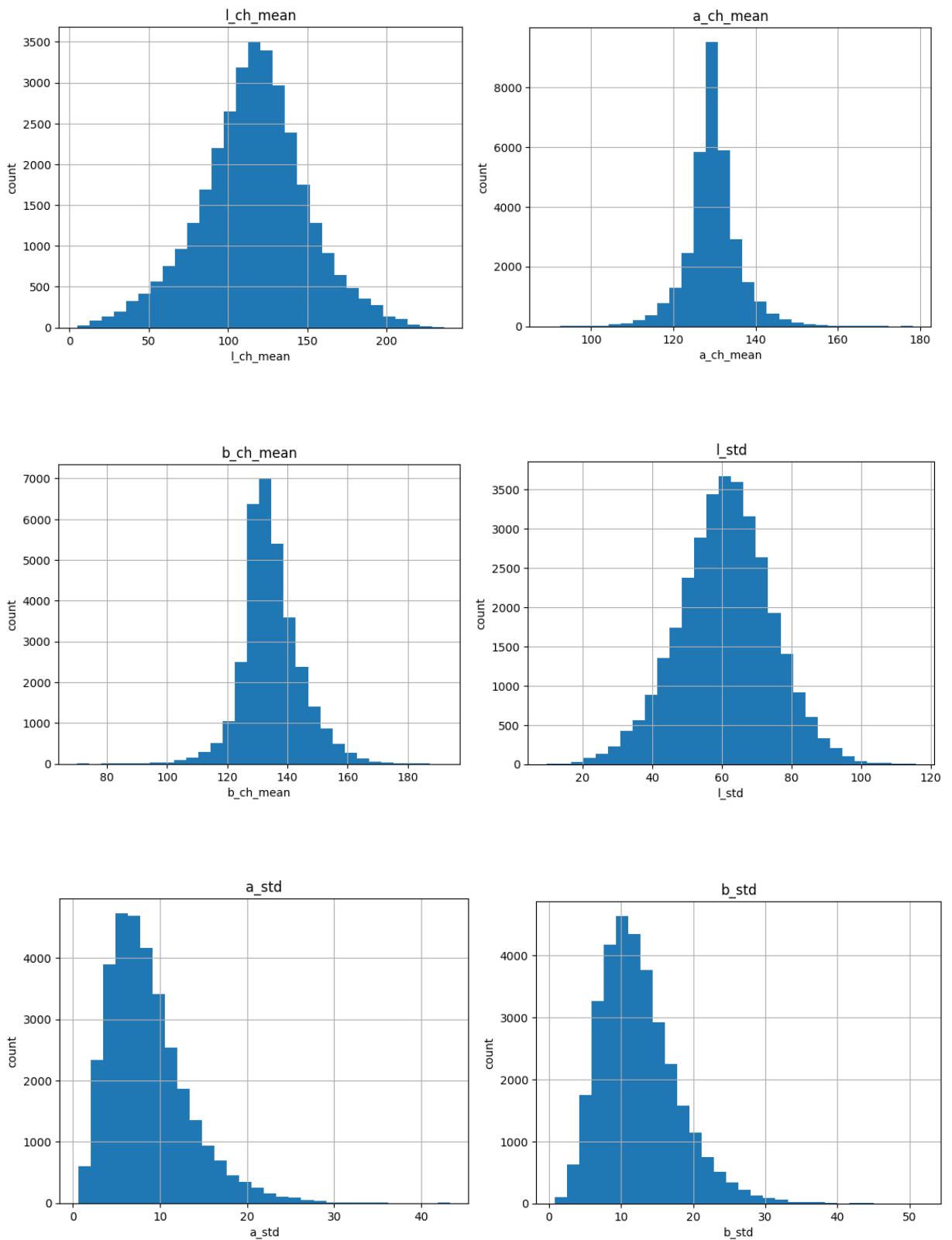


Ниже приведена матрица корреляций распределения значений стандартного отклонения цвета одного канала относительно другого и тепловая карта корреляций. Значения корреляции среднеквадратичных отклонений каналов *a* и *b* наибольшие, но все равно достаточно малы.

	L std	a std	b std
L std	1	0.034104	0.028978
a std	0.034104	1	0.514171
b std	0.028978	0.514171	1

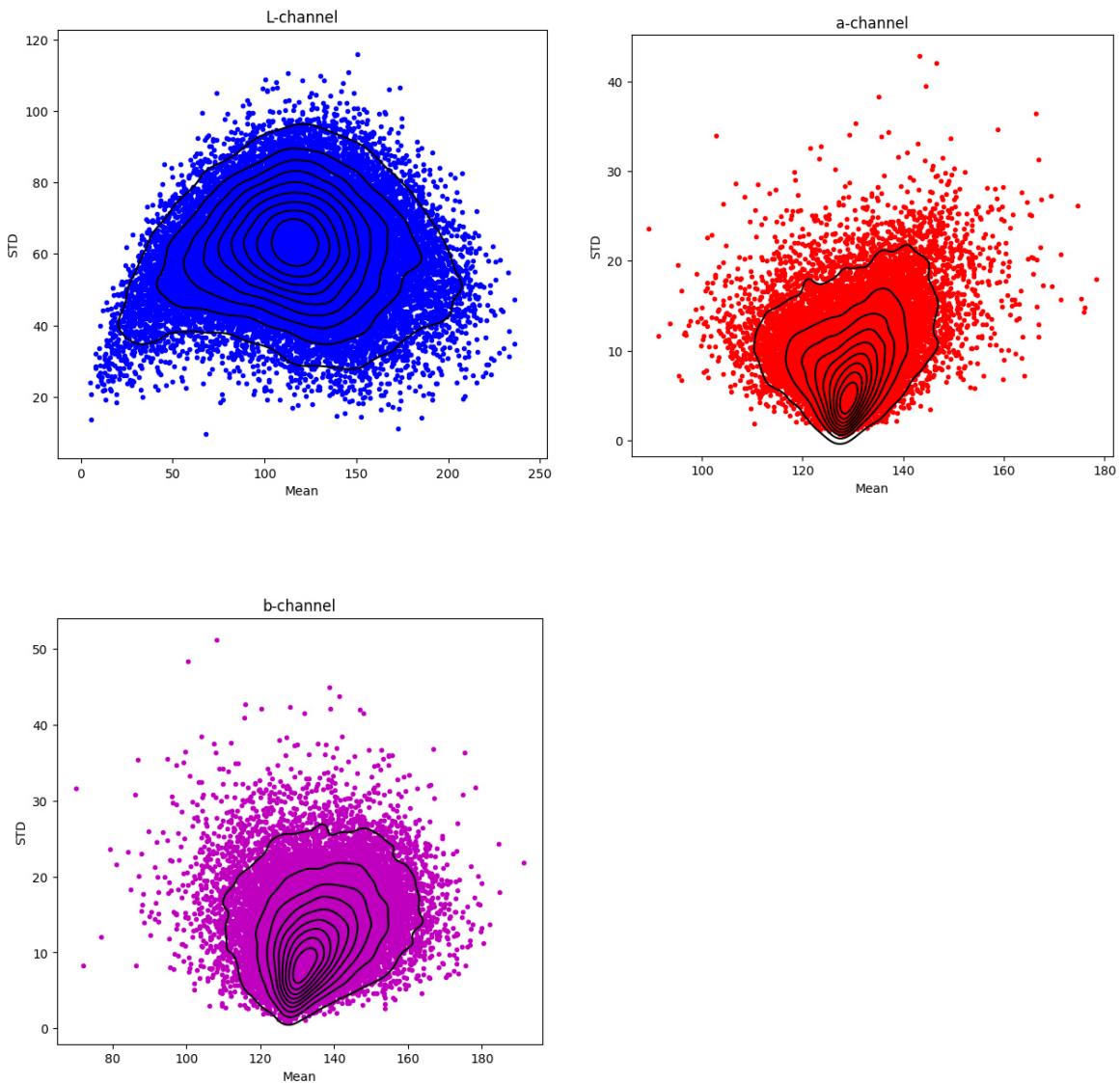


Ниже приведены гистограммы количества изображений, сгруппированных по средним арифметическим значениям по каналам и среднеквадратичным отклонениям по каналам.



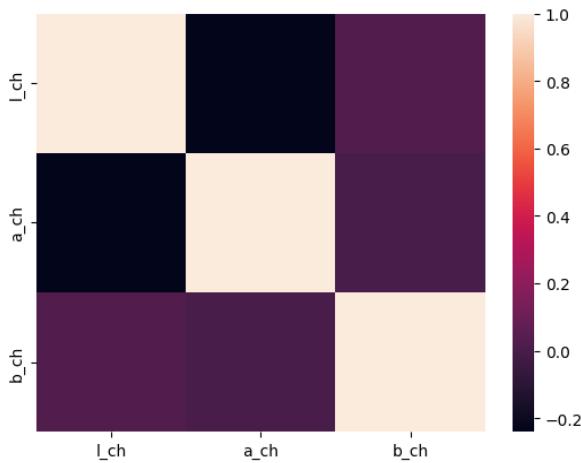
Датасет размером 128x128 пикселей

Ниже приведено распределение фотографий по цветовым каналам, где по горизонтальной оси откладывается среднее арифметическое значение канала для всех пикселей одного изображения, а по вертикальной оси среднеквадратичное отклонение значений канала в пикселях от среднего для одного изображения. Чёрным выделены линии плотности.



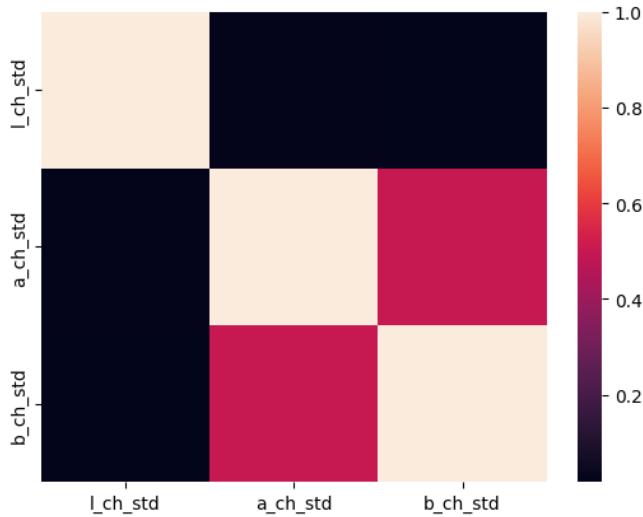
Ниже приведена матрица корреляций распределения среднего арифметического значения одного канала относительно другого и тепловая карта корреляции. Сильной корреляции нет.

	L канал	a канал	b канал
L канал	1	-0.240345	0.020958
a канал	-0.240345	1	-0.002667
b канал	0.020958	-0.002667	1

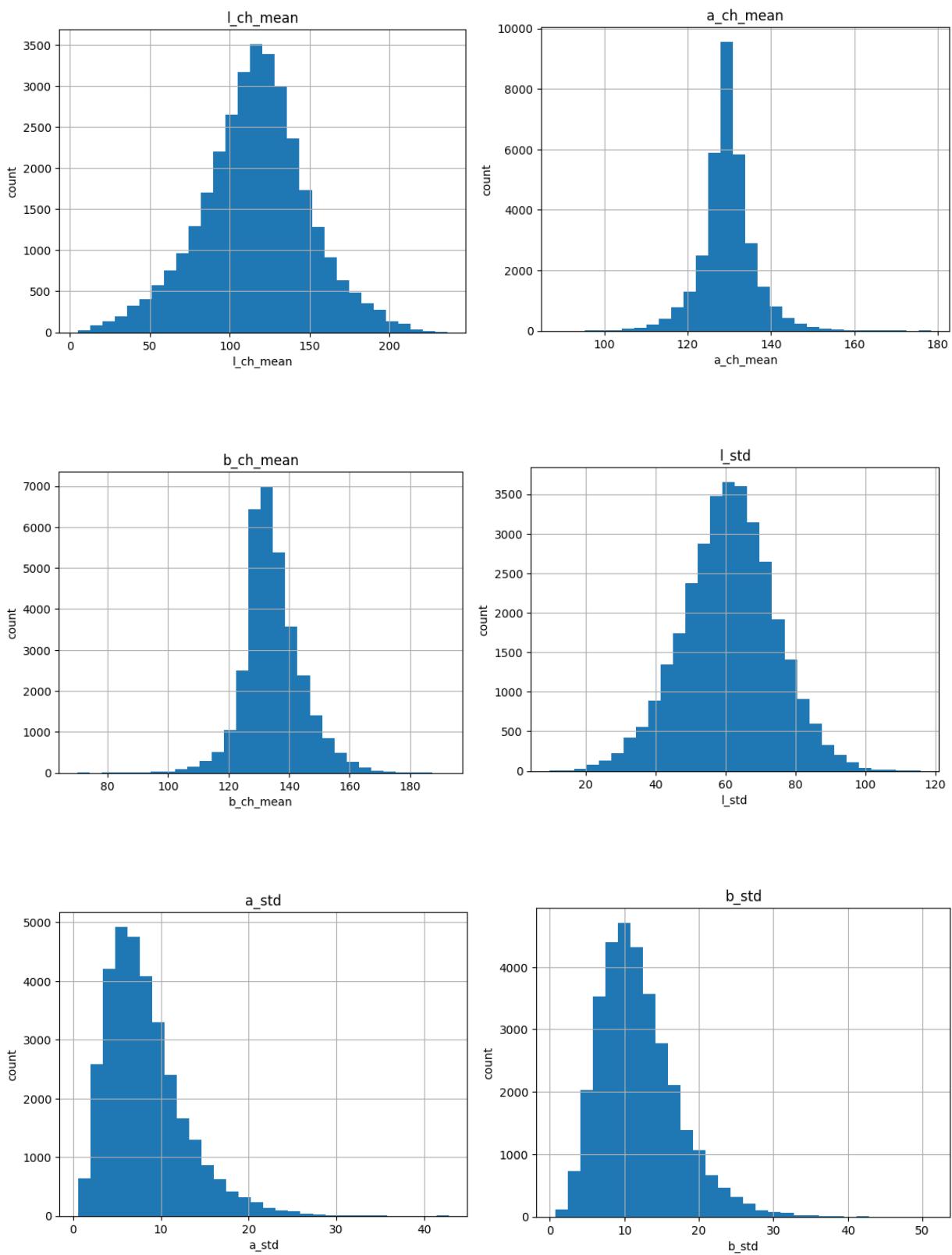


Ниже приведена матрица корреляций этих значений и тепловая карта корреляции. Значения корреляции среднеквадратичных отклонений каналов *a* и *b* наибольшие, но все равно достаточно малы.

	L std	a std	b std
L std	1	0.026064	0.018500
a std	0.026064	1	0.500508
b std	0.018500	0.500508	1



Ниже приведены гистограммы количества изображений, сгруппированных по средним арифметическим значениям по каналам и среднеквадратичным отклонениям по каналам.



Заключение

Видимых зависимостей между цветовыми каналами в формате Lab не обнаружено. Есть слабая корреляция между стандартными отклонениями в а и b каналах. Предположительно имеется зависимость между объектами, расположенными на фотографиях и цветом, но данный аспект не исследовался поскольку на вход нейросети не будет подаваться описание фотографии.

Далее было проведено разделение L канала от изображения и его сохранение, для создания черно-белого фото используемого в датасете колоризации. Так же на его основе создавался зашумленный вариант изображения для датасета подавляющей шум нейросети. Код соответствующий данным операциям описан в приложении.

Приложение

Импорты и константы

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import os
from typing import Tuple

MIRFLICKR_PATH = 'MIRFLICKR'
PROKUDIN_PATH = 'prokudin_gorskii_images'
FLICKR30_PATH = 'FLICKR30'
PATH_SAVE = 'SP'
FINAL_DATASET_256 = 'DATASET_256'
FINAL_DATASET_128 = 'DATASET_128'
L_DATASET_256 = 'L_DATASET_256'
L_DATASET_128 = 'L_DATASET_128'

OUT_SIZE_256 = (256, 256)
OUT_SIZE_128 = (128, 128)
MIN_IMG_SIZE = 256
```

Построение распределение размеров фотографий

```
def size_graph(path: str):
    names = [os.path.join(path, i) for i in os.listdir(path)]
    fig, ax = plt.subplots()
    ax.set_xlabel('Height')
    ax.set_ylabel('Width')
```

```

for name in names:
    img = cv2.imread(name, cv2.IMREAD_COLOR)
    ax.plot(img.shape[0], img.shape[1], marker='o', color='b', fillstyle = 'full')
    ax.plot([min(min(ax.axes.get_xlim()), 256), max(max(ax.axes.get_xlim()), 256)], [256, 256], ls =':', color = 'r')
    ax.plot([256, 256], [min(min(ax.axes.get_ylim()), 256), max(max(ax.axes.get_ylim()), 256)], ls =':', color = 'r')

size_graph(FLICKR30_PATH)
size_graph(PROKUDIN_PATH)

```

Отсев фотографий малого размера

```

def size_selection(path: str, min_size: int) -> None:
    names = [path + '\\\\' + i for i in os.listdir(path)]
    for name in names:
        img = cv2.imread(name, cv2.IMREAD_COLOR)
        if img.shape[0] < min_size or img.shape[1] < min_size:
            os.remove(name)

size_selection(FLICKR30_PATH)

```

Отображение распределения среднего значения в канале и его стандартного отклонения

```

def color_map(path:str):
    names = [os.path.join(path, i) for i in os.listdir(path)]
    fig, axis = plt.subplots(3, figsize=(7, 21))

    axis[0].set_title('L-channel')
    axis[1].set_title('a-channel')
    axis[2].set_title('b-channel')

```

```

axis[0].set_xlabel('Mean')
axis[0].set_ylabel('STD')
axis[1].set_xlabel('Mean')
axis[1].set_ylabel('STD')
axis[2].set_xlabel('Mean')
axis[2].set_ylabel('STD')

l_ch_ = []
a_ch_ = []
b_ch_ = []
l_ch_std = []
a_ch_std = []
b_ch_std = []

for name in names:
    img = cv2.imread(name, cv2.IMREAD_COLOR)
    img_lab = np.array(cv2.cvtColor(img, cv2.COLOR_BGR2LAB))
    df_ch = pd.DataFrame({'l_ch': img_lab[:, :, 0].flatten().flatten(),
                          'a_ch' : img_lab[:, :, 1].flatten().flatten(),
                          'b_ch' : img_lab[:, :, 2].flatten().flatten()})
    l_ch_.append(df_ch['l_ch'].mean())
    a_ch_.append(df_ch['a_ch'].mean())
    b_ch_.append(df_ch['b_ch'].mean())
    l_ch_std.append(df_ch['l_ch'].std())
    a_ch_std.append(df_ch['a_ch'].std())
    b_ch_std.append(df_ch['b_ch'].std())

df_chs = pd.DataFrame({'l_ch': l_ch_, 'a_ch': a_ch_, 'b_ch': b_ch_,
                       'l_std':l_ch_std,
                       'a_std': a_ch_std, 'b_std': b_ch_std})

axis[0].scatter(l_ch_, l_ch_std, marker = '.', color = 'b')
axis[1].scatter(a_ch_, a_ch_std, marker = '.', color = 'r')
axis[2].scatter(b_ch_, b_ch_std, marker = '.', color = 'm')

```

```

sns.kdeplot(df_chs, x='l_ch', y='l_std', ax=axis[0], color='k')
sns.kdeplot(df_chs, x='a_ch', y='a_std', ax=axis[1], color='k')
sns.kdeplot(df_chs, x='b_ch', y='b_std', ax=axis[2], color='k')

color_map(FLICKR30_PATH)

color_map(PROKUDIN_PATH)

```

Отбор фотографий по стандартному отклонению в каналах

```

def color_select(path:str, path_save: str):
    names = [os.path.join(path, i) for i in os.listdir(path)]

    for name in names:
        img = cv2.imread(name, cv2.IMREAD_COLOR)
        img_lab = np.array(cv2.cvtColor(img, cv2.COLOR_BGR2LAB))
        df_ch = pd.DataFrame({'l_ch': img_lab[:, :, 0].flatten().flatten(),
                             'a_ch' : img_lab[:, :, 1].flatten().flatten(),
                             'b_ch' : img_lab[:, :, 2].flatten().flatten()})
        if df_ch['a_ch'].std() < 1 or df_ch['b_ch'].std() < 1:
            cv2.imwrite(os.path.join(path_save, os.path.split(name)[-1]),
                       img)

color_select(FLICKR30_PATH, PATH_SAVE)

```

Удаление фотографий после отбора

```

names = [os.path.join(PATH_SAVE, i) for i in os.listdir(PATH_SAVE)]
for name in names:
    img = cv2.imread(name, cv2.IMREAD_COLOR)
    img_lab = np.array(cv2.cvtColor(img, cv2.COLOR_BGR2LAB))
    try:
        os.remove(FLICKR30_PATH + '\\\\' + name.split('\\\\')[-1])
    except:

```

...

Распределения средних значений разных каналов

```
def corr_channels_map(path:str):
    names = [os.path.join(path, i) for i in os.listdir(path)]
    fig, axis = plt.subplots(3, figsize=(7, 21))

    axis[0].set_title('L-a channel')
    axis[1].set_title('a-b channel')
    axis[2].set_title('L-b channel')

    axis[0].set_xlabel('L mean')
    axis[0].set_ylabel('a mean')
    axis[1].set_xlabel('a mean')
    axis[1].set_ylabel('b mean')
    axis[2].set_xlabel('L mean')
    axis[2].set_ylabel('b mean')

    l_ch_ = []
    a_ch_ = []
    b_ch_ = []

    for name in names:
        img = cv2.imread(name, cv2.IMREAD_COLOR)
        img_lab = np.array(cv2.cvtColor(img, cv2.COLOR_BGR2LAB))
        df_ch = pd.DataFrame({'l_ch': img_lab[:, :, 0].flatten().flatten(),
                             'a_ch' : img_lab[:, :, 1].flatten().flatten(),
                             'b_ch' : img_lab[:, :, 2].flatten().flatten()})
        l_ch_.append(df_ch['l_ch'].mean())
        a_ch_.append(df_ch['a_ch'].mean()))
        b_ch_.append(df_ch['b_ch'].mean()))
```

```

df_chs = pd.DataFrame({'l_ch': l_ch_, 'a_ch': a_ch_, 'b_ch': b_ch_})
axis[0].scatter(l_ch_, a_ch_, marker = '.', color = 'b')
axis[1].scatter(a_ch_, b_ch_, marker = '.', color = 'r')
axis[2].scatter(l_ch_, b_ch_, marker = '.', color = 'm')
sns.kdeplot(df_chs, x='l_ch', y='a_ch', ax=axis[0], color='k')
sns.kdeplot(df_chs, x='a_ch', y='b_ch', ax=axis[1], color='k')
sns.kdeplot(df_chs, x='l_ch', y='b_ch', ax=axis[2], color='k')

corr_channels_map(PROKUDIN_PATH)
corr_channels_map(FLICKR30_PATH)

```

Коэффициент корреляции между средними значениями канала

```

def corr_channels(path:str):
    names = [os.path.join(path, i) for i in os.listdir(path)]

    l_ch_ = []
    a_ch_ = []
    b_ch_ = []

    for name in names:
        img = cv2.imread(name, cv2.IMREAD_COLOR)
        img_lab = np.array(cv2.cvtColor(img, cv2.COLOR_BGR2LAB))
        df_ch = pd.DataFrame({'l_ch': img_lab[:, :, 0].flatten().flatten(),
                             'a_ch' : img_lab[:, :, 1].flatten().flatten(),
                             'b_ch' : img_lab[:, :, 2].flatten().flatten()})
        l_ch_.append(df_ch['l_ch'].mean())
        a_ch_.append(df_ch['a_ch'].mean())
        b_ch_.append(df_ch['b_ch'].mean())

    df_chs = pd.DataFrame({'l_ch': l_ch_, 'a_ch': a_ch_, 'b_ch': b_ch_})
    corr = df_chs.corr()
    print(corr)
    sns.heatmap(corr)

```

```
corr_channels(PROKUDIN_PATH)
corr_channels(FLICKR30_PATH)
```

Распределения среднеквадратичных отклонений

```
def corr_std_channels(path:str):
    names = [os.path.join(path, i) for i in os.listdir(path)]

    l_ch_ = []
    a_ch_ = []
    b_ch_ = []

    for name in names:
        img = cv2.imread(name, cv2.IMREAD_COLOR)
        img_lab = np.array(cv2.cvtColor(img, cv2.COLOR_BGR2LAB))
        df_ch = pd.DataFrame({'l_ch': img_lab[:, :, 0].flatten().flatten(),
                             'a_ch' : img_lab[:, :, 1].flatten().flatten(),
                             'b_ch' : img_lab[:, :, 2].flatten().flatten()})
        l_ch_.append(df_ch['l_ch'].std())
        a_ch_.append(df_ch['a_ch'].std())
        b_ch_.append(df_ch['b_ch'].std())

    df_chs = pd.DataFrame({'l_ch_std': l_ch_, 'a_ch_std': a_ch_, 'b_ch_std': b_ch_})
    corr = df_chs.corr()
    print(corr)
    sns.heatmap(corr)

corr_std_channels(PROKUDIN_PATH)
corr_std_channels(FLICKR30_PATH)
```

Корреляции среднеквадратичных распределений разных каналов

```
def corr_std_channels_map(path:str):
    names = [os.path.join(path, i) for i in os.listdir(path)]
    fig, axis = plt.subplots(3, figsize=(7, 21))

    axis[0].set_title('L-a channel')
```

```

axis[1].set_title('a-b channel')
axis[2].set_title('L-b channel')

axis[0].set_xlabel('L std')
axis[0].set_ylabel('a std')

axis[1].set_xlabel('a std')
axis[1].set_ylabel('b std')

axis[2].set_xlabel('L std')
axis[2].set_ylabel('b std')

l_ch_std = []
a_ch_std = []
b_ch_std = []

for name in names:
    img = cv2.imread(name, cv2.IMREAD_COLOR)
    img_lab = np.array(cv2.cvtColor(img, cv2.COLOR_BGR2LAB))
    df_ch = pd.DataFrame({'l_ch': img_lab[:, :, 0].flatten().flatten(),
                          'a_ch' : img_lab[:, :, 1].flatten().flatten(),
                          'b_ch' : img_lab[:, :, 2].flatten().flatten()})

    l_ch_std.append(df_ch['l_ch'].std())
    a_ch_std.append(df_ch['a_ch'].std())
    b_ch_std.append(df_ch['b_ch'].std())

df_chs = pd.DataFrame({'l_std':l_ch_std,
                      'a_std': a_ch_std, 'b_std': b_ch_std})
axis[0].scatter(l_ch_std, a_ch_std, marker = '.', color = 'b')
axis[1].scatter(a_ch_std, b_ch_std, marker = '.', color = 'r')
axis[2].scatter(l_ch_std, b_ch_std, marker = '.', color = 'm')
sns.kdeplot(df_chs, x='l_std', y='a_std', ax=axis[0], color='k')

```

```
sns.kdeplot(df_chs, x='a_std', y='b_std', ax=axis[1], color='k')
sns.kdeplot(df_chs, x='l_std', y='b_std', ax=axis[2], color='k')

corr_std_channels_map(PROKUDIN_PATH)
corr_std_channels_map(FLICKR30_PATH)
```

Нормировка размеров

```
def downscaling(path: str, out_path: str, size: Tuple[int, int]):
    names = [os.path.join(path, i) for i in os.listdir(path)]
    for name in names:
        img = cv2.imread(name, cv2.IMREAD_COLOR)
        img = cv2.resize(img, size)
        cv2.imwrite(os.path.join(out_path, os.path.split(name)[-1]), img)
```

```
downscaling(PROKUDIN_PATH, FINAL_DATASET_256, OUT_SIZE_256)
downscaling(FLICKR30_PATH, FINAL_DATASET_256, OUT_SIZE_256)
downscaling(PROKUDIN_PATH, FINAL_DATASET_128, OUT_SIZE_128)
downscaling(FLICKR30_PATH, FINAL_DATASET_128, OUT_SIZE_128)
```

Гистограммы количества по разным характеристикам каналов для итоговых датасетов

```
def histogram(path: set):
    names = [os.path.join(path, i) for i in os.listdir(path)]
    l_ch_ = []
    a_ch_ = []
    b_ch_ = []
    l_ch_std = []
    a_ch_std = []
    b_ch_std = []
```

```

for name in names:

    img = cv2.imread(name, cv2.IMREAD_COLOR)

    img_lab = np.array(cv2.cvtColor(img, cv2.COLOR_BGR2LAB))

    df_ch = pd.DataFrame({'l_ch': img_lab[:, :, 0].flatten().flatten(),
                          'a_ch' : img_lab[:, :, 1].flatten().flatten(),
                          'b_ch' : img_lab[:, :, 2].flatten().flatten()})

    l_ch_.append(df_ch['l_ch'].mean())
    a_ch_.append(df_ch['a_ch'].mean())
    b_ch_.append(df_ch['b_ch'].mean())
    l_ch_std.append(df_ch['l_ch'].std())
    a_ch_std.append(df_ch['a_ch'].std())
    b_ch_std.append(df_ch['b_ch'].std())

df_chs = pd.DataFrame({'l_ch_mean': l_ch_, 'a_ch_mean': a_ch_,
                       'b_ch_mean': b_ch_, 'l_std':l_ch_std,
                       'a_std': a_ch_std, 'b_std': b_ch_std})

for i in df_chs.columns:

    axarr = df_chs.hist(column=i, bins=30)

    for ax in axarr.flatten():

        ax.set_xlabel(i)
        ax.set_ylabel("count")

histogram(FINAL_DATASET_256)
histogram(FINAL_DATASET_128)

```

Отделение L канала

```

def l_separator(path: str, out_path: str):

    names = [os.path.join(path, i) for i in os.listdir(path)]
    for name in names:

        img = cv2.imread(name, cv2.IMREAD_COLOR)
        img_l = np.array(cv2.cvtColor(img, cv2.COLOR_BGR2LAB))[:, :, 0]
        cv2.imwrite(os.path.join(out_path, os.path.split(name)[-1]), img_l)

```

```
l_separator(FINAL_DATASET_256, L_DATASET_256)
l_separator(FINAL_DATASET_128, L_DATASET_128)
```

Добавление шума к фотографии

```
def add_noise(images, noise_factor=0.1):
    noise = np.random.normal(loc=0.0, scale=noise_factor, size=images.shape)
    noisy_images = images + noise
    return np.clip(noisy_images, 0., 1.)
```