Reducing DNN Labelling Cost using Surprise Adequacy: An Industrial Case Study for Autonomous Driving

Jinhan Kim
KAIST
Daejeon, Republic of Korea
jinhankim@kaist.ac.kr

Jeongil Ju
Hyundai Motor Group
Seoul, Republic of Korea
littlespice@hyundai.com

Robert Feldt
Chalmers University
Gothenburg, Sweden
robert.feldt@chalmers.se

Shin Yoo
KAIST
Daejeon, Republic of Korea
shin.yoo@kaist.ac.kr

ABSTRACT

Deep Neural Networks (DNNs) are rapidly being adopted by the automotive industry, due to their impressive performance in tasks that are essential for autonomous driving. Object segmentation is one such task: its aim is to precisely locate boundaries of objects and classify the identified objects, helping autonomous cars to recognise the road environment and the traffic situation. Not only is this task safety critical, but developing a DNN based object segmentation module presents a set of challenges that are significantly different from traditional development of safety critical software. The development process in use consists of multiple iterations of data collection, labelling, training, and evaluation. Among these stages, training and evaluation are computation intensive while data collection and labelling are manual labour intensive. This paper shows how development of DNN based object segmentation can be improved by exploiting the correlation between Surprise Adequacy (SA) and model performance. The correlation allows us to predict model performance for inputs without manually labelling them. This, in turn, enables understanding of model performance, more guided data collection, and informed decisions about further training. In our industrial case study the technique allows cost savings of up to 50% with negligible evaluation inaccuracy. Furthermore, engineers can trade off cost savings versus the tolerable level of inaccuracy depending on different development phases and scenarios.

CCS CONCEPTS

• **Software and its engineering**;

KEYWORDS

Software Testing, Autonomous Driving, Deep Neural Network

1 INTRODUCTION

Machine Learning technologies such as Deep Neural Networks (DNNs) are increasingly used as components in complex software and industrial systems deployed to customers. While much research has focused on how to improve and then test the performance and robustness of these components their increased use poses a number of additional challenges to software engineers and managers. For example, while training and retraining after refinements (of data or model setup) of the DNNs are not primarily labour but compute intensive it can take considerable time and thus delay the development process. There are also considerable costs involved in collecting, ensuring the quality of, and then labelling the data to enable supervised training. A fundamental challenge is also to judge how much additional training should be done and on which data.

Individual solutions to several of these problems have been proposed. For example, so-called active learning has been proposed to reduce labelling costs, but is often relatively complex involving ensembles of networks [1] or limits the type of deep networks that can be used [5]. Here, we investigate if a simple metric for quantifying how surprising an input is to a DNN, can be used to support several of these real-world engineering challenges in an industrial setting. In particular, we focus on a key step in the processing pipeline of an autonomous, so-called self-driving, car: the segmentation of images into separate objects for later processing of dangers as well as planning of steering and actions. This is a complex, embedded, and a real-time system encompassing both hardware and software and needing to use state of the art DNN technologies. While much DNN research has focused on image recognition and classification and later on object detection, semantic segmentation of images is harder still. To ensure that a DNN-based component of the car can perform this task robustly is a major challenge in the automotive industry and, as such, can serve as a testbed for the supporting technology we propose. Not only is the task technically very challenging, it also has one of the highest labelling costs.

The approach we have evaluated with our industrial partner exploits our previously proposed Surprise Adequacy (SA) metric for estimating how surprising a new input is to a DNN [6]. This metric was introduced primarily as a test adequacy criterion, i.e. a way to select test cases and evaluate if a DNN is sufficiently robust and of high enough quality. However, the actual measure quantifies how different a new input is to the ones the network has already seen. Since those are the ones the network has been trained on, and thus where it should perform well, we can further exploit the metric to guide labelling, training and retraining scenarios. Here we evaluate this potential in a real, industrial setting and based on the key challenges identified by the practitioners in the company.

The rest of the paper is organised as follows. Section 2 introduces the DNN based semantic segmentation and further details its challenges. Section 3 briefly describes the Surprise Adequacy (SA) from our previous work, and explains how it is applied to the semantic segmentation DNN models; it also introduces a new type of Surprise Adequacy metric called Mahalanobis Distance based SA. Section 4 describes the datasets we use, and presents our research questions. Section 5 presents and discusses the evaluation results of our proposed technique. Section 6 lays out threats to validity, and Section 7 discusses the related work. Finally, Section 8 concludes with a reference to future work.

(a) Input Image (b) Output Segmentation

Figure 1: An input and output of semantic segmentation

2 SEMANTIC SEGMENTATION FOR AUTONOMOUS DRIVING

Object recognition techniques for images can be categorised based on the number of objects being recognised, as well as the precision of the location information being extracted. Image classification, which is the subject of many existing work on DNN development and testing [6, 13, 17], simply aims to put a single label on the entire input image, which often contains a single or main object. Object detection, on the other hand, accepts images that contain multiple objects, and aims to put not only labels but also bounding boxes to each object in the image [4]. Finally, semantic segmentation aims to partition the input image into meaningful parts by labelling each *pixel* in the image. Figure 1 shows a semantic segmentation example with an input on the left and its output on the right.

Semantic segmentation is a critical task for autonomous driving, as it allows the vehicle to correctly recognise the traffic scene around itself thus enabling analysis, detection of dangers, as well as being the basis for planning and action [14]. This section briefly describes the basic principles of semantic segmentation, and the challenges of developing a DNN based semantic segmentation module for autonomous driving in an industrial setting.

2.1 DNN Based Object Segmentation

A fundamental principle shared by all existing approaches is that semantic segmentation of an image consists of many instances of the pixel classification problem. In image classification, we typically learn low level features placed across the entire image using convolution, and subsequently combine these to reach a classification result [15, 16]. In semantic segmentation, we need to learn and represent multiple low level features for each pixel, resulting in more complicated architectures.

Figure 2 shows the three deepest (sets of) layers of the semantic segmentation DNN model trained by the R&D Division at Hyundai Motor Company (details irrelevant to our discussion are left out). The model can segment objects in N_c different classes, takes an input image of w_{in} by h_{in}, and returns the segmentation results in output images of size w_{out} by h_{out} (both w_{in} and w_{out} are set to 512, and h_{in} and h_{out} are set to 336). The DNN model learns the latent features used for segmentation via the use of various convolution layers [16]. With the use of convolution layers, the internal representation of the input image is significantly smaller than the input and output image: let us denote the size of the internal representation by w_0 by h_0 (which is 64 by 42 in the case of the model we study). Note that the use of the smaller internal representation is not specific to the model studied by us, and used by other existing semantic segmentation techniques [11, 19].

Figure 2a shows the latent feature layer, L_f, each *pixel* of which is associated with d latent features. The next layer, L_c, shown in fig. 2b, performs the pixel-level classification by computing the softmax scores for N_c class labels. Finally, the output layer, L_o, upsamples L_c to the output image size, w_{out} by h_{out}. Let $L_o(x, y)$ be the softmax score vector, z, of length N_c: the class label of the pixel is $argmax(z)$. The result of the segmentation can be represented as an image of size w_{out} by h_{out}, in which each pixel has the specified colour of its class label. Note that, while the layers shown in Figure 2 are specific to the industrial model we study here, a similar structure can be found in other semantic segmentation models.

2.2 Labelling Cost for Semantic Segmentation

All supervised learning requires manually labelled datasets, which is costly to build. Semantic segmentation requires particularly expensive labelling: unlike image classification, for which the act of labelling is entering the name of the object in the given image, semantic segmentation requires pixel classification via the act of colouring different areas occupied by different types of objects. Figure 3 shows an example manual labelling. The task is both laborious and time consuming.

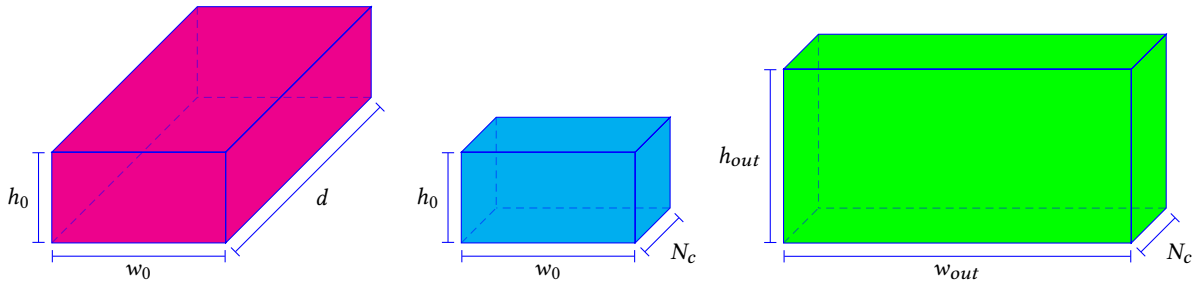
While the exact internal cost of labelling at Hyundai Motor Company will remain confidential, the scale of the problem can be conveyed by looking at the publicly available data labelling services. Semantic segmentation is the most expensive image-based labelling task provided by Google Cloud.[1] Each segment labelled by an individual worker becomes a *unit*: each month, the first 50,000 units are available at 0.87 USD per unit, and the following 950,000 units at 0.85 USD per unit.

In the dataset we study, each image typically contains five to ten segments; if we require at least three individual workers per image for robustness, each image will consume 15 to 30 units. However, note that our "images" are actually frames from video that is captured during driving. Assuming the standard 24fps (frames per second), one second of video results in 360 to 720 units, costing anywhere roughly between 300 and 600 USD. Even if we consider more attractive bulk pricing, the cost is clearly non-trivial; any savings that can be made without detrimental effects on the accuracy of the training process will be important. The primary goal of this study is to see whether SA can help to reduce this cost by acting as a surrogate measure for model performance. If there is a strong correlation between SA and the model performance, we can use SA to prioritise inputs that must be manually labelled: inputs with lower SA can either be skipped, or be given lower priority in labelling urgency, as the model is more likely to handle them correctly.

2.3 Guiding Iterative Retraining

Unlike traditional software system whose code is written by human developers, DNNs are trained from training data [20]. Consequently, if the performance of a trained DNN model is not satisfactory, it has to go through a series of retraining instead of patching. To improve the model performance, it is widely accepted that the training data

[1] Refer to https://cloud.google.com/ai-platform/data-labeling/pricing#labeling_costs



(a) The deepest latent feature layer, L_f **(b) Internal softmax layer, L_c** **(c) Output layer, L_o**

Figure 2: Dimensions of the object segmentation output layer. Each point in the w_0 by h_0 plane in fig. 2a is associated with d latent features. These are aggregated into N_c softmax scores in the softmax layer shown in fig. 2b. Finally, these scores are upsampled to the output dimension, w_{out} by h_{out} in the output layer shown in fig. 2c.

Figure 3: Example manual labelling

for retraining should be curated carefully [6, 10]. We posit that SA can improve the effectiveness of iterative retrainings, by enabling the engineers to choose inputs that are sufficiently diverse from the existing training data.

3 SURPRISE ADEQUACY

This section describes how Surprise Adequacy (SA) [6] has been applied to our semantic segmentation DNN. The basic idea of SA is that distances between the internal values of a neural network, i.e. the values calculated in each node of the network when activated by an input, carry information about how similar these inputs are.

Thus, if we simply 'run' the network on an input and save all the node activation values in a (long) vector, we can compare such vectors and thus quantify how distant inputs or groups of inputs are from each other. If the vector has a large distance to the vectors seen by the network during training, the internal computations of the network are different, and we consider the input causing the vector to be surprising. From a software engineering perspective this is natural, since the node activation values correspond to state values during the execution of a software function. Since testing on non-surprising inputs is not likely to uncover incorrect DNN behaviour, a goal of DNN testing becomes to find inputs that are adequately surprising. Hence, the name.

In SA terminology, these vectors of node activation values are called Activation Traces (ATs) and different distances between them can be used for different purposes in DNN training, testing, and engineering [6]. In general terms, we can do one-to-one SA, by comparing individual ATs to each other, one-to-many SA to quantify how distant one AT is to a set of others, or we can do many-to-many SA by comparing two groups of ATs to each other. The original SA paper [6], proposed the one-to-one distance-based SA, DSA, to explore class boundaries for DNN classification tasks and the one-to-many likelihood-based SA (LSA) as a general SA metric for both regression and classification. In the industrial setting explored here we primarily need a one-to-many distance metric and thus focus on LSA. However, LSA is computationally costly so for industrial applicability we searched for ways of making it more efficient to compute.

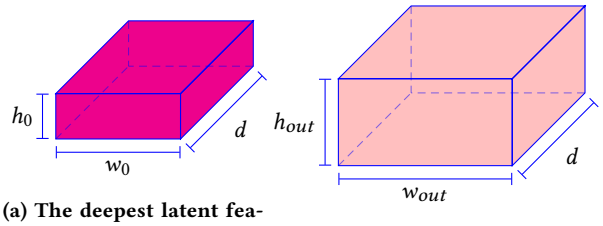
Below, we first describe how we extract Activation Traces (ATs) from the DNNs used for semantic segmentation in autonomous cars, and then present the actual computation of SA values we use in this study. In particular, we describe a new type of SA, called Mahalanobis Distance Based SA (MDSA) with preferable scaling properties.

3.1 Activation Traces for Object Segmentation

Since semantic segmentation is essentially classification of each pixel in the input image, the output is also an image with the same dimension as the input. However, as can be seen in Figure 2, the semantic segmentation DNN we studied performs classification first (when going from L_f to L_c) and upsamples the *results* of the classification to the output dimension (when going from L_c to L_o). This poses a problem for the SA analysis due to its nature. The surprise we want to measure essentially captures how similar the features of an unseen pixel are to the features of pixels in the training data. If we do not consider pixels and the corresponding features separately based on their real class labels, the resulting surprise will end up capturing how similar an unseen pixel is to all the pixels in the training data: this is likely to be extremely noisy. Consequently, to compute SA per class label, we need to map the classification features (a vector of length d) to a label (the value of a pixel in the provided label image). However, the feature vector of length d only exists for pixels in w_0 by h_0 plane, with no one to one mapping from feature vectors to pixels in the output: we have $w_{out} \times h_{out}$ labels, but only $w_0 \times h_0$ feature vectors (with $w_0 < w_{out}$, etc).

Figure 4 shows how we circumvent this problem by modifying the model architecture specifically for AT extraction. The semantic segmentation model performs classification in a smaller dimension (w_0 by h_0) and upsamples the resulting softmax scores; we, instead, upsample the features directly so that we can have one to one mapping between features and pixel labels. The same upsampling algorithm that is used when going from L_c to L_o has been applied to L_f to obtain the *instrumentation* layer, L_{AT}. To extract ATs, we simply store L_{AT} during execution. The AT vector for a pixel at (w, h) in the model output of size $w_{out} \times h_{out}$ is the feature vector of length d at (w, h) in L_{AT}.

Note that, for each pixel, the feature (AT) vector of length d that we extract can be thought of as the activation values of the deepest, final fully connected layer before the softmax layer of the single pixel classifier DNN. Also, given that the classification takes place at the pixel level, we are no longer bound by *images* as the unit of analysis. Instead, each individual pixel from a specific class label will be the unit of analysis.



(a) The deepest latent feature layer, L_f **(b) Activation Trace layer, L_{AT}**

Figure 4: Activation Trace (AT) extraction for semantic segmentation

3.2 Likelihood Based SA

Likelihood Based SA (LSA) uses Kernel Density Estimation (KDE) to summarise the ATs. Let P_c be the set of pixels that belong to class c. Using the notation from our previous work [6], let $A_{L_f}(P_c)$ be the set of individual ATs, $\alpha_{L_f}(p)$, for pixel p in P_c. Given a new input pixel, x, we first perform the KDE:

$$\hat{f}(x) = \frac{1}{|A_{L_f}(P_c)|} \sum_{x_i \in P_c} K_H \left(\alpha_{L_f}(x) - \alpha_{L_f}(x_i) \right) \quad (1)$$

Here, H denotes the bandwidth matrix, and K is a Gaussian kernel function. Based on KDE, we compute the Likelihood based Surprise Adequacy of the new input x as:

$$LSA(x) = -\log(\hat{f}(x)) \quad (2)$$

3.3 Mahalanobis Distance Based SA

While LSA was shown [6] to be an effective way to quantify how far an activation trace is from a set of other traces, its performance suffers as this set becomes larger. The problem is that the KDE requires the new activation trace to be compared to all the existing ones in the set, e.g. the summation in Equation 1 above is over ATs for all inputs in set P_c. This is unsuitable for typical, industrial use cases where the training data may contain a very large number of inputs, leading to significantly many activation traces of non-trivial lengths. In summary, *LSA*, despite being effective, can have unfavourable scaling properties for large-scale, industrial application.

For the industrial setting of this paper, we addressed this shortcoming by seeking a summary of the activation traces seen so far. A natural choice is to use the Mahalanobis distance [2]. This is a generalisation of the Euclidean distance that takes correlation in a dataset into account and can measure the distance between a point (here: an activation trace as a vector) and a distribution (here: the distribution of previously seen traces we want to compare the new point to). Its use in software engineering is rare but it was part of a method for software defect prediction [8]; in the context of DNN models, it has been successfully applied to detect out-of-distribution inputs [7]. Here, instead of having to loop over all the activation traces for points in P_c we can precalculate the mean, μ_c and covariance matrix, S_c and can then calculate the Mahalanobis Distance based Surprise Adequacy (MDSA) for an input x:

$$MDSA(x) = \sqrt{(\alpha_{L_f}(x) - \mu_c)^T S_c^{-1} (\alpha_{L_f}(x) - \mu_c)} \quad (3)$$

This measures the distance from an AT to the centre of the previous ATs while taking the amount of variation among the latter, in the direction to the new point, into account. Since the inversion of the covariance matrix can be cached for the set P_c, calculating the MDSA involves only five elementary, linear algebra operations. This leads to considerable speedups in practice.

4 EXPERIMENTAL DESIGN

This section describes our experimental design via the datasets we use as well as the research questions we study.

4.1 Datasets

Table 1 shows the four different datasets used in this paper. Initially, training and test datasets of small scale, T_r and T_t, are used to study the feasibility of SA analysis (RQ1) and to conduct the labelling cost study (RQ2). Subsequently, an additional and larger dataset from a subsequent data collection campaign has been made available. We conduct the retraining guidance study (RQ3) using the training (T_r') and test (T_t') datasets from this second batch.

Table 1: Four different datasets used in this paper

Name	Size	Classes	Description	RQs
T_r	16,549	12	The 1st training data	RQ1, 2
T_t	2,186	12	The 1st test data	
T_r'	60,532	14	The 2nd training data	RQ3
T_t'	10,000	14	The 2nd test data	

Manually generated segmentation labels have also been provided for all four datasets. There are 14 segmentation classes, which are shown in Table 2. Both T_r and T_t are initially labelled with 12 classes, while T_r' and T_t' use two additional, more fine grained class labels (vehicles are further segmented into bodies and wheels, and external road structures are segmented from void regions). While lanes and road markers are divided into separate classes for easier segmentation based on features such as different colours

and shapes, we will treat them as only two semantic class groups: class group 123 (lanes) and class group 45678 (road markers, i.e., drawings and writings on the road). The void class represents the dark areas outside the road. In T_r and T_t, all areas outside the road have been labelled as void. Consequently, the areas labelled as void lack consistent features and we expect noisy results (as no meaningful segmentation is possible based on their own features). In T_r' and T_t', if there are any visible structures outside, they have been labelled separately as general structures. Finally, class 9 has been excluded from our analysis, as T_t contained only 32 images with toll gate markers.

Table 2: Segmentation Classes

Class	Description	Class	Description
0	Void	7	Road Marker (Numbers)
1	Lanes (White)	8	Road Marker (Crosses)
2	Lanes (Blue)	9	Toll Gate Marker
3	Lanes (Yellow)	10	Vehecles
4	Road Marker (Arrows)	11	Road Area
5	Road Marker (Shapes)	12	Vehicle Wheels (T_r', T_t' only)
6	Road Marker (Characters)	13	General Structures (T_r', T_t' only)

The images in all of these datasets are frames of various segments of video, which has been recorded using fisheye view cameras mounted on the data collection vehicle [21]. We use `ffmpeg`[2] library to extract video frames into bitmap images. Video segments are not necessarily consecutive and include various driving conditions, such as different road conditions and time of day, etc.

We train state-of-the-art semantic segmentation models using datasets described above. All models are implemented using Python and PyTorch [12]. The maximum epoch is set to 300, batch size to 128, learning rate to 10^{-5}, weight decay to 0.0005: all hyperparameters have been empirically tuned by the R&D Division at Hyundai Motor Company.

4.2 Research Questions

We ask the following three research questions:

RQ1. Feasibility: the first research question is a sanity check that the AT extraction described in Section 3.1, and the SA analysis that follows, work as expected. Does the model perform worse with more surprising inputs? We answer RQ1 by plotting, and by reporting Spearman correlation between, the SA values and the model performance metrics. The Spearman correlation is a natural choice, in this case, since it does not make assumptions about the underlying distribution of data and can better handle non-linearities that are to be expected in DNNs.

To capture the model performance, we use Intersection over Union (IoU); the standard evaluation metric for semantic segmentation. IoU is the ratio between the intersection of the predicted segment (i.e., region) and the label segment and the union of the two segments. When the predicted segment is exactly the same shape as the labelled segment, IoU becomes 1.0. The less they overlap the closer to 0.0 the IoU becomes. In practice, we compute the IoU for a specific segment class by considering all pixels that belong to that class, instead of computing IoU for each independent segment. If objects from n classes are present in a single image, we get n different IoU values.

RQ2. Cost Efficiency: the second research question concerns how much labelling cost can be saved by using SA values as a guide. To study the trade off between labelling cost saving and the resulting inaccuracy in model evaluation, we simulate a scenario in which we do not label the x% of images with the lowest SA. If SA correlates well with model performance the images with the lowest SA values can be expected to add very little performance even if they were labelled. RQ2 is answered by plotting the inaccuracy from not labelling those images against x.

By not labelling images with low SA values, we explicitly accept some inaccuracy in model evaluation. We measure the model *inaccuracy* for the skipped images by reporting the complement of IoU, i.e., 1 - IoU. Additionally, we also adopt the standard accuracy metric: for this, we consider the segmentation for a specific class in an entire image to be *problematic* only if the IoU for the given class is below a pre-determined threshold. This choice was made based on the experience from the engineers at the company. Suppose we set the threshold to 0.5 and consider vehicles in the given image: we will consider a segmentation to be problematic only when the predicted vehicle segments collectively cover less than 50% of the label vehicle segments. By not labelling some images, we are effectively accepting the predicted segmentation as correct. Consequently, the inaccuracy simply becomes the ratio of unlabelled images whose IoU is below the given threshold. We use IoU threshold values from 0.5 to 0.9 with the interval of 0.1.

RQ3. Retraining Effectiveness: finally, we investigate whether SA can guide iterative retrainings. Is data augmentation based on high SA effective, or is adding more data simply sufficient? We augment T_r with three different sets of inputs sampled from T_r': 1) images with high SA vehicle segments, 2) images with low SA vehicle segments, and 3) randomly chosen images with vehicle segments. These input sets are designed to evaluate the guidance provided by SA values as well as the different number of added images. We expect the high-SA set to lead to more increased performance than the low-SA set and include the randomly selected vehicle set as a control. After (re-)training additional DNN models, each using augmented datasets, we compare their performance for segmentation of the vehicle class. It is widely believed that more diverse training data can improve the model performance [6, 10, 13] and diverse test inputs are important in software testing, in general [3]. Since SA captures the distance between the training data and new input, by definition, adding additional training data with high SA values will diversify the training data more, when compared to data with low SA values. Note that prior art on traditional software testing indicates that diversity within the whole set of added test data should probably be considered [3]; we leave more adaptive and complex retraining schemes for future work.

[2] https://ffmpeg.org

5 EVALUATION

This section describes the tooling we developed to apply SA to semantic segmentation at Hyundai Motor Company, and present the answers to RQs based on experimental results.

5.1 SA Analysis Pipeline

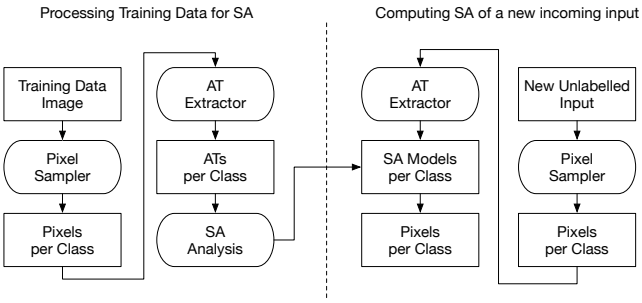


Figure 5: Overall Process of SA Analysis

Figure 5 shows the overall process of SA analysis. Given a DNN semantic segmentation model, M, and the original training data, T_r, we first sample 100 pixels that belong to each class (Pixel Sampler), out of 10,000 images randomly sampled from T_r. The sampling is necessary, as each image in T_r contains over 17K pixels: processing all pixels would incur impractical computational cost, in particular for our baseline SA technique of LSA. Once pixels are sampled, we then extract the corresponding ATs (AT Extractor), using the Activation Trace layer described in Section 3.1. Since each image contains different combinations of classes, we get 265K to 1M ATs per class. For each class, we again sample 10K ATs to build our SA models: we compute KDE for LSA, or the mean vector and the covariance matrix for MDSA.

When a new unlabelled input becomes available, we first sample 1,000 pixels per predicted class label, and extract their ATs. By running these ATs through the SA models obtained earlier, we can compute SA values *for each pixel* in the new input. While this reflects the essence of semantic segmentation (i.e., pixel classification), it is much more natural for us to think in terms of images, and not pixels. Consequently, we take the mean SA of all pixels in a specific class as the class SA of that image. The class group SA of an image is computed as the mean of z-score standardised class SA values of all classes that appear in the image. Note that this is another simplification done for immediate industrial applicability; future work should investigate if the distribution SA values per class and/or image could be leveraged for further improvements.

5.2 Visualisation

Once the SA analysis is completed for a new test input, multiple *views* become available for the image: the original input image, the result of segmentation, and an image whose pixel values are the SA values of the input image. If the manual labelling is available, we can also place the label image, and the difference between the model segmentation and the manual labelling. By combining the sets of these images in the order of original video frame, we can generate a live video visualisation of SA analysis. The side-by-side visualisation makes it easier to compare different views, and to spot any local high SA hotspots with their source locations in the original input.

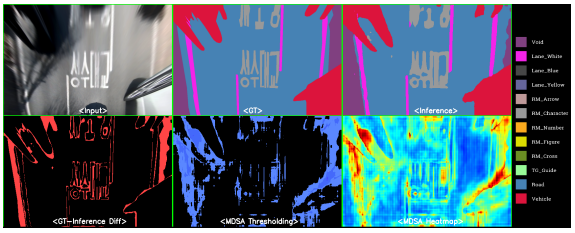


Figure 6: Example frame of video visualisation

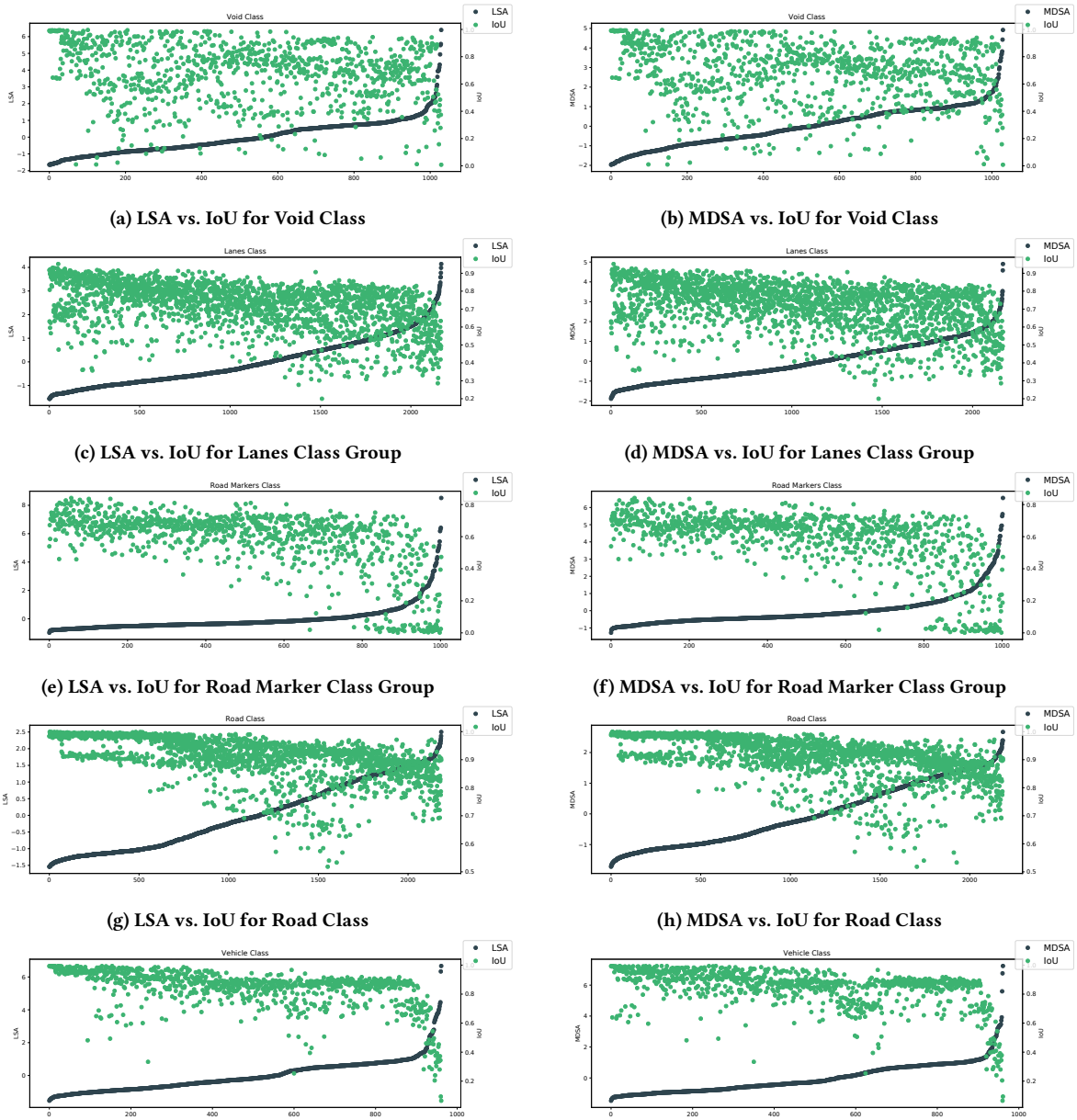
Figure 6 shows a frame from the video visualisation that contains manual labelling information. The top row shows original input, manual labelling, and inferred segmentation, from left to right; the bottom row shows the diff between manual labelling and model segmentation, pixels whose MDSA values are above a given threshold, and the MDSA heatmap. A short video clip is available from https://youtu.be/N7wKFx8pcsU.

5.3 Results

Using the analysis pipeline outlined in Section 5.1 and the visualisation described in Section 5.2, we answer the RQs below.

5.3.1 RQ1: Feasibility. Figure 7 shows the plots of IoU against both LSA and MDSA for all test images in T_t set. For each input image in T_t, one green and one black circle are plotted: the green represents the IoU for a class or class group in the image, whereas the black represents either the LSA or MDSA value for the image. Based on the definition of Surprise Adequacy, we expect IoU and SA to be negatively correlated. Basically, the higher the surprise of an input the harder we expect the segmentation to be, thus leading to lower IoU values (remember that the minimum IoU value would mean there is no overlap between segments, i.e. the works possible model performance). The effect is most clearly visible in the vehicle class (fig. 7i and fig. 7j), but also observed in all other classes and class groups, except the void class, which we expected to be noisy.

Table 3 shows the results of the Spearman rank correlation analysis between SA and IoU. The second column shows the number of images that contain the class or class group. The third and fifth columns contain ρ_{LSA} and ρ_{MDSA}, Spearman rank correlation coefficients between the class IoU of an image, and the class LSA/MDSA value of an image, respectively. The fourth and the sixth columns contain the corresponding p-values: all correlations are statistically significant. Apart from the void class, which we expected to be noisy, all other class and class groups show medium to strong negative correlation, with correlation coefficients ranging from -0.5 to -0.715. Additionally, we note that MDSA shows correlation values very similar to that of LSA, despite the low computational cost. A simple profiling results suggest that MDSA is at least two orders of magnitude faster: our LSA computation based on `scipy` [18] can, on average, process about 249 pixels per second, whereas the MDSA computation can process 54,980 pixels per second. For all class and class groups, the Spearman rank correlation coefficient



(a) LSA vs. IoU for Void Class

(b) MDSA vs. IoU for Void Class

(c) LSA vs. IoU for Lanes Class Group

(d) MDSA vs. IoU for Lanes Class Group

(e) LSA vs. IoU for Road Marker Class Group

(f) MDSA vs. IoU for Road Marker Class Group

(g) LSA vs. IoU for Road Class

(h) MDSA vs. IoU for Road Class

(i) LSA vs. IoU for Vehicle Class

(j) MDSA vs. IoU for Vehicle Class

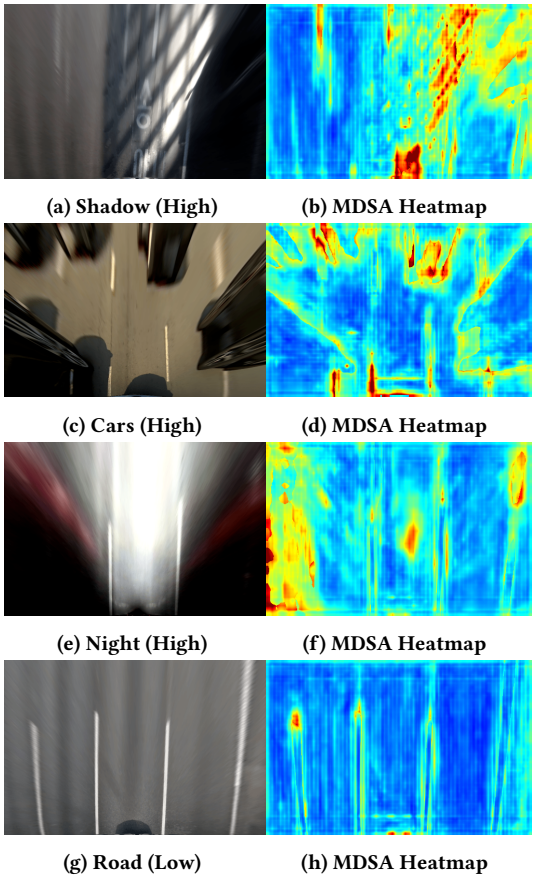
Figure 7: Plots of LSA and MDSA against IoU: each green circle represents the image-wide IoU value for the given class/class group, whereas each black circle represents the SA value of the image. We expect SA and IoU to be negatively correlated.

between MDSA and LSA values is greater than 0.98 (not shown in the table). Based on these results, we answer RQ1 that SA analysis is successfully applicable to semantic segmentation: input images with high SA values tend to result in low IoU performance. Furthermore, we also conclude that MDSA can successfully replace LSA at a much lower cost.

Figure 8 shows the representative high and low SA image examples, taken from images with the top and bottom five percent SA values in T_t. We have identified three categories of high SA images: shadows with complex patterns, existence of multiple vehicles, and strong headlight during night time. In the corresponding MDSA heatmaps, the regions with high SA tend to be where the categorical features are present. On the other hand, the low SA images mostly contain road surfaces only, without other vehicles or shadows. Given the negative correlation between SA values and segmentation performance, we expect engineers to be able to plan the subsequent data collection campaigns accordingly, based on qualitative assessments like this without manual labelling.

Table 3: Spearman Rank Correlation Between SA and IoU

Class Group	# of Img.	ρ_{LSA}	p	ρ_{MDSA}	p
Void	1058	-0.214	2.094e-12	-0.224	1.551e-13
Lanes	2169	-0.5	9.063e-138	-0.456	1.016e-111
Road Markers	1022	-0.592	7.617e-98	-0.608	1.662e-104
Road	2186	-0.715	0	-0.718	0
Vehicle	972	-0.639	8.149e-113	-0.556	7.76e-80



(a) Shadow (High) (b) MDSA Heatmap

(c) Cars (High) (d) MDSA Heatmap

(e) Night (High) (f) MDSA Heatmap

(g) Road (Low) (h) MDSA Heatmap

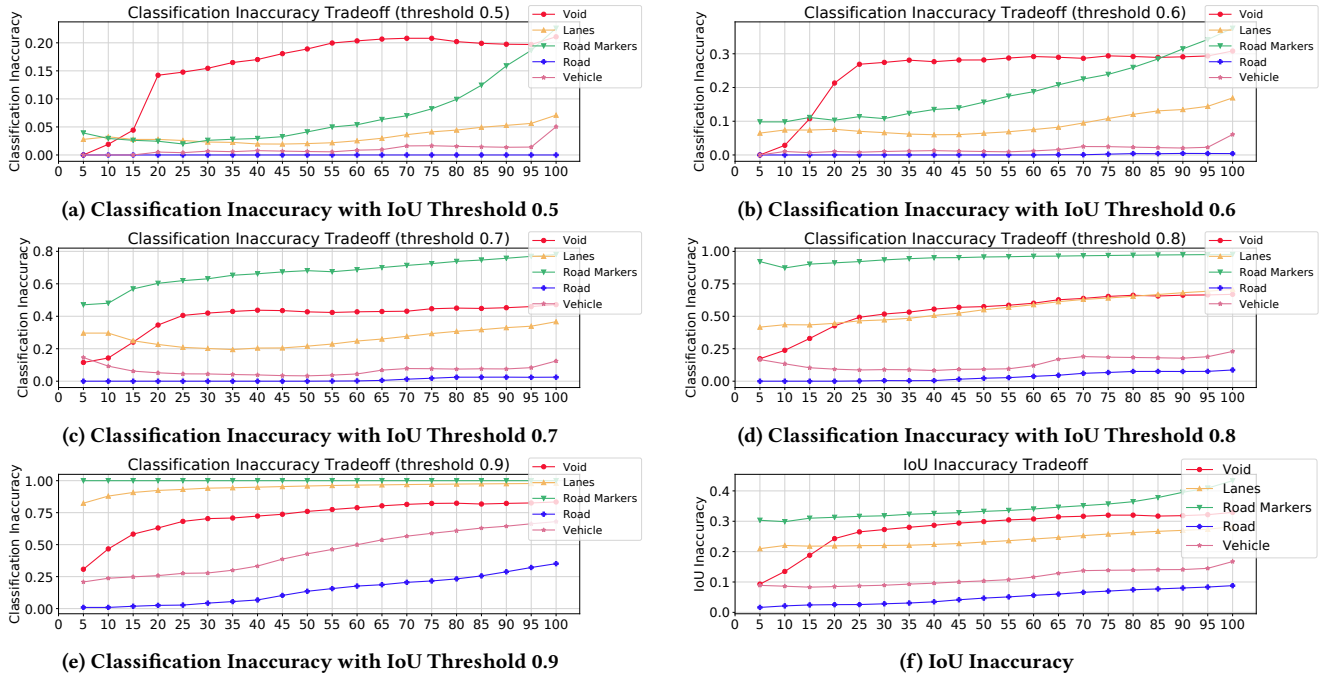
Figure 8: Input images with high (fig. 8a, fig. 8c, fig. 8e) and low MDSA (fig. 8g), paired with corresponding MDSA heatmaps

5.3.2 RQ2: Cost Effectiveness. Figure 9 shows the trade-off between missed inaccuracy and labelling cost saving offered by MDSA (chosen for its superior performance above). The x-axis shows the percentage of inputs we will skip to label, by selecting them starting from the lowest MDSA values. The y-axis shows the inaccuracy caused by cost saving. By not labelling these images, we are effectively accepting the model segmentation as ground truth (i.e., IoU = 1.0), when, in reality, they may contain errors (i.e., IoU < 1.0). As described in Section 4.2, we consider images with class IoU below threshold to be problematic, and measure the proportion of problematic input images we miss because we do not label them, resulting in the classification inaccuracy plots with various threshold values (fig. 9a to fig. 9e). For example, consider the green upside down triangle (▼) at $(x, y) = (55\%, 0.05)$ in fig. 9a. The data point suggests that, even when we forego labelling 55% of the low SA inputs, only 5% of the skipped images will be actually below the IoU threshold of 0.5 for the 'Road Markers' class. We also measure the difference between the real IoU values of these images, and the perfect IoU value of 1.0, which we assume for the sake of cost reduction, resulting in the IoU inaccuracy plot in fig. 9f. The higher the y-axis value is, the more inaccuracy we are forced to accept. Consequently, we expect the inaccuracy to grow as cost saving increases. We also expect inaccuracies to grow faster when higher IoU threshold is applied for classification inaccuracy plots. Comparing fig. 9a to fig. 9e, the plotted lines all move upwards, indicating higher levels of inaccuracies and, thus, confirming the expectation.

The plots in Figure 9 largely confirm our expectations yet show surprisingly attractive trade-off. In Figure 9f, the IoU inaccuracy incurred by not labelling 40% of the test inputs is 0.1 for the 'Vehicle' class, meaning that, on average, the images we do not label shows IoU almost 0.9. Similarly, when we consider images with IoU values below 0.5 threshold to be problematic, less than 5% of images we skip to label will be actually problematic with vehicle, lanes, and road maker class and class groups, when we save up to 50% of labelling cost, see fig. 9a. Note that, while we are studying the trade-off with the hindsight of having the results of manual labelling, the actual decision to save labelling cost can be made solely based on SA values without the labelling. The amount of saving can be guided either by a study like this, or by a pilot study about the trade-off that is unique to the DNN model being developed. Based on the observed trade-off, we answer RQ2 that SA can be successfully used to reduce the cost of manual labelling.

5.3.3 RQ3. Retraining Effectiveness. To answer RQ3, we compare the impact of adding high and low SA images to the base training data. In addition to T_r and T_t, From a large number of images from a separate and independent data collection campaign has been made available for RQ3. We divided this pool of 70,532 images into T_r' and T_t'. The set T_t' will be used to evaluate our retraining; the set T_r' has been further divided into multiple subsets as follows.

- T_r' [**B**ase] set: we randomly chose 16,750 images out of T_r' to use as the base training set for retraining.
- T_r' [SA Level, SIZE] sets: we chose images that have more than 1,000 pixels labelled as vehicle class out of $T_r' \setminus T_r'$ [Base], resulting in 24,898 images. We then compute SA for these 24,898 images, sample nine subsets based on parameters Size and SA Level, and add them to T_r' [Base]. The resulting datasets are denoted by T_r' [SA Level, Size]. Size can be either '**L**arge', '**M**edium', and '**S**mall': a 'Large' subset contains 5,025 images (30% of 16,750), a 'Medium' 3,375 images (20% of 16,750), and a 'Small' 1,675 images (10% of 16,750). SA Level can be either '**H**igh', '**L**ow', or '**R**andom': a 'High' subset is sampled from the top 30% images sorted in the descending order of SA values, a 'Low' subset from the bottom 30%, and a 'Random' subset completely randomly. Finally, we add each of these nine subsets to T_r' [**B**ase], resulting in nine datasets.



(a) Classification Inaccuracy with IoU Threshold 0.5

(b) Classification Inaccuracy with IoU Threshold 0.6

(c) Classification Inaccuracy with IoU Threshold 0.7

(d) Classification Inaccuracy with IoU Threshold 0.8

(e) Classification Inaccuracy with IoU Threshold 0.9

(f) IoU Inaccuracy

Figure 9: Plots of inaccuracies against ratio of saving: x-axis represents the ratio of input images that will not be labelled, and y-axis represents the inaccuracies in the un-labelled images. The more we save (i.e., do not manually label), the more inaccuracy we have to accept.

In total, we generate ten new training datasets, and perform 10 (re)trainings, using the same set of hyperparameters used in RQ1 and RQ2. Subsequently, we compare the average vehicle class IoU, obtained by each of the trained models, using images in T_t'.

Table 4 and Table 5 show the vehicle body and wheel class IoUs from the nine retrainings with additional data, along with the corresponding IoU from the original model trained using T_r' [**B**ase]. Row-wise maximum values are typeset in bold, while column-wise maximum values are underlined. In all cases, augmented datasets resulted in higher IoU than the result from T_r' [**B**ase]. However, the comparison between the augmented results reveals an interesting interplay between the size of the added datasets and their SA levels.

Column-wise, larger datasets tend to produce higher IoU values: this pattern is observed for high SA and random datasets in both Table 4 and Table 5. Row-wise, adding high SA images is more effective when we are adding a smaller number of images (e.g., compare T_r' [H, S] with T_r' [R, S]). However, when more images become available, the SA level has less impact: this can be seen from the comparison of T_r' [H, L] and T_r' [R, L]. While results are limited due to the practical constraint on the number of retrainings we could perform, we cautiously suggest that high SA images are more impactful on retraining when only relatively fewer new images can be added. However, with a sufficient number of new images, the diversity within the new images appears to take over. This is in line with general results on diversity-driven testing [3], where the diversity is most important early on during test selection. Thus, this calls for a new augmentation technique that considers both the diversity with respect to the training dataset (captured by SA) and the diversity within the data being augmented (currently not being captured as a metric, but set-based metrics [3] might help), as well as a larger scale empirical evaluation. We answer RQ2 that, while data augmentation based on high SA is effective for small sized augmentations, the effectiveness of data augmentation shows complex interplay between the size and the SA, calling for further investigation.

Table 4: IoU for vehicle body class after retraining with various additional training datasets: bold and underlined represent row and column maximum.

Dataset	IoU	Dataset	IoU	Dataset	IoU
T_r' [B]	0.3831	T_r' [B]	0.3831	T_r' [B]	0.3831
T_r' [H, S]	**0.4329**	T_r' [L, S]	0.4305	T_r' [R, S]	0.4246
T_r' [H, M]	0.4253	T_r' [L, M]	**0.4397**	T_r' [R, M]	0.4269
T_r' [H, L]	0.4392	T_r' [L, L]	0.4359	T_r' [R, L]	**0.4417**

Table 5: IoU for vehicle wheel class after retraining with various additional training datasets: bold and underlined represent row and column maximum.

Dataset	IoU	Dataset	IoU	Dataset	IoU
T_r' [B]	0.3640	T_r' [B]	0.3640	T_r' [B]	0.3640
T_r' [H, S]	**0.4175**	T_r' [L, S]	0.4158	T_r' [R, S]	0.4132
T_r' [H, M]	0.4125	T_r' [L, M]	**0.4242**	T_r' [R, M]	0.4160
T_r' [H, L]	0.4250	T_r' [L, L]	0.4217	T_r' [R, L]	**0.4302**

6 THREATS TO VALIDITY

Threats to internal validity include the correctness of the studied models as well as the SA analysis pipeline. To mitigate threats, all the model we study has been internally trained and validated at Hyundai Motor Company by multiple domain experts; the SA analysis pipeline follows the same approach that has been publicly replicated and reproduced [6]. Threats to external validity concern any issues that may restrict the degree to which the results generalise. While the observed results are all specific to the models and the data we studied, the SA analysis has been shown to work with data from other domains [6], and also worked as expected when applied to separate, independent dataset for RQ3. Our analysis pipeline applies aggressive sampling to reduce computational cost and, therefore, to increase the practical applicability. If more powerful computational resources are available, we expect processing more data will improve the accuracy of SA analysis. The retraining experiment for RQ3 was constrained by the available computational resources. We will conduct a larger scale empirical evaluation with novel augmentation techniques as future work. Finally, threats to construct validity concern whether we are measuring what we claim to measure. To mitigate this concern, we primarily use intersection over union, which is a well understood, standard evaluation metric for semantic segmentation task.

7 RELATED WORK

There are many existing work on testing of DNN models. DeepXplore [13] and DeepGauge [9] introduced multiple coverage criteria that measure the diversity of input sets: using a more diverse set of inputs is more likely to reveal misbehaviours of the DNN under test. DeepTest [17] focuses on improving Neuron Coverage, one of the coverage criteria proposed in DeepXplore, by applying systematic perturbations to existing input images. Surprise Adequacy [6] proposed test adequacy based on the distance between a single new input and the set of inputs in the training data, thereby enabling the comparison between individual inputs. Since labelling cost depends on the number of input images, we use SA to make the labelling decision for individual input images.

Another body of existing work on DNN testing focuses on improving the model performance. MODE [10] aims to *debug* model misbehaviour by selecting inputs that are relevant to misbehaviour for retraining. Apricot [22], on the other hand, first trains multiple DNN models using reduced datasets (called rDLMs), and directly manipulates the neural weights of the misbehaving DNN towards the average of the weights of correctly behaving rDLMs and away from the average of those of misbehaving rDLMs. Our approach in this paper is close to that of MODE, in that we go through retraining instead of manipulating the model weights directly. However, our focus is on choosing inputs to use with a focus on the cost of selecting them, which, in turn, depends on the labelling cost. As far as we know, this is the first industry case study that looks at the trade-off between manual labelling cost and accuracy of DNN evaluation/retraining. Similarly, work on active learning for reducing labelling effort [1, 5] are typically lab experiments or done on publicly available datasets rather than in industrial practice.

8 CONCLUSION AND FUTURE WORK

We propose a technique to reduce manual labelling cost during the development of a DNN based semantic segmentation module for autonomous driving in the automotive industry. Semantic segmentation has a high cost of labelling, as the act of manual labelling involves high precision pixel classification with high resolution input images. We exploit the negative correlation between Surprise Adequacy (SA) and model performance to decide for which images we can skip labelling. We also use SA to guide the selection of newly collected inputs to be added to the base training dataset, so that the model performance can be effectively improved. The proposed technique is evaluated in an industry case study involving a real world semantic segmentation DNN model and actual road data, both trained and collected by Hyundai Motor Company. The result shows that 30 to 50% of manual labelling cost can be saved with negligible impact on evaluation accuracy, and that SA can effectively guide input selection for retraining without incurring high labelling cost.

The proposed labelling cost reduction technique used a fixed threshold (i.e., top x% of low SA inputs). Future work will consider more intelligent and adaptive decision mechanism to decide whether to manually label a new incoming input or not. This mechanism can rely on SA as well as other features of the input image to realise even more effective cost reduction while minimising evaluation inaccuracy. Future work should also explore other application domains and tasks since our base technique is fully general and not specific to semantic segmentation.

ACKNOWLEDGEMENT

Jinhan Kim and Shin Yoo have been supported by Hyundai Motors, Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT (2017M3C4A7068179), Engineering Research Center Program through the National Research Foundation of Korea (NRF) funded by the Korean Government MSIT (NRF-2018R1A5A1059921), and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2018-0-00769, Neuromorphic Computing Software Platform for Artificial Intelligence Systems). Robert Feldt has been supported by the Swedish Scientific Council (No.2015-04913, Basing Software Testing on Information Theory).

REFERENCES

[1] William H Beluch, Tim Genewein, Andreas Nürnberger, and Jan M Köhler. 2018. The power of ensembles for active learning in image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 9368–9377.

[2] Roy De Maesschalck, Delphine Jouan-Rimbaud, and Désiré L Massart. 2000. The mahalanobis distance. *Chemometrics and intelligent laboratory systems* 50, 1 (2000), 1–18.

[3] Robert Feldt, Simon Poulding, David Clark, and Shin Yoo. 2016. Test Set Diameter: Quantifying the Diversity of Sets of Test Cases. In *Proceedings of the IEEE International Conference on Software Testing, Verification, and Validation (ICST 2016)*. 223–233.

[4] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Gläser, F. Timm, W. Wiesbeck, and K. Dietmayer. 2020. Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges. *IEEE Transactions on Intelligent Transportation Systems* (2020), 1–20.

[5] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2017. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1183–1192.

[6] Jinhan Kim, Robert Feldt, and Shin Yoo. 2019. Guiding Deep Learning System Testing using Surprise Adequacy. In *Proceedings of the 41th International Conference on Software Engineering (ICSE 2019)*. IEEE Press, 1039–1049. https://doi.org/10.1109/ICSE.2019.00108

[7] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. 2018. A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks. In *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.). Curran Associates, Inc., 7167–7177. http://papers.nips.cc/paper/7947-a-simple-unified-framework-for-detecting-out-of-distribution-samples-and-adversarial-attacks.pdf

[8] Dimitris Liparas, Lefteris Angelis, and Robert Feldt. 2012. Applying the Mahalanobis-Taguchi strategy for software defect diagnosis. *Automated Software Engineering* 19, 2 (2012), 141–165.

[9] Lei Ma, Felix Juefei-Xu, Jiyuan Sun, Chunyang Chen, Ting Su, Fuyuan Zhang, Minhui Xue, Bo Li, Li Li, Yang Liu, Jianjun Zhao, and Yadong Wang. 2018. DeepGauge: Comprehensive and Multi-Granularity Testing Criteria for Gauging the Robustness of Deep Learning Systems. *CoRR* abs/1803.07519 (2018). arXiv:1803.07519 http://arxiv.org/abs/1803.07519

[10] Shiqing Ma, Yingqi Liu, Wen-Chuan Lee, Xiangyu Zhang, and Ananth Grama. 2018. MODE: Automated Neural Network Model Debugging via State Differential Analysis and Input Selection. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Lake Buena Vista, FL, USA) *(ESEC/FSE 2018)*. ACM, New York, NY, USA, 175–186. https://doi.org/10.1145/3236024.3236082

[11] Sachin Mehta, Mohammad Rastegari, Anat Caspi, Linda Shapiro, and Hannaneh Hajishirzi. 2018. ESPNet: Efficient Spatial Pyramid of Dilated Convolutions for Semantic Segmentation. arXiv:1803.06815 [cs.CV]

[12] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035.

[13] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. DeepXplore: Automated Whitebox Testing of Deep Learning Systems. In *Proceedings of the 26th Symposium on Operating Systems Principles* (Shanghai, China) *(SOSP '17)*. ACM, New York, NY, USA, 1–18. https://doi.org/10.1145/3132747.3132785

[14] Mennatullah Siam, Mostafa Gamal, Moemen Abdel-Razek, Senthil Yogamani, Martin Jagersand, and Hong Zhang. 2018. A comparative study of real-time semantic segmentation for autonomous driving. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 587–597.

[15] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR* abs/1409.1556 (2014). http://arxiv.org/abs/1409.1556

[16] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.

[17] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. 2018. DeepTest: Automated Testing of Deep-neural-network-driven Autonomous Cars. In *Proceedings of the 40th International Conference on Software Engineering* (Gothenburg, Sweden) *(ICSE '18)*. ACM, New York, NY, USA, 303–314. https://doi.org/10.1145/3180155.3180220

[18] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17 (2020), 261–272. https://doi.org/10.1038/s41592-019-0686-2

[19] Tianyi Wu, Sheng Tang, Rui Zhang, and Yongdong Zhang. 2018. CGNet: A Light-weight Context Guided Network for Semantic Segmentation. arXiv:1811.08201 [cs.CV]

[20] Shin Yoo. 2019. SBST in the Age of Machine Learning Systems: Challenges Ahead. In *Proceedings of the 12th International Workshop on Search-Based Software Testing* (Montreal, Quebec, Canada) *(SBST '19)*. IEEE Press, Piscataway, NJ, USA, 2–2. https://doi.org/10.1109/SBST.2019.000-2

[21] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. 2019. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. *CoRR* abs/1906.05113 (2019). arXiv:1906.05113 http://arxiv.org/abs/1906.05113

[22] H. Zhang and W. K. Chan. 2019. Apricot: A Weight-Adaptation Approach to Fixing Deep Learning Models. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 376–387.