

Reidentification of Trucks in Highway Corridors using Convolutional Neural Networks to Link Truck Weights to Bridge Responses

Rui Hou^a, Seongwoon Jeong^b, Kincho H. Law^b, Jerome P. Lynch^a

^a Dept. of Civil and Environmental Engineering, University of Michigan, Ann Arbor MI 48109;

^b Dept. of Civil and Environmental Engineering, Stanford University, Stanford CA 94305

ABSTRACT

The widespread availability of cost-effective sensing technologies is translating into an increasing number of highway bridges being instrumented with structural health monitoring (SHM) systems. Current bridge SHM systems are only capable of measuring bridge responses and lack the ability to directly measure the traffic loads inducing bridge responses. The output-only nature of the monitoring data available often leaves damage detection algorithms ill-posed and incapable of robust detection. Attempting to overcome this challenge, this study leverages state-of-the-art computer vision techniques to establish a means of reliably acquiring load data associated with the trucks inducing bridge responses. Using a cyber-enabled highway corridor consisting of cameras, bridge monitoring systems, and weigh-in-motion (WIM) stations, computer vision methods are used to track trucks as they excite bridges and pass WIM stations where their weight parameters are acquired. Convolutional neural network (CNN) methods are used to develop automated vehicle detectors embedded in GPU-enabled cameras along highway corridors to identify and track trucks from real-time traffic video. Detected vehicles are used to trigger the bridge monitoring systems to ensure structural responses are captured when trucks pass. In the study, multiple one-stage object detection CNN architectures have been trained using a customized dataset to identify various types of vehicles captured at multiple locations along a highway corridor. YOLOv3 is selected for its competitive speed and precision in identifying trucks. A customized CNN-based embedding network is trained following a triplet architecture to convert each truck image into a feature vector and the Euclidean distance of two feature vectors is used as a measure of truck similarity for reidentification purposes. The performance of the CNN-based feature extract is proved to be more robust than a hand-crafted method. Reidentification of the same vehicle allows truck weights measured at the WIM station to be associated with measured bridge responses collected by bridge monitoring systems.

Keywords: Computer vision, convolutional neural networks, deep learning, vehicle detection, truck reidentification, structural health monitoring

1. INTRODUCTION

Tremendous progress has been made over the past decade in computer vision technology. Deep learning-based approaches to object detection, object tracking and semantic segmentation have shown remarkable performance in solving fundamental problems in many application areas including in intelligent transportation systems (ITS). Particularly, learning-based vehicle detection and reidentification methods have been previously proposed for large-scale urban surveillance¹, traffic flow analysis (*e.g.*, vehicle speed estimation and accident detection)², and smart traffic signal control at road intersections³. Concurrent to the advancement of computer vision has been a growth in the capabilities of cost-efficient sensing technologies including low-cost sensors, wireless telemetry, and “smart” sensors capable of local data processing (*i.e.*, edge computing)⁴. These advances have impacted the structural health monitoring (SHM) field as witnessed by the increasing number of monitoring systems that have been deployed in operational bridges⁵. While existing bridge SHM systems can accurately measure bridge responses (*e.g.*, strain) and environmental conditions (*e.g.*, temperature), most lack the capability of directly measuring traffic loads that induce bridge responses; this renders health assessment algorithms ill-posed and hinders their accuracy and generality^{6,7}. The possibility of vehicle detection by computer vision stands to dramatically enhance bridge monitoring by integrating vehicle load data with bridge response data for SHM analysis.

In this study, a truck reidentification framework is proposed to track truck loads in highway corridors with the aim of linking truck loads to bridge responses. The framework utilizes traffic video feeds acquired at bridges and a WIM

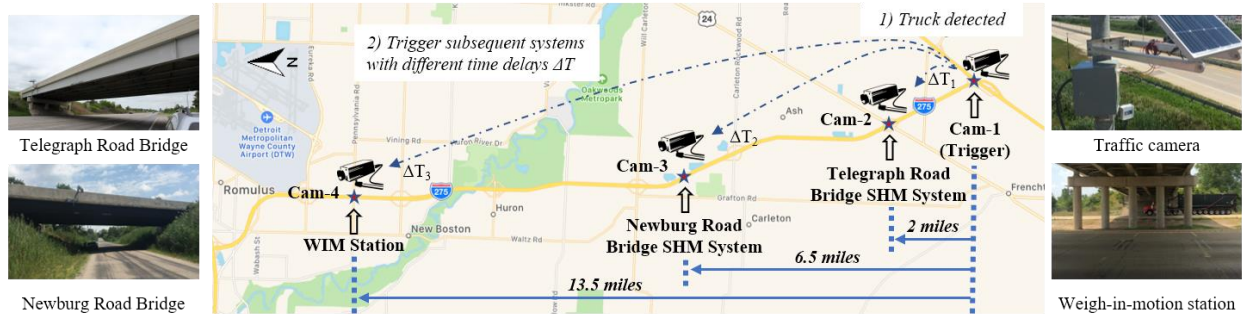


Figure 1. Geographic locations of the bridge structural health monitoring systems and WIM station with the trigger-based data acquisition strategy described.

station along the same corridor as the basis for tracking trucks. First, the framework employs a convolutional neural network (CNN) detector to identify and segment vehicle events at each camera location (*i.e.*, at each bridge or WIM station). Cameras are time synchronized with the bridge monitoring system and the WIM station to ensure trucks identified are linked to the bridge responses and vehicle weight measurements collected, respectively. Next, the trained vehicle detector is modified and fine-tuned within a triplet architecture to extract a vectorized feature representation of each truck image captured by the CNN detectors. Using a mutual nearest neighbor strategy, the Euclidean distance between the feature vectors of each captured truck is selected to serve as a scalar metric of truck similarity; high similarity implies the truck is successfully reidentified. Once a truck is detected at the bridge and later reidentified at the WIM station, measured bridge responses can be accurately associated with the weight parameters (*i.e.*, gross vehicle weight, axle weight distributions, and axle spacings) of the truck as measured by the WIM station. Such a framework allows bridge responses to be precisely mapped to their corresponding vehicular loads. To demonstrate the proposed framework, a cyber-physical system (CPS) architecture consisting of two bridge SHM systems, four traffic cameras and a WIM station is established along a 20-mile highway corridor in southeast Michigan. In this paper, the CPS instrumentation installed along the connected highway corridor is first presented. Next, multiple CNN models are explored for vehicle detection at each measurement point in the CPS (namely at the bridges and WIM station). The various CNN models are compared using precision-recall curves to identify the most robust single-stage detector. In the next section, the implementation of the truck reidentification algorithm is presented. Finally, the paper concludes with a summary of key findings and a description of future work.

2. CONNECTED STRUCTURAL HEALTH MONITORING NETWORK

A cyber-physical system (CPS) is installed along the 20-mile I-275 northbound highway corridor between Newport, Michigan and Romulus, Michigan. The CPS architecture is designed to correlate truck loads to bridge responses as acquired by SHM systems installed on the bridges. The I-275 corridor consists of three types of data acquisition systems: bridge SHM systems, a WIM station and traffic cameras. The SHM systems installed on the bridges utilize wireless sensors to measure bridge responses to vehicular loads. The WIM station measures the gross vehicle weight and the weight distribution on the truck axles. The camera system captures real-time traffic video. The geographic locations of the network components including the SHM systems on the Telegraph Road and Newburg Road bridges, the WIM station and the four cameras (denoted as CAM-1 through CAM-4) are illustrated in Figure 1. The northbound travel distance from CAM-1 where the truck is first identified to the Telegraph Road Bridge, Newburg Road Bridge and WIM station are 2 miles (3.2 km), 6.5 miles (10.4 km) and 13.5 miles (21.7 km), respectively.

2.1 Bridge Structural Health Monitoring Systems

The Telegraph Road Bridge (TRB) and Newburg Road Bridge (NRB) have both been instrumented with wireless sensing networks (WSN) in 2011 and 2016, respectively, for bridge condition monitoring^{8,9}. Both bridges are built in 1973 and carry three lanes of I-275 northbound traffic. The TRB has three spans and a total length of 224 feet (86.3 m). Located 4.5 miles (7.2 km) north of the TRB, the NRB is a single-span bridge spanning 105 feet (32.0 m). Both bridges have a concrete-deck-on-steel-plate-girder composite superstructure consisting of seven girder lines. With the focus being truck reidentification, this paper only presents aspects of the WSN design that are essential to the understanding of the proposed framework; more details can be found in previous publications of the WSN implementations^{8,9}. Each bridge monitoring

system is composed of a WSN of *Narada* wireless sensing nodes communicating (using IEEE802.15.4) directly to a single base station. The base station consists of a single board computer (NVIDIA Jetson TX2 module) which is connected to an LTE modem (AT&T Velocity MF861) for Internet access and to a CC2420 IEEE 802.15.4 RF transceiver for wireless communication with the *Narada* nodes. The function of the base station is to: 1) send operational commands to the *Narada* nodes (*e.g.*, sleep, wake-up); 2) time synchronize the nodes before each data collection cycle; 3) collect sensor time-histories from the nodes; and, 4) push bridge response data aggregated to cloud-hosted services. A cloud-based data management system is built on Microsoft Azure for long-term storage and curation of data¹⁰. The base station is powered by a 12V 40 A-hr sealed lead acid (SLA) battery continuously recharged by a 160W solar panel. Similarly, each *Narada* node is powered by a 12V 2.9 A-hr SLA battery and a small solar panel. Up to four analog sensors (*e.g.*, strain gages, accelerometers) can be interfaced at one time to each *Narada* sensing node for structural response sensing¹¹.

2.2 Weigh-in-motion Station

The weigh-in-motion (WIM) station underneath the Pennsylvania Road Bridge at the north-end of the I-275 corridor measures the weight configuration of passing trucks without altering the flow of traffic. Located 7 miles (11.3 km) north of the NRB, it is the only WIM station on this segment of I-275. Being a two-lane type II station, quartz sensors are only installed under the pavement of the right-most and middle lanes. Each measured weight record contains 9 attributes: 1) measurement timestamp, 2) Federal Highway Administration (FHWA) vehicle class, 3) vehicle speed, 4) vehicle gross weight, 5) number of axles, 6) axle weights, 7) axle spacing, 8) vehicle direction, and 9) lane assignment. The WIM station is operated and maintained by the Michigan Department of Transportation (MDOT). The station continuously collects data and communicates data via a wired fiber network to a remote database maintained by MDOT. These databases can be queried to acquire truck data.

2.3 Traffic Camera System

As highlighted in Figure 1, four traffic cameras are installed along the corridor. The first is at the interchange with I-75 to identify trucks as they enter I-275 at its southern-most point. The next three cameras are at the TRB, NRB and WIM station. Each traffic camera consists of a Logitech C930e webcam enclosed in a water-proof enclosure. It is controlled by an NVIDIA Jetson TX2 module and powered by a 12V 40 A-hr SLA battery (charged using a 160W solar panel). The TX2 module is connected to the Internet through an LTE modem (AT&T Velocity MF861) and to the camera via a wired USB cable. The TX2 contains an embedded integrated 256-core NVIDIA Pascal GPU and a hex-core ARMv8 64-bit CPU complex, which enables stable recording of the camera and real-time embedded image processing (*e.g.*, vehicle detection). The TX2 is used herein to hold the computer vision algorithms for truck identification. The TX2 commands the camera to capture videos at a frame rate of 10 FPS with a resolution of 1280x720 pixels.

2.4 Data Acquisition Process

The data acquisition process of the system is controlled via the Internet using communications based on the User Datagram Protocol (UDP) so as to minimize latency. Each system component including the cameras and the bridge SHM systems are controlled locally by TX2 modules that simultaneously communicate with one another via cellular access to the Internet. To capture truck loads passing the bridges, the bridge SHM systems are programmed to collect data using a trigger-based strategy, as illustrated in Figure 1. The first camera (CAM-1) acts as the primary system trigger by using an embedded detector to identify in real-time trucks entering I-275. Once CAM-1 observes a truck entering the instrumented corridor, it sends a message to each of the system components located north on the corridor to activate the data collection process with an estimated start based on the truck speed. The camera and bridge SHM systems at the same location (*e.g.*, TRB and NRB) have synchronized clocks and start their data collection processes simultaneously. After the camera or bridge SHM system is triggered, it is programmed to collect data for a fixed duration (*e.g.*, 2 mins). It should be noted that multiple trucks can be captured at each location including at the WIM station even though the trigger only needs to detect one truck to activate the network. The video feed and bridge sensor data collected at each location for the same data collection cycle (*i.e.*, collected by the same trigger) is uploaded to the cloud database after the collection process is completed¹⁰.

3. VEHICLE DETECTION

Object detection is a fundamental problem in the field of computer vision that has been extensively studied over the last decade¹². The goal of object detection is to automatically identify, distinguish and locate objects from a camera image.

Recent advances in deep learning have improved object detection techniques. This study explores deep learning methods for vehicle detection using roadside video. There are two main types of learning-based object detectors (*i.e.*, one-stage and two-stage) that utilize convolutional neural networks (CNN) for object detection. Two-stage detectors, such as Regions with CNN features (R-CNN)¹³, Fast R-CNN¹⁴, Faster R-CNN¹⁵ and their variants, generate a plausible set of bounding box regions to identify an object in the first stage and perform object classification and detection refinement in each candidate bounding box in the second stage. Alternatively, one-stage detectors such as Single Shot Detector (SSD)¹⁶, You Only Look Once (YOLO)^{12,17,18}, RetinaNet¹⁹ and their variants, cast object detection into a single regression problem which predicts object classes, bounding boxes and confidence scores simultaneously within a single network inference. While two-stage detectors generally enjoy higher precision compared to one-stage detectors, their computational speeds are slower¹⁹. As fast inference is essential for real-time triggering in this study, only one-stage detectors are considered. Four one-stage object detection paradigms proposed by other researchers are tested for truck identification: YOLOv3¹⁸, YOLOv3-tiny¹⁸, RetinaNet¹⁹ and SSD¹⁶. All four architectures utilize CNNs as the backbone for their feature extractions. Camera images of a pre-determined resolution and size are set as the CNN input while the CNN output is the location (*i.e.*, bounding box) and detection confidence of the target object within the box. A brief description of the four architectures will be presented; interested readers are invited to read the implementation details of each method in the original papers.

3.1 You Only Look Once (YOLO)

The YOLO architecture is a series of single-stage CNN models designed for real-time object detection. When fed with an input image, YOLO first divides the image into a finite number of square cells (S by S). Each cell is responsible for predicting a fixed number of potential object bounding boxes whose centroids fall inside the cell (although the box can be bigger than the cell)¹². The number of predicted objects, B , depends on the number of selected bounding box priors which approximately represent the shape and dimension of the final bounding boxes¹⁷. Associated with each bounding box are 5 items including its x and y coordinates, width, height and a confidence score (between 0 and 1) that measures the likelihood the box contains an object and the accuracy of the box's dimension. Also assigned to the box are C conditional class probabilities corresponding to C detectable object classes with each reflecting the likelihood of the bounding box belonging to an assigned class. The YOLO architecture utilizes its convolutional layers to encode an input image into a tensor with a shape of $S*S*B*(5+C)$ which contains totally $S*S*B$ prediction outputs¹⁷. For each predicted bounding box, its final class confidence scores which has C entries equal the product of the box confidence and the C conditional class probabilities. A softmax function or independent logistic classifiers can be used to determine the final class for each predicted bounding box depending on the specified architecture. Considering there might be multiple bounding boxes for the same object, the YOLO architecture applies non-maximal suppression (NMS) to remove duplications with lower confidence. A global confidence threshold can also be set such that the model only offers object detection predictions with confidence scores higher than the threshold. The third generation of the YOLO architecture, YOLOv3¹⁸, adopts a deep 53-layer CNN architecture (*i.e.*, Darknet-53) with skip connections to extract features and make predictions. Skip connections are those where the outputs of one layer may "skip" numerous sequential layers in the network and sums these skipping outputs to the outputs of lower layers. Skip connections were first proposed in the residual network (ResNet) to enable the convolutional layers to preserve an identity mapping of its input²⁰. Another feature of YOLOv3 is that it incorporates up-sampling techniques similar to feature pyramid networks (FPN) so that object predictions can be performed at different spatial scales thereby allowing smaller objects in an image to be accurately predicted²¹. YOLOv3-tiny follows the architectural principles of YOLOv3 but has a more compact CNN backbone (*i.e.*, 13 convolutional layers) for faster execution in real-time systems. YOLOv3 has three variants supporting image inputs of different sizes. This study selects the YOLO models that accept 416-by-416 pixel images as inputs for a balance of speed and accuracy¹⁸. The input image resolution is fixed by the CNN architecture; images from cameras of different resolutions must be modified to conform to the required input image resolution.

3.2 RetinaNet

The RetinaNet CNN architecture consists of a backbone network and two task-specific subnetworks. The backbone network stacks an FPN on top of ResNet in order to extract several feature maps at different spatial scales from the input image^{19,21,22}. The two subnetworks are responsible for object classification and convolutional bounding box regression based on the feature maps extracted by the backbone network¹⁹. In addition to its innovative unifying architecture, RetinaNet utilizes a novel loss function called focal loss. Focal loss was designed to resolve the challenging issue of class imbalance which is often encountered in one-stage detectors¹⁹. Towards this end, focal loss dynamically scales a cross-entropy function during training to allow the training to focus more on those objects difficult to classify. The version which



Figure 2. Sample images in the customized vehicle detection data set: (top row) images from the Internet; (middle) images from the I-275 corridor; (bottom) images from the I-275 corridor under varying weather conditions.

has the ResNet-50-FPN as its backbone is used in this study because it offers a good balance between speed and precision¹⁹. Input images of the RetinaNet model are resized such that the shorter side equals 600 pixels

3.3 Single Shot Detector (SSD)

The SSD architectures select the VGG-16²² CNN architecture as the base network but discards the fully connected layers and adds auxiliary convolutional layers for extracting features at multiple scales¹⁶. The SSD divides each feature map into grid cells where each cell has a set of default boxes of different dimensions and aspect ratios associated¹⁶. For each default bounding box, the confidence for all object classes are predicted and the box location is adjusted to better match the object location through regression performed by the CNN. The SSD version which takes 300-by-300 pixels images as the network inputs is adopted in this study.

3.4 Model Training

The primary objective of this study is to detect three vehicle classes from camera images: large trucks, pick-up trucks and cars (*e.g.*, sedans, SUV, vans). The four selected single-stage object detectors (YOLOv3, YOLOv3-tiny, RetinaNet, SSD) have their initial CNN architectures fixed with preliminarily weights pre-trained on ImageNet²³ included. These four detectors are further fine-tuned and later tested using a diverse set of images manually prepared to improve the detectors for vehicle detection. The resultant data set is a combination of pictures of trucks crawled from the Internet and those manually selected from the traffic images captured by the four cameras installed along the I-275 corridor. Ground truth bounding boxes are manually labelled in all figures by the authors using a customized annotation tool. The entire data set used to train the four detectors contains a total of 3,500 vehicle images. Approximately 70% of the pictures are used for CNN training with 30% held for testing. As listed in Table 1, the training set has 2,600 vehicle images that contain a total of 4,391 vehicle instances. In contrast, the test set is composed of 900 vehicle images with a total of 1,226 vehicle instances. One additional arrangement of the split is that all the vehicle pictures obtained online belong to the training data set. The reason for selecting online images for training is to generalize the detection capability of the CNN so that it can potentially detect vehicles that may not commonly appear in the videos captured on the I-275 highway corridor. To further generalize the detection capabilities of the four detectors, vehicle images from the Internet and the I-275 cameras are intentionally diversified to include a wide range of weather and illumination conditions (*e.g.*, snow, rain or glare). This tactic ensures the final trained CNN architectures will offer robust detection performance even under harsh operational circumstances. Some sample images used to train the CNN architectures are shown in Figure 2.

The four object detectors are based on open-source scripts published by the authors of the original work. More specifically, the implementation of YOLOv3 and YOLOv3-tiny is based on Darknet which is an open source neural network framework written in C and CUDA²⁴. RetinaNet is based on Detectron which is a software system developed by Facebook based on Python and the Caffe2 deep learning framework for novel object detection models²⁵. The SSD implementation adopts the original open source code using the Caffe framework¹⁶. The model parameters of all four CNN architectures are initialized using the publicly available weights pre-trained based on the ImageNet dataset²³. Only minor changes are made to the original scripts to accommodate the training process based on the images and hardware available (*i.e.*, GPU platform) in this study. For example, the number of classes, the number of GPUs, the learning rate and the batch

Table 1. Statistics of vehicles in the training and testing dataset

Stage	Number of Images	Pct. (%)	Total Vehicle Instances	Pct. (%)	Large Trucks	Pct. (%)	Pick-ups Trucks	Pct. (%)	Small Cars	Pct. (%)
Training	2600	74%	4391	78%	2947	84%	344	66%	1100	70%
Testing	900	26%	1226	22%	566	16%	178	34%	482	30%
Total	3500	100%	5617	100%	3513	100%	522	100%	1582	100%

Table 2. Test results of four selected single-stage vehicle detection models trained using the customized dataset

Model	$mAP^{0.5}$	$mAP^{0.75}$	Time	$AP^{0.5}_{truck}$	$AP^{0.5}_{pickup}$	$AP^{0.5}_{car}$	$AP^{0.75}_{truck}$	$AP^{0.75}_{pickup}$	$AP^{0.75}_{car}$
YOLOv3	93.64%	92.56%	19.7 ms	98.91%	88.44%	93.58%	98.91%	85.73%	93.04%
YOLOv3-Tiny	88.11%	86.20%	3.2 ms	97.46%	82.13%	84.73%	96.65%	79.25%	82.71%
RetinaNet	93.98%	92.50%	77.3 ms	97.67%	90.02%	94.25%	96.80%	86.82%	93.89%
SSD	90.68%	88.57%	48.2 ms	98.03%	81.77%	92.26%	95.89%	78.58%	91.25%

size are modified according to the hybrid (*i.e.*, Internet and from I-275) training set. Both the training and testing processes are performed on an NVIDIA Titan Xp GPU.

3.5 Model Testing

After training, the four detectors are evaluated using the independent testing data set. The performance of each detection is based on average precision (AP) per class²⁶, mean AP (mAP)²⁶ and inference speed. Precision is a measure of the true positives among all of the positive detections in the testing data. Recall is the number of true positives identified among all the positive (true positive plus false negatives). Hence, average precision is an average of the maximum precision metrics at various recall rates. In the four object detectors, a global confidence threshold can be established to control the lower bound of the confidence score that a detector can have, and thus the number of prediction outputs. Consequently, a change of the global confidence threshold leads to a change of the precision and recall obtained on the test data. AP is calculated by averaging maximum precisions at different recall levels per class and mAP is the mean of AP of all detectable classes. In this study, mAP and AP per class are both evaluated for two different intersection over union (IOU) thresholds: 0.5 and 0.75. A higher IOU threshold means the predicted bounding box of the detected object needs to have a larger overlap with the ground truth to be considered as a correct detection. Since predicted bounding boxes are also used for further data analysis such as truck reidentification, the detection accuracy of each single-stage detector at a threshold of 0.75 is explored in addition to the most common threshold of 0.5. The detailed evaluation results for the four detectors are shown quantitatively in Table 2 with the most competitive result bolded in each column. The IOU threshold is denoted using superscripts such as $mAP^{0.5}$. In addition, the precision-recall curves per vehicle class of the four detectors with two different IOU thresholds are presented in Figure 3 for comparison. The inference time refers to the average time needed to perform vehicle detection on one image by forward execution of the CNN architecture.

As shown in Figure 3, all four detectors achieve relatively high mAP s. This may be because vehicles have low variance on their appearance and the test images are limited to ones captured from the four identical cameras along the I-275 corridor. More specifically, YOLOv3 shows the highest accuracy for truck detection while RetinaNet slightly outperforms others in pickup and car detection due to the use of focal loss which addresses the class imbalance between large truck classes and small car classes. However, the lightweight YOLOv3-tiny model exhibits the fastest inference time while still achieving a sufficiently high level of accuracy, especially for truck detection. Consequently, the YOLOv3-tiny model is deployed in the NVIDIA TX modules in the field for real-time truck detection at CAM-1 where latency is the most important performance metric. The YOLOv3-tiny model can run at 15 FPS when embedded in NVIDIA TX2, which satisfies the requirement of real-time detection. When using the model, the confidence threshold is set to 0.8 with the model only returning predictions with class probabilities higher than this threshold. This ensures false alarms are minimized in the deployed system. YOLOv3 is selected as the primary detector used to detect trucks in the bridges and WIM station images offline because real-time performance is not required. As a result, the higher precision and recall performance of YOLOv3 is desirable for confirmation of trucks crossing the bridges and WIM station. The YOLOv3 detection is an offline post-processing process performed on a server using an NVIDIA Titan Xp GPU after all data is uploaded into the cloud. Some qualitative detection results are shown in Figure 4.

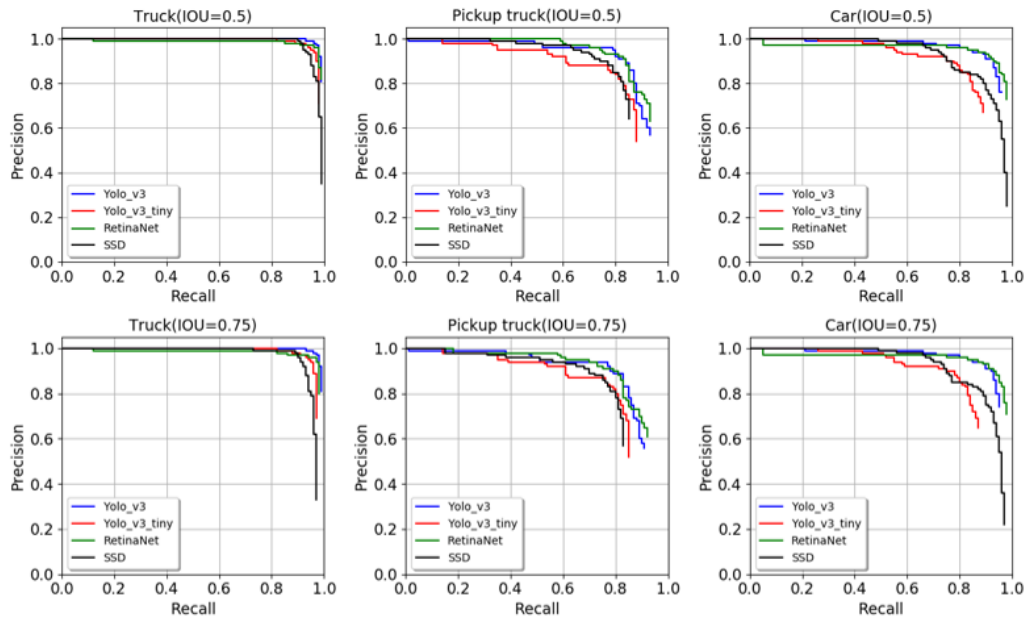


Figure 3. Comparison of precision-recall curves of the trained detection models per object category using different IOU threshold.

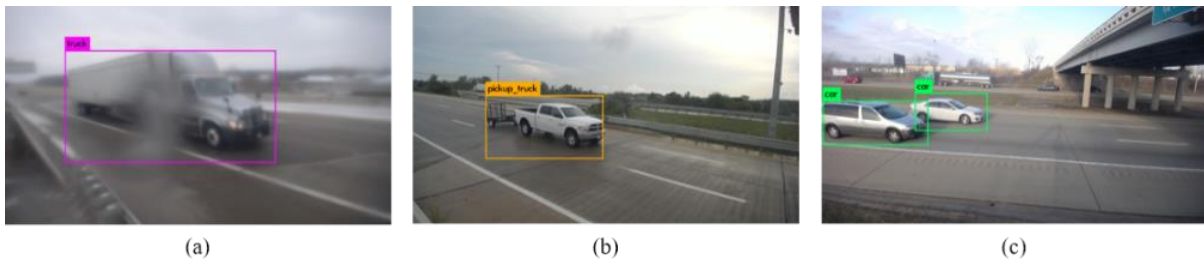


Figure 4. Qualitative vehicle detection results from the trained YOLOv3 model: (a) truck; (b) pickup truck; (c) cars

4. TRUCK REIDENTIFICATION

The YOLOv3-tiny single-stage detector trained in the previous section and embedded in CAM-1 is used for real-time truck detection with the CPS architecture designed to autonomously trigger the bridge monitoring systems and subsequent cameras along the I-275 corridor. This triggering strategy is designed to capture bridge responses and truck weight measurements for the same truck as it may travel north on I-275. To confirm the same truck is evident at each section of the corridor, the YOLOv3 detector is implemented offline to extract truck events at each bridge and WIM station using videos from CAM-2, CAM-3 and CAM-4. Identified truck objects at each camera are collected during the data collection window triggered by CAM-1. This section describes the truck reidentification method used to match the same truck at different locations based on a mutual nearest neighbor strategy using extracted features from the truck objects identified. As multiple images (video frames) are associated with a single truck, the image that has the largest bounding box while shows the entire truck head is selected for the reidentification purpose. The image is cropped around the bounding box to remove redundant background information. Such an image is assumed to contain the most useful information of that truck's appearance and thus gives the best accuracy for reidentification.

Two approaches are explored to extract features from cropped truck images and to measure the similarity between a pair of images. One approach is to use traditional hand-crafted features and match detected key points between two candidate images. For example, the percentage of matched key points can be a direct measure of similarity between two



Figure 6. Same truck with top matched feature points highlighted by the MODS algorithm: (left) TRB and (right) NRB.

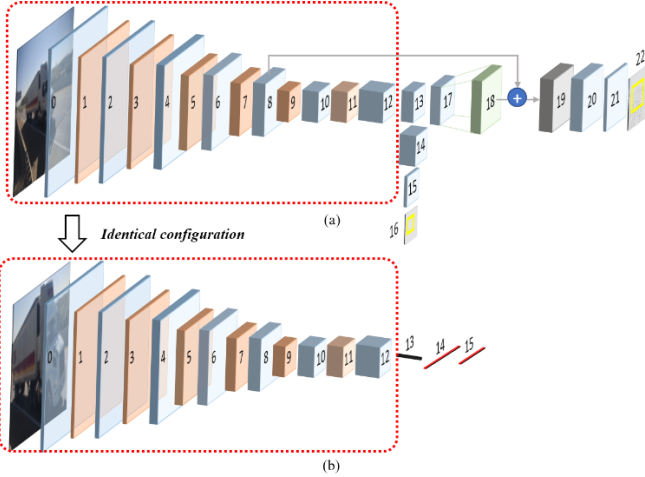


Figure 5. (a) The traditional YOLOv3-tiny CNN architecture; (b) the modified CNN architecture for truck image matching.

Table 3. Layer types and primary parameters of the embedding network for truck matching (Figure 6b)

Layer	Type	#Filter	Size	Stride
0	Convolutional	16	3	1
1	Max pooling		2	2
2	Convolutional	32	3	1
3	Max pooling		2	2
4	Convolutional	64	3	
5	Max pooling		2	2
6	Convolutional	128	3	1
7	Max pooling		2	2
8	Convolutional	256	3	1
9	Max pooling		2	2
10	Convolutional	512	3	1
11	Max pooling		2	2
12	Convolutional	1024	3	1
13	GAP			13
14	Fully connected		1024	
15	Fully connected		512	

trucks from two different images. The second approach is to extract a feature vector from each image using a learning-based CNN model and use the reciprocal of the Euclidean distance between the two feature vectors as a similarity metric.

4.1 Hand-crafted Feature Extraction

In the first approach, the truck reidentification problem is posed as a wide baseline stereo matching problem. Stereo matching aims to estimate the fundamental matrix of a geometric transformation between images and to extract a common set of key points given two images with distinct views (*e.g.*, views differing by rotation, translation or significant changes in illumination) of an identical object²⁷. In a standard baseline matching problem, the first step is to detect key points and describe local features in two input images separately. The second step involves tentatively linking extracted key points of the two images followed by geometric verification using parameter estimation techniques such as RANSAC^{28,29}. In this study, two bounding box images of the same truck taken at different locations are treated like two images taken at the same location with distinct views. Thus, the ratio of the number of matched key points to the number of total extracted key points from the two input images can be used as a metric to describe the similarity between them. Due to large view transformations (*e.g.*, tilt) and environment changes (*e.g.*, illumination) associated with images taken at different locations in the I-275 corridor, commonly used short baseline stereo algorithms such as SIFT³⁰ and SURF³¹ prove to be unsuccessful in robustly detecting key points; they are not considered further due to their lower than desired performance levels. As an alternative, the Affine-SIFT (ASIFT) algorithm which is designed for wide baseline stereo matching is explored³². ASIFT is fully affine invariant and has improved matching accuracy by synthesizing a series of views for both input images through image deformation (*i.e.*, scale and tilt) before the feature detection step³². The approach entails n^2 independent matching problems derived from the original single matching problem where n is the number of synthesized (deformed) views per image. ASIFT also has a step of filtering out accumulated false matches that enhances its performance³². ASIFT is more robust than SIFT and SURF but its speed is not sufficiently fast in this application. In this study, the SIFT and SURF implementations in OpenCV and the ASIFT code published by the original author are evaluated.

The final approach explored is a wide baseline stereo matching algorithm called matching on demand with view synthesis (MODS)²⁹. In addition to view synthesis introduced in ASIFT, the most significant advantage of MODS is the employment of a combination of different detectors and descriptors. Following an iterative strategy, MODS adaptively

attempts more powerful but slower detectors and descriptors until sufficient proof is found for a match²⁹. In the meantime, MODS also synthesizes more views for each image, if necessary. Such an on-demand approach tackles the matching problem with a balance between matching robustness and computational speed. The implementation of MODS published in C++ by the original author is adopted in this study²⁹. An example of matched pairs using the MODS algorithm is shown in Figure 5 with matched key points highlighted between the two views of the same truck at TRB and NRB. It is observed that the algorithm works well even in the case of harsh weather conditions (e.g., raining or snowing) or blurred images due to low light conditions and a slow shutter speed. As shown, MODS is able to match two truck images even when the views are transformed and with a partial occlusion of the complete truck. However, MODS performance is degraded with low contrast images such as those with glare or haze.

4.2 CNN-based Feature Extraction

CNNs have been shown to be a powerful tool for the extraction of meaningful features (e.g., lines, shapes, colored patterns) from images; as a result, they can potentially be used for object reidentification. Within a CNN, each intermediate convolutional layer is responsible for extracting a feature map from previous network outputs. Some particular combination of those convolutional layers can be used to serve as a feature extractor to convert the characteristics of an image into a feature vector^{33,34}. In this study, a CNN-based approach to vehicle reidentification is considered; the approach leverages the previously trained YOLOv3-tiny model to extract features from traffic images for truck reidentification. The first 12 layers of the YOLOv3-tiny model are preserved and combined with: 1) a global average pooling (GAP) layer³⁵ (layer 13) and 2) two fully connected layers as a new embedding network (layers 14 and 15). The architectures of the two models are illustrated in Figure 6 and the layer types and primary parameters of the embedding network are listed in Table 3¹⁸. The first 12 layers of the embedding network are identical to those of the YOLOv3-tiny network and thus each convolutional layer is followed by a batch normalization and then activated by a leaky rectified linear unit (ReLU)¹⁸. The total number of convolutional layers and fully connected layers in the CNN, seven and two, respectively, are determined by ablation experiments in order to extract the most profitable features for vehicle reidentification. The idea of adding the GAP layer is to enforce each feature map to represent meaningful characteristics of trucks and to reduce the dimensionality of the input fed into the subsequent fully connected layers to try to avoid overfitting. The proposed embedding network takes 416-by-416 pixels images as inputs, which is similar to the original YOLOv3-tiny model but is fed cropped truck images instead of the entire raw traffic images. The network output is a 512-entry feature vector for each truck image for matching.

Two general architectures, Siamese network³⁶ and triplet network³⁷, are used during training to learn similarity metrics between input images. The goal is to let the embedding network learn to extract vision features useful for truck reidentification. The primary differences between the two architectures are their input forms and the loss functions used. As shown in Figure 7(a), two identical CNNs, namely the prementioned embedding network, exist in a Siamese network with both CNNs sharing the same weights at all times³⁶. During training, a pair of images either of the same truck or of different trucks is taken as an input to the Siamese network with each image fed to one of the two CNNs. The outputs of the two CNNs (*i.e.*, two feature vectors) are then passed to a contrastive loss function³⁶. The contrastive loss function, L , has the form:

$$L_{contrastive} = (1 - Y) \frac{1}{2} (D_w)^2 + (Y) \frac{1}{2} \{\max(0, m - D_w)\}^2 \quad (1)$$

$$D_w = \|G_w(x_1) - G_w(x_2)\|_2 \quad (2)$$

where Y is the label of an image pair that is set to 0 if two input images belong to the same truck and 1 if they are from different trucks, D_w is the Euclidean distance between the outputs of the two branch networks which can be computed by Equation (2) where $G_w(x_1)$ and $G_w(x_2)$ are the vectorized outputs of the two CNNs, and m is a positive distance margin. The role of the contrastive loss function is to train the embedding networks to pull the feature vectors of two similar trucks together while separating those of dissimilar trucks. If the distance between two different trucks is larger than the specified margin, it then stops contributing to the loss.

A triplet network, shown in Figure 7(b), has three identical embedding networks. All three subnetworks have identical weights. During training, the network is fed with a triplet of truck images without any explicit labels attached. Among the three input images, one image is the anchor image, one is the positive image which shows the same truck as the anchor image and the last is the negative image of a truck that is different from the anchor image³⁷. A triplet loss

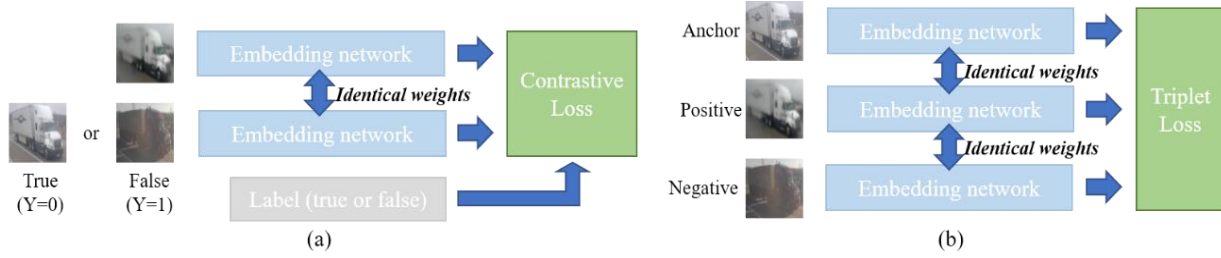


Figure 7. Coupled network architectures for the embedding network training: (a) Siamese network architecture; (b) triplet network architecture.



Figure 8. Samples of manually prepared positive truck image pairs.

function (Equation (3)) is employed to learn the parameters of the embedding network, in such a way that the feature vector of the positive truck image is trained to be closer to that of the anchor image than the negative sample by a margin α ³⁷.

$$L_{triplet} = \sum_{i=0}^n \left[\left\| G_w(x_i^a) - G_w(x_i^p) \right\|_2^2 - \left\| G_w(x_i^a) - G_w(x_i^n) \right\|_2^2 + \alpha \right]_+ \quad (3)$$

To train the embedding network, 1,300 truck image pairs are manually picked by the authors. Within each pair, the two images belong to the same truck but are captured at different locations in the I-275 corridor. Each traffic image is cropped and resized to 416-by-416 pixels to remove background objects; samples are shown in Figure 8. To prevent overfitting of the embedding network during training, the datasets are also augmented by flipping the images both vertically and horizontally, which results in a total of 3,900 image pairs for training. During the training of the Siamese network, negative pairs are randomly generated in each iteration by combining a truck image from one pair to a different truck image that has been processed identically (i.e., flipping vertically, flipping horizontally or unchanged). The total number of negative pairs is set to be equal to that of positive pairs in each iteration. The negative samples in each data set for the triplet network are generated following the same logic. Following this approach, the Siamese and triplet networks see an abundance of different negative samples than positive samples throughout the entire training process.

The CNN of the embedding network is initialized using the corresponding weights of the trained YOLOv3-tiny model. The weight of this part is frozen for the first 5 epochs and then fine-tuned in the training process along with the fully connected layers which are randomly initialized using an Xavier uniform distribution³⁸. A dropout rate of 0.5 is also added before the first fully connected layer to overcome overfitting. The two embedding networks are trained using the Adam optimizer³⁹ with a base learning rate of 5×10^{-4} and a batch size of 32. The entire process is implemented using the PyTorch framework⁴⁰ and is executed on an NVIDIA Titan Xp GPU. After the training of the embedding network, the reciprocal of the Euclidean distance (i.e., L2 distance) between the two truck images' embedding vectors is selected to be the proxy for their similarity.

4.3 Reidentification Strategy

A mutually nearest neighbor strategy is adopted in this study for the reidentification of the truck events captured at different locations along the corridor within the same data acquisition cycle. In a single cycle, multiple truck events can be captured at each corridor data collection site. Also, the sets of trucks captured at two adjacent locations may not be identical because trucks may enter or exit the corridor in between the sites. To perform truck reidentification between two locations, each possible pair of truck images are compared and measured by the similarity metrics discussed in Section 4.1 and 4.2. A correlation matrix with m rows and n columns is constructed for storing the computed pairwise indices, where m and n are

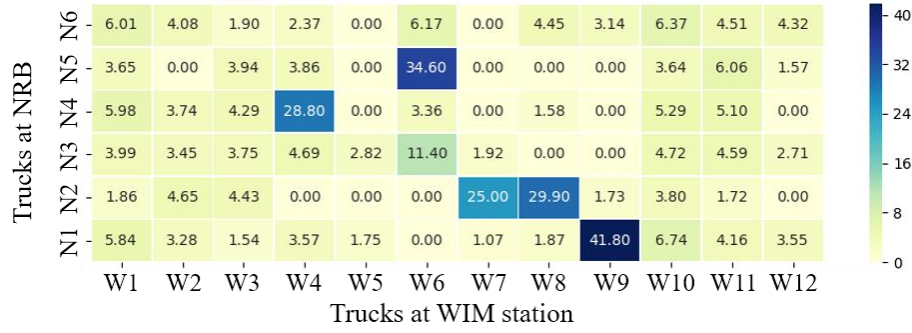


Figure 9. A sample of correlation matrix for truck reidentification by MODS between two data sets collected at different locations.

Table 4. Evaluation results of truck reidentification using MODS, embedding network trained from Siamese and triplet network

<i>Method</i>		<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>Speed per comparison (sec)</i>	<i>Device</i>
CNN-based	triplet	100%	95.00%	0.9743	0.0035	GPU
	Siamese	100%	93.57%	0.9668	0.0032	GPU
Wide baseline	MODS	98.49%	93.57%	0.9597	1.73	CPU
	ASIFT	99.22%	91.43%	0.9517	20.52	CPU
Short baseline	SIFT	93.68%	63.57%	0.7574	0.06	CPU
	SURF	90.13%	52.14%	0.6606	0.08	CPU

the number of truck events captured at the first location and the second location, respectively. A sample of the correlation matrix is plotted in Figure 9 to showcase the matching between the NRB and WIM station using MODS. Two images with index i and j are considered to contain the same truck if and only if the similarity measure with a coordinate of (i, j) is greater than a pre-determined threshold (e.g., 12) and is the maximum metric of the i -th row and the j -th column at the same time. In other words, truck i and truck j are mutually nearest neighbors. For instance, considering the matrix shown in Figure 9 (NRB trucks compared to WIM station trucks), four matched truck pairs are extracted: namely, W4 & N4, W6 & N5, W7 & N2 and W9 & N1, if the threshold is set to be 12.

4.4 Evaluation

To the evaluate the performance of the two approaches (i.e., hand-crafted features and CNN-based embedding network) for truck reidentification, 150 truck image pairs that are not included in the training set of the CNN-based embedding networks are selected to serve as a testing data set. The 150 images are split into 15 sets of 10 trucks to mimic the real application in this study, which leads to 1500 (15*10*10) similarity metric calculations. The performance of the models is evaluated in terms of precision, recall, F1 score⁴¹ and inference speed. The F1 score is the harmonic mean of precision and recall which can calculated as $(2 * recall * precision) / (recall + precision)$ ⁴¹. The results are compared in Table 4. Since the CNN embedding networks trained from the Siamese and the triplet network follow the same procedure during testing, they have inference speeds close to one another. In addition, cumulative match characteristic (CMC) curves averaged from all 15 test sets are plotted in Figure 10. According to the evaluation results, the embedding network trained from the triplet architecture gains the highest F1 score as well as the fastest speed; it is thus employed for the truck reidentification in the production version of the system. Nevertheless, all methods fail to distinguish some extremely similar truck pairs. For examples, some trucks seen at different locations along the corridor may come from the same freight transportation company or they are of the same truck type with the same color. Besides, a truck image in one set might be matched with a distinct but similar truck in another set if the same truck is not captured in the second set.

4.5 Integrated Data

With the truck reidentification framework in place, truck events (bridge responses to truck loads and truck load information measured by the WIM station) are integrated using the matched truck images. An example of the integrated data set corresponding to the same truck load is illustrated in Figure 11. While a dense network of various types of sensors are installed in the two bridges, this paper only showcases strain responses measured by one strain gauge (Hitec HBWF-35-

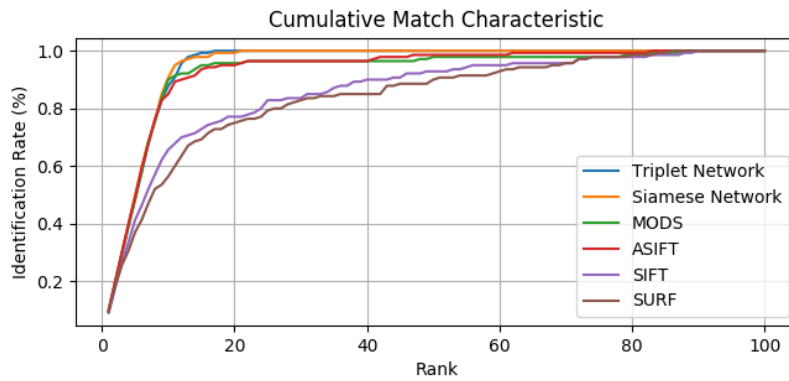


Figure 10. Cumulative match characteristic curves for truck matching using SURF, SIFT, ASIFT, MODS, and CNN-based embedding networks (Siamese and triplet networks).



Figure 11. Example of integrated bridge response data and truck load weight data

125-6-10GP-TR) in each bridge to demonstrate the proposed computer vision framework. In each bridge, the selected strain gage is installed 3 inches (7.6 cm) above the top surface of the bottom flange on the middle girder web at mid-span. The strain gauge is notated as ST for the TRB and SN for the NRB in Figure 11. Both sensors are programmed to have a sampling frequency of 100 Hz.

5. CONCLUSIONS

In this paper, a computer vision-based truck reidentification framework is proposed to automatically link truck-induced bridge response data measured by bridge SHM systems to truck load data measured by a nearby WIM station. A connected network composed of two bridge SHM systems, one WIM station and four traffic cameras is installed along the I-275 northbound corridor between Newport, Michigan and Romulus, Michigan to evaluate the framework. The first stage of the truck reidentification involves training CNN models for vehicle detection. Trained using a customized data set, four advanced CNN architectures are evaluated in this study in terms of average precision and inference speed. Being the fastest detector, YOLOv3-tiny is selected for real-time detection of incoming trucks to trigger the subsequent systems so that all systems can track the same trucks along the corridor. YOLOv3 is used to identify truck events captured by the cameras at the bridge SHM systems and the WIM station offline due to its high precision and fast execution speed. At the second

stage, features are extracted from identified truck images to match identical trucks captured at different locations following a mutual nearest neighbor fashion. Multiple hand-crafted feature extraction algorithms and customized CNN-based feature extraction models are compared for truck reidentification purpose. It is shown that the CNN-based embedding network trained using a triplet architecture outperforms its hand-crafted counterparts. Linked by the matched truck images, the corresponding data collected by different systems (i.e., bridge SHM systems and WIM station) can be fused. The paper showcases a sample of the integrated multi-source data set. Such data sets can be potentially utilized to develop highly innovative methods for bridge SHM and load rating. Future work will be focused on adopting a deeper CNN architecture to extract features from truck images and visualizing extracted features for a better interpretation of the model working mechanism.

ACKNOWLEDGEMENTS

The research is supported by a collaborative project funded by the National Science Foundation (Grant No. EECS-1446330 to Stanford University and Grant No. EECS-1446521 to the University of Michigan). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. The authors also wish to thank the Michigan Department of Transportation for access to the Telegraph Road Bridge and the Newburg Road Bridge and for offering support during installation of the wireless monitoring system.

REFERENCES

- [1] Liu, X., Liu, W., Ma, H. and Fu, H., "Large-scale vehicle re-identification in urban surveillance videos," *Multimed. Expo (ICME), 2016 IEEE Int. Conf.*, 1–6 (2016).
- [2] Chang, M.-C., Wei, Y., Song, N. and Lyu, S., "Video analytics in smart transportation for the AIC'18 challenge," *CVPR Work. AI City Chall.* (2018).
- [3] Datondji, S. R. E., Dupuis, Y., Subirats, P. and Vasseur, P., "A survey of vision-based traffic monitoring of road intersections," *IEEE Trans. Intell. Transp. Syst.* **17**(10), 2681–2698 (2016).
- [4] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M. and Ayyash, M., "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surv. tutorials* **17**(4), 2347–2376 (2015).
- [5] Seo, J., Hu, J. W. and Lee, J., "Summary review of structural health monitoring applications for highway bridges," *J. Perform. Constr. Facil.* **26**(4), 371–376 (2012).
- [6] Khan, S. M., Atamturktur, S., Chowdhury, M. and Rahman, M., "Integration of structural health monitoring and intelligent transportation systems for bridge condition assessment: current status and future direction," *IEEE Trans. Intell. Transp. Syst.* **PP**(99), 1–16 (2016).
- [7] Sohn, H., "Effects of environmental and operational variability on structural health monitoring," *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.* **365**(1851), 539–560 (2007).
- [8] O'Connor, S. M., Zhang, Y., Lynch, J. P., Ettouney, M. M. and Jansson, P. O., "Long-term performance assessment of the Telegraph Road Bridge using a permanent wireless monitoring system and automated statistical process control analytics," *Struct. Infrastruct. Eng.* **13**(5), 604–624 (2017).
- [9] Hou, R., Jeong, S., Wang, Y., Law, K. H. and Lynch, J. P., "Camera-based triggering of bridge structure health monitoring systems using a cyber-physical system framework," *Int. Work. Struct. Heal. Monit. 2017 (IWSHM 2017)* (2017).
- [10] Jeong, S., Hou, R., Lynch, J. P., Sohn, H. and Law, K. H., "A scalable cloud-based cyberinfrastructure platform for bridge monitoring," *Struct. Infrastruct. Eng.* **0**(0), 1–21 (2018).
- [11] Swartz, R. A., Jung, D., Lynch, J. P., Wang, Y., Shi, D. and Flynn, M. P., "Design of a wireless sensor for scalable distributed in-network computation in a structural health monitoring system," *Proc. 5th Int. Work. Struct. Heal. Monit.*, 12–14 (2005).
- [12] Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., "You only look once: Unified, real-time object detection," *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, 779–788 (2016).
- [13] Girshick, R., Donahue, J., Darrell, T. and Malik, J., "Rich feature hierarchies for accurate object detection and semantic segmentation," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 580–587 (2014).

- [14] Girshick, R., “Fast R-CNN,” Proc. IEEE Int. Conf. Comput. Vis. **2015 Inter**, 1440–1448 (2015).
- [15] Ren, S., He, K., Girshick, R. and Sun, J., “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” IEEE Trans. Pattern Anal. Mach. Intell. **39**(6), 1137–1149 (2017).
- [16] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y. and Berg, A. C., “SSD: Single shot multibox detector,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) **9905 LNCS**, 21–37 (2016).
- [17] Redmon, J. and Farhadi, A., “YOLO9000: better, faster, stronger,” arXiv Prepr. **1612** (2016).
- [18] Redmon, J. and Farhadi, A., “YOLOv3: an incremental improvement” (2018).
- [19] Lin, T. Y., Goyal, P., Girshick, R., He, K. and Dollár, P., “Focal loss for dense object detection,” Proc. IEEE Int. Conf. Comput. Vis. **2017–October**, 2999–3007 (2017).
- [20] Wu, S., Zhong, S. and Liu, Y., “Deep residual learning for image recognition,” *Multimed. Tools Appl.*, 1–17 (2017).
- [21] Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B. and Belongie, S., “Feature pyramid networks for object detection,” Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017 **2017–January**, 936–944 (2017).
- [22] Simonyan, K., Andrew Zisserman and Zisserman, A., “Very deep convolutional networks for large-scale image recognition,” *Iclr*, 1–14 (2015).
- [23] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Jan, C. V, Krause, J. and Ma, S., “ImageNet large scale visual recognition challenge.”
- [24] Redmon, J., “Darknet: Open source neural networks in C” (2016).
- [25] Girshick, R., Radosavovic, I., Gkioxari, G., Dollár, P. and He, K., “Detectron” (2018).
- [26] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J. and Zisserman, A., “The Pascal visual object classes challenge,” *Ijcv* **88**(2), 303–338 (2010).
- [27] Pritchett, P. and Zisserman, A., “Wide baseline stereo matching,” *Comput. Vision*, 1998. Sixth Int. Conf., 754–760 (1998).
- [28] Fischler, M. A. and Bolles, R. C., “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Commun. ACM* **24**(6), 381–395 (1981).
- [29] Mishkin, D., Matas, J. and Perdoch, M., “MODS: Fast and robust method for two-view matching,” *Comput. Vis. Image Underst.* **141**, 81–93 (2015).
- [30] Lowe, D. G., “Object recognition from local scale-invariant features,” *Comput. vision*, 1999. Proc. seventh IEEE Int. Conf. **2**, 1150–1157 (1999).
- [31] Bay, H., Tuytelaars, T. and Van Gool, L., “Surf: Speeded up robust features,” *Eur. Conf. Comput. Vis.*, 404–417 (2006).
- [32] Morel, J.-M. and Yu, G., “ASIFT: A new framework for fully affine invariant image comparison,” *SIAM J. Imaging Sci.* **2**(2), 438–469 (2009).
- [33] Zeiler, M. D. and Fergus, R., “Visualizing and understanding convolutional networks,” *Eur. Conf. Comput. Vis.*, 818–833 (2014).
- [34] Zagoruyko, S. and Komodakis, N., “Learning to compare image patches via convolutional neural networks,” *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 4353–4361 (2015).
- [35] Lin, M., Chen, Q. and Yan, S., “Network in network,” arXiv Prepr. arXiv1312.4400 (2013).
- [36] Hadsell, R., Chopra, S. and LeCun, Y., “Dimensionality reduction by learning an invariant mapping,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.* **2**, 1735–1742 (2006).
- [37] Schroff, F., Kalenichenko, D. and Philbin, J., “FaceNet: A unified embedding for face Recognition and Clustering,” *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, 815–823 (2015).
- [38] Glorot, X. and Bengio, Y., “Understanding the difficulty of training deep feedforward neural networks,” *Proc. Thirteenth. Int. Conf. Artif. Intell. Stat.*, 249–256 (2010).
- [39] Kingma, D. P. and Ba, J., “Adam: A method for stochastic optimization,” arXiv Prepr. arXiv1412.6980 (2014).
- [40] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L. and Lerer, A., “Automatic differentiation in PyTorch,” *NIPS-W* (2017).
- [41] Sasaki, Y. and others., “The truth of the F-measure,” *Teach Tutor mater* **1**(5), 1–5 (2007).