Rank Position Forecasting in Car Racing

Bo Peng[1] Jiayu Li[2] Selahattin Akkas[2]
Takuya Araki[3] Ohno Yoshiyuki[3] Judy Qiu[1]
[1,2]Indiana University
[3]NEC Corporation Japan
[1]{pengb, xqiu}@indiana.edu [2]{ jl145, sakkas}@iu.edu
[3]{takuya_araki, ohno.yoshiyuki}@nec.com

Abstract—**Rank position forecasting in car racing is a challenging problem when using a Deep Learning-based model over time-series data. It is featured with highly complex global dependency among the racing cars, with uncertainty resulted from existing and external factors; and it is also a problem with data scarcity. Existing methods, including statistical models, machine learning regression models, and several state-of-the-art deep forecasting models all perform not well on this problem. By an elaborate analysis of pit stop events, we find it critical to decompose the cause-and-effect relationship and model the rank position and pit stop events separately. In choosing a sub-model from different neural network models, we find the model with weak assumptions on the global dependency structure performs the best. Based on these observations, we propose RankNet, a combination of the encoder-decoder network and a separate Multilayer Perception network that is capable of delivering probabilistic forecasting to model the pit stop events and rank position in car racing. Further with the help of feature optimizations, RankNet demonstrates a significant performance improvement, where MAE improves 19% in two laps forecasting task and 7% in the stint forecasting task over the best baseline and is also more stable when adapting to unseen new data. Details of the model optimizations and performance profiling are presented. It is promising to provide useful interactions of neural networks in forecasting racing cars and shine a light on solutions to similar challenging issues in general forecasting problems.**

I. Introduction

Forecasting the rank position in motorsports is a challenging problem. First, the status of a race is highly dynamic. It is a collective effect of many factors, including the skills of the drivers, the configuration of the cars, the interaction among the racing 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 is a significant challenge for forecasting the future accurately. A successful model needs to capture the complex dependencies and express uncertainty. Secondly, motorsports forecasting has a data scarcity challenge where available real-time data are limited because only one trajectory for each car can be observed during the race. Moreover, some factors, such as pit stop and car crash, make huge impacts on the racing dynamics but are irregular and rare, which appear less than 5% in available data. These so-called "*extreme events*" [17] are a critical part of a model as we will show in our RankNet design. Deep learning-based forecasting has observed its success across domains in recent years. However, we find that the state-of-the-art models in this field are simulation methods or machine learning methods, all highly rely on the domain knowledge [1], [10], [14], [26]. Simply applying a deep learning model alone does not deliver better forecasting.

On the one hand, deep learning forecasting models have advantages over traditional statistical and machine learning methods in their powerful representation learning capability. They can alleviate the high dependency on domain knowledge as well as costly feature engineering. Recent representative deep forecasting models, such as DeepAR(2017) [22],DeepState(2018) [21], DeepFactor(2019) [29] and N-BEATS(2020) [19] are all able to capture local data dependency within a single time-series and global dependency among multiple time-series. On the other hand, deep forecasting models share common disadvantages, such as sample inefficiency that the model requires more training data to train and the difficulty in modeling causal dependency. Furthermore, different assumptions behind each model may limit its applicability to a specific problem. For example, DeepState is a state-space model that assumes a linear-Gaussian transition structure and assumes the time-series are conditional independent of the model parameters. DeepFactor, as a factor model, requires the data to be exchangeable time series and to model the global dependency explicitly by line combination of global factors. Since the rank forecasting problem is challenging due to its high dynamics and global dependence among the cars, models with strong global dependency structure assumptions do not perform well.

In this paper, IndyCar [3] presents a challenging time-series use case from the real-time racing events. The main contributions of the paper are as follows:

- We investigate how to build a deep forecasting model that tackles the dynamic global dependency issues in the rank position forecasting problem. The central idea is to build sub-models by model decomposition based on cause-and-effect factors of rank position. In this way, we can train the model on *extreme events* more efficiently and effectively.
- Different choices of deep learning models are explored and sub-models are proposed in our final solution **RankNet**, a car racing forecasting model combined with a deep encoder-decoder network, a probabilistic multi-layer perceptron(MLP) network, and domain knowledge-based optimizations. We've made this work and the Ranknet [1] model available as open source.

[1]https://github.com/DSC-SPIDAL/rankpredictor

- We conduct extensive experiments showing that RankNet improves forecasting performance significantly compared with statistical, machine learning, and deep learning baselines where MAE improvements more than 7% consistently.

II. Problem Statement

A. Background

Indy500 is the premier racing event of the IndyCar series. Each year, 33 cars compete on a 2.5-mile oval track for 200 laps. The track is split into several sections or timeline and SF/SFP indicates the start and finish line on the track or the pit lane respectively. A local communication network broadcasts race information to all the teams, following a general data exchange protocol [3].

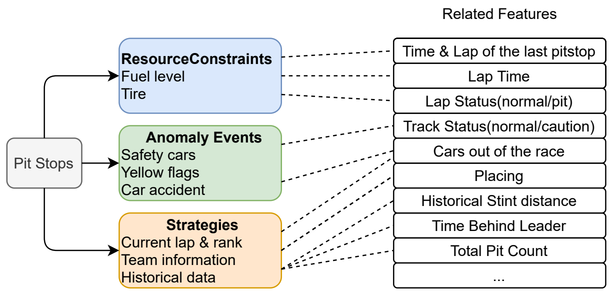


Fig. 1: Main factors affecting Pit stop

Rank position is the order of the cars crossing SF/SFP. In motorsports, a **pit stop** is a pause for refueling, new tires, repairs, mechanical adjustments, a driver change, a penalty, or any combination of them [6]. **Stint** refers to the gap between pitstops, or the gap between a pitstop and the start or end of the race. Unexpected events happen in a race, including mechanical failures or a crash. Depending on the severity 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 rises 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 another green flag raised.

B. Rank position forecasting problem and challenges

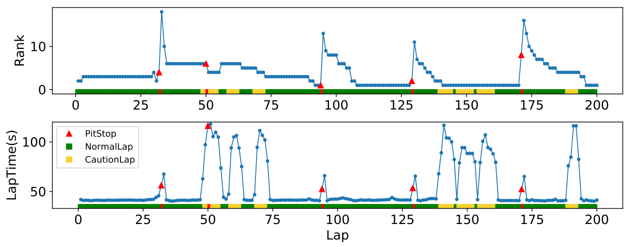


Fig. 2: Data examples of Indy500-2018. Rank and LapTime sequence of car12, the final winner.

The task of rank position forecasting is to predict a car's future rank position given the race's observed history. Fig.2 shows a typical $Rank$ and $LapTime$ sequence. Both of them are stable most of the time, indicating the race's predictable aspects that the driver's performance is stable. However, they both show abrupt changes when the **racing status**, including $LapStatus$ and $TrackStatus$, changes. Pit stop slows down the car and leads to a loss of rank position temporarily in the next few laps. Caution laps also slow down the car but do not affect the rank position much.

Fig.2 demonstrates the highly dynamic characteristics of this problem. The data sequence contains different phases, affected by the racing status. As for pit stop decisions, a team will have an initial plan for pit stops before the race, and the team coach will adjust it dynamically according to the status of the race. 'Random' events, such as mechanical failures and crashes, also impact the decision. 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 instance of its distribution is observed in one race. Therefore, even the cause-effect relationship between pit stop and rank position is known, forecasting of rank is still challenging due to the uncertainty in pit stop events. Figure3 illustrates that the existing statistical, machine learning, and deep learning models give poor rank forecasting performance.

C. Pitstop analysis

Pit stops play a critical role in the uncertainty of the rank position forecasting. Previous studies [11], [14], [26] did some preliminary analysis of the factors that affect pit stop. In this section, we study the causes of pit stops based on the data of Indy500, which helps us select the main features to build the deep learning model. As in Fig. 1, we divide the causes of pit stop into three categories: **resource constraints**, **anomaly events**, and **race strategies**.

a) Resource constraints: The distance between the two pit stops is limited by the car's fuel tank volume and its tires. As in Fig. 4(a), no car runs more than 50 laps before entering the pit stop.

b) Anomaly events: Anomaly events are usually caused by mechanical failure or car accidents. When a severe accident occurs, $TrackStatus$ will change to Yellow Flag, which will change the pit stop strategy. In the Indy500 dataset, the number of the normal pit and caution pit are close, 777 and 763, respectively. These two types of pit stops show significant differences. In Fig. 4(a), the normal pit is a bell curve, and the caution pit scatters more evenly; In Fig. 4(b), we observe three linear sections in the CDF curve of the lap distance of the normal pit. The lower section of short distance pit may be caused mainly by unexpected reasons, such as mechanical failures, and keeps a low probability of less than 10%. The upper part of the long-distance pit occurs when many caution laps happen, in which case the cars run at a reduced speed that significantly reduces tire wear and fuel burn for a distance traveled. From these observations, modeling all pit stops from raw data would be difficult, and modeling the normal pit data with the short distance section removed should be more stable.

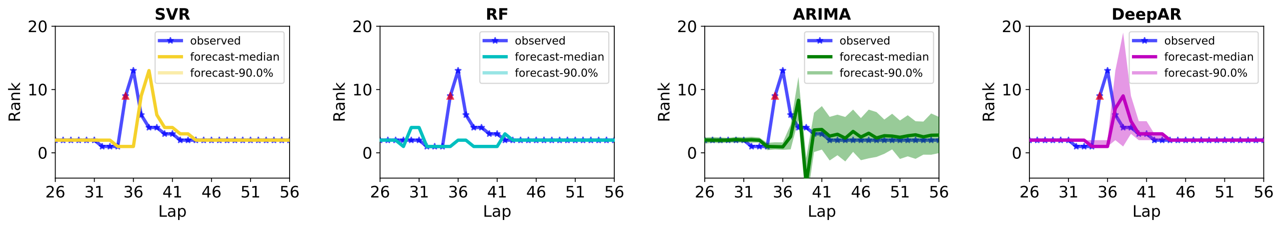


Fig. 3: Two laps forecasting results around pit stop lap 34 for car12 in Indy500-2019. (a)(b) Machine learning regression models. SVM learns a model very close to a two laps delay. RandomForest fails to predict the change around pit stop. (c) Statistical methods. ARIMA provides uncertainty predictions but lower performance, with difficulty to model the highly dynamics. (d) DeepAR, a state-of-the-art LSTM encoder-decoder model with uncertainty forecasting, also performs not well around pit stop.

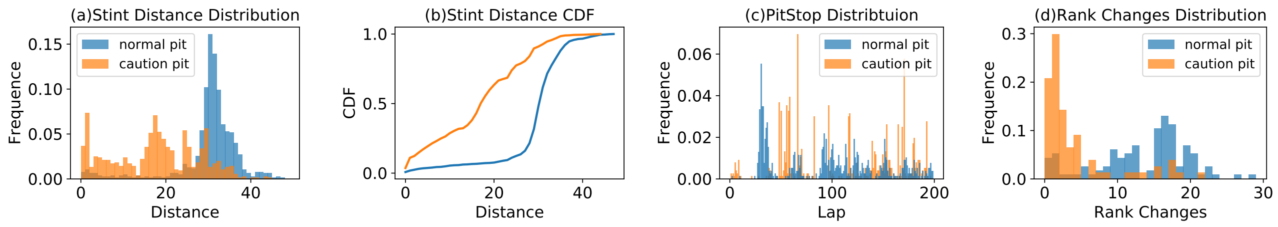


Fig. 4: Statistics and analysis of pit stop. Stint refers to laps between two consecutive pit stops. Pit stops occurred on caution lap denoted as Caution Pit, otherwise Normal pit. (a)(b) Distribution of stint distance. Normal pits and caution pits are different. (c) Considerable uncertainty of where pit stops occur. (d) Caution pits have much fewer impacts on rank position compared with normal pits.

c) Race strategies: In real competitions, the team coach also needs to make pit stop decisions considering the rivals' strategy and the collaboration among the team members, which are difficult to summarize with simple rules. To better understand race strategy, we need to combine the ranking, team information, and historical data of past races to train the model.

III. Methodology

A. Modeling uncertainty in high dynamic time-series

We explore the rank position forecasting as a sequence-to-sequence modeling problem. We use $z_{i,L}$ to denote the value of sequence i at lap L, $\mathbf{x}_{i,L}$ to represent the co-variate that is assumed to be known at any given lap. An encoder-decoder architecture is employed to map a input sequence $[z_{i,1}, z_{i,2}, \ldots z_{i,L_0}]$ to the output sequence $[z_{i,L_0+1} \ldots z_{i,L_0+k}]$. Here L_0 represents the length of the input sequence, and k represents the prediction length. Note that lap number L is relative, i.e. $L = 1$ corresponds the beginning of the input, not necessarily the first lap of the actual race.

We follow the idea proposed in [22] to deliver probabilistic forecasting to model the uncertainty. Instead of predicting the value of the target variable in the output sequence directly, a neural network predicts all parameters θ of a predefined probability distribution $p(z|\theta)$ by its output h.

For example, to model a Gaussian distribution for real-value data, the parameter $\theta = (\mu, \sigma)$ can be calculated as: $\mu(h_{i,L}) = W_\mu^T h_{i,L} + b_\mu$, $\sigma(h_{i,L}) = log(1 + exp(W_\sigma^T h_{i,L} + b_\sigma))$. The final output $z_{i,L}$ is then sampled from this distribution.

Our goal is to model the conditional distribution

$$P(z_{i,L_0+1:L_0+k}|z_{i,1:L_0}, \mathbf{x}_{i,1:L_0+k})$$

We assume that our model distribution $Q_\Theta(\mathbf{z}_{i,L_0+1:L_0+k}|z_{i,1:L_0}, \mathbf{x}_{i,1:L_0+k})$ consists of a product of likelihood factors

$$\begin{aligned} & Q_\Theta(\mathbf{z}_{i,L_0+1:L_0+k}|\mathbf{z}_{i,1:L_0}, \mathbf{x}_{i,1:L_0+k}) \\ &= \prod_{L=L_0+1}^{L_0+k} Q_\Theta(z_{i,L}|z_{i,1:L-1}, \mathbf{x}_{i,1:L_0+k}) \\ &= \prod_{L=L_0+1}^{L_0+k} p(z_{i,L}|\theta(\mathbf{h}_{i,L}, \Theta)) \end{aligned} \tag{1}$$

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

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

where h is a core function that is implemented by a deep neural network (e.g. LSTM or Transformer) parametrized by Θ.

The encoder-decoder architecture provides an advantage by supporting the incorporation of covariates known in the forecasting period. For example, in sales demand forecasting, holidays are known to be critical factors in achieving good predictions. In our case, caution laps and pit stops are essential factors to the rank position. But, different from the holidays, these variables in the future are unknown at the time of forecasting, leading to the need to decompose the cause effects in building the model.

B. Decomposition and extreme events modeling

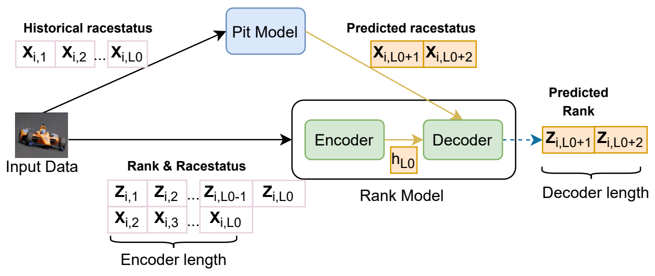
Changes of race status, including pit stops and caution laps, cause the phase changes of the rank position sequence.

TABLE I: Summary of input data used in **RankNet** model

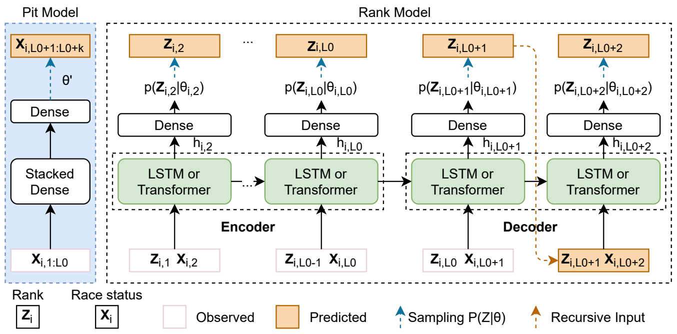
Variable	Feature	Domain	Description
Race status $\mathbf{X}_i$	$TrackStatus(i, L)$	T/F	Status of each lap for a car i, normal lap or caution lap.
	$LapStatus(i, L)$	T/F	Whether lap L is a pit stop lap or not for car i
	$CautionLaps(i, L)$	$\mathbb{N}$	At Lap L, the count of caution laps since the last pit lap of car i.
	$PitAge(i, L)$	$\mathbb{N}$	At lap L, the count of laps after the previous pit stop of car i.
Rank $\mathbf{Z}_i$	$Rank(i, L)$	$\mathbb{N}$	There are $Rank(i, L)$ cars that completed lap L before car i
	$LapTime(i, L)$	$\mathbb{R}+$	Time used by car i to complete lap L.
	$TimeBehindLeader(i, L)$	$\mathbb{R}+$	Time behind the leader of car i in lap L.

For a straightforward solution to address this dependency issue, we model the race status and rank position together and jointly train the model in 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 stops six times in a race. Therefore, LapStatus, a binary vector with a length equals to 200, contains only six ones, 3% effect data. TrackStatus, indicating the crash events, is even harder to predict.


Historical racestatus
$\mathbf{X}_{i,1}$ $\mathbf{X}_{i,2}$ ... $\mathbf{X}_{i,L0}$
Pit Model
Predicted racestatus
$\mathbf{X}_{i,L0+1}$ $\mathbf{X}_{i,L0+2}$
Input Data
Encoder
Decoder
h_{L0}
Predicted Rank
$\mathbf{Z}_{i,L0+1}$ $\mathbf{Z}_{i,L0+2}$
Rank & Racestatus
$\mathbf{Z}_{i,1}$ $\mathbf{Z}_{i,2}$ ... $\mathbf{Z}_{i,L0-1}$ $\mathbf{Z}_{i,L0}$
$\mathbf{X}_{i,2}$ $\mathbf{X}_{i,3}$ ... $\mathbf{X}_{i,L0}$
Rank Model
Decoder length
Encoder length

(a) Forecasting process: 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 drawn from the learned distribution. Features contained in the vectors $\mathbf{X}_i$ and $\mathbf{Z}_i$ are shown in Table I.


Pit Model
Rank Model
$\mathbf{X}_{i,L0+1:L0+k}$
θ'
Dense
Stacked Dense
$\mathbf{X}_{i,1:L0}$
$\mathbf{Z}_{i,2}$
$\mathbf{Z}_{i,L0}$
$\mathbf{Z}_{i,L0+1}$
$\mathbf{Z}_{i,L0+2}$
$p(\mathbf{Z}_{i,2}|\theta_{i,2})$
$p(\mathbf{Z}_{i,L0}|\theta_{i,L0})$
$p(\mathbf{Z}_{i,L0+1}|\theta_{i,L0+1})$
$p(\mathbf{Z}_{i,L0+2}|\theta_{i,L0+2})$
Dense
$h_{i,2}$
$h_{i,L0}$
$h_{i,L0+1}$
$h_{i,L0+2}$
LSTM or Transformer
Encoder
Decoder
$\mathbf{Z}_{i,1}$ $\mathbf{X}_{i,2}$
$\mathbf{Z}_{i,L0-1}$ $\mathbf{X}_{i,L0}$
$\mathbf{Z}_{i,L0}$ $\mathbf{X}_{i,L0+1}$
$\mathbf{Z}_{i,L0+1}$ $\mathbf{X}_{i,L0+2}$
Rank $\mathbf{Z}_i$
Race status $\mathbf{X}_i$
Observed
Predicted
Sampling P(Z|θ)
Recursive Input

(b) PitModel is a MLP predicting next pit stop lap given features of RaceStatus history. RankModel is stacked multi-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. 5: **RankNet** architecture

We propose a RankNet model to decompose the cause-effect of race status and rank position, as shown in Fig. 5(a), is composed of 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 with these predicted race status inputs.

C. *RankNet architecture*

The details of ***RankNet*** is shown in our neural network architecture for the two sub-models in Fig. 5(b). PitModel adopts a simple multilayer perceptron network(MLP), denoted as Stacked Dense layer in Fig. 5(b). RankModel adopts an encoder-decoder architecture, which is widely used in sequence modeling. Both encoder and decoder is a deep neural network capable of modeling long-range context dependencies, such as the classical multi-layer recurrent neural network(RNN) with LSTM cells [15], or more recently successful Transformer [27]. A dense layer converts the output of the encoder and decoder into the parameter of a predefined distribution.

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 at a 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 to the input. For example, DriverId represents the skill level of the driver.

Transformations are applied to these basic features to extract new features. Embedding for categorical DriverId is utilized. Accumulation sum transforms the binary status features into 'age' features, generating features such as CautionLaps and PitAge. Table I summarizes the definition of these features. Due to the data-sparse issue for pit stop events, instead of sequences input and output, PitModel uses CautionLaps and PitAge as input for better data efficiency 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. To deliver a variable-length prediction, e.g., in predicting the rank positions between two pit stops, we apply fixed-length forecasting recursively by using the previous output as input for the next prediction. For the probabilistic output, we take 100 samples for each forecasting. And the final rank positions of the cars are calculated by sorting the sampled outputs. The training and prediction process of RankNet is shown in Algorithm 1 and Algorithm 2.

IV. Experiments

A. *Dataset*

We evaluate our model on the car racing data of IndyCar series [2]. Due to the data scarcity in car racing, we have to incorporate more data to learn a stable model. Using the

Algorithm 1: RankNet Training in minibatch

```
input : A minibatch ($batch\_size = B$) of time serises
        $\{z_{i,1:L_0+k}\}_{i=1,...B}$ and associated covariates
        $\{\mathbf{x}_{i,1:L_0+k}\}_{i=1,...B}$.
1 for $i = 1 \ldots B$ and $L = L_0 + 1 \ldots L_0 + k$ do
2     Calculate the current state
      $\mathbf{h}_{i,L} = h(\mathbf{h}_{i,L-1}, z_{i,L-1}, \mathbf{x}_{i,L})$ through the neural
      network.
3     Calculate the parameter $\theta_{i,L} = \theta(\mathbf{h}_{i,L})$ of the predefined
      distribution $p(z|\theta)$.
4 The loss is obtained by log-likelihood:
      $$\mathcal{L} = \sum_{i=1}^{B} \sum_{L=L_0+1}^{L_0+k} \log p(z_{i,L}|\theta(\mathbf{h}_{i,L})) \quad (2)$$
5 Apply the ADAM optimizer to update the weights of the
  neural network by maximizing the log-likelihood $\mathcal{L}$.
```

Algorithm 2: Forecasting with RankNet

```
input : $\{\mathbf{x}_{i,1:L0}\}, \{z_{i,1:L0}\}$ , model trained with
        prediction_length $k$, forecasting start position $L_0$,
        end position $L_P$.
// Forecasting pit stops using PitModel.
1 Calculate $\mathbf{x}_{L_0+1:L_P}$; set future TrackStatus to zero.
2 while $L_0 < L_P$ do
      // Rank Model
3     Input the historical data at lap $L \leq L_0$ into the
      RankModel to obtain the initial state $\mathbf{h}_{i,L_0}$.
4     for $L = L_0, ...L_0 + k - 1$ do
5         Input $\{z_{i,L}, \mathbf{x}_{i,L+1}, \mathbf{h}_{i,L}\}$ into the RankModel to
          get $\theta_{i,L+1}$
6         Random sampling $\tilde{z}_{i,L+1} \sim p(\cdot|\theta_{i,L+1})$.
7         Update $z_{i,L+1}$ with $\tilde{z}_{i,L+1}$
8     $L_0 += k$
9 return $\tilde{z}_{i,L_P}$
```

historical data that is a long time ago can be ineffective because many factors change along the time, including the drivers' skills, configurations of the cars, and even the race rules. The same year data of other races are 'similar' in the status of the drivers, cars, and rules, but different shapes and lengths of the track lead to different racing dynamics.

In this paper, we select races of Motor Speedway after 2013 with at least 5 years of data each, and after removing corrupted data, get a dataset of 25 races from four events, shown in Table. II. Fig. 6 shows the data distribution by two statistics for this dataset. Among all the events, Indy500 is the most dynamic one, which has both the largest PitLapsRatio and RankChangesRatio; Iowa is the least.

We train models separately for each event. Races of the first five years are used as the training dataset, and the remains are used as testing data, which are labeled in Fig. 6. Since Pocono has only five years of data in total, its training set uses four of them. First, we start from Indy500 and use Indy500-2018 as a validation set. Then we investigate the generalization capability of the model on data of the other events.

TABLE II: Summary of the data sets.

Event	Year	Track Length	Track Shape	Total Laps	#Records	Usage
Indy500	2013-2017	2.5	Oval	200	6600	Training
Indy500	2018	2.5	Oval	200	6600	Validation
Indy500	2019	2.5	Oval	200	6600	Test
Iowa	2013, 2015-2018	0.894	Oval	250	6000	Training
Iowa	2019	0.894	Oval	300	7200	Test
Pocono	2013, 2015-2017	2.5	Triangle	160	3840	Training
Pocono	2018	2.5	Triangle	200	4800	Test
Texas	2013-2017	1.455	Oval	228	5472	Training
Texas	2018-2019	1.455	Oval	248	5704	Test

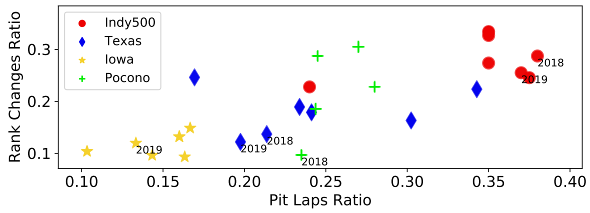


Fig. 6: Data distribution of Indycar Dataset. PitLapsRatio is the pit stop laps # divided the total laps #. RankChangesRatio refers to the ratio of laps with rank position changes between consecutive laps.

B. Baselines and implementation

As far as we know, there is no open-source model that forecasts rank position in car racing and no related work on the IndyCar series.

First, we have a naive baseline which assumes that the rank positions will not change in the future, denoted as CurRank. Secondly, We implement machine learning regression models as baselines that follow the ideas in [25] which forecast changes of rank position between two consecutive pit stops, including RandomForest, SVM, and XGBoost that do pointwise forecast. Thirdly, we test with four latest deep forecasting models as the choice of RankModel, including DeepAR(2017) [22], DeepState(2018) [21], DeepFactor(2019) [29], N-BEATS(2020) [19].

PitModel has three implementations. For example for RankNet, we have 1) RankNet-Joint is the model that trains the target with pit stop jointly without decomposition; 2) RankNet-Oracle is the model with ground truth TrackStatus and LapStatus as covariates input. It represents the best performance obtainable from the model, given the caution and pit stop information for a race; 3) RankNet-MLP employs model decomposition with a separate PitModel. Table. IV summarizes the features of all the models.

We build our model RankNet with the Gluonts framework [8]. RankNet is based on the DeepAR implementation in Gluonts, shares the same features, including sharing parameters between encoder and decoder, both implemented as stacking of multiple LSTM layers.

C. Model optimization

For machine learning baselines, we tune the hyper-parameters by grid search. For deep models, we tune the

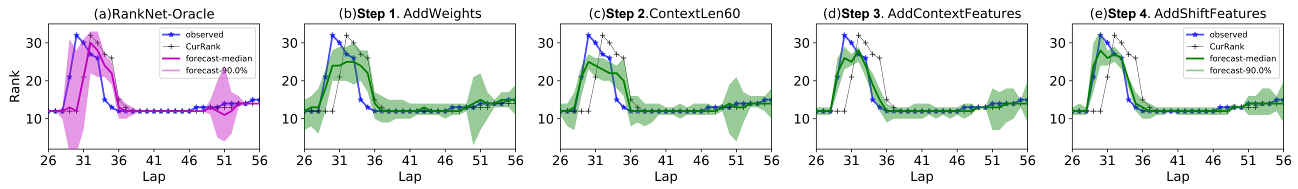


Fig. 7: Steps of RankNet model optimizations on two laps forecasting for Car13 Indy500-2018.(a)Basic RankNet model trained with Oracle race status features and context_length=40. (b)Adding larger weights to the loss for instances with rank changes, set the optimal weight to 9. (c)Tuning on parameter context_length, set optimal length to 60. (d)Adding context features, including LeaderPitCount: # of leading cars(based on the rank position at lap A-2) that go to pit stop at lap A; TotalPitCount:# of cars that go to pit stop at lap A. (e)Adding shift features, including Shift_RaceStatus: lapstatus and trackstatus of the future at lap A+2; Shift_TotalPitCount:# of cars that go to pit stop at lap A+2.

parameter of encoder length, loss weight, and use the default value of other hyper-parameters in the GluonTs implementation, as in Table. III. The model is trained by ADAM optimizer with an early stopping mechanism that decays the learning rate when the loss does not improve for 10 epochs until reaching a minimum value. Fig. 7 shows the process of further model optimization, starting from a basic RankNet model, optimizations are added step by step and tuned on the validation dataset.

TABLE III: Dataset statistics and model parameters

Parameter	Value
# of time-series	227(Indy500), 619(All)
# of training examples	32K(Indy500), 117K(All)
Granularity	Lap
Domain	$\mathbb{R}+$
Encoder length	**[20,40,60,80,100]**
Decoder length k	2
Loss weight	**[1-10]**
Batch size B	32
Optimizer	ADAM
Learning rate	1e-3
LR Decay Factor	0.5
# of lstm layers	2
# of lstm nodes	40
Training time	2h

TABLE IV: Features of the rank position forecasting models.

Name	RankModel	PitModel	Optimization
CurRank			
ARIMA	ARIMA		
RandomForest	RandomForest		custom features [25]
SVM	SVM		
XGBoost	XGBoost		
DeepAR	DeepAR		
DeepState	DeepState		
DeepFactor	DeepFactor		
N-BEATS	N-BEATS		
DeepAR-Oracle	DeepAR	Oracle	raw race status features (Table.I)
DeepState-Oracle	DeepState	Oracle	
DeepFactor-Oracle	DeepFactor	Oracle	
N-BEATS-Oracle	N-BEATS	Oracle	not support covariates
RankNet-Joint	DeepAR	Joint	
RankNet-Oracle	DeepAR	Oracle	loss weight + new race status features (Fig.7)
RankNet-MLP	DeepAR	MLP	

D. *Evaluation*

RankNet is a single model that can forecast both short-term rank position and long-term change of rank position between pitstops. First, we use mean absolute error(MAE) to evaluate all the sequences' average forecasting accuracy since they have the same units. Secondly, we evaluate the accuracy of the leader's correct predictions, denoted as Top1Acc, and the accuracy of correct predictions of the sign of the change which indicates whether a car achieves a better rank position or not, denoted as SignAcc.

Thirdly, a quantile based error metric ρ-risk [23] is used to evaluate the probabilistic forecasting performance. 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. The values of Top1Acc and SignAcc are the larger, the better, and the other metrics are less the better.

E. *Short-term rank position forecasting*

Table V shows the evaluation results of two laps rank position forecasting. CurRank demonstrates good performance. 73% leader prediction correct and 1.16 mean absolute error on Indy500-2019 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 patterns of the rank position variations are not easy to learn from history. When adding an oracle PitModel, DeepAR-Oracle shows a 28% improvement in MAE over CurRank. By adding further optimizations, RankNet-Oracle, which adopts the same RankModel network as DeepAR, achieves significantly better performance than CurRank, with 23% better in Top1Acc and 51% better in MAE. These results demonstrate the effectiveness of model decomposition and domain knowledge-based optimizations.

Comparing the performance of four state-of-the-art deep forecasting models as the choice of RankModel, we find DeepAR and N-BEATS obtains similar performance. However, N-BEATS is limited in supporting covariates, which prevents its adoption. DeepState and DeepFactor demonstrate inferior forecasting performance on this problem. We speculate

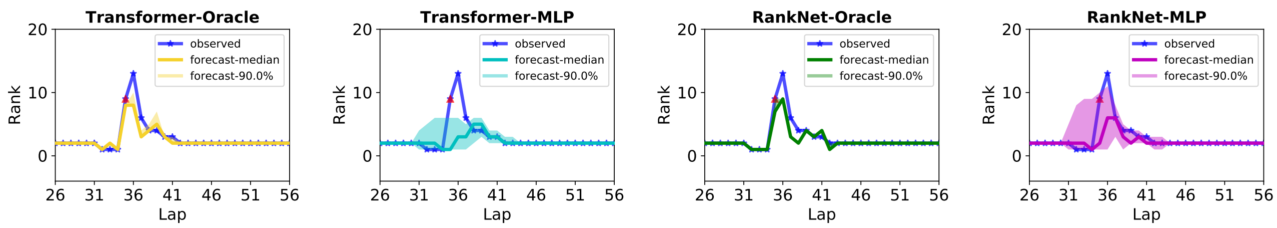


Fig. 8: RankNet forecasting results of two laps in the future for car12 in Indy500-2019.

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

	All Laps				Normal Laps				PitStop Covered Laps			
Model	Top1Acc	MAE	50-Risk	90-Risk	Top1Acc	MAE	50-Risk	90-Risk	Top1Acc	MAE	50-Risk	90-Risk
CurRank	0.73	1.16	0.080	0.080	0.94	0.13	0.009	0.009	0.55	2.09	0.144	0.144
ARIMA	0.54	1.45	0.110	0.105	0.70	0.47	0.047	0.042	0.40	2.32	0.166	0.162
RandomForest	0.62	1.31	0.091	0.091	0.78	0.39	0.027	0.027	0.47	2.14	0.147	0.147
SVM	0.73	1.16	0.080	0.080	0.94	0.13	0.009	0.009	0.55	2.09	0.144	0.144
XGBoost (2014)	0.64	1.25	0.086	0.086	0.76	0.27	0.019	0.019	0.54	2.12	0.146	0.146
DeepAR (2017)	0.73	1.22	0.086	0.085	0.93	0.21	0.018	0.017	0.55	2.12	0.147	0.145
DeepState (2018)	0.56	1.95	0.137	0.133	0.73	0.85	0.062	0.059	0.41	2.93	0.203	0.199
DeepFactor (2019)	0.01	10.44	0.684	0.683	0.00	10.79	0.714	0.712	0.02	10.13	0.658	0.657
N-BEATS (2020)	0.70	1.21	0.083	0.083	0.87	0.19	0.013	0.013	0.55	2.12	0.146	0.146
DeepAR-Oracle	0.88	0.84	0.063	0.060	0.93	0.20	0.016	0.015	0.84	1.42	0.105	0.099
DeepState-Oracle	0.73	1.38	0.096	0.093	0.85	0.72	0.051	0.050	0.63	1.98	0.136	0.133
DeepFactor-Oracle	0.01	8.30	0.523	0.521	0.01	8.56	0.542	0.541	0.01	8.06	0.506	0.504
RankNet-Oracle	0.90	0.57	0.045	0.040	0.94	0.20	0.021	0.018	0.87	0.90	0.067	0.061
RankNet-Joint	0.64	1.74	0.153	0.144	0.78	0.82	0.096	0.089	0.52	2.56	0.203	0.194
RankNet-MLP	**0.79**	**0.94**	**0.067**	**0.046**	**0.94**	**0.21**	**0.022**	**0.018**	**0.65**	**1.59**	**0.107**	**0.072**

that strong model assumptions on the global dependency structure prevent a model's success in this highly dynamic forecasting problem. DeepState assumes a linear-Gaussian transition structure, and DeepFactor requires exchangeable time series, which may not hold in the racing data. On the contrary, N-BEATS and DeepAR learn similarity among time series by sharing the same network in training without introducing strong structure assumptions. And also this is a data sparse problem, which prefers the model that can provide forecasts for items that have little history available, where DeepAR has advantages [22]. These results demonstrate the effectiveness of the encoder-decoder architecture for this problem.

Other machine learning models and RankNet-Joint all failed to get better accuracy than CurRank. RankNet-MLP, our proposed model, is not as good as RankNet-Oracle, but still able to exceed CurRank by 7% in Top1Acc and 19% in MAE. It also achieves more than 20% improvement of accuracy on 90-risk when probabilistic forecasting gets considered. Evaluation results on PitStop Covered Laps, where pit stop occurs at least once in one lap distance, show RankNet-MLP and RankNet-Oracle's advantages come from their capability of better forecasting in these *extreme events* areas. A visual comparison of RankNet over the baselines are demonstrated in Fig.8 and Fig.3.

F. Stint rank position forecasting

Table VI shows the results of the task of forecasting the rank position changes between consecutive pit stops. CurRank can not predict changes, thus gets the worst performance. Among

TABLE VI: Rank position changes forecasting between pit stops

Model	SignAcc	MAE	50-Risk	90-Risk
CurRank	0.15	4.33	0.280	0.262
RandomForest	0.51	4.31	0.277	0.276
SVM	0.51	4.22	0.270	0.249
XGBoost (2014)	0.45	4.86	0.313	0.304
DeepAR (2017)	0.37	4.08	0.265	0.268
DeepState (2018)	0.51	4.88	0.317	0.397
DeepFactor (2019)	0.54	9.51	0.622	0.668
N-BEATS (2020)	0.47	4.29	0.274	0.290
RankNet-Joint	0.60	5.83	0.388	0.486
RankNet-MLP	**0.65**	**3.79**	**0.245**	**0.169**
RankNet-Oracle	0.67	3.41	0.229	0.203

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. The performance of RankNet-MLP obtains significantly better accuracy and improves between 9% to 30% on the four metrics over SVM. For the four deep learning models, only DeepAR obtains better MAE performance over SVM. But RankNet-MLP is still achieving 7% improvements over DeepAR. Moreover, it forecasts future pit stops and thus different race status possibilities, which are not supported by the other baselines. RankNet is promising to be a tool to investigate and optimize the pit stop strategy.

G. RankNet with Transformer

In this experiment, we replace LSTM-based RNN with the Transformer implementation from the GluonTs library, which has multi-head attention(8 heads) and the dimension of the

TABLE VII: Two laps forecasting task on other races. MAE improvements is compared over CurRank on PitStop covered laps.

	MAE Improvement(Train by Indy500)				MAE Improvement(Train by same event)			
Dataset	RankNet -MLP	Random Forest	RankNet -Joint	Transformer -MLP	RankNet -MLP	Random Forest	RankNet -Joint	Transformer -MLP
Indy500-2019	**0.24**	-0.02	-0.08	0.12	**0.24**	-0.02	-0.08	0.12
Texas-2018	**0.11**	-2.13	-0.22	0.02	**0.15**	-0.10	-0.11	0.07
Texas-2019	**0.01**	-1.63	-0.29	-0.15	**0.10**	-0.13	-0.15	-0.02
Pocono-2018	**0.09**	-2.25	-0.02	-0.17	**0.06**	-1.51	-0.09	0.02
Iowa-2019	**0.09**	-1.03	-0.09	0.03	**0.09**	0.09	-0.07	0.05

transformer network is 32. As in Table. VII, LSTM based RankNet demonstrates consistently a slightly better performance over Transformer based implementation. We speculate that this is due to the small data size in our problem which limits the Transformer to obtain better performance.

H. Generalization to new races

In the left column of Table.VII, the models are trained by the Indy500 training set, then tested on other race data. In the right column, the models are trained by the training set from the same event. RandomForest, as a representative of the machine learning methods, has its performance drops badly in the left column, indicating its incapability of adapting to the new data. On the contrary, RankNet-MLP obtains decent performance even when testing on unseen races. It shows the advantages of RankNet model on generalizing to race data from the different data distribution.

Pocono-2018 is a special case where RankNet-MLP trained by Pocono is worse than the model trained by Indy500. As in Fig. 6, Pocono-2018 has a small RankChangesRatio where CurRank delivers good performance; moreover, Pocono-2018 has the largest RankChangesRatio distance to other races from the same event, which makes it harder in forecasting with the trained model by the other races in Pocono.

I. Evaluation of RankNet Model's Training Efficiency

When considering the deployment of RankNet in car racing events, continuous learning by incorporating racing data streams, and updating the model in real-time would be critical in rank position forecasting. In this section, we study the efficiency aspects of model training to answer the following questions:

1) What challenges exist to accelerate the training process?
2) Which device is preferred to this need?

We re-implement RankNet with Tensorflow that supports many kinds of devices as an accelerator, and conducted performance evaluation of the model training with hardware in Table VIII.

TABLE VIII: Specification of hardware used in the evaluation.

	CPU	GPU	VE
Model	Xeon Gold 6226	V100S-PCIe	Type 10BE
# of sockets	2	1	1
# of cores	12 x 2	5120	8
Memory B/W	131.13 GB/s x 2	1134 GB/s	1350 GB/s
Peak perf. (float)	1.996 TF x 2	16.4 TF	4.32 TF
Host processor	-	Xeon Gold 6226	

We run experiments on SX-Aurora TSUBASA [30] as a novel vector engine architecture. It is composed of an x86 processor and a PCI card called Vector Engine (VE). VE contains a vector processor; the length of the vector register is 256 elements, which is much larger than SIMD instructions of general purpose processors like x86. Though VE can execute a whole program, TensorFlow for SX-Aurora TSUBASA [7] uses VE as an accelerator that executes the functions offloaded from TensorFlow that is running at x86. In this research, we extended TensorFlow for SX-Aurora TSUBASA to support LSTM. As is shown in Table VIII, CPU, GPU, and VE have different characteristics. Both GPU and VE have better memory bandwidth and peak performance than CPU. GPU has better peak performance than VE; VE has better memory bandwidth than GPU.

Figure 9 shows the structure of LSTM that is the main component of RankNet. Figure 9 (a) depicts an LSTM cell. It consists of multiple primitive operations such as matmul, bias add, and other element-wise operations. LSTM has recurrent self connection, which can be represented as a loop as shown in Fig. 9 (a). To avoid loop overhead, TensorFlow has functionality to unroll the loop like Fig. 9 (b). We used the same batch size 32 as used in the model evaluation. In this case, the execution cost and internal parallelism of each operation becomes small, which makes it difficult to get better performance with an accelerator. Increasing the batch size alleviates this situation, but scarifies generalization performance.

Figure 10 shows the performance of training speed of Rank Model, which is the main training part of RankNet. Bars with label CPU, GPU, and VE show performance of the normal configuration with loop unrolling. As it shows, accelerators cannot improve the performance compared to CPU in this case. This is because the overhead of calling these operations at an accelerator becomes significant and cannot be amortized by speed up of parallel execution of these operations.

Performance of VE is better than GPU. This is because TensorFlow for SX-Aurora TSUBASA has functionality of combining the operation offloading. That is, operations to be offloaded are queued until the result is required; then multiple operations are offloaded at a time. It reduced offloading overhead, but the performance of VE is still lower than CPU.

To improve this situation, cuDNN library for GPU has the functionality to offload unrolled LSTM cells as one operation [5], [9]. Within this operation, multiple matmuls across the timesteps are merged into a larger one to increase the parallelism. By combining the small operations into one, offloading

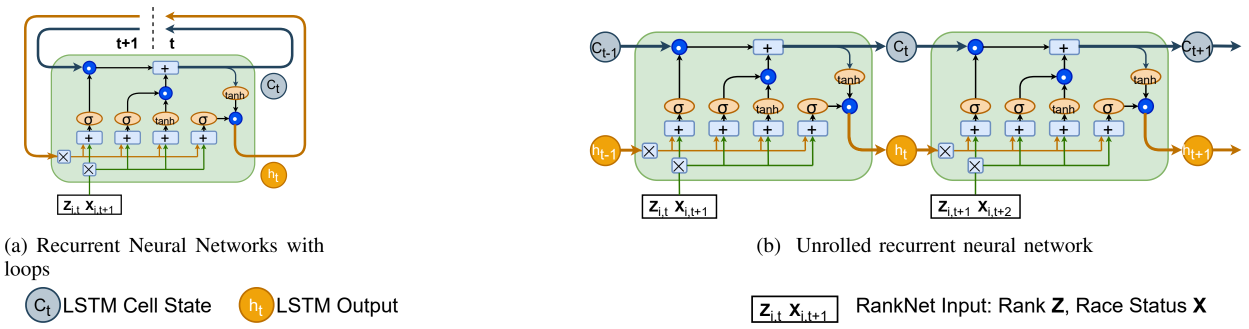


Fig. 9: LSTM cell has recurrent self-connection, which can be represented as a loop as shown in (a). To avoid loop overhead, TensorFlow has functionality to unroll the loop like (b).

overhead can be reduced. We added the same functionality to TensorFlow for SX-Aurora TSUBASA in a library called vednn. They are shown as GPU (cuDNN) and VE (vednn) in Figure 10.

In both cases, the performance of training speed has improved and is better than CPU. The performance of VE is still better than GPU. This is because the remaining offloading overhead is smaller in the case of VE with combined offloading, and memory bandwidth is more important than peak performance in this size of the computation.

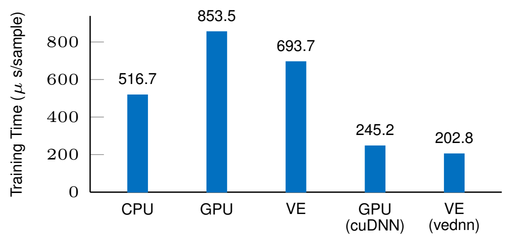


Fig. 10: Training time/sample (μs). Due to the reduced offloading cost and higher memory bandwidth, the VE outperforms the GPU.

V. Related Work

Forecasting in general: *Decomposition to address uncertainty.* Decomposition and ensemble are often used to separate the uncertainty signals from the normal patterns and model them independently. [20] utilizes the Empirical Mode Decomposition [16] 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 [24], winner of M4 forecasting competition [4], hybrids exponential smoothing to capture non-stationary trends per series, learns global effects by RNN, and ensembles the outputs finally. 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.

Modeling extreme events. Extreme events [17] are featured with the rare occurrence, difficult to model, and their prediction is probabilistic. Autoencoder shows improved results in capturing complex time-series dynamics during *extreme events*, such as [18] for uber riding forecasting and [31] which decomposes normal traffic and accidents for traffic forecasting. [12] proposes to use a memory network with attention to capture the *extreme events* pattern and a novel loss function based on extreme value theory. We have classified the *extreme events* in car racing with different categories in our work, and modeled the more predictable pit stops in normal laps by MLP with probabilistic output. Further exploring autoencoder and memory network can be one of our future works.

Express uncertainty in the model. [13] first proposed to model uncertainty in deep neural networks by using dropout as a Bayesian approximation. [31] followed this idea and successfully applied it to large-scale time series anomaly detection at Uber. Our work follows the idea in [22] that parameterizes a fixed distribution with the output of a neural network. [28] adopts the same idea and apply it to weather forecasting.

Car racing forecasting: *Simulation-based method:* Racing simulation is widely used in motorsports analysis [14] [10] [1]. A racing simulator models different factors that impact lap time during the race via equations with fine-tuned parameters. [14] presents a simulator that reduces the race time calculation error to around one second for the 2017 Formula 1 Etihad Airways Abu Dhabi Grand Prix. But the pit stop information for every driver is required as input.

Machine learning-based method: [25] [11] is a series of work forecasting the decision-to-decision loss in rank position for each racer in NASCAR. [25] 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. Our work aims to build forecasting that relies less on domain knowledge and focuses on investigating the pit stop model.

VI. Conclusion

This paper investigates deep learning to solve the challenging problem of modeling time-series data with high uncertainty and extreme events. With the IndyCar racing data, we find that the model decomposition based on the cause-and-effect relationship is critical to improving the rank position forecasting performance. We compare several state-of-the-art deep forecasting models: DeepAR, DeepState, DeepFactors, and N-BEATS. The results show that they cannot perform well on the global dependency structure. Therefore, we propose RankNet, a combination of the encoder-decoder network and a separate MLP network capable of delivering probabilistic forecasting, to model the pit stop events and rank position in car racing. In this way, we incorporate domain knowledge to enhance the deep learning method. Our proposed model achieves significantly better accuracy than baseline models in the rank position forecasting task. The advantages of needing fewer feature engineering efforts and providing probabilistic forecasting have enabled us with refined racing strategy optimizations.

There are several future directions for this work. Since there are a few related work, racing car data sets and their performance evaluation in this paper can contribute to the autonomous racing challenge for automobile, robotics, and automation forecasting. Car racing is an event with observed data changing in real-time and a major challenge lies in the lack of training data on extreme events. Applying transfer learning in this problem could be one important direction of future work.

VII. Acknowledgements

We gratefully acknowledge support from the NEC Grant for High-Performance Systems and Analytics for Big Data, NSF Grant 1918626 for Expedition: Global Pervasive Computational Epidemiology, and NSF Grant 1835631 for CINES: A Scalable Cyberinfrastructure for Sustained Innovation in Network Engineering and Science. We also appreciate the system support offered by FutureSystems.

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 Understanding-The-Sport. https://www.indycar.com/Fan-Info/INDYCAR-101/Understanding-The-Sport/Timing-and-Scoring. visited on 04/15/2020.

[4] M4 Competition. https://forecasters.org/resources/time-series-data/m4-competition/. visited on 04/15/2020.

[5] Optimizing Recurrent Neural Networks in cuDNN 5. https://devblogs.nvidia.com/optimizing-recurrent-neural-networks-cudnn-5/.

[6] PitStop. https://en.wikipedia.org/wiki/Pit_stop. visited on 04/15/2020.

[7] TensorFlow for SX-Aurora TSUBASA. https://github.com/sx-aurora-dev/tensorflow.

[8] 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*, June 2019.

[9] J. Appleyard, T. Kocisky, and P. Blunsom. Optimizing performance of recurrent neural networks on gpus. *arXiv preprint arXiv:1604.01946*, 2016.

[10] 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.

[11] C. L. W. Choo. *Real-time decision making in motorsports: analytics for improving professional car race strategy*. PhD Thesis, Massachusetts Institute of Technology, 2015.

[12] 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*, pages 1114–1122, New York, NY, USA, 2019.

[13] 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.

[14] 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.

[15] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[16] 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.

[17] H. Kantz, E. G. Altmann, S. Hallerberg, D. Holstein, and A. Riegert. Dynamical interpretation of extreme events: predictability and predictions. In *Extreme Events in Nature and Society*, pages 69–93. Springer, Berlin, Heidelberg, 2006.

[18] 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.

[19] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *Proceedings of International Conference on Learning Representations(ICLR)*, 2020.

[20] X. 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.

[21] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski. Deep state space models for time series forecasting. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7785–7794. Curran Associates, Inc., 2018.

[22] D. Salinas, V. Flunkert, and J. Gasthaus. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *arXiv:1704.04110 [cs, stat]*, Apr. 2017.

[23] 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.

[24] 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.

[25] T. Tulabandhula. *Interactions between learning and decision making*. PhD Thesis, Massachusetts Institute of Technology, 2014.

[26] 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.

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[28] 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*, pages 2087–2095, 2019.

[29] 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.

[30] Y. Yamada and S. Momose. Vector engine processor of NEC's brand-new supercomputer SX-Aurora TSUBASA. In *30th Symposium on High Performance Chips*, pages 19–21, 2018.

[31] 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.