Tim Medvedev (Individual team)

CSS487 Project Technical Writeup

Fall Quarter 2022

Cartoon Recognition

Introduction

My project focused on the attempt to accurately process images and classify them as being cartoons, or photographs taken in real life. Cartoons have several notable features that separate them from photographs.

First, as they are drawings, they have much clearer edges and stronger gradients. Second, they often have a much more restricted color palette than many photographs. Cartoons also often have clear text and other elements of graphic design that can only be introduced through drawings.

Process

For my first attempt I tried to focus on the strong edges present in cartoons. I first tried to simply use the OpenCV Canny() Canny Edge Detection function to count the total nonzero edges in an image. Unfortunately, after some testing it was quite inconclusive to try and classify images strictly off the number of edges detected within. I found some photographs to have an unexpectedly small number of edges detected, and some cartoons to have very many. One noticeable feature though, was that the Canny() function did do very well in picking up the strong edges of the cartoons in the right places. With photographs the edge detection function detected the strongest edges, while failing to identify the exact contours of the shapes that the edges belonged to. With the cartoons the edge detection function did very well in outlining all of the shapes in the image.

After finding some trouble with this approach, I focused on the color differences between cartoons and photographs. My first approach that did have some success and I ended up keeping, was the attempt to measure the gradient differences in the images. In this approach I used color histograms to measure the colors of the pixels. The approach involves taking the given image, blurring it slightly, then calculating histograms for the original and blurred image, then comparing the histograms. The reasoning behind this was that if a highly detailed photograph was blurred, its blurred histogram would have be more identical to its original image, than a cartoon's blurred histogram would be. The Cartoon would have larger differences in the blurred histogram due to the stronger gradients associated with the strong drawn edges. My method used a rolling total to add

up the decimal percentage in the bins found to be different, and add a 1 in the bins found to be the same. Then the total value was divided by the number of bins and used to classify the image. I had trouble using the OpenCV CalcHist method, so I used our histogram code from Program 3. I found this method to not be very precise, and only useful when paired with another method later on.

One other method that I had read about seemed like a significant improvement over this previous method, although I did have trouble implementing it properly. The method attempts to take advantage of the uniform colors of cartoons. The method is to take a number of the most common colors in an image, attempt to recreate that image using only those colors, then measure the difference in the recreated image with the original. I had trouble implementing this method with my own histogram code, and wasn't sure about the most effective way to recreate an image using only a certain number of histogram bins. This did help me to work on the third method I used.

The third method I worked on to help classify the images worked on a similar principle to the method described previously. My reasoning was that the previous method took advantage of how "colorful" an image might be, so I sought to measure the overall "colorfulness" of an image. My method measures the difference between red and green, then adds that to the difference of red and green compared to blue. My thinking was that cartoons are more likely to have a simple color like (255, 0, 0), and it would have it in a large patch. In a photograph, depending on the lighting, even if an object has a color such as (255, 0, 0), it will not be that way across its whole surface. A cartoon is more likely to have pixels with a larger quantifiable difference across the surface of a shape. My method takes these differences and adds them up for every single pixel within the image, and then divides the total by the total number of pixels in order to get an average difference between the three-color channels. I was pleasantly surprised to see this helped the accuracy of my model quite a bit. For instance, I found the images with an average difference of over 200 to be only cartoons.

My final step was to create an integer that would represent the chance of how likely a processed image is in fact a cartoon. For the first method the threshold I used was that every image with a histogram similarity of lower than 0.7 would have a 1 added to its "cartoon level". This would grant it the status of "possibly a cartoon". A higher value means a higher level of similarity between the original and blurred histograms. For the third method, anything with a value of over 200 was deemed to be "likely a cartoon", while anything greater than the threshold of 145 also had a 1 added to its cartoon level. Any image with a cartoon level of 2 or greater was deemed "likely a cartoon", while images with a cartoon level of 1 were deemed "possibly a cartoon", anything else was deemed "likely a photo".

I also attempted to implement face detection on cartoon faces as originally planned. I planned to use the OpenCV Haar Cascade tutorial and methodology but found the trouble with cartoon faces to be the wide variety. Specific models for cascades must be trained to specific styles of cartoon face as otherwise they simply have too much variety in their shapes and contours to train one model to work for all cartoons.

Conclusion and Results

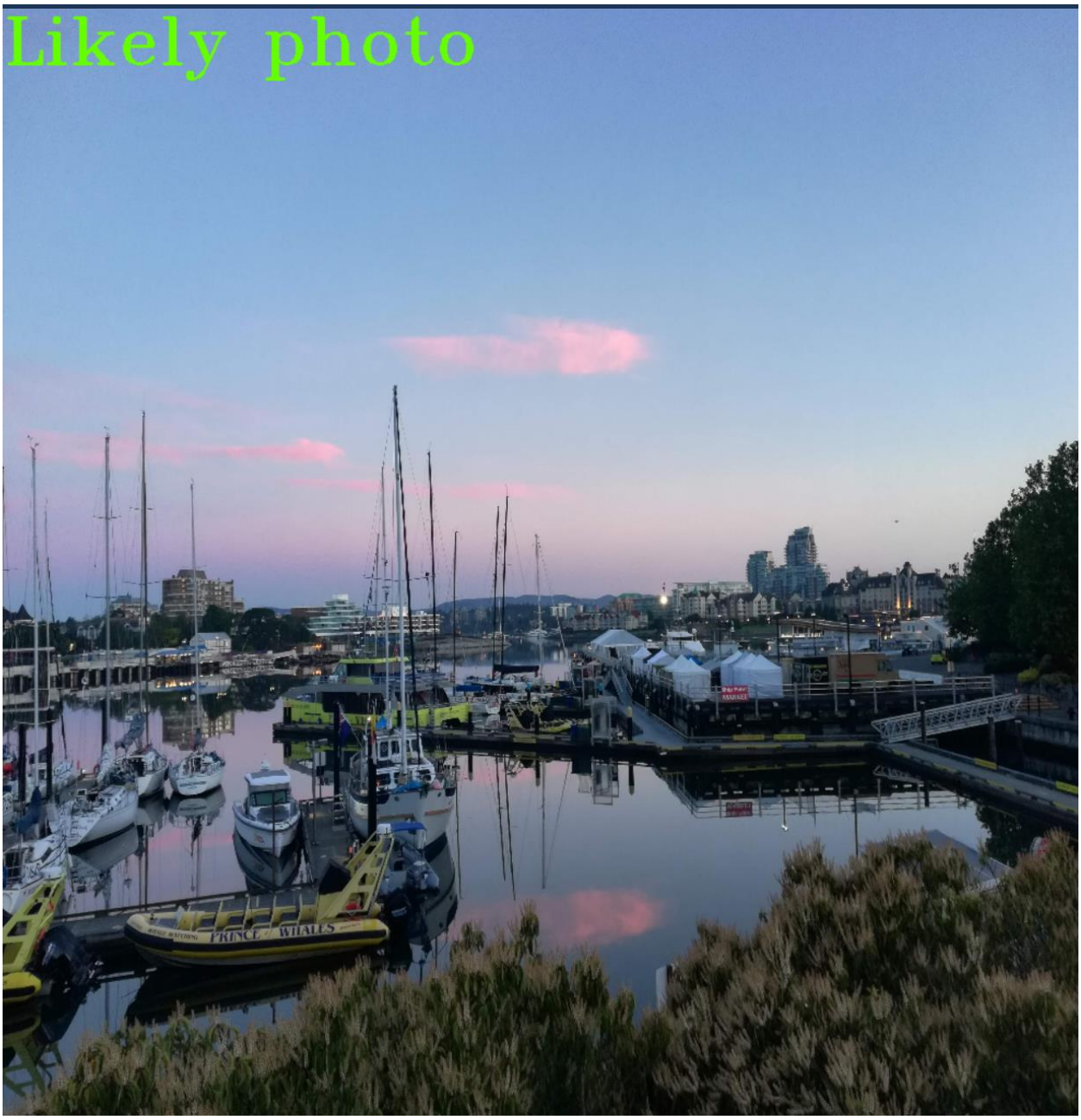
In conclusion I really feel that I learned a lot about image classification and colors. It was very interesting working with the color histograms to try and implement the methods I read about. I still want to return to this project and try to properly implement the method of recreating an image with a subsection of the most common colors. I really enjoy cartoons and always have, so it was really fascinating for me to think about some of their arbitrary qualities and how one would maybe have a computer see their differences. It was a very fun project for me to work on and I feel like I would use it myself, it could be used to classify stills in a digital music library. The accuracy is imprecise, but I feel it could improve greatly. The best and most current method would most certainly be to train a large neural network model on identifying cartoons and photographs.

It was a joy taking this class, and while it was difficult the feedback and lessons were really eye opening and fascinating. I have gained a whole new appreciation for computer vision and its possibilities.

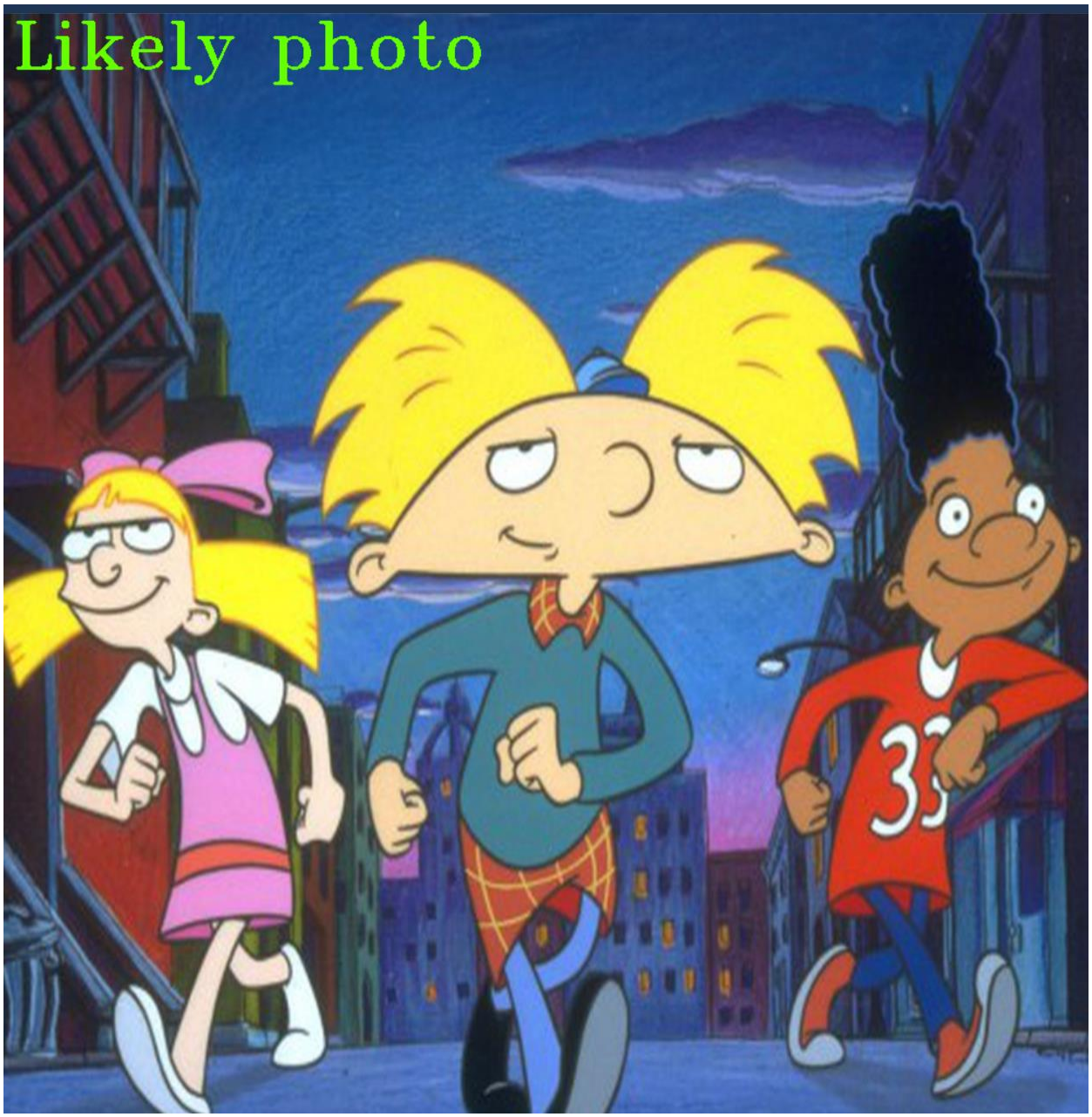
Images:

This image from the power puff girls is an example of a partially correct identification

This image is an example of a correct photo identification



This image from Hey Arnold is an example of a false negative



This image from Arthur is an example of an accurate cartoon identification

This image is an example of a partially false positive, luckily there were no complete false positives found.

Table:

PC is “Probably a cartoon”

LC is “Likely a cartoon”

LP is “Likely a photo”

C/P images are realistic artworks that share qualities of both

Column1	HistogramComparison	ColorfulMetric	Edges	FinalResult	ActualIdentity
1	0.7189	146.335	20675	PC	C/P
2	0.8886	221.365	9702	PC	C
3	0.8409	340.713	18453	LC	C
4	0.872	134.145	28717	LC	C
5	0.9272	143.65	23764	LP	C
6	0.8509	104.304	21218	LP	P
7	0.8111	150.43	51838	LP	P
8	0.907502	126.191	20771	PC	P
9	0.843	80.936	20506	LP	C
10	0.9161	97.347	35408	LP	P
11	0.8623	105.729	44290	LP	P
12	0.879534	115.321	8930	LP	P
13	0.775	76.1463	70720	LP	P
14	0.7908	113.981	14052	LP	P
15	0.8997	63.393	6326	LP	P
16	0.7971	99.774	39772	LP	P
17	0.8602	102.756	29747	LP	P
18	0.98	137.113	20362	LP	P
19	0.8997	213.312	3527	LC	C
20	0.6476	52.3701	17556	PC	P
21	0.9994	209.202	6407	LC	C
22	0.783384	252.935	56316	LC	C
23	0.7782	157.071	25396	PC	C
24	0.809	260.712	25915	LC	C
25	0.7735	120.466	58639	LP	C/P
26	0.8282	203.11	40994	LC	C
27	0.751933	193.809	86626	PC	C
28	0.790348	164.681	36948	PC	P
29	0.8222	132.599	49213	LP	C
30	0.83267	177.434	70453	PC	C
31	0.9727	252.236	674	LC	C
32	0.9275	291.887	10015	LC	C
33	0.943743	142.729	18999	PC	C

34	0.9034	193.158	23163	PC	C
35	0.8487	203.467	29119	LC	C
36	0.7893	200.897	81169	LC	C
37	0.6594	192.204	80089	LC	C
38	0.7568	164.214	54532	PC	C
39	0.8121	176.786	70044	PC	C
40	0.7334	229.407	29500	LC	C
41	0.9072	191.38	7569	PC	P
42	0.7332	148.478	46889	PC	C

Out of 27 cartoon images 15 were deemed “Likely a cartoon” and another 8 were deemed “Probably a cartoon”. 3 were false negatives deemed to be probably a photo

So the perfect accuracy is not much better than a coin toss at 55%, but including the “probablies”, it is about 85%

Out of 14 photographs 4 were deemed “Probably a cartoon”, and 10 were deemed “Likely a photograph. No photograph was deemed “Likely a cartoon”

Works Cited

Griffith, Chris. “Cartoon or Photo? - Image Detection with Python.” *Code Calamity*, 15 Dec. 2021, codecalamity.com/cartoon-or-photo-image-detection-with-python/#opencv-gradient-differences. Accessed 11 Dec. 2022.

“Python - Detecting If Image Is a Cartoon or If It Is a Real Person with OpenCV.” *Stack Overflow*, 10 Mar. 2016, stackoverflow.com/questions/22264469/detecting-if-image-is-a-cartoon-or-if-it-is-a-real-person-with-opencv. Accessed 11 Dec. 2022.

Zaid, Mustafa & George, Loay & Al-Khafaji, Ghadah. (2015). Distinguishing Cartoons Images from Real-Life Images. International Journal of Advanced Research in Computer Science and Software Engineering. 5. 91-95.

Ianeva, Tzvetanka I. et al. “Detecting cartoons: a case study in automatic video-genre classification.” *2003 International Conference on Multimedia and Expo. ICME '03. Proceedings (Cat. No.03TH8698)* 1 (2003): I-449.