Vehicle Part Damage Analysis Platform for Autoinsurance Application

Kundjanasith Thonglek[1], Norawit Urailertprasert[2], Patchara Pattiyathanee[3], and Chantana Chantrapornchai[4]

ABSTRACT: An automatic vehicle damage detection platform can increase the market value of car insurance. In this paper, we present a damage vehicle part detection platform, called Intelligent Vehicle Accident Analysis (IVAA), which provides an artificial intelligence as a service (AIaaS). It helps automatically assess vehicle parts' damage and severity level. There are four main elements in the service system which support four stakeholders: insurance experts, data scientists, operators, and field employees. Insurance experts utilize the data labeling tool to label damaged parts of a vehicle in a given image in a training data building process. Data scientists iterate to the deep learning model building process for continuous model updates. Operators monitor the visualization system for daily statistics related to the number of accidents based on locations. Field employees use LINE Official integration to take photos of damaged vehicles at accident sites and retrieve repair estimations. We deploy the CapsNet model to localize the damage region and classify it into 5 damage levels for a vehicle part. The accuracy of the localization is 91.53 % and the accuracy of the classification is 98.47%. IVAA provides near real-time inference. The usability evaluation of the proposed platform is separated into two aspects. First, it got 4.69 of 5.0 scores in a usability test of the application module. Second, it got 4.66 of 5.0 scores in a usability test of the intelligence module.

Keywords: AI as a service, Object Detection, Car Damage Detection, Image Processing

DOI: 10.37936/ecti-cit.2021153.223151

Article history: received October 27, 2019; revised March 17, 2020; accepted January 19, 2021; available online November 11, 2021

1. INTRODUCTION

The role of auto-insurance companies is to provide services to their customers supporting the claims process. Providing fast service in the field and fast damage repair quotations are the keys success to satisfy their customers. The traditional approach may take 15 minutes to an hour of waiting for a user to get the repair quotation from the insurance experts at the company where the car must be seen before the quotation can be done. Field employees spend a lot of time to inspect the vehicle at an accident site in the traditional claim process.

The traditional claiming process begins with an appraisal where either the insurance company will send someone out to the customer car to evaluate the damage, or the customer brings the car to the company or a registered body shop. This is usually a time consuming process. With the advancement of artificial intelligence, the traditional claim processing time can be shorted while the customer satisfaction is increased. The assistance of artificial intelligence can allow the field employee to process the claim automatically and can complete the quotation in minutes. Our proposed service system can be integrated with the existing system.

[1]The author is with NARA Institute of Science and Technology, Japan., E-mail: thonglek.kundjanasith.ti7@is.naist.jp

[2]The author is with Vidyasirimedhi Institute of Science and Technology, Thailand., E-mail: norawit.u_s18@vistec.ac.th

[3,4]The authors are with Department of Computer Engineering Kasetsart University, Thailand., E-mail: patchara.p@ku.th and patchara.p@ku.th

Artificial intelligence is an advanced technology that emphasizes creation of intelligent machines that work and react like humans. The core areas of artificial intelligence are knowledge, reasoning, problem solving, perception, learning, and the ability to manipulate objects. The deep learning technique is an effective methodology to build an intelligent agent. The area is quite mature for recognition tasks.

In this paper, we provide a platform to help automate the vehicle claims process. The platform contains four elements which involve the four stakeholders: field employees, insurance experts, operators, and data scientists. Line chatbot allows the images of the car damages to be submitted through the system for damage assessment and claim filing. This enables the field employees to do less work on sites. The claim filing and images are recorded through the central database. The web application allows the operators to perform visualization and analyze the claim status and accident cases. The data labeling tool aids the insurance experts to annotate the vehicle parts and damage levels. The annotated data is saved in the database and used for training to regularly update the damage classification model. Finally, the deep learning APIs allow the pipeline of maintaining the up-to-date model and gateway to interact with the LINE chatbot. The four elements together enable an autoinsurance company to automate the claims handling process using artificial intelligence

The rest of this paper is organized as follows. Section 2 describes related research work about artificial intelligence as a service and image processing. Section 3 explains the system design of our system. Section 4 presents the implementation of each element. Section 5 evaluates the accuracy of the damage classification model and user satisfaction. Section 6 presents the conclusion and future work of this research.

2. RELATED WORK

Artificial intelligence has greatly improved the effectiveness of both manufacturing and service systems. Recent commercial systems, such as IBM Watson, have been established to provide cloud services for facilitating creating AI and machine learning applications. Watson is able to report the possible defects on the car and show the types of checking performed. The system contains three elements: Watson Visual recognition, a web server, and a mobile application[1]. The Watson Visual recognition part utilizes the recognition services from IBM clouds for damage classification. Figure 1 presents an example of using the application that analyzes the car damage from input photos.

Another example is the car damage detective software which is an open source project on Github by Neokt[2]. It analyzes the damaged vehicle parts with a convolutional neural network. The recognition model was trained based on 3 models and uses images

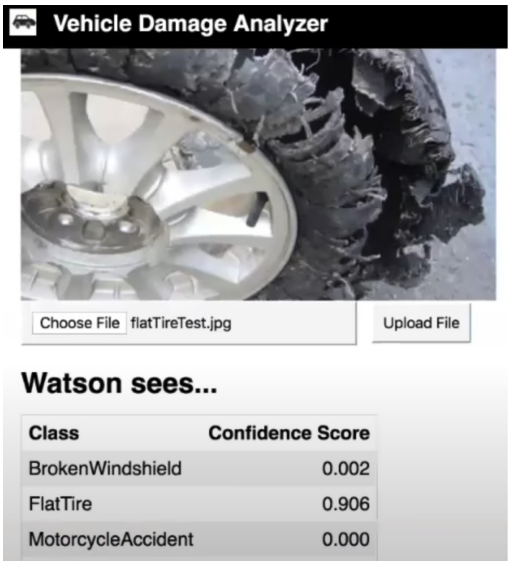


Fig.1: *Car damage analysis based on IBM Watson [1].*

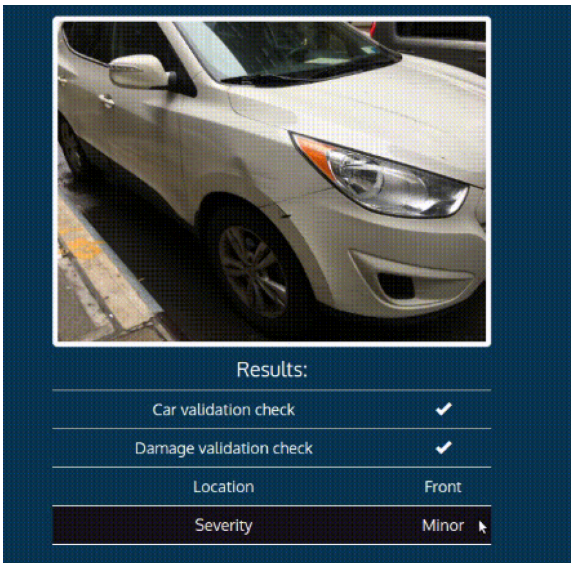


Fig.2: *Car damage detective[2].*

scraped from Google for training. Model 1 is used to classify whether the vehicle is damaged or not. Model 2 is to classify location of the damage into 3 classes (front, rear, side). Model 3 classifies the severity of the damage into 3 levels. The location accuracy and detection accuracy are around 79% and 71% respectively. The result of the application is shown in Figure 2.

As we have seen from the previous literature, the car damage analysis is based on the model which uses image processing with machine learning techniques. For image processing techniques, the template matching method is a naive approach for finding similar patterns in the image [3]. The extension has gray scale-based matching and edge-based matching routines [4]. The gray scale-based matching reduces the

computation time, and can result in running up to 400 times faster than the baseline method. Edge-based matching performs the matching only on an edge of an object [5, 6]. It returns a gray scale image, where each pixel denotes how closely the neighbourhood of that pixel matches with a template.

Using deep learning is a popular approach for object classification and detection tasks. The convolutional neural network (CNN) is the popular model for these tasks. Convolutional neural networks use translated replicas of learned feature detectors. This allows them to translate knowledge about good weight values acquired at one position in an image to other positions. To build high-quality graph embeddings, it is important to not only detect the presence of different structures around each node but also preserve their detailed properties such as position, direction, connection, etc. However, encoding these properties' information in the form of scalar values means activating elements in a vector one-by-one, which is exponentially less efficient than encoding them with distributed representations.

For the object detection task, a CNN is useful for both location detection and classification at the same time. The most commonly used network is Faster Region-based Convolutional Neural Networks (R-CNN) [7]. Faster R-CNN is actually an improved version of R-CNN (Region-based Convolutional Neural Networks) [8]. The algorithm of this network breaks down into 2 separate stages. In the first stage, regions within the image that are likely to contain our object of interest (called ROI) are identified. In the second stage, a convolutional neural network is run on each proposed region and outputs the object category score and the corresponding bounding box coordinates that contain the object. The original version of R-CNN uses selective search to generate proposed regions for the first stage, then uses a pre-trained VGG-16 to extract features from the proposed region and feeds those features into SVM to generate the final predictions.

Faster R-CNN is an improved version which runs faster and is more accurate. One of the major modifications of Faster R-CNN is to use a CNN to generate the object proposals as opposed to selective search in the prior version [9]. This layer is called the Region Proposal Network (RPN). RPN first uses a base network (e.g. VGG-16, ResNet-101, etc.) to extract features (more precisely, feature maps) from the image. It then partitions the feature maps into multiple square tiles and slides a small network across each tile successively. The small network assigns a set of object confidence scores and bounding box coordinates to each tile location. The RPN is designed to be trained end-to-end in a fully convolutional manner.

Hilton et al. proposed CapsNet which was the combination of segmentation and recognition into a single inference process using parse trees[10]. A capsule is a group of neurons whose activity vector represents the parameters of a specific type of entity such as an object or an object part [10]. CapsNet is composed of capsules rather than neurons.

A capsule is a small cluster of neurons that learns to determine the exact object in a given image and produces a vector whose length indicates the estimated probability that the object is present and whose orientation encodes the posture characteristics of the object [11]. If the object is moved slightly, the capsule will produce a vector with almost the same length but a slightly different direction. Consequently, the capsules have a similar composition.

3. METHODOLOGY

This section describes our Intelligent Vehicle Accident Analysis system platform (IVAA). We divide the section into three parts: (1) system elements, (2) system software architecture, and (3) deep learning model.

3.1 System Elements

Figure 3 displays four stakeholders of the IVAA system: insurance experts, data scientists, operators, and field employees. The platform has four tools for these four users: a data labeling tool for insurance experts, deep learning APIs for data scientists, a web monitoring application for operators, and a LINE chatbot to interact with the back-end server for field employees.

Data labeling is one of the time consuming tasks. The traditional labeling software such as LabelImg [12] and Imglab [13] works as an offline application. We integrated the data labeling tool and provided a web interface where the users can collaboratively work on the labeling task. The labeling tool returns a downloadable JSON file for the user for future use. VueJS is used as a front end framework with a Rest API server. The labeling tool is useful for adding more information to the labeled image of damage for future retraining.

Deep learning APIs are gateways which are specifically designed for data scientists and developers to train and make use of the model in applications. For training, the API returns the model identification (model ID) to the user as a link for the model deployment. For testing, the API returns with the list of damaged parts and their damage levels on the vehicle after an accident along with the confidence levels.

For the operators, the web monitoring application shows historical data that contains the number of cases, the number of processed images, and the number of days that system has operated. Graphical visualization on the system contains heat map visualization which represents the frequency of accidents by location and dates.

For field employees, we provide the LINE chatbot. The employee sends the damaged car images and the

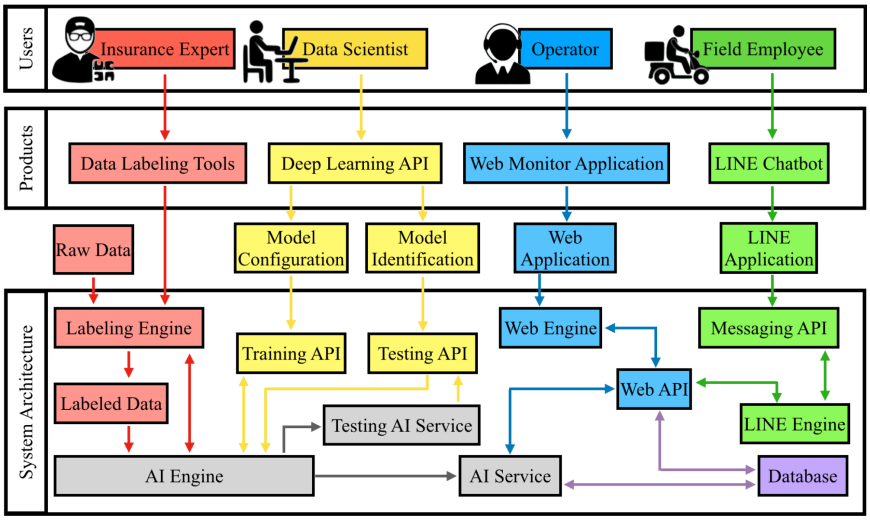


Fig.3: *System elements.*

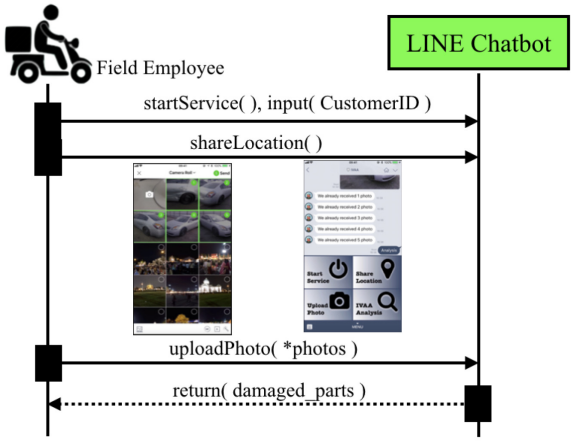


Fig.4: *LINE official integration sequence diagram.*

chatbot gives the resulting car model and price table images, along with the list of body shop locations. Note that we adopt LINE as a user interface because the requirement of the interface in the proposed system is only to receive the damaged vehicle image and return the result, and the LINE application is able to provide these features. Furthermore, we consider the installation of the application due to the fact that theLINE application can be installed on both iOS and Android.

Figure 4 shows the workflow of the field employees via the chatbot. The information details of an accident case, such as the customer ID, the accident location, and the damaged car images, are sent to the chatbot. In the backend, the deep learning testing API is executed to recognize the damaged parts and classify their damage levels from the submitted photos. The relevant information about the customer and accident are also recorded in the main database.

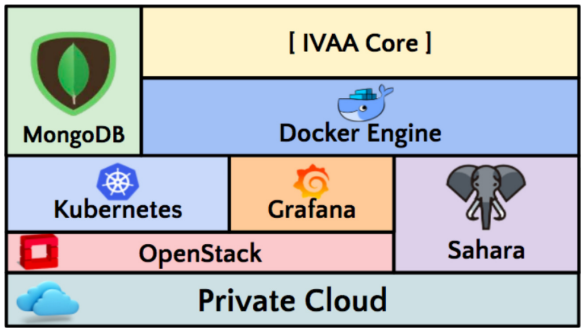


Fig.5: *Software stack.*

3.2 System Software Architecture

Our services can be deployed as a private cloud system with the hardware specification in Table 1. The private server is used for modeling, the model deployment website, and database hosting.

Table 1: *The hardware specification.*

Hardware	**Specification**
CPU	Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz
GPU	NVIDIA Tesla K40c
RAM	24 GB
HDD	4 TB
Internet connection speed	100 Mbps

Figure 5 shows the software stack of the system. OpenStack is used for computational resource management such as memory, CPU, network and other resources to provide for each of the specified tasks in the container. The visualized resource monitoring part is divided into 2 sections. The first section uses Grafana to monitor resources on the private cloud via OpenStack. In the second section, Sahara is used to monitor resources on the private cloud directly. Kubernetes provides scaling computational resources on each docker container. Docker engine is used to

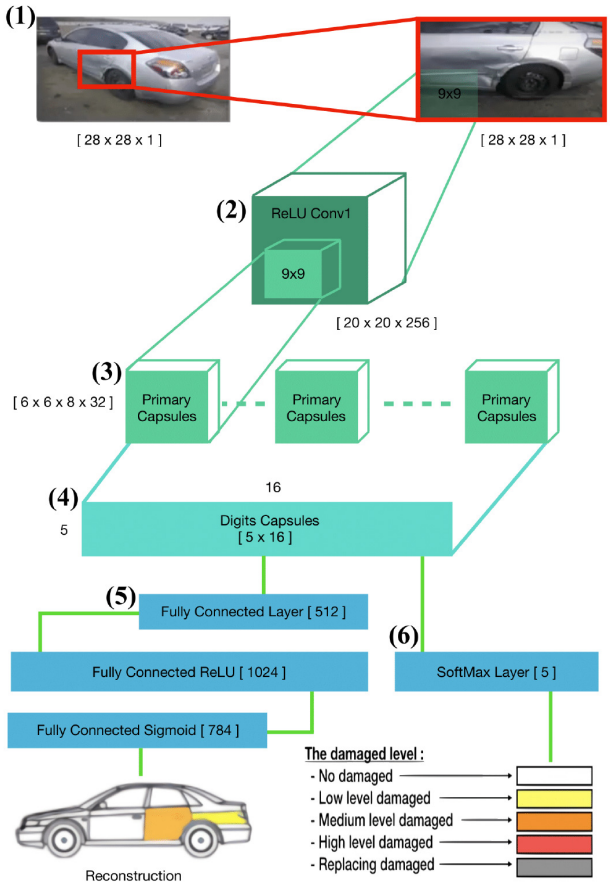


Fig.6: *The architecture of IVAA CapsNet model*

manage visualization container resources defined for task management of the IVAA Core. MongoDB is the main database for this work since it can be scaled out to support more data size.

3.3 Deep Learning Model

For the damage recognition, we utilize the capsule neural networks (CapsNet) to locate and classify levels of damage in the vehicle photos. The confidence level is returned from the deep learning API.

The process of utilizing CapsNet is shown in Figure 6. There are six main steps in the figure. The input image is resized to 28× 28. In (1), the damaged part is obtained from the input photo by cropping into the size 9 × 9 which is used as an input to the next step. The part is reconstructed by minimizing the squared difference between the reconstructed one and the input. Next, in (3), the two convolutional layers are used to reshape from 32 primary capsules to 6 × 6 × 8. In (4), these primary capsules are fed into higher layer capsules. A total of 5 capsules with 16 dimensions each and the margin loss are calculated with these higher layer capsules to predict the class probability. In (5), the model computes the location of the damaged vehicle part. In (6), softmax is used to compute the damage level of the damaged vehicle part.

We keep each trained model's replica on a single GPU. It turns out that for the memory footprint of layers with large activation, the size is disproportionate to the amount of GPU memory. By omitting the batch-normalization on top of those layers, we were able to increase the overall number of Inception blocks substantially. We hope that with better utilization of computing resources, making this trade-off will become unnecessary.

4. IMPLEMENTATION RESULT

We divide the implementation into 4 parts: (1) data labeling tool (2) deep learning APIs (3) web monitor application and (4) LINE chatbot.

4.1 Data Labeling

In Figure 7, an insurance expert uploads the damaged vehicle image to the IVAA system. He can select the part and label the photo of a damaged vehicle. The tool makes the labeling process easier than manual labeling. We also provide a tool to download the labeled and unlabeled photo data from the system to the local machine, saving it for future use.

Labeling data, or data annotation, is a preliminary step and a time consuming task in deep learning. Having an easy-to-use annotation tool can shorten the data pre-processing time. It provides the initial setup data for machine learning tasks.

The labels used for building the models may come from multiple experts and it is possible that the experts may have different subjective opinions on how some of the patient cases should be labeled. We have studied this scenario by designing a multi-expert learning framework that assumes the information on who labeled the case is available. Our framework explicitly models different sources of disagreements and lets us naturally combine labels from different human experts to obtain a consensus classification model representing the model groups of experts are converging to and also individual expert models.

4.2 Deep learning APIs

For deep learning APIs, we create the CapsNet model with 5 classes of damage severity as shown in Table 2. In Figure 8, the network is used to recognize many photos obtained from many perspectives. The parts are mapped to images below where the filled color shows the damage level for the damaged part. Levels of damage parts is based on rules from the Thai General Insurance Association (TGIA), an organization that promotes and supports the non-life insurance industry including accident insurance.

Deep learning APIs in our system are an important extension because the user can employ our system by connecting to the APIs. This allows adding a specific training dataset and retraining the deep learning

Fig.7: *Data labeling tool.*

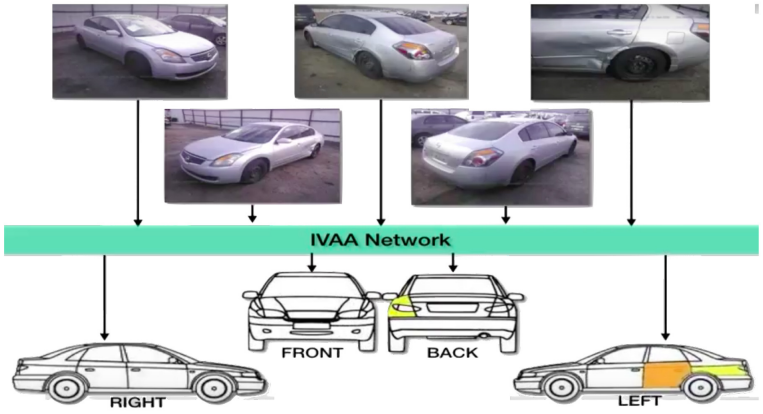


Fig.8: *IVAA network recognizing the photos.*

Table 2: *Representing the damaged levels.*

Damaged Level	Color
No damaged	White
Low level damaged	Yellow
Medium level damaged	Orange
High level damaged	Red
Replacing damaged	Gray

model effectively. Incremental retraining allows incremental improvement of model accuracy even when having only limited computing power.

4.3 Web Application

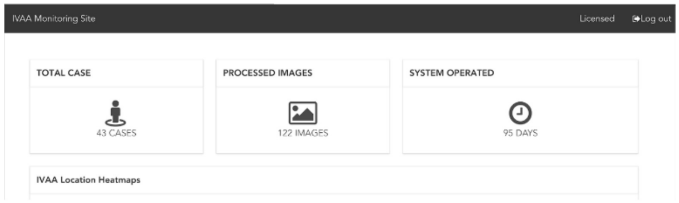
Figure 9 presents the example pages of our web application using the VueJS framework with the Bulma CSS framework. An application shows the visualization where the user can interact with the data from the system database.

The user must login through the web application first. Figure 9a shows the dashboard on our system. Dashboard is a data visualization tool that allows all users to understand and analyze what is going on. The dashboard provides an objective view of performance metrics and serves as an effective foundation for further analysis. Figure 9b is a heat map showing the volume of accident events by location. Fading color is based on the density of accident cases at the locations.

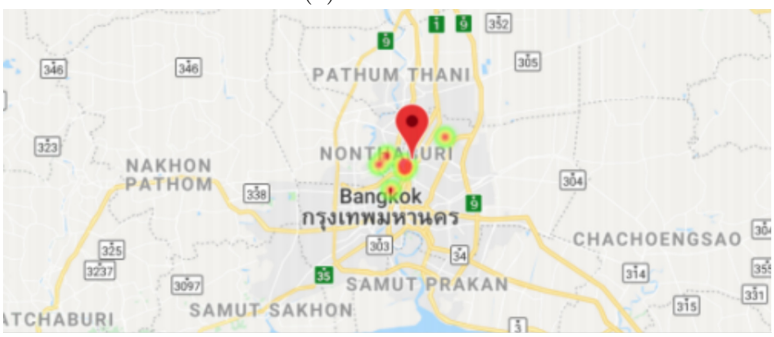
An accident case can be inserted via the case insertion page. For each accident case, the user is required to specify the case identification number, customer identification number, accident location, and to upload the photo of the damaged vehicle involved in an accident. A drag and drop approach is provided for uploading the photo with more convenience. The submitted case can be reviewed for approval and the system shows the case's information in Figure 9c. The case information page visualizes the damage level on four views of the vehicle. Moreover, it indicates the clarification of repaired price based on the damage level. Furthermore, the operator can also look up all historical accident cases. The search can be done by case ID, customer ID, and accident date. The detail button is used to show the detailed information.

4.4 LINE Chatbot

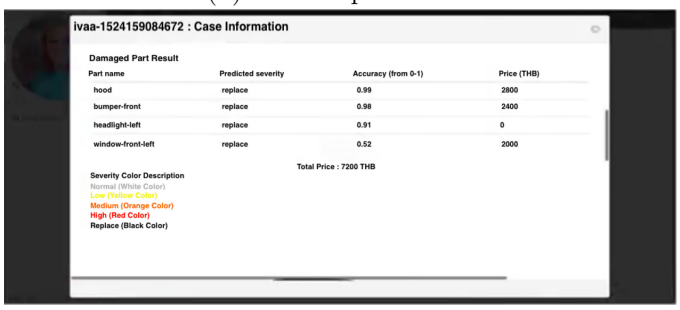
The LINE chatbot is shown in Figure 10. Using LINE messaging APIs, the image and text data can be redirected from the Line server to our database. When a field worker sends the chatbot a message, a webhook is triggered and the LINE Platform sends a request to the webhook URL. The server sends a request to the LINE platform to respond to the user. The requests are sent over HTTPS in JSON format. the LINE software developer kit has allowed



(a) dashboard.



(b) Heat map location.



(c) Case Information.

Fig.9: *Web monitoring Application.*

our IVAA website to be shared across LINE. The Line user can also post the IVAA web page onto their Line timelines to make it visible to all their friends. The LINE official account enables the field employee or customer to send the damaged car images via the LINE Official account to get the damage level and claim quotation.

There are four menus for the Line chatbot: (1) start service, (2) share location, (3) upload photo, and (4) IVAA analysis, as shown in Figure 10.

After adding IVAA as a friend, the Line user can send the "start service" at (1.1). Then, the new case is opened, and the unique case identification number is assigned to the Line user in (1.1). The number is used for follow-up of the case as well. Then, in (1.2), the system asks for customer identification number to perform authentication as (1.3) in Figure 10.

Next, the Line chatbot requires the Line user to share the place of an accident location as in (2.1) in Figure 11. The LINE user shares the location using the menu (2). Sharing an accident location (2.2) allows the company to send a field worker to the site immediately.

In menu (3), the Line user uploads the damaged vehicle's photo in the accident form shown in Figure 10. The user takes photos of the damaged vehicle, including the font-side view, the back-side view, the left-side view, and the right-side view is shown (3.1) in Figure 12. After that, the system acknowledge the number of photos that the user sent.

After the images are submitted, the menu (4) in Figure 13 is selected. Our deep learning model performs the analysis and returns the damage level of the vehicle parts using different colors as shown (4.2) in Figure 13. In addition, the chatbot can further provide a repair quotation estimate if the quotation service API is available.

5. EVALUATION

We conducted an evaluation that compared the three models: IVAA classification model, template matching, and object detection, on the selected data sets. We compared our model with two different techniques using the accuracy of detecting the damaged vehicle parts without identifying the damage levels. First, template matching is a technique in digital image processing for finding small parts of an image which matches a template image. The object detection algorithm typically uses extracted features and learning algorithms to recognize instances of an ob-

Fig.10: *LINE Chatbot Menu.*

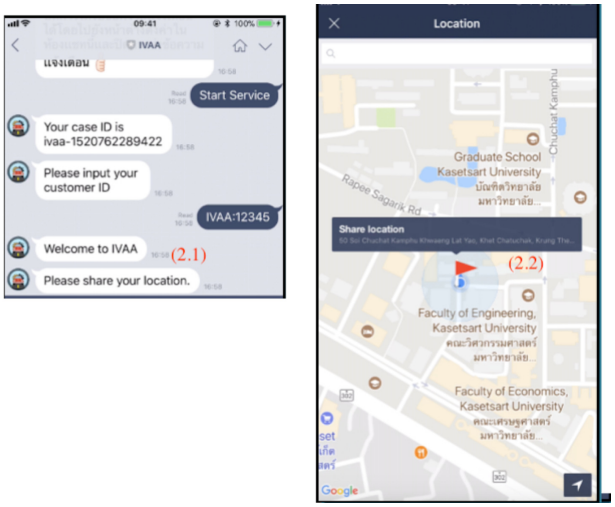


Fig.11: *LINE Chatbot (share location).*

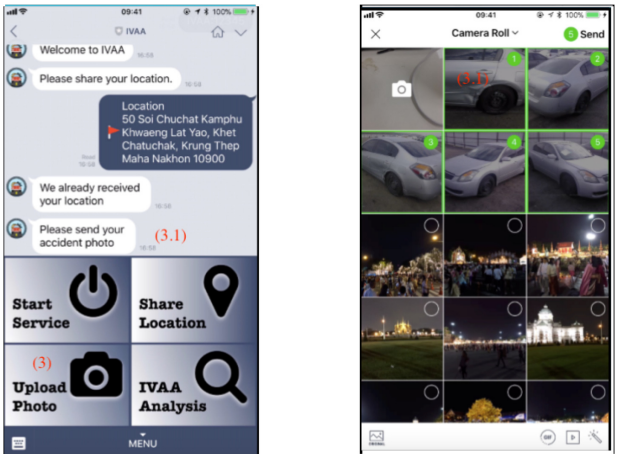


Fig.12: *LINE Chatbot (upload photo).*

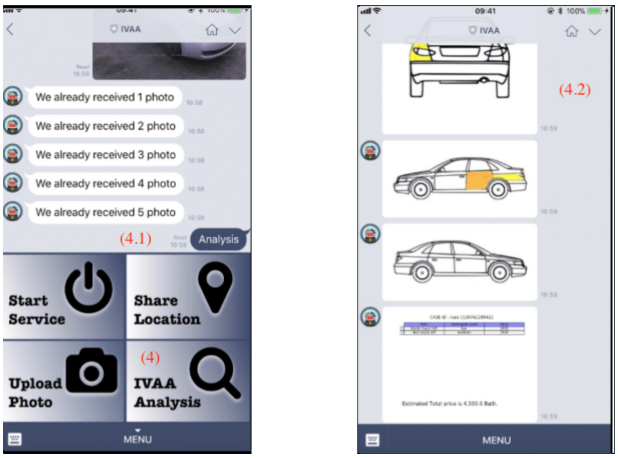


Fig.13: *LINE Chatbot (analysis).*

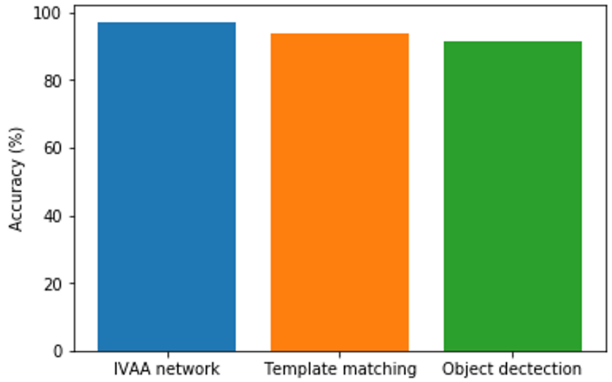


Fig.14: *Accuracy on Toyota Camry dataset.*

ject category.

We applied template matching to match the damaged vehicle part with the input image. But it is not accurate since the template matching can match only damaged vehicle parts that have been seen before. Template matching was implemented using the OpenCV library based on [1]. Secondly, we applied object detection to detect the damaged vehicle parts. The object detection library obtained from Tensorflow employs Faster R-CNN. Faster R-CNN is an object detection algorithm that is similar to R-CNN. This algorithm utilises the Region Proposal Network (RPN) that shares full-image convolutional features with the detection network in a more cost-effective manner than R-CNN and Fast R-CNN.

The accuracy of finding the car parts with various algorithms is measured using the Toyota Camry image set available on [2]. The data set is gathered from `https://carscales.com.au`, the National Highway Traffic Safety Administration, and Thai General Insurance Association. The data set contains 1,624 images. We divided 80% for training and 20% for test-

[1] https://www.geeksforgeeks.org/template-matching-using-opencv-in-python

[2] https://gitlab.com/Intelligent-Vehicle-Accident-Analysis

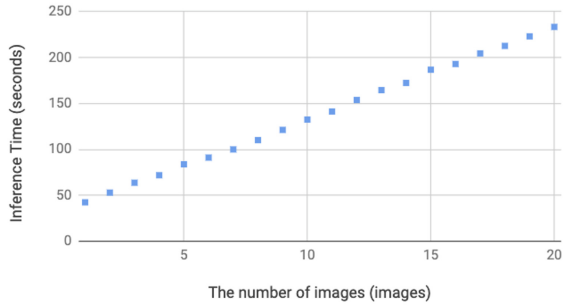


Fig.15: *The inference time.*

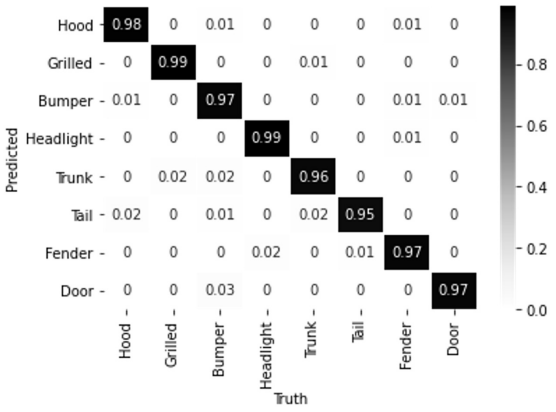


Fig.16: *The confusion matrix of IVAA network.*

ing. Moreover, we apply CapsNet to increase model accuracy by increasing input data dimensions for the damaged vehicles. Our IVAA network provided accuracy of up to 98.47% as shown in Figure 14. IVAA has greater accuracy than using the template matching approach (93.58%). The object detection approach of traditional computer vision techniques explores multiple paths where the algorithm is simplified, yet it can achieve higher accuracy but with less computational cost (91.53%). In the chatbot application, we set the threshold for bounding box detection and severe classification to 98.47%. For our case, Intersection over under (IoU) for our proposed system is 93.28%.

Figure 15 presents the inference time as the number of images grows until 20 images on our private cloud. This shows service capability when lots of inference requests come in. The average inference time per image is 12.12 seconds on our private cloud.

Figure 16 shows the confusion matrix when using the IVAA network to classify the parts of vehicles. It shows that our model can detect the damaged vehicle parts very accurately. The data set and the comparison code of the tested car are available [3]. In addition, we provide the script to train the pre-trained neural network model for new images in our project's gitlab repository.

We also evaluated our users's satisfaction with our

[3] https://gitlab.com/Intelligent-Vehicle-Accident-Analysis

Table 3: *Usability test of application module.*

Aspects	**score**
usability	4.93
reliability	4.76
security	4.56
interface	4.66
availability	4.56
Average	4.69

Table 4: *Usability test of intelligence module.*

Aspects	**score**
Prediction's speed	4.76
Prediction's accuracy	4.56
Prediction's expectation	4.60
Input data format	4.53
Output data format	4.83
Average	4.66

applications in two aspects: application usage and intelligence module. For application aspects, A questionnaire was used. It measures system satisfaction in two modules: (1) application module test in Table. 3, and (2) intelligence module test in Table. 4. For the application aspect, there are usability, reliability, security, interface, and availability [14]. Example questions include: "What is your satisfaction for the application interface?", "What is your satisfaction for the prediction speed and accuracy?", etc.

Table 3 shows the usability test results of the application module in the proposed system. The usability results show the capacity of a system to provide a condition for the users to perform the task safely. The proposed system achieves 4.93/5 for usability. For the reliability aspect, we obtain 4.76/5. For the security score, we obtain 4.56/5. For the ease of interface, we obtain 4.66/5. For availability, the score is 4.56/5. The average overall score is 4.69/5.

Table 4 shows the usability test of the intelligence module in the proposed system. The proposed system yields 4.76/5 for the satisfaction score of prediction speed. The prediction speed is defined as the time from sending the images until receiving the result. The prediction accuracy score is 4.56/5 This shows the measurement of the user satisfaction with the prediction's accuracy based on human observation. The prediction's expectation score is 4.60/5. The satisfaction scores for input data and with the output data format are 4.53/5 and 4.83/5 respectively. The average overall score is 4.66/5 for the whole intelligence module. The aspects are prediction speed, accuracy, expectation satisfiability, input format satisfiability, and output format satisfiability.

We surveyed the opinions from 30 general users. The average score for each aspect is shown. The score is out of 5. The average overall score is 4.69/5 for the application side and 4.66/5 for the intelligence module. There are 93.3% of users who highly recommend

the system to their friends or companies.

Our IVVA system provides an end-to-end platform for facilitating auto-insurance damage assessment and accident reporting. From the customer side, he/she can use the chatbot to report the accident and ask for damage assessment. From the insurance company aspect, the company can use the web application for monitoring accident cases. For the data scientists, the platform provides the tools for image labeling, APIs to retrain the model, and model deployment. The system implementation is based on Docker, which is convenient to deploy.

6. CONCLUSION

The Intelligent Vehicle Accident Analysis System (IVAA) is an artificial intelligence platform as a service. It provides an end-to-end solution for an auto-insurance company. It consists of four modules: the first module is a data labeling (for insurance experts), the second module is deep learning API for data scientists, the third module is the web monitoring application for the operators, and the fourth module is LINE official integration for field employees. Other open source solutions usually offer only damage classification feature using offline photos. The docker container is used for easy deployment. The current deep learning model used is CapsNet, which can predict correctly up to 98.47 % when testing on the proposed Toyota Camry data set. The average image inference time is 13.12 seconds. We evaluated user satisfaction for our application and intelligence modules aspects. Scores were 4.69/5 and 4.66/5 respectively.

Future work includes integrating our proposed system with the driver's application and automating the whole process of retraining when adding more images using scheduling. A database of body shops is also being collected.

References

[1] D. A. Ferrucci, "Introduction to "this is watson"," *IBM Journal of Research and Development*, vol. 56, no. 3.4, pp. 1:1-1:15, 2012.

[2] T. Neo, "Car damage detective," `https://github.com/neokt/car-damage-detective`, Dec. 2017.

[3] B. R. Meijer, "Rules and algorithms for the design of templates for template matching," in *Proceedings of International Conference on Pattern Recognition*, 1992, pp. 760-763.

[4] H. Y. Kim and S. A. de Araújo, "Grayscale template-matching invariant to rotation, scale,translation, brightness and contrast," in *Advances in Image and Video Technology*, D. Meryand L. Rueda, Eds. Berlin, Heidelberg: SpringerBerlin Heidelberg, 2007, pp. 100–113.

[5] F. Jurie and M. Dhome, "A simple and efficienttemplate matching algorithm," in *Proceedings ofInternational Conference on Computer Vision*, vol. 2, 2001, pp. 544–549 vol.2.

[6] A. Hofhauser, C. Steger and N. Navab, "Edge-based template matching and tracking for perspectively distorted planar objects," in *Advancesin Visual Computing*, Berlin, Heidelberg:Springer Berlin Heidelberg, 2008, pp. 35–44.

[7] S. Ren, K. He, R. Girshick and J. Sun, "Fasterr-cnn: Towards real-time object detection withregion proposal networks," *IEEE Transactionson Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[8] R. Girshick, J. Donahue, T. Darrell and J. Ma-lik, "Rich feature hierarchies for accurate objectdetection and semantic segmentation," in *Proceedings of International Conference on Computer Vision*, 2014, pp. 580–587.

[9] J. R. Uijlings, K. E. Sande, T. Gevers and A. W.Smeulders, "Selective search for object recogni-tion," *International Journal Computer Visxion*, vol. 104, no. 2, p. 154–171, Sep. 2013.

[10] S. Sabour, N. Frosst and G. E. Hinton,"Dynamic routing between Capsules," in *Advances in Neural Information Processing Systems*, I. Guyon, Ed.Curran Associates, Inc.,2017, pp. 3856–3866.

[11] M. D. G. Mallea, P. Meltzer, and P. J. Bentley,"Capsule neural networks for graph classificationusing explicit tensorial graph representations," *CoRR*, vol. abs/1902.08399, 2019.

[12] Tzutalin, "Labelimg," Free Software:MIT License, 2015. [Online]. Available: `https://github.com/tzutalin/labelImg`

[13] Z. A. Shaikh, U. A. Khan, M. A. Rajput and A. W. Memon, "Machine learning based number plate detection and recognition," in *Proceedings of the 5th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2016)*, pp. 327–333, 2016.

[14] I. Sommerville, *Software Engineering*, 9th ed.USA: Addison-Wesley Publishing Company,2010.

Kundjanasith Thonglek is a member of Software Design and Analysis laboratory at Nara Insitute of Science and Technology (NAIST). He graduated from Kasetsart University with a Bachelor of Engineering in Software and Knowledge Engineering. He had an internship at Data Science and Analytics Research Group, Thailand's National Electronics and Computer Technology Center. His research areas include infrastructure technology, software design and architecture, high-performance computing, and computational graphs.

Norawit Urailertprasert currently is a Ph.D. student in Information Science and Technology at Vidyasirimedhi Institute of Science and Technology (VISTEC). He graduated from Kasetsart University with a BEng. in Software and Knowledge Engineering. In 2017, he had an internship at City University of Hong Kong. His research interests include computer vision, framework for scalable machine learning, quantum computing, and high-performance computing.

Patchara Pattiyathanee is working for brands as an e-commerce consultant and implementation expert for the company. He graduated from Kasetsart University with a bachelor of engineering (major of software and knowledge engineering). He had an internship at Nagahama Institute of Bio-Science and Technology on research topic of mobile application development for e-health devices. His areas of interest are data science applications for business development, growth hacking by using data-driven techniques, and modern machine learning frameworks.

Chantana Chantrapornchai received her bachelor's degree in computer science from Thammasat University of Thailand, in 1991. Also, in 1993, she earned her master's degree from Northeastern University at Boston, College of Computer Science. She obtained her Ph.D. degree from the University of Norte Dame, Department of Computer Science and Engineering, in 1999. Currently, she is an associated professor with the Department of Computer Engineering, Faculty of Engineering, Kasetsart University, Thailand. Her research interests include parallel computing, big data processing, deep learning application, semantic web, computer architecture, and fuzzy logic. She is also affiliated with the HPCNC laboratory. She is currently a principle investigator of the GPU Education program and NVIDIA DLI Ambassador at Kasetsart University.