AI based Geospatial Vehicle Prediction

Mr. Sumit Kumar Banerjee
Computer Science & Engineering
Guru Nanak Institute of Technology
Kolkata, India

Mrs. Priyanka Chakraborty
Computer Science & Engineering
Guru Nanak Institute of Technology
Kolkata, India

Mr. Rafiqul Islam
Computer Science & Engineering
Guru Nanak Institute of Technology
Kolkata, India

***Abstract*— Today pictures are exceptionally imperative, totally different applications counting with computerized highlights like self-driving cars, partisan imaging and numerous more. For all these applications, we require importance from people in watching changes within the environment. We know that the surface of soil is exceptionally huge to cover physically but it is exceptionally troublesome to screen it. Thus, we must move into computerization to utilize the restricted proficiently accessible assets. Now-a-days, the existing ancient strategies and calculations are not valuable enough to illuminate the issue. To solve, we utilize profound learning techniques (a portion of machine learning). Within the strategy, we make and apply the vehicle question discovery to draw a rectangular boundary-box over the anticipated vehicle pictures. The training precision of our showing model is 96.7%.**

Keywords—Deep learning, Image classification, Satellite images

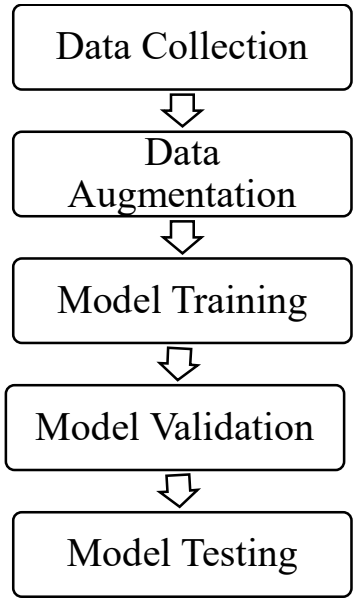
I. INTRODUCTION

Vehicle detection using artificial intelligence is using the CNN algorithm to detect vehicles either in images or video formats. Using labeled dataset training, these models train various vehicles like cars, trucks, buses, and motorcycles. This method is crucial in traffic management, smart transport systems, etc. Its application starts from smart collision avoid systems like parking systems etc. By correctly predicting vehicles, it saves traffic accidents with efficient traffic flows, and improves the capability of surveillance. In artificial intelligence methods, the accuracy of vehicle detection systems expects to increase smarter transport solutions. The paper development of these systems may result in a safe, efficient and better society.

II. PROPOSED METHODOLOGY

Geospatial satellite vehicle image detection involves using machine learning techniques to classify and detect vehicles within these images. We divided our proposed work into the following phases. They are Data Collection, Data Augmentation, Model Training, Model Validation, and Model Testing. Figure – 1 shows the possible proposed phases.

FIGURE 1: PROPOSED PHASES



A. Data Collection

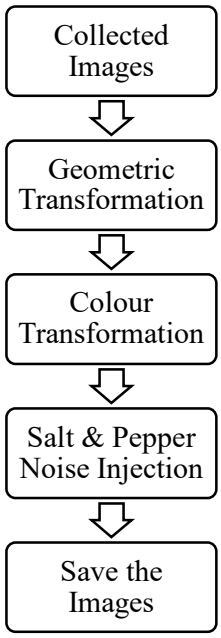
We collect our dataset from a verified online resource using web scraping. Our dataset is basically an image based dataset. Then, we convert them in to 200 x 200 pixels images.

B. Data Augmentation

Data augmentation is a technique used to improve the accuracy and amount of data by applying various transformations to the existing data. We use the following augmentation to improve the accuracy of the model:

- **Geometric Transformations**: After collecting the dataset, we implement geometric transformations such as rotation, flipping, etc.
- **Colour Transformations:** After geometric transformations of the dataset, we use colour transformations such as gamma scaling, hue, brightness and contrast adjustment respectively.
- **Salt & Pepper Noise Injection:** After colour transformations, we use salt and pepper noise as noise injection in to our proposed dataset.

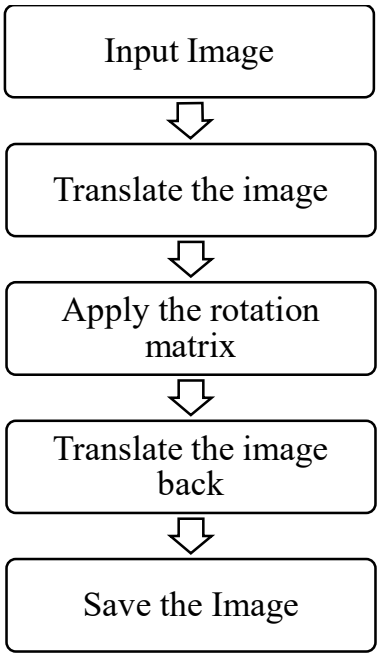
FIGURE 2: IMAGE AUGMENTATION



1) Rotation

Rotation contains the following steps that helps us to rotate an image with respect to center.

FIGURE 3: IMAGE ROTATION PROCESS



a) Input Image

The initial step of rotating an image is getting input from collected image dataset.

b) Translate Image

Then, we translate the input image with respect to the center of the image. Let, consider the coordinate of the central image pixel is (x_c, y_c). Now, we translate each pixel of the image from (x_c, y_c) to $(x - x_c, y - y_c)$.

c) Apply the rotation matrix

Then, we apply the translation over the rotation matrix. We apply the following formula to compute the rotation matrix R is as follows:

$$R = \begin{pmatrix} \cos t & -sint \\ \sin t & cost \end{pmatrix}$$

Where, t = rotation angle (from 0 degree to 359 degree)

Now, we compute new coordinates (x', y') after rotation:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = R * \begin{bmatrix} x \\ y \end{bmatrix}$$

d) Translate the image back

Now, we translate the rotated image coordinates back in to the original coordinates by adding (x_c, y_c).

e) Save the image

Lastly, we save the image into our local image directory.

2) Horizontal Flipping

After rotation, we implement the horizontal flipping technique. Basically, it used to swap each pixel values (x, y) with the pixel (w – 1 – x, y).

Hence, Q (x, y) = O (w – 1 – x, y)

Where Q (x, y) is the new pixel and O (w – 1 – x, y) is the original pixel value.

3) Vertical Flipping

After horizontal flipping, we implement the vertical flipping technique. Basically, it used to swap each pixel values (x, y) with the pixel (x, h – 1 – y)

Hence, Q (x, y) = O (x, h – 1 – y)

Where Q (x, y) is the new pixel and O (x, h – 1 – y) is the original pixel value.

4) Scaling

After flipping, we apply scaling by changing height and width. To compute scaling, we apply the following formula for each pixel:

$$x' = x * S_x, y' = y * S_y$$
$$\text{And, } Q(x', y') = O(x, y)$$

Where, Q (x', y') is the scaled image and O (x, y) is the original image, S_x is the scaling factor of x coordinate and S_y is the scaling factor of y coordinate.

5) Translation

After scaling, we do translation for each pixel of image dataset. The new coordinate (x', y') of any pixel at the original coordinate (x, y) is computed as follows:

$$x' = x + \Delta x \text{ and } y' = y + \Delta y$$

Where, $(\Delta x, \Delta y)$ is a certain offset coordinate.

For homogenous coordinates, we apply the matrix T as follows:

$$T = \begin{bmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{bmatrix}$$

For any sample point (x, y, 1), translated coordinates likelihood (x', y', 1) is calculated as:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = T * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

6) Gamma Scaling

Gamma scaling is a non-linear operational method, used to encode and decode the luminance values in image processing. To compute gamma scaling, we use the following formula:

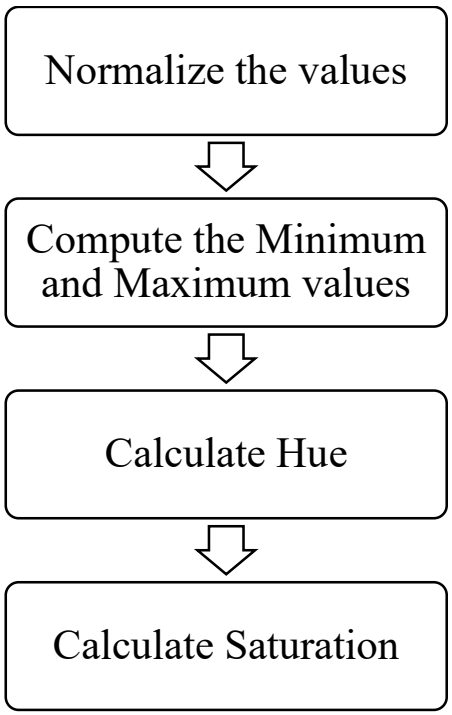
$$Pixel\ correction = \left(\frac{Original\ pixel\ value}{255}\right)^{\gamma} * 255$$

Here, γ is the gamma value. If $\gamma < 1$, the image will be brighter; if $\gamma > 1$, the image will be darker. We use 0.75 and 1.5 as gamma value respectively.

7) Adjust Hue and Saturation value

We apply the following calculation to adjust the hue and saturation respectively as shown in Figure 4.

FIGURE 4: PROCESS FOR ADJUSTMENT OF HUE AND SATURATION VALUE



a) Normalize the values

In this step, we normalize the values by dividing 255 with to respect to each pixel value.

$$R' = \frac{R}{255}, G' = \frac{G}{255}, and\ B' = \frac{B}{255}$$

b) Compute the minimum and maximum value

Now, we compute the minimum and maximum value over R', G' and B', and find out the difference.

$$Color_{maximum} = maximum(R', G', B')$$

$$Color_{minimum} = minimum(R', G', B')$$

$$\Delta = Color_{maximum} - Color_{minimum}$$

c) Calculate Hue

After that, we assume difference is zero if and only if Hue (H) is zero. Then, we calculate the Hue using following formulas:

$$If\ C_{maximum} = R' then, H = 60 * \left(\frac{G' - B'}{\Delta}\ modulo\ 6\right)$$

$$If\ C_{maximum} = G'\ then, H = 60 * \left(\frac{B' - R'}{\Delta} + 2\right)$$

$$If\ C_{maximum} = B'\ then, H = 60 * \left(\frac{R' - G'}{\Delta} + 4\right)$$

d) Calculate Saturation

In this step, we calculate the Saturation using the following formula.

$$S = \begin{cases} 0, & If\ Color_{maximum} = 0 \\ \dfrac{\Delta}{Color_{maximum}}, & elsewhere \end{cases}$$

8) Brightness Adjustment

After adjusting the Hue and Saturation, we adjust the brightness of each image of the dataset using the following formulas:

$$I'(x, y) = minimum(maximum(I(x, y) + B, 0), 255)$$

Where, I' (x, y) is the pixel of the new image, I (x, y) is the brightness and B is the brightness factor.

9) Contrast Adjustment

After adjusting the brightness, we adjust the contrast of each image of the dataset using the following formulas:

$$I'(x, y) = \alpha * (I(x, y) - \mu) + \mu$$

Where, I (x, y) is the original pixel, I' (x, y) is the new pixel value after adjusting contrast, α is the contrast factor, and μ is the midpoint in the range from 0 to 255, and sets to 128. Also, α must satisfy the following condition:

$$\alpha = \begin{cases} greater\ than\ 1\ then, & contrast\ increases \\ 0 < \alpha < 1\ then, & contrast\ decreases \end{cases}$$

10) Salt & Pepper noise injection

We implement the following formula to compute the salt and pepper noises for our color image dataset:

$$num\ of\ salt = ceiling\left(\frac{density * size * 0.5}{3}\right)$$

$$num\ of\ pepper = ceiling\left(\frac{density * size * 0.5}{3}\right)$$

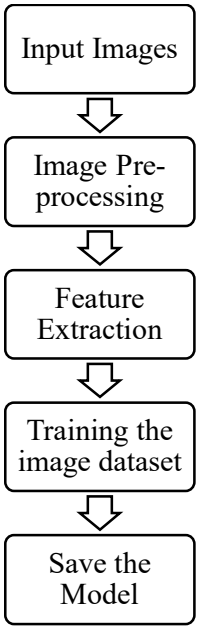
$$I' = \begin{cases} 0\ if \sum_{i=0}^{num\ of\ pepper} random\ (0, \quad i-1) \\ 1\ if \sum_{i=0}^{num\ of\ salt} random(0, \quad i-1) \end{cases}$$

After computing the I' (noisy image), we convert them in to color mode.

C. Model Training

Model training phase contains three sub-phases such as Image Pre-processing, Feature Extraction, and Training phase. We split our dataset in to 80% as training and 20% as validation.

FIGURE 5: MODEL TRAINING



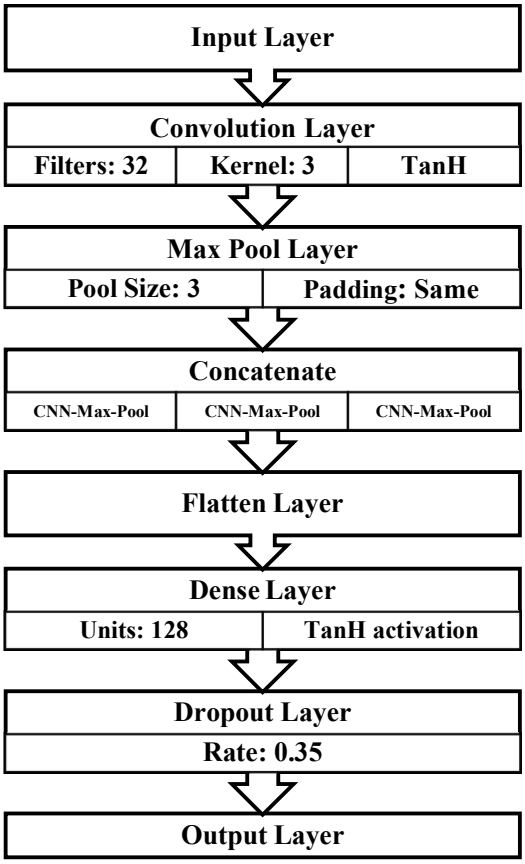
1) Image Pre-processing

We pre-process images using "image_dataset_from_directory" function by setting image size in 200 x 200 pixels. And also, we set the batch size to 16 with random seed is to 16. Also, we use bicubic interpolation to get smooth image features.

2) Feature Extraction

We apply feature extraction after image pre-processing. In this phase, we convert the images into numerical features, and convert labels into categorical features.

FIGURE 6: TRAINING PHASE IN DETAILS



3) Training Phase

In model training phase, we implement the following:

a) Convolution layer:

Let us assume that, an input image I with the H (height) * W (width) dimension and a filter K with the dimension of F_H * F_W, then the convolution is computed as

$$O(i,j) = \sum_{m}^{F_H-1} \sum_{n}^{F_W-1} I(i+m, j+n) * K(m,n)$$

Where, O (I, j) is the position at (i, j), I (i + m, j + n) is the input value at position at I (i + M, j + n), and K (m, n) is the filter value of (m, n) position.

And also, we compute the output dimensions using the following formulas:

$$O_H = \left\lfloor \frac{H + 2 * P - F_H}{S} \right\rfloor + \beta$$

$$O_W = \left\lfloor \frac{W + 2 * P - F_W}{S} \right\rfloor + \beta$$

Where, H = height, W = width, F_H = filter height, F_W = filter width, P = padding, S = Strides, and β = bias and it sets to 1.

b) Tangent Hyperbolic activation function (or TanH)

Tangent Hyperbolic activation function is used to faster convergence on training time and the range between the TanH activation function is from -1 to 1.

$$TanH\ (y) = \frac{e^{y} - e^{-y}}{e^{y} + e^{-y}}$$

c) Max pool layer

To reduce the dimensionality, we use the Max-pool layer which is computed as:

$$O(i,j) = maximum\ \big(I(i+m, j+n)\big);\ \forall m,n\ \in poolsize$$

Where, pool-size = the size of the window.

d) Concatenate leyer

This layer combines the three CNN based Max-pool layer with respect to axis set to 1.

e) Flatten layer

This layer is required to perform from 3-dimension vectors to 1-dimension vectors by preserving the batch size.

f) Dense layer

In this layer we compute output using the following formula

$$Output = TanH\ (Inputs * Weights\ +\ Bias)$$

g) Dropout layer

We use the following formula to compute the dropout layer:

$$Dropout = \frac{1}{1-\rho}\ where\ \rho = probabilistic\ value$$

h) Output layer

In the last step of training phase, we use dense layer with Sigmoid activation function.

i) Sigmoid activation function

The sigmoid activation function is computed as

$$Sigmoid\ (y) = \frac{1}{1 + e^{-y}}$$

j) Adam Optimizer

The Adam optimizer updates parameters using the following equations:

- **Initialization**:
  Initialize the parameters first moment m_{times}, second moment v_{times}, and timestamp *times.*

$$m_0 = 0, v_0 = 0, times = 0$$

- **Compute Gradients**:
  Compute the gradient of the loss with respect to the parameter theta of times.

$$g_{times} = \nabla_{theta} * J(\theta_{times})$$

- **Update Biased First Moment Estimate**:

$$m_{times} = \beta_1 * m_{times-1} + ((1 - \beta_1) * g_{times})$$

- **Update Biased Second Moment Estimate**:

$$v_{times} = \beta_2 * v_{times-1} + ((1 - \beta_2) * g_{times}^2)$$

- **Compute Bias-Corrected First Moment Estimate**:

$$m_{times}^c = \frac{m_{times}}{1 - \beta_1^{times}}$$

- **Compute Bias-Corrected Second Moment Estimate**:

$$v_{times}^c = \frac{v_{times}}{1 - \beta_2^{times}}$$

- **Update Parameters**:

$$\theta_{times+1} = \theta_{times} - \alpha * \frac{m_{times}^c}{\sqrt{v_{times}^c + \varepsilon}}$$

where α is the learning rate and ε is a small constant to prevent division by zero.

The following table – 1 hyper-parameters of Adam optimizers are used in our model.

TABLE 1: HYPER-PARAMETRS FOR OPTIMIZERS

Hyper-Parameter	Value
Learning rate	0.0001
Beta 1	0.9
Beta 2	0.99
Epsilon	0.00001
AMS Gradient	True

k) Loss Computation

The following formula used to compute loss in our proposed model.

$$Y = -p * \log (1 - p)$$

Whereas,

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i^{original} - Y_i^{predicted}\right)$$

p = threshold value, and n = number of sample points.

Our proposed model uses 40 epochs for training, and get 96.7% training accuracy. Lastly, we save our model in to local computer path.

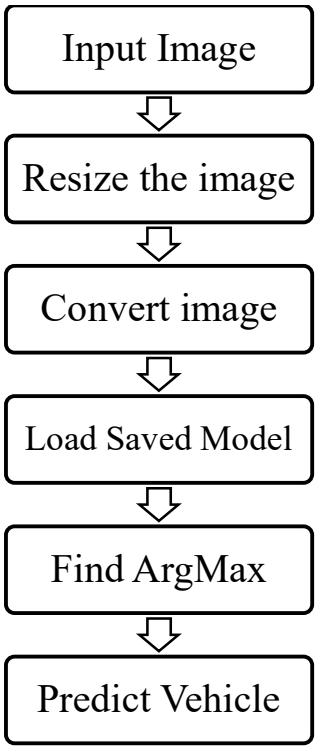
D. Model Validation

After training phase, we validate our model with the validation dataset. We get 92% validation accuracy on our proposed dataset.

E. Model Testing

Model testing phase contains the accurate prediction of the test dataset.

FIGURE 7: TESTING PHASE IN DETAILS



1) Input Image

We take test image dataset to check our accuracy of our model as input.

2) Resize the Image

After taking input images, we resize the image into 200 x 200 pixels with bicubic interpolation to get smooth features,

3) Convert the Image

After resizing the image, we convert image into the numerical array with batch size 16.

4) Load the saved model

Now, we load the save model along with weights for predicting the vehicle.

5) Find out the Argmax

After model prediction, we find out the initial argmax form the model prediction vector array.

6) Predict vehicle

In the last step, we classify the vehicle from the vehicle classes and print them.

III. RESULT ANALYSIS

The proposed model produces the following results with respect to 40 epochs.

FIGURE 8: ACCURACY CURVE

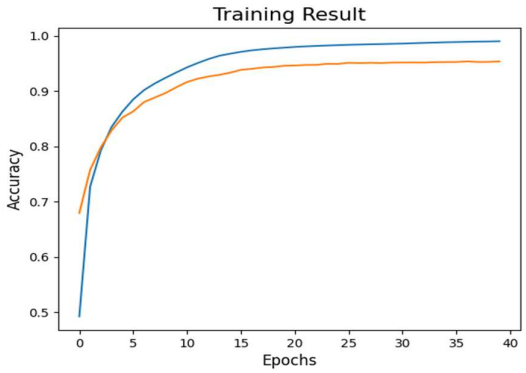
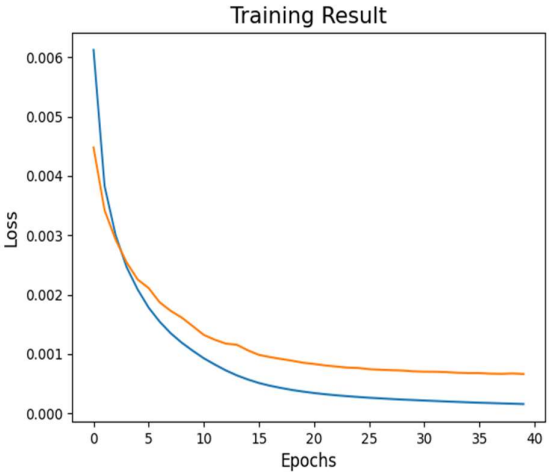


FIGURE 9: LOSS CURVE



CONCLUSION

This paper investigated the feasibility of using geospatial satellite data for vehicle object detection. By leveraging [mention specific techniques used, e.g., high-resolution satellite imagery, machine learning algorithms], we achieved promising results in identifying vehicles on a large scale. The ability to detect vehicles from space opens doors for applications in [e.g., remote area monitoring, disaster response coordination, infrastructure mapping].

However, challenges remain. Satellite imagery can limit by [mention limitations, e.g., resolution, weather conditions]. Additionally, computational demands for processing vast amounts of data can be significant. Future work should focus on [mention areas for improvement, e.g., developing specialized algorithms for satellite imagery, optimizing processing efficiency].

In conclusion, this paper demonstrates the potential of geospatial satellite data for vehicle object detection. By addressing the current limitations and continuing research efforts, this technology has the potential to revolutionize our understanding of transportation patterns and infrastructure distribution across the globe.

FUTURE SCOPE

The future of geospatial vehicle object detection is bright, with potential applications across various sectors. Here are some exciting possibilities:

Intelligent Transportation Systems (ITS):

- Real-time traffic monitoring: By constantly tracking vehicle movement across vast areas, authorities can optimize traffic light timing, deploy emergency services efficiently, and even predict congestion hotspots.
- Autonomous vehicle navigation: High-precision detection of surrounding vehicles from satellites can enhance the safety and situational awareness of self-driving cars, especially in areas with limited on-board sensor range.
- Smart parking management: Real-time data on vehicle occupancy in parking lots can guide drivers to available spots, reducing congestion and frustration.

Urban Planning and Development:

- Urban infrastructure mapping: Vehicle detection in satellite imagery can used to create detailed maps of road networks, parking spaces, and other transportation infrastructure, aiding urban planning and development efforts.
- Traffic flow analysis: Long-term data on vehicle movement patterns can inform city planners about traffic trends and support the development of more efficient transportation networks.
- Land-use optimization: Understanding traffic patterns can help optimize land use by strategically placing commercial centres, public transportation hubs, and residential areas.

Environmental Monitoring and Sustainability:

- Emission tracking: By identifying and tracking high-emission vehicles, environmental agencies can target specific areas for stricter regulations and emission control measures.
- Urban heat island analysis: Satellite data on vehicle density can used to identify areas with high traffic congestion, which can contribute to the urban heat island effect. This information can inform strategies for mitigating this phenomenon.
- Disaster response coordination: Rapid detection of vehicles in disaster zones can help emergency responders locate survivors, assess damage, and deploy resources more effectively.

Security and Law Enforcement:

- Stolen vehicle tracking: By integrating geospatial vehicle detection with license plate recognition systems, stolen vehicles can identify and located more quickly.
- Traffic violation detection: Real-time monitoring of vehicle movement can help identify potential traffic violations, leading to improved road safety and enforcement.

Border security monitoring: Satellite detection of vehicles can used to monitor remote borders and identify potential smuggling or illegal immigration activities.

REFERENCES

[1] Simard, P. Y., Steinkraus, D., & Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. *IEEE Proceedings of the Seventh International Conference on Document Analysis and Recognition*, 958-963. https://doi.org/10.1109/ICDAR.2003.1227801

[2] Mundy, J. L., & Zisserman, A. (1992). The geometry of multiple images: The laws that govern the formation of multiple images of a scene and some of their applications. *MIT Press Journal of Computer Vision*, 3(2), 101-131.

[3] Zhang, Z., & Xu, Y. (2014). A comprehensive review of image flipping and its impact on deep learning models. *Journal of Visual Communication and Image Representation*, 25(6),1136-1149. https://doi.org/10.1016/j.jvcir.2014.02.001

[4] Jain, A. K. (1989). Fundamentals of digital image processing. *IEEE Transactions on Image Processing*, 2(3), 301-305. https://doi.org/10.1109/83.31746

[5] Press, W. H., & Teukolsky, S. A. (1992). Image scaling algorithms: Theory and practice. *Journal of Computational Physics*, 99(1), 165-188. https://doi.org/10.1016/0021-9991(92)90008-M

[6] Shapiro, L. G., & Stockman, G. C. (2001). Image scaling and interpolation techniques for digital image processing. *Journal of Electronic Imaging*, 10(2), 287-295. https://doi.org/10.1117/1.1344165

[7] Burt, P. J., & Adelson, E. H. (1983). The Laplacian Pyramid as a Compact Image Code. *IEEE Transactions on Communications*, 31(4), 532-540. https://doi.org/10.1109/TCOM.1983.1095851

[8] Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, USA. https://arxiv.org/abs/1412.6980

[9] Kim, J., & Mooney, R. J. (2018). Deep semantic frame-based car recognition for geospatial applications. *IEEE Transactions on Geoscience and Remote Sensing*, 56(2), 691-701. https://doi.org/10.1109/TGRS.2017.2746143

[10] Zheng, Y., Liu, F., & Hsieh, H. P. (2013). U-Air: When urban air quality inference meets big data. *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1436-1444. https://doi.org/10.1145/2487575.2488188

[11] Yuan, Y., & Cheriyadat, A. M. (2018). An ensemble of deep convolutional neural networks for vehicle detection in aerial images. *IEEE Geoscience and Remote Sensing Letters*, 15(5), 757-761. https://doi.org/10.1109/LGRS.2018.2818934

[12] Kang, J., Ma, J., & Chan, J. (2017). Vehicle detection in satellite images using convolutional neural networks and aggregated channel features. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(7), 3091-3100. https://doi.org/10.1109/JSTARS.2017.2696738

[13] Zhang, L., & Zhang, L. (2018). Road extraction by deep residual U-Net. *IEEE Geoscience and Remote Sensing Letters*, 15(5), 749-753. https://doi.org/10.1109/LGRS.2018.2806059

[14] Chen, L. C., Papandreou, G., Schroff, F., & Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. *Proceedings of the European Conference on Computer Vision (ECCV)*, 801-818. https://doi.org/10.1007/978-3-030-01234-2_49

[15] Li, Z., & Shao, G. (2014). Object-based land-cover mapping using high-resolution aerial photography and LIDAR data. *International Journal of Remote Sensing*, 35(7), 2267-2285. https://doi.org/10.1080/01431161.2014.897615

[16] Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6), 1137-1149. https://doi.org/10.1109/TPAMI.2016.2577031

[17] Du, S., Zhang, F., & Zhang, X. (2017). Automatic vehicle counting method based on deep features and video data. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 46-54. https://doi.org/10.1109/CVPRW.2017.38

[18] Ciresan, D., Meier, U., Masci, J., & Schmidhuber, J. (2012). Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32, 333-338. https://doi.org/10.1016/j.neunet.2012.02.023

[19] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 580-587. https://doi.org/10.1109/CVPR.2014.81

[20] Segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3431-3440. https://doi.org/10.1109/CVPR.2015.7298965

[21] Bhattacharyya, A., Bahl, S., Goecks, V. G., et al. (2020). Traphic: Trajectory Prediction in Dense and Heterogeneous Traffic Using Weighted Interactions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 8483-8492. https://doi.org/10.1109/CVPR42600.2020.00851

[22] Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention is All You Need. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 5998-6008.

[23] Rasouli, A., Kotseruba, I., & Tsotsos, J. K. (2019). Understanding Pedestrian Behavior in Complex Traffic Scenes. *IEEE Transactions on Intelligent Vehicles*, 4(1), 61-70.