Journal of Scientific & Industrial Research
Vol. 83, September 2024, pp. 989-1000
DOI: 10.56042/jsir.v83i9.7388

CapsNet-based Precise and Rapid Traffic Sign Detection through AI in Adverse Environmental Scenarios

Ravinder Kaur[1], Jitendra Singh[1]* & Swati Sharma[2]

[1]Department of Computer Science and Engineering, Faculty of Engineering and Technology, SRM Institute of Science and Technology, Delhi - NCR Campus, Delhi - Meerut Road, Modinagar, Ghaziabad, Uttar Pradesh 201204, India

[2]Department of Information Technology, Meerut Institute of Engineering and Technology, Meerut, India

Received 18 December 2023; revised 18 April 2024; accepted 30 August 2024

This research presents an innovative system for real-time recognition of traffic signs, leveraging Artificial Intelligence (AI) to achieve high-performance identification, particularly under challenging conditions prevalent in traffic scenarios such as occlusions, atmospheric haze, and noise. The proposed system employs an advanced Capsule Network (CapsNet) algorithm for object recognition, trained extensively on a diverse dataset encompassing images of various traffic signs. Notably, the system demonstrates concurrent detection capabilities for multiple traffic signals, accurately categorizing them based on their respective classes. To address distortions caused by alterations in the camera's perspective, including rotation, torsion, and elongation, the system effectively employs techniques ensuring precise alignment with the pertinent traffic sign. Furthermore, pre-processing techniques are utilized to resolve ambiguity and distortions in complex traffic scenarios. Empirical validation of the proposed methodology is conducted through experimentation with authentic traffic sign images obtained from diverse environmental contexts. Comparative assessments across diverse datasets representing prominent traffic sign domains affirm the efficacy of the proposed approach. The outcomes showcase a noteworthy precision level, achieving a recognition accuracy of 99.16% for traffic signs. In contrast, conventional rule-based systems under identical conditions exhibit accuracy rates ranging between 80–90%. The AI-driven system demonstrates real-time operational feasibility, positioning it as a fitting candidate for applications in traffic management and intelligent transportation systems.

Keywords: Autonomous driving vehicles, Artificial intelligence, Intelligent transportation, Neural networks, Traffic sign detection

Introduction

As the global population continues to expand, there has been a substantial surge in the vehicular populace on roadways. Regrettably, this escalation in traffic volume has concurrently engendered an upswing in vehicular accidents and ensuing fatalities. The causative factors behind these accidents encompass various influences, including driver fatigue, substance abuse, and suboptimal road conditions.[1] In response to this predicament, the research and engineering communities have diligently endeavored to devise remedies capable of mitigating accident risks while enhancing the overarching safety of vehicular operation.[1,2] Notably, autonomous driving systems have materialised as a pivotal focal point within the automotive domain, bearing the goal of augmenting road safety and driving comfort.[3,4] Sinha and Umrao (n.d.). These systems are devised to assist drivers and vehicles in expeditiously discerning and effectively reacting to unsafe traffic scenarios.[5] Among many technology choices, camera-based solutions stand out as cost-effective. They benefit from the fast advancements in computer vision techniques designed for analyzing and altering images.[6,7] The acquisition of environmental insights pertinent to vehicular operation transpires via external environment sensing, a procedure executed externally to the vehicular apparatus. Concomitant objectives encompass identifying proximate vehicles, pedestrians, traffic signage, traffic luminance apparatus, and sundry ancillary entities ties.[8]

However, the accuracy of collected environmental data can be affected by various factors, including weather conditions, road conditions, and lighting conditions. Ergo, the design of these systems necessitates deliberate contemplation of these multitudinous variables. It's crucial to implement the system on hardware that's fast and precise. This is important because the identified objects need to be

*Author for Correspondence
E-mail: jitendrs@srmist.edu.in

communicated in real-time to both the operator and the system.[9] Conventionally, the preponderance of conducted research has hinged upon computational substrates, predominantly situated within the precincts of computer architectures, as opposed to transpiring upon mobile platforms, thereby-gendering constraints on the mobility and portability of the system.[10] While the deployment of mobile Graphical Processing Unit (GPU) platforms proffers escalated performance benchmarks at judicious fiscal outlays.[11], reliance upon a smartphone-centric paradigm characterized by constrained capabilities may potentially compromise the aspired echelons of performance efficacy and precision.

The evolution of autonomous driving technology has witnessed a pronounced surge in the adoption of deep learning paradigms for traffic sign detection.[8] Deep learning, an enclave within the realm of machine learning, entails training artificial neural networks to acquire the ability to discern and assimilate patterns within data.[9] In the realm of traffic sign detection, recent studies[12] have shown that deep learning methods excel in quickly and accurately detecting traffic signs.[10,13] Deep learning techniques demonstrate the inherent competence to autonomously extract salient features from unprocessed data, notably images, obviating the necessity for manual feature engineering.[9,14,15] This attribute renders them inherently suitable for the agencies of traffic sign detection, given their capacity to internalize intricate visual attributes such as colour, contour, and textual content, variables that can diversify across distinct sign categories and fluctuating luminance conditions.[16] Among the deep learning architectures, Convolutional Neural Networks (CNNs) are prominent exemplars extensively harnessed in traffic sign detection efforts.[17] CNNs are adept at discerning features spanning a spectrum of abstraction levels, encompassing rudimentary attributes like edges and vertices to sophisticated constructs such as shapes and entities.[14] This quality aptly equips them for detecting traffic signs spanning diverse spatial dimensions and orientations. A cardinal challenge in the domain of traffic sign detection resides in surmounting occlusion scenarios, wherein intervening entities might partially or entirely obscure traffic signs. Deep learning methodologies have been marshalled to navigate this challenge, resorting to region-centric methodologies that concentrate on the most visually accessible segments of the sign.[18,19] Furthermore, augmentation strategies that manipulate the data, including cropping and mirroring, can be judiciously employed to engender augmented training datasets, thereby enhancing the model's resilience in coping with occlusion-induced perturbations. Integrating deep learning techniques into the landscape of traffic sign detection has exhibited auspicious prospects for bolstering the accuracy and dependability of autonomous driving systems. As the domain continues to evolve, the ascendancy of deep learning approaches is poised to intensify, conferring a pivotal role in advancing secure and efficient autonomous vehicular operation.

Despite their commendable efficacy, Convolutional Neural Network (CNN)-based algorithms for object detection exhibit certain constraints when employed in traffic sign detection.[20] Among these limitations is the considerable demand for substantial quantities of annotated data for training. This prerequisite can entail substantial time investments and financial outlays to amass.[21] Moreover, these algorithms may impede accurately identifying traffic signs that experience partial occlusion or concealment by surrounding entities such as vegetation, towers, or vehicular obstructions.[22,23] This susceptibility to occlusion scenarios can engender false negatives, whereby the system fails to successfully detect a traffic sign within the visual field.[24] A further limitation manifests in the computational intensiveness of CNN-based algorithms. This facet can potentially hamper real-time detection capabilities, particularly when deployed on resource-constrained platforms such as embedded systems or mobile devices.[25]

Capsule Networks (CapsNet) represent a contemporary paradigm within deep learning architectures, exhibiting notable promise across diverse domains, including traffic sign detection.[26] CapsNet introduces advancements vis-`a-vis the conventional Convolutional Neural Networks (CNNs) across multiple dimensions.[27] Principally, CapsNet excels in accommodating rotational and positional variances inherent in images, a pivotal attribute in traffic sign detection scenarios wherein signs may manifest diverse orientations and positions.[28] This proficiency is attributed to the introduction of a novel neural entity termed a "capsule," adept at encapsulating multifaceted attributes of an object, encompassing attributes like spatial disposition,

alignment, and proportions.[29,16] Furthermore, CapsNet demonstrates the capacity to accommodate inputs of variable lengths, a trait that proves advantageous when detecting signs exhibiting varying geometries and dimensions. In contrast, CNNs mandate inputs of fixed dimensions, potentially incurring information degradation or distortion.[28] Conclusively, CapsNet attains efficiency gains relative to CNNs regarding parameter utilisation, which is requisite for achieving elevated accuracy levels. This efficiency stems from the incorporation of dynamic routing within CapsNet, an operative framework facilitating the discernment of inter-capsule relationships, thereby curbing network redundancies.[12,30] Notwithstanding these advancements, like other machine learning paradigms, CapsNets are not impervious to imperfections and exhibit avenues for refinement.[30] Herein, we proffer prospective strategies for elevating the efficacy of Caps Nets:

1. **Data Augmentation:** Capsule Networks (CapsNets) necessitate an extensive corpus of training data to attain heightened accuracy levels. However, procuring a substantial dataset can be beset with challenges or fiscal constraints.[31] Data augmentation methodologies furnish a strategy to artificially amplify the training dataset's dimensions by applying diverse transformations, encompassing rotations, scaling, and mirroring, to the existing images. This augmentation regimen enhances CapsNets' capacity to internalise more robust and transferable features.
2. **Dynamic Routing Algorithm:** The routing algorithm operational within CapsNets constitutes a pivotal constituent that governs the dissemination of information amidst capsules.[30] Though efficacious, the prevailing iterative routing-by-agreement algorithm can exert computational intensiveness and temporal latency.[32] Contemporary investigations are probing alternate routing algorithms to maintain or surpass prevailing performance metrics while ameliorating computational demands.[33,34]
3. **Ensemble Learning:** Ensemble learning embodies a methodology that amalgamates diverse machine learning models to elevate performance benchmarks synergistically. CapsNets can be amalgamated with other neural network archetypes.[12] or machine learning frameworks to engineer more resilient and precise models.
4. **Hyperparameter Tuning:** The efficacy of CapsNets can evince sensitivity concerning the selection of hyperparameters, encompassing learning rate, batch size, and regularisation intensity. Techniques of hyperparameter tuning, spanning grid search or Bayesian optimisation can be harnessed to ascertain the optimal hyperparameters encapsulating CapsNets' functionality.[35]
5. **Transfer Learning:** Transfer learning constitutes a strategy by which pretrained models undergo fine-tuning for a novel task with limited data availability. CapsNets can undergo pre-training on expansive datasets like ImageNet, which is succeeded by fine-tuning a more circumscribed dataset germane to the specific task. This augmentation strategy serves to elevate CapsNets' performance within data-scarce contexts.

Background and Related Work

Traffic sign recognition has witnessed transformative advancements, predominantly fueled by the rise of deep learning techniques. This section elucidates key contributions that have delineated the trajectory of this domain, encapsulating architectural innovations and the overarching challenges that underscore the research landscape.

Guo *et al.,* present an integrated approach combining connected and automated vehicles (CAVs) with urban traffic signal control systems.[36] Their study meticulously examines the potential synergies and challenges of integrating CAVs to enhance traffic signal control and optimize urban traffic flow. The research explores various aspects, including communication protocols, coordination strategies, and optimization algorithms, effectively showcasing the potential of CAVs. The authors highlight the benefits of this integration, such as reduced congestion, improved safety, and minimized environmental impact. Additionally, they underscore the necessary considerations and challenges that must be addressed to realize these benefits.[36]

Addressing Cybersecurity vulnerabilities, infrastructure requisites, and harmonisation with conventional vehicular components, this comprehensive analysis provides a foundational framework for researchers, policymakers, and practitioners yet invites further exploration into the intricacies of real-world deployment, case studies, and future CAV-centric traffic signal control systems trajectories.

Capsule networks' unique capabilities are utilized to detect traffic signs in real-time.[37] Addressing the imperative need for swift and accurate traffic sign recognition, capsule networks are harnessed to

encapsulate hierarchical relationships and spatial configurations of objects. The study's distinctive focus on real-time detection reinforces its relevance in practical scenarios. Empirical validations affirm the proposed capsule network architecture's efficacy in achieving precise detection and classification of traffic signs, asserting its superiority over existing methods. Notwithstanding these achievements, the study would benefit from a more comprehensive exploration of capsule network limitations, particularly within traffic sign detection. Additionally, providing deeper insights into the dataset, computational complexities, and potential deployment challenges would contribute to a more nuanced understanding of the proposed approach's strengths and constraints.

In hyperspectral image classification, Arun *et al*. introduce a novel spatial-spectral classifier based on CapsuleNet architecture.[38] By integrating capsule networks, capable of capturing intricate hierarchical features and spatial relationships within image data, the authors endeavour to harness the synergistic potential of both spatial and spectral information. The study's empirical validation demonstrates the superior performance of the CapsuleNet-based classifier compared to conventional methods and contemporary benchmarks. While this achievement highlights the architecture's effectiveness, a more exhaustive explanation of its computational complexity and training requisites is warranted. Moreover, delving deeper into potential constraints, such as sensitivity to hyperparameters and handling large-scale hyperspectral datasets, would yield a more comprehensive understanding of the proposed method's applicability in hyperspectral image classification.

Present a novel strategy to enhance protein gamma-turn prediction by integrating Inception capsule networks.[39] The study focuses on accurately identifying gamma-turns, which have pivotal implications in protein structure prediction and functional analysis. By employing Inception capsule networks, the authors harness the architecture's capability to discern intricate hierarchical features and spatial relationships inherent within protein sequences. Empirical evaluations underscore the superiority of the proposed approach over traditional methods and state-of-the-art techniques. However, further exploration of capsule network limitations and an in-depth analysis of computational complexity and training demands would contribute to a comprehensive assessment of the proposed method's efficacy in protein structure analysis.

A capsule network is proposed by Kumar (2018) for detecting traffic signs.[40] The authors leverage this architecture's unique capability to capture hierarchical relationships and spatial configurations intrinsic to traffic sign attributes. Empirical validations demonstrate that the capsule network-based model significantly outperforms conventional convolutional neural networks (CNNs) and other contemporary methods in detecting and classifying traffic signs across various scenarios. However, the study does not provide an in-depth discussion of the limitations of capsule networks in the context of traffic sign detection. [40]

Moreover, providing additional insights into the dataset, computational complexities, and potential challenges in real-world deployment would augment understanding of the proposed model's strengths and limitations. Furthermore, a broader validation encompassing larger and more diverse datasets and a comparative analysis against existing approaches would contribute to a comprehensive assessment of the proposed model's performance.

The intersection of deep neural networks and network traffic classification within the realm of robot communication is addressed by Ge *et al*.[41] The authors navigate the challenge of effectively classifying network traffic to facilitate efficient and secure communication in robotic systems. Deep neural networks are harnessed to capture intricate patterns and features within network traffic data. Empirical validations affirm the proposed approach's efficacy in classifying diverse network traffic types, from control messages to sensor data and video streams.[42] While showcasing the architecture's performance, the study would benefit from an expanded discussion of deep neural network limitations within real-world scenarios. A deeper analysis of computational requirements and training considerations would enrich an understanding of the proposed approach's potential and limitations in robot communication.

Ge *et al.,* explore the intersection of deep neural networks and network traffic classification within the realm of robot communication.[41] The authors tackle the challenge of effectively classifying network traffic to enable efficient and secure communication in robotic systems. By utilizing deep neural networks, they capture intricate patterns and features within network traffic data. Empirical validations confirm

the efficacy of the proposed approach in classifying diverse types of network traffic, including control messages, sensor data, and video streams.[42] However, while the study showcases the architecture's performance, it would benefit from a more extensive discussion on the limitations of deep neural networks in real-world scenarios. Additionally, a deeper analysis of computational requirements and training considerations would enhance the understanding of the proposed approach's potential and limitations in robot communication.

Proposed Approach

The genesis of neural networks traces back to the inception of the inaugural network, characterized by a solitary scalar output. This output was pivotal in aggregating local pool activities, enabling the network to discern recurring patterns. Evolution in network architecture, exemplified by Convolutional Neural Networks (CNNs), brought about transformative alterations, resulting in the CNN partitioning of the original image into distinct halves. Conversely, deploying neural networks attains heightened efficacy by incorporating multidimensional features, denoted as" capsules." These capsules undertake intricate internal calculations on input data, encapsulating their findings within vectors that yield refined outputs. Within this framework, each capsule undergoes training to discern entities within visual inputs, a task implicit in a predefined set of constraints, such as locality or deformation range. Subsequently, the capsule generates outputs encompassing both a probability estimate and a suite of entity parameters, all of which remain bound by these imposed constraints.

Illustratively, a local condition serves as an exemplar of such a constraint. This compilation of entity attributes extends beyond mere precise positioning and deformational attributes to encompass illumination conditions pertinent to the visual entity under scrutiny. A well-functioning capsule demonstrates local invariance in the probability of a visual entity's presence, signifying its constancy as it traverses the apparent manifold nested within the capsule's coverage area. A notable proposition lies in the form of an" equivariant" entity parameter. This instance parameter signifies the internal coordinate of the entity across the appearance manifold. Consequently, as the entity traverses the appearance manifold in response to alterations in the observation condition, the instance parameter's value undergoes shifts, as exemplified in Fig. 1.

In Fig. 1, as per Eq.1, L_i, where i varies from 1 to n are low-level image features. Si represents the relationship between the high-level feature H_i and the low-level feature L_i. c_i is a constant, and final output V_{C1} is output in the form of a vector of the capsule c1.

CapsNet

The CapsNet, denoting "capsule network," represents a novel neural network architecture designed to alleviate the limitations inherent in conventional Convolutional Neural Networks (CNNs) concerning the delineation of hierarchical and spatial relationships within images.[43] Central to the CapsNet paradigm is" capsules," coherent assemblies of neurons that functionally supersede individual visual components or attributes. Notably divergent from their CNN counterparts, capsules can encapsulate the presence and instantiation parameters of entities depicted within an image.[43] The CapsNet architecture introduces novel terminological constructs of notable significance, meriting elucidation:

- **Capsules are groupings of neurons representing certain** things or characteristics in a picture. Capsules may be found in the visual cortex. Each capsule will provide a vector as its output. This vector will contain the instantiating parameters (such as posture and size) and the likelihood that the entity will be present.
- **Routing by Agreement:** Capsules interact with one another using a process known as dynamic routing, in which lower-level capsules transfer their outputs to higher-level capsules depending on agreement ratings. Through this routing procedure, higher-level capsules can collect information from lower-level capsules that agree on the existence of certain entities.
- **Length of the Capsule:** The dimensionality of the

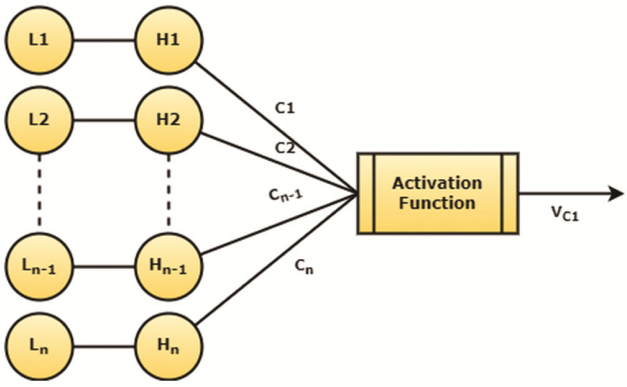

Fig. 1— Hierarchical Capsule Propagation

output vector produced by a capsule indicates the entity's probable presence. In this context, the length of the vector assumes the role of a quantifiable measure denoting the level of confidence associated with the entity's existence. Notably, longer vectors correspondingly signify heightened overall likelihoods of the entity presence within the given visual input.

The architecture of CapsNet is thoughtfully organized into a hierarchical arrangement of layered capsules, thereby cultivating a structured framework that encapsulates and preserves spatial relationships among diverse elements present within the visual composition. To illustrate, envision a capsule adept at discerning traffic signals within a photograph and subsequently furnishing a tri-dimensional vector of a specific extent. In a subsequent phase, this capsule manipulates the image of a stop sign through spatial rotations. Intriguingly, even as the vector undergoes spatial reorientation, signifying an alteration in the observed traffic sign's configuration, the capsule persists in its conviction of having identified the traffic sign. This tenacious adherence to its initial classification prevails despite the capsule's cognisance of the dynamic alteration in the visual object's status. The concept of invariance pursued by CapsNets deviates from the notion of invariance as realised by conventional CNNs, which predominantly rely on mechanisms such as maximum pooling for attaining in-variance. The invariance that CapsNets seeks is distinct—it arises from neural activity undergoing variation while an object traverses the image. Notably, despite these fluctuations, the capsule maintains a constant probability of detecting the object, thereby achieving a form of invariance characterised by preserving detection likelihood despite spatial transformations.

High-level features can be extracted from the underlying low-level features Prakash *et al*. (2016) using the spatial relationship between the low and high-level features as per equation 1. Such an extraction is also shown in Fig. 1.

$$H_i = S_h|_l . L_i \quad \ldots (1)$$

where, H_i is high-level features, $S_h|_l$. is the spatial relationship between high and low-level features and low-level features are represented as L_i.

Activation Function

CapsNet employs a "squashing" or "non-linear squashing" activation function. The output vectors (capsule activations) are guaranteed lengths between zero and one. The existence probability of any object or feature can be represented via normalised activations according to the squashing function. The squashing function is mathematically defined as Eq. 2.

$$v_j = \frac{\left||s_j|\right|^2}{1+\left||s_j|\right|^2} * \left(\frac{s_j}{\left||s_j|\right|}\right) \quad \ldots (2)$$

where, s_j represents the input vector to a capsule, and v_j represents the output vector after applying the squashing function. The double vertical bars denote the Euclidean norm or length of the vector.

There are two main characteristics of the squashing function. The first one squashes short vectors in which the squashing function magnifies the length of short input vectors, bringing them closer to the value 1. When the capsules discover objects, this serves to highlight their existence. The second one saturates long vectors. When applied to input vectors of excessive length, the squashing function reduces their dimensions, bringing them closer to 1. This helps keep the length from being too big, which keeps the activations within the acceptable range of [0, 1]. Squashing helps the CapsNet accurately record instantiation probability and existence probabilities for entities. After squashing, the length of the output vectors is used to calculate agreement ratings between capsules, which aids the dynamic routing process. Notably, capsules' input vectors undergo element-wise squashing, with each vector dimension being processed separately by the squashing function.

Dynamic Routing

CapsNet relies heavily on a technology known as dynamic routing to make it possible for capsules to talk to one another and share data. It helps the capsules agree on whether or not certain things or qualities exist and their characteristics. When using dynamic routing, the routing weights between capsules are adjusted repeatedly to favour connections between capsules that agree and punish those that disagree. The network's routing and representations are improved via this iterative approach. The algorithmic steps show how CapsNet's dynamic routing algorithm works.

- **Initialisation:** Set the relatively small positive numbers b_{ij} as the initial values for the routing coefficients. The correlation between capsules i and j indicates how much they agree using these

coefficients. In most cases, all capsule pairings will have the same beginning values.

- **Prediction**: Based on its calculations, each capsule at a lower level i produces its output vector u_i (called the "prediction"). An entity's or feature's instantiation parameters, such as orientation and probability, are represented by this vector in the output.
- **Weighted Sum:** To do this, we multiply the resultant vectors u_i from capsules at a lower level by their respective routing coefficients b_{ij}. Here, we consider the current routing weights when combining the predictions from lower-level capsules.
- **Summation:** For each higher-level capsule j, acquire the overall input s_j by summing the weighted predictions across all lower-level capsules.
- **Non-linearity:** For each capsule at the upper level, apply a non-linear squashing function $squash()$ on the whole input, s_j, element by element. This squashing function keeps the vector's direction intact while reducing its length (magnitude) to 0 and 1.
- **Routing Update:** If there is no agreement between the outputs of capsules i and j, then the routing coefficients b_{ij} should be updated accordingly. Dot-producing the output vectors of lower-level capsules with the sum of the inputs of higher-level capsules yields the amount of agreement. To guarantee a reasonable probability distribution, the softmax function is often used to calculate the agreement.
- **Iteration:** To incrementally improve the routing method, steps 3–6 should be repeated many times. With each iteration, the routing weights are fine-tuned in light of the agreement ratings, making links between capsules that agree stronger and those between disagreeing capsules weaker.
- **Output:** Each iteration results in a higher-level capsule's output representing the probability of occurrence and instantiation parameters for a given entity or feature in the input data.

By repeatedly modifying the routing weights based on agreement scores, CapsNet's dynamic routing method enables capsules to jointly determine things' existence and attributes. This iterative procedure aids in using Grammarly for punctuation, proving routing, creating more accurate representations, and identifying network hierarchies. Information from many lower-level capsules is integrated into higher-level capsules through dynamic routing, allowing CapsNet to accommodate fluctuations in input data, capture complicated spatial connections, and develop invariant representations. It encourages the development of more precise and reliable representations, which improves the network's object detection and classification capabilities.

CapsNet Architecture

Original Architecture

The original proposed CapsNet architecture, as shown in Fig. 2, has a very simple architecture consisting of just 3 layers: 1 fully linked and 2 convolutional. When it comes to extracting simple characteristics, CNN excels. CapsNets, on the other hand, represent an "instance" of an object, making it more suited for defining complex instances. Thus, the convolutional layer is added at the bottom of CapsNets to retrieve the underlying feature.[43]

CapsNet assumes that the input it receives is either an image or some data structure with a higher dimension. Assume a 32 × 32 pixel input size. The primary capsule layer is taught to recognize very simple visual components or patterns in the entering image. Let's suppose that there are 32 primary

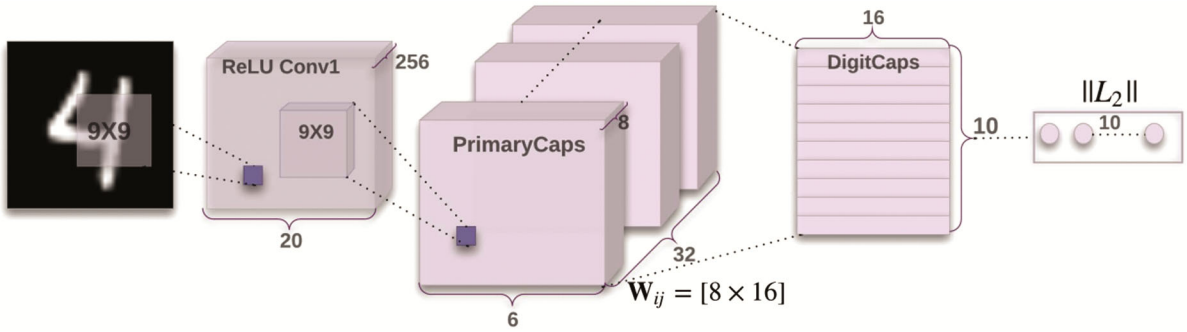

Fig. 2 — A simple CapsNet with 3 layers

capsules in this stratum. Each primary capsule creates a vector with a size of eight that contains the instantiation parameters and the existence probability for a particular feature. The output form of the main capsules layer is (32, 8, 6, 6), which makes sense given that the vector dimension is 8 and the spatial size is 6 × 6 after the convolutional procedures have been applied. The capsule layer comprises higher-level capsules, each responsible for expressing a unique entity or attribute of the input. Let's say this layer holds 10 capsules. Every secondary capsule gets its input from the parent capsules. Then, it constructs a vector with 16 dimensions containing the instantiation parameters and existence probability for the entity it represents. Therefore, the output form of the capsule layer is a square with coordinates of 10 and 16, respectively. The dynamic routing process is carried out in the space between the major capsules and capsule layers. The routing approach continuously updates the routing weights ($w_{i,j}$) by making use of the agreement ratings that are shared between the primary capsules (u_i) and the capsules that make up the capsule layer (v_j). A conclusive classification is determined after considering the findings of the capsule layer. The output of the capsule contains a vector that represents the presence of a certain item or feature, if one exists.

Modified Architecture

Since the framework of the Standard CapsNet is so simple, and just one conversion layer is utilised to extract low-level data, this is not sufficient. Our suggested multi-layer version of CapsNet employs several convolution layers instead of the original model's single convolution layer. Integrating a convolutional neural network (CNN) layer, or many layers before the CapsNet can potentially enhance the network's overall performance. This is achieved by capitalising on the feature extraction capabilities inherent in CNNs and the feature combining and preservation capabilities offered by CapsNets. The integration of this architectural approach proves advantageous in several computer vision applications, such as object identification, segmentation, and other tasks that consider spatial connections among features.

Algorithm 1: CapsNet with 3 Convolutional Layers

Input: Input Image
Output: Capsule Outputs

Initialize CNN weights W_{cnn} and biases ca_n;
Initialize CapsNet weights W_{caps} and biases b_{caps};
CNN 1: Apply N_1 convolutional filters to the input image:
$Z_{cnn1} = \text{Convolve}(I, W_{cnn1}) + b_{cnn1}$;
AF 1: Apply a non-linear activation function to the outputs of Convolutional Layer 1:
$A_{af1} = \text{ActivationFunction}(Z_{cnn1})$;
PL 1: Perform max pooling on the outputs of Activation Function 1:
$P_{pl1} = \text{MaxPooling}(A_{af1})$;
CNN 2: Apply N_2 convolutional filters to the pooled outputs:
$Z_{cnn2} = \text{Convolve}(P_{pl1}, W_{cnn2}) + b_{cnn2}$;
AF 2: Apply a non-linear activation function to the outputs of Convolutional Layer 2:
$A_{af2} = \text{ActivationFunction}(Z_{cnn2})$;
PL 2: Perform max pooling on the outputs of Activation Function 2:
$P_{pl2} = \text{MaxPooling}(A_{af2})$;
CNN 3: Apply N_3 convolutional filters to the pooled outputs:
$Z_{cnn3} = \text{Convolve}(P_{pl2}, W_{cnn3}) + b_{cnn3}$;
AF 3: Apply a non-linear activation function to the outputs of Convolutional Layer 3:
$A_{af3} = \text{ActivationFunction}(Z_{cnn3})$;
CapsNet Layer: Feed the outputs of Activation Function 3 to the CapsNet:
$C_{caps} = \text{CapsuleLayer}(A_{af3}, W_{caps}, b_{caps})$;
Dynamic Routing: Perform dynamic routing in the CapsNet:
$C_{routed} = \text{DynamicRouting}(C_{caps})$;
Capsule Outputs: Get the final output capsules from the CapsNet:
$\mathbf{O}_{caps} = \text{FinalCapsuleOutputs}(\mathbf{C}_{routed})$;

Analyzing the effect of varying the number of convolutional layers in a model is crucial for understanding its performance and scalability. Compare the performance of the model with varying numbers of convolutional layers against each other and against a baseline. In the proposed multi-layer version of CapsNet, each channel's convolution is built from a collection of convolution kernels with sizes ranging from 3 × 3 to 5 × 5 to 7 × 7 to 9 × 9, with the combined outputs from all three channels serving as input to the subsequent layer. There are three convolution channels before the Primary Capsule layer in Fig. 3. Three-by-three and seven-by-seven convolution kernels are used in the first channel. Although bigger convolution kernels may enhance the receptive field and provide more data, smaller convolution channels have the potential to reduce network parameters while preserving accuracy.

The original 9×9 convolution kernel is used in the second channel to keep the same size receptive field. The first channel's 7×7 convolution kernel is replaced with a mixture of 3×3 and 5×5 convolution kernels in the final channel. The key objective in this context is to increase the depth of the neural network, expanding its effect while ensuring the preservation of the same perceptual area. Each of the three channels produces 128 unique feature maps that are 20×20 in size. Finally, the output of the three channels is concatenated in the final dimension using the concatenating function to produce a feature map that is more informative about the image's essential characteristics. Iterative routes from basic to complex features are implemented using capsules after the Primary Capsule layer.

The algorithm for the proposed approach is shown in Algorithm 1, where CNN 1, CNN 2, and CNN 3 represent the three convolutional layers, and N1, N2, and N3 represent the number of filters in each layer. Similarly, AF 1, AF 2, and AF 3 denote the activation functions applied after each convolutional layer. PL 1, PL 2, and PL 3 are the Pooling Layers.

Experiment and Results

Experiments have been performed on a public database of traffic images collected as the German Traffic Sign Recognition Benchmark (GTSRB). The dataset consists of exactly 12,570 images with the extension of PPM (ppm) and a training dataset with exactly 39,252 images. All of the images in the dataset have been shrunk to a more manageable 32 × 32 pixels in preparation for pickling. Noise in traffic scenarios can manifest as random variations in pixel intensity, artifacts caused by low-light conditions or sensor limitations, and interference from surrounding vehicles or infrastructure. This noise can arise from various sources, including sensor imperfections, lighting conditions, motion blur, compression artifacts, and interference from other electronic devices.

We have compiled three sets: a training set, a test set, and a validation set. A data dictionary consisting of the four key/value pairs features, labels, sizes, and codes is the output of our data, which is loaded in Python pickle. The raw pixel data of the traffic sign images is stored in the features 4D array. The pictures of traffic signs are labelled in a one-dimensional array called labels. The image's original width and height are stored in the tuple's width and height in the list sizes. A list of tuples coords, with values of (x1, y1, x2, y2), stands for the coordinates of the image's bounding box as shown in Fig. 4. The images from each subgroup are displayed using Matplotlib, and a histogram is generated using numpy. We employed randomisation, grey-scaling, local histogram equalisation, and normalisation throughout the data

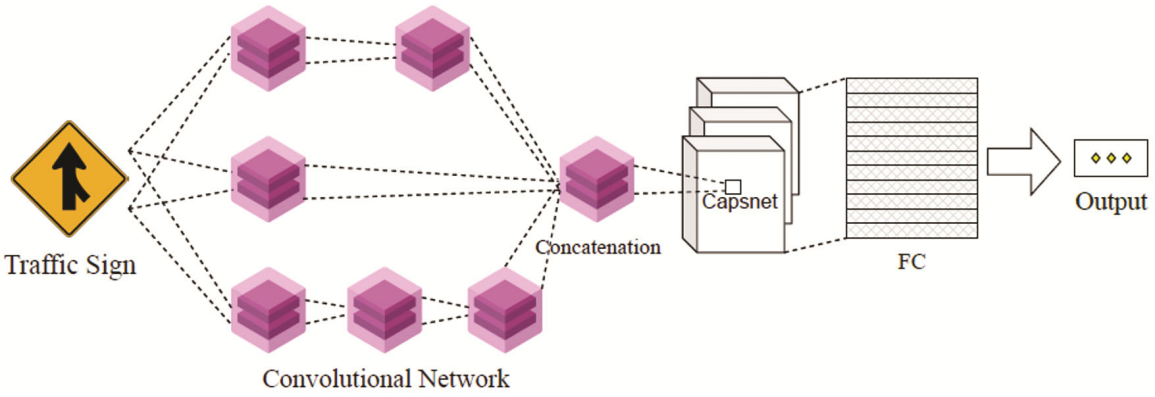

Fig. 3 — Modified CapsNet architecture having multiple convolutional layers

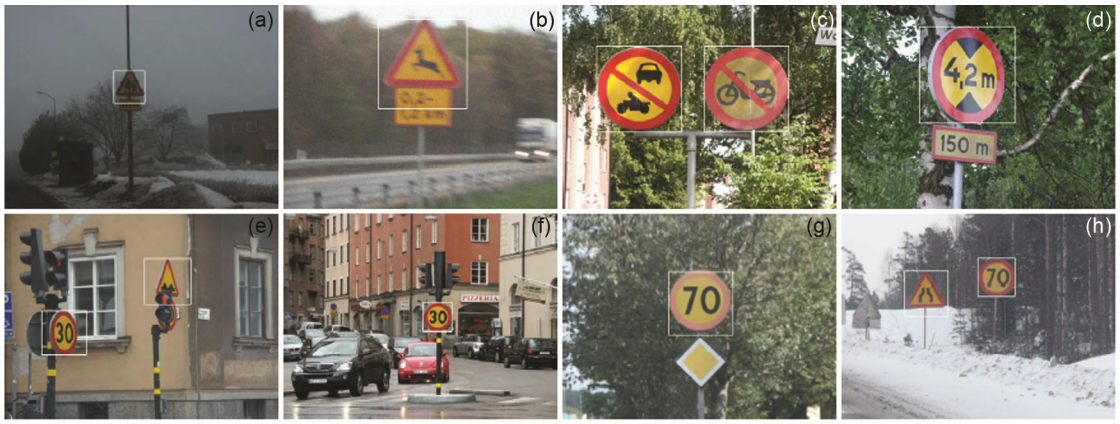

Fig. 4 — Result of Image bounding-box generation

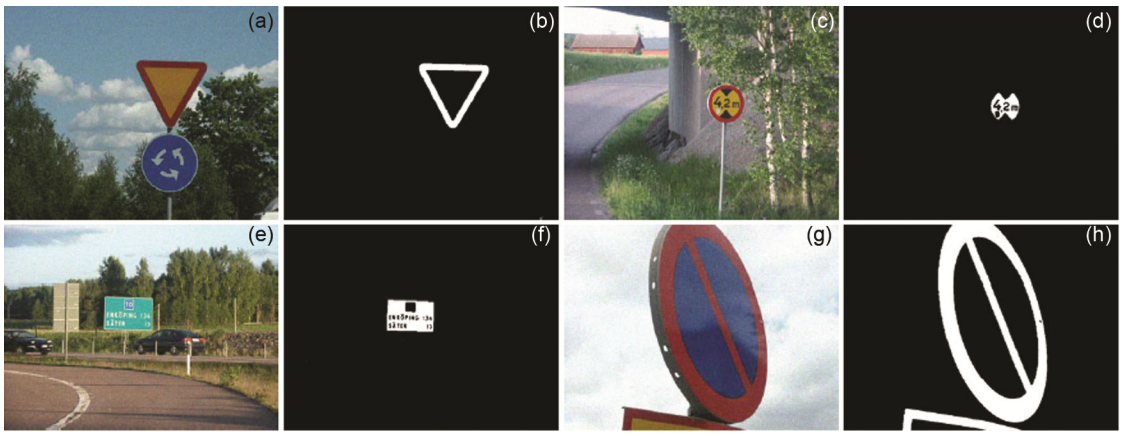

Fig. 5 — Pre-Processing results (a) A traffic sign in daylight, (b) Result of segmentation, (c) Occluded traffic sign, (d) Result of Segmentation, (e) A distant traffic sign, (f) Result of Segmentation, (g) An angular view of traffic sign, (h) Result of Segmentation

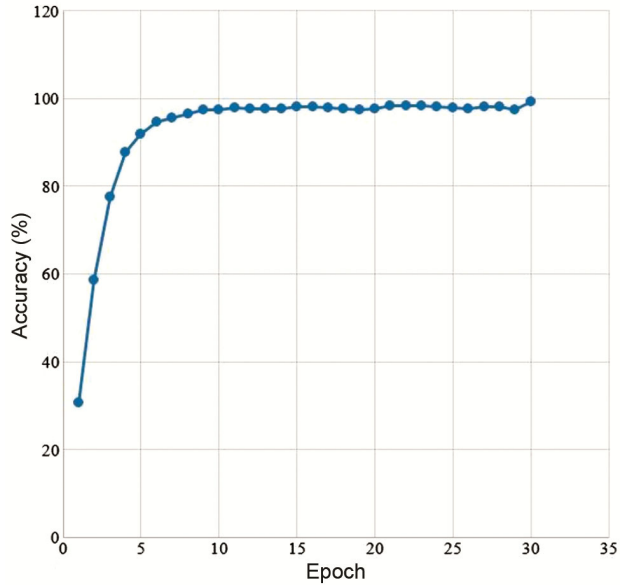

Fig. 6 — Validation Accuracy obtained after using CapsNet

preparation step. Software Dependencies for the experiment are shown in Table 1.

Table 1 — Software Dependencies for the conducted experiments

Software Dependencies	
Ubuntu	16.04
Anaconda	23.1.0
Python	3.10.9
Tensor Flow 0.12.1 (GPU support)	

The visual representation of the pre-processing outcomes for the images is shown in Fig. 5, which includes traffic images captured under various conditions, such as distant views, daylight scenes, foggy environments, and noisy and obstructed scenarios. Experimental results in Fig. 5 demonstrate the impact of noise reduction techniques on the performance of the proposed method, particularly under challenging conditions characterized by occlusions, atmospheric haze, and noise.

The efficacy of the pre-processing phase applied to the images is evident across all cases. The maximum degree of validation accuracy attained is 99.16% after 30 epochs (Fig. 6).

Table 2 — Performance comparison with existing state-of-arts

Author	Detection Method	Accuracy
Satti *et al*.[44]	CNN	98.2%
Liu *et al*.[45]	CNN	98.97%
He *et al*.[46]	CNN, SVM, ResNet 101	95.77%
Mishra & Goyal[47]	CNN	98.37%
Proposed Model	Multi-Layer CapsNet	99.16%

Accuracy is a commonly used performance metric because it provides a simple and intuitive measure of overall correctness. In this paper we used the accuracy for traffic sign detection and compared with the existing state-of-art (Table 2).

Conclusions

An innovative methodology for automating the recognition of traffic signs was introduced in this research paper, employing a modified Capsule Network (CapsNet) architecture. The proposed methodology includes a distinctive pre-processing phase, wherein input images undergo a sequence of transformations utilizing three distinct Convolutional Neural Networks (CNNs) before integration into the CapsNet model. Experimental findings demonstrate the efficacy of this approach, with a validation accuracy of 99.12%. This achievement underscores the system's position as a leading solution for traffic sign identification. The adaptability and resilience of the CapsNet architecture, particularly when combined with complementary neural network components, are emphasized. Future research can focus on further refining the methodology, exploring a broader range of dataset modifications, and assessing real-time deployment feasibility. Automated systems have the potential to enhance road safety and advance intelligent transportation systems. Continued advancements in deep learning models and strategies will facilitate ongoing growth and development in this domain.

References

1 Yurtsever E, Lambert J, Carballo A & Takeda K, A survey of autonomous driving: Common practices and emerging technologies, *IEEE access*, 8 (2020) 58443–58469.

2 Liu K, Ding R, Zou Z, Wang L & Tang W, A comprehensive study of weight sharing in graph networks for 3D human pose estimation, in *Computer Vision–ECCV 2020: 16th Europ Conf Glasgow, UK,* August 23–28, 2020, Proceedings, Part X 16 (Springer International Publishing) 2020, 318–334,.

3 Wang J, Liu J & Kato N, Networking and communications in autonomous driving: A survey, *IEEE Commun Surv Tutor*, **21(2)** (2018) 1243–1274.

4 Liu S, Liu L, Tang J, Yu B, Wang Y & Shi W, Edge computing for autonomous driving: Opportunities and challenges, Proceedings of the *IEEE*, **107(8)** (2019) 1697–1716.

5 Singh A, Rani P, Ramesh J V N, Athawale S V, Alkhayyat A H, Aledaily AN, ... & Sharma R, Blockchain-Based Lightweight Authentication Protocol for Next-Generation Trustworthy Internet of Vehicles Communication, *IEEE Trans Consum Electron* (2024).

6 Claussmann L, Revilloud M, Gruyer D & Glaser S, A review of motion planning for highway autonomous driving, *IEEE Trans Intell Transp Syst*, **21(5)** (2019) 1826–1848.

7 Verma S, Rani P, Gupta S, Sharma R, Yadav K, Aledaily A N & Alharbi M, An automated face mask detection system using transfer learning based neural network to preventing viral infection, *"Int J Expert Syst"*, **41(3)** (2024) e13507.

8 Fujiyoshi H, Hirakawa T & Yamashita T, Deep learning-based image recognition for autonomous driving, *IATSS Res*, **43(4)** (2019) 244–252.

9 Feng D, Haase-Schütz C, Rosenbaum L, Hertlein H, Glaeser C, Timm F, & Dietmayer K, Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges, *IEEE Trans Intell Transp Syst*, **22(3)** (2020) 1341–1360.

10 Yaqoob I, Khan L U, Kazmi S A, Imran M, Guizani N & Hong C S, Autonomous driving cars in smart cities: Recent advances, requirements, and challenges, *IEEE Network*, **34(1)** (2019) 174–181.

11 Sinha S & Umrao B K, License plate recognition: An insight to the proposed approach for plate localization and binarization technique.

12 Umrao B K & Yadav D K, Placement of virtual network functions for network services, *Int J Enterp Netw Manag*, **33(6)** (2023) e2232.

13 Umrao B K & Yadav D K, Algorithms for functionalities of virtual network: a survey, *J Supercomput*, **77(7)** (2021) 7368–7439.

14 Li Y, Ma L, Zhong Z, Liu F, Chapman M A, Cao D & Li J, Deep learning for lidar point clouds in autonomous driving: A review, *IEEE Trans Neural Netw Learn Syst*, **32(8)** (2020) 3412–3432.

15 Mozaffari S, Al-Jarrah O Y, Dianati M, Jennings P & Mouzakitis A, Deep learning-based vehicle behavior prediction for autonomous driving applications: A review, *IEEE Trans Intell Transp Syst*, **23(1)** (2020) 33–47.

16 Rani P & Sharma R, Intelligent transportation system performance analysis of indoor and outdoor internet of vehicle (iov) applications towards 5g, *Tsinghua Sci Technol*, **29(6)** (2024) 1785–1795.

17 Chao Q, Bi H, Li W, Mao T, Wang Z, Lin M C & Deng Z, A survey on visual traffic simulation: Models, evaluations, and applications in autonomous driving, *Comput Graph Forum*, **39(1)** (2020) 287–308.

18 Camara F, Bellotto N, Cosar S, Weber F, Nathanael D, Althoff M & Fox C, Pedestrian models for autonomous driving part ii: high-level models of human behavior, *IEEE Trans Intell Transp Syst*, **22(9)** (2020) 5453–5472.

19 Rani P & Sharma R, An experimental study of *IEEE 802.11* n devices for vehicular networks with various propagation loss models*, In International Conference on Signal Processing and Integrated Networks*, *Singapore: Springer Nature Singapore*, (2022) 125–135.

20 Lavanya R, Hiremath S K, Cheruku S, Bhargava S & Bhandari S, CNN in periodontology: A review, *J Orofac Health Sci*, **16(1)** (2020) 64–70.

21 Yamashita R, Nishio M, Do R K G & Togashi K, Convolutional neural networks: an overview and application in radiology, *Insights Imaging*, **9** (2018) 611–629.

22 Dewi C, Chen R C, Yu H & Jiang X, Robust detection method for improving small traffic sign recognition based on spatial pyramid pooling, *J Ambient Intell Humaniz Comput*, **14(7)** (2023) 8135–8152.

23 Rani P & Sharma R, Intelligent transportation system for internet of vehicles based vehicular networks for smart cities, *Comput Electr Eng*, **105** (2023) 108543.

24 Singh A, Rahma M Z U, Rani P, Sharma R & Kariri E, Smart traffic monitoring through real-time moving vehicle detection using deep learning via aerial images for consumer application, *IEEE Trans Consum Electron* (2024).

25 Boda P L & Ramadevi Y, A systematic review on autonomous vehicle: Traffic sign detection and drowsiness detection, *In International Conference on Artificial Intelligence and Data Science*, Cham: Springer Nature Switzerland, (2021) 41–51.

26 Pawan S J & Rajan J, Capsule networks for image classification: *A review, Neurocomputing*, **509** (2022) 102–120.

27 Dombetzki L A, An overview over capsule networks, *Network Architectures and Services*, **10** (2018).

28 Goceri E, CapsNet topology to classify tumours from brain images and comparative evaluation, *IET Image Process*, **14(5)** (2020) 882–889.

29 Nazir M, Shakil S & Khurshid K, Role of deep learning in brain tumor detection and classification (2015 to 2020), *Comput Med Imaging Graph*, **91** (2021) 101940.

30 Kapadnis S, Tiwari N & Chawla M, Developments in capsule network architecture: a review, *In Intelligent Data Engineering and Analytics: Proceedings of the 9th International Conference on Frontiers in Intelligent Computing: Theory and Applications (FICTA 2021)*, Singapore: Springer Nature Singapore, 81–90.

31 Tiwari K, Umrao B K & Scholar M T, Leader election approach: A comparison and survey, *J Adv Comput Commun Technol*, **3** (2015) 2347–2804,

32 Patrick M K, Adekoya A F, Mighty A A & Edward B Y, Capsule networks–a survey, *J King Saud Univ - Comput Inf Sci*, **34(1)** (2022) 1295–1310.

33 Guarda L, Tapia J E, Droguett E L & Ramos M, A novel capsule neural network based model for drowsiness detection using electroencephalography signals, *Expert Syst Appl*, **201** (2022) 116977.

34 Shi R & Niu L, A brief survey on capsule network. *In 2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)* IEEE, 682–686.

35 Sharma B, Prakash R, Tiwari S & Mishra K K, A variant of environmental adaptation method with real parameter encoding and its application in economic load dispatch problem, *Appl Intell*, **47** (2017) 409–429.

36 Guo Q, Li L & Ban X J, Urban traffic signal control with connected and automated vehicles, *Transp Res Part C Emerg Technol*, **101** (2019) 313–334.

37 Mohana T & Akshaya V, Real-time traffic sign detection using capsule network, *In 2019 11th International Conference on Advanced Computing (ICoAC)* IEEE, 193–196.

38 Arun P V, Buddhiraju K M & Porwal A, Capsulenet-based spatial–spectral classifier for hyperspectral images, *IEEE J Sel Top Appl Earth Obs*, **12(6)** (2019) 1849–1865.

39 Fang C, Shang Y & Xu D, Improving protein gamma-turn prediction using inception capsule networks, *Sci Rep*, **8(1)** (2018) 15741.

40 Kumar A D, Novel deep learning model for traffic sign detection using capsule networks, *arXiv preprint arXiv*: (2018), 1805.04424.

41 Ge M, Yu X & Liu L, Robot communication: Network traffic classification based on deep neural network, *Front Neurorobot*, **15** (2021) 648374.

42 Chaudhary S, Indu S & Chaudhury S, Video-based road traffic monitoring and prediction using dynamic Bayesian networks, *IET Intell Transp Syst*, **12(3)** (2018) 169–176.

43 Sabour S, Frosst N & Hinton G E, Dynamic routing between capsules, *Adv Neural Inf Process Syst*, **30** (2017).

44 Satti S K, Devi K S, Dhar P & Srinivasan P, Enhancing and classifying traffic signs using computer vision and deep convolutional neural network, *In Machine Learning, Image Processing, Network Security and Data Sciences: Second International Conference, MIND* 2020, Silchar, India, July 30–31 2020, Proceedings, Part I 2, Springer Singapore, 243–253.

45 Liu Z, Li D, Ge S S & Tian F, Small traffic sign detection from large image, *Appl Intell*, **50(1)** (2020) 1–13.

46 He S, Chen L, Zhang S, Guo Z, Sun P, Liu H & Liu H, Automatic recognition of traffic signs based on visual inspection, *IEEE Access*, **9** (2021) 43253–43261.

47 Yadav S P, Jindal M, Rani P, de Albuquerque V H C, dos Santos Nascimento C & Kumar M, An improved deep learning-based optimal object detection system from images, *Multim Tools Appl*, **83(10)** (2024) 30045–30072.