*Transactions of KSAE, Vol. 32, No. 10, pp.843-852 (October, 2024)*




pISSN 1225-6382 / eISSN 2234-0149
**DOI http://dx.doi.org/10.7467/KSAE.2024.32.10.843**

# 위험 시나리오에 대한 트랜스포머 네트워크 기반 자율주행차량의 충돌 및 궤적 예측 알고리즘 개발

**이 성 우[1] · 정 영 훈[2] · 송 봉 섭[*1,2]**

아주대학교 기계공학과[1] · 아주대학교 D.N.A.융합학과[2]

# Multi-Task Prediction of Collision and Trajectories Based on Transformer Network for Safety-Critical Scenarios of Automated Vehicles

Sungwoo Lee[1] · Younghun Jeong[2] · Bongsob Song[*1,2]

[1]*Department of Mechanical Engineering, Ajou University, Gyeonggi 16499, Korea*
[2]*Department of Data, Networks, and AI, Ajou University, Gyeonggi 16499, Korea*
***(Received 5 June 2024 / Revised 11 July 2024 / Accepted 9 August 2024)***

**Abstract :** This research proposed a method for predicting collisions and trajectories using a transformer network with parallel computing capabilities for multiple vehicles. The accurate prediction of the driving trajectories of surrounding vehicles is essential for the decision-making processes of autonomous vehicles(AVs). Furthermore, the ability to predict imminent collisions can significantly enhance the safety of AVs. However, although several studies have addressed these issues individually, it is rare to find research that tackles both problems simultaneously. Hence, to facilitate the multitasks of collision prediction and trajectory prediction, we modified the prediction head associated with the final layer of the network to enable multitasking capabilities. This study explored two model architectures: one with an encoder-decoder structure and another with only a decoder. The performance of the proposed algorithm is compared with existing algorithms in the literature. The results demonstrate that our suggested algorithm outperforms previous methods in terms of parallelization.

**Key words** : Collision prediction(충돌 예측), Trajectory prediction(궤적 예측), Transformer(트랜스포머), Deep learning (딥러닝), Risk assessment(위험도 평가), Scenario-based assessment(시나리오 기반 평가)

## Nomenclature

| | |
|---|---|
| $x$ | : longitudinal position of surrounding vehicle |
| $y$ | : lateral position of surrounding vehicle |
| $v_x$ | : longitudinal velocity of surrounding vehicle |
| $v_y$ | : lateral velocity of surrounding vehicle |
| $a_x$ | : longitudinal acceleration of surrounding vehicle |
| $\psi$ | : heading angle of surrounding vehicle |
| $w$ | : width of surrounding vehicle |
| $l$ | : length of surrounding vehicle |
| $CP$ | : collision probability |
| $\omega$ | : input horizon |
| $\tau$ | : prediction horizon |
| $Q$ | : query matrix for attention layer |
| $K$ | : key matrix for attention layer |
| $V$ | : value matrix for attention layer |
| $W^Q$ | : weight matrix for query matrix |
| $W^K$ | : weight matrix for key matrix |
| $W^V$ | : weight matrix for value matrix |
| $N_e$ | : number of encoders |
| $N_d$ | : number of decoders |
| $h$ | : number of heads |

*Corresponding author, E-mail: bsong@ajou.ac.kr

## 1. 서 론

전 세계적으로 상용화되는 차량의 운전자지원시스템(Advanced driver assistance system) 또는 자율주행기술 보급 확대에 따라 카메라, 레이더 등의 환경 센서 장착이 증가하고 있다. 하지만 자율주행 기능이 고도화되더라도 센서나 프로세서의 오작동, 신호 위반과 같은 예기치 못한 상황에서 사고가 발생할 수 있다. 이 때 안전성 확보를 위한 능동 또는 수동 안전 시스템의 작동을 위해서는 주변 차량에 대한 충돌 및 궤적을 예측하는 기술이 필수적으로 필요하다.

충돌 예측 기술은 크게 모델 기반 접근 방법과 데이터 기반 접근 방법으로 분류할 수 있다. 모델 기반 접근 방법은 크게 결정론적(Deterministic), 확률론적(Probabilistic) 방법으로 구분되며 위험도 인자와 사전에 설정된 임계값(Threshold)을 사용하여 충돌을 예측한다. 결정론적 접근 방법은 차량의 움직임이 물리법칙을 따른다고 가정하며 일반적으로 기구학적 모델(Kinematic model) 또는 동역학적 모델(Dynamic model)을 사용하여 위험도를 계산한다.[1] 확률론적 접근 방법은 충돌이 예상되는 영역에 해당하는 확률을 누적하여 Collision Probability(CP)를 계산하며 측정값이나 동적 모델의 불확실성을 고려할 수 있다.[2] 이러한 모델 기반 접근 방법은 자동 긴급 제동장치(Automatic emergency braking)와 같이 양산되는 차량의 주행 보조 장치에 오랜 기간 사용되고 검증되어 왔지만 일반적으로 위험도 인자와 정해진 임계값을 사용하여 판단하기 때문에 다양한 시나리오에 대해 판단 성능이 저하될 수 있으며 임계값을 경험적으로 보정(Calibration)하는데 많은 시간이 소요되는 한계점이 존재한다. 최근 딥러닝 모델의 발전과 함께 데이터 기반 접근 방법을 통해 충돌 예측 성능을 높이는 연구가 진행되고 있다. Multi-Layer Perceptron(MLP)을 기반으로 주변 차량의 거리, 속도, 가속도와 같은 상태값(State)을 입력 특징(Input feature)으로 사용하여 전방 차량 추돌 경보 기술이 제안되었다.[3] 또한 Convolution Neural Network(CNN)를 기반으로 시계열 데이터를 이미지화 하는 기법인 Gramian angular summation field matrix를 사용해 충돌을 예측하는 연구도 찾아볼 수 있다.[4] 이러한 연구들은 모델 기반 알고리즘 대비 우수한 성능을 보이지만 측정값에 노이즈가 존재하며 복잡한 주행 상황이 포함된 실도로 데이터에 대해 오판단(False alarm)이 발생한다. 이러한 문제를 개선하기 위해 모델 기반 알고리즘과 데이터 기반 알고리즘을 통합한 연구가 제안되었다. 특히, 위험도(CP), 궤적 예측, 위치와 속도 등의 상태 값을 모두 사용하여 Abstracted bird's eye view 이미지를 생성한 뒤 CNN을 사용하는 CP-CNN 알고리즘을 기반으로 충돌을 예측했다. 이를 통해 사고 시나리오를 포함하는 시뮬레이션 데이터에 대해 높은 판단 성능을 유지하며 실도로 데이터에 대한 오판단이 개선됨을 보였다.[5]

궤적 예측 기술은 최근 데이터 기반 접근 방법으로 활발하게 연구되고 있다. Long-Short Term Memory(LSTM) 기반 인코더-디코더(Encoder-decoder) 구조와 주변 차량의 위치, 속도 정보를 사용하여 궤적을 예측하는 연구가 존재한다.[6] 추가로 궤적 예측 성능 향상을 위해 어텐션(Attention) 기술을 사용한 연구가 존재한다. 인코더-디코더 구조에 어텐션 메커니즘을 추가하여 주변 차량 간의 상호작용을 모델링하였으며 기존 LSTM 인코더-디코더 대비 높은 성능을 보였다.[7] 추가로 자연어 처리 분야에서 큰 성공을 거둔 트랜스포머(Transformer)가 궤적 예측을 위해 사용되고 있다. 트랜스포머 모델은 인코더-디코더 구조에 어텐션 메커니즘만을 사용한다. 이로 인해 시계열 입력 데이터의 장, 단기 의존성(Long and short-range dependency)을 동시에 처리할 수 있으며, 입력되는 순차(Sequence) 데이터를 순서대로 처리하는 것이 아니라 한 번에 처리함으로써 더 빠른 실행 속도를 보여준다.[8] 하지만 기존의 트랜스포머 모델은 디코더에서 입력된 데이터를 병렬적으로 출력하지 못하는 문제가 있다. 이를 개선하기 위하여 Non-Autoregressive Transformer(NAT) 모델이 제안되었고, 인코더의 입력 값을 복사하여 디코더에 넣는 방식을 사용하는 가장 기본적인 모델에서 기존 방식과 거의 동일한 성능을 가지며 10배 이상의 빠른 실행 속도를 보였다.[9] 이후에 트랜스포머 모델의 디코더만 사용하여 자연어를 처리하는 기술이 제안되었다. 인코더 블록을 제거함으로써 기존 인코더-디코더 모델에 필요한 파라미터 수를 거의 절반으로 줄일 수 있었으며 인코더-디코더 구조를 갖는 트랜스포머 대비 우수한 성능을 보였다.[10,11]

다중 작업을 단일 모델을 이용하여 처리하기 위하여 Muti-Task Learning(MTL) 접근방법이 적용되고 있다. 이러한 MTL기반 접근 방법은 단일 작업에 대해 단일 모델을 사용한 경우보다 데이터, 학습 시간, 연산 속도 관점에서 효율적인 것으로 알려져 있다.[12] 이러한 특징을 이용하여 이미지 처리 분야에서 Depth estimation, Semantic segmentation, 2D detection, Edge detection과 같은 여러 작업을 동시에 처리하는 연구도 찾아볼 수 있다. 동일한 이미지 입력에 대해 공유된 인코더와 작업 별로 분리된 디코더 구조를 제안하였으며, 이때 관련도가 높은 작업에 대하여 동시에 학습할 경우 모델의 성능이 더 향상되는 특징을 가지고 있다.[13] 최근 차량 궤적 예측 분야에서도

MTL을 이용하여 예측 궤적과 거동 의도를 동시에 출력하는 연구가 진행되고 있다. 주변 차량의 과거 상태 정보를 동일한 입력으로 하여 작업마다 분리된 디코더를 통해 결과를 출력한다.[14] 이러한 연구에서 영감을 받아 본 연구에서는 트랜스포머 모델과 디코더 모델을 사용하여 충돌 및 궤적 예측 결과를 동시에 출력하는 기술을 제안하고자 한다.

## 2. 기존 연구 및 문제 정의

모델 기반 접근 방법과 데이터 기반 접근 방법을 통해 충돌을 예측하는 연구가 제안되어왔다. 모델 기반 알고리즘은 데이터 기반 알고리즘 대비 충돌 예측 성능이 떨어지는 것으로 알려져 있고 데이터 기반 알고리즘은 실도로 데이터에 대한 오판단 성능이 저하되는 것으로 알려져 있다. 이후 모델 기반 알고리즘과 데이터 기반 알고리즘을 통합하여 충돌을 판단하는 CP-CNN 알고리즘이 충돌 예측과 오판단 성능에 있어 개선된 결과를 보였다. 해당 알고리즘은 주변 차량에 대한 궤적 예측과 위험도를 계산한다. 이후 추상화(Abstraction) 이미지에 각 차량의 예측된 위치와 위험도 정보를 순차적으로 표기한다.

Fig. 1은 Cut-in하는 파란색 상대 차량이 주행하는 시뮬레이션 데이터에 대한 추상화 결과를 나타낸다. 추상화 이미지에서 초록색 곡선은 차선을 의미하며 중앙 하단의 검은색 사각형은 자차량을 뜻한다. 빨간색으로 채워진 사각형은 상대 차량의 현재 위치, 크기, 폭, 위험도 정보를 반영한 것이며 빨간색 빈 사각형은 1s 예측 위치를 표기한 것이다.

반면 Fig. 2에서는 주변 차량이 6대가 존재하는 실도로 데이터에 대한 추상화 결과를 보여주고 있다. 이와 같이 객체가 늘어나면 각 차량에 대해 물리 기반 궤적 예측 및 위험도 알고리즘 계산 시간이 증가하며 추상화 이미지에 모든 객체의 정보가 표기되는 시간이 추가적으로 증가한다. 따라서 자차량 주변 객체 숫자가 늘어남에 따라 충돌 예측에 필요한 연산시간이 급격하게 증가될 수 있다.

본 연구에서는 이 문제를 완화하기 위해 주변 객체에 대해 병렬 연산(Parallel computation)이 가능한 트랜스포머 네트워크를 사용하여 객체 수 증가에 따라 연산 시간이 급격하게 증가하는 문제를 해결하고자 한다. 트랜스포머 네트워크의 경우 CP-CNN과 다르게 주변 객체 해당 시계열 정보를 추상화하는 과정이 필요 없으며 입력 특징을 결합(Concatenate)한 입력 형태를 사용할 수 있다. 따라서 전체 주변 차량에 대한 관측 가능한 정보를 결합한 입력을 기반으로 주변 차량에 대한 충돌 및 궤적 예측 결과를 출력하고자 한다.

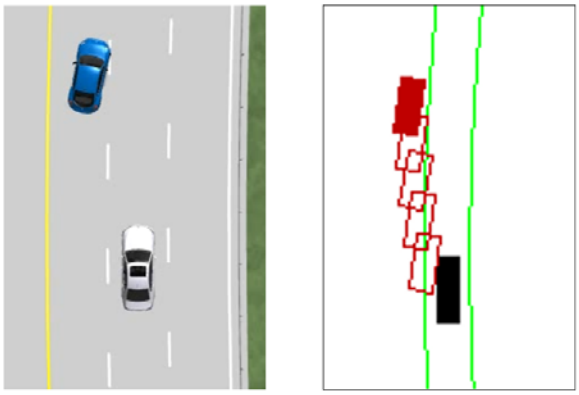


Fig. 1 Snapshot(left) and abstraction(right) of simulation data

Fig. 2 Snapshot(left) and abstraction(right) of real-driving data

## 3. 충돌 및 궤적 예측 알고리즘

### 3.1 네트워크의 입출력 정의

제안하는 네트워크는 주변 차량 전체에 대해 관측 가능한 궤적을 결합한 $\chi$를 입력으로 사용하여 주변 차량에 대한 충돌 및 궤적 예측 결과 $Y$를 출력하며 다음과 같이 정의된다.

$$\chi = \{\chi_t^i | i \in N\} \tag{1}$$

$$Y = \{Y_t^i | i \in N\} \tag{2}$$

여기서 $N$은 $t$시점의 전체 차량 숫자를 의미한다. $t$ 시점의 $i$ 번째 차량에 대한 입력은 다음과 같이 정의된다.

$$\chi_t^i = [x\ y\ v_x\ v_y\ a_x\ w\ l\ \psi]_{[t-(\omega-1):t]}^i \in \mathcal{R}^{\omega \times 8} \tag{3}$$

$t$ 시점의 $i$ 번째 차량에 대한 모델의 출력 벡터는 다음과 같이 정의된다.

$$Y_t^i = [x\ y\ c\ ]_{[t+1:t+\tau]}^i \in \mathcal{R}^{\tau\times 4} \tag{4}$$

충돌 예측($c$)은 안전(Safe)과 자차량에 충돌하는 경우(Precrash)를 포함하며 다음과 같이 정의된다.

$$c = [c_1\ c_2] = [\text{Safe Precrash}] \tag{5}$$

### 3.2 네트워크의 구조

인코더-디코더 트랜스포머와 디코더 트랜스포머(Decoder-only transformer)를 사용하는 충돌 및 궤적 예측 모델을 제안한다. 다중 차량들의 과거 정보(위치, 속도, 가속도, 폭, 길이, 헤딩각)를 입력으로 받아 충돌과 미래 궤적을 예측한다.

#### 3.2.1 인코더-디코더 트랜스포머 모델

Fig. 3에서 보는 것과 같이 인코더-디코더 트랜스포머 모델은 $N_e$개의 인코더와 $N_d$개의 디코더를 가지고 있으며 각각은 어텐션, Feed forward, Residual connection 모듈로 구성되어 있다. 인코더와 디코더에는 동일한 입력 값을 복사하여 각각 Positional encoding을 적용한 값을 입력하는 기본적인 NAT모델의 입력방식을 사용한다. 입력 간의 연관성을 찾는 기능은 어텐션 모듈에서 이루어진다. 각각의 멀티 헤드 어텐션(Multi-head attention)은 $h$개의 가중 내적 어텐션(Scaled dot product attention)으로 이루어져 있다. 가중 내적 어텐션은 아래의 식과 같이 쿼리(Query) 행렬 $Q$와 키(Key)행렬 $K$의 내적을 계산하는 과정을 포함한다.

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)\cdot V \tag{6}$$

위 과정을 $h$번의 반복을 통해 결합(Concatenate)된 출력은 아래와 같다.

$$\text{Multihead}(Q,K,V) = \text{Concat}(\text{head}_1,\cdots,\text{head}_\text{h})W^O \tag{7}$$

여기서 $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$이며
$W_i^Q \in \mathcal{R}^{d_{model}\times d_k}, W_i^K \in \mathcal{R}^{d_{model}\times d_k},$
$W_i^V \in \mathcal{R}^{d_{model}\times d_v}, W^O \in \mathcal{R}^{hd_v\times d_{model}}, i = 1,\cdots,h$ 이다.

주어진 입력에 대해 트랜스포머 모델이 서로 다른 시점에 대해 셀프 어텐션(Self attention)을 계산하며 이를 통해 시계열에 대한 패턴 정보를 획득한다. Fig. 3 우측의 디코더는 궤적과 충돌 예측 결과를 출력하며 2개의 멀티 헤드 어텐션 모듈을 포함한다. MTL을 적용하기 위하여 디코더의 마지막에 연결된 Fully Connected(FC) layer는 2개로 구성되어 있다. 각각 궤적과 충돌 예측 결과를 출력하며, 예측 지평선(Prediction horizon) 1s에 대하여 궤적 예측에 해당하는 Tensor의 크기는 20 × 2이며 20은 예측 지평선, 2는 예측 궤적의 종, 횡방향 위치를 의미한다. 나머지 20 × 2 Tensor는 충돌 예측 해당 결과이며 Softmax 함수를 적용하여 최종 충돌 예측 결과를 출력한다.

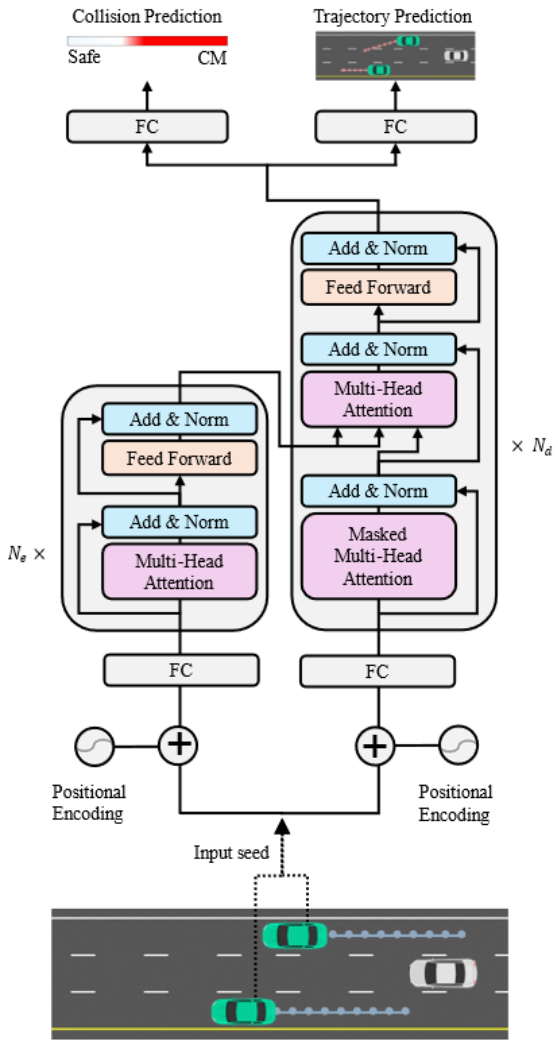



Fig. 3 Model illustration of transformer network for collision and trajectory prediction

#### 3.2.2 디코더 트랜스포머 모델

충돌 및 궤적 예측을 위한 두번째 모델로 디코더 트랜스포머모델을 사용한다. 인코더-디코더 트랜스포머 모

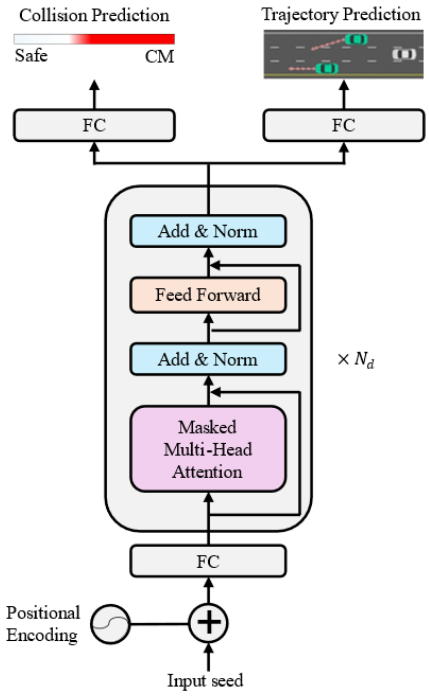



Fig. 4 Model illustration of decoder-only transformer for collision and trajectory prediction

델에서 인코더 블락을 제거하고 디코더만 사용하는 모델이다. 인코더 블락을 제거함으로써 기존 트랜스포머에서 사용되는 모델 파라미터를 거의 절반으로 줄일 수 있다.[10] Fig. 4에서 볼 수 있듯이 인코더-디코더 트랜스포머의 디코더 블락과 다르게 Masked multi-head attention 이후 바로 Feed forward로 들어가는 구조로 이루어져 있다. 인코더에서 전달받는 입력이 없기 때문에 기존 디코더의 Multi-head attention layer가 생략되어 있다. 인코더-디코더 트랜스포머 모델과 동일하게 MTL을 위한 2개의 FC를 가지며, 동일한 출력값을 갖는다.

### 3.3 네트워크의 구현

제안된 모델은 파이토치(Pytorch) 라이브러리를 이용하여 파이썬(Python)으로 구현하였다. 인코더-디코더 트랜스포머 모델의 경우 $N_e = 2, N_d = 1, d_{model} = 32, h = 2$가 사용되었으며 디코더 트랜스포머 모델의 경우 $N_d = 4, d_{model} = 32, h = 2$가 사용되었다. 최적화는 아담 옵티마이저(Adam Optimizer)를 사용하였다.[15] 학습률(Learning rate)은 OneCycleLR 스케쥴러(Scheduler)를 사용하여 적용하였다. 이를 통해 초기 학습률 0.001에서 최대 학습률 0.01까지 올라간 후 초기 학습률보다 낮은 학습률로 학습을 진행하였다. 배치 사이즈(Batch size)는 1,024로 정의했다. 네트워크에 사용된 파라미터는 경험적으로 여러 후보를 적용하여 그중 상대적으로 높은 성능을 보이는 값으로 선정했다. 추가로 Intel Core i9-12900KF CPU 3.19 GHz and NVIDIA RTX 3090 Ti 하드웨어 환경에서 학습 및 평가를 진행했다.

MTL에서 $M$개의 작업을 학습할 때 사용하는 다음과 같은 손실함수를 사용한다.

$$\mathcal{L}_{tot}(\chi, Y_{1:M}) = \sum_{i=1}^{M} \lambda_i \mathcal{L}_i(\chi, Y_i) \tag{8}$$

여기서 $\chi$는 입력 값이며 $Y_i, (i = 1, 2, \cdots M)$는 작업 별 라벨을 의미한다. 최종 손실함수 $\mathcal{L}_{tot}$는 각 작업 별 가중치 $\lambda_i$를 손실함수 $\mathcal{L}_i$에 곱한 값을 더하여 구한다. 제안된 모델의 학습을 위하여 사용하는 손실함수에서 $M$ 값은 2이며, 충돌 예측과 궤적 예측의 손실함수에 가중치를 곱한 값을 더하여 사용한다. 충돌 예측은 분류(Classification) 문제로 생각될 수 있다. 분류 문제에서 일반적으로 사용되는 손실 함수인 Cross entropy loss를 본 연구에서 사용하였으며 수식은 아래와 같다.

$$\mathcal{L}_1 = -\sum_i^C c_i \log \hat{c}_i \tag{9}$$

여기서 $c_i$ 는 충돌 여부에 대한 Ground Truth(GT) 이고, $\hat{c}_i$은 충돌 예측 결과이다.

예측 궤적과 실제 궤적 사이의 차이를 반영하는 Mean Squared Error(MSE) loss를 궤적 예측에 사용하였으며, 해당 손실 함수의 수식은 아래와 같다.

$$\mathcal{L}_2 = \frac{1}{\tau}\sum_1^\tau((x_t - \hat{x}_t)^2 + (y_t - \hat{y}_t)^2) \tag{10}$$

최종 손실 함수는 충돌 예측 손실 함수와 궤적 예측 손실 함수를 결합한 값을 사용했다.

## 4. 실험 결과

실험결과는 사고를 포함하는 시뮬레이션 데이터와 센서 노이즈를 포함하며 높은 복잡도를 갖는 실도로 주행 데이터를 사용했다. 제안하는 모델은 충돌 및 궤적 예측을 동시에 출력하므로 충돌 예측 성능에 대해 기존 알고리즘과 성능 비교한 이후 궤적 예측 성능에 대해 비교한다. 먼저 충돌 예측 성능에 대한 평가를 위해 사용한 대조군은 다음과 같다. 모델 기반 충돌 예측 알고리즘은 Collision probability를 사용한다.[2] 데이터 기반 충돌 예측 알고리즘은 CP-CNN를 포함한다.[5] 이어서 궤적 예측을 위한 대조군은 데이터 기반 알고리즘인 Vanilla LSTM과 Sequence-to-sequence를 포함한다.[6]

### 4.1 데이터셋

충돌을 예측하는 네트워크 개발을 위해서는 사고를 포함하는 데이터가 필요하다. 하지만 현재 자율주행기술 개발을 위해 생성된 공개 데이터셋에는 사고 상황을 포함하고 있지 않아 시나리오 기반 시뮬레이션 데이터를 생성하여 사용하였다. 시뮬레이션 데이터만 사용할 경우 실도로 주행 데이터에 대한 판단 성능이 저하될 수 있기 때문에 환경 센서가 장착된 실험 차량을 통해 취득한 데이터를 추가로 사용하였다.

위험(Safety-critical) 시뮬레이션 데이터 취득을 위한 시나리오 생성 과정은 크게 2가지 단계로 구분할 수 있다.[16] 먼저 시나리오 생성(Scenario generation) 단계에서 교통 사고 데이터 통계 분석을 통해 기본(Logical) 시나리오를 생성한다. 기본 시나리오 도출을 위해 2012년부터 2014년까지 국내 교통사고 분석 시스템(TAAS)과 2007년부터 2018년까지 유럽의 교통 사고 데이터(The initiative for the global harmonization of accident data)의 차대차 사망 사고 데이터를 분석하여 5개의 비교차로(Non-junction) 시나리오를 선정하였다. 선정된 시나리오는 Vehicle following, Vehicle cutting in, Lead vehicle stopped, Vehicles changing lanes, Backing up으로 5가지 시나리오이며 미국 National Highway Traffic Safety Administration(NHTSA)의 Precrash scenario typology를 기반으로 명명하였다.[17,18]

그리고 시나리오 선정 단계(Scenario selection)에서 파라미터 공간(Parameter space)에 대한 N-wise 샘플링을 통해 위험 시나리오 데이터를 생성하였다. 생성 과정에서 Support vector machine을 기반으로 안전한 시나리오를 줄이는 기법을 적용하였다.[19] 시뮬레이션 데이터의 경우 6,468개의 상세 시나리오를 사용하였으며 이는 약 47시간의 주행시간과 약 3,465 km의 주행거리에 해당한다.

실도로 주행 데이터는 Fig. 5와 Table 1에서 볼 수 있듯이 전방 레이더, 전방 비전, 코너 레이더, GPS가 장착된 실험차량을 이용하여 획득하였다. 취득한 센서 데이터는 산업용 PC에 저장되었으며 센서융합 알고리즘을 거쳐 주변 차량에 대한 융합 트랙을 생성하였다. 이를 통해 개별 센서에서 출력되는 정보 대비 정확한 상태 정보를 사용하여 충돌 및 궤적 예측 기술을 개발하였다. 데이터 취득을 위해 서로 다른 7명의 운전자가 실험 차량을 국내의 도심과 고속도로 환경에서 주행하여 주변 환경에 대한 데이터를 획득하였으며 주행 경로는 Fig. 6에 나타낸 것과 같다. 실도로 주행 데이터는 약 1,773 km, 약 25시간에 해당한다. 시뮬레이션 데이터는 1개 데이터의 길이가 약 20s에 해당하지만 실도로 데이터는 1개 데이터의 길이가 2분에 해당한다. 따라서 학습 및 평가 시에 사용되는 데이터의 길이를 유사하게 맞추기 위해 실도로 데이터를 20s 길이의 Snippet으로 분리했으며 총 3,953개의 Snippet을 사용하였다.

생성한 시뮬레이션 데이터와 실도로 주행 데이터를 랜덤으로 분류하여 학습과 평가에 사용하였다. Table 2에서 볼 수 있듯이 데이터의 70 %를 학습에 사용하고 나머지 30 %를 평가에 사용하였다.

충돌에 대한 GT는 시뮬레이션 결과를 이용하여 자동으로 라벨링했다. 본 연구에서 시뮬레이션을 위해 IPG CarMaker를 사용했다.[20] IPG CarMaker에서는 자차량에 충돌 센서를 장착할 수 있으며 이 센서에서는 Sample period 마다 주변 객체와의 충돌 여부를 출력한다. 이 정

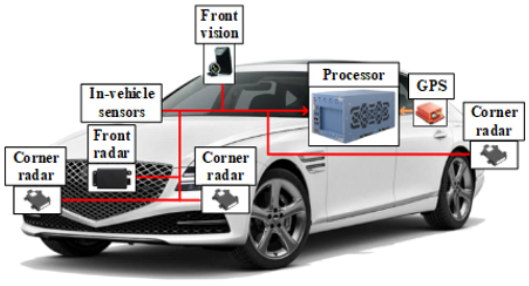



Fig. 5 Configuration of the experimental vehicle

Table 1 Specifications of perception sensors

| Sensor | | Detection range | Field of view |
|---|---|---|---|
| Corner radar | | 80 m | ±75 deg |
| Front radar | Short range | 60 m | ±45 deg |
| | Long range | 200 m | ±10 deg |
| Front vision | | 170 m | ±50 deg |


Fig. 6 Driving routes for experimental data collection

Table 2 Training and test data

| | Training | Test | Sum |
|---|---|---|---|
| Simulation data (number of concrete scenario) | 4,554 | 1,914 | 6,468 |
| Experimental data (number of snippet) | 2,768 | 1,185 | 3,953 |
| Sum | 7,322 | 3,099 | 10,421 |

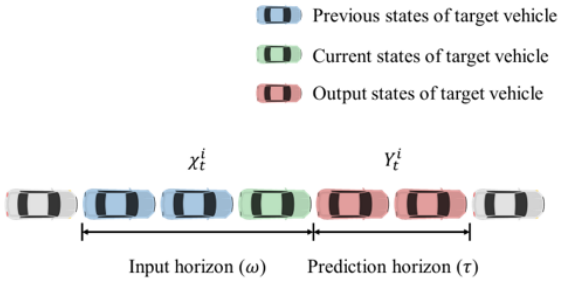



Fig. 7 Example of training set extraction from driving data

보를 이용하여 생성한 시뮬레이션 데이터에 대해 충돌 발생 유무와 충돌이 발생한 시점($t_i$)을 라벨링할 수 있다. 실도로 주행 데이터의 경우 사고가 발생하지 않으므로 전체 데이터가 안전으로 라벨링되었다.

$$c = \begin{cases} \text{Precrash}, & \text{if collision sensor on} \\ \text{Safe}, & \text{Otherwise} \end{cases} \tag{11}$$

궤적 예측을 위해 전체 데이터셋을 이용하여 Fig. 7과 같이 과거 궤적의 길이(Input horizon)와 예측 지평선 길이(Prediction horizon)를 만족하는 궤적을 생성하였다. Input horizon($\omega$)과 Prediction horizon($\tau$)는 모두 1s 로 설정하여 학습 데이터셋을 추출하였다. 이 때 사고를 포함하는 데이터에 대해서는 학습 데이터셋 추출을 아래와 같이 진행하였다. 마지막 메뉴버(Maneuver) 변경 시점($t_m$)과 충돌 전 특정 시점($t_i-\varepsilon$)을 비교하여 두 시점 중에서 충돌과 가까운 시점을 Precrash의 시작 시점($t_s$)으로 설정하였다. 이 시점부터 충돌 시점($t_i$) 사이에 해당되는 궤적을 충돌은 충돌 학습 데이터셋으로 나머지는 안전 학습 데이터셋으로 사용하였다. 이 때 $\varepsilon$의 값은 0.5s를 사용하였다.

$$t_s = \begin{cases} t_m, & t_m > t_i - \varepsilon \\ t_i - \varepsilon, & \text{Otherwise} \end{cases} \tag{12}$$

안전한 데이터에 대해서는 관심영역 내에 차량이 존재하는 시점에 대해 학습 데이터셋을 추출하였다.

### 4.2 평가 지표

충돌 예측의 평가를 위해 Confusion matrix를 사용한다. Confusion matrix에서 Positive는 실제 충돌이 발생하는 데이터, Negative는 안전한 데이터를 의미한다. 실제 충돌이 발생하는 시점과 충돌을 예측하는 시점의 비교를 통해 네트워크 출력 결과를 4가지로 분류할 수 있다. True Positive(TP)는 충돌 시점 이전 1.5s 내에 충돌로 예측한 경우를 의미한다. True Negative(TN)는 안전한 데이터를 안전하다고 예측한 경우에 해당한다. False Positive (FP)는 사고 발생 데이터에서 충돌 시점에서 1.5s 보다 일찍 충돌로 예측하거나 안전한 데이터에서 충돌을 예측한 경우에 해당한다. False Negative(FN)는 충돌 1.5s 전부터 충돌 시점 사이에 충돌을 예측하지 못한 경우에 해당한다. 위의 4가지 지표를 기반으로 다음의 평가 인자를 계산할 수 있다.[21] False Positive Rate(FPR)은 안전한 데이터를 위험하다고 예측한 비율이다. False Negative Rate(FNR)은 사고가 발생한 데이터를 안전으로 예측한 비율이다. Accuracy(ACU)는 전체 데이터에 대해 올바르게 예측한 비율이며 언급한 평가지표들은 아래와 같이 계산할 수 있다.

$$FPR = FP/(FP + TN) \tag{13}$$

$$FNR = FN/(FN + TP) \tag{14}$$

$$ACU = (TP + TN)/(TP + FN + FP + TN) \tag{15}$$

추가로 알고리즘이 충돌 전 얼마나 빠르게 충돌을 예측할 수 있는지 평가하기 위해 충돌 전 판단 시점($t_d$)을 사용한다. 충돌 전 판단 시점은 충돌 시점($t_i$)에서 알고리즘이 처음으로 충돌을 예측한 시점($t_c$)의 차이를 의미하며 아래와 같다.

$$t_d = t_i - t_c \tag{16}$$

궤적 예측 결과는 예측된 위치와 실제 위치의 차이를 나타내는 Root Mean Squared Error(RMSE)를 사용하여 평가했으며 해당 지표는 아래와 같다.[14]

$$\boldsymbol{RMSE} = \sqrt{\frac{1}{N_{test}}\left(\sum_{j=1}^{N_{test}} \frac{1}{\tau}\sum_{t=1}^{\tau}\left(x_t^j - \hat{x}_t^j\right)^2\right)} \tag{17}$$

여기서 $N_{test}$는 궤적 수를 의미하며 $x_t^j$는 $j$번째 궤적의 $t$ 시점일 때 실제 위치, $\hat{x}_t^j$는 $j$번째 궤적의 $t$시점일 때 예측된 위치를 의미한다.

### 4.3 평가

Table 3에서 사고 시나리오를 포함하는 시뮬레이션 데이터에 대한 충돌 예측 알고리즘의 성능을 볼 수 있다. 모델 기반 알고리즘인 Collision probability의 Accuracy가 가장 낮지만 연산 시간은 가장 적게 소요된 것을 볼 수 있다. 데이터 기반 알고리즘 중 CP-CNN은 가장 높은 Accuracy 성능을 보였으나 가장 많은 연산 시간이 소요되는 것을 확인할 수 있다. Accuracy 관점에서 CP-CNN 대비 인코더-디코더 트랜스포머 알고리즘은 약 2 %의 성능 저하가 발생되었으며, 디코더 트랜스포머 알고리즘의 경우 약 4 %의 성능 저하가 발생하였다. 평균 판단 시점은 CP-CNN 대비 트랜스포머 기반 알고리즘이 약 0.3s가 늦다. 이와 같이 트랜스포머 기반 알고리즘이 CP-CNN 대비 일부 성능 저하가 있지만 약 절반 정도의 연산 시간이 소요되었다.

실도로 주행 데이터에 대한 충돌 예측 알고리즘의 성능이 Table 4에 표기되어 있다. 자율주행차량의 상용화 관점에서 사용될 수 있는 모델 기반 알고리즘이 센서 노이즈가 존재하며 높은 복잡도를 갖는 실도로 주행 데이터에 대해 높은 성능을 보이는 것을 확인할 수 있다. CP-CNN과 디코더 트랜스포머 알고리즘은 Collision probability와 유사한 성능을 보인다. 인코더-디코더 트랜스포머 알고리즘은 Accuracy 관점에서 약 1~2 %의 성능이 저하되었다. 연산 시간 관점에서 CP-CNN이 제안하는 트랜스포머 알고리즘 대비 약 2~3배 더 소요되는 것을 확인할 수 있다. 제안하는 알고리즘의 경우 객체 수가 적은 시뮬레이션 데이터 대비 많은 객체를 포함하는 실도로 주행 데이터에 대해서 연산 시간이 크게 증가하지 않은 것을 확인할 수 있다.

궤적 예측에 대한 성능은 Table 5에서 확인할 수 있다. 제안하는 인코더-디코더 트랜스포머가 LSTM 대비 약 1~2 ms의 연산 시간이 증가하지만 종, 횡방향 위치오차는 모두 감소하였다. 그리고 Sequence-to-sequence 알고리즘 대비 인코더-디코더 트랜스포머 알고리즘이 종방향 위치 오차에 대해 좋은 성능을 보이지만 횡방향 위치 오차는 성능이 저하되었다. 하지만 약 절반의 연산 시간이 소요되었다. 반면 제안하는 디코더 트랜스포머는 LSTM과 유사한 연산 시간이 소요되지만 종, 횡방향 위치 오차에 대해 가장 낮은 성능을 보였다.

충돌 예측 알고리즘 중 성능이 높은 CP-CNN과 제안하는 인코더-디코더 트랜스포머 알고리즘의 주변 차량 대수에 따른 연산 시간이 Fig. 8에 나타나 있다. CP-CNN의 경우 주변 차량의 숫자가 증가할수록 연산 시간이 점진적으로 증가하다가 주변 차량이 8대가 되는 경우 약 10 ms가 급격하게 증가하였다. 반면 제안하는 알고리즘의 경우 충돌 예측과 궤적 예측을 동시에 수행하지만 주변 차량 대수가 증가하더라도 약 10 ms 근처의 연산 시간이 소요되는 것을 확인할 수 있다. 특히, 주변 차량이 8대인 경우에 두 알고리즘의 연산 시간이 약 20 ms까지 차이가 발생하는 것을 볼 수 있다.

Fig. 9는 5.75s에 사고가 발생하는 Cut-in 시나리오에 대한 충돌 및 궤적 예측 결과이다. Fig. 9(a) 좌측 그림을 보면 4.5s에 파란색 전방 차량이 하얀색 자차량의 왼쪽 차로에서 Lane following하고 있다. 제안하는 인코더-디코더 트랜스포머 네트워크가 해당 주행 상황을 안전하다고 판단하여 Bird's eye view에서 전방 차량이 하얀색을 띄고 있다. 궤적 예측에 대해서는 실제 주행 경로(빨간색 점)와 예측 궤적(파란색 점)이 큰 차이를 보이고 있지 않다. Fig. 9(b)는 5.1s에 전방 차량이 Cut-in을 하면서

Table 3 Performance comparison of collision prediction based on simulation data

| | *ACU* (%) | *FNR* (%) | *FPR* (%) | $Avg\ t_d$ (*s*) | Computing time(*ms*) |
|---|---|---|---|---|---|
| Collision probability[2)] | 90.9 | 8.8 | 9.4 | 0.78 | 1.2 |
| CP-CNN[5)] | 95.0 | 4.6 | 5.4 | 0.88 | 13.2 |
| Proposed (D) | 91.1 | 7.1 | 10.5 | 0.50 | 5.8 |
| Proposed (ED) | 93.3 | 3.2 | 9.8 | 0.52 | 7.58 |

Table 4 Comparison of collision prediction performance based on experimental data

| | *ACU* (%) | *FNR* (%) | *FPR* (%) | $Avg\ t_d$ (*s*) | Computing time(*ms*) |
|---|---|---|---|---|---|
| Collision probability[2)] | 99.2 | - | 0.8 | - | 2 |
| CP-CNN[5)] | 99.2 | - | 0.8 | - | 28.5 |
| Proposed (D) | 99.2 | - | 0.8 | - | 6.0 |
| Proposed (ED) | 98.8 | - | 1.2 | - | 8.01 |

Table 5 Comparison of trajectory prediction performance

| | $RMSE_x(m)$ | $RMSE_y(m)$ | Computing time(*ms*) |
|---|---|---|---|
| LSTM | 0.90 | 0.53 | 5.9 |
| Seq2Seq[6)] | 0.89 | 0.35 | 17.5 |
| Proposed (D) | 1.21 | 0.60 | 5.95 |
| Proposed (ED) | 0.83 | 0.45 | 7.84 |

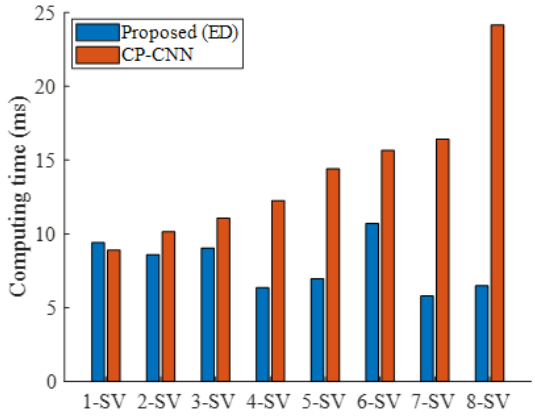
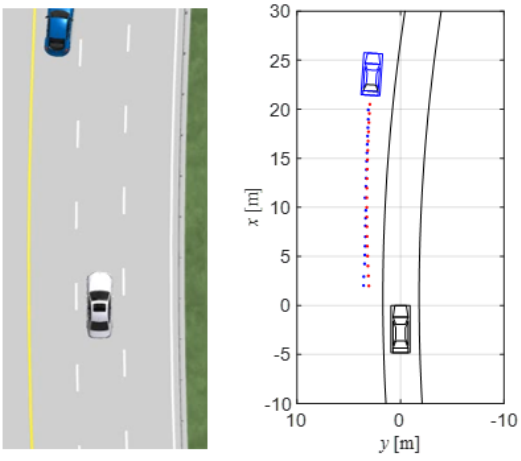
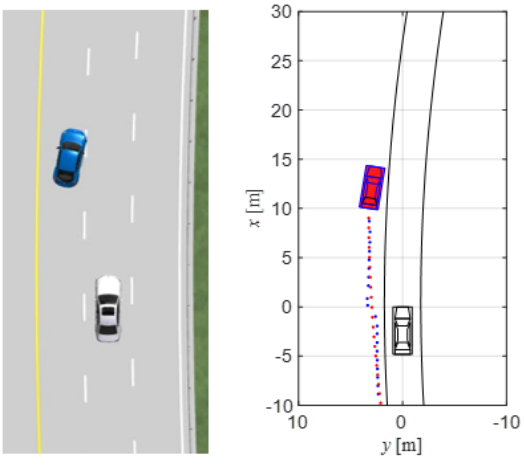



Fig. 8 Comparison of computing time based on the number of surrounding vehicles (SV)




(a) Snapshot and bird's eye view at 4.5s

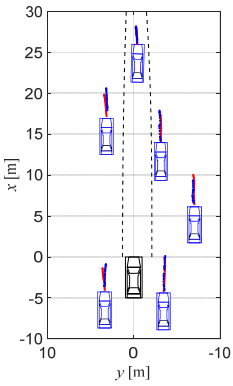



(b) Snapshot and bird's eye view at 5.1s

Fig. 9 The collision and trajectory prediction results for the simulation data

(a) Front camera image

(b) Rear camera image




(c) Bird's eye view

Fig. 10 The collision and trajectory prediction results for the experimental data

자차량이 주행하는 차로로 진입하기 시작하는 상황이다. 이 시점은 실제 충돌 발생 0.65s전에 해당하며 제안하는 네트워크가 Cut-in 차량에 대한 충돌을 예측하여 Bird's eye view 상에서 전방 차량이 빨간색으로 채워지는 것을 볼 수 있다. 그리고 예측된 궤적의 횡방향 위치가 실제 주행 경로의 횡방향 위치 대비 오차가 발생하지만 Lane following 상황과 유사하게 큰 차이가 발생하지 않는 것을 확인할 수 있다.

Fig. 10은 실도로 주행 데이터에 대한 충돌 및 궤적 예측 결과이다. 해당 데이터는 고속도로에서 다수의 주변 차량과 함께 주행하는 상황이다. 제안하는 네트워크가 주변 차량에 대해 충돌을 감지하지 않았으므로 Bird's eye view 상의 모든 차량이 하얀색으로 표기된다. 궤적 예측의 경우 예측 궤적(파란색 점)과 실제 주행 경로(빨간색 점)와의 차이가 크지 않은 것을 볼 수 있으며 1대의 차량을 포함하는 시뮬레이션 대비 다수의 차량이 존재하는 실도로 주행 환경에서도 충돌 예측과 궤적 예측을 포함하는 다중 작업이 잘 수행되는 것을 확인할 수 있다.

## 5. 결 론

본 연구에서는 트랜스포머 네트워크에 MTL을 적용하여 충돌 예측과 궤적 예측을 동시에 수행하는 방법을 제안하였다. 충돌을 예측하기 위해 사고가 포함된 시뮬레이션 데이터와 실도로 주행 데이터를 모두 이용해 알고리즘을 개발 및 평가하였다. 충돌 예측의 경우 기존 데이터 기반 알고리즘들 중 모델 기반 접근 방법과 데이터 기반 접근 방법을 통합한 알고리즘과 유사한 성능을 보임을 확인하였다. 그리고 주변 객체가 많은 주행환경에 대해 약 2배 적은 연산 시간이 소요되었으며 이를 통해 제안하는 알고리즘이 임베디드 관점에서 장점이 있을

수 있음을 확인하였다. 향후 계획으로는 시나리오를 다양화하고 Input feature에 위험도 인자를 추가할 예정이다. 추가적으로 차량 간 상호작용을 고려하여 충돌 예측 및 궤적 예측 성능을 고도화 하고자 한다.

## 후 기

본 연구는 국토교통부/국토교통과학기술진흥원의 지원으로 수행되었습니다(과제번호 22AMDP-C162184-02).

### References

1) K. Kim, D. Kim and K. Yi, “Development of Lateral Collision Risk Index,” KSAE Annual Conference Proceedings, pp.1376-1381, 2012.
2) J. Jansson, Collision Avoidance Theory with Application to Automotive Collision Mitigation, Ph. D. Dissertation, Linköping University, Sweden, 2005.
3) D. Lee and H. Yeo, “Real-Time Rear-End Collision-Warning System Using a Multilayer Perceptron Neural Network,” IEEE Transactions on Intelligent Transportation Systems, Vol.17, No.11, pp.3087-3097, 2016.
4) X. Wang, J. Liu, T. Qiu, C. Mu, C. Chen and P. Zhou, “A Real-Time Collision Prediction Mechanism with Deep Learning for Intelligent Transportation System,” IEEE Transactions on Vehicular Technology, Vol.69, No.9, pp.9497-9508, 2020.
5) S. Lee, B. Song and J. Shin, “Collision Prediction in an Integrated Framework of Scenario-Based and Data-Driven Approaches,” IEEE Access, Vol.12, pp.55234-55247, 2024.
6) S. Park, B. Kim, C. Kang, C. Chung and J. Choi, “Sequence-to-Sequence Prediction of Vehicle Trajectory via LSTM Encoder-Decoder Architecture,” Proceedings of the IEEE Intelligent Vehicles Symposium, pp.1672-1678, 2018.
7) K. Messaoud, I. Yahiaoui, A. Verroust-Blondet and F. Nashashibi, “Relational Recurrent Neural Networks for Vehicle Trajectory Prediction,” Proceedings of the IEEE Intelligent Transportation System Conference, pp.1813-1818, 2019.
8) A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser and I. Polosukhin, “Attention is All You Need,” Advances in Neural Information Processing Systems, pp.5998-6008, 2017.
9) J. Gu, J. Bradbury, C. Xiong, V. O. K. Li and R. Socher, “Non-Autoregressive Neural Machine Translation,” International Conference on Learning Representations, arXiv preprint arXiv:1711.02281, 2017.
10) P. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser and N. Shazeer, “Generating Wikipedia by Summarizing Long Sequence,” arXiv preprint arXiv:1801.10198, 2018.
11) A. Radford, K. Narasimhan, T. Salimans and I. Sutskever, “Improving Language Understanding by Generative Pre-Training,” 2018.
12) M. Crawshaw, “Multi-Task Learning with Deep Neural Networks: A Survey,” arXiv:2009.09796, 2020.
13) T. Standley, A. Zamir, D. Chen, L. Guibas, J. Malik and S. Savarese, “Which Tasks Should Be Learned Together in Multi-Task Learning,” International Conference on Machine Learning, pp.9120-9132, 2018.
14) K. Gao, X. Li, B. Chen, L. Hu, J. Liu, R. Du and Y. Li, “Dual Transformer Based Prediction for Lane Change Intentions and Trajectories in Mixed Traffic Environment,” IEEE Transactions on Intelligent Transportation Systems, Vol.24, No.6, pp.6203-6216, 2023.
15) D. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” arXiv preprint arXiv: 1412.6980, 2014.
16) S. Riedmaier, T. Ponn, D. Ludwig, B. Schick and F. Diermeyer, “Survey on Scenario-Based Safety Assessment of Automated Vehicles,” IEEE Access, Vol.8, pp.87456-87477, 2020.
17) W. Najm, J. Smith and M. Yanagisawa, “Pre-Crash Scenario Typology for Crash Avoidance Research,” National Highway Traffic Safety Administration, Washington, DC, USA, DOT HS 810 767, 2007.
18) E. Thron, S. Kimmel and M. Chaka, “A Framework for Automated Driving System Testable Cases and Scenarios,” National Highway Traffic Safety Administration, Washington, DC, USA, DOT HS 812 623, 2018.
19) J. Lee, U. Jung and B. Song, “Critical Scenario Generation for Collision Avoidance of Automated Vehicles Based on Traffic Accident Analysis and Machine Learning,” Transactions of KSAE, Vol.28, No.11, pp.817-826, 2020.
20) IPG Automotive, CarMaker Reference Manual Version 7.0.2, IPG Automotive GmbH, 2014.
21) R. Song and B. Li, “Surrounding Vehicles’ Lane Change Maneuver Prediction and Detection for Intelligent Vehicles: A Comprehensive Review,” IEEE Transactions on Intelligent Transportation Systems, Vol.23, No.7, pp.6046-6062, 2022.