Terrain surveillance system with drone and applied machine vision

Shreya Viswanath¹, Rohith Jayaraman Krishnamurthy¹*, Sunil Suresh¹

¹Department of Applied Computing Sciences, Madras Scientific Research Foundation, Chennai, Tamil Nadu, India.

*rohith.jk@ubc.ca

Abstract. Road accidents are a major contribution to the Annual death rates all over the world. India, ranks first globally in the number of fatalities from road accidents. According to the Ministry of Roads & Transportation, India saw over 440,000 road accidents in 2019. As a result, over 150,000 lives were lost. Poor road conditions contribute to these directly and indirectly. In India, safety standards and conditions of roads are maintained by local bodies in a given area of jurisdiction. While there have been several attempts at improving the quality of roads, weren't instrumental in giving proper results [42]. A recent study suggested that Artificial Intelligence (AI) might help achieve the goals. Some of the AI applications have had better results when powered with Computer Vision. While computer vision has been previously used to identify faults in roads, it is not widely implemented or made available for public use. Road inspection still largely remains a time-consuming manual task, hindering the maintenance process in most cities. Moreover, being unaware of unattended faults on roads is often the cause of road accidents, especially in rough weather conditions that make it impossible for drivers to visually gauge any dangers on their route. The proposed model uses a transfer-learning approach; using Mask R-CNN in identifying the defects at an instance level segmentation. As adding this, it requires less labelling and an additional mask helps in blocking out extra noise around the images. This paper trains a Mask R-CNN architecture-based model to identify potholes, discontinuous roads, blind spots, speed bumps, and the type of road--gravel, concrete, asphalt, tar, or mud--with a dataset of images obtained from a drone. The model is further trained to create depth maps and friction estimates of the roads being surveyed. Once trained, the model is tested on a drone-captured live feed of roads in Chennai, India. The results, once sufficiently accurate, will be implemented in a practical application to help users assess road conditions on their path.

1. Introduction

Road safety has been dwindling at an alarming rate over the past few decades. The main reason for the latter is either the driver’s fault or mechanical reasons. However, faulty road conditions combined with low visibility are partly to blame for this [42]. In India alone, nearly 4,37,396 accidents occur every year due to bad roads, as reported by the Ministry of Road Transport and Highways, India [1,42]. Manual inspection is labor intensive and extremely time-consuming, which has caused researchers to develop other methods that streamline this process and make it easier for authorities to keep track and rectify faulty road conditions at the earliest.
There have been some advances in the usage of machine learning for fault detection, mostly potholes, over the past few years. Song et al. [2] used a smartphone sensor to obtain movement information, which was then fed into an Inception V3 based classifier [3] to scale up the network in a way that utilizes the added computation as efficiently as possible. Zhang et al. [4] used CrackNet for pixel-perfect accuracy for pavement distress detection and classification. Maeda et al. [5] used a dataset of smartphone-captured road images to train a Convolutional Neural Network to detect eight classes of road damage under various weather and lighting conditions. Another group [6] tested different versions of MobileNet and Inception models to distinguish between pothole and non-pothole in images. A novel method of feeding thermal images of roads into a CNN was proposed by Aparna et al. [7] to eradicate the effects of varying weather and visibility of roads on the accuracy of pothole detection. Dhiman et al. [8] implemented transfer learning using Mask R-CNN and YOLO V2 (a single-stage real-time object detection model); the former was found to have better precision and recall, while the latter was more suitable for real-time predictions. Two other machine learning algorithms, Least Squares Support Vector Machines (LS-SVM) and Artificial Neural Networks (ANNs) were trained [9] to produce a classification accuracy rate of 89% and 86%, respectively. Several other methods have also been tried and implemented for similar pothole and road distress classification tasks [10, 11, 12, 13].

This paper proposes employing a Mask R-CNN architecture for instance segmentation which creates a soft pixel mask in the image over the pothole location in addition to a bounding box as in other object detection architectures. In addition to potholes, the model has also been trained to detect and classify speed bumps and different kinds of roads, such as those made of mud, gravel, concrete, asphalt, and tar. The motivation behind the development of this classifier is to support data collection for more effective road maintenance and help reduce the number of road accidents.

2. Architecture Used and System Specification

2.1 Computer Vision
Computer vision is a subset of artificial intelligence in which a computer is trained to give it a human-like ability not just to see, but also understand and make inferences about the observed data [14, 15]. Machine learning algorithms allow for a computer to learn from observed data without being explicitly programmed; this is made possible by using training data without any explicit rules being defined by the human operator [14]. Object classification, detection, localization, and instance segmentation are some of the tasks that can be performed. The main goal is to extract useful information for applications such as Optical Character Recognition (OCR) [16], medical imaging [17], surveillance [18], biometrics [19], and so on.

2.2 Remote Sensing
Remote sensing is the process of observing and acquiring information about an object without being in physical contact with it. Some examples of the information that can be obtained by remote sensing are depths and heights of geographical features, emissions, water levels, earthquake detection, etc. There are two kinds of remote sensing – active and passive. This paper uses passive remote sensing in which the light reflected or emitted by the object of interest is detected by the passive sensor. Here, video and photography of roads are used to identify and classify road types and different faults. This photography and live feed for testing is facilitated by a drone and the proposed machine learning model carries out the identification and classification steps. Once classified, these defects are represented on an app using the Google Maps API.

2.3 Convolutional Neural Networks (CNNs)
CNNs are deep learning architectures that consist of convolutional layers, pooling layers, an activation function, and a fully connected layer. The convolutional layers act as feature extractors. It extracts
features from the image such as edges and corners. This layer takes an image matrix and a filter as input and convolves the two to form a feature map. The pooling layers produce a fixed size feature map from non-uniform inputs, thereby reducing the dimensionality and number of training parameters. This decreases the computational power required to process the data and also extracts dominant features that are augmentation invariant. Finally, the fully connected layer flattens the image into a vector and produces the output which classifies the input images [20]. Some widely used powerful CNN architectures are AlexNet, VGGNet, and ResNet [21, 22, 23].

2.4 Region Based Convolutional Neural Networks (CNNs)
In R-CNN, a CNN is first pre-trained in image classification tasks. The algorithms then use selective search to propose regions in the image and checks if any of these regions contain an object. The selective search identifies multiple regions that can be combined to form a larger region based on various parameters like similarity in color, size, or shape. Once these regions are identified, the final Region of Interest is proposed as a potential object location. Next, feature vectors are created by passing these Regions of Interests through a CNN, which reduces the dimensions of the image. The feature vectors are then passed through a Support Vector Machine (SVM) which classifies the feature vectors into different object classes. A confidence score is generated which depicts how confident the algorithm is. The final proposed regions are ones with an Intersection over Union (IOU is an evaluation metric used to measure the accuracy of an object detector algorithm on any given dataset) overlap of over 0.3. For a more accurate localization, a linear regression model is trained to rectify the bounding box offset in the images [24].

2.5 TensorFlow and Keras
TensorFlow is an open-source software library primarily used for the training, inference, and deployment of deep neural networks. Keras is a high-level API written in Python. It facilitates the quick building of neural networks without requiring in-depth knowledge of the mathematics behind them. This is especially advantageous for those who wish to use neural without actually understanding it. Keras also eliminates the need to learn TensorFlow as it uses it in the backend. The proposed algorithm uses Keras as it is more intuitive and requires much lesser coding. Domain Knowledge helps TensorFlow and Keras to get a better understanding the environment in which the target operates thereby allowing it to produce better accuracy and output.

2.6 Mask R-CNN
The Matterport Mask R-CNN [25] has been used for this surveillance algorithm which uses the Mask R-CNN [26] source code built on a Resnet101-FPN backbone. The Mask R-CNN is nothing but an extension of Faster R-CNN [27]. In the Faster R-CNN, the Region Proposal Network (RPN) proposes object bounding boxes and uses ROI Pooling to output a class label and bounding box. The Mask R-CNN (Figure 1) follows the same procedure but has an added output – a binary mask for each Region of Interest (ROI). The Mask R-CNN architecture has been chosen because for detecting objects such as potholes, which often have arbitrary shapes, generating a mask would result in much more accurate predictions. For an instance segmentation task, ROIPool [28] used in Faster R-CNNs would result in unnecessary offsets in mask predictions. Hence, an ROIAlign [27] layer is used in its place, which uses bilinear interpolation for quantization. This means that coordinates of the feature map will have float values which is essential for creating pixel-accurate masks.
In Faster R-CNN, the loss function is the sum of losses due to classification and bounding box regression.

\[ L = L_{cls} + L_{box} \]

In Mask R-CNN, the multitask loss function is similar to that of Faster R-CNN with an added loss due to segmentation. On each ROI, the loss is defined as,

\[ L = L_{cls} + L_{box} + L_{mask} \]

Where,

- \( L_{cls} \) = Loss of Classification
- \( L_{box} \) = Loss of Bounding box regression
- \( L_{mask} \) = Loss of mask.

A per pixel sigmoid is applied to the mask branch and \( L_{mask} \) is the average binary cross-entropy loss only on the kth mask, thereby allowing for mask generation for each class without having to compete with other classes [26]. Defining it as such has resulted in favorable instance segmentation predictions.

\[ L_{mask} = \frac{1}{m^2} \sum \left[ y_{ij} \log y_{ij}^{-k} + \left( 1 - y_{ij} \right) \log \left( 1 - y_{ij}^{-k} \right) \right] \]

Where \( y_{ij} \) is the label of a cell (i,j); \( m \times m \) is the size of the true mask for the region; \( y_{ij}^{-k} \) is the predicted value of the same cell; and \( k \) is the ground-truth class.

2.7 Transfer Learning

Data collection is expensive, time-consuming, and in many situations, not feasible. The collection of repeated data and meeting the requirements of a high number of data points is time striking. In order to mitigate the problem of scarce training data, the transfer learning approach is used. It is against the notion that training data must be “independent and identically distributed.” In this approach, a model that is pre-trained on similar data can be modified and trained to adapt to the task at hand. This reduces the need for large datasets and the training time since it does not have to be trained from scratch with random weight initialization [29, 30]. This algorithm uses weights pre-trained on the COCO dataset [31] which were then adapted to identify the different object classes proposed within the current
dataset classifying road types and damages. The model was trained on Google Research’s product, Collaboratory, using the Python 3 Google Compute Engine Backend (GPU).

3. Hyperparameters Specific to Mask R-CNN

Hyperparameters are values that control the learning process in a machine learning workflow. These parameters are defined by the user before beginning the training process. The hyperparameters unique to Mask R-CNN are as follows:

**Backbone** – A backbone acts as the feature extractor, identifying features like edges and corners. In this paper, a Resnet 101 architecture is used which has a longer training time than Resnet-50 but better accuracy (1-2%).

**Train_ROIs_Per_Image** – This parameter defines the maximum number of ROIs that a Region Proposal Network can generate for a particular input image. The default value of 200 has been used here, but this can be reduced to speed up the learning process.

**Max_GT_Instances** – This parameter assigns an upper limit to the number of ground truth object instances per image. It has been set to 100 for this network architecture but can be reduced to reduce training time.

**Detection_Min_Confidence** – This parameter has been set to 0.7. It determines the level of confidence at which an instance can be classified. That is, it skips detections with confidence less than 70%. The value can be lowered if false positives are acceptable, or can be increased if greater accuracy of classification is required.

**Image_min_dim and Image_max_dim** – These parameters define the size of the image. By default, the images are resized to 1024x1024, but the size can be lowered to speed up training. In this paper, the minimum and maximum dimensions are 800 and 1024 respectively.

**Rpn_class_loss** – This parameter defines the loss due to improper classification of anchor boxes by the Region Proposal Network.

**Rpn_bbox_loss** – This parameter corresponds to localization accuracy of the Region Proposal Network and needs to be tuned if an object is detected correctly but there is an offset from the actual location of the object.

**Mrcnn_class_loss** – This parameter defines the loss due to incorrect classification of a detected object and will need to be tuned if an object is incorrectly classified.

**Mrcnn_bbox_loss** – This parameter defines the loss due to inaccurate localization of bounding box on an accurately classified object.

**Mrcnn_mask_loss** – This parameter accounts for the loss due to inaccuracy at the pixel level of mask generation on accurately classified objects.

The initially assigned value for all the loss weights is 1.0.

| HYPERPARAMETER          | ASSIGNED VALUE |
|-------------------------|----------------|
| Train_ROIs_Per_Image    | 200            |
| Max_GT_Instances        | 100            |
| Detection_Min_Confidence| 0.7            |
| Image_min_dim           | 800            |
| Image_max_dim           | 1024           |
4. Methodology

The experimental methodology (Figure 3) goes as follows:

Figure 3 Experimental Methodology
Several images of different roads and obstacles are gathered with the help of a Drone Camera. The 2D images are then converted to a 3D format for Instance Segmentation. The processed images are fed into an algorithm with appropriate network architecture. The network architecture is then modified by changing the value of the hyperparameter as per the requirement (Table 1, Figure 2). Then, the “Terrain Surveillance System” algorithm was trained until it reached the minimum loss function. Later the algorithm was optimized to obtain the optimal accuracy. The inference graphs thus obtained are then integrated with Google Maps API with the help of the DB2 Database, which is then tested live with a Drone Camera.

![Figure 4 Complete System Architecture](image)

### 4.1 Data Collection and Labelling

There is no single dataset of road features to be able to set a benchmark for road damage detection and compare results against. The dataset of this algorithm is derived from multiple sources. For the pothole images, a Public Domain pothole dataset was used [32]. For the rest of the images i.e., speedbumps, tar roads, asphalt roads, concrete roads, mud roads, and gravel roads, a Chrome extension, Image Downloader [33] was used to scrape hundreds of images off Google Images. Augmentation techniques such as flip were used on some images to increase the size of the dataset.

Since the proposed algorithm adopts a supervised learning approach, it is necessary to annotate the training and test images. It is essential that the data is labelled as accurately and precisely as possible because the quality of the model depends on the standard of the training data itself. The images were labelled an image annotating and bounding box generating tool [34]. The tool requires the user to create bounding boxes capturing the region of interest and label them as required. The bounding box is represented by two location coordinates. The annotations were saved as .xml files in PASCAL VOC format, containing data about location and size of the object. Some other annotation formats are JSON and .txt files. There are eight classes in the custom dataset out of which seven are for the aforementioned categories and one is for the background. The dataset consists of 1219 images resized to 1024 x 1024 pixels, of which 1074 were used for training and 145 were used for testing.

### 4.2 Conversion of 2D images to 3D format for image segmentation

Taking inspiration from [36], this network (Figure 5) uses "deconvolution" layers (i.e., a learned upsampling filter) that upsamples the incoming feature maps. A feature representation that has the same size as the input image is given after the summation of all upsampled feature maps from each level. A SoftMax transform is then applied across channels at each spatial location and another convolution is performed on the summed feature representation. The output thus obtained is
interpreted as a probabilistic disparity-like map. This map and the left view to the selection layer are then fed, which outputs the right view.

![Figure 5 2D to 3D conversion model architecture.](image)

Deconvolutional layers were used to upsample lower layer feature maps before feeding them to the final representation. This was done by reversing the forward and backward computations of a convolutional layer [37].

For Upsampling by factor S, a deconvolutional layer with 2S-by-2S kernel, S by S Stride, and S/2 by S/2 padding was used. The kernel weight \( w \) is then measured by:

\[
C = \frac{2S - 1 - (S \text{mod} 2)}{2S}
\]

\[
W_{i,j} = \left(1 - \left|\frac{i}{S - C}\right|\right)\left(1 - \left|\frac{j}{S - C}\right|\right)
\]

Compared to traditional 2D to 3D conversion by reconstructing with selection network, this network predicts a probability distribution across possible disparity values \( d \) at each pixel location \( D_{i,j}^d \), where \( \sum_{i,j} D_{i,j}^d = 1 \) for all \( i,j \) [38].

4.3 Modifications to the Matterport Mask R-CNN

To train the model on a custom dataset two classes need to be overwritten—the Config class and Dataset class.

A subclass of the Config class was created and named RoadSafetyConfig. The NAME and NUM_CLASSES parameters were overridden with the necessary modifications. As mentioned, the number of classes was changed to 8 and the default batch size of 1 was retained owing to limited GPU power. A new dataset subclass was created and seven different classes within it were defined to load the dataset. A new dataset class was created and seven different classes within it were defined to build the dataset. Similarly, a new config class, RoadSafetyConfig, was created and the NAME and NUM_CLASSES parameters were overridden with the necessary modifications.

4.4 Training

To train a machine learning model, it is provided with some test data to learn from. This test data contains the attributes it is expected to detect. The learning algorithm finds patterns in the data which map input attributes to the target attributes and the output is a model trained to detect these patterns and classify new data accordingly. In technical terms, each neuron in a neural network has hyperparameters i.e., weight(\( w \)) and bias(\( b \)) values associated with it. While training from scratch, these weights are assigned randomly. Since this algorithm uses transfer learning, the weights have
been pre-trained on other datasets. The aim of the training process is to find the ideal values of $w$ and $b$ for each neuron in the neural network. The hyperparameters can be tuned by forward propagation and back propagation. First, the algorithm captures all the temporarily assigned hyperparameter values, and then, during backpropagation, the values are updated with ones that minimize the error function. Multiple such epochs are performed to determine the ideal hyperparameters that minimize loss function to the required extent, ideally a value of 0.05 or lower. Loss rate represents the summation of errors in the model. Lower the loss function, lesser the errors and vice versa. In the pre-trained model, only the ‘head’ layers are trained at first at a rate of 0.001. The learning rate is the amount by which weights are updated after each epoch. It is essential to ensure that the learning rate is neither too high nor too low, as the former could result in increased training time while the latter could result in non-ideal parameters due to being updated too fast. The other layers of the model are frozen so as to avoid tampering with information in future training steps. The next step is fine-tuning where the model is unfrozen and all the layers are trained at once at a rate of 0.0001. The low learning rate is used in order to avoid overfitting. Fine tuning allows the pre-trained model to recognize classes it was not previously trained on i.e., the new classes introduced. The model was trained for 2550 epochs of 100 steps each. Once the model is sufficiently trained to obtain minimal loss function, inference process is carried out to test the performance of the model on new data that it has not encountered before. Unlike the training process, inference does not run a backward pass to update weights based on obtained result. Since the model was trained using Keras, the output Keras weights need to be converted to a TensorFlow frozen inference graph. A script [35] was run to carry out the conversion. Once converted, inference was run on some test images and masks were generated at a high level of accuracy.

![Validation Curve Suggesting Good Fit](image)

The initially assigned value for all the Mask R-CNN specific loss weights is 1.0 and the values after training are mentioned in the table below.

| LOSS WEIGHT        | ASSIGNED VALUE | FINAL VALUE (After training) |
|--------------------|----------------|-----------------------------|
| Rpn_class_loss     | 1              | 0.00027426                  |
| Rpn_bbox_loss      | 1              | 0.0067                      |
| Mrcnn_class_loss   | 1              | 0.0034                      |
| Mrcnn_bbox_loss    | 1              | 0.0052                      |
| Mrcnn_mask_loss    | 1              | 0.0426                      |
4.5 Integration of proposed algorithm with Google maps API

The Google Maps Application Program Interface (API) provides users with an interface that accesses map, navigation, and satellite information. It also allows users to understand the concept of space and distance [39]. As mentioned in Figure 8, the system runs on B/S architecture. The datasheet obtained from the "Terrain Surveillance System" algorithm is stored in the DB2 database as data layers. The DB2 is an RDBMS (i.e., Relational Database Management System) designed to store, analyse and efficiently retrieve the data. JAVA's popular middleware struts, spring and hibernate are used by the application layer and the logical layer.

The system uses an SSH structure to connect the database with the Google Map-Server. It also shows the result on the JSP page formulated by AJAX [40,41].

The system’s concrete function includes:
- Enlargement, reduction, and dragging of geometrical position
- Fast map location fixation
- Eagle-eyed, the satellite synthesizes, the Street view function
- Real-time traffic flow function
- Data on type of road- asphalt, mud, tar, concrete, and so on
- Data on any obstacle or hurdle on road including potholes, speed breakers, diversions, accidents, and so on
- Multi-spot measuring distance function.
Figure 9 Representation of Terrain Surveillance System with Google Maps API. Black dots represent Potholes, Red lines represent Speed Bumps

5. Testing

A drone flew around several streets in Chennai to get aerial shots of the roads. These pictures were captured at different times of the day and in varying weather conditions to test the model’s accuracy in different lighting and angles. The model recognizes features, extracts them, and forms a feature map that is used to classify them into different classes. Next, Regions of Interest are proposed for further localization of different features. The final ROIs are ones with an IoU greater than 0.3. Finally, pixel masks are generated to accurately determine the location of a particular road type or feature.

The model was tested in real-time with a live feed from the drone. DJI Inspire 2 (Figure 11) is used to collect data for training the custom model and testing it. The drone camera has a flight time of 27 minutes, a range of 7 kilometres and a payload of 4.5 kilograms. The drone camera has a resolution of 1080p and a highly sophisticated image processor.

Figure 10 Comparison of losses (The table shows that Mask RCNN is a better choice.)
The following results were obtained:

![Concrete Road](image1)  
**Figure 12** Concrete Road  
![Potholes](image2)  
**Figure 13** Potholes  
![Speed Bump](image3)  
**Figure 14** Speed Bump

6. **Summary**

This paper has tried to address the unchecked hurdles and obstacles on roads that lead to such accidents. An algorithm was proposed to identify, evaluate and analyse various parameters of roads to help reduce the number of accidents. It was trained with network architectures such as SSD Mobilenet, Faster RCNN, CNN, and Mask RCNN. The results were compared, and Mask RCNN proved to be the best. The Mask R-CNN model was trained to detect and map out different road types and defects, and a loss function of 0.0582 was attained. On running inference, the model generated pixel masks and classified images with an accuracy of 81% at a speed of 443ms. The algorithm was then clubbed with Google Maps API via DB2 database and the drone camera to help make a better navigation guidance system that gives holistic information about the chosen routes. Further, the deployment of the model with integrated sensors can be used to send alerts to authorities to fix the faults as soon as possible to avoid mishaps. This implementation could lead to a steep decrease in the number of road accidents.

**References**

[1] “ROAD ACCIDENTS IN INDIA.” -2018, 2019; vol-1 pp 200-215; 2020. [Online]. Available: www.morth.nic.in.

[2] H. Song, K. Baek, and Y. Byun, “Pothole detection using machine learning,” Adv. Sci. Technol. Lett., vol. 150, pp. 151–155, Feb. 2018.

[3] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in Proc. CVPR, 2016, pp. 2818–2826.

[4] A. Zhang et al., “Automated Pixel-Level Pavement Crack Detection on 3D Asphalt Surfaces Using a Deep-Learning Network,” Comput. Civ. Infrastruct. Eng., vol. 32, no. 10, pp. 805–819, Oct. 2017, doi: 10.1111/mice.12297.

[5] H. Maeda, Y. Sekimoto, T. Seto, T. Kashiyama, and H. Omata, “Road Damage Detection and Classification Using Deep Neural Networks with Smartphone Images,” Comput. Civ. Infrastruct. Eng., vol. 33, no. 12, pp. 1127–1141, Dec. 2018, doi: 10.1111/mice.12387.

[6] K. E. An, S. W. Lee, S. K. Ryu, and D. Seo, “Detecting a pothole using deep convolutional neural network models for an adaptive shock observing in a vehicle driving,” in 2018 IEEE International Conference on Consumer Electronics, ICCE 2018, Mar. 2018, vol. 2018-January, pp. 1–2, doi: 10.1109/ICCE.2018.8326142.

[7] Aparna, Y. Bhatia, R. Rai, V. Gupta, N. Aggarwal, and A. Akula, “Convolutional neural networks-based potholes detection using thermal imaging,” J. King Saud Univ. - Comput. Inf. Sci., Feb. 2019, doi: 10.1016/j.jksuci.2019.02.004.

[8] A. Dhiman and R. Klette, “Pothole Detection Using Computer Vision and Learning,” IEEE Trans. Intell. Transp. Syst., vol. 21, no. 8, pp. 3536–3550, Aug. 2020, doi: 10.1109/TITS.2019.2931297.
[9] N. D. Hoang, “An Artificial Intelligence Method for Asphalt Pavement Pothole Detection Using Least Squares Support Vector Machine and Neural Network with Steerable Filter-Based Feature Extraction,” Adv. Civ. Eng., vol. 2018, pp. 1–12, Jun. 2018, doi: 10.1155/2018/7419058.

[10] M. Eisenbach et al., “How to get pavement distress detection ready for deep learning? A systematic approach,” in Proceedings of the International Joint Conference on Neural Networks, Jun. 2017, vol. 2017-May, pp. 2039–2047, doi: 10.1109/IJCNN.2017.7966101.

[11] H. Chen, M. Yao, and Q. Gu, “Pothole detection using location-aware convolutional neural networks,” Int. J. Mach. Learn. Cybern., vol. 11, no. 4, pp. 899–911, 2020, doi: 10.1007/s13042-020-01078-7.

[12] A. Kumar, Chakrapani, D. J. Kalita, and V. P. Singh, “A modern pothole detection technique using deep learning,” in 2nd International Conference on Data, Engineering and Applications, IDEA 2020, Feb. 2020, pp. 1–5, doi: 10.1109/IDEA49133.2020.9170705.

[13] Srinidhi, Gorityala and SM, Renuka Devi, Pothole Detection using CNN and AlexNet (June 10, 2020). 2nd International Conference on Communication & Information Processing (ICCIP) 2020, Available at SSRN: https://ssrn.com/abstract=3648822 or http://dx.doi.org/10.2139/ssrn.3648822

[14] D. Gries and F. B. Schneider, Texts in Computer Science, 1998 vol. 42. 2010. https://doi.org/10.1007/978-1-4757-2920-7

[15] S. Khan, H. Rahmani, S. A. A. Shah, and M. Bennamoun, “A Guide to Convolutional Neural Networks for Computer Vision,” Synth. Lect. Comput. Vis., vol. 8, no. 1, pp. 1–207, 2018, doi: 10.2200/s00822ed1v01y201712cov015.

[16] S. Shreya, Y. Upadhyay, M. Manchanda, R. Vohra and G. D. Singh, "Optical Character Recognition using Convolutional Neural Network," 2019 6th International Conference on Computing for Sustainable Global Development (INDIACOM), New Delhi, India, 2019, pp. 55-59.

[17] T. Pun, G. Gerig, and O. Ratib, “Image analysis and computer vision in medicine,” Comput. Med. Imaging Graph., vol. 18, no. 2, pp. 85–96, Mar. 1994, doi: 10.1016/0895-6111(94)90017-5.

[18] Shenpu, G., Saleem Durai, M.A. Intelligent video surveillance: a review through deep learning techniques for crowd analysis. J Big Data 6, 48 (2019). https://doi.org/10.1186/s40537-019-0212-5

[19] S. Arora and M. P. S. Bhatia, "A Computer Vision System for Iris Recognition Based on Deep Learning," 2018 IEEE 8th International Advance Computing Conference (IACC), Greater Noida, India, 2018, pp. 157-161, doi: 10.1109/IADCC.2018.8692114.

[20] Indolia, S., Goswami, A. K., Mishra, S. P., & Asopa, P. (2018). Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach. Procedia Computer Science, 132, 679–688. doi:10.1016/j.procs.2018.05.069

[21] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84–90.

[22] Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[23] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). doi:10.1109/cvpr.2016.90

[24] Girshick, R. Girshick, J. Donahue, S. Member, T. Darrell, and J. Malik, “Region-based Convolutional Networks for Accurate Object Detection and Segmentation,” pp. 1–16.

[25] Waleed Abdulla, Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow, 2017, Github, GitHub repository, https://github.com/matterport/Mask_RCNN

[26] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).

[27] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, 2017, doi: 10.1109/TPAMI.2016.2577031.
[28] R. Girshick, “Fast R-CNN,” Proc. IEEE Int. Conf. Comput. Vis., vol. 2015 International Conference on Computer Vision, ICCV 2015, pp. 1440–1448, 2015, doi: 10.1109/ICCV.2015.169.

[29] Kůrková, V., Manolopoulos, Y., Hammer, B., Iliadis, L., & Maglogiannis, I. (Eds.). (2018). Artificial Neural Networks and Machine Learning – ICANN 2018. Lecture Notes in Computer Science. doi:10.1007/978-3-030-01424-7.

[30] Larsen-Freeman, D. (2013). Transfer of Learning Transformed. Language Learning, 63, pp.107-129.

[31] Lin TY. et al. (2014) Microsoft COCO: Common Objects in Context. In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8693. Springer, Cham. https://doi.org/10.1007/978-3-319-10602-1 48

[32] Sachin Patel. (2018). Pothole Image Data-Set, Version 2. Retrieved September 29, 2020 from https://www.kaggle.com/sachinpatel21/pothole-image-dataset/version/2.

[33] Image Downloader- Google Chrome August 2020- https://chrome.google.com/webstore/detail/image-downloader/cnppnohrfphjiggeabnkjhpaldj

[34] Tzutalin. LabelImg. Git code (Jan 2015). https://github.com/tzutalin/labelImg

[35] Bendangnuksung, MRCNN Model conversion, Oct 2020, Github, Github repository, https://github.com/bendangnuksung/mrcnn_serving_ready

[36] A. Pejic, S. Pletl and B. Pejic, "An expert system for tourists using Google Maps API," 2009 7th International Symposium on Intelligent Systems and Informatics, 2009, pp. 317-322, doi: 10.1109/SISY.2009.5291141.

[37] Fischer, P., Dosovitskiy, A., Ilg, E., H’ausser, P., Hazurba,s, C., Golkov, V., van der Smagt, P., Cremers, D., Brox, T.: Flownet: Learning optical flow with convolutional networks. arXiv preprint arXiv:1504.06852 (2015)

[38] Mathieu, M., Couprie, C., LeCun, Y.: Deep multi-scale video prediction beyond mean square error. arXiv preprint arXiv:1511.05440 (2015)

[39] Fang-Ming Hsu; Yu-Tzeng Lin; Tu-Kuang Ho (2012). Design and implementation of an intelligent recommendation system for tourist attractions: The integration of EBM model, Bayesian network and Google Maps, 39(3), 3257–3264. doi:10.1016/j.eswa.2011.09.013

[40] HE Cheng-wan, WAN Ming. The Application and Research on Google Maps Based on Ajax [J]. Computer Knowledge and Technology(Academic Exchange), 2008,3(11):973-974

[41] XU Zhuokui, On Applications of AJAX in WEBGIS [J]. Journal of Geomatics,2007,32(1):3-5

[42] “ROAD AND TRAFFIC ACCIDENTS”- Accidental Deaths & Suicides in India 2019, pp 117-128- https://ncrb.gov.in/sites/default/files/Chapter-1A-Traffic-Accidents_2019.pdf