Smart Edge Lens and Cloud Optimization - Methodology 1 - Road Traffic Congestion Detection Using K-means Clustering and CNN-VGG16

1st Tejmul Movin
Department of Cs/Ai
Rishihood University, Sonipat, India
sansar.t23csai@nst.rishihood.edu.in

2nd Sahil Sarawgi
Department of Cs/Ai
Rishihood University, Sonipat, India
sahil.s23csai@nst.rishihood.edu.in

3rd Abhishek Meena
Department of Cs/Ai
Rishihood University, Sonipat, India
abhishek.m23csai@nst.rishihood.edu.in

***Abstract*—This paper presents Methodology 1 for traffic congestion classification using a two-stage hybrid approach combining K-means clustering for congestion labeling and Convolutional Neural Network (CNN) with VGG-16 architecture for spatial feature extraction. The methodology processes 4,500 optimized video frames to classify traffic density into three levels: Low, Medium, and High. Frame pooling through CNN-VGG16 reduces spatial dimensions while preserving critical traffic information. The approach achieves approximately 60% accuracy in congestion level classification. The methodology provides a supervised learning framework suitable for predefined traffic condition detection and real-time traffic monitoring applications.**

***Index Terms*—K-means Clustering, CNN, VGG-16, Traffic Congestion, Classification, Video Analysis**

I. Introduction

Traffic congestion is a critical challenge in urban transportation systems, affecting public safety, economic efficiency, and environmental quality. Real-time identification and classification of traffic density levels enables proactive traffic management and dynamic routing optimization. Traditional manual monitoring is labor-intensive and inefficient. This paper presents Methodology 1, employing K-means clustering for congestion labeling combined with CNN-VGG16 architecture for automated spatial feature extraction and congestion classification from video data.

II. Data Preprocessing and Preparation

Road traffic video data was collected from multiple road segments under varying congestion levels and lighting conditions. The raw video streams were first decoded into image frames at a fixed sampling rate. Approximately 15,000 frames were initially extracted from the complete video dataset.

To make the pipeline feasible on limited computational resources (GPU memory and training time) while still preserving the evolution of traffic patterns, a two-stage frame quantization and temporal sampling strategy was adopted.

A. Two-Stage Frame Quantization

The main objective of quantization was to reduce the number of frames while retaining frames that adequately represent changes in congestion, such as build-up, dissipation, or sudden density changes.

1) Stage 1: Uniform Temporal Down-Sampling: In the first stage, a simple uniform sampling was performed by selecting every second frame from the original sequence. If N_0 denotes the total number of extracted frames, the number of frames after this stage is approximately

$$N_1 \approx \frac{N_0}{2}. \tag{1}$$

For $N_0 \approx 15{,}000$, this results in about 7,500 frames.

This step removes highly redundant consecutive frames that differ only slightly, while maintaining temporal continuity of the traffic flow. Since traffic congestion typically changes over seconds rather than milliseconds, halving the temporal resolution does not significantly affect congestion-related information.

2) Stage 2: Odd-Indexed Frame Selection: In the second stage, only odd-indexed frames from the already down-sampled sequence were retained. Conceptually, this is equivalent to keeping one frame out of every two in the N_1 sequence:

$$N_2 \approx \frac{N_1}{2} \approx \frac{N_0}{4}. \tag{2}$$

From the original 15,000 frames, the final set contains approximately 4,500 frames.

Selecting odd-indexed frames helps to avoid potential bias toward the very beginning of short temporal segments and further reduces redundancy. At this point, the effective temporal sampling rate is reduced to roughly one out of every four original frames, which still captures congestion trends (e.g., transition from free-flow to heavy congestion) while achieving nearly 70% reduction in data volume.

Overall, the two-stage quantization:

- Reduces the number of frames from 15,000 to 4,500 (approximately 30% of the original),

- Preserves representative frames spanning different congestion states,
- Makes training and inference with VGG-16 practical on modest hardware.

B. Spatial Preprocessing and Normalization

After temporal quantization, each of the 4,500 selected frames was preprocessed to ensure compatibility with the CNN-VGG16 architecture and to improve training stability:

- **Resolution Standardization:** All frames were resized to a fixed spatial resolution of 224 $\times$ 224 pixels, matching the input size expected by VGG-16. This step ensures consistent input dimensions across the dataset.
- **Color Space Conversion:** Frames were converted to three-channel RGB format to align with the pre-trained ImageNet VGG-16 weights, which are defined on RGB images.
- **Pixel Normalization:** Pixel values were scaled and normalized using ImageNet mean and standard deviation statistics. If I denotes a pixel intensity and μ and σ denote the per-channel mean and standard deviation, the normalized value $\hat{I}$ is given by:

  $$\hat{I} = \frac{I - \mu}{\sigma}. \tag{3}$$

  This normalization improves convergence and reduces sensitivity to illumination variations.

C. Noise Reduction and Visual Enhancement

To further enhance the robustness of feature extraction, light-weight enhancement operations were applied:

- **Mild Gaussian smoothing** was used to reduce high-frequency sensor noise without blurring critical structures such as vehicle edges and lane markings.
- **Brightness and contrast adjustments** were applied when necessary to compensate for underexposed or overexposed frames, especially during dawn, dusk, or harsh sunlight.
- For particularly low-light scenes, **histogram equalization** was applied to improve visibility of vehicles and road boundaries.

These steps yielded visually consistent and enhanced frames that are better suited for downstream K-means clustering and CNN-based feature extraction.

D. Final Prepared Dataset

After the two-stage quantization and preprocessing pipeline, the final dataset consists of approximately 4,500 spatially normalized and visually enhanced frames. This dataset achieves a balance between:

- **Information preservation**, by retaining key temporal variations in congestion,
- **Computational efficiency**, by reducing the volume of data to a manageable size for VGG-16 training and inference.

These prepared frames form the input to Stage 1 (K-means clustering for congestion labeling) and Stage 2 (CNN-VGG16-based supervised learning), as described in the subsequent sections.

III. Methodology

A. Stage 1: K-means Clustering for Congestion Labeling

K-means clustering partitions traffic frames into three distinct congestion categories based on visual features:

$$J = \sum_{i=1}^{k} \sum_{j=1}^{n_i} ||x_j^{(i)} - \mu_i||^2 \tag{4}$$

where $k = 3$ clusters represent Low, Medium, and High congestion levels, $x_j^{(i)}$ are data points, and μ_i are cluster centroids.

Process:

- Extract initial features from 4,500 frames (vehicle count, lane occupancy, density metrics)
- Initialize k=3 cluster centers randomly
- Iteratively assign frames to nearest centroid
- Update centroids until convergence
- Map clusters to congestion levels based on feature characteristics

K-means efficiently partitions frames without requiring extensive manual annotation, enabling unsupervised pre-labeling of training data for subsequent supervised learning stages.

B. Stage 2: CNN-VGG16 for Spatial Feature Extraction

VGG-16 is a deep convolutional neural network with 16 weighted layers optimized for image classification. Architecture consists of five convolutional blocks with max-pooling layers:

VGG-16 Configuration:

- **Input:** RGB frames (224×224 pixels)
- **Convolutional Blocks:** 5 blocks with 3×3 filters, increasing depth (64→512 channels)
- **Pooling:** 2×2 max-pooling reduces spatial dimensions progressively
- **Fully Connected Layers:** 3 dense layers for classification (4096→4096→3 neurons)
- **Output:** Probability distribution across three congestion classes

Feature Pooling Mechanism: VGG-16 performs hierarchical feature extraction through spatial pooling:

$$\text{Pool}_{22}(L) = \max_{i,j \in \text{window}} L_{i,j} \tag{5}$$

This pooling operation aggregates local spatial features into broader context representation, capturing vehicle patterns, lane characteristics, and traffic density at multiple scales. Progressive pooling reduces feature maps from 224×224 to 1×1 while maintaining semantic information critical for congestion classification.

C. Training Pipeline

Data Preparation:

- K-means generated pseudo-labels for 4,500 frames (Low/Medium/High)
- Train-validation-test split: 70%-15%-15%
- Data augmentation: rotation, brightness adjustment, horizontal flip
- Frame normalization using ImageNet statistics

Model Training:

- Transfer learning from ImageNet pre-trained VGG-16 weights
- Fine-tune final fully connected layers (3-class output)
- Loss function: Cross-entropy with softmax activation
- Optimizer: Adam with learning rate 0.001
- Batch size: 32, Epochs: 50
- Early stopping based on validation accuracy

IV. Results and Performance

The methodology achieved approximately 60% accuracy on test frames with three-class congestion classification:

TABLE I
Classification Performance

Class	Precision	Recall	F1-Score
Low Congestion	0.65	0.58	0.61
Medium Congestion	0.58	0.62	0.60
High Congestion	0.62	0.64	0.63
Overall	**0.62**	**0.61**	**0.61**

Confusion matrix analysis reveals improved performance on extreme congestion levels (High/Low) with moderate accuracy on Medium congestion. Performance limitations stem from class imbalance in K-means pre-labeling and overlapping feature characteristics between Medium and Low congestion.

V. Advantages and Limitations

Advantages:

- Predefined output classes (Low/Medium/High) suitable for traffic management decision-making
- Transfer learning from ImageNet reduces training data requirements
- Efficient spatial pooling captures multi-scale traffic patterns
- Computationally efficient inference for real-time applications
- Well-established CNN architecture with proven reliability

Limitations:

- Requires manual verification and correction of K-means pseudo-labels
- Limited to three predefined congestion classes
- 60% accuracy insufficient for critical safety applications
- Local convolutional features may miss global context
- Struggles with Medium congestion classification due to ambiguous boundaries
- Cannot detect novel anomalies beyond predefined classes

VI. Computational Considerations

VGG-16 with 138 million parameters requires moderate computational resources. With 4,500 frames, total processing time approximately 8-10 seconds on standard GPU. Inference optimization through quantization and pruning can reduce deployment requirements. Frame reduction (70% compression) enables edge device deployment for traffic monitoring systems.

VII. Comparative Perspective

Methodology 1 provides a supervised baseline for congestion classification with interpretable outputs. However, limitations in accuracy and flexibility motivate development of unsupervised approaches (Methodology 2) that eliminate labeling requirements and detect novel anomalies beyond predefined classes.

VIII. Conclusion

Methodology 1 presents a supervised hybrid approach combining K-means clustering and CNN-VGG16 for traffic congestion classification. The two-stage pipeline achieves 60% accuracy in three-class congestion detection. While effective for predefined traffic monitoring, limitations in accuracy and anomaly detection capability indicate need for advanced methodologies. Future work includes improving class imbalance through weighted loss functions, exploring attention mechanisms for global context, and hybrid approaches combining supervised and unsupervised learning.

References

[1] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in Proc. Int. Conf. Learn. Represent., 2015, pp. 1–14.

[2] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data clustering: A review,” ACM Comput. Surv., vol. 31, no. 3, pp. 264–323, 1999.