Real-time Activation Pattern Monitoring and Uncertainty Characterisation in Image Classification

Shenglin Wang[1], Peng Wang[2], Lyudmila Mihaylova[1] and Matthew Hill[3]

[1]Department of Automatic Control and Systems Engineering, University of Sheffield, S1 3JD, UK
Email: {swang119,l.s.mihaylova}@sheffield.ac.uk
[2]Manchester Metropolitan University, Manchester, M15 6BH, UK, Email: p.wang@mmu.ac.uk
[3]Defence Science and Technology Laboratory (Dstl), Salisbury, SP4 0JQ, UK, Email: mhill2@dstl.gov.uk

Abstract—**Deep neural networks (DNNs) have become very popular recently and have proven their potential especially for image classification. However, their performance depends significantly on the network structure and data quality. This paper investigates the performance of DNNs and especially of faster region based convolutional neural networks (R-CNNs), called faster R-CNN when the network testing data differ significantly from the training data. This paper proposes a framework for monitoring the neuron patterns within a faster R-CNN by representing distributions of neuron activation patterns and by calculating corresponding distances between them, with the Kullback-Leibler divergence. The patterns of the activation states of 'neurons' within the network can therefore be observed if the faster R-CNN is 'outside the comfort zone', mostly when it works with noisy data and data that are significantly different from those used in the training stage. The validation is performed on publicly available datasets: MNIST [1] and PASCAL [2] and demonstrates that the proposed framework can be used for real-time monitoring of supervised classifiers.**

Index Terms—**Activation Pattern, Run-time Monitoring, Faster R-CNN, Reliability, Decision Making, Uncertainty Quantification**

I. Introduction

Deep Neural Networks (DNNs) have attracted increasing attention [3–5] both in academia and industry during the past decades. They have been intensively investigated in the fields such as robotics [6, 7], autonomous driving [8] and manufacturing [9] which require high levels of safety due to involvements of human. Especially the performance of deep learning methods for image classification under uncertainties has been investigated. The [10, 11] summarise the recent state-of-the-art and how different uncertainties impact DNN methods. The quantification of uncertainties can be performed by propagating a tensor normal distribution as a prior of a Convolutional Neural Network (CNN) [12]. The mean and variance of the Gaussian distribution are propagated within a CNN framework called PremiUm-CNN developed in [12]. The variance is especially informative and a small variance means accurate classification results. Another approach uses the Hamming distance [13] which characterises well the difference and similarity between binary strings.

Although there is a number of approaches that are able to quantify uncertainties in CNNs, such as [14–17], there is a necessity of expanding these studies with different types of uncertainties - in the data: gradual and abrupt, due to environmental changes, including lighting and meteorological conditions, camera motion and other factors. Other effects can be intentionally introduced, such as adversarial attacks and are aimed to cause CNNs to make mistakes. It is important to identify when a trained CNN model performs inference correctly in order to provide a trustworthy result [18–20]. Ideally, CNN models have highly reliable performance with those inputs that have features similar to their training data sets. However, calculating similarity between inputs and training sets directly is of high computational complexity due to the reason that samples may have very high dimension.

This paper develops a faster R-CNN supervised classification framework able to quantify the impact of data on the performance of the classifier. The network has testing data that are not the same as the training data, hence the network is put outside its 'comfort zone', i.e. a wrong decision could be made by the network. This may results in potential hazards to human especially in those scenarios with human involvements. Hence, inspired by ideas from [21], this work presents an improved runtime activation pattern monitoring algorithm for monitoring the R-CNN features representations for different image inputs. The patterns of the 'neurons' inside the faster R-CNN are monitored with different data. The approach uses the Hamming distance to characterise the difference and similarity between binary strings and the this is combined with the Kullback-Leibler divergence.

The main contributions of this work consist in the following: 1) distributions of the neuron activation patterns are calculated using the Hamming distance between the current activation pattern and the central activation pattern. Next, the closeness of these distributions is characterised based on Kullback-Leibler divergence; 2) Monitoring zones are constructed based on decision making, by taking the patterns with the corresponding probability values and the changes in the patterns are visualised; 3) The efficiency of the monitoring framework is demonstrated over MNIST and PASCAL datasets.

The remainder of this paper is organised as follows. Section II provides the related work. The methodology proposed is elaborated in Section III. Section IV provides the experimental results and analysis, and the paper is concluded in Section V.

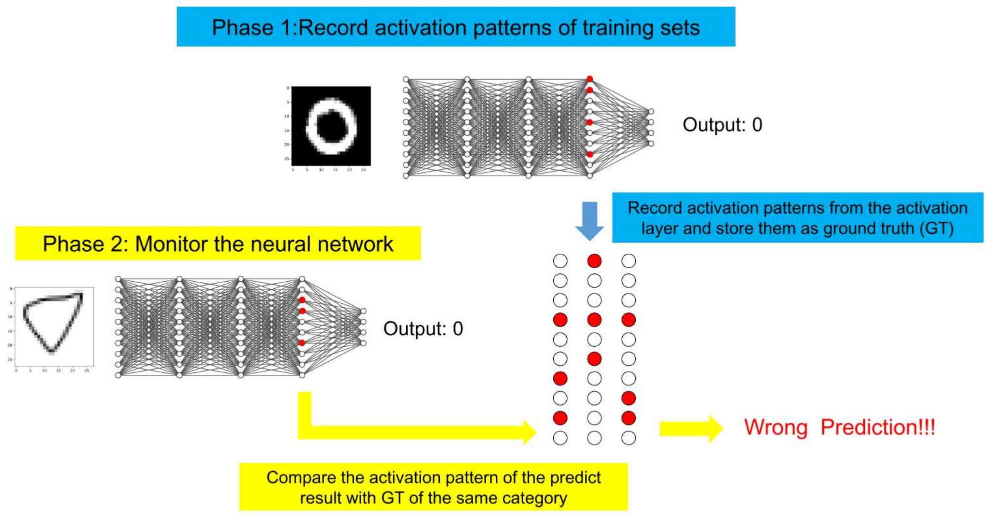

Phase 1:Record activation patterns of training sets
Output: 0
Record activation patterns from the activation layer and store them as ground truth (GT)
Phase 2: Monitor the neural network
Output: 0
Wrong Prediction!!!
Compare the activation pattern of the predict result with GT of the same category

Fig. 1. Overview of real-time activation pattern monitoring

II. Related Work

Uncertainty in DNNs is of significant importance when it is applied to safety critical tasks. DNNs are unexpectedly sensitive to small changes in the input data, including adversarial attacks, unseen objects and occlusions. These changes could lead to failures of correct and reliable decision making.

Many approaches have been developed such as Bayesian approaches [22–24] which address decision making by finding the most likely in terms of probability. Uncertainty is quantified by generating the ensemble or performing dropout in operation. Such methods overcome the difficulties of executing Bayesian inference directly but still are of high computational cost resulting in poor real-time performance. Consequently, it is unrealistic to apply Bayesian approaches in some scenarios where fast response is required, e.g. autonomous driving and real-time tracking.

Other works focus on verification problems of neural networks [25]. Runtime verification algorithms monitor the violation of correctness properties. A series of methods about monitoring activation patterns of neural networks have been proposed [19–21, 26]. Cheng proposed a boolean abstraction method for monitoring DNNs where they only considered about activation patterns of the final layers with respect to ReLU activation function. They achieved an efficient monitor based on operations on boolean logic with a binary decision diagram (BDD) which is of low computation in the MNIST dataset. The key of runtime monitoring activation patterns is that Cheng created the γ-comfort zone to collect enough activation patterns with correct predictions as the ground-truth. Nevertheless, when the number of patterns and monitored neurons as well as the abstraction parameter γ increase, it raises the challenge in storing patterns and monitoring computational costs especially in object detection. In this work, we focus on dealing with monitoring in the view of distribution of patterns. Different from Cheng's abstracting patterns by Hamming distance, we apply the Hamming distance directly as a distance metric to avoid the storage and reduce the computational complexity of large ground-truth activation patterns.

III. Methodology

A. Activation Pattern Representation

A DNN model is defined as $\mathbf{y} = \mathbf{F}(\boldsymbol{\theta}, \mathbf{x})$ with parameter $\boldsymbol{\theta}$ where y is the output of the model and $\mathbf{x}$ is the input of the model. The model can classify $\{c_1, \ldots, c_l\} \in C$ classes and consequently, $\mathbf{y} \in C$.

The common activation function applied in the activation layers of DNN models in this paper is the ReLU function that is in the form of

$$\sigma(a) = \max\{a, 0\}. \tag{1}$$

In this paper, we consider a neuron in the activation layer is *activated* when its output is greater than zero. Let's denote the output of the last activation layer in a DNN model as $\{v_1, \ldots, v_d\}$, with d the dimension of the last activation layer. The architecture of the proposed activation pattern monitoring approach is shown on Figure 1. We can then define the binary activation pattern as follows:

Definition 1 (Binary Activation Pattern): Given the output of the last activation layer $\{v_1, \ldots, v_d\}$, the activation pattern of a certain class c is defined as

$$P^c = \big(p(v_1), \ldots, p(v_d)\big), \tag{2}$$

where $p(\cdot)$ defined in (3) is a function that maps a real number $v \in \mathbb{R}$ into binary numbers.

$$p(v) = \begin{cases} 1 & v > 0, \\ 0 & \text{otherwise}. \end{cases} \tag{3}$$

For datasets such as MNIST and PASCAL, there are more than one class to be detected and classified. For clarity, let $\mathcal{T}$ denote the training dataset, and $\mathcal{T}_c \subseteq \mathcal{T}$ denote images in the training dataset contain objects with label c. The activation patterns of $\mathcal{T}_c$ can then be organised as

$$\mathcal{P}_c = \Big\{P_0^c, \dots, P_i^c, \cdots, P_n^c\Big\}, \tag{4}$$

where n indicates the number of patterns of class c. Activation patterns of the whole training dataset can be defined similarly and denoted as $\mathcal{P}$, with $\mathcal{P}_c \subseteq \mathcal{P}$ stands. Figure 2 show examples of different activation patterns from different layers of Number 1 in MNIST dataset [1].

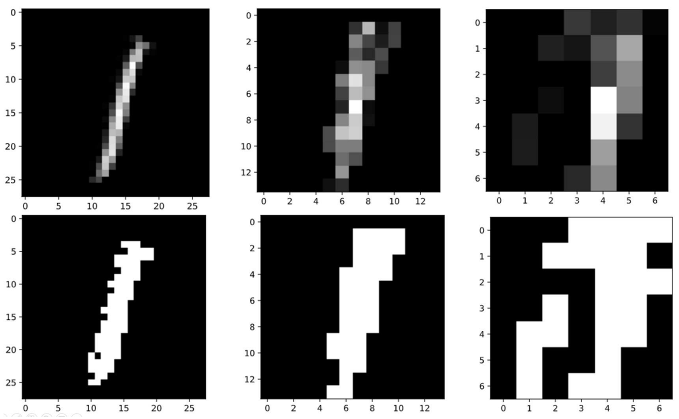


Fig. 2. Visualisation of activation layers (first row) and the corresponding activation patterns (second row) of Number 1 in MNIST dataset [1]. The activation pattern becomes more abstract from left to right as the layer in DNNs gets deeper.

B. Central Activation Patterns

Given a class c, we can expect similar activation patterns for objects contained in various input images. This enables us to accumulate activation patterns of class c during the training process and hence find a central activation pattern that can represent class c for further applications.

We define the *central activation pattern* $\widetilde{P}^c$ of class c as

$$\widetilde{P}^c \triangleq \arg\max_{P} \textstyle\sum_{i=0}^{n} \mathbf{H}\left(P, P_i^c\right), P_i^c \in \mathcal{P}_c, \tag{5}$$

where $\mathbf{H}(\cdot,\cdot)$ indicates the Hamming distance between two binary patterns P and P_i^c.

In this paper, the Dynamic Programming (DP) algorithm [27] is exploited to solve (5). Following we summarise how DP is applied in our case.

Here we denote the minimum sum of Hamming distances:

$$\tau^c[j] = \min \sum \mathbf{H}\left(P[:j], P_i^c[:j]\right), \tag{6}$$

where $[:j]$ represents neurons from first to j-th in activation patterns and there are d neurons in total. Since the neurons in the activation patterns are independent on each other, the iterative update rule of $\tau^c[j]$ is,

$$\tau^c[j] = \tau^c[j-1] + \min \sum \mathbf{H}\left(q_j^c, p_i^c(v_j)\right), \tag{7}$$

where $q_j^c \in \{0,1\}$ is the j-th neuron and $p_i(v_j)$ is j-th neuron of activation pattern P_i^c. By inferring q_j^c from 1 to d where d is also the dimension of activation patterns, that is

$$\widetilde{P}^c[j] \triangleq \arg\min_{q_j^c} \sum \mathbf{H}\left(q_j^c, p_i^c(v_j)\right), \tag{8}$$

we finally obtain the central activation pattern of class c: $\widetilde{P}^c = (q_1^c, \dots, q_d^c)$.

C. Activation Pattern Distance Distribution

So far we have the central activation pattern $\widetilde{P}^c$ and a set of other activation patters. They share the same class. First we calculate the Hamming distance between the central activation pattern and every other activation patterns from the considered set. Then we will have a set of results for the Hamming distance. Then we use the Hamming distance to calculate sub-intervals.

While the central activation pattern $\widetilde{P}^c$ is representative for a certain class c, the extraction of activation patterns and comparison with $\widetilde{P}^c$ remains a challenge for real time activation pattern monitoring. The situation gets worse when the dimension of the activation pattern increases. To cope with the challenge, this paper further propose the activation pattern distribution, which aims at distinguishing difference classes efficiently.

Given $\mathcal{P}_c$ and $\widetilde{P}^c$, the Hamming distances between $P_i^c \in \mathcal{P}_c$ and $\widetilde{P}^c$, with $i = 0, \cdots, n$ are first calculated, which are denoted as $\mathcal{D}_c = \{D_0^c, \cdots, D_i^c, \cdots, D_n^c\}$. We then partition the interval $[\min(\mathcal{D}_c), \max(\mathcal{D}_c)]$ into m sub-intervals evenly and calculate the number of distances falling into each sub-interval. Let's denote the results as $N^c = \{N_0^c, \cdots, N_j^c, \cdots, N_m^c\}$, then the activation pattern distribution α is defined as follows

$$\alpha = N^c/n = \Big\{N_0^c/n, \cdots, N_j^c/n, \cdots, N_m^c/n\Big\}. \tag{9}$$

With (9), we can calculate the distribution of each class. To distinguish different classes, we employ the Kullback-Leibler (KL) divergence as a metric, which is defined as

$$KL(\alpha||\beta) = \sum_j \alpha(j) \log\Big(\frac{\alpha(j)}{\beta(j)}\Big), \tag{10}$$

where β represents a distribution where the classes between the central activation pattern $\widetilde{P}^c$ and the activation pattern set $\mathcal{P}_{c^\star}$ are different between c and $c^\star$. In α the $\mathcal{P}_c$ and $\widetilde{P}^c$ share the same class c.

D. Choice of Thresholds and Monitoring Zones

In this paper, we are interested in two types of distance distributions of the patterns of neurons compared with central activation patterns. The first type is denoted as 'Same', which indicates that the activation patterns and the central patterns are from the same object class. On the contrary, 'Different' is used to indicate that the activation patterns are from objects of different class compared to that of the central patterns. To distinguish the two pattern distance distributions, we define two thresholds S_0 and $S_{0.05}$ to build three monitoring zones: (0, S_0), (S_0, $S_{0.05}$) and ($S_{0.05}$, $+\infty$).

The threshold S_0 characterises the shortest distance between the central activation pattern and activation patterns with different object class from the central activation patterns, i.e., the very first recorded distance from 0 in 'Different' seen in Figure 3 and Figure 4.

The $S_{0.05}$ threshold represents the distance where the accumulative probability from 0 of 'Different' distribution is 5%, i.e., $\int_0^{+\infty} Dist_{`Different'}(x')dx' = 0.05$, and $S_{0.05} = x'$.

Therefore, the interval (0, S_0) can be defined as a comfort zone which means the predicted result is trusted. When distances between prediction activation patterns and central activation patterns are in (S_0, $S_{0.05}$), it will be considered as a 'warning signal' which requires extra attention (manual) to aid decision-making of neural networks. As for distances in ($S_{0.05}$, $+\infty$), the predictions are taken as 'not trust-able'.

E. The Activation Pattern Monitoring Algorithm

In the proposed monitoring algorithm, activation states of neurons from the close-to-output layer of the DNN model is monitored. To accomplish this, a two-phase algorithm is implemented as depicted in Fig. 1. The details are given in **Algorithm 1**. In Phase 1, the pre-trained model is fed with the training dataset again and the activation patterns of training samples will be recorded and stored as the ground-truth (GT). After Phase 1, when a new input comes to the model, the activation pattern and prediction of the model will be the output. In Phase 2, the activation pattern is compared with the GT with the same label to find out their differences by calculating their Hamming distances. If the distance is larger than a threshold, the prediction is defined as a problematic decision that is unacceptable.

Algorithm 1 Real-time Activation Pattern Monitoring

```
Phase 1: Record activation patterns of training set $\mathcal{T}$
for $\mathcal{T}_c \subseteq \mathcal{T}$ do
    for $\mathbf{x} \in \mathcal{T}_c$ do
        $y \leftarrow \mathbf{F}(\mathbf{x})$ and $P^c$ is the activation pattern of $\mathbf{x}$
        if $y = c$ then
            $\mathcal{P}_c \leftarrow \mathcal{P}_c \cup P^c$
        end if
    end for
end for
/⋆ Generate central patterns of different classes ⋆/
Phase 2: Monitor the Neural Network
$y' \leftarrow \mathbf{F}(\mathbf{x}')$ and $P'$ is the activation pattern of $\mathbf{x}'$
for $c \in C$ do
    if $y' = c$ then
        /⋆ Calculate the shortest Hamming distance between $P'$ and $\mathcal{P}_c$ ⋆/
        Dist $\leftarrow \mathbf{H}(P', \mathcal{P}_c)$
        if Dist $\in (0, S_0)$ then
            Print "$y'$ is trusted"
        else if Dist $\in (S_0, S_{0.05})$ then
            Print "Require Human Judgement"
        else
            Print "$y'$ is not trusted"
        end if
    end if
end for
```

IV. Experiments and Analysis

To verify the proposed algorithm, we applied it in two tasks: 1) image classification on the MNIST dataset; 2) object detection on the PASCAL dataset.

A. Datasets and Implementation Details

1) Classification on MNIST Dataset: MNIST [1] dataset is a digital hand-written dataset which contains number 0-9, where the training dataset contains 60,000 images while the testing dataset consists of 10,000 images In this paper, activation patterns of the last activation layer with 40 neurons are monitored.

Table I presents the classification results on MNIST datasets. We treat activation patterns from those images with correct predictions on training set as the ground-truth activation patterns to generate central activation pattern $\widetilde{P}$ of different classes.

TABLE I
Prediction Results on MNIST

Datasets	Correct	Wrong	Accuracy
Training	59,605	395	99.34%
Testing	9,881	119	98.81%

2) Object Detection on the PASCAL Dataset: The PASCAL VOC 2007 dataset [28] contains 20 classes with around 10k images and 24k annotated objects. There are mainly four categories, i.e. **Vehicles**, **Households**, **Animals**, and **Person**. Each category contains several object classes. We have listed them in Table II and numbered them to facilitate further descriptions [2].

TABLE II
PASCAL object classes

Vehicles	Households	Animals	Person
Aeroplane: 0	Bottle: 4	Bird: 2	Person: 14
Bicycle: 1	Chair: 8	Cat: 7	
Boat: 3	Dining table: 10	Cow: 9	
Bus: 5	Potted plant: 15	Dog: 11	
Car: 6	Sofa: 17	Horse: 12	
Motorbike: 13	TV/Monitor: 19	Sheep: 16	
Train: 18			

In classification tasks, the ground-truth activation patterns are simply from those activation patterns with correct classifications on training set. Different from classification tasks, it is hard to extract the ground-truth activation patterns in object detection by using the same strategy.

Faster R-CNN[29] is implemented for detecting and recognising PASCAL objects in this paper. Different from extracting the last activation layers only in the classification task, we need to detect objects, i.e. determine labels and bounding boxes of the objects first, then the corresponding patterns in the close-to-output activation layers are extracted.

In object detection, both predicted labels and intersection over union (IoU) between the predicted bounding boxes and the ground-truth bounding boxes should be considered and the IoU is defined as

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}. \tag{11}$$

The predictions with $IoU > 0.5$ are defined as True Positive (TP), while those with IoU lower than 0.5 are defined as False Positive (FP) [28]. The number of TP/FP in training set and testing set is shown in Table III.

TABLE III
NUMBERS OF TP&FP ON PASCAL

Datasets	TP	FP	Accuracy
Training	12,411	26,429	32%
Testing	11,133	28,208	28%

In this case, we treat activation patterns of TP in the training set as the ground-truth patterns and apply them to generate central activation patterns for 20 classes.

B. Validation Results and Analysis

Fig. 3 and Fig. 4 show distributions of distances between the central patterns and activation patterns. In both figures, we use 'Same' to represent activation patterns with the same classes as the central activation patterns while 'Different' are those patterns with different classes. From these two distributions under different classes, activation patterns with the same classes as the central activation have shorter distances compared to those with different classes. By analysing Hamming distance distributions of activation patterns as well as their Kullback-Leibler divergences, it is confirmed that activation patterns with the same class are clusterred to their corresponding central activation pattern. As for a class, by using 'Same'/'Different' distributions, three monitoring zones can be built for monitoring the decision made by a neural network.

TABLE IV
MONITORING CLASSIFICATION RESULTS ON MNIST AND PASCAL

Dataset		Autonomous Correct Classification	Manual Human Decision	Misclassified
MNIST	Training	63.77%	35.56%	0.66%
	Testing	62.62%	36.73%	0.65%
PASCAL	Training	60.16%	36.24%	3.60%
	Testing	55.69%	38.99%	5.32%

Monitoring experiments are implemented on the MNIST and PASCAL datasets and the results are shown in Table IV. As for a test image, the neural network outputs a predicted result and its activation pattern. The distance between the pattern and its corresponding central activation pattern is calculated and to which zone the distance belongs to is also obtained. Table IV presents the monitoring classification results in different datasets. We use 'Autonomous Correct Classification' to represent faster R-CNN correct decisions - when the neural network works well. 'Manual' means additional human involvement is made in the decision making. 'Misclassified' represents that the proposed algorithm misclassifies the prediction made by the network. We can see that the proposed algorithm achieved low misclassified monitoring results, i.e. over 99% accuracy of prediction in both training and testing sets of MNIST. As for complicated object detection tasks, the monitoring process within faster R-CNN can also achieve a good performance with over 96% accuracy in the training phase and 94% accuracy during testing over PASCAL datasets.

V. CONCLUSIONS

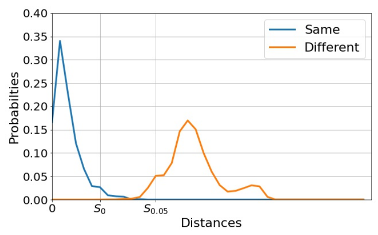
This paper presents a real-time activation pattern monitoring algorithm of the faster R-CNN in image classification and object detection. The real-time activation pattern monitoring algorithm is introduced to provide extra resilience in decision making for DNNs based systems. First the Kullback-Leibler divergence is calculated to find how different two distributions of the monitored patterns are. Next, the Hamming distance is calculated for decision making purposes. It gives the distance between the activation pattern of the current input and the corresponding central activation pattern. In this way a monitoring zone is represented and gives a level of trust in the obtained results. The proposed monitoring algorithm has been thoroughly verified over two different computer vision tasks: image classification and object detection - with MNIST and PASCAL data sets - and demonstrates its capacity and achieves very good monitoring performances.

Acknowledgements We are grateful to Dstl for funding this work via the project "Monitoring Activation patterns", to EPSRC through EP/T013265/1 project NSF-EPSRC: "ShiRAS. Towards Safe and Reliable Autonomy in Sensor Driven " and the support for ShiRAS by the USA National Science Foundation under Grant NSF ECCS 1903466.

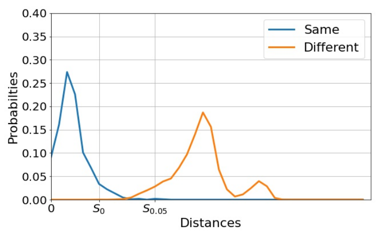
REFERENCES

[1] Y. LeCun, C. Cortes, and C. Burges, "MNIST handwritten digit database," 2010.

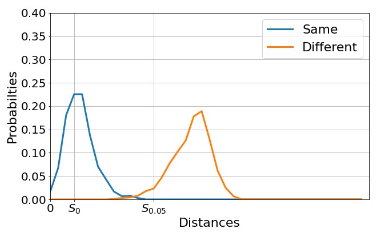
[2] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual



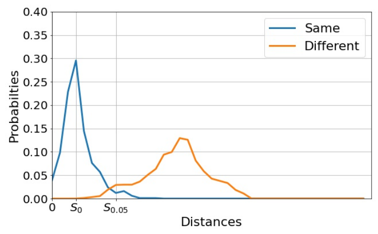
(a) Class: 0



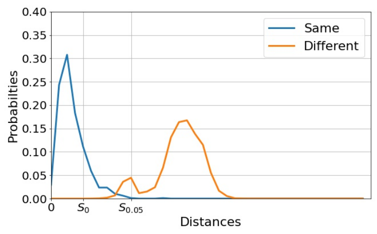
(b) Class: 1



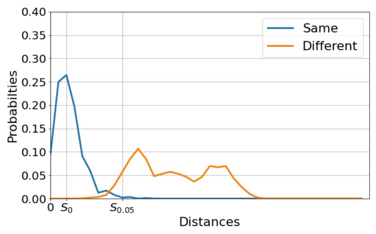
(c) Class: 2



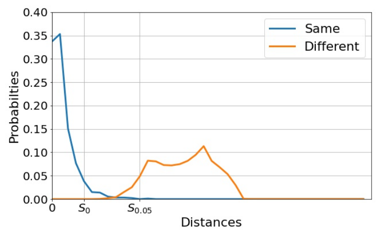
(d) Class: 3



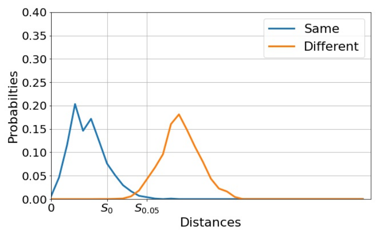
(e) Class: 4



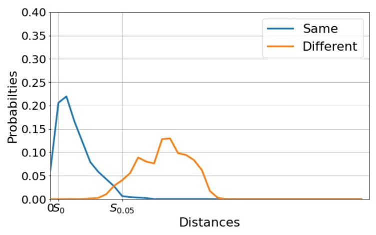
(f) Class: 5



(g) Class: 6

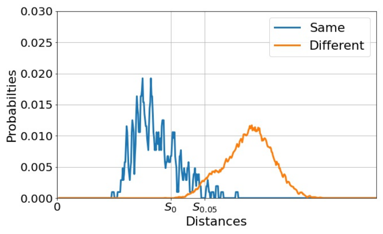


(h) Class: 7

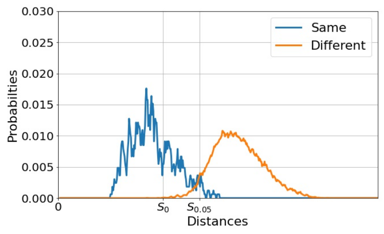


(i) Class: 8

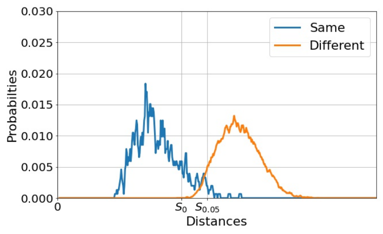
Fig. 3. From (a) to (i) represent the activation pattern distributions of digital number from 0 to 8 on MNIST dataset. 'Same' represents the distribution which the central activation pattern is compared to the activation patterns with the same class, while 'Different' represents the distribution which the central activation pattern is compared to the activation patterns with different classes.



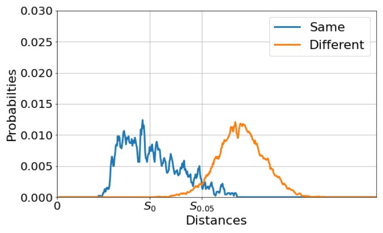
(a) Class: Aeroplane



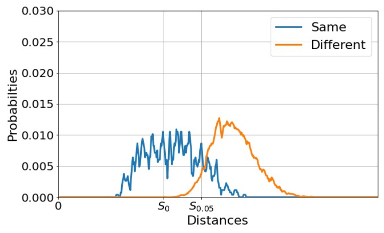
(b) Class: Bird



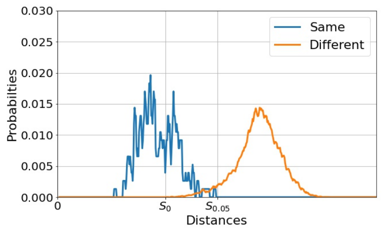
(c) Class: Bottle



(d) Class: Car



(e) Class: Chair



(f) Class: Dining table

Fig. 4. From (a) to (f) represent the activation pattern distributions of different classes on PASCAL dataset. 'Same' represents the distribution which the central activation pattern is compared to the activation patterns with the same class, while 'Different' represents the distribution which the central activation pattern is compared to the activation patterns with different classes.

object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, 2015.

[3] V. Sze, Y. Chen, T. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[4] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artificial Intelligence Review*, pp. 1 – 62, 2020.

[5] Z. Li, W. Yang, S. Peng, and F. Liu, "A survey of convolutional neural networks: Analysis, applications, and prospects," *ArXiv*, vol. abs/2004.02806, 2020.

[6] D. Kim, S.-H. Kim, T. Kim, B. B. Kang, M. Lee, W. Park, S. Ku, D. Kim, J. Kwon, H. Lee, J. Bae, Y.-L. Park, K.-J. Cho, and S. Jo, "Review of machine learning methods in soft robotics," *PLOS ONE*, vol. 16, no. 2, pp. 1–24, 02 2021.

[7] C. Duan, S. Junginger, J. Huang, K. Jin, and K. Thurow, "Deep Learning for Visual SLAM in Transportation Robotics: A review," *Transportation Safety and Environment*, vol. 1, no. 3, pp. 177–184, 01 2020.

[8] S. M. Grigorescu, B. Trasnea, T. T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *J. Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.

[9] Z. Kang, C. Catal, and B. Tekinerdogan, "Machine learning applications in production lines: A systematic literature review," *Computers & Industrial Engineering*, vol. 149, p. 106773, 2020.

[10] H. Wang and D.-Y. Yeung, "A survey on bayesian deep learning," *ACM Computing Surveys (CSUR)*, vol. 53, no. 5, pp. 1–37, 2020.

[11] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, V. Makarenkov, and S. Nahavandi, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Information Fusion*, vol. 76, pp. 243–297, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253521001081

[12] D. Dera, N. Bouaynaya, G. Rasool, R. Shterenberg, and H. Fathallah-Shaykh, "PremiUm-CNN: Propagating uncertainty towards robust convolutional neural networks," *IEEE Transactions on Signal Processing*, pp. 1–1, 2021.

[13] R. W. Hamming, "Error detecting and error correcting codes," *The Bell system technical journal*, vol. 29, no. 2, pp. 147–160, 1950.

[14] G. Carannante, D. Dera, G. Rasool, N. C. Bouaynaya, and L. Mihaylova, "Robust learning via ensemble density propagation in deep neural networks," in *Proc. of the IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2020, pp. 1–6.

[15] Y. Sun, X. Huang, D. Kroening, J. Sharp, M. Hill, and R. Ashmore, "Structural test coverage criteria for deep neural networks," in *Proceedings from the IEEE/ACM 41st International Conf. on Software Engineering: Companion Proceedings*, 2019, pp. 320–321.

[16] D. Dera, G. Rasool, and N. Bouaynaya, "Extended variational inference for propagating uncertainty in convolutional neural networks," in *Proceeings of the IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2019, pp. 1–6.

[17] P. Wang, N. C. Bouaynaya, L. Mihaylova, J. Wang, Q. Zhang, and R. He, "Bayesian neural networks uncertainty quantification with cubature rules," in *Proceedings from the 2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–7.

[18] L. Swiniarski, J. Bruna, and K. Cho, "Study of activation patterns," 2018. [Online]. Available: https://github.com/lucas-swiniarski/Activation-Patterns

[19] C.-H. Cheng, C.-H. Huang, T. Brunner, and V. Hashemi, "Towards safety verification of direct perception neural networks," *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1640–1643, 2020.

[20] C.-H. Cheng, C.-H. Huang, and G. Nührenberg, "nn-dependability-kit: Engineering neural networks for safety-critical autonomous driving systems," *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–6, 2019.

[21] C. Cheng, G. Nührenberg, and H. Yasuoka, "Runtime monitoring neuron activation patterns," in *Proceedings of 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 300–303.

[22] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of International Conference on Machine Learning*, 2016, pp. 1050–1059.

[23] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" 2017.

[24] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," 2017.

[25] M. Leucker and C. Schallhart, "A brief account of runtime verification," *The Journal of Logic and Algebraic Programming*, vol. 78, no. 5, pp. 293–303, 2009.

[26] C.-H. Cheng, "Provably-robust runtime monitoring of neuron activation patterns," *ArXiv*, vol. abs/2011.11959, 2020.

[27] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[28] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[29] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2016.