- use OpenCV for image colour identification.  
- use Trackbar function to define the upper and lower limit  
- convert rgb to hsv for accurate colour detection even in various saturation and hue values  
- create an array of all colours lying between upper and lower limit.   
- Threshold the hsv image to get only the selected colour. Create a mask for the colour detection using the inRange function   
- bitwise AND the mask and the original image to isolate only the detected region of the required colour  
- use a ML algorithm like KNN/SVM to classify the detected colour

While there are various algorithms used for colour-based image classification, declaring a single "best" option is challenging. The choice truly depends on **specific factors** like your desired level of **accuracy, real-time constraints, and computational resources**. Here is a breakdown of two prominent options and their suitability for real-time applications:

**1. K-Nearest Neighbours (KNN):**

**Strengths:**

* Efficiency: KNN is a relatively simple algorithm with low computational complexity compared to other methods like CNNs. This makes it faster, allowing for potentially faster classification of images in real-time scenarios.
* No training needed: Unlike methods that require training on large datasets, KNN utilizes existing data points for comparison. This eliminates the need for a separate training phase, saving time and potentially enabling real-time deployment.

**Weaknesses:**

* Curse of dimensionality: KNN's performance suffers with high-dimensional data like large images with many colour channels. This can significantly slow down the classification process in real-time applications.
* Data storage: KNN needs to store all training data points in memory for comparison. This can become memory intensive for large datasets, impacting real-time performance, especially on resource-constrained devices.

**2. Convolutional Neural Networks (CNNs):**

* **Strengths:**
  + **High accuracy:** CNNs have demonstrated superior accuracy in various image classification tasks, including colon-based classification. They can learn complex relationships between colours and features, leading to better performance.
  + **Adaptability:** CNNs can be fine-tuned for specific tasks and can handle high-dimensional data effectively.
* **Weaknesses:**
  + **Computational complexity:** Training and deploying CNNs require high computational resources, making them potentially slower and less suitable for resource-constrained real-time applications compared to KNN.
  + **Training data dependency:** CNNs require large amounts of labelled training data for optimal performance, which can be a challenge to acquire depending on the application.

**Choosing the right algorithm:**

Therefore, the suitability of KNN for real-time applications:

* Is best suited for simpler tasks with limited colour variations and smaller images to minimize the impact of dimensionality.
* Requires careful consideration of data size and available resources to ensure real-time performance is achievable.