An Evaluation of "Crash Prediction Networks" (CPN) for Autonomous Driving Scenarios in CARLA Simulator

Saasha Nair,[1, 2] **Sina Shafaei,** [3] **Daniel Auge,** [3] **Alois Knoll** [3]
[1] TeCIP Institute, Scuola Superiore Sant'Anna, Pisa, Italy
[2] Department of Excellence in Robotics & AI, Scuola Superiore Sant'Anna, Pisa, Italy
[3] Technical University of Munich, Germany
saasha.nair@santannapisa.it, sina.shafaei@tum.de, daniel.auge@tum.de, knoll@in.tum.de

Abstract

Safeguarding the trajectory planning algorithms of autonomous vehicles is crucial for safe operation in mixed traffic scenarios. This paper proposes the use of an ensemble of neural networks to work together under the moniker of "Crash Prediction Networks". The system comprises multiple independent networks, each focusing on a different subset of sensory inputs. The aim is for the networks to work together in unison to reach a consensus of whether a vehicle might enter a catastrophic state to trigger an appropriate intervention. The proposed approach would act as an additional layer of safety by supervising the decision making module of an autonomous vehicle.

Though the proposed approach encompasses all the sensors and allied paraphernalia, the scope of this paper is exclusively limited to exploring safety monitors for visual sensors. The approach can, however, be extrapolated to other sensors. The evaluation was conducted using the CARLA simulator for simple driving scenarios studying the benefits of modeling temporal features to capture the motion in the environment. Additionally, the paper studies the importance of 'accounting for uncertainty' in models dealing with vehicle safety.

Introduction

Safety, as defined by Avizenis (Avizienis et al. 2004), is the "absence of catastrophic consequences on the user(s) and the environment". Therefore, safety engineering (Törngren et al. 2018) relates to "methods used to assess, eliminate and/or reduce risk to acceptable level". Safety in vehicles requires implementing vehicle-level behaviours (Koopman and Wagner 2017) that ensures safe and reliable performance across different contexts (McAllister et al. 2017). This could entail safety by construction, analysis, and verification/validation. Currently, however, testing is the main method used for ensuring safety in automated vehicles, and includes the following types (Huang et al. 2016): -

1. *Software testing*: helps to ensure the correctness of the program at the source code level via unit tests and testing tools.
2. *X-in-the-Loop (XiL) testing*: as introduced in (Stellet et al. 2015; Riedmaier et al. 2018), combines real-world and simulated components for testing the functionality of parts of the system. The software, hardware and the vehicle itself can be tested while simulating the residual parts.
3. *Testing in real traffic*: allows to study how the autonomous vehicle reacts to real-world driving scenarios on public roads with other participating actors.

Safety concerns are still seen to be a major impediment in the wide-scale adoption of autonomous vehicles (Rudolph, Voget, and Mottok 2018; Kalra and Paddock 2016; McAllister et al. 2017; Koopman and Wagner 2016). Safety monitors can potentially help to alleviate the safety concerns surrounding autonomous vehicles, by observing the inputs to and outputs from the various modules contained in the driving pipeline. These monitors usually differentiate between three main states that the system of interest, in this case, the car, may end up in (Machin et al. 2016). The first being the *catastrophic state*, that is one where the damage has already been done and from which the system cannot be recovered. The remaining two states, namely the safe state and the warning state, are non-catastrophic. In the *safe state* the system behaves as expected, without any constraints. The *warning state* is where the system is close to being involved in a catastrophe but can still be salvaged. For a system to transition from a safe to a catastrophic state, it must always pass through the warning state, thus the warning state can be thought of as the margin, where applying the correct intervention brings the system back to the safe state. Thus, it requires defining a safety strategy that identifies potential warning states and specifies corresponding safety rules to guide the response of the autonomous system.

The problem with existing approaches to safety monitoring, such as "Safety Monitoring Framework (SMOF)" (Machin et al. 2016), is that they require hand-picked and hard-coded limiters defined at the design stage describing what counts as *safe* or *unsafe* intervals of input and output. However, in neural networks, rather than the behaviour being hard-coded into the system, the models learn to detect patterns from the data they were trained on. This makes it difficult to combine neural networks with the existing safety monitoring techniques, and might in fact limit the expressive power and freedom of neural networks (Ashmore, Calinescu, and Paterson 2019). Thus this paper introduces "Crash Prediction Networks" (CPN) (Nair

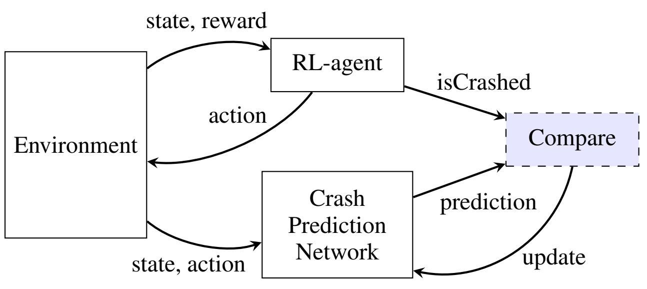

Figure 1: Training phase of the crash prediction network (Nair et al. 2019)

et al. 2019) as a 'safety by construction' solution for monitoring modules composed of neural networks. The aim of this paper is to explore the effectiveness of CPN in monitoring various aspects of vehicular safety. CPN for this purpose has been evaluated in a simulated environment and in the presence of dynamic obstacles. Additionally, the potential enhancement of prediction accuracy has also been investigated using temporal features in the architecture of the networks.

Overall Architecture

Consider an end-to-end setup for the driving system with a range of sensors that can be trained in simulation, the driving module uses information about the state of the environment to decide whether to continue straight, turn or apply brakes. The Crash Prediction Network can be thought of as an envelope around this driving module. The aim of CPN is to study the action decision obtained as output from the driving module, to determine whether it is likely to lead to a crash given the sensory information about the state. Safety monitors need to be extremely robust and reliable, thus to this effect CPN is suggested to be implemented as an ensemble (Ying Zhao, Jun Gao, and Xuezhi Yang 2005) of neural networks. Each network differs either in architecture or the subset of sensory data it consumes as input. The networks then work in unison to reach a consensus on whether the vehicle should continue with the current action or trigger an intervention.

During the training phase for CPN (Figure 1), a dataset is created by allowing a Reinforcement Learning driving agent to interact with the environment, and storing the information of the states encountered and the action taken along with the outcome. This allows the training of the CPN to be posed as a classification problem with two classes 'safe' or 'unsafe' indicating whether the action decision with the given state information led to a collision or not. During the operational phase (Figure 2), the ensemble of networks that compose CPN observe the input from the various sensors and the decision proposed by the driving module to predict the "safeness" of the outcome. If the proposed action is likely to lead to a catastrophic state, a predefined intervention is triggered thereby filtering out potentially dangerous action decisions, else, the proposed action is executed. The predefined intervention, also referred to as the "Fail Safe Mode" (as seen in Figure 2), can be of the form of transferring control to the human driver, or pulling over to the shoulder of the road and so on. This paper does not delve into the details of the intervention, but rather focuses on developing the technique to identify when the intervention should be triggered.

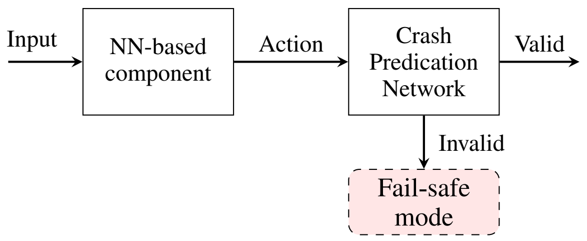

Figure 2: Operation phase of the crash prediction network (Nair et al. 2019)

A potential problem that might be encountered is that CPN might lose its relevance in the real-world over time due to the dynamic and constantly evolving state of the operational environment. The setup of CPN makes it possible to train the network in an iterative manner, which can be used to combat this problem. Implementing iterative training with CPN would require the continuous collection of live driving data during the operational phase and training CPN on the newly collected data at regular intervals. Additionally, an advantage with this technique is that it is not tightly coupled with the nature of the driving agent, though the experiments use an RL-based driving agent, one could easily swap it for any other driving agent.

Though the proposed architecture and techniques suggest the use of an ensemble of networks each focusing on a subgroup of different sensors, the scope of this evaluation is limited only to visual data collected using RGB cameras mounted on the hood of the car in simulation. The paper studies the importance of accounting for temporal features in the data on the prediction accuracy of CPN by modeling two different network architectures, namely "Simple CPN" and "Spatio-Temporal (ST)-CPN" as depicted in Figure 3. Simple CPN uses a single frame, *i.e.*, an image of the current state to make a prediction about the level of safety of the next state based on the proposed action of the driving module. This is achieved by using a VGG network (Simonyan and Zisserman 2014) (trained from scratch with the dataset collected in CARLA as described in the following section) for feature extraction, which is then concatenated with the action decision to perform the classification. The ST-CPN architecture, on the other hand, takes as input an N-frame long history, *i.e.*, the last N-frames encountered before the current state, along with the proposed driving decision.

Additionally, the importance of accounting for uncertainty is also studied in this evaluation. Standard deep learning uses point estimates for predictions (Goodfellow, Bengio, and Courville 2016). Thus, even when the model encounters inputs that are dissimilar to the ones that it was trained on, it might counterintuitively generate a high probability score, thereby making probability scores an unreliable

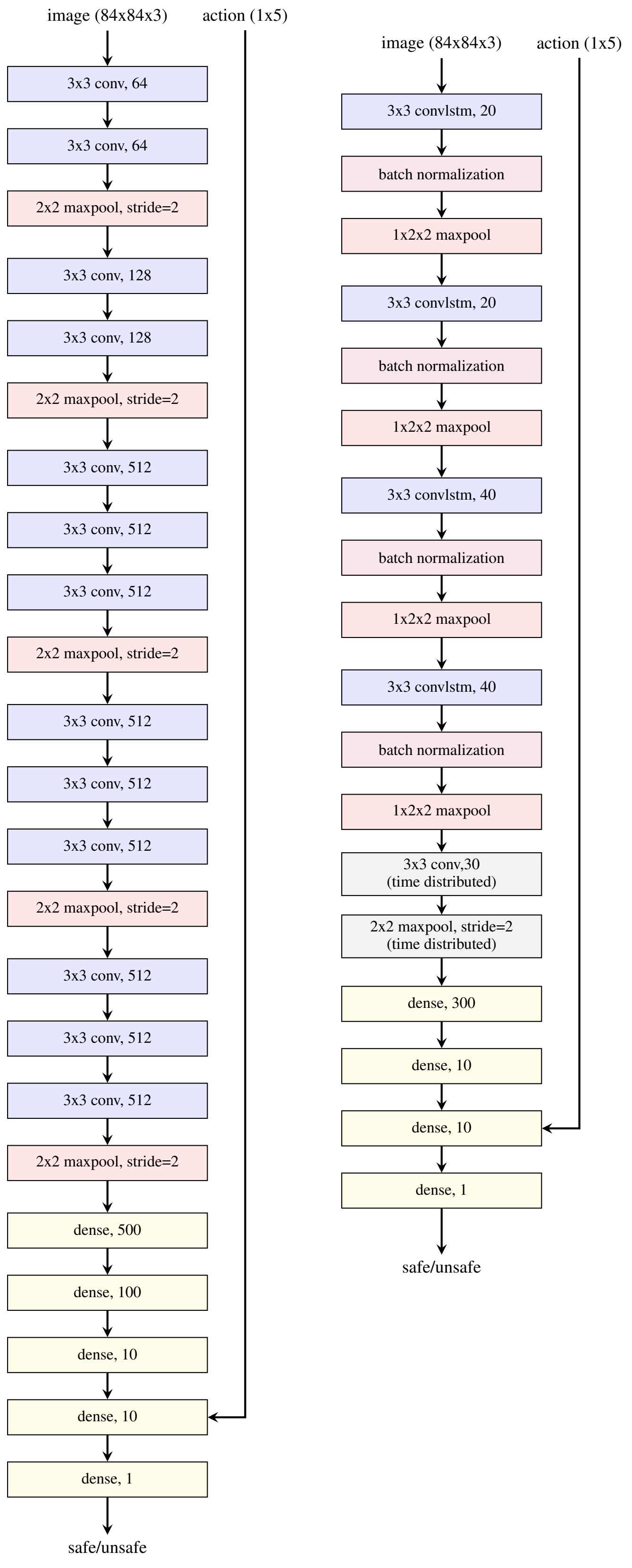

Figure 3: The Simple CPN (left), and ST-CPN (right) architectures used in conducting experiments

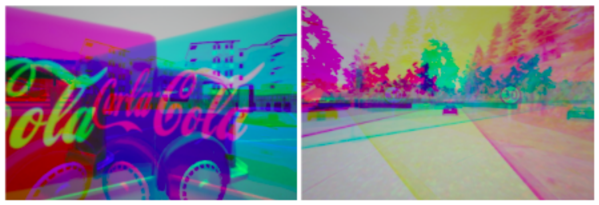

Figure 4: Format of the dataset used by Simple CPN

estimate of the model's confidence. This can be combated with the use of Bayesian deep learning which allows for a probabilistic approach to predictions by inferring distributions over the model parameters (Gal 2016; Kwon et al. 2020). Beside generating uncertainty estimates, Bayesian deep learning also helps reduce over-fitting. However, such models are difficult to train, and usually have intractable objective functions. Thus, in this work, we explore the need for accounting for uncertainty in safety monitors, like CPN, with the use of MC-Dropout (Gal and Ghahramani 2016) to approximate the Bayesian function.

Experiments and Evaluations

As stated in the previous section, the scope of the experiments was limited to inputs from RGB cameras placed on the hood of the car. Additionally, the focus of the networks was on preventing "locally avoidable catastrophes" (Saunders et al. 2018), *i.e.*, ones that can be avoided by adjusting the course of action when danger is imminent. This simplifying assumption eliminates the need for long-term strategic planning and focuses only at the point of failure. The experiments were conducted using *Carla 0.9.6*, "an open-source simulator for urban driving" (Dosovitskiy et al. 2017). The CARLA simulator provides a "Scenario Runner", which acts as an additional layer over the simulator, to support the testing of driving scenarios laid out by NHTSA as a list of pre-crash typologies (Najm et al. 2007). To study the importance of temporal features in safety prediction, ST-CPN is compared against the single frame input of Simple CPN. The evaluation in this paper, uses a history length of 10, meaning the last 10 image frames encountered by the ego vehicle are fed as input to the ST-CPN model. Both the models are extended for further experiments with uncertainty by applying MC-Dropout (Gal and Ghahramani 2016). To this effect, a Dropout layer with a probability of 0.4 is applied during training and inference after each of the trainable layers (namely, conv, convlstm and dense) in the models.

Dataset

Creating a representative dataset is a vital part of the deep learning pipeline. Data for the experiments in this paper was collected by allowing the ego vehicle backed by an RL-agent to drive in and interact with the simulated environment in CARLA. The simulator provides pre-built environments, called "Towns". Towns 01, 03 and 04 were used for training and validation, while towns 02 and 05 were used for testing. For the initial tests of the proposed approach presented here, the scale of the experiment was fairly limited with 18000 images (12000 safe and 6000 unsafe) used for training and 9000 (6000 safe and 3000 unsafe) used for testing.

For the experiments discussed here, the ego vehicle used in the simulation was equipped with three RGB cameras, placed on the left, right and center of the far-front of the hood of the car. The cameras enabled the ego vehicle to better perceive its surroundings, allowing for a wider field of view. As mentioned before, the two network architectures required two different formats of data. Thus, for the *Simple CPN* model single frames of images were stored. The RGB images from the three cameras were first converted to gray-scale to reduce the effect of color on the decision making of the neural network since the network only needs to detect the presence of an obstacle, and not the type of the obstacle. The single-channel gray-scale images from the three cameras were then combined depth-wise to create a single three-channel image of dimension 84x84x3 (as depicted in Figure 4), such that each channel represented one of the gray-scale images. To extend the data to the ST-CPN model, a similar procedure was followed to store a concatenation of the last 10 image frames per step, such that 10 image frame were stacked vertically to generate an (84x10)x84x3 image. Before being fed to the model as input, the single long image was processed into a series of 10 images of dimensions 84x84x3 akin to a short video. The two networks perform binary classification, such that the final dense layer contains a single neuron. Thus, a decision threshold value of 0.6 was used, such that if the output layer neuron produces a value greater than 0.6 then the state-action pair is classified as unsafe.

Evaluation Metrics

In real world scenarios, unsafe crash states are comparatively rare, which often leads to imbalanced datasets. This information is captured in the collected dataset by enforcing a mild imbalance, as can be noticed in the description of the dataset in the previous section. Thus, accuracy, the most commonly used metric in deep learning, does not suffice as it could lead to a false sense of success. Since falsely classifying unsafe states as safe is much worse than vice versa, the main focus of the models should be on reducing *false negatives*. Therefore, *recall* is an important metric for the CPN models, which has been captured in this paper via *precision-recall* curves.

Simple CPN with Static Obstacles Only

Before moving on to complex scenarios, it was necessary to test if a deep-learning based model could help predict the possibility of a crash based on state and action information. As a sanity check, in this first experiment the Simple CPN model was tested with only static obstacles, which meant crashes into walls, fences, rocks, crates on the road and other static objects in an urban scenario. With a test accuracy of 0.7907, the model was able to predict crash situations. However, the accuracy was inadequate to be practically usable.

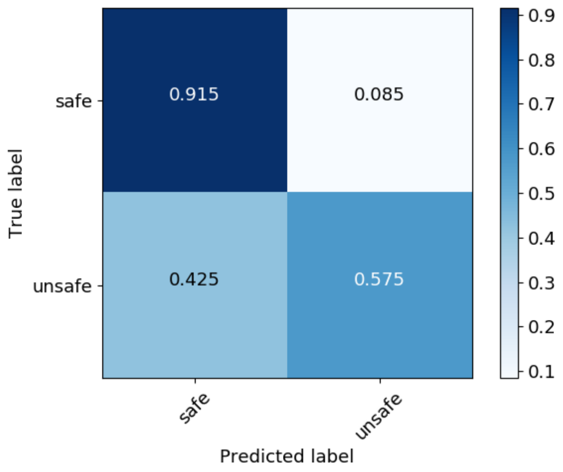

Figure 5: Simple CPN model on test set with dynamic obstacles

Simple CPN with Dynamic Obstacles

Seeing the results of the previous experiment, the simulation environment for the following experiments was extended to include dynamic obstacles in the form of 2- and 4- wheeler vehicles. The Simple CPN model was tasked with taking as input an image of the current state along with the proposed action decision to predict if the next state would be ‘unsafe’. The model was able to replicate the success of the previous experiment, achieving a test accuracy of 0.8018 and AUC-PRC score of 0.7624.
As per the confusion matrix depicted in Figure 5, the number of false negatives were quite high. However, for safety-critical applications such as autonomous vehicles, it is important to reduce false negatives to as low as possible. In order to improve the results of the classification, class weights were introduced in the Simple CPN architecture. The class weights were used during training in the ratio of 1:2, such that the penalty of wrongly classifying a crash case applied to the loss was double that of the penalty of wrongly classifying a non-crash case. The results summarized in table 1 show a slight improvement of the overall accuracy and an increased recall score.

ST-CPN with Dynamic Obstacles

Despite the presence of dynamic objects in the environment, the Simple CPN made its decision based on only a single frame of information. This does not allow the network to model the motion of the ego vehicle as well as other obstacles in the environment. To better deal with moving objects, the ST-CPN model was introduced, which uses ConvLSTM to process a contiguous series of 10 frames of images.

Looking at Figure 7 and Figure 5, it is evident that the ST-CPN model was able to much better identify “unsafe” situations, thereby reducing the number of false negatives. This was further visible on comparing the results of the ST-CPN model against the Simple CPN model in table 1 on the test set. The disadvantage however was that, due to the

Table 1: Comparison of classification metrics on the test set with clear weather.

Type	ACCURACY	RECALL	PRECISION	AUC-PR	Note
Simple CPN	0.8018	0.57	0.77	0.7624	
Simple CPN	0.8131	0.68	0.74	0.7706	with loss adaption
ST-CPN	0.8773	0.74	0.87	0.8951	
Bayesian Simple CPN	0.8015	0.57	0.77	0.7624	Uncertainty: 0.0162
Bayesian ST-CPN	0.8281	0.63	0.71	0.8274	Uncertainty: 0.0166
Bayesian Combined	0.8328	0.62	0.71	0.8382	Uncertainty: 0.0222

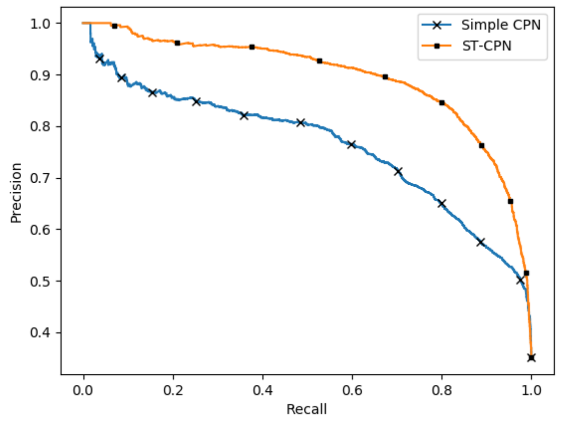

Figure 6: Performance of the Simple CPN model against the ST-CPN model on the test set with dynamic obstacles

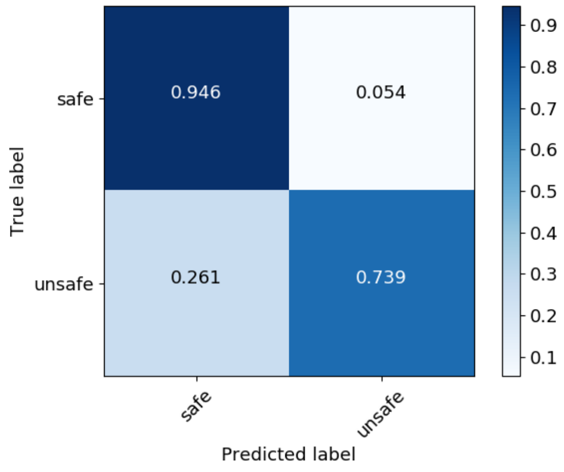

Figure 7: ST-CPN model on test data with dynamic obstacles

higher complexity of the model, the inference time of ST-CPN increases by a factor of 10 when compared to that of Simple CPN.

Following the performance of the ST-CPN model, a study on the reaction of the model to out-of-distribution data, *i.e.*, data that is slightly different from the conditions that the model was trained on, was performed. For this 3000 images with clear weather, similar to training conditions, and 3000 images with rainy weather were collected. These were referred to as "*Test small*" and "*Test rainy*", respectively. As can be seen in table 2 and Figure 8, the rainy condition causes the model to misclassify comparatively more "unsafe" scenarios, thereby dropping the performance of the model.

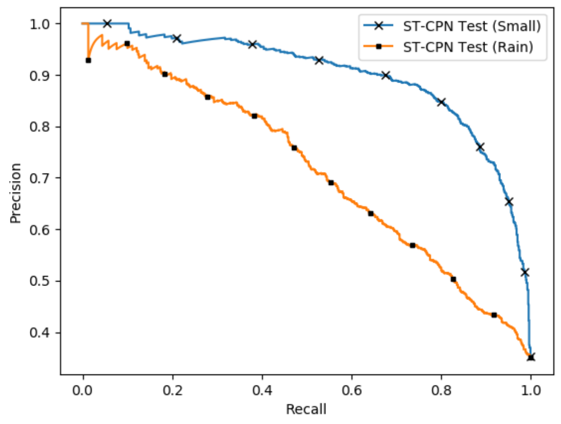

Figure 8: Precision-recall curve comparing the performance of the ST-CPN model on test sets with clear (default) and rainy weather conditions

Simple CPN with Uncertainty in dynamic environment

Though accounting for temporal features in the prediction model helped improve the performance, it suffered from drop in performance when encountering out-of-distribution data. Since the real-world is constantly evolving and cannot be modeled completely in the training data, it is necessary to have in place techniques for the models to deal with such data. As pointed out in an earlier section, standard deep

Table 2: Comparison of classification metrics on test sets with clear and rainy weather.

Type	ACCURACY	RECALL	PRECISION	AUC-PR	Note
Bayesian Simple CPN	0.9443	0.90	0.93	0.9740	Uncertainty: 0.0139, training set
Bayesian Simple CPN	0.8030	0.57	0.78	0.7679	Uncertainty: 0.0163, test set
Bayesian Simple CPN	0.6173	0.69	0.45	0.5408	Uncertainty: 0.0212, rainy test set
ST-CPN	0.8816	0.75	0.88	0.8983	test set
ST-CPN	0.7770	0.45	0.79	0.7140	rainy test set

learning gives no information about the confidence of the model in its prediction. Thus, both the models from the previous experiments were extended with "MC-Dropout" (Gal and Ghahramani 2016) by placing a dropout layer after every convolutional and dense layer, except the output layer. The dropout was then applied not just during training but also during testing, wherein each input was used to generate T predictions to calculate the mean and variance. The variance on each observation/data point is an indication of how certain the model is in its prediction, which in turn shows how similar the test data is to the training data. Since performance improvements of using class weights were negligible, the Bayesian version of the Simple CPN model was trained without class weights.

The advantages of using uncertainty becomes clear when using a dataset that differs considerably from the training data. Thus, the performance of Bayesian Simple CPN is evaluated on "*Test small*" and "*Test rainy*" from the the previous experiment. As can be seen from table 2, the test set with clear weather has similar uncertainty estimates as the training set, however, while using the test set with the rainy weather, the uncertainty estimate increases. Thus, even with a change as small as variation in weather can increase uncertainty. The uncertainty estimates can therefore be useful in building trust in the prediction of the CPN models.

The benefit of estimating the confidence of the model in its decision however comes at the cost of a significantly longer inference time. During evaluations, the model took 10 times longer to compute the class labels and their corresponding confidence values.

ST-CPN with Uncertainty in Dynamic Environment

To study the effect of uncertainty estimates, combined with the benefits of modeling temporal features, the ST-CPN network, similar to the previous experiment with Simple CPN, was extended using MC-Dropout. To ensure that the model was comparable to the Bayesian version of the Simple CPN model of the previous experiments, both of the models were trained to have a similar validation accuracy of about 0.80. Additionally, to capture the essence of the proposed CPN model with multiple independent neural networks, the outputs of the "Bayesian Simple CPN" model and the "Bayesian ST-CPN" model were combined as a weighted average with higher weight being assigned to the latter. This model is referred to as the "Combined" model.

Evaluating the performance over a range of probability thresholds, the combined model performs slightly better than both individual models, as seen in Figure 9, thereby showing the benefit of modeling CPN as an ensemble of diverse networks working in unison to make a prediction about the future state.

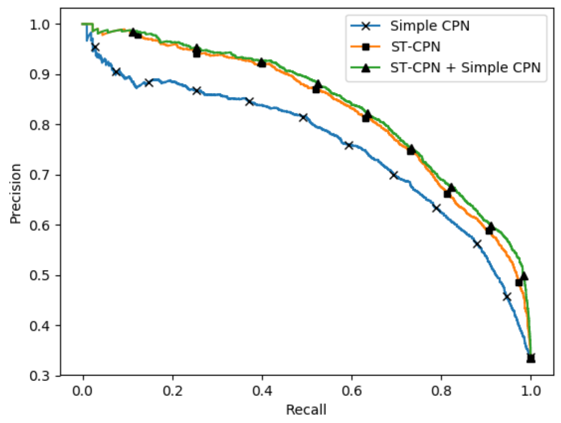

Figure 9: Precision-recall curve comparing the performance of the Bayesian versions of Simple CPN model, ST-CPN model and a weighted combination of the two models

Conclusion

The paper evaluated the proposed system of "Crash Prediction Network" and demonstrated its advantage in exploiting the expressiveness of neural networks, while also maintaining robustness due to the ensemble-like setup. CPN provides redundancy to the setup by adding a layer of "safety prediction". CPN has the power to *re-check* the decisions of the driving module, thereby, sharing the load of safe decision making with the driving module. More importantly, it should be noted that CPN is not just a solution for futuristic autonomous vehicles, but can in fact be integrated with current levels of automation to monitor driving decisions of human drivers to ensure safer and more conservative driving, thereby reducing human error on roads.

Based on the evaluations discussed in the paper, the importance of accounting for temporal features to model motion in the environment is evident. Thus, future work involves extending the CPN model to support other sensory data as input, along with examining longer history lengths.

Furthermore, one could increase the level of granularity of the prediction label beyond just 'safe' and 'unsafe'. This would allow for a more sensitive setup with the ability to trigger situation-specific interventions, which is not currently possible. Additionally, the networks that form the ensemble could be extended beyond merely sensory inputs to make it geographical region and/or rule-specific with considerably little effort.

References

Ashmore, R.; Calinescu, R.; and Paterson, C. 2019. Assuring the machine learning lifecycle: Desiderata, methods, and challenges. *arXiv preprint arXiv:1905.04223* .

Avizienis, A.; Laprie, J.-C.; Randell, B.; and Landwehr, C. 2004. Basic concepts and taxonomy of dependable and secure computing. *IEEE transactions on dependable and secure computing* 1(1): 11–33.

Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; and Koltun, V. 2017. CARLA: An open urban driving simulator. *arXiv preprint arXiv:1711.03938* .

Gal, Y. 2016. Uncertainty in deep learning. *University of Cambridge* 1.

Gal, Y.; and Ghahramani, Z. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, 1050–1059.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep learning*. MIT press.

Huang, W.; Wang, K.; Lv, Y.; and Zhu, F. 2016. Autonomous vehicles testing methods review. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 163–168. IEEE.

Kalra, N.; and Paddock, S. M. 2016. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice* 94: 182–193.

Koopman, P.; and Wagner, M. 2016. Challenges in autonomous vehicle testing and validation. *SAE International Journal of Transportation Safety* 4(1): 15–24.

Koopman, P.; and Wagner, M. 2017. Autonomous vehicle safety: An interdisciplinary challenge. *IEEE Intelligent Transportation Systems Magazine* 9(1): 90–96.

Kwon, Y.; Won, J.-H.; Kim, B. J.; and Paik, M. C. 2020. Uncertainty quantification using Bayesian neural networks in classification: Application to biomedical image segmentation. *Computational Statistics & Data Analysis* 142: 106816.

Machin, M.; Guiochet, J.; Waeselynck, H.; Blanquart, J.-P.; Roy, M.; and Masson, L. 2016. Smof: A safety monitoring framework for autonomous systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 48(5): 702–715.

McAllister, R.; Gal, Y.; Kendall, A.; Van Der Wilk, M.; Shah, A.; Cipolla, R.; and Weller, A. 2017. Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning. International Joint Conferences on Artificial Intelligence, Inc.

Nair, S.; Shafaei, S.; Kugele, S.; Osman, M. H.; and Knoll, A. 2019. Monitoring safety of autonomous vehicles with crash prediction networks. In *SafeAI@ AAAI*.

Najm, W. G.; Smith, J. D.; Yanagisawa, M.; et al. 2007. Pre-crash scenario typology for crash avoidance research. Technical report, United States. National Highway Traffic Safety Administration.

Riedmaier, S.; Nesensohn, J.; Gutenkunst, C.; Düser, T.; Schick, B.; and Abdellatif, H. 2018. Validation of X-in-the-Loop Approaches for Virtual Homologation of Automated Driving Functions. In *GSVF-Symposium. Graz*.

Rudolph, A.; Voget, S.; and Mottok, J. 2018. A consistent safety case argumentation for artificial intelligence in safety related automotive systems.

Saunders, W.; Sastry, G.; Stuhlmueller, A.; and Evans, O. 2018. Trial without error: Towards safe reinforcement learning via human intervention. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 2067–2069. International Foundation for Autonomous Agents and Multiagent Systems.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* .

Stellet, J. E.; Zofka, M. R.; Schumacher, J.; Schamm, T.; Niewels, F.; and Zöllner, J. M. 2015. Testing of advanced driver assistance towards automated driving: A survey and taxonomy on existing approaches and open questions. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, 1455–1462. IEEE.

Törngren, M.; Zhang, X.; Mohan, N.; Becker, M.; Svensson, L.; Tao, X.; Chen, D.-J.; and Westman, J. 2018. Architecting Safety Supervisors for High Levels of Automated Driving. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 1721–1728. IEEE.

Ying Zhao; Jun Gao; and Xuezhi Yang. 2005. A survey of neural network ensembles. In *2005 International Conference on Neural Networks and Brain*, volume 1, 438–442. doi:10.1109/ICNNB.2005.1614650.