تمرین سری 1 بینایی ماشین **زهرا عبادی**

برای پیاده سازی سیستم بازیابی تصویر بر اساس ویژگی رنگ در دو فضا ی رنگی RGB و HSV :

ابتدا یک تابع برای محاسبه ی هیستوگرام در دو فضای رنگی RGB و HSV به نام calculate_histogram تعریف میکنیم :

این تابع دو پارامتر به عنوان ورودی میگیرد : image (نماینده تصویر ورودی) و color_space (تعیین فضای رنگی برای استفاده). تابع پارامتر color_space را بررسی می کند تا تعیین کند که آیا هیستوگرام در فضای رنگی RGB یا HSV محاسبه می شود . اگر (color_space = **'rgb'**) باشد، هیستوگرام را با استفاده از تابع cv2.calcHist در کانال های تصویر ورودی(کانال های BGRدر OpenCV) محاسبه می کند .اگر(color_space = **'hsv'**) باشد، تصویر را با استفاده از cv2.cvtColor به فضای رنگی HSV تبدیل می‌کند و سپس هیستوگرام را محاسبه می‌کند.

هیستوگرام محاسبه شده با استفاده از hist.flatten() مسطح می شود .این یک هیستوگرام چند بعدی را به یک آرایه یک بعدی تبدیل می کند .سپس هیستوگرام مسطح شده با تقسیم هر مقدار بر مجموع همه مقادیر موجود در هیستوگرام با استفاده از hist /= hist.sum() نرمال می شود.

سپس یک تابع برای مقایسه هیستوگرام های دو تصویر به نام compare_histograms تعریف میکنیم :

این تابع دو پارامتر به عنوان ورودی میگیرد :hist1 و hist2 که نشان دهنده هیستوگرام هایی است که باید مقایسه شوند. این تابع از تابع distance.euclidean از ماژول scipy.spatial برای محاسبه فاصله اقلیدسی بین دو هیستوگرام استفاده می کند .فاصله اقلیدسی اندازه گیری فاصله خط مستقیم بین دو نقطه در یک فضا است .در این زمینه، عدم تشابه بین دو هیستوگرام را اندازه گیری می کند.

تابع بعد را برای محاسبه ی امتیاز شباهت نرمال شده بر اساس فاصله اقلیدسی بین دو هیستوگرام به نام calculate_similarity_score تعریف میکنیم :

این تابع دو پارامتر به عنوان ورودی میگیرد : input_hist (هیستوگرام تصویر ورودی) و most_similar_hist (هیستوگرام مشابه ترین تصویر).این تابع از تابع compare_histograms برای محاسبه فاصله اقلیدسی بین input_hist و most_similar_histاستفاده می کند. امتیاز شباهت با استفاده از فرمول: 1/(distance + 1) محاسبه می شود .هدف از استفاده از این فرمول تبدیل فاصله اقلیدسی به امتیاز شباهت است .فاصله‌های کوچک‌تر منجر به نمرات شباهت بزرگ‌تر می‌شود که نشان‌دهنده شباهت بیشتر است.فرمول استفاده شده در تابع تضمین می کند که امتیاز بالاتر زمانی که هیستوگرام ها مشابه هستند بیشتر است و با کمتر شدن شباهت هیستوگرام ها به 0 نزدیک می شود.

کد نوشته شده برای سه تابع تعریف شده در صفحه قبل به صورت زیر است :

```
def calculate_histogram(image, color_space):

    if color_space == 'rgb':
        # Calculate the histogram using 8 bins for each channel (0-255)
        hist = cv2.calcHist([image], [0, 1, 2], None, [8, 8, 8], [0, 256, 0, 256, 0, 256])

    elif color_space == 'hsv':
        # Convert the image to HSV color space
        hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
        # Calculate the histogram using 8 bins for each channel (0-255)
        hist = cv2.calcHist([hsv_image], [0, 1, 2], None, [8, 8, 8], [0, 256, 0, 256, 0, 256])

    else:
        # Raise an error if the color space is neither 'rgb' nor 'hsv'
        raise ValueError("Invalid color space. Choose 'rgb' or 'hsv'.")

    # Flatten the histogram and normalize it
    hist = hist.flatten()
    hist /= hist.sum()
    return hist

def compare_histograms(hist1, hist2):
    # Calculate the Euclidean distance between two histograms
    return distance.euclidean(hist1, hist2)

def calculate_similarity_score(input_hist, most_similar_hist):
   # Calculate the similarity score between two histograms
    return 1 / (1 + compare_histograms(input_hist, most_similar_hist))
```

حال یک تابع برای آنالیز شباهت بین تصویر ورودی و مجموعه ای از عکس ها، از جمله نمایش ۱۰ تصویر برتر مشابه همراه با فاصله ی آن ها با تصویر ورودی و برچسب آن ها، محاسبه دقت، نمایش مشابه ترین تصویر و ارائه امتیاز شباهت به نام analyze_similarity تعریف میکنیم :

این تابع 4 پارامتر به عنوان ورودی میگیرد : input_photo (تصویر ورودی)، input_hist (هیستوگرام تصویر ورودی)، distances(برای ذخیره فواصل) و color_space(تعیین فضای رنگی برای استفاده). در اینجا توضیحی در مورد هر بخش از تابع آورده شده است :

مرتب سازی فواصل : فواصل بین هیستوگرام ها به ترتیب صعودی با استفاده از distances.sort (key=lambda x: x[0]) مرتب می شوند .

10عکس برتر : 10عکس برتر با کمترین فاصله از لیست فواصل مرتب شده انتخاب می شوند. نمایش تصاویر : تابع display_imagesبرای نمایش عکس ورودی و 10 عکس برتر مشابه در یک شبکه فراخوانی می شود .عناوین هر تصویر بر اساس برچسب ها و فاصله آنها ایجاد می شود .

محاسبه دقت : تعداد عکس هایی که به درستی تطبیق داده شده اند (دارای برچسب مشابه با مشابه ترین عکس) در 10 عکس برتر محاسبه می شود .درصد دقت :درصد دقت بر اساس تعداد عکس هایی که به درستی تطبیق داده شده اند محاسبه می شود.

مشابه ترین آنالیز عکس :مشابه ترین عکس از فواصل مرتب شده انتخاب شده و هیستوگرام آن محاسبه می شود .محاسبه امتیاز شباهت :امتیاز شباهت با استفاده از تابع calculate_similarity_score محاسبه می شود و هیستوگرام ورودی با مشابه ترین هیستوگرام مقایسه می شود.

نمایش مشابه ترین عکس :مشابه ترین عکس با استفاده از تابع display_images با عنوانی که نشان می دهد شبیه ترین عکس است نمایش داده می شود .نتایج چاپ :درصد دقت و امتیاز شباهت در کنسول چاپ می شود.

کد این تابع به صورت زیر است :

```
def analyze_similarity(input_photo, input_hist, distances, color_space):

    # Sort the list of distances based on the first element (distance)
    distances.sort(key=lambda x: x[0])

    # Select the top 10 photos with the smallest distances
    top_10_photos = distances[:10]
    # Generate titles for the top 10 photos with their labels and distances
    top_10_titles = [f"({i+1}) label: {label} | Dist: {dist:.2f}" for i, (dist, label, _) in enumerate(top_10_photos)]

    # Display the input photo and the top 10 similar photos with titles
    display_images(input_photo, [photo for _, _, photo in top_10_photos], top_10_titles)

    # Count how many of the top 10 photos have the same label as the closest photo
    correct_count = sum(1 for _, label, _ in top_10_photos if label == distances[0][1])
    # Calculate the accuracy percentage
    accuracy_percentage = (correct_count / 10) * 100

    # Print the number of correct photos and the accuracy percentage
    print(f"\033[1m{correct_count}\033[0m photos of those ten are in the same group as the input photo: \033[1m{accuracy_percent

    # Select the most similar photo
    most_similar_photo = distances[0][2]
    # Calculate the histogram of the most similar photo
    most_similar_hist = calculate_histogram(most_similar_photo, color_space)

    # Calculate the similarity score between the input and the most similar photo
    similarity_score = calculate_similarity_score(input_hist, most_similar_hist)

    # Display the input photo and the most similar photo with a title
    display_images(input_photo, [most_similar_photo], [f"Most Similar"])
    print(f"\033[1mSimilarity Score: {similarity_score:.2f}")
```

یک تابع نیز برای نمایش بصری از عکس ورودی و مجموعه ای از عکس های مشابه با عناوین مربوطه به نام display_images تعریف میکنیم :

این تابع 4 پارامتر به عنوان ورودی میگیرد : input_photo (تصویر ورودی)، similar_photos (تصاویر مشابه)، titles (عناوین تصاویر) ، rows (تعداد سطرها) و cols (تعداد ستون ها) . کد این تابع به صورت زیر است :

```
def display_images(input_photo, similar_photos, titles, rows=4, cols=4):

    plt.figure(figsize=(20, 15))

    # Display the input photo
    plt.subplot(rows, cols, 1)
    plt.imshow(cv2.cvtColor(input_photo, cv2.COLOR_BGR2RGB))
    plt.title("Input Photo")
    plt.axis("off")

    # Display the similar photos with titles
    for i, (photo, title) in enumerate(zip(similar_photos, titles), start=1):
        plt.subplot(rows, cols, i + 1)
        plt.imshow(cv2.cvtColor(photo, cv2.COLOR_BGR2RGB))
        plt.title(title)
        plt.axis("off")

    plt.show()
```

حال یک تابع main تعریف کرده تا از توابع نوشته شده برای مجموعه داده corel استفاده کنیم :

این تابع 3 پارامتر به عنوان ورودی میگیرد : input_photo_path (آدرس تصویر ورودی)، dataset_path (آدرس دیتاست) و color_space(تعیین فضای رنگی برای استفاده).

تابع عکس ورودی مشخص شده توسط input_photo_path را با استفاده از cv2.imreadمی خواند .هیستوگرام عکس ورودی را با استفاده از تابع calculate_histogram محاسبه می کند.سپس روی پوشه‌های موجود در مجموعه داده مشخص شده توسط dataset_pathتکرار می‌شود .برای هر پوشه، روی فایل‌ها تکرار می‌شود و بررسی می‌کند که آیا تصاویر معتبر هستند یا خیر.

مقایسه هیستوگرام ها :برای هر تصویر معتبر در مجموعه داده، عکس را می خواند، هیستوگرام آن را محاسبه می کند و با استفاده از تابع compare_histograms با هیستوگرام عکس ورودی مقایسه می کند فاصله ها، برچسب ها و عکس ها در لیست فاصله ها ذخیره می شوند.پس از جمع آوری فواصل، تابع analyz_similarity را برای تجزیه و تحلیل و نمایش نتایج مشابه فراخوانی می کند.

```
if __name__ == "__main:"__
```

مسیر عکس ورودی، مسیر مجموعه داده و فضای رنگ را مشخص می کند.

کد تابع main برای مجموعه داده corel به صورت زیر است :

```
def main(input_photo_path, dataset_path, color_space):

    # Load input photo and calculate its histogram
    input_photo = cv2.imread(input_photo_path)
    input_hist = calculate_histogram(input_photo, color_space)

    # Initialize an empty list to store distances
    distances = []

    # Loop through each folder in the dataset path
    for folder in os.listdir(dataset_path):
        folder_path = os.path.join(dataset_path, folder)

        # Check if the item in the folder path is a directory
        if os.path.isdir(folder_path):
            # Loop through each file in the folder
            for filename in os.listdir(folder_path):
                file_path = os.path.join(folder_path, filename)

                 # Check if the item in the file path is a file
                if os.path.isfile(file_path):

                    # Read the photo from the file path
                    photo = cv2.imread(file_path)
                    # Calculate histogram for the photo
                    photo_hist = calculate_histogram(photo, color_space)
                    # Calculate Euclidean distance between input photo and current photo
                    dist = compare_histograms(input_hist, photo_hist)
                    # Append the distance, folder (label), and photo to the distances list
                    distances.append((dist, folder, photo))

    # Analyze the similarity between input photo and the dataset
    analyze_similarity(input_photo, input_hist, distances, color_space)

if __name__ == "__main__":

    input_photo_path = 'Documents/corel_dataset/training_set/horses/799.jpg'
    dataset_path = "Documents/corel_dataset/test_set"

    main(input_photo_path, dataset_path, color_space='rgb')
```

حال تابع main را بازنویسی کرده تا از توابع نوشته شده برای مجموعه داده CIFAR-10 استفاده کنیم :

این تابع نیز 3 پارامتر به عنوان ورودی میگیرد: input_photo_path (آدرس تصویر ورودی)، dataset_path (آدرس دیتاست) و color_space(تعیین فضای رنگی برای استفاده).تابع مجموعه داده CIFAR-10 را بارگیری می کند، آخرین تصویر را از هر کلاس استخراج می کند، کلاس خاصی را به عنوان ورودی انتخاب می کند و سپس عکس ورودی را با بقیه مجموعه داده بر اساس هیستوگرام های رنگی مقایسه می کند در اینجا توضیحی در مورد هر بخش از تابع آورده شده است :

مجموعه داده CIFAR-10 با استفاده از cifar10.load_data() بارگیری می شود .برای هر کلاس (0 تا 9)، آخرین تصویر از مجموعه آموزشی استخراج شده و در لیست last_pictures ذخیره می شود .می توانید شاخص کلاس مورد نظر (input_class_index)را برای عکس ورودی مشخص کنید :.عکس ورودی انتخاب شده به عنوان "input_photo.jpg" با

استفاده از OpenCV (cv2.imwrite) ذخیره می شود .عکس ورودی ذخیره شده بارگیری می شود و هیستوگرام رنگی آن با استفاده از تابع calculate_histogram محاسبه می شود .فواصل اقلیدسی بین هیستوگرام عکس ورودی و هر عکس در مجموعه آموزشی با استفاده از تابع compare_histograms محاسبه می شود .فاصله ها، برچسب ها و عکس ها در لیست فاصله ها ذخیره می شوند.پس از جمع آوری فواصل، تابع analyz_similarity را برای تجزیه و تحلیل و نمایش نتایج مشابه فراخوانی می کند.

if __name__ == "__main:"__

مجموعه داده CIFAR-10 در صورتی که قبلاً موجود نباشد در یک فهرست مشخص دانلود می شود .سپس تابع اصلی با پارامترهای مشخص شده فراخوانی می شود.

کد تابع main برای مجموعه داده CIFAR-10 به صورت زیر است :

```
def main(input_photo_path, dataset_path, color_space):

    # Load CIFAR-10 dataset
    (x_train, y_train), (_, _) = cifar10.load_data()

    # Extract the last pictures from each class
    last_pictures = []

    # Loop through each class label (0 to 9)
    for class_label in range(10):

        # Get the indices of samples belonging to the current class
        class_indices = np.where(y_train == class_label)[0]
        # Get the index of the last sample in the current class
        last_picture_index = class_indices[-1]
        # Retrieve the last picture from the training set for the current class
        last_picture = x_train[last_picture_index]
        # Append the last picture to the list
        last_pictures.append(last_picture)

    # Choose the index of the class you want to use as input (0 to 9)
    input_class_index = 0  # Replace with the desired class index

    # Save the image to a file (replace 'path_to_save_image.jpg' with your desired file path)
    cv2.imwrite('input_photo.jpg', cv2.cvtColor(last_pictures[input_class_index], cv2.COLOR_RGB2BGR))

    # Set input_photo_path to the saved file path
    input_photo_path = 'input_photo.jpg'

    # Load the input photo and calculate its histogram
    input_photo = cv2.imread(input_photo_path)
    input_hist = calculate_histogram(input_photo, color_space)
```

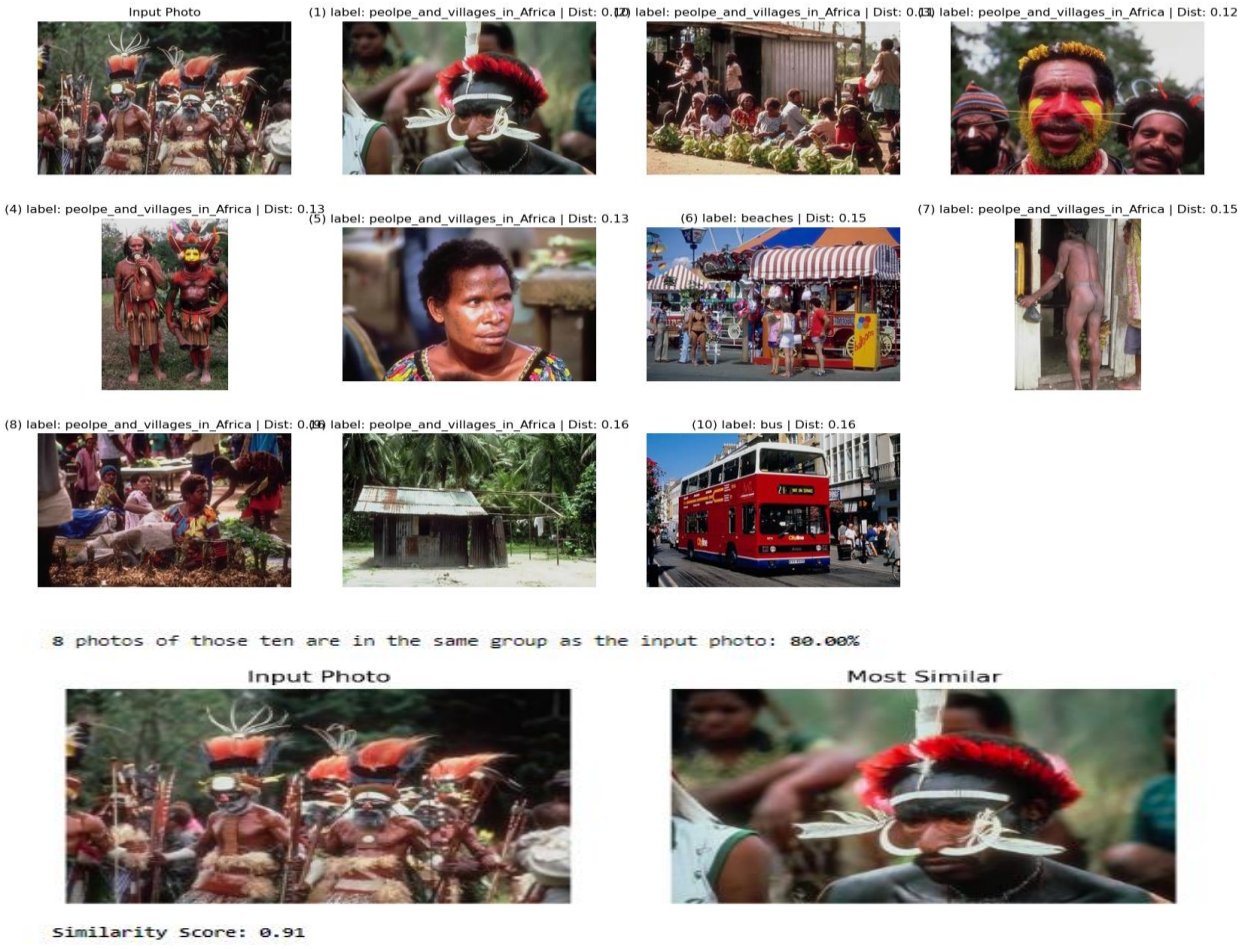
```
    # Initialize an empty list to store distances
    distances = []

    # Loop through each photo in the training set
    for i, photo in enumerate(x_train):

        # Calculate histogram for the current photo
        photo_hist = calculate_histogram(photo, color_space)
        # Calculate Euclidean distance between input photo and current photo
        dist = compare_histograms(input_hist, photo_hist)
        # Append the distance, label, and photo to the distances list
        distances.append((dist, y_train[i, 0], photo))

    # Analyze the similarity between the input photo and the dataset
    analyze_similarity(input_photo, input_hist, distances, color_space)

if __name__ == "__main__":

    # Specify the local directory to download CIFAR-10
    dataset_path = os.path.expanduser('~/.keras/datasets/')

    # Download CIFAR-10 dataset if not already downloaded
    cifar10_dir = get_file('cifar-10-batches-py/test-batch', origin='https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz',

    main(input_photo_path, dataset_path, color_space='rgb')
```

برای هر مجموعه داده ۳ نمونه خروجی در دو فضای رنگی RGB و HSV آورده شده است. (بقیه خروجی ها به همراه کد در فایل zip ارسال میشود.)

خروجی کد های بالا برای مجموعه داده corel به صورت زیر است :

برای آخرین عکس پوشه ی peolpe_and_villages_in_Africa یعنی عکس 99.jpg در فضای رنگی RGB :

۱۰ تصویر برتر مشابه با تصویر ورودی به همراه ditance و lable مربوط به هر کدام در بالای آن مشخص شده است .



الف) با توجه به خروجی کد شبیه ترین تصویر با تصویر ورودی در کنار هم نمایش داده شده است و میتوان فهمید که در یک دسته قرار دارند یا خیر . همچنین امتیاز شباهت برای شبیه ترین تصویر در زیر آن نوشته شده است . (امتیاز شباهت برای این مثال ۰,۹۱ می باشد)

ب) با توجه به خروجی کد مشخص شده است که چند درصد از تصاویر همگروهی این تصاویر در ۱۰ تصویر اول آمده است(در این مثال ۸ تصویر از ۱۰ با تصویر ورودی همگروهی هستند.)

درصد بازیابی سیستم بازیابی تصویر بر اساس محتوا به این صورت محاسبه میشود که چند تا از تصاویر بازیابی شده با تصویر ورودی در یک دسته قرار میگیرند و آن را به صورت درصد بیان میکنند .در این تمرین برای ۱۰ تصویر برتر مشابه آن را محاسبه کرده و در زیر تصاویر مشابه به صورت درصدی بیان کردیم. (در این مثال برابر است با ۸۰ درصد)

برای آخرین عکس پوشه ی peolpe_and_villages_in_Africa یعنی عکس 99.jpg در فضای رنگی HSV :


```
9 photos of those ten are in the same group as the input photo: 90.00%
```


```
Similarity Score: 0.91
```

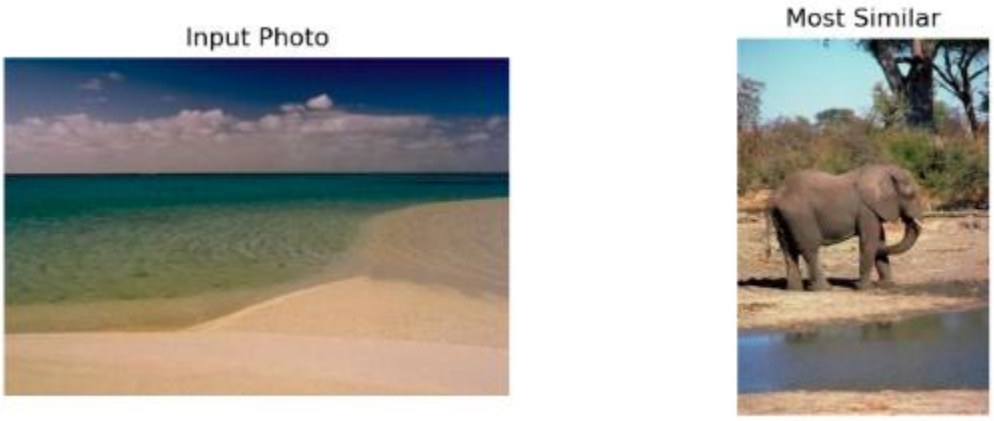
برای آخرین عکس پوشه ی beaches یعنی عکس 199.jpg در فضای رنگی RGB :


```
4 photos of those ten are in the same group as the input photo: 40.00%
```


```
Similarity Score: 0.82
```

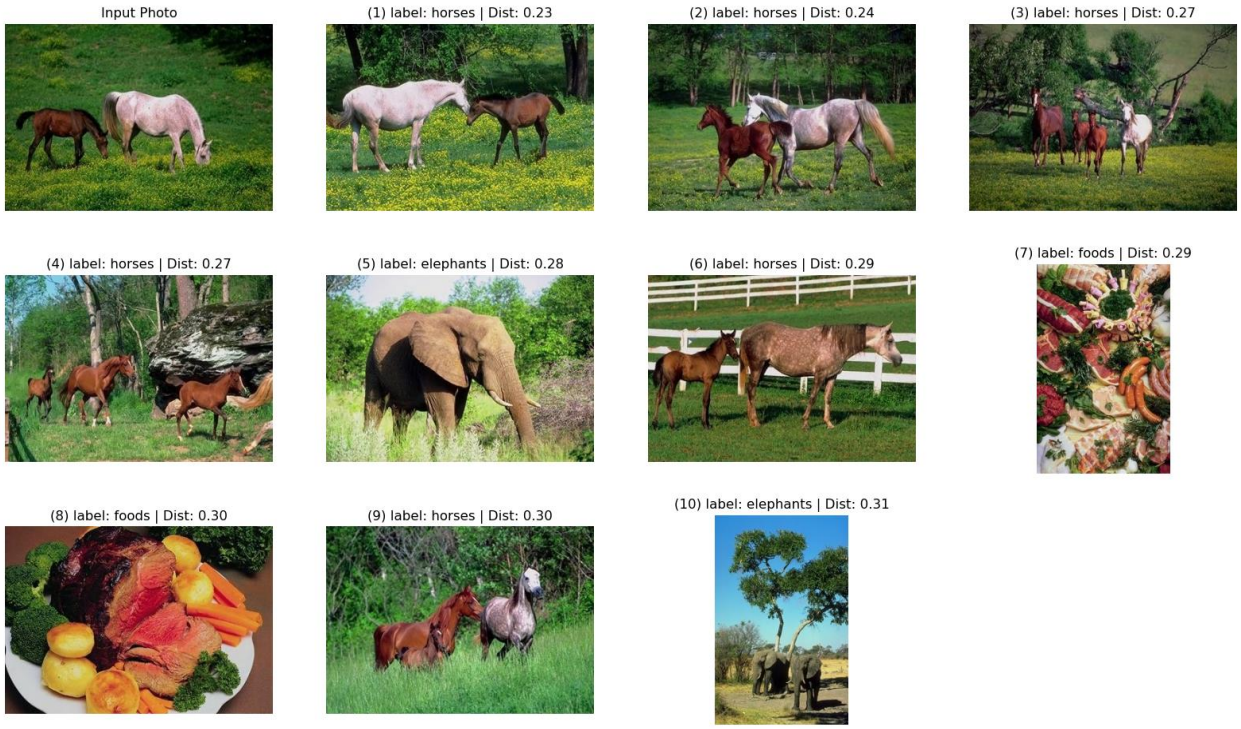
برای آخرین عکس پوشه ی beaches یعنی عکس 199.jpg در فضای رنگی HSV :


```
3 photos of those ten are in the same group as the input photo: 30.00%
```

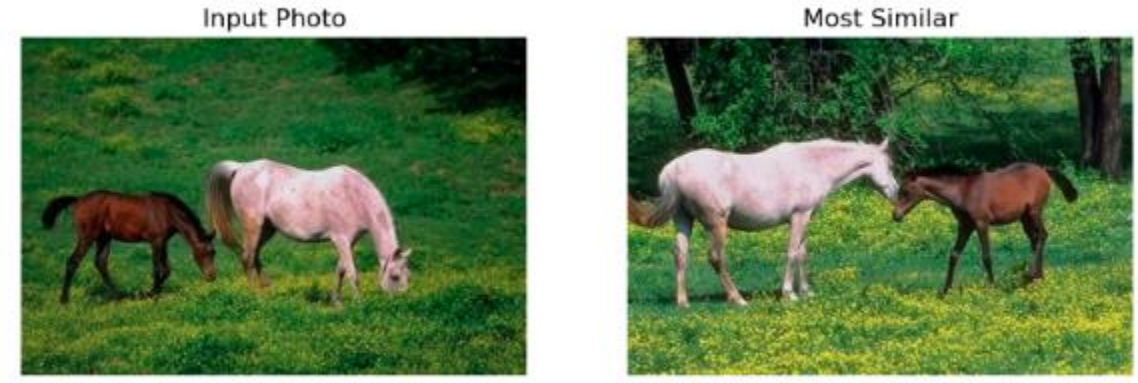



```
Similarity Score: 0.82
```

برای آخرین عکس پوشه ی horses یعنی عکس 799.jpg در فضای رنگی RGB :

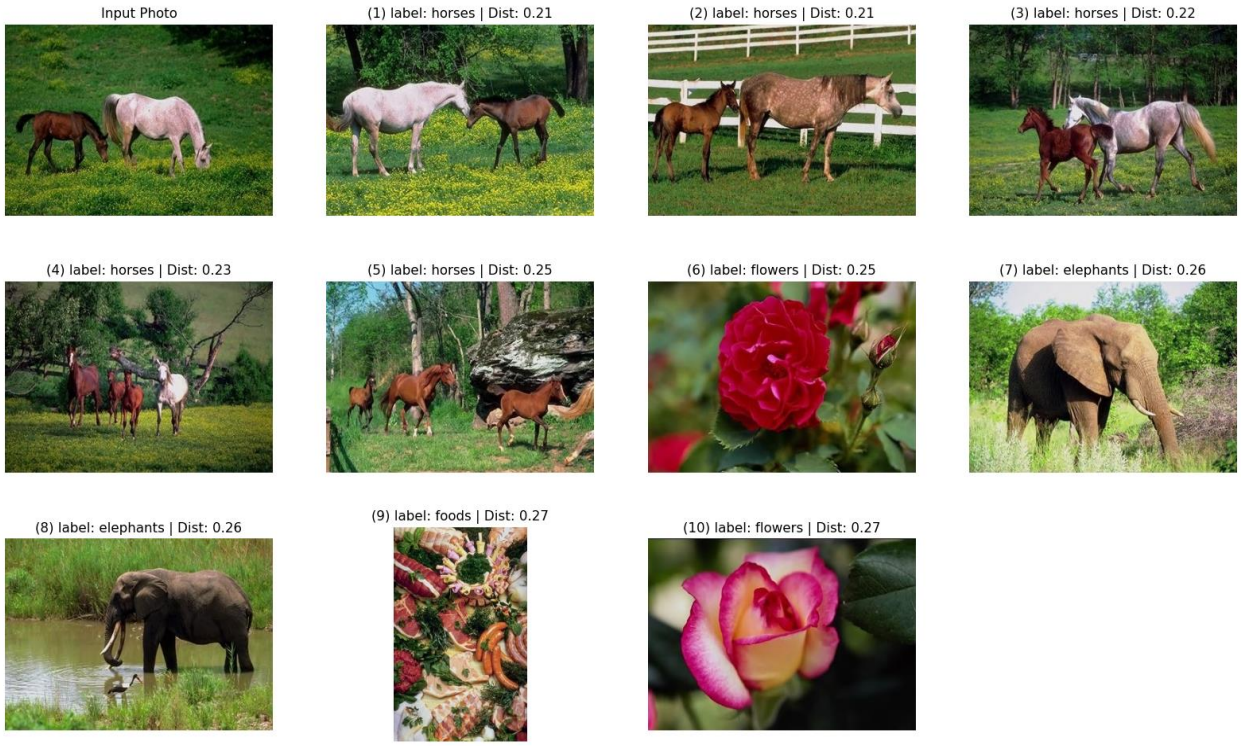


6 photos of those ten are in the same group as the input photo: 60.00%



Similarity Score: 0.81

برای آخرین عکس پوشه ی horses یعنی عکس 799.jpg در فضای رنگی HSV :



5 photos of those ten are in the same group as the input photo: **50.00%**

Similarity Score: 0.83

خروجی کد های بالا برای مجموعه داده cifar10 به صورت زیر است :

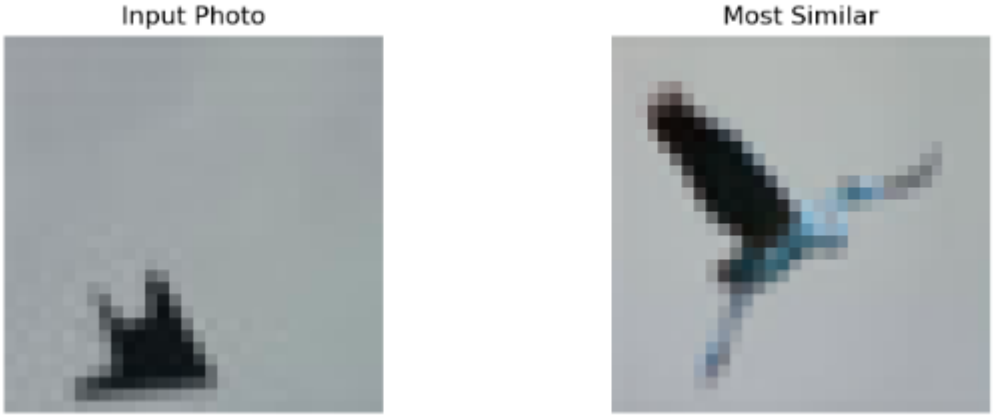
برای آخرین عکس کلاس [0] در فضای رنگی RGB :


```
4 photos of those ten are in the same group as the input photo: 40.00%
```


```
Similarity Score: 0.81
```

برای آخرین عکس کلاس [0] در فضای رنگی HSV :

4 photos of those ten are in the same group as the input photo: **40.00%**



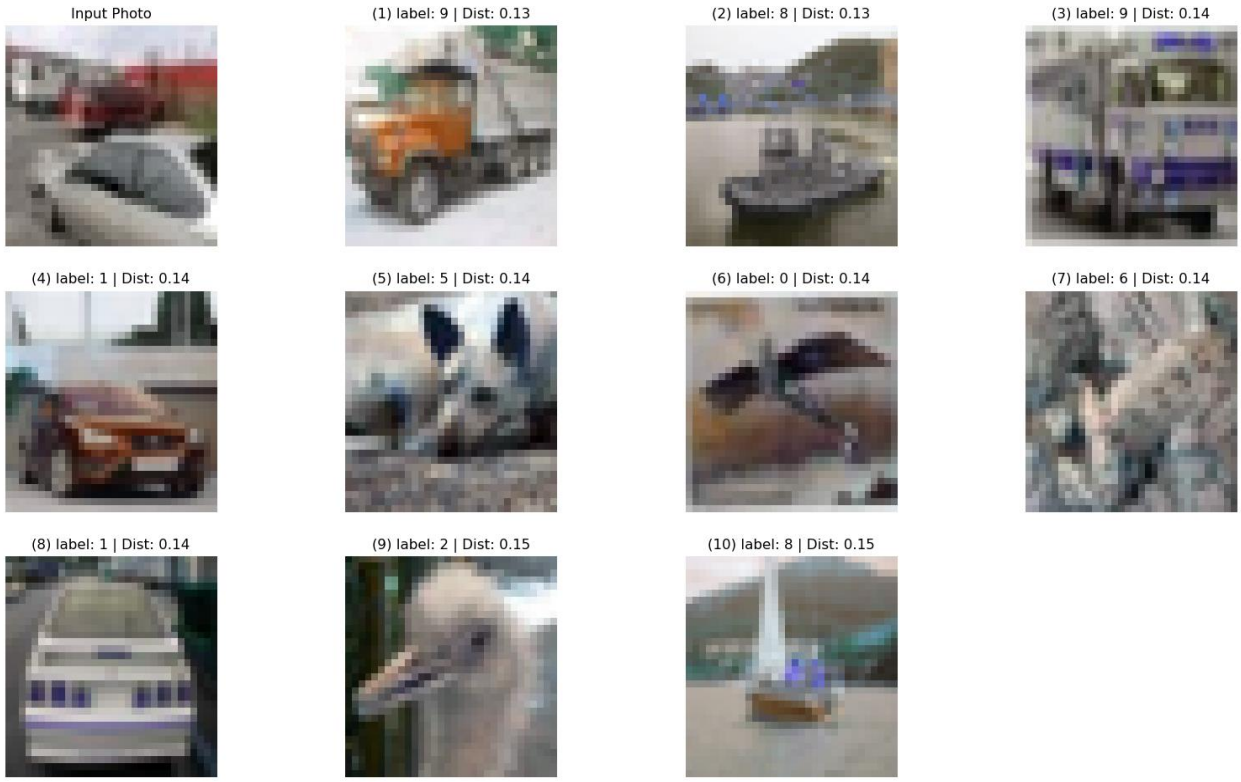
Similarity Score: 0.87

برای آخرین عکس کلاس [1] در فضای رنگی RGB :

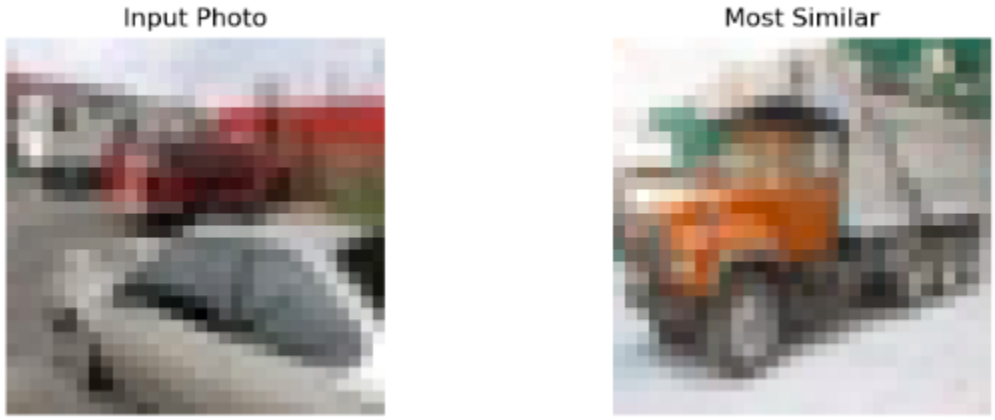
1 photos of those ten are in the same group as the input photo: **10.00%**

Similarity Score: 0.90

برای آخرین عکس کلاس [1] در فضای رنگی HSV :

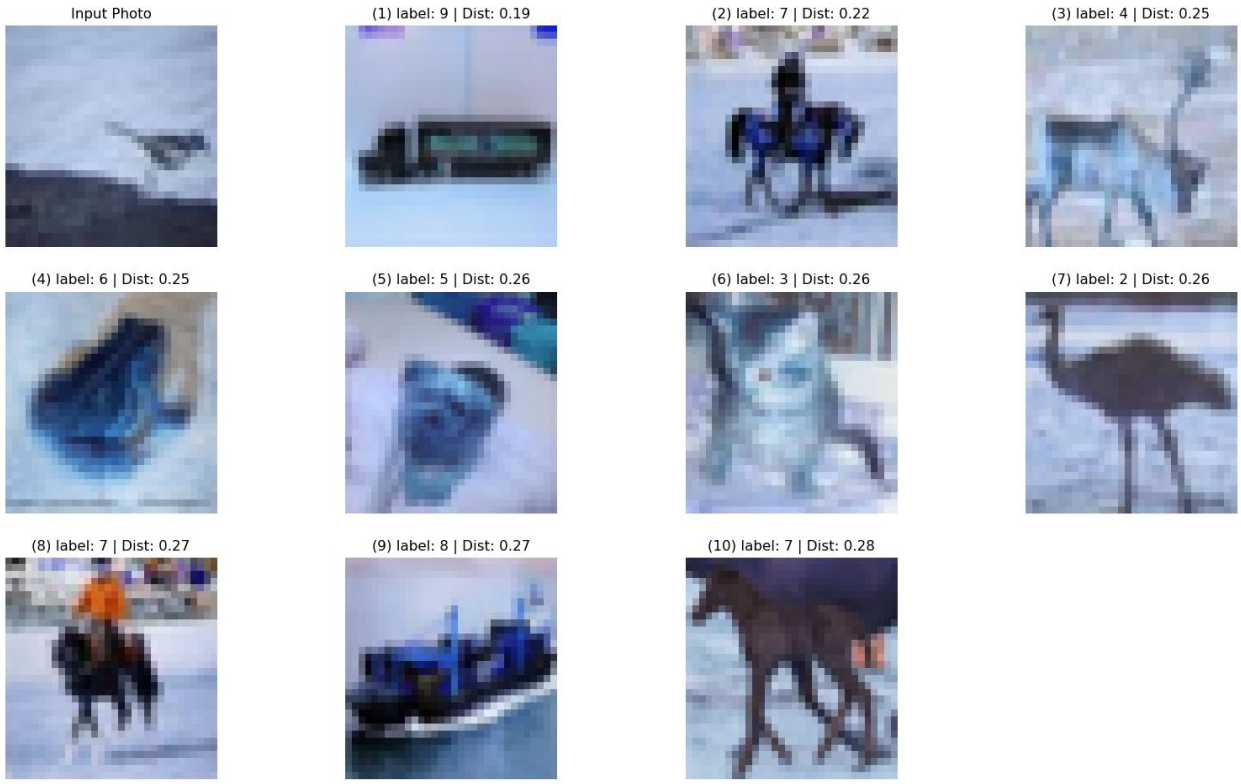


2 photos of those ten are in the same group as the input photo: **20.00%**

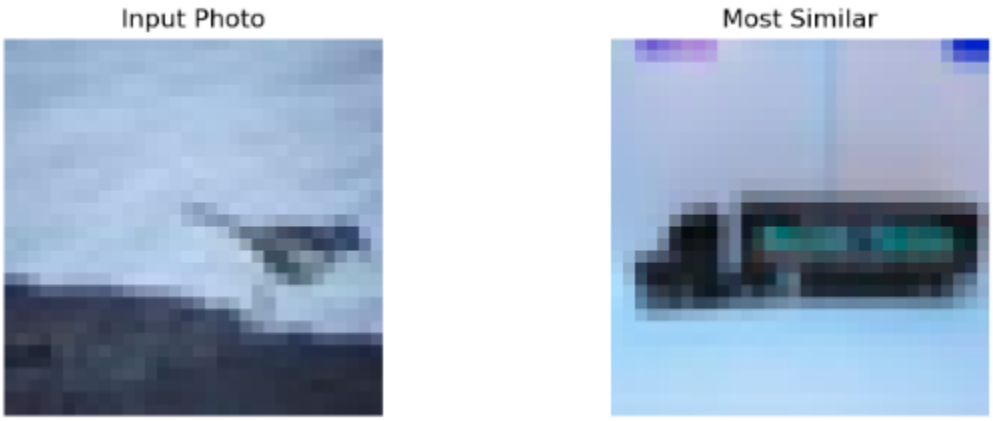


Similarity Score: 0.88

برای آخرین عکس کلاس [2] در فضای رنگی RGB :



1 photos of those ten are in the same group as the input photo: **10.00%**



Similarity Score: 0.84

برای آخرین عکس کلاس [2] در فضای رنگی HSV :

3 photos of those ten are in the same group as the input photo: **30.00%**

Similarity Score: 0.83