Machine learning vs non-machine learning algorithms to detect circles in images

Luckman Qasim • Mamun Or Rashid • Md Golam Mahmud Chowdhury

Abstract

Circle detection has been an important area of interest for many years in computer science. Some of the examples of detection of circular objects in images include, help identify abnormal cells, atomic structures, informative street signs, etc. Though the general algorithm appears very elementary, the optimized algorithm for detecting circular objects is strategically very substantial. In this research we are considering Convolutional Neural Network as a machine learning approach to detect circles in comparison to the Hough algorithm with Canny-edge detection for detecting circles. We will also look at how the machine learning approach augments the results in compared to the general approach or not, and the limitations of both methods.

Background

The identification of circular shapes is achievable using various algorithms created during computational development history. Frameworks like OpenCV have built-in functions that make use of these algorithms to detect circles. We are in our discourse experimenting on two such algorithms which have been constructed as an infrastructure to solve image processing using machine and non-machine learning approaches. The results will accentuate factors that represent the effectiveness of the algorithms, some of which are performance, time taken, edge detected image quality, and most importantly detecting circles from noises.

Central to this investigation are two principal methodologies renowned for their prowess in circle detection: Convolutional Neural Networks (CNNs) and the Hough Circle Transform. CNNs, as deep learning architectures, have revolutionized image analysis by learning intricate spatial hierarchies of features, thereby excelling in complex pattern recognition. On the other hand, the Hough Circle Transform, rooted in geometric parameterization, offers a robust approach by identifying circles based on their radius and center coordinates in a parameter space.

The primary objective of this research is to comprehensively compare these algorithms in terms of accuracy, computational efficiency, and adaptability to varied image environments. Such a comparative analysis is essential to discern the optimal scenarios for deploying these methodologies and understand the trade-offs between accuracy and computational cost.

This study presents a detailed evaluation conducted on diverse datasets encompassing varying levels of noise, blur, circle properties, and image dimensions. Through a meticulous examination of performance metrics such as Intersection over Union (IoU) for accuracy assessment and Average Execution Time (AET) for computational efficiency, this research aims to provide insights into the strengths and limitations of both CNNs and the Hough Circle Transform.

Convolutional Neural Network:

Convolutional Neural Networks (CNNs) have been instrumental in circle detection, leveraging their ability to learn intricate spatial hierarchies of features from images. The Convolution Layer in CNN mainly deals with feature extraction using convolution, where kernels of different dimensions are used to go over the input images and create feature maps that are passed into nonlinear activation functions to outputs zero if the input is less than zero and otherwise outputs the input value itself (Yamashita et al., 2018).

The pooling layers do downsampling of the featured maps to provide translation invariance to small shifts and distortions and decrease the number of subsequent learnable parameters. The Fully Connected layer takes the output featured maps and converts it into a 1D array, which is connected to more connected layers, where inputs are connected to outputs by learnable weights. The Last layer employs a specific activation function that converts the output of the network into a probability distribution over the classes, to decide on the learning success (Yamashita et al., 2018).

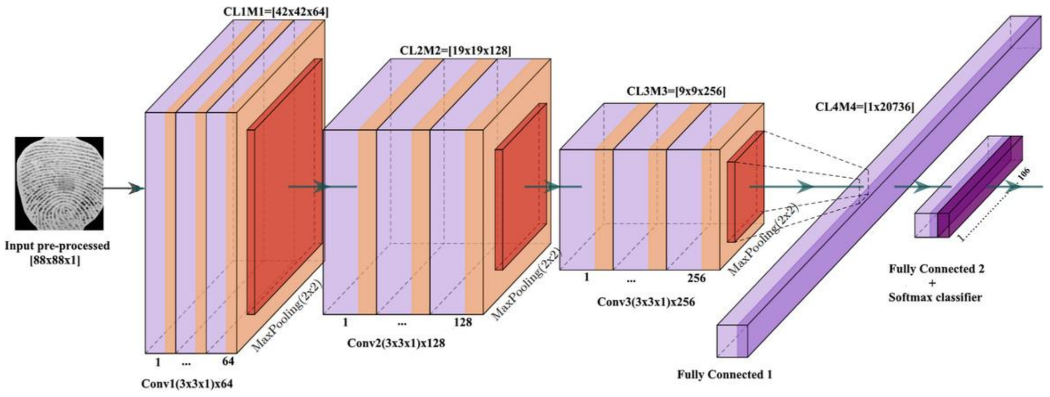


Figure 1: Convolutional Neural Network and its layers (Saha, 2018).

Our developed CNN model operates by ingesting input images and processing them through a series of convolutional layers. These layers are adept at extracting nuanced patterns and spatial relationships that are crucial for detecting circles. The architectural design of our CNN involve seven layers, five of them (self.L1 – self.L5) are convolutional with batch normalization, ReLU activation, and pooling layers, enabling it to capture hierarchical features that signify circular shapes, while the final two layers, i.e. Fully Connected layer and Last layer, are geared towards predicting the coordinates of circle centers and their respective radii.

The performance of our CNN is notably reliant on the quality and diversity of the training dataset. To generalize well across various image scenarios, the CNN necessitates a comprehensive dataset that encapsulates a wide array of circle sizes, positions, and environmental complexities.

Furthermore, the computational demands during both training and inference stages can be substantial due to the network's depth and complexity.

The CNN algorithm we have certain limitations. For it to work properly we need to provide it with a big enough dataset, which could be heavy on resources, because not all datasets are available freely and for a CNN to work accurately. Secondly, it only works with an image of fixed size (200x200 in our case), which means that if we try to find the circle in an image which is not the same size, we will have to crop or resize the image, which can distort the circle to be detected. Thirdly, our algorithm works on detecting a single circle only, as detecting multiple circles with the CNN algorithm is a whole new and difficult task.

Hough Circle Transform with Canny Edge:

The Hough Circle Transform stands as a valuable technique in image processing, specifically designed for circle detection through geometric parameterization. Its operational principle revolves around identifying circles within an image by representing each circle with its radius and center coordinates in a parameter space. By utilizing this parameter space representation, the algorithm effectively discerns circular shapes embedded within the image.

The algorithm takes in a set of parameters a, b and r for a combination of circles with a1, b1, r1, a2, b2, etc., and make feature points out of them. These feature points vote onto the accumulator array to find the presence of a circle. Each feature point is taken as the radius of an arbitrary circle, known as the edge points which are contributed to an accumulator space, where multiple circles overlap to give the intersection point, which is the center point of the circle of interest (Rizon et al., 2005).

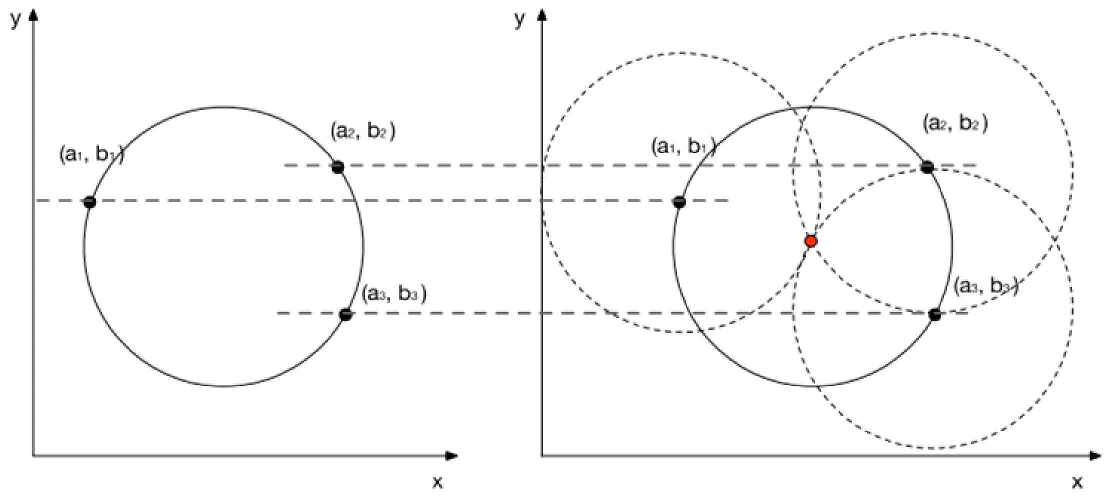


Figure 2: Finding center of circle by multiple arbitrary circle overlap (Rodzi et al., 2017).

To enhance its circle detection capabilities, the Hough Circle Transform is often combined with Canny Edge detection. This integration involves an initial step using the Canny Edge Detector to emphasize edges within the image. This preparatory phase aims to refine the image by accentuating edges and reducing irrelevant information, optimizing it for subsequent circle identification.

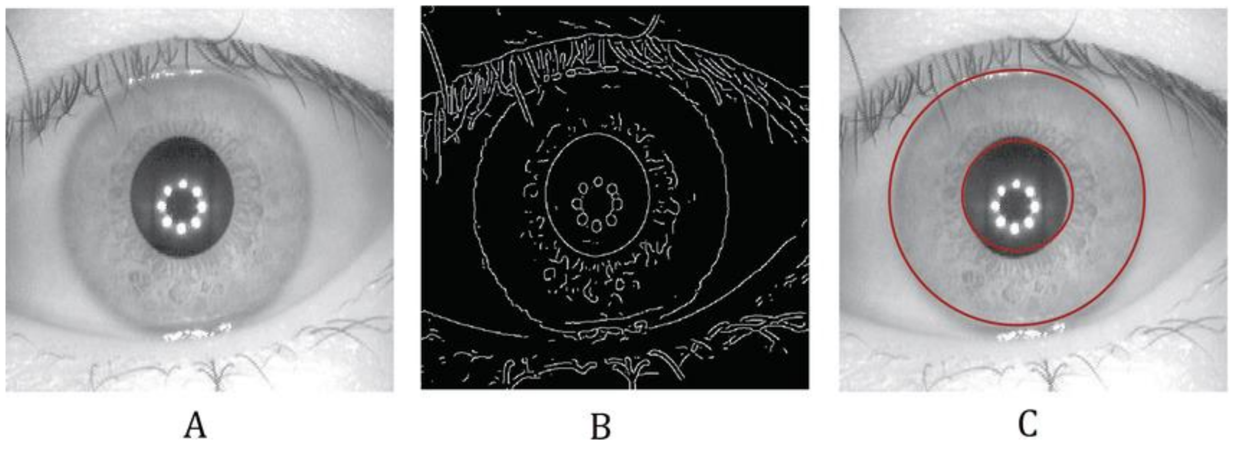


Figure 3: Canny edge detection to Hough Circle Transform (Rana et al., 2019).

Our Hough algorithm, has parameters edges, detected from the OpenCV Canny algorithm, cv2.HOUGH_GRADIENT, dp, the inverse ratio of the accumulator resolution (smaller dp value means a higher resolution in the accumulator), minDist, minimum distance between the centers of detected circles, param1, higher threshold for the edge detection, param2, accumulator threshold for circle detection (smaller values of param2 can result in more circles being detected), and the minRadius and maxRadius, set to 0, to search for the entire range of possible radii in the image.

Notably, the Hough Circle Transform comes with certain limitations that affect its functionality. One such limitation involves the manual definition of parameters, including minimum and maximum circle radii. These parameters significantly influence the algorithm's sensitivity and accuracy in detecting circles. Additionally, the method's sensitivity to noise levels within images poses challenges, as variations in noise can impact the algorithm's ability to accurately identify circles.

Code Implementation:

For the code implementation part, we had two main algorithms that we had to deal with. For the CNN part, even though we got the code from GitHub (Hsouri,2022), we still had to make some adjustments to the code to make it suit our project.

Our main changes were in the dataset.py and validation.py files. The original model was trained on about 200,000 images with high noise. The noise was so high that it was almost impossible for the model to predict a normal circle image with low or no noise. To counter that, we wrote our own *add_salt_and_pepper_noise* function which added noise to the image with the salt and pepper probabilities passed in as parameters to the function, with the default being 0.02, which is what we used for our dataset. The second change in *dataset.py* was to remove the original noise function for the images and add random noise and blur on about 25% of the images. Now *dataset.py* saved the images and their parameters in a csv file for the dataset.

Once the model was trained, we had to change *validation.py* to test the predictions. Originally, it just tested 1000 images and calculated their mean prediction success rate. We changed it to just take the image as a parameter of the *find_circle_cnn* function, which saves the detected image with a red detected circle on top of the original image and outputs the detection circle centre and radius for comparison.

Moving on to the Hough Circle detection, this algorithm was fairly simple to write. We used the OpenCV's Hough Circle function with default parameters and the Canny Edge Detector before that, to make the edges more prominent, so it's easier for the Hough algorithm to detect circles. This code is contained in the *canny_circle_detection.py.* This file also returns the centre and radius of the circle and saves the detected image. In case the algorithm detected more than one circle, we had it return None since our images would contain one circle only.

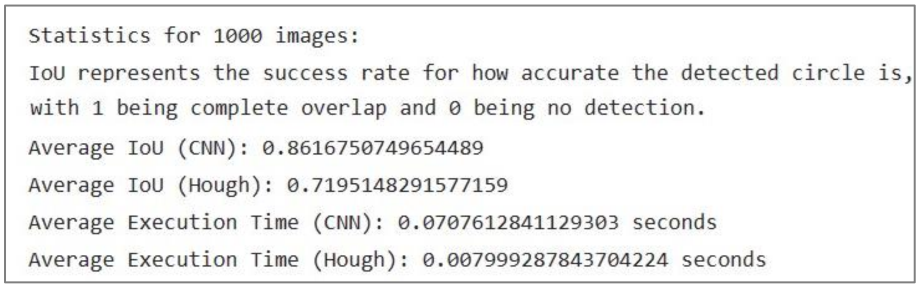
In the main project directory, we have *create_circles.py*, which generates and saves 1000 circles in the folder named 'test'. Then we run the *compare_detection_stats.py* to loop through those 1000 circles and use both Hough and CNN to detect the circles and output the average success rate and execution time for each algorithm. Finally, we created the *detecting_read_circles.py*, which loops thorough the 'real_test' directory and detects and saves the detected circle in the same folder, using both algorithms, for comparison.

Datasets and training:

Our dataset consisted of randomly computer-generated circles which are drawn using the *create_circles.py* in our project repository. To get rid of the 'too much noise problem' mentioned earlier, and to reduce training cost and time consumption, we retrained our CNN model with a noise level of 0.02 for a dataset of 10,000 images, a number small for a machine learning code, with dimensions of 200x200 pixels. It comprised of white backgrounds with randomly positioned black circles with varying properties like size, shapes, and blurriness, further enhanced with varying levels of salt-and-pepper noise. This retraining process spanned approximately 16 hours to enable the CNN to learn and adjust its parameters effectively.

Results and Comparison:

As both the algorithms use the overlapping of the circles to determine the success rate of both of their algorithms to detect circles, we use the Intersection of Union (IoU) property of the result which is a probability of detecting the circle accurately. On the other hand, we are looking at the Average Execution Time (AET) for the time for both algorithms. When we run the function *compare_detection_stats.py* both the algorithms generate the results based on the properties we are using to determine their performance and accuracy. After the completion of the test, a file named stats.txt is created with the results for the properties of both algorithms shown in Fig. 4.



```
Statistics for 1000 images:
IoU represents the success rate for how accurate the detected circle is,
with 1 being complete overlap and 0 being no detection.
Average IoU (CNN): 0.8616750749654489
Average IoU (Hough): 0.7195148291577159
Average Execution Time (CNN): 0.0707612841129303 seconds
Average Execution Time (Hough): 0.007999287843704224 seconds
```

Figure 4: Results on the algorithms from stats.txt file

The results clearly show that for the 1000 computer generated circles, the CNN algorithm has a better probability of accurately detecting the circles. This is because it was trained on over 10,000 images of the same criteria, with varying degrees of noises and blur, and different size and position of the circles. But nonetheless, the circles are in the same environment. The Hough Circle Transform on the other hand, has no learning properties and is basically detecting circles with the base algorithm, has a well performance, although not as thorough as the CNN. The CNN program is seen to achieve 86.2% of intersection with the test circles whereas the Hough Circle Transform scored around 72.0%. If we consider the time taken by each algorithm, we see however that the CNN is disadvantaged to the Hough algorithm. AET for the Hough algorithm is 0.008 seconds which is around 9 times less than what we see for the CNN.

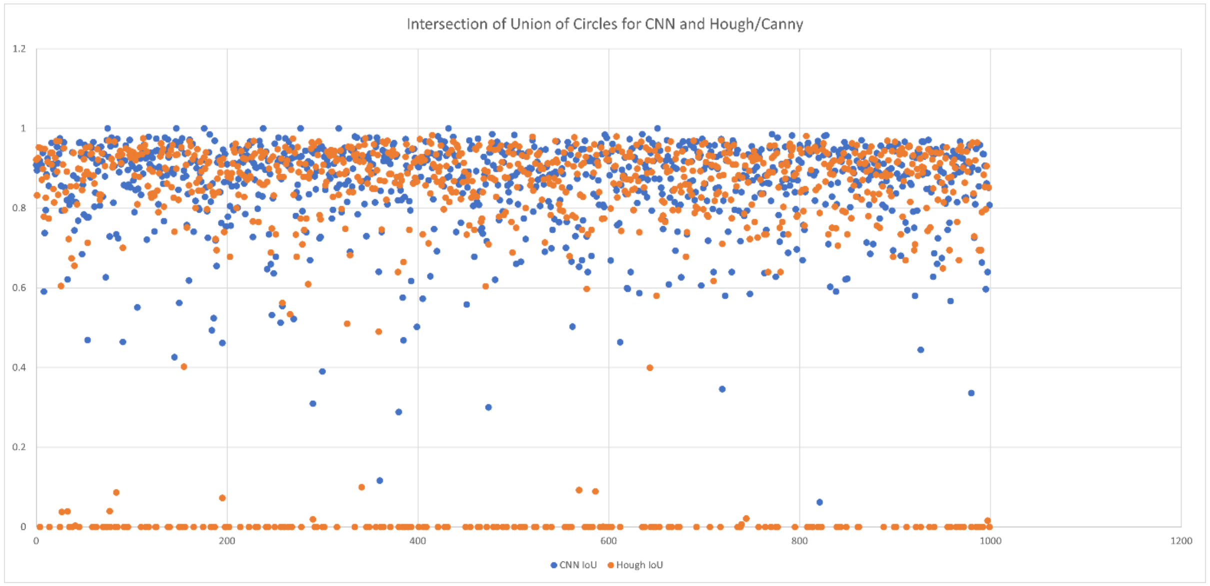


Figure 5: Machine vs non-machine learning test datasets.

Constructively we have also created a graph in Fig. 5 that shows all the instances of the dataset that were created and tested on. The horizontal axis is the number of test circles, and the vertical axis represents the IoU. Here we see for the computer-generated dataset the CNN outperforms the Hough Transform by miles. The Hough Transform has many instances where it made no intersection with the circles in those images, whereas the CNN has no instances where it couldn't make anywhere near the correct positioning of the circle. Due to a sample test on 10,000 images the test results for the CNN are however scattered throughout the graph. But for the Hough detection it's an 'all or none' phenomenon.

The test renders images from which we can compare the effectiveness of the algorithms. The images that we get from the *Test 1*, comprises of the computer-generated circles which yields better results for the CNN than Hough Circle. As we already acknowledge it, we will look at some comparisons where the Hough Circle fails while the CNN prevails. As we see in the first two pictures Fig 6.a and 6.b, they are competitively providing outstanding results on detecting the circles.

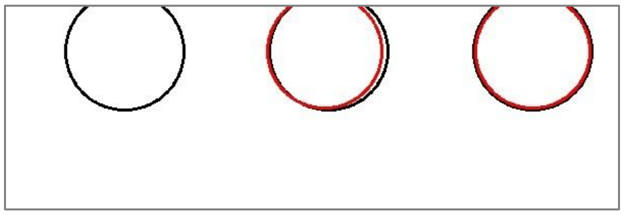


Figure 6.a. CNN(center) vs Hough/Canny(right)

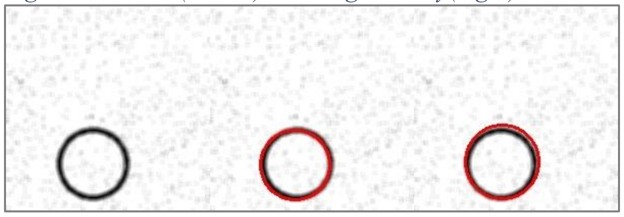


Figure 6.b. CNN(center) vs Hough/Canny(right)

Here we see, amidst the blur, noises and size differences, the Hough transformation performs well. However, in the next few pictures the cases are otherwise.

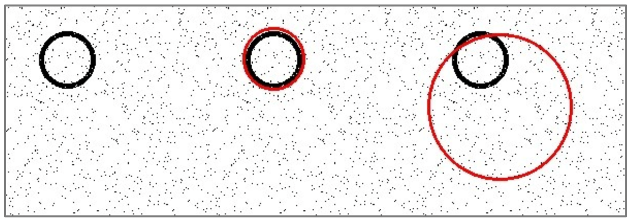


Figure 7.a. CNN(center) vs Hough/Canny(right)

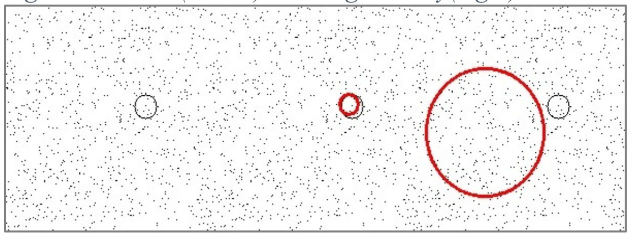


Figure 7.b. CNN(center) vs Hough/Canny(right)

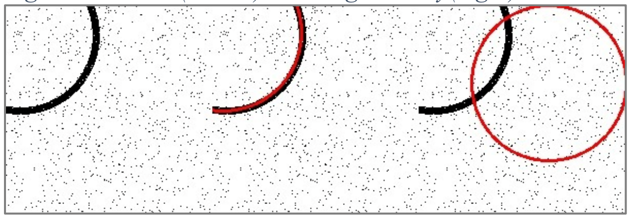


Figure 7.c. CNN(center) vs Hough/Canny(right)

As we can see that these pictures clearly show the constraints of the Hough Transform circle detection. In Fig. 7.a. there is no blurriness, despite which the Hough algorithm couldn't detect the circle. The fact that the Canny Edge algorithm is crucial to edge out the object before the Hough Transform performs, it is evident here that the Canny Edge algorithm couldn't make a discerning circular for the Hough algorithm. The edge in this picture is thick although having specs of white dots unlike the same circle that is smoothened instead in Fig 7.b. As a matter of fact, the edge detection might not have been successful for the whole algorithm to work efficiently.

The Hough algorithm always takes in a fixed range of the radius of the circle, whereas the circle in the Fig 7.b is too small for the Hough algorithm to detect, even though the *min* and *max radius* was set to 0, and the *dp* is set to 1 for our algorithm. For Hough to detect circles it also requires drawing the circle from its center which it calculates from arbitrary circles in the picture. In Fig 7.c. the center of the circle is out of the boundary by a distance greater than the radius of the circle of the picture.

The *Test 2* was primarily focused on detecting the circles in real-life examples/applications. For this test, we took a few 200x200 images of real-life circular objects, for example, a glass bead or a clock. Then we used both CNN and Hough Circles to detect the circles in these images and output the detected circle in a separate image for comparison.

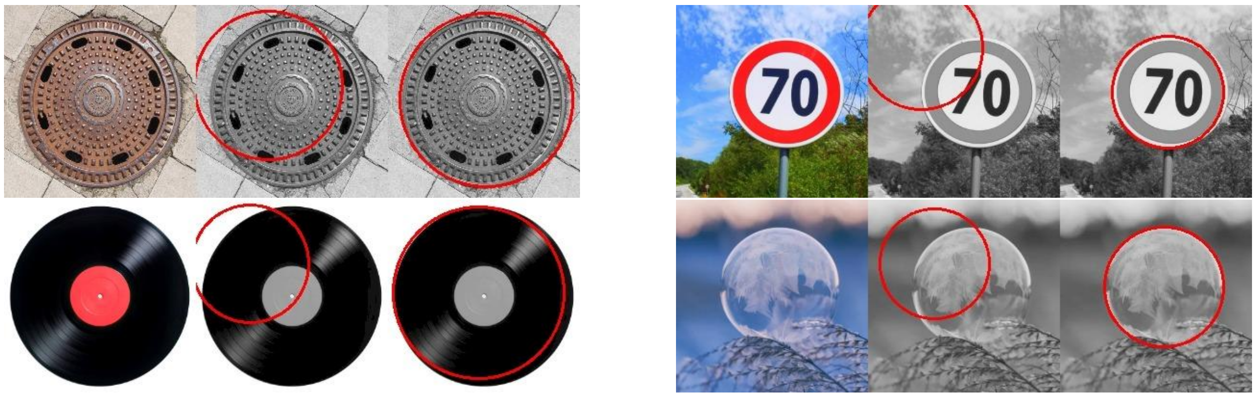


Figure 8: Test of both algorithms, CNN(center) vs Hough/Canny(right), on real circular object images.

In this test, the results were rather surprising we see in Fig.. The CNN algorithm had a hard time detecting circles in any of these images and failed dreadfully, whereas the Hough Circle algorithms, with pretty much the default parameters, did a great job detecting most of these circles. The reason for this is because CNN was trained in a different environment of images. CNN is more accurate when trained on the correct dataset, but when trying to detect general circles, the Hough algorithm is a much better option because it can detect any kind of circles by just changing the parameters a little.

Summary

To sum it all up, from the results of the two tests conducted, we can see that CNN is a powerful algorithm. It can easily detect circles in a noisy and blurry image, with a random radius and center of the circle, if we provide it with a big enough and accurate dataset for it to detect patterns in the images. Even though it sounds like a very good option to detect a circle, it still has a lot of limitations. This brings us to its competitor, the Hough algorithm which though not as accurate as CNN, for the first test at least, is still a powerful tool which allows us to detect multiple circles in an image of any size without the need for any training proving how strong of an algorithm it is, with just about 30 lines of code. CNN should only be used when you are working on very specific applications, e.g. mutated cell, or antibody detection, where the environment is comprised of noise and collateral data in medical imaging. In that case, CNN would be a much better option, because you have a lot of data available, and those circles are not easy to detect with Hough. For much faster applications, like detecting something from a high-speed camera the Hough algorithm would be a much better option because it is

9x faster than CNN. Hough algorithm resembles effectiveness, flexibility, and speed, and can be deployed for general detection purposes.

References:

Hsouri. (2022). *Circle-detection-CNN: A convolutional neural network model for detecting the parameters of the circle present inside of a given image under the presence of noise.* GitHub. https://github.com/hsouri/Circle-Detection-CNN.git

Rana, H. K., Azam, Md. S., Akhtar, Mst. R., Quinn, J. M. W., & Moni, M. A. (2019). A fast iris recognition system through optimum feature extraction. *PeerJ Computer Science*, *5*. https://doi.org/10.7717/peerj-cs.184

Rizon, M., Yazid, H., Saad, P., Md Shakaff, A. Y., Saad, A. R., Sugisaka, M., Yaacob, S., Mamat, M. R., & Karthigaya, M. (2005). Object detection using circular hough transform. *American Journal of Applied Sciences*, *2*(12), 1606–1609. https://doi.org/10.3844/ajassp.2005.1606.1609

Rodzi, A. H., Mohd Zin, Z., & Ibrahim, N. (2017). Eye gaze detection using hough circle transform. *International Journal of Applied Engineering Research 12(24):14452-14459*.

Saha, S. (2018, December 15). *A guide to Convolutional Neural Networks - the eli5 way*. Saturn Cloud Blog. https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/

Yamashita, R., Nishio, M., Do, R. K., & Togashi, K. (2018). Convolutional Neural Networks: An overview and application in Radiology. *Insights into Imaging*, *9*(4), 611–629. https://doi.org/10.1007/s13244-018-0639-9

Contributions:

- Mamun Rashid – Finding resource materials for the algorithms, research on CNN and Hough Circle algorithms. (Report: Abstract, Background, CNN, Hough Circle)
- Md Golam Mahmud Chowdhury– Writing the research proposal and providing implementation ideas, research on results and comparisons. (Report: Results and comparisons, dataset, and training)
- Luckman Qasim – Final check on research prospects, code implementations and dataset training. (Report: Code implementations, summary)