Technical Appendix

Table of contents

Technical Appendix A: Dataset Imbalance

window (ms)	Time-in-advance (ms)	train 0:1 ratio	test 0:1 ratio	training set total	test set total
100	300	64.005	65.991	514513	59488
	600	30.396	31.790	513546	60005
	800	22.269	24.897	509958	63137
	1000	17.676	18.626	512723	60037
	1500	11.627	10.788	518189	53752
500	300	62.483	62.216	502467	56136
	600	29.649	31.297	499699	58587
	800	22.067	21.819	502782	55336
	1000	17.433	17.352	502319	55514
	1500	11.408	11.217	501853	55048
1000	300	60.930	60.104	486464	53527
	600	29.351	27.046	489251	50538
	800	21.580	21.301	486079	53522
	1000	17.117	16.899	485999	53267
	1500	11.383	10.486	487477	50424
1500	300	59.805	57.319	470266	51787
	600	28.597	28.584	469557	52245
	800	21.354	20.463	471296	50202
	1000	17.080	15.790	472309	48724
	1500	11.340	11.415	467090	52129
2000	300	59.031	56.361	455634	48470
	600	28.533	27.736	454249	49484
	800	21.376	20.606	454501	48764
	1000	17.029	17.080	452248	50497
	1500	11.569	11.243	451596	48997

Table S1: Experimental settings and corresponding dataset statistics. 0: non-crash, 1: crash.

This section is referred to in Section 4.1 in the submitted paper, which discusses the preprocessing of the raw dataset.

In Table S1, we can see that the class imbalance exists in all window size and time-in-advance configurations. As time-in-advance shortens, the imbalance issue appears more severe, from 10.79 to as high as 65.99 in the test sets.

Again, to address this imbalance issues, we tried to use balanced class weights when designing our loss function, by assigning crash class a weight equal to the 0:1 imbalance ratio while non-crash samples a weight of 1. However, this approach did not results in improvement in model performance.

Technical Appendix B: AUC vs. P@0.95R

Models	Window Size	AUC at Time-In-Advance			P@0.95R at Time-In-Advance		
		300ms	600ms	1000ms	300ms	600ms	1000ms
MLP	500ms	0.9992 ± 0.0003	0.997 ± 0.0003	0.9748 ± 0.0017	0.7863 ± 0.0601	0.669 ± 0.0243	0.2901 ± 0.0145
	1000ms	0.9995 ± 0.0003	0.9969 ± 0.0004	0.9753 ± 0.0025	0.8598 ± 0.0531	0.6714 ± 0.0299	0.3099 ± 0.0117
	1500ms	0.9987 ± 0.001	0.9967 ± 0.0005	0.9751 ± 0.0021	0.7423 ± 0.1488	0.6579 ± 0.0371	0.3119 ± 0.0167
CNN	500ms	0.9995 ± 0.0002	0.9967 ± 0.0005	0.9734 ± 0.0025	0.8401 ± 0.0389	0.6545 ± 0.03	0.2872 ± 0.0171
	1000ms	0.9994 ± 0.0003	0.9967 ± 0.0008	0.9754 ± 0.0025	0.8225 ± 0.0534	0.6576 ± 0.0486	0.3047 ± 0.0132
	1500ms	0.9995 ± 0.0002	0.997 ± 0.0005	0.9778 ± 0.0014	0.8525 ± 0.0596	0.6758 ± 0.0395	0.3228 ± 0.014
LSTM	500ms	0.9998 ± 0.0001	0.9979 ± 0.0004	0.9778 ± 0.0023	0.9202 ± 0.036	0.7541 ± 0.0391	0.3171 ± 0.0162
	1000ms	0.9997 ± 0.0002	0.998 ± 0.0003	0.9794 ± 0.002	0.9038 ± 0.0599	0.7516 ± 0.0311	0.3434 ± 0.0249
	1500ms	0.9997 ± 0.0002	0.9977 ± 0.0004	0.9798 ± 0.0016	0.9108 ± 0.0563	0.7294 ± 0.0268	0.3424 ± 0.018
GRU	500ms	0.9998 ± 0.0002	0.9979 ± 0.0004	0.9781 ± 0.0024	0.9194 ± 0.0573	0.7505 ± 0.0326	0.322 ± 0.0227
	1000ms	0.9998 ± 0.0001	0.9979 ± 0.0004	0.979 ± 0.002	0.9144 ± 0.0332	0.7436 ± 0.0323	0.3405 ± 0.0217
	1500ms	0.9997 ± 0.0003	0.9974 ± 0.0004	0.9801 ± 0.0013	0.9062 ± 0.0589	0.7135 ± 0.0294	0.3435 ± 0.0161
stacked LSTM	500ms	0.9997 ± 0.0002	0.998 ± 0.0002	0.9786 ± 0.0017	0.898 ± 0.0785	0.7521 ± 0.0248	0.3259 ± 0.0172
	1000ms	0.9998 ± 0.0001	0.9981 ± 0.0004	0.9799 ± 0.002	0.9334 ± 0.0325	0.7618 ± 0.0399	0.3474 ± 0.0236
	1500ms	0.9998 ± 0.0001	0.9981 ± 0.0002	0.9803 ± 0.0012	0.9246 ± 0.0376	0.7688 ± 0.0132	0.3493 ± 0.0164
stacked GRU	500ms	0.9997 ± 0.0003	0.9982 ± 0.0002	0.979 ± 0.0017	0.897 ± 0.0658	0.7663 ± 0.0263	0.3283 ± 0.0145
	1000ms	0.9998 ± 0.0001	0.9982 ± 0.0003	0.9801 ± 0.0019	0.9202 ± 0.0498	0.7647 ± 0.0262	0.3477 ± 0.0184
	1500ms	0.9997 ± 0.0003	0.9979 ± 0.0003	0.9808 ± 0.0013	0.9042 ± 0.0673	0.7492 ± 0.0174	0.3581 ± 0.0187

Table S2: AUC and precision at 0.95 recall (P@0.95R) scores averaged over 10-fold cross-validation for various model type, window size, and time-in-advance combinations. Note that 600ms is the most practical time-in-advance.

This section is referred to in Section 5.1 in the submitted paper, which describes our approach to choosing the main evaluation metric.

In Table S2, we can see that all model types for the same time-in-advance duration have high AUC values of at least 0.973. On the other hand, when we evaluate the model using P@0.95R, which is the evaluation metric suitable for our application, the differences between different models within the same time-in-advance duration is more pronounced. This is one of the reasons why we favor using P@0.95R over AUC for evaluating how well our model will perform in application.

Technical Appendix C: Shorter Time-in-Advance Resulting in Higher Precision at Recall

Time-in-advance	P@0.85R	P@0.90R	P@0.95R	P@0.99R
300	0.9801 ± 0.0138	0.9588 ± 0.0289	0.9202 ± 0.0498	0.745 ± 0.2489
400	0.9811 ± 0.009	0.9611 ± 0.0203	0.9168 ± 0.0291	0.7816 ± 0.0542
500	0.9324 ± 0.0341	0.8868 ± 0.0445	0.8098 ± 0.0474	0.6305 ± 0.0405
600	0.918 ± 0.0231	0.8604 ± 0.0272	0.7647 ± 0.0262	0.5336 ± 0.0392
700	0.8274 ± 0.047	0.7556 ± 0.0456	0.6334 ± 0.0321	0.4018 ± 0.0382
800	0.7665 ± 0.0158	0.6772 ± 0.0169	0.5432 ± 0.0203	0.3103 ± 0.0185
900	0.6665 ± 0.014	0.5659 ± 0.0176	0.4253 ± 0.0244	0.2377 ± 0.0179
1000	0.5818 ± 0.0217	0.4829 ± 0.0212	0.3477 ± 0.0184	0.193 ± 0.0153
1100	0.5051 ± 0.0279	0.4096 ± 0.0248	0.2956 ± 0.019	0.175 ± 0.0207
1200	0.4578 ± 0.0207	0.3722 ± 0.0158	0.2677 ± 0.0163	0.1551 ± 0.0085

Table S3: precision at different recall values and time- in-advance durations, for stacked GRU at 1000ms window size.

This section is referred to in Section 6 in the submitted paper, where we discuss how our model could help prevent the crashes in our paradigm.

In Table S3, we observe that, at the same window size, shorter time-in-advance durations of 300-500ms result in much higher precision values at a recall as high as 0.95 than do the 500-1000ms durations; in this case, human may have trouble to react in this very short duration, but we could create an automated system informed by the crash prediction model to respond immediately to predicted crashes. Therefore, by having an automated system powered by a model with both high precision and high recall, we could prevent many more crashes from happening in our paradigm.