Rank Position Forecasting in Car Racing

Bo Peng[1] Jiayu Li[1] Selahattin Akkas[1]
Fugang Wang[1] Takuya Araki[2] Ohno Yoshiyuki[2] Judy Qiu[1]
[1]Indiana University
[2]NEC Corporation Japan
{pengb, jl145, sakkas, fuwang, xqiu}@indiana.edu
{takuya_araki, ohno.yoshiyuki}@nec.com

***Abstract*—**

I. Introduction

Deep learning based forecasting has been successfully applied in different domains, such as demand prediction [24], traffic prediction [25], clinical state progression prediction [28], epidemic forecasting [11], etc. It has the advantages over traditional statistical models that is good at learning complex models from multiple correlated high dimensional time series. Compared with other machine learning models, deep learning models are featured with powerful representation learning capability which relieves the dependency on the need of domain knowledge and the high cost of feature engineering.

However, when addressing the forecasting problem in the specific domain of motor sports, we found that the state-of-the-art models in this field are based on simulation methods or machine learning methods, all highly depend on the domain knowledge [1], [15], [20], [33]. Simply applying a deep learning model here does not deliver better forecasting performance.

The forecasting problem in motor sports is challenging. First, the status of the race is highly dynamic, which is the collective effect of many factors, including the skills of the drivers, the configuration of the cars, the interaction among the cars, the dynamics of the racing strategies and events out of control, such as mechanical failures and unfortunate crashes that are hardly avoidable during the high speed racing. **Uncertainty** resulted from exogenous factors is a critical obstacle for the model to accurately forecast future. A successful model needs to incorporate these cause effects and express the uncertainty. Secondly, motor sports forecasting is a sparse data problem, that available data are limited, because in each race only one trajectory for each car can be observed. Moreover, some factors, such as pit stop and crash, make huge influences to the race dynamic, but are irregular and rare which appear less than 5% in available data. Modeling these **extreme events** are critical part of a model.

In this work, taking IndyCar series [4] as a use case, we investigate these challenging issues in the forecasting problem, and explore the solutions based on deep learning models. We build car racing forecasting models based on deep encoder-decoder network architecture, model the uncertainty that is rooted from the dynamics of the racing, and achieve comparable forecasting performance compared with state-of-the-art machine learning models. The work also enables strategy optimization.

II. Problem Statement

A. Background

Indy500 is the premier event of IndyCar series. Each year, 33 cars compete on a 2.5 mile oval track for 200 laps. Fig. 1 shows the oval track setup. The track is split into several sections, or timeline. E.g., SF/SFP indicate the start and finish line on the track or the pit lane respectively. Sensors are buried under the track at the boundaries of the sections, collecting the timing information of the racing cars when they pass the position. A local communication network is employed to broadcast these collected information to all the teams, following a general data exchange protocol [4].

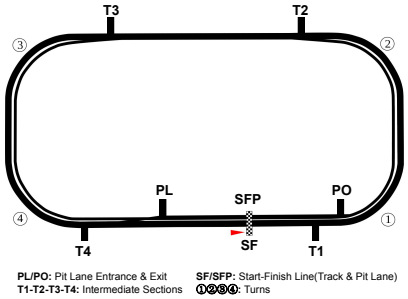


Fig. 1: Indy500 track setup [4].

Rank position is based on best time or race position, updated at each timeline during the race. The final race results are based on laps led and crossing order over SF/SFP. The start position of each car is assigned according to its performance in qualifying races. Before the beginning of a race, the cars proceed forward row by row on the track, calls a warm up period. When the race begins, all cars start to accelerate and the timing starts at the time the first car crossing SF. Later on, each car's rank position of a lap is calculated by its elapsed time that finishing the lap.

In motorsports, a **pit stop** is a pause for refuelling, new tyres, repairs, mechanical adjustments, a driver change, as a penalty, or any combination of the above [9]. As in Fig. 1, a car turns into inner track at T3, slows down, enters into pit lane between PL and PO. After refuel and tire changes, the car merges back to the main track at T2.

Unexpected events happen in a race, including mechanical failures or a crash. Depending to the serious level of the event,

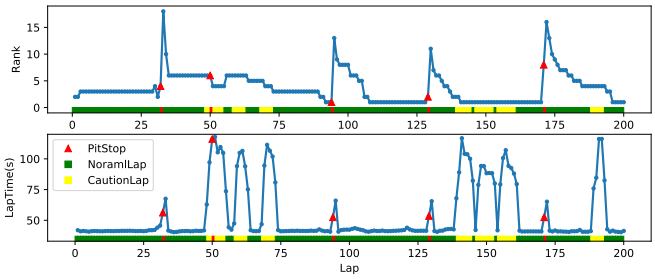
sometimes it leads to a dangerous situation for other cars to continue the racing with high speed on the track. In these cases, a full course yellow flag raises to indicate the race entering a **caution laps** mode, in which all the cars slow down and follow a safety car and can not overtake until an other green flag raised.

B. Rank position forecasting problem and challenges

Rank	CarId	Lap	LapTime	Time Behind Leader	Lap Status	Track Status
1	1	31	44.6091	0	T	G
2	12	31	45.6879	1.6026	T	G
3	21	31	43.3229	2.6397	T	G
...	...	...	...	...	...	...
31	32	49	114.6894	115.965	T	Y
33	33	46	429.0577	14.2668	P	Y

T : normal lap P : pitstop lap G : green flag Y : yellow flag/caution lap

(a) Rank can be calculated by LapTime and TimeBehindLeader. LapStatus and TrackStatus indicate racing status of pitstops and cuation laps.



(b) Rank and LapTime sequence of car12, the final winner. Sequence dynamics correlate to racing status.

Fig. 2: Data examples of Indy500-2018.

The task of rank position forecasting is to predict the future rank position of a car given all observed history of the race. Fig.2(a) shows the data collected by the sensors through the on-premises communication network, the training data used in this work. Fig.2(b) shows a typical Rank and LapTime sequence. Both of them are stable in most of time, indicating the predictable aspects of the race that the performance of the driver is stable. However, they both show abruptly changes when the **racing status**, including LapStatus and TrackStatus, changes. Pit stops slow down the car and lead to lose of rank position temporarily in the next few laps. Caution laps also slow down the car but does not affect rank position apparently.

Fig.2(b) demonstrates the characteristic of highly dynamic of this problem. *The data sequence contains different phases, affected by the racing status.* As for pit stop decisions, a team will have a initial plan for pit stops before the race, and the coach of the team will adjust it dynamically according to the status of the race. 'Random' events, such as mechanical failures and crashes, also make impacts the decision. Therefore, even the cause effect between pit stop and rank position is known, forecasting of rank is still challenging due to the uncertainty in pit stop events. A few laps of adjustment to the pit stop strategy may change the whole course of the race. However, when assuming the pit stop on each lap is a random variable, only one realization of its distribution is observed in one race.

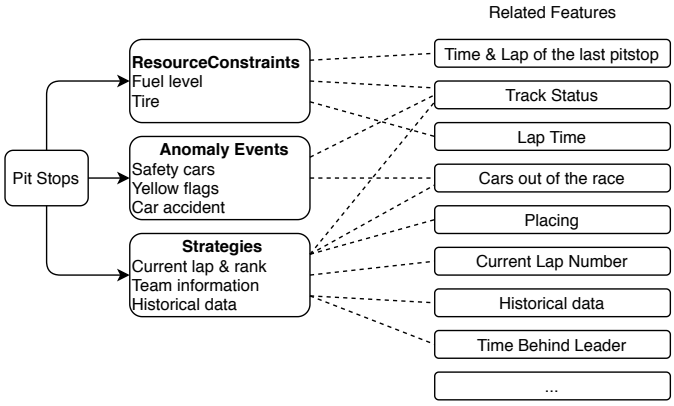


Fig. 3: The main factors affecting Pit stop and their corresponding features.

These challenges, if not addressed accordingly, will hurt the forecasting performance of the model. Fig.5 illustrates the limitations of typical existing methods.

C. Real-time leading car prediction

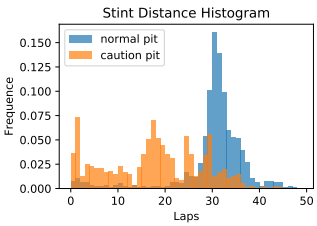
This project aims to predict the leading car in the future through telemetry data generated in real time during the race. In the Indy500 race and other events, each car is equipped with multiple sensors. These sensors will record the car's position, speed, acceleration and other information in real time. After some preprocessing, this information can be transformed into a multivariate time series. We will use LSTM to predict these time series, so as to achieve the purpose of real-time predicting the game. Given a prediction step t_p, and a time point t_0 in the game, we predict the following two events: a) Whether the currently leading car continue to lead at time $t_0 + t_p$. b): Which car is the leading car at time $t_0 + t_p$.

III. Models

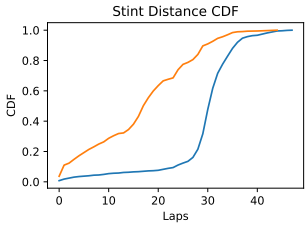
A. Pitstop factor analysis and feature selection

According to Figure 2, we see that the timing of Pit stop has a great influence on the ranking in car racing. Previous studies [16], [21], [33] did some preliminary analysis of the factors that affect pit stop. In this section, we will deeply study the causes of Pit stop based on the data of Indy500, which will help us select the main features and build a deep learning model. We divide the causes of Pit Stop into three categories: **resource constraints**, **anomaly events**, and **human strategies**.

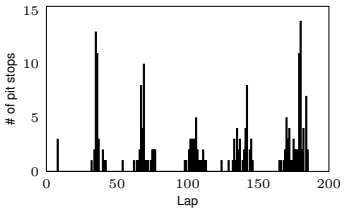
a) Resource constraints: The distance between the two pit stops is limited by the car's fuel tank volume and the car's tires. Therefore, the car must enter the pit stop every distance (30-40 laps). Fig. 4 (a) shows this rule. This type of pit stop is called a normal pit stop.



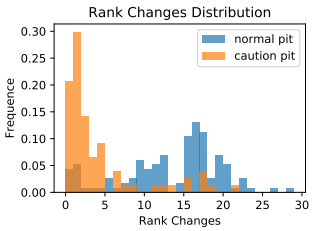
(a) Stint (laps between two consecutive pit stops) Distance



(b) Stint Distance CDF

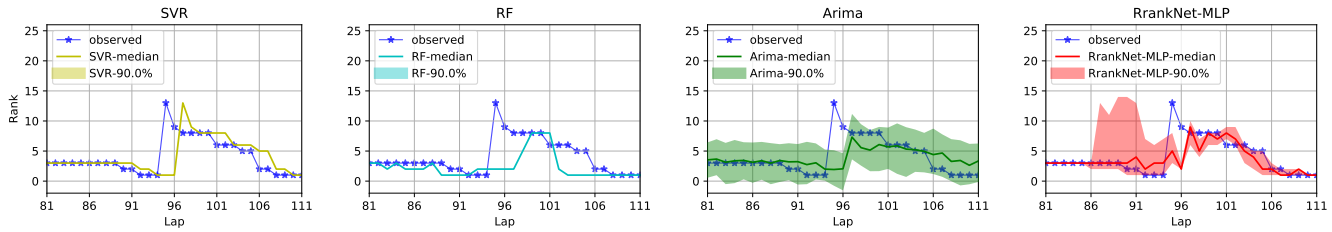


(c) Pit Laps Distribution

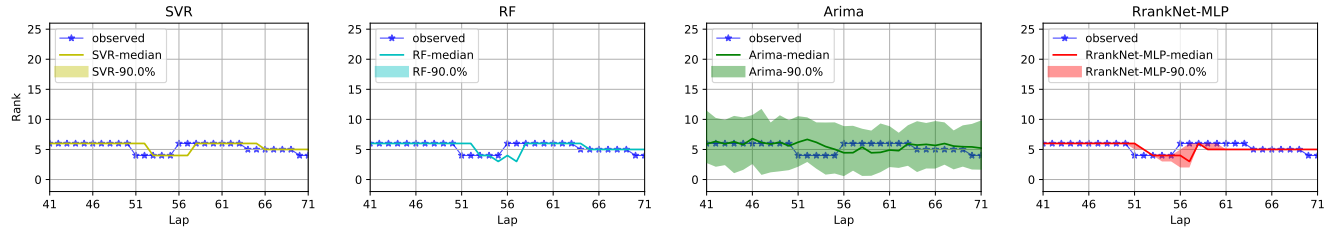


(d) Distribution of the rank position changes before and after a pit stop. Caution pits has much less impacts on rank position compared with normal pits.

Fig. 4: Statistics and analysis of the cause of pit stop.



(a) Pit stop at lap 94 (normal pit).



(b) Pit stop at lap 50 (caution pit).

Fig. 5: Illustration of the limitation of existing methods. Forecasting results of two laps in the future for car12. Pit stop at lap 94. (a)(b)Machine learning models. SVM overfits to the normal laps(laps before 90). RandomForest fails to predict the change around pit stop. (c)Statistical methods. ARIMA provides uncertainty predictions but lower performance, with difficulty to model the highly dynamics in the sequence. (d)Proposed method. RankNet accurately captures the data pattern in normal laps and successes in predicting the dynamics with uncertainty around pit stop.

b) Anomaly events: Anomaly events are usually caused by mechanical failure or car accidents. When a serious accident occurs, $TrackStatus$ will change to Yellow Flag, which will change the strategy of pit stop. Fig. 4 shows the statistics on the lap distance between two pit stops. In the Indy500 dataset, the number of normal pit and caution pit are close, 777 and 763 respectively. These two type of pit stops show significant differences. In Fig. 4(b), normal pit is a bell curve and caution pit scatter more evenly; In Fig. 4(b), the CDF curve shows the lap distance of normal pit can be split into three sections, roughly from [0-23,24-40,40-]. The lower section of short distance pit may caused mainly by unexpected reason, such as mechanism failures, and keeps a low probability less than 10%. The upper part of long distance pit are mainly observed when lot of caution laps happen, in which case the cars run at a reduced speed that results in greatly reduced tyre wear and fuel burn for a distance travelled. From these observations, modeling pit stop on the raw pit data could be challenging and modeling on the normal pit data plus removing the short distance section are more stable.

c) Human strategies: In actual competitions, participants also need to make decisions based on information such as the progress of the competition and the current ranking. In most cases, human strategies are very delicate and difficult to summarize with simple rules. One of the more obvious examples is as follows: Fig 4(c) shows the 2018 Pit Laps Distribution. Most racers choose to enter Pit stop on lap 170. This is because the total length of the game is 200 laps, and most contestants plan to start sprinting from lap 171. In order to better understand human strategy, we need to combine the ranking, team information, and historical data of past races to train the model.

TABLE I: Summary of features used in the model

Feature	Domain	Description	Source
$LapTime(N, L)$	$\mathbb{R}+$	The time it took for car # N to complete lap L.	[3]
$LapDistance(N, T)$	$\mathbb{R}+$	The distance of car # N from the starting line at time T. (Available for year 2017 - 2018)	
$PitstopLap$	$\mathbb{N}$(List)	A list of the laps where car # N entered the pitstop.	[3]
$Rank(N, L)$	$\mathbb{N}$	There are $Rank(N, L)$ cars that completed lap L before car # N	[2]
$Placing(N, T)$	$\mathbb{N}$	At time t, there are $Placing(N, t)$ cars in front of car N.	[3]
$TrackStatus(N, L)$	$\{0, 1\}$	Status of each lap for a car #N, normal lap or caution lap.	[2]
$TimeBehindLeader(N, L)$	$\mathbb{R}+$	Time behind the leader of car #N in lap #L.	[2]
$CautionLaps(N, L)$	$\mathbb{N}$	At Lap L, the count of caution laps since the last Pit Lap of car N.	
$PitAge(N, L)$	$\mathbb{N}$	At lap L, the count of laps after the previous pit stop of car N.	
CarID	$\{0, 1\}^{33}$	Categorical car ID	

B. Modeling uncertainty in high dynamic sequences

We treat the rank position forecasting as a sequence-to-sequence modeling problem with the assumption that there are enough information contained in the history to forecast the future. An encoder-decoder architecture is employed to map a input sequence $z_{i,1:t}$ to the output sequence $z_{i,t+1:t+k}$. To modeling the uncertainty, we follow the idea proposed in [29] to deliver probabilistic forecasting. Instead of predicting value of target variable in the output sequence directly, a probabilistic forecasting network predicts all parameters θ (e.g., mean and variance) of the probability distribution. A fixed distribution $p(z_{i,t}|\theta_{i,t})$ is parameterized by the output of the network $h_{i,t}$.

We use $z_{i,t}$ to denote the value of time series i at time t, $\mathbf{x}_{i,t}$ to represent the covariate that are assumed to be known at any given time. Our goal is to model the conditional distribution

$$P(z_{i,t_0:T}|z_{i,1:t_0-1}, \mathbf{x}_{i,1:T})$$

We assume that our model distribution $Q_\Theta(\mathbf{z}_{i,t_0:T}|z_{i,1:t_0-1}, \mathbf{x}_{i,1:T})$ consists of a product of likelihood factors

$$\begin{aligned} Q_\Theta(\mathbf{z}_{i,t_0:T}|\mathbf{z}_{i,1:t_0-1}, \mathbf{x}_{i,1:T}) &= \prod_{t=t_0}^{T} Q_\Theta(z_{i,t}|z_{i,1:t-1}, \mathbf{x}_{i,1:T}) \\ &= \prod_{t=t_0}^{T} p(z_{i,t}|\theta(\mathbf{h}_{i,t}, \Theta)) \end{aligned} \tag{1}$$

parametrized by the output $h_{i,t}$ of an autoregressive recurrent network

$$\mathbf{h}_{i,t} = h(\mathbf{h}_{i,t-1}, z_{i,t-1}, \mathbf{x}_{i,t}, \Theta)$$

where h is a function that is implemented by a multi-layer recurrent neural network with LSTM cells parametrized by Θ.

a) Training: The training process is divided into the following steps:

1) At each time step t, input the covariate $\mathbf{x}_{i,t}$, the value of the previous time step $z_{i,t-1}$, and the state $\mathbf{h}_{i,t-1}$ of the previous time step. Calculate the current state $\mathbf{h}_{i,t} = h(\mathbf{h}_{i,t-1}, z_{i,t-1}, \mathbf{x}_{i,t})$ through the neural network.
2) Calculate the parameter $\theta_{i,t} = \theta(\mathbf{h}_{i,t})$ of the likelihood $p(z|\theta)$.
3) Maximize the log-likelihood:

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{t=t_0}^{T} \log p(z_{i,t}|\theta(\mathbf{h}_{i,t}))$$

b) Prediction: After the training is completed, historical data is fed into the network to obtain the initial state. Then ancestral sampling is used to obtain the prediction results. The prediction phase is divided into the following steps

1) As shown in Fig. 5(b), an independent PitModel is used to predict the covariates $\mathbf{x}_{t+1:t+k}$.
2) Input the historical data at time step $t < t_0$ into the network to obtain the initial state $\mathbf{h}_{i,t_0-1}$.
3) For time steps $t, t+1, ...T$, randomly sample at each time step to obtain sample $\tilde{z}_{i,t} \sim p(\cdot|\theta_{i,t})$. E.g., for a Gaussian distribution,

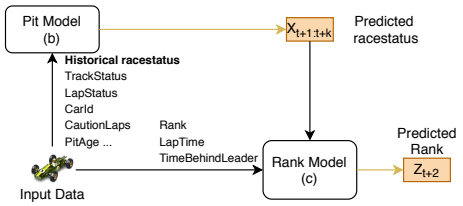
$$p(z|\mu, \sigma) = (2\pi\sigma^2)^{-1/2} \exp(-(z-\mu)^2/(2\sigma^2)) \tag{2}$$

This sampled value is used as the input for the next time step.
4) Repeat step 3 to get a series of z samples. These sampled values can be used to calculate the desired target values, such as quantiles, expectations, etc.

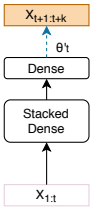
Encoder-decoder architecture provides an advantage by supporting to incorporate covariates known in the forecasting period. For example, in sales demand forecasting, holidays are known to be important factors to achieve good predictions. These variables can be expressed as covariates inputs into both the encoder network and decoder network. As we know, caution laps and pit stops are important factors to the rank position, therefore, can be considered as covariate inputs. But, different from the holidays, these variables in the future are unknown at the time of forecasting, leading to the need of decomposing the cause effects in building the model.

C. Modeling extreme events and cause effects decomposition

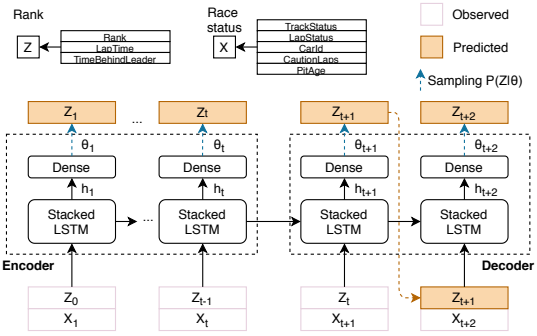
Changes of race status, including pit stops and caution laps, cause the phase changes of the rank position sequence. As a direct solution to address this cause effect, we can model the race status and rank position together and joint train the model in



(a) Process of forecasting. History data first feed into PitModel to get RaceStatus in the future, then feed into RankModel to get Rank forecasting. The output of the models are samples drawed from the learned distribution.



(b) PitModel is a MLP predicting next pit stop lap given features of RaceStatus history.



(c) RankModel is stacked 2-layers LSTM encoder-decoder predicting rank for next prediction_len laps, given features of historical Rank and RaceStatus, and future RaceStatus predicted by PitModel.

Fig. 6: RankNet architecture.

TABLE II: Dataset statistics and model parameters

# of time series	231
Granularity	Lap
Domain	$\mathbb{R}+$
Encoder length	40
Decoder length	2
# of training examples	32K
Item input embedding dimension	57
Item output embedding dimension	30
Batch size	32
Optimizer	ADAM
Learning rate	1e-3
# of lstm layers	2
# of lstm nodes	40
Running time	2h

the encoder-decoder network. In this case, target variable $z_{i,t}$ is a multivariate vector $[Rank, LapStatus, TrackStatus]$. However, this method fails in practice due to data sparsity. The changes of race status are rare events, and targets of rare events require different complexity of models. For example, on average a car goes to pit stop for six times in a race. Therefore, LapStatus, a binary vector with length equals to 200, contains only six ones, 3% effect data. When a long length of context for rank position encoder is necessary, pit stops are more like a first order Markov chain, i.e., the next pitstop mainly depends on the previous one rather than the ones further in the history. TrackStatus, indicating the crash events, is even harder to predict.

We propose to decompose the cause effect of race status and rank position in the model. RankNet, as shown in Fig. 6(a), is composed with two sub-models. First, a PitModel forecasts the future RaceStatus, in which LapStatus is predicted and TrackStatus is set to zeros assuming no caution laps in the future. Then the RankModel forecasts the future Rank sequence.

Target variable Z_t is not limited to Rank, it can be any variable that enables the calculation of the rank position, including: the rank position(Rank), time spend for a lap(LapTime), time behind the leader in a lap(TimeBehindLeader). Rank, as observed, is discrete variable which means its value lost subtle performance differences among the cars. LapTime can accurately reflect the performance of the race, but needs to accumulate from the first lap into elapsed time when calculate the rank position, and may suffer from accumulating error. The last one, TimeBehindLeader, indicates rank directly and contains correlated information across the cars.

RaceStatus is the most important feature in covariates X_t. TrackStatus indicates whether the current lap is a caution lap, in which the car follows a safety car in controlled speed. LapStatus indicates whether the current lap is a pit stop lap, in which the car cross SF/SFP in the pit lane. Some other static features can also be added into the input. For example, CarId represents the skill level of the driver and performance of the car.

Transformations are applied on these basic features to generate new features. Embedding for categorical CarId is utilized. Accumulation sum transforms the binary status features into 'age' features, generating features such as CautionLapsand PitAge. Table I summarizes the definition of these features. For efficiency, instead of sequences input and output, PitModel in Fig.6(b) use CautionLaps and PitAge as input, and output a scalar of the lap number of the next pit stop.

A rank position forecasting network is trained with a fixed prediction length. In order to deliver a variable length prediction, e.g., in predicting the rank positions between two pit stops, we apply a fixed length forecasting regressively by using previous output as input for the next prediction.

IV. Experiments

A. Dataset

We evaluate our model on the car racing data of IndyCar series [2]. The timing and score log data covers all the race events of IndyCar series from 2008 to 2020. Take 2018 as an example, which contains 17 events, six of them are racing on the oval speedway.

One of the challenges in forecasting car racing is data scarcity, because only one trajectory of each car was observed in each race. We have to incorporate more similar data to learn

TABLE III: Short-term rank position forecasting(prediction leghth=2)

Dataset	Indy500-2018				Indy500-2019			
Model	Top1Acc	MAE	50-Risk	90-Risk	Top1Acc	MAE	50-Risk	90-Risk
CurRank	0.72	1.34	0.097	0.097	0.73	1.16	0.080	0.080
ARIMA	0.68	2.63	0.097	0.087	0.57	2.25	0.082	0.075
RandomForest	0.51	1.75	0.127	0.127	0.62	1.33	0.092	0.092
SVM	0.72	1.34	0.097	0.097	0.73	1.18	0.080	0.080
XGBoost	0.46	1.63	0.118	0.118	0.64	1.25	0.086	0.086
DeepAR	0.66	2.07	0.156	0.096	0.59	1.71	0.121	0.075
RankNet-Joint	0.73	1.75	0.140	0.086	0.68	1.63	0.116	0.073
RankNet-MLP	**0.77**	**1.24**	**0.086**	**0.077**	**0.78**	**1.07**	**0.072**	**0.061**
RankNet-Oracle	0.85	1.11	0.080	0.073	0.86	0.98	0.067	0.061

TABLE IV: Short-term rank position forecasting(prediction leghth=2) of Indy500-2019

Dataset	Normal lap				Lap with events			
Model	SignAcc	MAE	50-Risk	90-Risk	SignAcc	MAE	50-Risk	90-Risk
CurRank	0.95	0.11	0.01	0.01	0.60	1.86	0.13	0.13
RandomForest	0.80	0.38	0.03	0.03	0.51	1.93	0.13	0.13
SVM	0.95	0.11	0.01	0.01	0.59	1.86	0.13	0.13
XGBoost	0.79	0.23	0.02	0.02	0.55	1.92	0.13	0.13
RankNet-MLP	0.93	0.17	0.01	0.01	**0.67**	**1.67**	**0.11**	**0.09**
RankNet-Oracle	0.93	0.17	0.01	0.01	0.81	1.50	0.10	0.09

TABLE V: Rank position changes forecasting between pit stops

Dataset	Indy500-2018				Indy500-2019			
Model	SignAcc	MAE	50-Risk	90-Risk	SignAcc	MAE	50-Risk	90-Risk
CurRank	0.10	4.15	0.259	0.234	0.15	4.33	0.280	0.262
RandomForest	0.57	3.55	0.222	0.221	0.51	4.31	0.277	0.276
SVM	0.61	3.42	0.214	0.212	0.51	4.22	0.270	0.249
XGBoost	0.49	4.10	0.256	0.234	0.45	4.86	0.313	0.304
RankNet-MLP	**0.68**	**3.83**	**0.240**	**0.163**	**0.62**	**4.33**	**0.286**	**0.223**
RankNet-Oracle	0.71	3.28	0.207	0.189	0.66	3.62	0.234	0.215

a stable model. Using the historical data that are too long time ago can be inefficient because many factors changes along the time, including the skills of the drivers, configurations of the cars and even the rules of the race. The same year data of other races are 'similar' in the status of the drivers, cars and rules, but different length of the track leads to different racing dynamics. In this paper, we use the data of the same event, Indy500, from 2013 to 2019, with the data from 2013 to 2017 as training set and the other two years as test set.

B. Implementation and baselines

We build our model RankNet with the Gluonts framework [13]. Gluonts is a library for deep-learning-based time series modeling, enables fast prototype and evaluation. RankNet is based on the DeepAR [29] implementation in Gluonts, and share the same features, including: sharing parameters between encoder and decoder, encoder implemented as a stacking of two lstm layers(40 neurons by default), training with ADAM(learing rate start from 1e-3, end at 1e-5).

First, we have a naive baseline which assumes that the rank positions will not change in the future, denoted as CurRank. Secondly, We implemented machine learning models as baselines that follow the ideas in [32] which forecast change of rank position between two consecutive pit stops. As far as we know, there is no open source model that forecast rank position in car racing, and no related work on IndyCar series.

C. Evaluation

RankNet is a single model that able to forecast both short-term rank position(TaskA) and long term change of rank position between pitstops(TaskB). First, MAE/RMSE are general metrics to evaluate average accuracy of all the predictions, used for both tasks. Secondly, TaskA evaluates the accuracy of correct predictions of the leader, denoted as Top1Acc. TaskB evalutes the accuracy of correct predictions of the sign of the change which indicating whether a car achieves a better rank position or not, denoted as SignAcc.

Thirdly, a quantile based error metric ρ-risk [30] is used to evaluate the performance of a probabilistic forecasting. When a set of samples output by a model, the quantile ρ value of the samples is obtained, denoted as $\hat{Z}_\rho$, then ρ-risk is defined as $2(\hat{Z}_\rho - Z)((Z < \hat{Z}_\rho) - \rho)$, normalized by $\sum Z_i$. It quantifies the accuracy of a quantile ρ of the forecasting distribution.

D. Short-term rank position forecasting

Table III shows the evaluation results of four models in a two laps rank position forecasting task. CurRank is one baseline that predicts future with the current rank position. RandomForest, SVM and XGBoost are popular machine regression models that do point wise forecast. Detailed features are presented in appendix VII-A. DeepAR is the deep learning model using the same network architecture without

TrackStatus and LapStatus covariates, representing the state-of-the-art time series modeling approach. RankNet-Joint is the model that train target with pit stop jointly without decomposition. RankNet-Oracle is the encoder-decoder network with ground truth TrackStatus and LapStatus as covariates input. It represents the best performance that can be obtained from RankNet given the caution and pit stop information for a race. RankNet-MLP is the forecasting model with a separate MLP pit stop model. It forecasts rank position via forecasting pit stops first and use the result as covariate input for rank position forecasting.

As in Table III, CurRank demonstrates good performance. 72% leader prediction correct and 1.34 mean absolute error on Indy500-2018 indicates that the rank position does not change much within two laps. DeepAR is a powerful model but fails to exceed CurRank, which reflects the difficulty of this task that the pattern of the rank position variations are not easy to learn from the history. Other machine learning models, and RankNet-Joint all failed to get better accuracy than CurRank. By contrast, RankNet-Oracle achieves significant better performance than CurRank, with 19% better in Top1Acc and 17% better in MAE. RankNet-MLP, our proposed model, is not as good as RankNet-Oracle, but still able to exceed CurRank more than 7% in both Top1Acc and MAE. It also achieves more than 20% improvement of accuracy on 90-risk when probabilistic forecasting get considered. Detailed comparsion are presented in apeedix VII-B.

E. Stint rank position forecasting

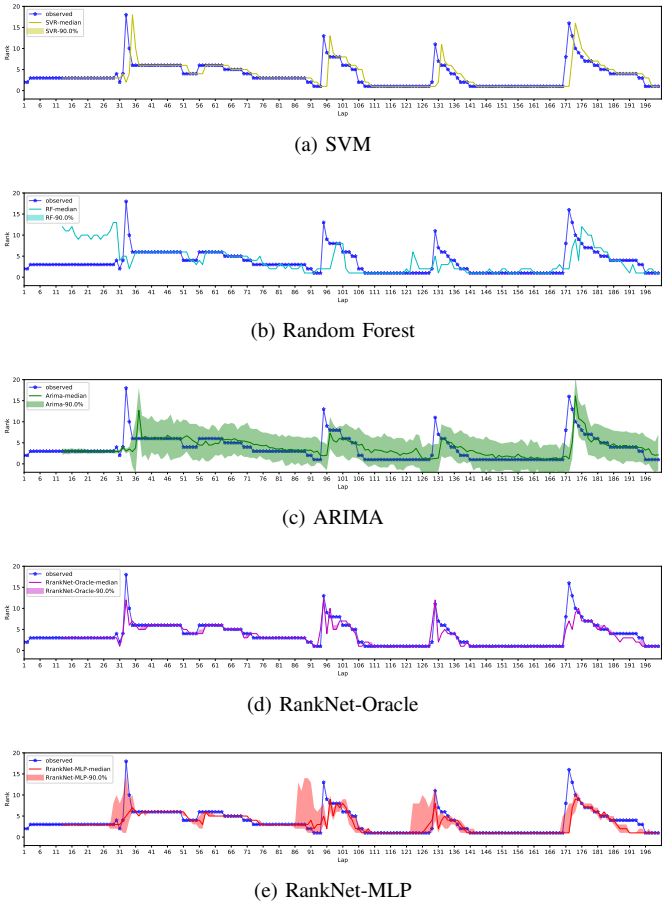
Table V shows the results on TaskB, forecasting the rank position changes between consecutive pit stops. CurRank can not predict changes, thus gets the worst performance. Among the three machine learning models, SVM shows the best performance. RankNet-Oracle demonstrates its advantages over all the machine learning models, indicating that once the pit stop information is known, long term forecasting through RankNet is more effective. Performance of RankNet-MLP obtains significant better accuracy, more than 10%, on SignAcc and 90-Risk over the machine learning models, while it has a worse MAE and 50-Risk, also more than 10%. But it forecasts future pit stops and thus different possibilities of race status, which are not supported by the other baselines. RankNet is promising to be used as a tool to investigate and optimize pit stop strategy. Detailed comparison are presented in appendix VII-C.

F. LSTM for real-time leading car prediction

We analyzed the factors affecting pit stop in Section III.A. Based on the above analysis, our training / prediction process is divided into the following three steps:

a) Stream data interpolation: There are two main sources of data: One is the game record from 2013 to 2019. The other is telemetry data for 2017 and 2018. The race record only includes the time spent in each section, and does not include the precise location of every two cars at a certain point in time. Telemetry data is a high-resolution data, each car will produce about 7 records per second, we can use telemetry data to estimate the position of each car at any time. In order to expand the training data set, we used interpolation to convert ordinary race records into a time series of car positions. If we assume that the speed of the car within each section does not change, then the position of the car at time T can be calculated as follows:$LapDistance(T) = L\frac{T-T_1}{T_2-T_1}$. T_1 and T_2 are the start and end time of the current section. L is the length of the section.

b) Data preprocessing: The preprocessing mainly includes two operations. 1) Data normalization, scale the input



(a) SVM

(b) Random Forest

(c) ARIMA

(d) RankNet-Oracle

(e) RankNet-MLP

Fig. 7: Rank position prediction result comparison. Forecasting results of two laps in the future for car12.

TABLE VI: Leading Car prediction on streaming data. (prediction length = 90 seconds)

Dataset	Indy500-2018		Indy500-2019	
Leader prediction	Acc	MAE	Acc	MAE
RandomForest	0.71	1.04	0.71	1.08
SVM	0.78	1.00	0.77	1.04
XGBoost	0.75	1.01	0.75	1.02
LSTM	0.80	0.95	0.79	0.34
Leader change prediction				
RandomForest	0.76	0.24	0.74	0.26
SVM	0.82	0.18	0.82	0.18
XGBoost	0.80	0.20	0.81	0.19
LSTM	0.83	0.34	0.84	0.35

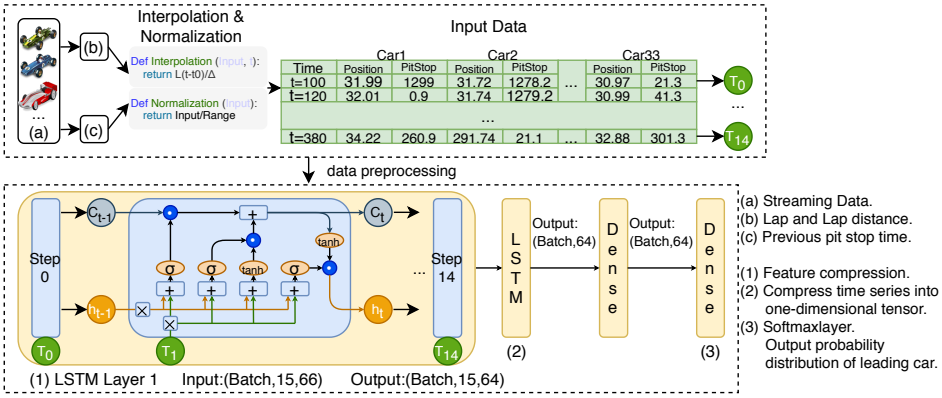


Fig. 8: Real-time LSTM prediction model.

data to the range of -1 to 1. 2) Data sorting. Due to the symmetry of the input data, that is, any data exchanged between two cars can still get a legal data set. Therefore, a model with more parameters is required to learn this symmetry. In order to avoid unnecessary complexity, we sort the data according to the position of the car. That is, the data of the current leading car is placed in the first column, the data of the currently ranked second car is placed in the second column, and so on. This helps to compress the model and improve performance.

c) LSTM model: In actual training, we use the following parameters. The input is a 15 * 66 tensor. The length of the time series is $S = 15$ steps, and the interval between each step is $\Delta_S = 20$ seconds. Label is one hot encoding of the leading car after t seconds in the future. We used data from year 2013 to 2018 for training. Then use the 2019 data to test the accuracy of the trained model.

According to section III-A. The prediction problem of racing cars has the characteristics of non-linearity, non-periodicity, randomness, and timing dependence. The traditional statistical learning model (Naive Bayes, SVM, Simple Neural Networks) is difficult to deal with the problem of time series prediction, since the model is unable to understand the time-series dependence of data. Traditional time series prediction models such as ARMA / ARIMA can only deal with linear time series with certain periodicity. The anomaly events and human strategies in the racing competition make these methods no longer applicable. Therefore, time series prediction models (RNN, GRU, LSTM, etc.) based on deep learning are more suitable for solving such problems. Table VI shows the experimental results, which verify our hypothesis that the time series prediction model based on deep learning obtained the highest accuracy.

G. Performance Study of the Leading Car Prediction LSTM Model

Although we used historical data to train the models, for the nature of these prediction tasks, real time training and continuous learning would be a more ideal approach in the actual event, thus performance would be a great concern. We have tackled this by using simple model architecture, and also have conducted experiments on various different platforms to find suitable ones. Table VII has the experiments settings and overall execution results.

One interesting result was that, unlike usual deep learning applications where GPU usually significantly outperform CPU platform, in our application where the model architecture is simple and training data-sets are not huge, CPU performs relatively well while GPU and VE has only modest speedup (while for certain implementation it was even slower). A performance study with tensorboard profiling can provide some insights why this is the case.

Figure 9 shows the percentage breakdown of overwall walltime spent on different operations based on tensorboard profiling results of experiments running on different platforms, while highlighting the most time-consuming major operations, as identified from the theoretical architecture of an LSTM cell as depicted in Figure 8.

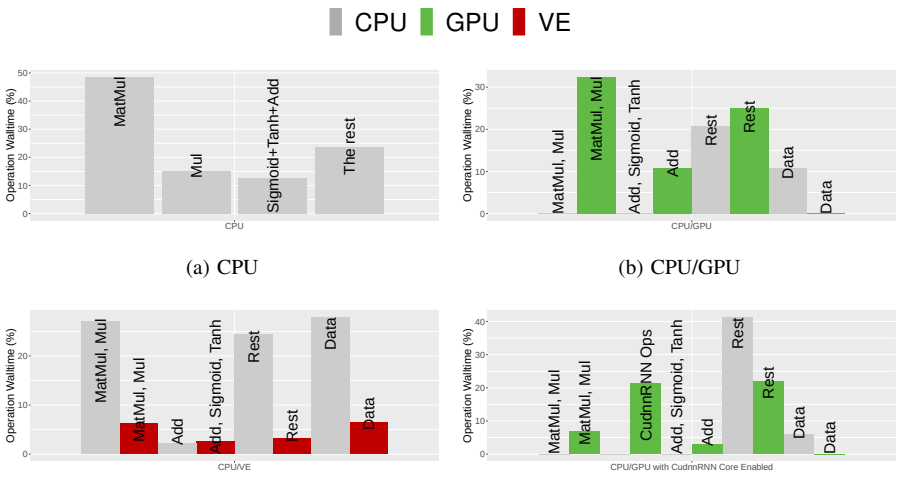
Intel vTune [6] and Advisor [5] profiling of experiments on CPU platform suggests similar results. Figure 10 shows the roofline chart of the IndyCar ranking prediction running on the CPU platform. The chart highlights the three operations - sum, product, and logistic - in the lower tensorflow implementation level. Based on the vTune profiling, these three operations spent over 85% overall CPU time among the similar group of operations that Intel Advisor identified as having most impact on performance and that are good candidates to look into for performance improvement. This corresponds well with the observation from tensorboard profiling, considering these top three operations are the lower level implementation of the MatMul, Mul, Add, and Sigmoid in our experimented CPU platform.

While on CPU platform the actual walltime were spent mostly on the kernel operations (as shown in Figure 9), as identified from the LSTM model architecture, the charts for GPU and VE platforms show quite different results.

For GPU, while using the same operation-by-operation approach without the support of CudnnRNN optimization (Figure 9b), the walltime spent on the same kernels were only about 40%, while over half of the walltime were from the auxiliary operations and data movement overhead. GPU

TABLE VII: Experiments Hardware Specification and Overall Performance of the Leading Car Prediction LSTM Model Running on Different Platforms. The model was implemented as a tensorflow keras model. CPU and GPU experiments used the official v2.0 tensorflow, while Vector Engine (VE) used a experimental forked version from the same version [10]. The cpu-only results were using only CPU resources on each specified platform. The results were training speed (μs/sample). The speed up showed the results with accelerator compared to the cpu-only results for that platform. For GPU platform, one result was with CudnnRNN support and the other one without.

Platform	CPU only	with Accelerator	speedup	Hardware Specification
CPU	268			Intel Xeon E5-2670 v3 @2.30GHz, with 128G RAM
CPU+GPU	299	397 244 (with CudnnRNN)	0.75 1.22	Intel Xeon CPU E5-2630 v4, with 128G RAM; GPU (V100-SXM2-16GB)
CPU+VE	262	223	1.17	Intel Xeon Gold 6126 CPU @2.60GHz, with 192G RAM; VE (SX-Aurora Vector Engine)



(a) CPU

(b) CPU/GPU

(c) CPU/VE

(d) CPU/GPU with CudnnRNN

Fig. 9: Operation Walltime Percentage on Different Platforms. On CPU platform (a), the kernel operations (MatMul, Mul, Add, Sigmoid, Tanh) accounts for over 75% of the overall walltime spent. On GPU platform (b), 2/3 walltime were spent on the accelerator side but only 40% were spent on the kernels and 25% were on other auxiliary operations. On VE platform (c) about 80% walltime spent on CPU side while the rest were offloaded to VE side. On GPU platform with CudnnRNN support (d), due to the CudnnRNN optimization the major operations were not the same kernel operations.

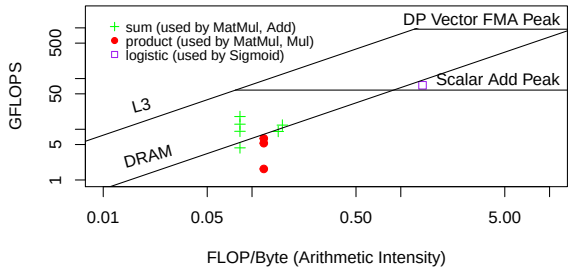


Fig. 10: Roofline chart of the IndyCar Ranking Prediction Model Running on the CPU Platform. The chart hightlights the operations in the lower tensorflow implementation level of the identified kernels - MatMul, Mul, Add, and Sigmoid.

with CudnnRNN optimization (Figure 9d) has improved this situation by minimizing the overhead and fully utilizing the hardware advantage of GPU. This optimization for LSTM and RNN models in CUDA library [8], [14] includes combining and streaming matrix multiplication operations to improve the parallelism as well as fusing point-wise operations to reduce data movement overhead and kernel invocation overhead, among other tricks for more complex model. The result was a faster execution time with a 1.22x speedup, comparing to the operation-by-operation approach which was actually slower than the CPU-only approach.

Vector Engine on NEC SX-Aurora machine provides another possible good fit due to its high memory bandwidth and high efficiency for vectorized operations. Figure 9c shows the same operation percentage breakdown and allocation between Vector Host (CPU) and VE. Among the kernel operations, Sigmoid and Tanh were all offloaded to the VE side, while for MatMul, Mul, and Add the operations occurred on both side. The offloading strategy of certain operations from VH to VE was based on the criteria that the potential walltime saved can offset the overhead from offloading, so the small insignificant calculations would be just handled on CPU side. Still, the overhead from auxiliary operations and data movement was quite significant. If similar fusing operation strategy were to

applied, this large portion of the overhead could be potentially minimized thus achieving higher speedup.

In either GPU or VE case, for more complex model and larger data we expect the benefit from operation offloading from CPU to accelerator would largely exceed the data movement and kernel launch overhead thus obtaining higher speedup. We have observed this during the iteration of model optimization, when we experimented on one previous similar but more complex model, in which we observed about 2x speedup for both GPU and VE.

V. Related Work

Forecasting in general: Forecasting is a heavily studied problem interested across domains. *decomposition to address uncertainty.* Classical statistical methods(e.g,ARIMA and exponential smoothing) and machine learning methods(e.g, SVR, Random Forest and XGB) are wildly applied in the problems with few of randomness, non-stationary and irregularity. To deal with the problem of high uncertainty, decomposition and ensemble is often used to separate the uncertainty signals from the normal patterns and model them in dependently. [26] utilizes the Empirical Mode Decomposition [22] algorithm to decompose the load demand data into several intrinsic mode functions and one residue, then models each of them separately by a deep belief network, finally forecast by the ensemble of the sub-models. Another type of decomposition occurs in local and global modeling. ES-RNN [31], winner of M4 forecasting competition [7], hybrids exponential smoothing to capture non-stationary trends per series and learn global effects by RNN, ensemble the outputs finally. Similar approaches are adopted in DeepState [27], DeepFactor [35]. In this work, based on the understanding of the cause effects of the problem, we decompose the uncertainty by modeling the causal factors and the target series separately and hybrid the sub-models according to the cause effects relationship. Different from the works of counterfactual prediction [12], [19], we do not discover causal effects from data.

modeling extreme events. Extreme events [23] are featured with rare occurrence, difficult to model, and their prediction are of a probabilistic nature. [24], [36] use a autoencoder to capture complex time-series dynamics during extreme events, show improved results. Autoencoder shows improve results in capturing complex time-series dynamics during extreme events, such as [24] for uber riding forecasting and [36] which decomposes normal traffic and accidents for traffic forecasting. [17] proposes to use a memory network with attention to capture the extreme events pattern and a novel loss function based on extreme value theory. In our work, we classify the extreme events in car racing into different categories, model the more preditable pit stops in normal laps by MLP with probabilistic output. Exploring autoencoder and memory network can be one of our future work.

express uncertainty in model. [18] first proposed to model uncertainty in deep neural network by using dropout as a Bayesian approximation. [36] followed this idea and successfully apply it to large-scale time series anomaly detection at Uber. Our work follows the idea in [29] that parameterizes a fixed distribution with the output of a neural network. [34] adopts the same idea and apply it to weather forecasting.

Car racing forecasting: *Simulation-based method:*Racing simulation is widely used in motor sports analysis [1], [15], [20]. To calculating the final race time for all the cars accurately, a racing simulator models different factors that impact lap time during the race, such as car interactions, tyre degradation, fuel consumption, pit stop etc., via equations with fine tuned parameters. Specific domain knowledge are necessary to build successful simulation models. [20] presents a simulator that reduce the race time calculation error to around one second for Formula 1 2017 Abu Dhabi Grand Prix. But, the author mentioned that user is required to provide the pit stop information for every driver as an input.

Machine learning-based method: [16], [32] is a series of work forecasting the decision-to-decision loss in rank position for each racer in NASCAR. [32] describes how they leveraged expert knowledge of the domain to produce a real-time decision system for tire changes within a NASCAR race. They chose to model the change in rank position and avoid predicting the rank position directly since it is complicated due to its dependency on the timing of other racers' pit stops. In our work, we aim to build a forecasting that rely less on domain knowledge and investigate the pit stop modeling.

VI. Conclusion

In this work, we build rank position forecasting model for car racing on IndyCar dataset. We adopt the deep learning modeling approach that aims to do it best to automatically extract features and learn effective model, reducing the dependency on domain experts in other machine learning and simulation approaches. A combination of encoder-decoder network and separate MLP network that capable to deliver probabilistic forecasting are proposed to model the pit stop events and rank position in car racing.

Through extensive evaluation experiments, we find that pit stop information is critical for rank position forecasting tasks. Our proposed model achieves significant better accuracy than baseline models when pit stop information are given. When using predicted pit stop information, the model obtains comparable performance in both the rank position forecasting task and change of the rank position forecasting task, with the advantages of needing less feature engineering efforts and providing probabilistic forecasting that enables racing strategy optimizations via our deep learning based model.

There are several limitations of this work. Since there are not many related work, the performance evaluation in this paper is still limited. An other major challenge lies in the limitation of the volume of the dataset. Car racing is a one time event that the observed data are always limited. Considering to add extra information with finer granularity could be one direction of our future work.

References

[1] Building a Race Simulator. https://f1metrics.wordpress.com/2014/10/03/building-a-race-simulator/. visited on 04/15/2020.

[2] IndyCar Dataset. https://racetools.com/logfiles/IndyCar/. visited on 04/15/2020.
[3] IndyCar Stats. https://www.indycar.com/Stats.
[4] IndyCar Understanding-The-Sport. https://www.indycar.com/Fan-Info/INDYCAR-101/Understanding-The-Sport/Timing-and-Scoring. visited on 04/15/2020.
[5] Intel Advisor. https://software.intel.com/content/www/us/en/develop/tools/advisor.html.
[6] Intel vTune. https://software.intel.com/content/www/us/en/develop/tools/vtune-profiler.html.
[7] M4Competition. https://forecasters.org/resources/timeseriesdata/m4-competition/. visited on 04/15/2020.
[8] Optimizing Recurrent Neural Networks in cuDNN 5. https://devblogs.nvidia.com/optimizing-recurrent-neural-networks-cudnn-5/.
[9] PitStop. https://en.wikipedia.org/wiki/Pit_stop. visited on 04/15/2020.
[10] TensorFlow for SX-Aurora TSUBASA. https://github.com/sx-aurora-dev/tensorflow.
[11] B. Adhikari, X. Xu, N. Ramakrishnan, and B. A. Prakash. EpiDeep: Exploiting Embeddings for Epidemic Forecasting. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 577–586, New York, NY, USA, 2019. ACM. event-place: Anchorage, AK, USA.
[12] A. M. Alaa, M. Weisz, and M. van der Schaar. Deep Counterfactual Networks with Propensity-Dropout. *arXiv:1706.05966 [cs, stat]*, June 2017. arXiv: 1706.05966.
[13] A. Alexandrov, K. Benidis, M. Bohlke-Schneider, V. Flunkert, J. Gasthaus, T. Januschowski, D. C. Maddix, S. Rangapuram, D. Salinas, J. Schulz, L. Stella, A. C. Türkmen, and Y. Wang. GluonTS: Probabilistic Time Series Models in Python. *arXiv:1906.05264 [cs, stat]*, June 2019. arXiv: 1906.05264.
[14] J. Appleyard, T. Kocisky, and P. Blunsom. Optimizing performance of recurrent neural networks on gpus. *arXiv preprint arXiv:1604.01946*, 2016.
[15] J. Bekker and W. Lotz. Planning Formula One race strategies using discrete-event simulation. *Journal of the Operational Research Society*, 60(7):952–961, 2009. Publisher: Taylor & Francis.
[16] C. L. W. Choo. *Real-time decision making in motorsports: analytics for improving professional car race strategy*. PhD Thesis, Massachusetts Institute of Technology, 2015.
[17] D. Ding, M. Zhang, X. Pan, M. Yang, and X. He. Modeling Extreme Events in Time Series Prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 1114–1122, New York, NY, USA, 2019. ACM. event-place: Anchorage, AK, USA.
[18] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.
[19] J. Hartford, G. Lewis, K. Leyton-Brown, and M. Taddy. Deep IV: a flexible approach for counterfactual prediction. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 1414–1423, Sydney, NSW, Australia, Aug. 2017. JMLR.org.
[20] A. Heilmeier, M. Graf, and M. Lienkamp. A Race Simulation for Strategy Decisions in Circuit Motorsports. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2986–2993, Nov. 2018.
[21] A. Heilmeier, M. Graf, and M. Lienkamp. A race simulation for strategy decisions in circuit motorsports. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2986–2993. IEEE, 2018.
[22] N. E. Huang, Z. Shen, S. R. Long, M. C. Wu, H. H. Shih, Q. Zheng, N.-C. Yen, C. C. Tung, and H. H. Liu. The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis. *Proceedings of the Royal Society of London. Series A: mathematical, physical and engineering sciences*, 454(1971):903–995, 1998. Publisher: The Royal Society.
[23] H. Kantz, E. G. Altmann, S. Hallerberg, D. Holstein, and A. Riegert. Dynamical Interpretation of Extreme Events: Predictability and Predictions. In S. Albeverio, V. Jentsch, and H. Kantz, editors, *Extreme Events in Nature and Society*, The Frontiers Collection, pages 69–93. Springer, Berlin, Heidelberg, 2006.
[24] N. Laptev, J. Yosinski, L. E. Li, and S. Smyl. Time-series extreme event forecasting with neural networks at uber. In *International Conference on Machine Learning*, volume 34, pages 1–5, 2017.
[25] B. Liao, J. Zhang, C. Wu, D. McIlwraith, T. Chen, S. Yang, Y. Guo, and F. Wu. Deep Sequence Learning with Auxiliary Information for Traffic Prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pages 537–546, New York, NY, USA, 2018. ACM. event-place: London, United Kingdom.
[26] K. Qiu, Y. Ren, P. N. Suganthan, and G. A. J. Amaratunga. Empirical Mode Decomposition based ensemble deep learning for load demand time series forecasting. *Applied Soft Computing*, 54:246–255, May 2017.
[27] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski. Deep state space models for time series forecasting. In *Advances in neural information processing systems*, pages 7785–7794, 2018.
[28] R. Ryan, H. Zhao, and M. Shao. CTC-Attention based Non-Parametric Inference Modeling for Clinical State Progression. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 145–154, Dec. 2019.
[29] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, Oct. 2019.
[30] M. W. Seeger, D. Salinas, and V. Flunkert. Bayesian intermittent demand forecasting for large inventories. In *Advances in Neural Information Processing Systems*, pages 4646–4654, 2016.
[31] S. Smyl. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1):75–85, Jan. 2020.
[32] T. Tulabandhula. *Interactions between learning and decision making*. PhD Thesis, Massachusetts Institute of Technology, 2014.
[33] T. Tulabandhula and C. Rudin. Tire changes, fresh air, and yellow flags: challenges in predictive analytics for professional racing. *Big data*, 2(2):97–112, 2014.
[34] B. Wang, J. Lu, Z. Yan, H. Luo, T. Li, Y. Zheng, and G. Zhang. Deep uncertainty quantification: A machine learning approach for weather forecasting. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2087–2095, 2019.
[35] Y. Wang, A. Smola, D. C. Maddix, J. Gasthaus, D. Foster, and T. Januschowski. Deep Factors for Forecasting. *arXiv:1905.12417 [cs, stat]*, May 2019. arXiv: 1905.12417.
[36] R. Yu, Y. Li, C. Shahabi, U. Demiryurek, and Y. Liu. Deep learning: A generic approach for extreme condition traffic forecasting. In *Proceedings of the 2017 SIAM international Conference on Data Mining*, pages 777–785. SIAM, 2017.

VII. Appendix

A. Feature engineering and parameter tuning for machine learning baselines

Three machine learning models, RandomForest, SVM and XGBoost, have been presented as baselines in the forecasting tasks.

First, performance of machine learning models rely heavily on feature engineering, e.g., [33] employs more than one hundred of features by the help of domain experts. In this work, we follow the similar ideas of feature extraction for the machine learning baselines in the following five aspects: global information, Rank, LapTime, PitStop and features of the near neighbors, as in Table. VIII. RankNet, as a deep learning based forecasting model, demonstrates its advantages over traditional machine learning methods in the low cost of feature engineering, as the recurrent neural network learns better feature representations from the input sequence than manual feature extractions deployed in Table. VIII.

Hyper-parameter tuning is another important factor for these machine learning models. We deployed grid search and cross validation to find the optimal hyper parameters. In contrast, RankNet has less hyper parameters.

B. Short-term rank position forecasting

Fig.11 demonstrate the performance improvement over a strong baseline CurRank for all the models. It shows that the three machine learning models, deep learning model DeepAR and the joint train model RankNet-Joint all failed to get better accuracy than

TABLE VIII: Features extracted for machine learning models. In short-term forecasting, a stage is defined as a window with length of the same *context_ length* in RankNet. In stint forecasting, a stage is the laps between two consecutive pit stops.

Type	Feature	Meaning
Gobal info	start_position	#rank at the beginning of the race
	stageid	sequence id of stage
	firststage	is the first stage?
Rank	start_rank	#rank at the forecasting position
	start_rank_ratio	#rank/totalcarnum
	top_pack	#rank in top5?
	bottom_pack	#rank in bottom5?
	average_rank	1st to 3rd moment of rank inprevious stage
	change_in_rank	
	rate_of_change	
	average_rank_all	1st to 3rd moment of rank inall previous stages
	change_in_rank_all	
	rate_of_change_all	
Laptime	laptime_green_mean_prev	mean and std of the lap time in green laps of the previous stage
	laptime_green_std_prev	
	laptime_green_mean_all	mean and std of the lap time in all green laps before
	laptime_green_std_all	
	laptime_mean_prev	mean and std of the lap time in all laps of the previous stage
	laptime_std_prev	
	laptime_mean_all	mean and std of the lap time in all laps before
	laptime_std_all	
Pitstop	pit_in_caution	the previous pit in caution lap
	laps_prev	lap number to the previous pitstop
	cautionlaps_prev	caution laps number to the previous pitstop
	pittime_prev	pit time of the previous pitstop
Neighbors' Info	nb_change_in_rank	features of three previous cars and three following cars
	nb_laptime_difference	

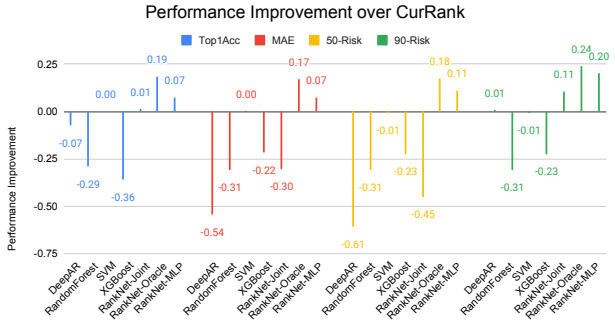


Fig. 11: Performance comparison for short-term forecasting models on Indy500-2018.

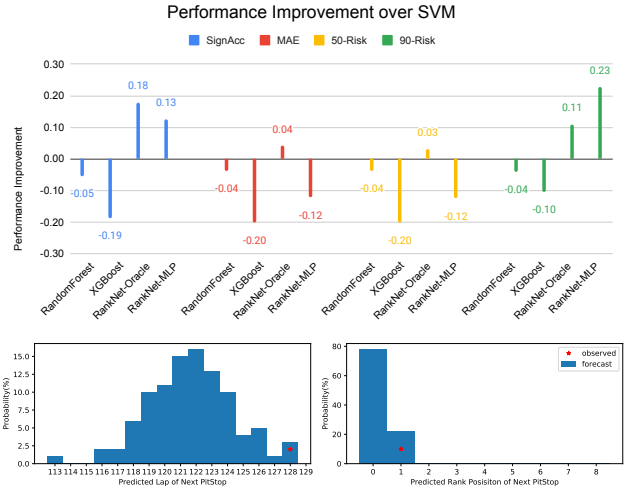


Fig. 12: Performance comparison for stint forecasting models on Indy500-2018.

CurRank. By contrast, RankNet-Oracle achieves significant better performance than CurRank, with 19% better in Top1Acc and 17% better in MAE. RankNet-MLP, our proposed model, is not as good as RankNet-Oracle, but still able to exceed CurRank more than 7% in both Top1Acc and MAE. It also achieves more than 20% improvement of accuracy on 90-risk when probabilistic forecasting get considered.

C. Stint rank position forecasting

For long-term forecasting, CurRank does not perform well. Fig.12(a) demonstrate the performance improvement over a baseline SVM for all the models. Performance of RankNet-MLP obtains significant better accuracy, more than 10%, on SignAcc and 90-Risk over the machine learning models, while it has a worse MAE and 50-Risk, also more than 10%.

Fig.12(b) shows an example of stint forecasting with 100 samples at pit stop lap 94 for car 12 in Indy500-2018. When the observed next pit stop lap is 128, RankNet-MLP predicts it with probability around 3%. Accordingly, observed rank at next pit stop is 1, and RankNet-MLP predicts it with probability 22%. This deviation comes from the uncertainty of extreme events and the bias of the trained model. Because of the sparsity of data, the PitModel is trained on data from 2013 to 2017, improvements of the cars enable them run longer between pit stops at 2018, which may bring bias.

D. Impact of Oracle Features

An oracle model, a model given the future pit stop information as input, can be used to represents the upper bound of performance that

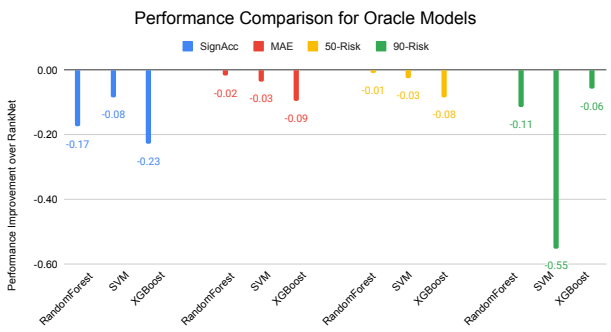


Fig. 13: Performance comparison of the oracle models for stint forecasting models, on Indy500-2018.

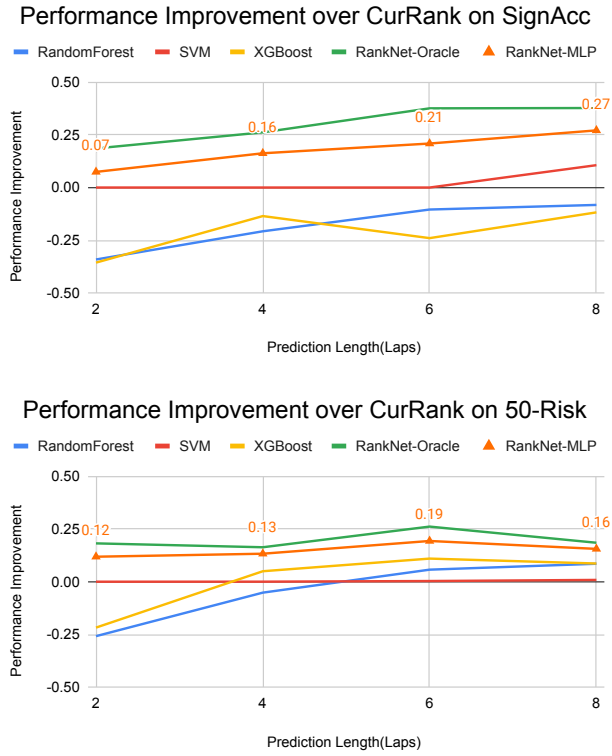


Fig. 14: Performance improvements over different prediction length, on Indy500-2018.

a model can obtain when evaluated with the observed data.

As shown in Fig. 13, the three machine learning models are inferior to RankNet over all the four metrics when oracle features are provided. It supports the choice of deep models that have the advantages in learning a better feature representations.

E. Impact of prediction length

RankNet-MLP shows stable performance advantages over other machine learning models when increasing the prediction length. Fig. 14 demonstrates the performance improvements over CurRank model on two metrics. Among all the models, RankNet-Oracle keeps the best accuracy and RankNet-MLP follows and shows significant better performance over the others.

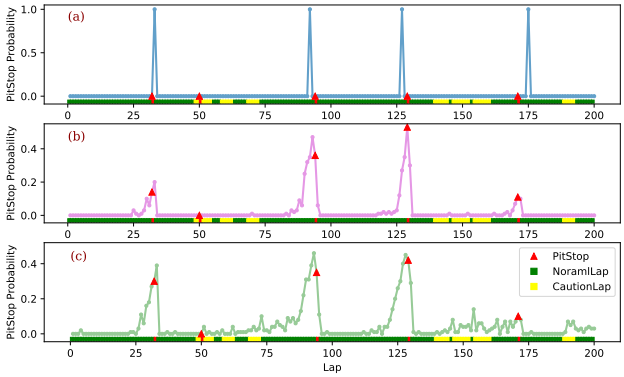


Fig. 15: Probability of pit stop in two laps forecasting, Car12 in Indy500-2018.(a) w/o uncertainty and trained on long normal pits. (b)with uncertainty and trained on long normal pits (c)with uncertainty and trained on full dataset.

F. PitModel Evaluation

In this sub-section, we evaluate the performance of PitModel in details. According to Section. III-A, we separate the pit stops into three categories: caution pits, long normal pits(stint length longer than 23) and short normal pits((stint length shorter than or equal to 23). Modeling trained on the raw pit data is supposed to be challenging and modeling on the long normal pits with the short distance section removed could to be more stable.

Table. IX and X shows the performance of PitModel trained on the long normal pits dataset and the full dataset respectively. A multilayer perceptron (MLP) of three layers with 10,5,5 neurons each is deployed. Indy500-2013 to Indy500-2017 are used for training and Indy500-2018 is used for testing. Beside MAE, 50-Risk and 90-Risk, two new evaluation metrics are utilized, which are F1 score and recall score for correctly predicting the occurrence of pitstop at two laps in the future. In order to demonstrate more detailed performance comparison, the evaluation results are itemized for the three categories of pit stops, and for two types of the model, with or without uncertainty forecasting support. Comparing across the categories of pit stops, all the models obtain best performance in the long normal pits testset, while fail in the short normal pits with extraordinary large MAE and zero recall. Caution pits show poor F1 and Recall, indicating that all the models are not capable to predict caution pits accurately. When comparing over the two types of model, the one with uncertainty support show significantly better performance than the one without uncertainty support. The model trained on long normal pits is marginally better than the one trained on the full dataset when testing on the long normal pits. Although the latter one shows some advantages in predicting caution pits, the previous one is still preferred because caution pits have much less impacts on the downstream task of rank position forecasting.

Fig. 15 demonstrates the probability of pit stop in two laps forecasting for Car12 in Indy500-2018. Models with uncertainty support(Fig.15b&c) shows obvious better coverage than the one without uncertainty output(Fig.15a), which misses all the ground truth. Model trained on long normal pits(Fig.15b) delivers more certain forecasting around the normal pit stops compared with the one trained on full dataset(Fig.15c), at the cost of failing to predict a short caution lap pit stop on lap 50.

G. Model feature comparison

See Table. XI

H. Impacts of the pit stop type on rank positions

TABLE IX: MLP PitModel performance comparison(trained by long normal pits data)

	With Uncertainty					Without Uncertainty				
TestSet	MAE	50-Risk	90-Risk	F1@2laps	Recall@2laps	MAE	50-Risk	90-Risk	F1@2laps	Recall@2laps
long normal pits	4.55	0.243	0.230	0.31	0.76	4.75	0.254	0.429	0.27	0.45
short normal pits	923.04	0.983	1.763	0.00	0.00	923.28	0.983	1.768	0.00	0.00
caution pits	10.47	0.913	0.269	0.02	0.03	10.47	0.913	0.301	0.01	0.01

TABLE X: MLP PitModel performance comparison(trained by all data)

	With Uncertainty					Without Uncertainty				
TestSet	MAE	50-Risk	90-Risk	F1@2laps	Recall@2laps	MAE	50-Risk	90-Risk	F1@2laps	Recall@2laps
long normal pits	6.04	0.322	0.121	0.33	0.52	122.04	6.546	1.309	0.00	0.00
short normal pits	924.08	0.984	1.749	0.00	0.00	837.72	0.892	1.594	0.00	0.00
caution pits	8.72	0.766	0.376	0.01	0.01	120.42	10.578	2.116	0.00	0.00

TABLE XI: Features comparison of the rank position forecasting models

Model	Feature Extraction	Uncertainty	PitModel	Variable Length Forecasting
Arima	Auto	N	N	Y
ML(RF,SVM,XGB)	Mannual	N	N	N
LSTM	Auto(LSTM Encoder)	N	N	N
DeepAR	Auto(LSTM Encoder)	Y	N	Y(LSTM Decoder)
RankNet-Joint	Auto(LSTM Encoder)	Y	Y(Joint Train)	Y(LSTM Decoder)
RankNet-MLP	Auto(LSTM Encoder)	Y	Y(Decomoposition)	Y(LSTM Decoder)

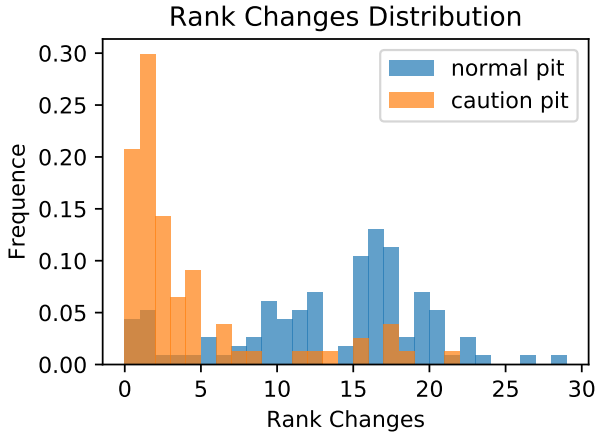


Fig. 16: Distribution of the rank position changes before and after a pit stop. Caution pits has much less impacts on rank position compared with normal pits.