Contents

List of Figures

List of Tables

1

Timing Analysis of Synchronous Neural Networks

2

Run-time Enforcement of Synchronous Neural Networks

3
Run-time Verification of Synchronous Neural Networks

3.1 Methodology

The system designed for this case study was made to reflect a Autonomous Vehicle (AV) and its object detection mechanisms. The system used multiple techniques to tackle the inherent issues of the AV system, i.e. weakness to perturbed inputs and misclassification detection. The system's sensors include an overhead, 360° Light Detection and Ranging (LiDAR) apparatus, and a single, frontal facing camera. A solitary camera was sufficient to prove the efficacy of this solution, however it is to be noted that AV systems generally use multiple cameras, facing different directions, so that the controller can make properly informed decisions. The system used can be seen in Figure 3.1.

The LiDAR for this system was accurate 93% of the time [7], to closely simulate a real LiDAR system. The simulated camera outputs consisted of test images from both the Visual Object Classes (VOC) 2012 [3] and German Traffic Sign Recognition Benchmark (GTSRB) [6] datasets, in a combination of people, vehicles and various traffic signs. The LiDAR and camera outputs were handled by different parts of the controller. The camera outputs were fed into a Meta Neural Network (MNN) (see Figure 3.2) where they were classified by shape, colour and object type.

Utilising synchronous semantics [1], a Meta Neural Network (MNN), containing three other MNN ensembles, was created. Each Synchronous Neural Network (SNN) was implemented in the synchronous language Esterel [2]. Synchronous programming guarantees code that is deterministic, causal and bounded. Additionally, the calculation of Worst Case Execution Time (WCET) and Worst Case Reaction Time (WCRT) is enabled by using Esterel with C. Each ensemble synchronously combined the outputs of three different convolutional SNNs [5], providing increased prediction accuracy for shape, colour and object type. These ensembles ran in synchronous concurrency, each taking four logical ticks to run. The outputs of each ensemble were then combined into batch of outputs forming the *predicted output.*

The system controller was encapsulated by a run-time enforcer [4] that used sensor fusion to check for misclassifications made by the MNN. A safety automata (timed automata) **??** was designed to verify predictions made by the MNN and ensure utmost safety at all times. The automata started in a safe state, where control of the vehicle was autonomously handled by the system controller. If a misclassification was detected, the enforced policy entered an unstable state, still under autonomous control. Once enough time passed without further misclassifications, the vehicle entered the safe state again. However, if another misclassification was detected while unstable, the enforced policy entered a violation state and forced control of the AV to the driver. The vehicle would not enter autonomous mode again until the system was restarted. A diagram of the enforced policy's safety automaton is shown in Figure 3.3. This type of run-time enforcement,

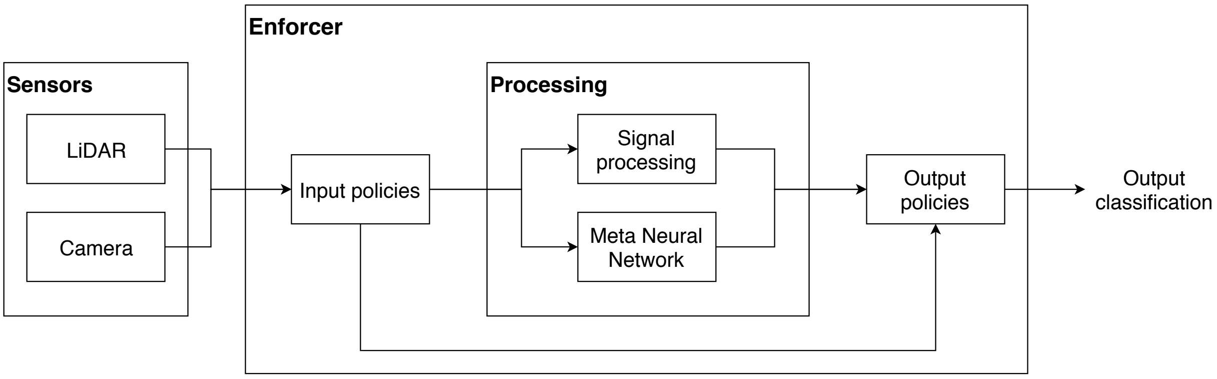


Figure 3.1: Block diagram showing the AV system with enforcer

where neither the inputs nor outputs of the sensors or controller are enforced, has been termed as *run-time verification*. *Run-time verification* refers to the verification of system parameters during run-time, while ensuring that the system is aware of any failed guards in the enforced policy.

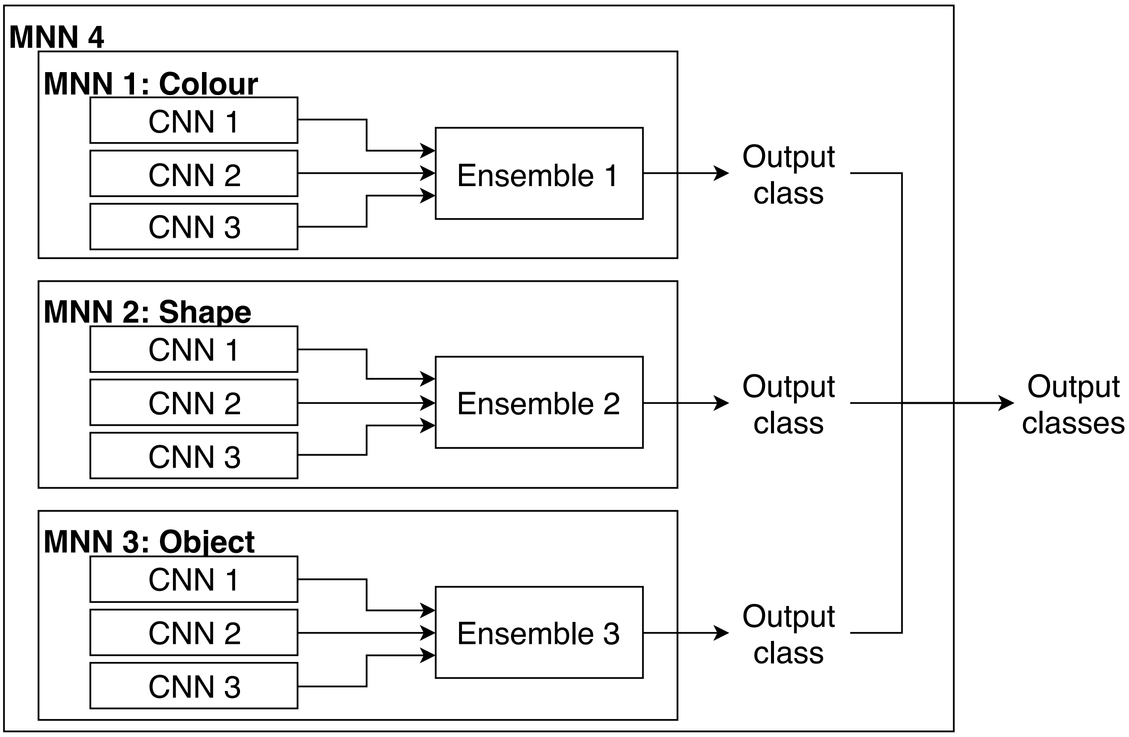
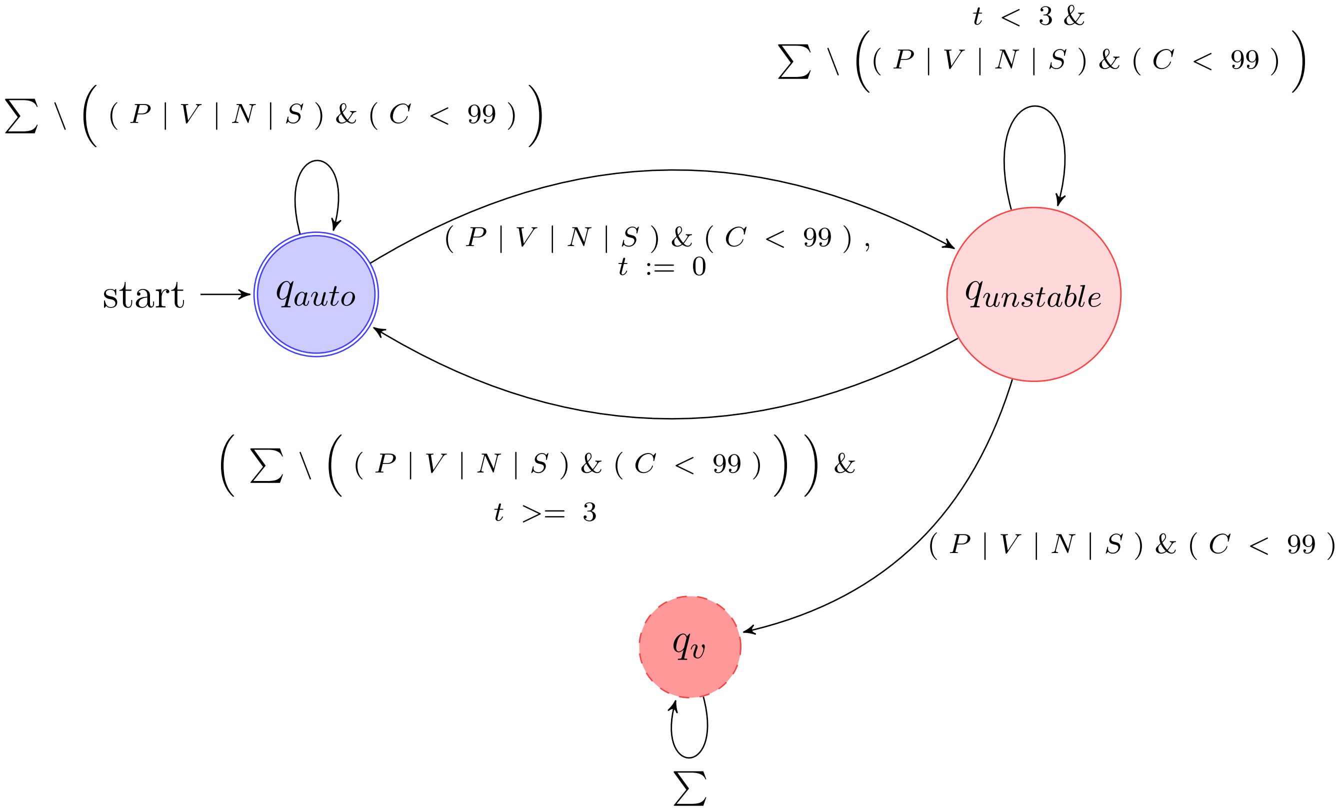


Figure 3.2: Block diagram showing the Meta Neural Network ensemble


$t < 3$ &
$\sum \setminus \Big((P \mid V \mid N \mid S) \, \& \, (C < 99) \Big)$
$\sum \setminus \Big((P \mid V \mid N \mid S) \, \& \, (C < 99) \Big)$
start
q_{auto}
$(P \mid V \mid N \mid S) \, \& \, (C < 99)$,
$t := 0$
$q_{unstable}$
$\Big(\sum \setminus \Big((P \mid V \mid N \mid S) \, \& \, (C < 99) \Big) \Big)$ &
$t >= 3$
$(P \mid V \mid N \mid S) \, \& \, (C < 99)$
q_v
$\sum$

- P: Misclassification of a person.
- V: Misclassification of a vehicle.
- N: Classification of an object when there is nothing.
- S: Misclassification of a traffic sign.
- C: Confidence rating of the SNN classification.
- t: Timer for the unstable state.

Figure 3.3: Enforcer policy for the AV prediction system
]

3.2 Results

This research provides a solution for two aspects of Autonomous Vehicle (AV) systems: predicting accurately with perturbations to the system's inputs and safely dealing with misclassifications by the system. The issue of input perturbations was addressed using a Meta Neural Network (MNN) of different convolutional Synchronous Neural Networks (SNNs), each SNN working in tandem to predict more accurately. Misclassification by the system's controller was addressed by implementing sensor fusion between cameras and LiDAR. This was done using a run-time enforcer that enforced a safety automaton.

To test the MNN's ability to deal with perturbations, the input images (taken from the VOC 2010 and GTSRB datasets) were perturbed by randomly replacing approximately 7% of the image pixels with randomly coloured pixels. Table 3.1 shows that without perturbations, the accuracy of the MNN was increased from 87.07% to 93.7%. Table 3.2 shows that the accuracy of the MNN hugely decreases when perturbations are present, from 87.07% to 41.6%. However, with sensor fusion and a relative safety policy, the accuracy is increased to 90.48%.

Test case	Person	Vehicle	Nothing of relevance	Street sign	Total
Number of misclassifications	1.23	3.82	3.27	4.61	12.93
Caught misclassifications	1.00	3.64	3.20	3.75	11.59
False negatives	0.79	0.90	0.37	2.90	4.96
Missed misclassifications	1.02	1.09	0.44	3.75	6.30

Table 3.1: Table showing results of the AV prediction SNN using the original images

Test case	Person	Vehicle	Nothing of relevance	Street sign	Total
Number of misclassifications	3.59	9.20	6.58	39.03	58.40
Caught misclassifications	2.73	8.00	6.37	35.87	52.98
False negatives	1.11	0.58	0.12	2.29	4.10
Missed misclassifications	1.97	1.78	0.32	5.45	9.52

Table 3.2: Table showing results of the AV prediction SNN using perturbed images

A
Appendix

References

[1] A. Benveniste, P. Caspi, S. A. Edwards, N. Halbwachs, P. Le Guernic, and R. De Simone, "The synchronous languages 12 years later," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 64–83, 2003.

[2] G. Berry and G. Gonthier, "The ESTEREL synchronous programming language: Design, semantics, implementation," *Sci. Comput. Program.*, vol. 19, no. 2, pp. 87–152, Nov. 1992.

[3] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[4] S. Pinisetty, P. Roop, S. Smyth, N. Allen, S. Tripakis, and R. von Hanxleden, "Runtime enforcement of cyber-physical systems," vol. 16, pp. 1–25, 09 2017.

[5] P. S. Roop, H. Pearce, and K. Monadjem, "Synchronous neural networks for cyber-physical systems," in *MEMOCODE '18 Proceedings of the 16th ACM-IEEE International Conference on Formal Methods and Models for System Design*, 2018.

[6] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, no. 0, pp. –, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608012000457

[7] P. Wei, L. Cagle, T. Reza, J. Ball, and J. Gafford, "LiDAR and Camera Detection Fusion in a Real Time Industrial Multi-Sensor Collision Avoidance System," *ArXiv e-prints*, Jul. 2018.