An Active Learning Reliability Method for Systems with Partially Defined Performance Functions

Jonathan Sadeghi
Five AI Ltd.
jonathan.sadeghi@five.ai

Romain Mueller
Five AI Ltd.

John Redford
Five AI Ltd.

1 Introduction

Estimating the probability that the performance of a system is satisfactory under uncertain or variable operating circumstances is an important step towards deploying such systems safely in the real world. This is especially important in safety critical application such as autonomous driving, where finding rare but catastrophic failures has been identified as a important challenge [1]. Powerful active learning approaches based on Gaussian processes (GP) have been proposed as a solution to this problem [2–7] and achieve state of the art performance (we describe related work in detail in Section B). However, such approaches cannot be applied to problems where the performance of the system may be *undefined* under certain specific circumstances, a situation which often occurs in the autonomous vehicle domain [1, 8–10] (for a motivating example see Appendix A). Naïvely masking away regions where the performance of the system is undefined would introduce discontinuities and leads to poor performance since Gaussian processes with stationary kernels are not well suited to the regression of discontinuous targets [11]. In this work, we extend these methods from first principles to the case where the performance function can be undefined by using a hierarchical model (termed hGP) for the system performance, where undefined performance is classified before the performance is regressed. The code is available at `https://github.com/fiveai/hGP_experiments/`.

We consider a system whose performance is described by a function $g : \mathcal{X} \mapsto \mathbb{R} \cup \{\text{NaN}\}$, where $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^k$ are random environmental variables affecting the system, $g(\boldsymbol{x}) < 0$ denotes an undesirable event (a failure), and $g(\boldsymbol{x}) = \text{NaN}$ is an event of unspecified performance. An undefined value does not indicate that an undesirable event has occurred for a particular $\boldsymbol{x}$, and therefore we wish to classify these $\boldsymbol{x}$ differently to the failure events. The rate of failures is quantified using the probabilistic threshold robustness (PTR) of the system, which we define as

$$p_f = \int_{\mathcal{X}} \mathbb{1}\left[g(\boldsymbol{x}) < 0 \cap g(\boldsymbol{x}) \neq \text{NaN}\right] p(\boldsymbol{x}) d\boldsymbol{x}, \tag{1}$$

where $\mathbb{1}$ is the indicator function and $p(\boldsymbol{x})$ is the probability density (mass) function of $\boldsymbol{x}$ [12]. Eq. (1) represents the probability that the system is in the failure state, while disregarding the 'uninteresting' cases where the performance is undefined. Note that we are not attempting to model the distribution of environment variables and treat $p(\boldsymbol{x})$ as given. Estimating p_f in Eq. (1) using a vanilla Monte Carlo simulation can be computationally expensive since identifying a failure rate lower than ϵ will typically require at least $1/\epsilon$ tests [13]. In order to reduce the required number of samples, we use the Adaptive Kriging Monte Carlo Simulation (AK-MCS) algorithm, a simple active learning technique based on Gaussian processes which was shown to provide an extremely efficient evaluation of the PTR measure for previously studied problems [2], and extend it to partially undefined performance functions. We perform experiments comparing our proposed methodology to several naïve baselines on problems for which the results are known analytically. We find that our approach produces a more accurate estimation of p_f and also that the surrogate model is a more accurate representation of the true performance function.

2022 NeurIPS Workshop on Gaussian Processes, Spatiotemporal Modeling, and Decision-making Systems.

2 Approach

We modify the AK-MCS active learning algorithm by using a different Gaussian Process and acquisition function to Echard et al. [2]. We use a hierarchical Gaussian process model for the rule function, consisting of separate regression and classification Gaussian processes, and a modified acquisition function which minimises the catastrophic event classification error to yield an optimal surrogate model of the rule function. Otherwise our proposed algorithm proceeds in the same way as the AK-MCS algorithm, i.e. an initial training set is chosen to train a Gaussian process, and then subsequent evaluations of the performance function, g, are chosen iteratively by maximising a function of the Gaussian process known as the acquisition function, which are then used to retrain the Gaussian process. The algorithm terminates when the coefficient of variation (CoV) of the failure probability computed using the Gaussian process is below some threshold, and the predicted misclassification probability is also below some threshold. This algorithm is shown in Algorithm 1.

Let y_* be the predicted performance for the test input $\boldsymbol{x}_* \in \mathcal{X}$ where $y \in \mathbb{R} \cup \{\text{NaN}\}$, and let the dataset of training examples $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) | i = 1, ..., n\}$. We model the predictive distribution $p(y_* | \boldsymbol{x}_*, \mathcal{D})$ hierarchically as

$$p(y_* | \boldsymbol{x}_*, \mathcal{D}) = \begin{cases} p(y_* | \boldsymbol{x}_*, \mathcal{D}, y_* \neq \text{NaN}) p(y_* \neq \text{NaN} | \boldsymbol{x}_*, \mathcal{D}) & \text{if } y_* \neq \text{NaN}, \\ p(y_* = \text{NaN} | \boldsymbol{x}_*, \mathcal{D}) & \text{if } y_* = \text{NaN}, \end{cases} \tag{2}$$

where $p(y_* | \boldsymbol{x}_*, \mathcal{D}, y_* \neq \text{NaN})$ is the predicted regression distribution for y_* at the test input $\boldsymbol{x}_*$ given that y_* is defined, and $p(y_* = \text{NaN} | \boldsymbol{x}_*, \mathcal{D})$ is the predicted classification probability that y_* is undefined for $\boldsymbol{x}_*$. We model these distributions with separate GPs; for $p(y_* | \boldsymbol{x}_*, \mathcal{D}, y_* \neq \text{NaN})$ GP regression is used, and for $p(y_* = \text{NaN} | \boldsymbol{x}_*, \mathcal{D})$ GP classification is used. The conditional prediction of the failure event can easily be calculated as $p(y_* < 0, y_* \neq \text{NaN} | \boldsymbol{x}_*, \mathcal{D})$, which can be used to define an acquisition function, $p_{\text{misclassification}}(\boldsymbol{x})$, based on probability of misclassification of $y_* < 0 \cap y_* \neq \text{NaN}$, as in Echard et al. [2]. We give more details about our modelling approach in Appendix C. Finally, our hierarchical model Eq. (2) can be used to compute the failure probability as

$$p_f \approx \int_{\mathcal{X}} \mathbb{1}[p(y_* < 0, y_* \neq \text{NaN} | \boldsymbol{x}_*, \mathcal{D}) > 0.5] p(\boldsymbol{x}_*) d\boldsymbol{x}_*. \tag{3}$$

The termination criteria for the AK-MCS algorithm will determine the error in the failure probability computed using the Gaussian process in Eq. (3), in addition to bounding the error of the hierarchical Gaussian process model. This ensures that the model is sufficiently accurate to be used by engineers to make predictions about the behaviour of the system.

Algorithm 1: Hierarchical Gaussian Process PTR Active Learning Method

Input: GP prior $\mathcal{GP}(0, k)$, termination threshold η, model $g(\boldsymbol{x})$
  Define proposal set $\mathcal{S}$: sample n_{mc} points from $p(\boldsymbol{x}_*)$.
  Define initial design of experiments: sample n_E points uniformly from $\mathcal{S}$ and evaluate with model $g(\boldsymbol{x})$ to define $\hat{\mathcal{S}} = \{(\boldsymbol{x}_i, y_i) | i = 1, ..., n_E\}$
  while $CoV > 0.1$ **do**
    while $\max_{\boldsymbol{x}} p_{\text{misclassification}}(\boldsymbol{x}) > \eta$ **do**
      Train GP on $\hat{\mathcal{S}}$
      Compute $\mu(\boldsymbol{x})$, $\sigma(\boldsymbol{x})$, $p_{\text{nan}}(\boldsymbol{x})$ from hierarchical GP for all $\boldsymbol{x} \in \mathcal{S}$.
      Choose most likely misclassified $\boldsymbol{x}$: $\boldsymbol{x}_* = \arg\max_{\boldsymbol{x} \in \mathcal{S}} p_{\text{misclassification}}(\boldsymbol{x})$
      Observe $y_* = g(\boldsymbol{x}_*)$ and Add $(\boldsymbol{x}_*, y_*)$ to $\hat{\mathcal{S}}$
    end while
    Estimate p_f using Monte Carlo simulation with Gaussian Process on $\mathcal{S}$ using Eq. (3)
    Calculate $CoV = \sqrt{\frac{(1-p_f)}{p_f |\mathcal{S}|}}$
    Sample n_{mc} points from $p(\boldsymbol{x}_*)$ and evaluate with model $g(\boldsymbol{x})$ to add to $\mathcal{S}$
  end while
Output: Fitted hierarchical GP and p_f computed using Eq. (3).

3 Experiments

Benchmark tasks: We evaluate our methodology on two benchmark problems where the system performance is partially undefined and for which p_f can be calculated analytically:

- *Toy function* The system performance (plotted in Fig. 4a) is given by

$$g(\boldsymbol{x}) = \begin{cases} \text{NaN} & \text{if } 0.215 < \boldsymbol{x} < 0.6, \\ \cos(8\boldsymbol{x}) & \text{otherwise,} \end{cases} \tag{4}$$

  where the uncertain variable x is distributed with $p(\boldsymbol{x}) = \mathcal{U}[0, 1]$.
- *Autonomous driving (AD)* A model of an autonomous vehicle joining a main road at a T-junction where the vehicle accelerates linearly at a_{ego} from stationary, whilst the approaching vehicle travels from x_a at v_a. The thresholded minimum lateral distance between the autonomous vehicle and an approaching vehicle is given by the performance function

$$g(x_a, v_a) = \begin{cases} \text{NaN} & \text{if } d_{\text{min}}(x_a, v_a) < d_{\text{threshold}} \text{ and } |x_a| < x_{\text{lim}}, \\ d_{\text{min}}(x_a, v_a) - d_{\text{threshold}} & \text{otherwise,} \end{cases} \tag{5}$$

  where $d_{\text{threshold}}$ is a constant safe distance threshold, x_{lim} is a upper limit of the perceptual range of ego, and the closest approach distance between the vehicles is defined as $d_{\text{min}}(x_a, v_a) = \max\left(-\left(x_a + v_a^2/(2a_{\text{ego}})\right), 0\right)$. A full derivation of the performance function and explanation of the physical variables and their associated distributions is provided in Appendix A. Undefined values here represent scenarios when the autonomous vehicle decides not to join the main road, and so the lateral distance along the road is undefined.

Baselines: The hierarchical GP (hGP) will be compared with the following baseline methodologies:

- *Masked GP*: AK-MCS [2] with a regression GP where NaN values are masked with positive constant $\alpha > 0$, i.e.

$$\tilde{g} = \begin{cases} g(\boldsymbol{x}) & \text{if } g(\boldsymbol{x}) \neq \text{NaN}, \\ \alpha & \text{if } g(\boldsymbol{x}) = \text{NaN}. \end{cases} \tag{6}$$

- *Active GP Classification* (GPC): similarly to Kapoor et al. [14], we apply a GP classifier to classify the event $y_* < 0 \cap y_* \neq \text{NaN}$ and use this to replace the GP in AK-MCS.

Hyperparameters for each algorithm are shown in Appendix D. Metrics used to compare the models are described in further detail in Appendix E.1 We repeat the experiments with different values of α ($\alpha = 0.1$, $\alpha = 0.5$, $\alpha = 1.0$) for the Masked GP in Appendix E.3, where no significant differences are observed for different values of α (results for $\alpha = 1$ are shown in this section).

Model Accuracy: Firstly we run each algorithm for 150 iterations, i.e. neglect the termination criteria by setting $\eta = 0$, in order to study the accuracy of the models independently of the termination criteria. Fig. 1 shows the convergence of the failure probability and maximum predicted misclassification probability for the models for the Toy and Autonomous experiments. In addition, in Appendix E.2 the convergence of the failure probability (p_f) and F_1score for the models is shown. We observe that the convergence for the hGP is far faster than for the other models. All methodologies obtain a value for p_f consistent with the analytically computed value when considering the error implied by the CoV. The convergence of p_f and predicted misclassification probability for the masked GP is erratic, which is indicative of the misspecification of the model. Visualising the fitted models in Fig. 2, reveals that the low F_1score and average precision are caused by the length scale for the masked GP becoming extremely short due to the discontinuity in the masked performance function, resulting in a large predicted variance and hence erroneous class scores. hGP outperforming GPC is unsurprising, as the hGP utilises more information about the magnitude of the performance function in order to make a more educated selection of the next point to query.

Overall, although all models eventually accurately estimate p_f, the hGP clearly provides a more accurate classification of the failure region, more stable training, and class probabilities which better represent the state of knowledge given the available data.

Termination Criteria: We analyse the ability of the models to terminate the active learning loop after an appropriate number of iterations by repeating the experiments in the previous section with

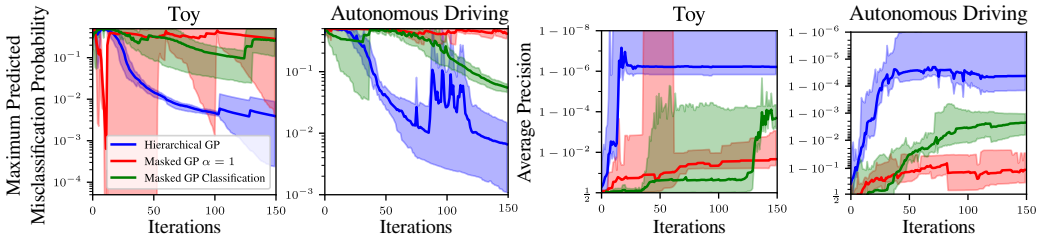


Figure 1: Convergence of average precision and maximum predicted misclassification probability for fitted GPs for the toy function and AD experiments. The shaded area represents the minimum and maximum of 5 repeats, and the dark line represents the mean.

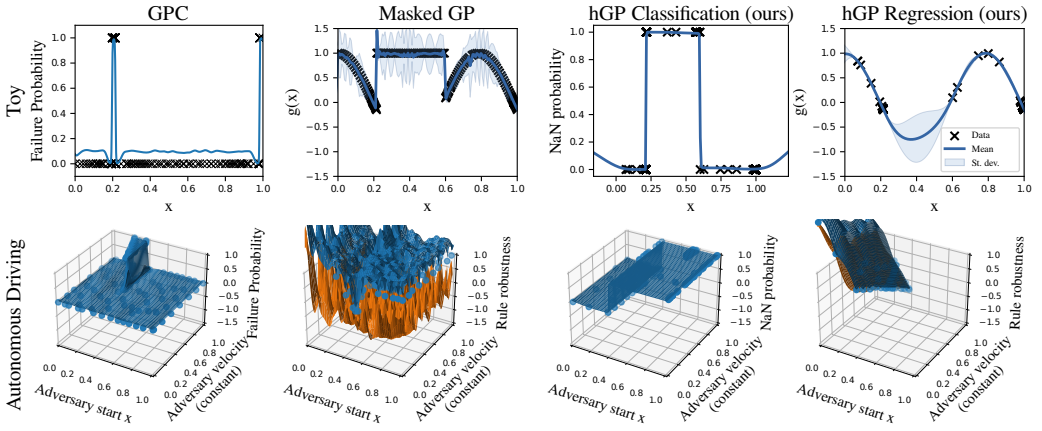


Figure 2: Plotted models for both experiments. For regression GPs, one standard deviation prediction bounds are shown for AD GPs as orange/blue surfaces and for the toy function GPs in shaded blue. For classification GPs only the predicted probability is shown. The points represent training data.

the termination criteria enabled. Table 1 shows the number of samples, the p_f, and F_1 score when the algorithm terminates. Appendix E.2 shows additional properties of the models upon termination. The hGP AK-MCS terminates with an accurate model in both experiments. For both experiments we see that GPC eventually learns an accurate model but does not terminate, because the maximum predicted misclassification probability remains high. In the toy function experiment the masked GP terminates far too early while the F_1 score is very low compared to hGP. The masked AK-MCS does not terminate for the AD experiment; the short length scale in the masked GP means that the predicted variance is large and a large amount of iterations will be required to significantly reduce this. For the AD experiment all models obtain the correct value of p_f within estimated error, however the F_1 score is much lower for the masked GP, suggesting the identified failure region is incorrectly located.

It is clear that due to the ability of the hGP to predict appropriate classification probabilities it is the only model which can reliably terminate when the model is sufficiently accurate.

Table 1: Comparison of p_f, F_1 score (F_1), and number of evaluations (N. Eval.) for all methodologies. N. Eval. includes the initial design of experiments (12 evaluations) and the number of iterations of the active learning algorithm. Number of iterations was capped at 150 and methods hitting the cap are marked with did not terminate (DNT). Mean and standard deviation for 5 repeats are shown.

Methodology	p_f Avg. (Std. Dev.)	F_1	N. Eval.	p_f	F_1	N. Eval.
	Toy function			Autonomous Driving		
Analytic	0.036906	N/A	N/A	0.0382	N/A	N/A
hGP (ours)	0.036 (0.0051)	1 (0.0011)	56 (3.4)	0.038 (0.0019)	1 (0.0013)	81 (21)
Masked GP	0.019 (0.0031)	0.66 (0.00013)	20 (3.2)	0.037 (0.0041)	0.92 (0.063)	DNT
GPC	0.037 (0.0026)	0.99 (0.003)	DNT	0.035 (0.0029)	0.98 (0.0049)	DNT

Acknowledgments and Disclosure of Funding

We gratefully acknowledge John Prater, Jamie McCallion, Tom Westmacott, and Iain Whiteside for valuable discussions at the inception of this project. We also wish to thank Majd Hawasly for spending his valuable time to provide comments on the work.

References

[1] Philip Koopman and Michael Wagner. Challenges in autonomous vehicle testing and validation. *SAE International Journal of Transportation Safety*, 4(1):15–24, 2016.

[2] B. Echard, N. Gayton, and M. Lemaire. AK-MCS: An active learning reliability method combining kriging and monte carlo simulation. *Structural Safety*, 33(2):145–154, 2011. ISSN 01674730. doi: 10.1016/j.strusafe.2011.01.002.

[3] Wanying Yun, Zhenzhou Lu, Yicheng Zhou, and Xian Jiang. AK-SYSi: an improved adaptive kriging model for system reliability analysis with multiple failure modes by a refined u learning function. *Structural and Multidisciplinary Optimization*, 59(1):263–278, 2019.

[4] Yu Inatsu, Shogo Iwazaki, and Ichiro Takeuchi. Active learning for distributionally robust level-set estimation. In *International Conference on Machine Learning*, pages 4574–4584. PMLR, 2021.

[5] Shogo Iwazaki, Yu Inatsu, and Ichiro Takeuchi. Bayesian quadrature optimization for probability threshold robustness measure. *Neural Computation*, 33(12):3413–3466, 2021.

[6] M Moustapha, S Marelli, and B Sudret. A generalized framework for active learning reliability: survey and benchmark. *arXiv preprint arXiv:2106.01713*, 2021.

[7] Chao Dang, Pengfei Wei, Matthias GR Faes, Marcos A Valdebenito, and Michael Beer. Parallel adaptive bayesian quadrature for rare event estimation. *Reliability Engineering & System Safety*, 225:108621, 2022. ISSN 0951-8320. doi: https://doi.org/10.1016/j.ress.2022.108621.

[8] S. M. Mahmud, Luis Ferreira, Md Hoque, and Ahmad Hojati. Application of proximal surrogate indicators for safety evaluation: A review of recent developments and research needs. *IATSS Research*, 41, 03 2017. doi: 10.1016/j.iatssr.2017.02.001.

[9] Demin Nalic, Tomislav Mihalj, Maximilian Baeumler, Matthias Lehmann, Arno Eichberger, and Stefan Bernsteiner. Scenario based testing of automated driving systems: A literature survey. In *FISITA web Congress*, 10 2020. doi: 10.46720/f2020-acm-096.

[10] Philip Koopman and Michael Wagner. Toward a framework for highly automated vehicle safety validation. *SAE Technical Paper, Tech. Rep*, 2018.

[11] Maliki Moustapha and Bruno Sudret. A two-stage surrogate modelling approach for the approximation of models with non-smooth outputs. In *UNCECOMP 2019 Proceedings of the 3rd International Conference on Uncertainty Quatification in Computational Sciences and Engineering*, pages 357–366, 2019.

[12] Hans-Georg Beyer and Bernhard Sendhoff. Robust optimization–a comprehensive survey. *Computer methods in applied mechanics and engineering*, 196(33-34):3190–3218, 2007.

[13] Jonathan Uesato, Ananya Kumar, Csaba Szepesvari, Tom Erez, Avraham Ruderman, Keith Anderson, Krishnamurthy Dj Dvijotham, Nicolas Heess, and Pushmeet Kohli. Rigorous agent evaluation: An adversarial approach to uncover catastrophic failures. In *International Conference on Learning Representations*, 2018.

[14] Ashish Kapoor, Kristen Grauman, Raquel Urtasun, and Trevor Darrell. Active learning with gaussian processes for object categorization. In *2007 IEEE 11th international conference on computer vision*, pages 1–8. IEEE, 2007.

[15] Robert E Melchers and André T Beck. *Structural reliability analysis and prediction*. John wiley & sons, 2018.

[16] Rui Teixeira, Maria Nogal, and Alan O'Connor. Adaptive approaches in metamodel-based reliability analysis: A review. *Structural Safety*, 89:102019, 2021.

[17] Halil Beglerovic, Michael Stolz, and Martin Horn. Testing of autonomous vehicles using surrogate models and stochastic optimization. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2017.

[18] Jingkang Wang, Ava Pun, James Tu, Sivabalan Manivasagam, Abbas Sadat, Sergio Casas, Mengye Ren, and Raquel Urtasun. AdvSim: Generating safety-critical scenarios for self-driving vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9909–9918, June 2021.

[19] Mark Koren, Saud Alsaif, Ritchie Lee, and Mykel J. Kochenderfer. Adaptive stress testing for autonomous vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–7, 2018. doi: 10.1109/IVS.2018.8500400.

[20] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning series. MIT Press, 2005. ISBN 9780262182539.

[21] GPy. GPy: A gaussian process framework in python. `http://github.com/SheffieldML/GPy`, since 2012.

[22] Andrei Paleyes, Mark Pullin, Maren Mahsereci, Cliff McCollum, Neil Lawrence, and Javier González. Emulation of physical processes with emukit. In *Second Workshop on Machine Learning and the Physical Sciences, NeurIPS*, 2019.

A Motivating example

Consider the case of an autonomous vehicle waiting to join a road at a T-junction, where other cars are travelling on the road at constant velocity. Assuming that the autonomous vehicle behaves deterministically with respect to variables like the initial starting positions and velocities of the vehicles, one could specify a measure of safe system performance, for example the distance of closest approach between the autonomous vehicle and the other vehicles as a function of these variables. Then an engineer could use an optimisation algorithm to find the conditions which cause the closest distance to become dangerously small. However, in some cases the closest distance function could be undefined because the autonomous vehicle never joined the road at all, for example if the autonomous vehicle's planner deems the situation to be unsafe. In scenario based testing of autonomous vehicles such rule functions, which have the potential to be only partially specified, are common [1, 8–10].

A.1 Mathematical Model

We consider a vehicle moving in the nearside lane at a constant velocity v_a, which has starting longitudinal position $x(t=0) = x_a < 0$ relative to the ego vehicle. The ego vehicle will perceive the adversarial vehicle if the distance between the two vehicles is less than the limiting distance for the sensor (x_{lim}), otherwise a false negative detection will occur. The scenario will be considered safe provided the distance between the vehicles in the same lane is no less than a threshold $d_{\text{threshold}}$, corresponding to the stopping distance of a typical vehicle. Ego will attempt to merge into the nearside lane and accelerate at its maximum velocity a_{ego} until it reaches v_a, however if ego perceives that this action will result in a collision due to the position and velocity of the adversarial vehicle then ego will not merge into the road and hence the longitudinal distance between the vehicles in the lane will be undefined. This is shown in Fig. 3.

Modelling each vehicle as a particle moving in a one dimensional space, we can write the rule numerically as

$$g(x_a, v_a) = \begin{cases} \text{NaN} & \text{if } d_{\text{min}}(x_a, v_a) < d_{\text{threshold}} \text{ and } |x_a| < x_{\text{lim}}, \\ d_{\text{min}}(x_a, v_a) - d_{\text{threshold}} & \text{otherwise}, \end{cases} \tag{7}$$

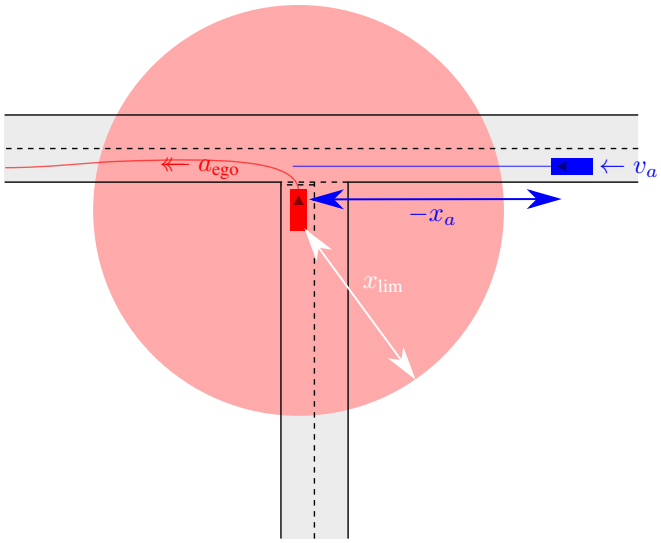
where we have defined the closest approach distance between the vehicles as

$$d_{\text{min}}(x_a, v_a) = \min_{t \in [0,\infty]} \left| \frac{1}{2} a_{\text{ego}} t^2 - (x_a + v_a t) \right| = \max\left(-\left(x_a + \frac{v_a^2}{2a_{\text{ego}}} \right), 0 \right). \tag{8}$$

A plot of the rule is shown in Fig. 4b. In order to ensure that the scale of the performance function is appropriate for the GP hyperparameters we have chosen, we rescale the performance function by dividing by 20 resulting in performance values of lower magnitude.

B Related Work

The PTR measure has recently become of interest in the robust optimisation literature [12], for example Inatsu et al. [4] show how system designs can be adjusted to optimise the measure. The measure has a much longer history in reliability engineering [15]. Moustapha et al. [6] and Teixeira et al. [16] provide reviews of active learning methods for estimating this measure. The Adaptive Kriging Monte Carlo methodology is perhaps the most well known of these methods, and achieves close to state of the art results [2, 3, 6]. Efficient methods of estimating the PTR measure also exist in reinforcement learning [13]. Beglerovic et al. [17] use a Bayesian optimisation approach to identify failure cases for an autonomous vehicle but do not exhaustively search for all $\boldsymbol{x}$ such that $f(\boldsymbol{x}) < 0$



Variable	Distribution / Value
x_a	$\mathcal{U}[-100\text{m}, 0\text{m}]$
v_a	$\mathcal{U}[10\text{m}, 15\text{m}]$
$d_{\text{threshold}}$	20 m
a_{ego}	2 ms^{-2}
x_{lim}	60 m

Figure 3: Depiction of our T-junction experiment. Left: Ego is shown in red merging into a road in a left hand traffic system where the adversarial car is shown in blue. The red circle represents the limits of the perception systems of ego. $d_{\text{threshold}}$, the smallest safe distance between ego and the adversarial car, is not shown. Right: Random variables and parameters.

and also do not calculate the PTR measure, p_f. Similarly, Wang et al. [18] use a realistic LiDAR simulator to modify real-world LiDAR data which can then be used to test end-to-end autonomous driving systems while searching for adversarial traffic scenarios with Bayesian Optimisation. A related problem in the autonomous vehicle space is finding the most likely $\boldsymbol{x}$, i.e. with largest $p(\boldsymbol{x})$, leading to $f(\boldsymbol{x}) < 0$ [19]. This is closely related to first order methods for estimating the PTR measure [15].

Although there exists literature related to Gaussian Process modelling for discontinuous targets [11], there is little literature on active learning specifically for the PTR measure for discontinuous targets.

C Hierarchical model details

In this section we provide additional details which describe the model proposed in Section 2 in further detail. Let $\boldsymbol{X}$ be a matrix containing all the $\boldsymbol{x}$ in $\mathcal{D}$ and Y be a vector containing all the y in $\mathcal{D}$. We assume that $p(y_*|\boldsymbol{x}_*, \mathcal{D}, y_* \neq \text{NaN})$ follows a Gaussian process, i.e. $p(y_*|\boldsymbol{x}_*, \mathcal{D}, y_* \neq \text{NaN}) = \mathcal{N}(\mu(\boldsymbol{x}_*), \sigma(\boldsymbol{x}_*))$ with

$$\mu(\boldsymbol{x}_*) = \boldsymbol{k}(\boldsymbol{x}_*, \boldsymbol{X})^\top (\boldsymbol{k}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2_{\text{noise}} \boldsymbol{I})^{-1} Y,$$
$$\sigma^2(\boldsymbol{x}_*) = \boldsymbol{k}(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{k}(\boldsymbol{x}_*, \boldsymbol{X})^\top (\boldsymbol{k}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2_{\text{noise}} \boldsymbol{I})^{-1} \boldsymbol{k}(\boldsymbol{X}, \boldsymbol{x}_*),$$

where $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive definite kernel with $0 < k(\boldsymbol{x}, \boldsymbol{x}) \leq 1$ for all $\boldsymbol{x} \in \mathcal{X}$, and $K(\boldsymbol{X}_1, \boldsymbol{X}_2)$ is matrix containing evaluations of the kernel at all points in $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, and σ_{noise} is a small positive constant which should be inversely proportional to how deterministic the evaluation of y is.

We assume that $p(y_* = \text{NaN}|\boldsymbol{x}_*, \mathcal{D})$ is given by a classification Gaussian process, i.e. $p(y_* = \text{NaN}|\boldsymbol{x}_*, \mathcal{D}) = \int \text{probit}(h) p(h|\boldsymbol{x}_*, \mathcal{D}) dh = p_{\text{nan}}(\boldsymbol{x}_*)$, where $p(h|\boldsymbol{x}_*, \mathcal{D})$ is the predictive distribution of a Gaussian process which can be calculated using the expectation propagation method as described in Rasmussen and Williams [20].

To calculate the predicted probability of misclassification for our model, recall that we are trying to define the classification boundary between the failure event $y_* < 0 \cap y_* \neq \text{NaN}$ and the complementary case. We classify this event based on $p(y_* < 0, y_* \neq \text{NaN}|\boldsymbol{x}_*, \mathcal{D}) = p(y_* < 0|\boldsymbol{x}_*, \mathcal{D}, y_* \neq \text{NaN}) p(y_* \neq \text{NaN}|\boldsymbol{x}_*, \mathcal{D}) > 0.5$. Therefore we calculate the predicted misclassification probability

as

$$p_{\text{misclassification}}(x_*) = \begin{cases} p(y_* < 0, y_* \neq \text{NaN}|\boldsymbol{x}_*, \mathcal{D}) & \text{if } p(y_* < 0, y_* \neq \text{NaN}|\boldsymbol{x}_*, \mathcal{D}) < 0.5, \\ 1 - p(y_* < 0, y_* \neq \text{NaN}|\boldsymbol{x}_*, \mathcal{D}) & \text{otherwise.} \end{cases} \tag{9}$$

where $p(y_* < 0, y_* \neq \text{NaN}|\boldsymbol{x}_*, \mathcal{D}) = \Phi\left(-\mu(\boldsymbol{x}_*)/\sigma(\boldsymbol{x}_*)\right) p(y_* \neq \text{NaN}|\boldsymbol{x}_*, \mathcal{D})$, where $\mu(\boldsymbol{x}_*)$ and $\sigma(\boldsymbol{x}_*)$, are the predicted mean and standard deviation of the regression Gaussian process, and Φ is the standard normal CDF.

We calculate the conditional failure probability for the regression Gaussian process using $p(y_* < 0|\boldsymbol{x}_*, \mathcal{D}, y_* \neq \text{NaN}) = \Phi\left(-\mu(\boldsymbol{x}_*)/\sigma(\boldsymbol{x}_*)\right)$. It follows when $p(y_* \neq \text{NaN}|\boldsymbol{x}_*, \mathcal{D}) = 1$, Eq. (9) reduces to the form in Echard et al. [2], i.e. $p_{\text{misclassification}}(\boldsymbol{x}_*) = \Phi\left(-|\mu(\boldsymbol{x}_*)|/\sigma(\boldsymbol{x}_*)\right)$.

In this paper we only consider systems with a single performance function, however Yun et al. [3] demonstrate how acquisition functions for multiple performance functions can be combined when one is interested in a combined PTR measure for the performance functions. This can be applied to our hierarchical model directly.

Finally, we note that our modifications to the AK-MCS algorithm are fairly general and only involve changing the performance function model and acquisition function, and therefore these changes could possibly also be used with different active learning algorithms. We do not explore these possible applications in this paper and instead leave this as a topic for future research.

D Experimental Hyperparameters

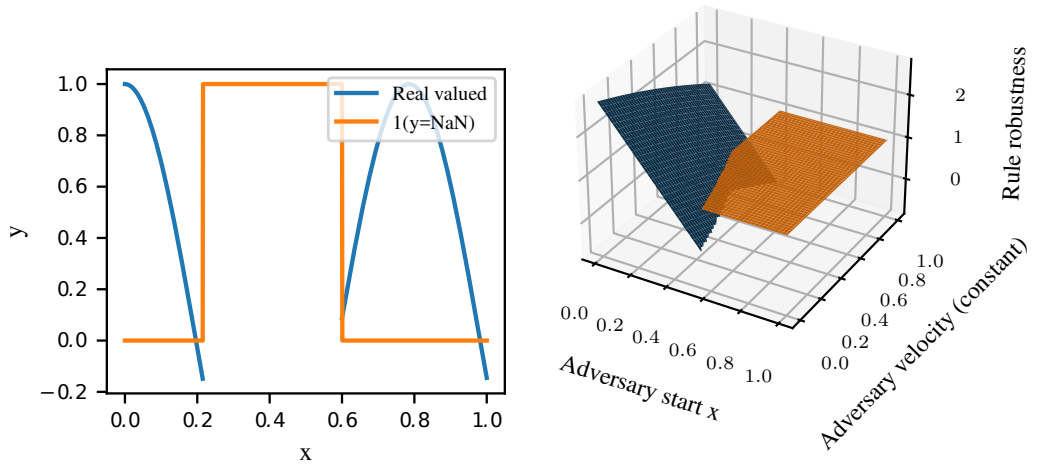
For the classification part of the hierarchical GP we use a Matern52 kernel and fix the variance to 10^5 as we find that this ensures a quick convergence in practice, and corresponds to a prior belief that the classification GP should model a deterministic function. For all regression GPs we use a Matern52 kernel with variance and lengthscale determined by optimisation on the training data with some relatively weak constraints: the lengthscale falls in $[0, 0.2]$ and variance falls in $[0.5, 1]$. For GPC we use a Matern52 kernel with variance 100 and length scale constrained within $[0, 0.2]$. For all GPs we set the likelihood variance to a small positive number (0.005^2), representing a belief that the system performance is deterministic. For AK-MCS we add $n_{mc} = 5 \times 10^3$ proposal samples when the maximum misclassification is below $\eta = 0.02$ and terminate the algorithm when the coefficient of variation is below 0.1, which are similar criteria to those used in Echard et al. [2]. In all experiments, the initial design of experiments is 12 samples. The Gaussian Process models were created using GPy [21], and Emukit was used for the active learning algorithms [22].

E Additional Experimental Results

E.1 Metrics

We use the following metrics, some of which were used in the original AK-MCS paper [2], and some of which we introduce specifically to measure aspects of performance related to our problem.

- Maximum predicted misclassification ($\max_{\boldsymbol{x} \in \mathcal{S}} p_{\text{misclassification}}(\boldsymbol{x})$) bounds the "risk" suffered over $p(\boldsymbol{x})$, and hence can be used to measure the convergence of the *internal* state of the algorithm, i.e. it does not require ground truth data. Used in Echard et al. [2].
- Failure probability (p_f) can be used to measure if the identified failure region has the correct size, by comparing to the p_f computed in some other way as ground truth (i.e. analytically). Used in Echard et al. [2].
- Coefficient of Variation (CoV) is used to assess the algorithms internal uncertainty in the estimated p_f, and can be calculated as described in Algorithm 1. Used in Echard et al. [2].
- F_1score can be used to check if the identified failure region is correctly located in the space of x. We introduce this metric because in reliability problems the failure probability is usually low and hence the class distribution is imbalanced, and F_1score is known to perform well in such cases. The F_1score is defined as $F_1 = \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$ where precision and recall are measured with a test set of 10^5 points, treating failure as the positive class and the safe region as the negative class. Used in Iwazaki et al. [5].



(a) System performance for toy function experiment (Eq. (4)).

(b) System performance for T-junction merging experiment (Eq. (5) with input features normalised). The blue surface shows the real valued performance and the orange surface shows the undefined performance masked with $\alpha = 1.0$.

Figure 4: Plots of ground truth performance functions.

- Average precision penalises an incorrectly located failure region in a similar way to F_1 score, however it has the additional advantage that the ranking of the predicted class scores is tested. Average precision is insensitive to correctly ranked but miscalibrated scores. It is also measured with a test set of 10^5 points.

E.2 Additional Figures

Fig. 5 and Fig. 6 show the convergence of the failure probability (p_f), maximum predicted misclassification probability, average precision and F_1 score for the models for the toy function and autonomous driving experiments. The ground truth performance function for the toy function this experiment is plotted in Fig. 4a. In Table 2 we show the number of samples required for each algorithm to terminate and the p_f, average precision and F_1 score when the algorithm terminates.

E.3 Comparing values of α for the Regression GP

In Fig. 7-8 we repeat the analysis from Section 3 to show the convergence of failure probability (p_f), average precision, maximum predicted misclassification probability and F_1 score for fitted masked Gaussian processes with different α. In comparison to Hierarchical GP, all masked GPs are similar. All models show slower convergence than Hierarchical GP. Table 3 shows the termination results for different values of α.

Table 2: Comparison of p_f, coefficient of variation (CoV), F_1score (F_1), Average Precision (AP) and number of evaluations (N. Eval.) for all methodologies. The total function evaluations includes the initial design of experiments (12 evaluations) and the number of iterations of the active learning algorithm. Number of iterations was capped at 150 and methods hitting the cap are marked with did not terminate (DNT). Mean and standard deviation for 5 repeats are shown.

Methodology	p_f Avg. (Std. Dev.)	CoV	F_1	AP	N. Eval.
	Toy function				
Analytic	0.036906	N/A	N/A	N/A	N/A
Hierarchical GP (ours)	0.036 (0.0051)	0.074 (0.0061)	1 (0.0011)	1 (6.1e-07)	56 (3.4)
Masked GP $\alpha = 1.0$	0.019 (0.0031)	0.096 (0.013)	0.66 (0.00013)	0.56 (0.041)	20 (3.2)
GPC	0.037 (0.0026)	0.073 (0.0027)	0.99 (0.003)	1 (0.00027)	DNT
	Autonomous Driving				
Analytic	0.0382	N/A	N/A	N/A	N/A
Hierarchical GP (ours)	0.038 (0.0019)	0.071 (0.0019)	1 (0.0013)	1 (1.7e-05)	81 (21)
Masked GP $\alpha = 1.0$	0.037 (0.0041)	0.073 (0.0044)	0.92 (0.063)	0.89 (0.097)	DNT
GPC	0.035 (0.0029)	0.074 (0.003)	0.98 (0.0049)	1 (0.0023)	DNT

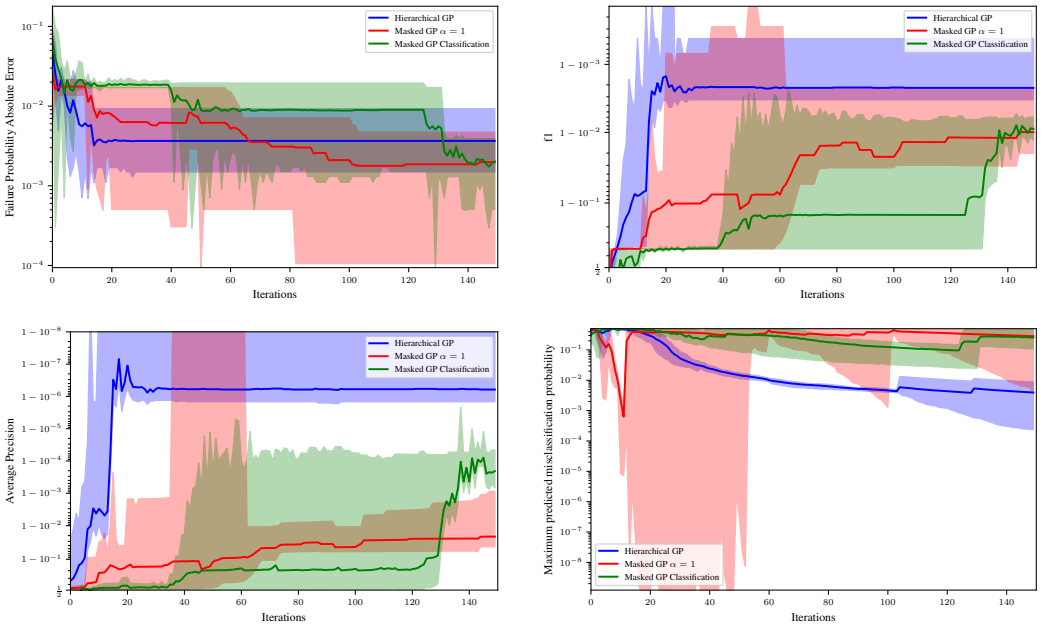


Figure 5: Convergence of failure probability (p_f), average precision, maximum predicted misclassification probability and F_1score for fitted Gaussian processes for the simple toy function. The shaded area represents the minimum and maximum of 5 repeats, and the dark line represents the mean. All models eventually obtain a p_f which is correct within the Monte Carlo error calculated using the CoV (approximately 0.0027).

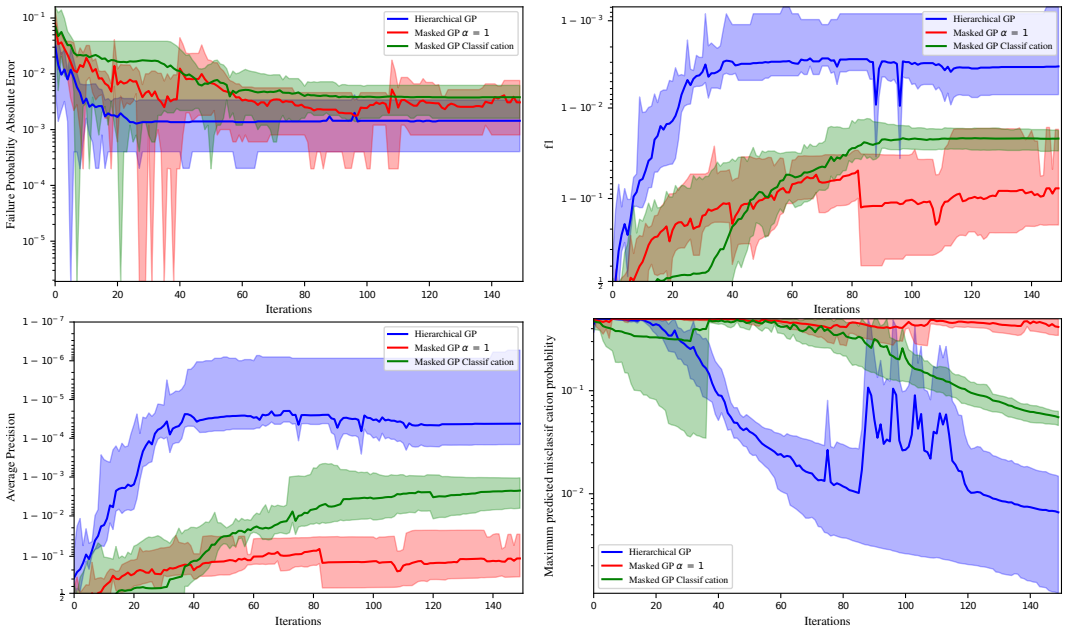


Figure 6: Convergence of failure probability (p_f), average precision, maximum predicted misclassification probability and F_1 score for fitted Gaussian processes for the autonomous driving experiment. The shaded area represents the minimum and maximum of 5 repeats, and the dark line represents the mean. All models eventually obtain a p_f which is correct within the Monte Carlo error calculated using the CoV (approximately 0.0027).

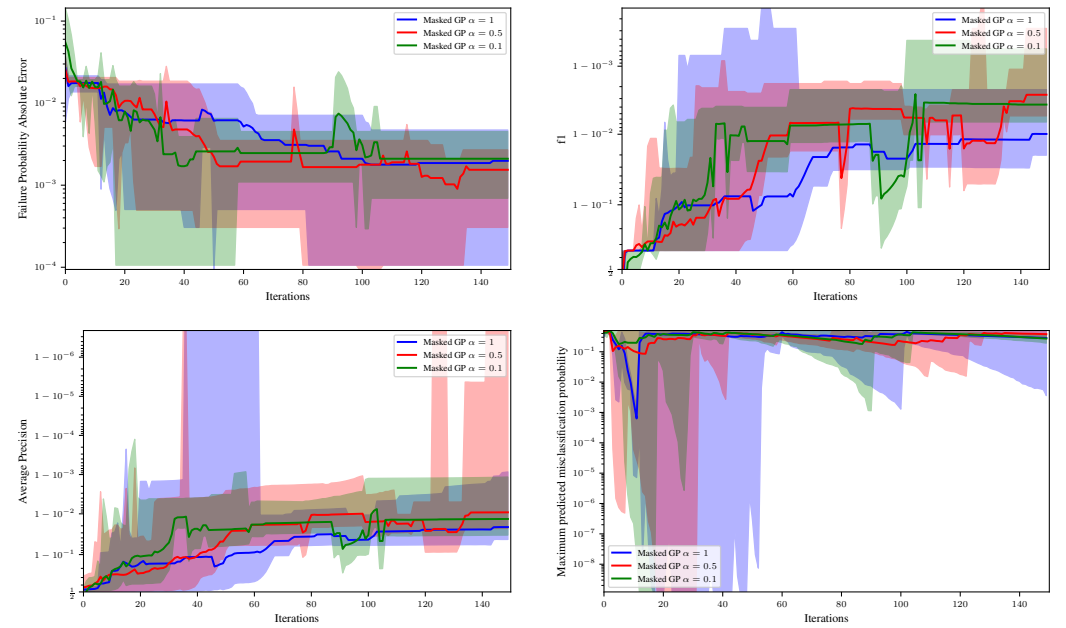


Figure 7: Convergence of failure probability (p_f), average precision, maximum predicted misclassification probability and F_1 score for fitted masked Gaussian processes with different α for the simple toy function. The shaded area represents the minimum and maximum of 5 repeats, and the dark line represents the mean. All models eventually obtain a p_f which is correct within the Monte Carlo error calculated using the CoV (approximately 0.0027).

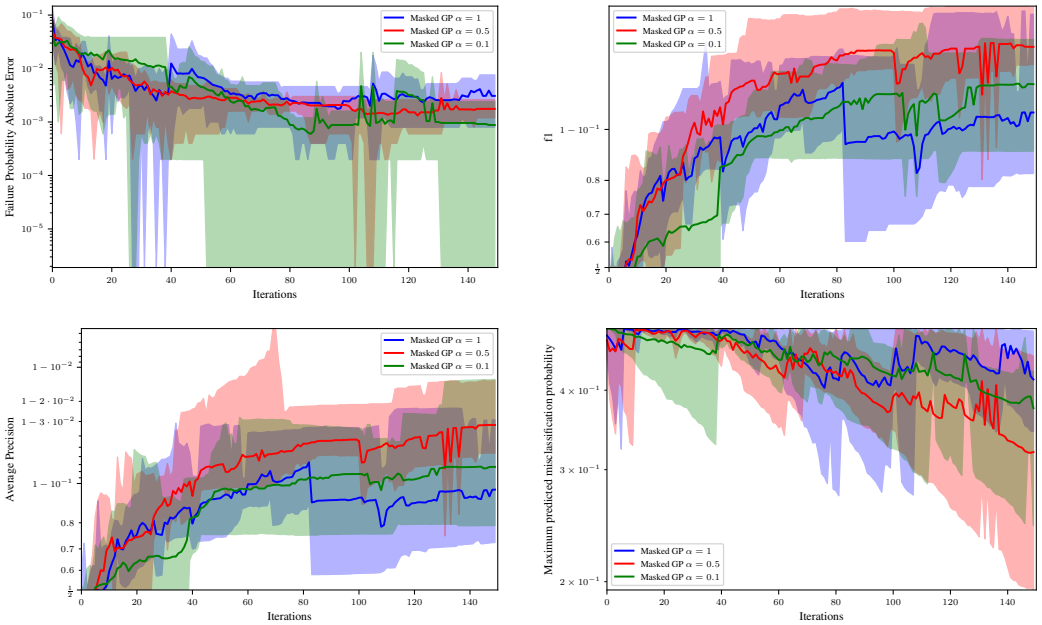


Figure 8: Convergence of failure probability (p_f), average precision, maximum predicted misclassification probability and F_1score for fitted masked Gaussian processes with different α for the autonomous driving experiment. The shaded area represents the minimum and maximum of 5 repeats, and the dark line represents the mean. All models eventually obtain a p_f which is correct within the Monte Carlo error calculated using the CoV (approximately 0.0027).

Table 3: Comparison of p_f, coefficient of variation (CoV), F_1score (F_1), Average Precision (AP) and number of evaluations (N. Eval.) for fitted masked Gaussian processes with different α. The total function evaluations includes the initial design of experiments (12 evaluations) and the number of iterations of the active learning algorithm. Number of iterations was capped at 150 and methods hitting the cap are marked with did not terminate (DNT). Mean and standard deviation for 5 repeats are shown.

Methodology	p_f Avg. (Std. Dev.)	CoV	F_1	AP	N. Eval.
			Toy function		
Masked GP $\alpha = 0.1$	0.025 (0.0089)	0.08 (0.0098)	0.79 (0.19)	0.71 (0.26)	75 (78)
Masked GP $\alpha = 0.5$	0.018 (0.0012)	0.086 (0.018)	0.66 (6.7e-05)	0.56 (0.055)	19 (4.9)
Masked GP $\alpha = 1.0$	0.019 (0.0031)	0.096 (0.013)	0.66 (0.00013)	0.56 (0.041)	20 (3.2)
			Autonomous Driving		
Masked GP $\alpha = 0.1$	0.038 (0.0014)	0.071 (0.0014)	0.95 (0.047)	0.93 (0.078)	DNT
Masked GP $\alpha = 0.5$	0.038 (0.0019)	0.072 (0.0019)	0.97 (0.0093)	0.97 (0.017)	DNT
Masked GP $\alpha = 1.0$	0.037 (0.0041)	0.073 (0.0044)	0.92 (0.063)	0.89 (0.097)	DNT