Tihan Hackathon - Documentation
Team Charaka

ANIRUDH SRINIVASAN
CS20BTECH11059

CHIRAG MEHTA
AI20BTECH11006

PRANAV BALASUBRAMANIAN
AI21BTECH11023

DISHANK JAIN
AI20BTECH11011

Contents

1 Problem Statement

Detecting and classifying traffic lights for autonomous driving and traffic management is a crucial component of modern intelligent transportation systems. This task involves using computer vision and machine learning techniques to identify and categorize traffic lights in a given environment.

1.1 Objective

The primary goal of detecting and classifying traffic lights is to enable autonomous vehicles to understand and respond to traffic signals, ensuring safe and efficient navigation on the road.

1.2 Problem Statement

Detecting and classifying traffic lights involves two main aspects:

1. **Traffic Light Detection:** Locating the position of traffic lights in the camera's field of view.
2. **Traffic Light Classification:** Identifying the state of each traffic light, i.e., whether it's red, yellow, green, or off.

1.3 Importance

1. **Autonomous Driving:** In autonomous vehicles, detecting traffic lights is critical for obeying traffic laws and ensuring safe interactions with other road users.
2. **Traffic Management:** For traffic management systems, real-time data on traffic light states can be used to optimize traffic flow, reduce congestion, and improve overall road safety.

1.4 Challenges

1. **Variability in Traffic Lights:** Traffic lights come in different shapes, sizes, and styles, making it challenging to design a universal detection and classification system.
2. **Environmental Conditions:** Weather conditions (e.g., rain, fog, snow) and lighting conditions (day/night) can affect the visibility of traffic lights.
3. **Complex Intersections:** At complex intersections with multiple traffic lights, the system must correctly associate each light with the appropriate lane.

1.5 Approaches

1. **Computer Vision:** Use cameras and image processing techniques to detect and classify traffic lights.
2. **Machine Learning:** Implement machine learning models, such as Convolutional Neural Networks (CNNs), to recognize traffic light states.
3. **Sensor Fusion:** Combine data from cameras with data from other sensors like LiDAR and radar for improved accuracy.

1.6 Data Collection

1. Collecting a diverse dataset of annotated traffic light images is crucial for training machine learning models.
2. Data should encompass various traffic light configurations, lighting conditions, and weather scenarios.

1.7 Model Training

1. Train machine learning models to recognize traffic lights and their states. This involves:
2. Data preprocessing (resizing, normalization, etc.).
3. Training a model to detect traffic light locations (object detection).
4. Training a model to classify the state of each detected traffic light.

1.8 Real-Time Operation

1. In practice, the system should operate in real-time to provide timely responses to changing traffic conditions.
2. Efficient algorithms and hardware acceleration may be necessary to achieve low-latency performance.

1.9 Integration

Integrating the traffic light detection and classification module into the overall autonomous driving or traffic management system is essential for making use of the detected information.

1.10 Safety Considerations

Failures in traffic light detection can have severe safety implications. Redundancy, validation, and fallback mechanisms are crucial to ensure safe operation.

1.11 Regulatory and Ethical Considerations

Compliance with local traffic laws and regulations, as well as ethical considerations regarding privacy and data handling, are paramount.

In summary, detecting and classifying traffic lights for autonomous driving and traffic management is a complex but vital task for ensuring road safety, efficient traffic flow, and the success of autonomous vehicle deployments. It relies on a combination of computer vision, machine learning, and real-time data processing technologies.

2 Detection

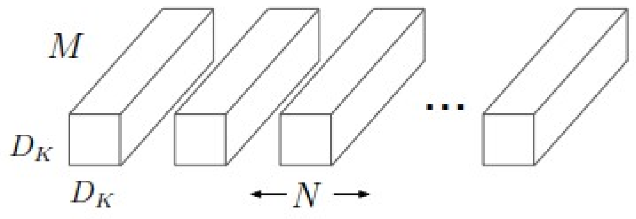
MobileNetV1, also known simply as MobileNet, is a deep learning architecture specifically designed for efficient on-device vision applications. It was introduced by Google researchers in the paper titled "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications." MobileNetV1 is part of the MobileNet family of models and is well-known for its lightweight and computationally efficient design, making it suitable for deployment on resource-constrained devices like mobile phones and embedded systems. Here's an explanation of the key characteristics and components of MobileNetV1:

2.1 Depthwise Separable Convolution

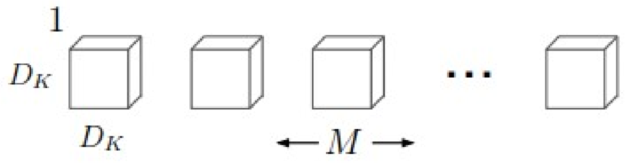
MobileNetV1's core innovation is the use of depthwise separable convolution layers. This convolutional operation consists of two main steps:

1. **Depthwise Convolution:** This step applies a separate convolution operation for each channel (depth) of the input feature map, reducing computational cost.

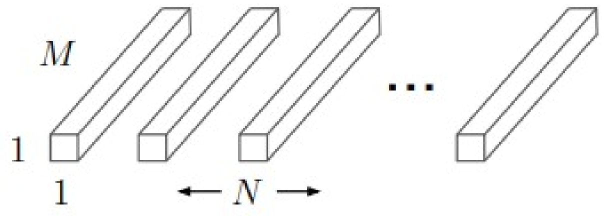
2. **Pointwise Convolution:** After depthwise convolution, a 1x1 convolution layer (pointwise convolution) is applied to combine information across channels. This helps in capturing complex features and interactions.



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Figure 2. The standard convolutional filters in (a) are replaced by two layers: depthwise convolution in (b) and pointwise convolution in (c) to build a depthwise separable filter.

2.2 Width Multiplier and Resolution Multiplier

MobileNetV1 introduces two hyperparameters that allow you to control the model's size and computational complexity:

1. **Multiplier (α):** Scales the number of filters (channels) in each layer. Smaller values of α reduce the model's size and computational cost.

2. **Resolution Multiplier (ρ):** Scales the input image resolution. Lower values of ρ result in smaller input images and further reduce computation.

2.3 Model Architecture

1. MobileNetV1 typically consists of a series of depthwise separable convolution layers interleaved with pointwise convolution layers.
2. It starts with a standard 3x3 convolution layer as the initial layer.
3. The depthwise separable convolution layers are grouped into multiple blocks, and the number of filters in each block is determined by the width multiplier.

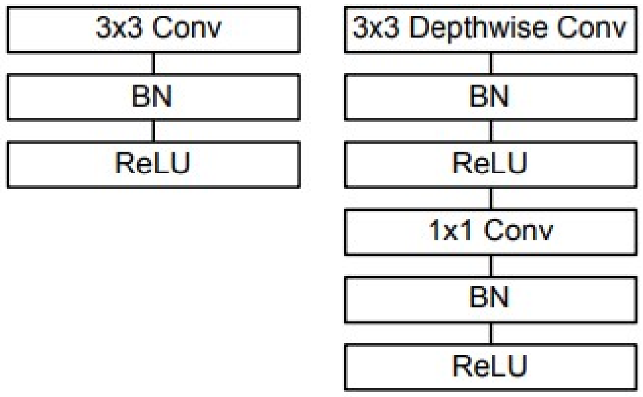


Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

2.4 Efficiency and Speed

1. MobileNetV1 is designed to be highly efficient, with a relatively small number of parameters compared to traditional CNNs.
2. The model is optimized for real-time or near-real-time inference on mobile and embedded devices.
3. Its efficiency makes it suitable for a wide range of computer vision tasks, including image classification, object detection, and semantic segmentation.

2.5 Trade-offs

1. While MobileNetV1 excels in efficiency, it may sacrifice some accuracy compared to larger and more complex CNN architectures.

2. The choice of α and ρ impacts the trade-off between model size and accuracy.

In summary, MobileNetV1 is a lightweight convolutional neural network architecture designed for efficient on-device vision applications. Its depthwise separable convolution and hyperparameters like width multiplier and resolution multiplier make it highly adaptable to various computational constraints while maintaining reasonably good performance for a range of computer vision tasks.

3 Color Classification

We have used a simple method to find the color of a traffic light using the HSV color space. HSV stands for Hue, Saturation, and Value, and it is a color representation that separates color information from brightness and saturation.

3.1 Convert Image to HSV

The first step is to convert the input image from its original color space (typically RGB) to the HSV color space. HSV representation separates the color information (hue) from brightness (value) and saturation (saturation). This separation makes it easier to work with and analyze colors.

3.2 Use High Saturation Pixels

1. Saturation (S) represents the vividness or intensity of a color. High saturation values indicate vibrant and pure colors, while low saturation values represent shades of gray or desaturated colors.

2. In this step, you filter the pixels in the image to select only those with high saturation values. The exact threshold for what constitutes ”high” saturation may vary depending on the image and lighting conditions. A typical threshold might be around 0.5 to 0.7, but it can be adjusted based on the specific requirements of your application.

3.3 Map Hue Values to a Color

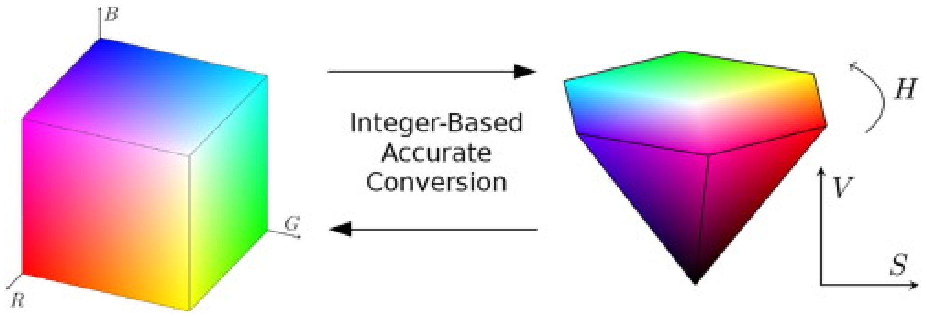
1. After isolating the pixels with high saturation, you can use the hue values (H) of these pixels to determine the color of the traffic light.

2. The hue values in the HSV color space are represented as a continuous circular spectrum, where:

    - Red hues are close to 0 (or 1 if the range is normalized to [0, 1]).
    - Yellow hues are between red and green.
    - Green hues are around 0.33 (or 120 degrees in degrees representation).
    - Blue hues are between green and red.

    - The hue values loop back from 1 to 0, creating a circular spectrum.

3. To map the hue values to specific traffic light colors, you can define ranges or thresholds. For example:
    - If hue falls within a small range around 0 (or 1), you can classify it as red.
    - If hue falls within a range between red and green, it's yellow.
    - If hue falls within a range around 0.33, it's green.
    - All other hue values may be considered non-traffic light objects or undefined.



3.4 Traffic Light Color Identification

1. Analyze the hue values of the high-saturation pixels to determine which color category they fall into based on the defined hue ranges.

2. You can count the number of pixels within each hue range to identify the dominant color.

3.5 Output

The output of this algorithm is the detected color of the traffic light, which can be one of the predefined categories: red, yellow, green.

It's important to note that this algorithm is a simplified approach to traffic light color detection and may not work perfectly in all scenarios. Factors such as lighting conditions, image quality, and the presence of other objects can affect its accuracy. More advanced algorithms, like those used in autonomous vehicles, often incorporate additional techniques and machine learning for robust traffic light detection and recognition in real-world environments.

4 References

1. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications

2. LISA Traffic Light Dataset