NEAT: Neural Attention Fields for End-to-End Autonomous Driving

Kashyap Chitta\textsuperscript{1,2} Aditya Prakash\textsuperscript{1} Andreas Geiger\textsuperscript{1,2}
\textsuperscript{1}Max Planck Institute for Intelligent Systems, Tübingen \textsuperscript{2}University of Tübingen
\{firstname.lastname\}@tue.mpg.de

Abstract

Efficient reasoning about the semantic, spatial, and temporal structure of a scene is a crucial prerequisite for autonomous driving. We present NEural ATtention fields (NEAT), a novel representation that enables such reasoning for end-to-end imitation learning models. NEAT is a continuous function which maps locations in Bird’s Eye View (BEV) scene coordinates to waypoints and semantics, using intermediate attention maps to iteratively compress high-dimensional 2D image features into a compact representation. This allows our model to selectively attend to relevant regions in the input while ignoring information irrelevant to the driving task, effectively associating the images with the BEV representation. In a new evaluation setting involving adverse environmental conditions and challenging scenarios, NEAT outperforms several strong baselines and achieves driving scores on par with the privileged CARLA expert used to generate its training data. Furthermore, visualizing the attention maps for models with NEAT intermediate representations provides improved interpretability.

1. Introduction

Navigating large dynamic scenes for autonomous driving requires a meaningful representation of both the spatial and temporal aspects of the scene. Imitation Learning (IL) by behavior cloning has emerged as a promising approach for this task [5, 10, 16, 53, 78]. Given a dataset of expert trajectories, a behavior cloning agent is trained through supervised learning, where the goal is to predict the actions of the expert given some sensory input regarding the scene [48]. To account for the complex spatial and temporal scene structure encountered in autonomous driving, the training objectives used in IL-based driving agents have evolved by incorporating auxiliary tasks. Pioneering methods, such as CILRS [16], use a simple self-supervised auxiliary training objective of predicting the ego-vehicle velocity. Since then, more complex training signals aiming to reconstruct the scene have become common, e.g. image auto-encoding [47], 2D semantic segmentation [29], Bird’s Eye View (BEV) semantic segmentation [37], 2D semantic prediction [27], and BEV semantic prediction [58]. Performing an auxiliary task such as BEV semantic prediction, which requires the model to output the BEV semantic segmentation of the scene at both the observed and future time-steps, incorporates spatiotemporal structure into the intermediate representations learned by the agent. This has been shown to lead to more interpretable and robust models [58]. However, so far, this has only been possible with expensive LiDAR and HD map-based network inputs which can be easily projected into the BEV coordinate frame.

The key challenge impeding BEV semantic prediction from camera inputs is one of association: given a BEV spatiotemporal query location \((x, y, t)\) in the scene (e.g. 2 meters in front of the vehicle, 5 meters to the right, and 2 seconds into the future), it is difficult to identify which image pixels to associate to this location, as this requires reasoning about 3D geometry, scene motion, ego-motion, and intention, as well as interactions between scene elements. In this paper, we propose NEural ATtention fields (NEAT), a

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure1.png}
\caption{Neural Attention Fields. We use an MLP to iteratively compress the high-dimensional input into a compact low-dimensional representation \(c_i\) based on the BEV query location \((x, y, t)\). Our model outputs waypoint offsets and auxiliary semantics from \(c_i\) continuously and with a low memory footprint. Training for both tasks jointly leads to improved driving performance on CARLA.}
\end{figure}
flexible and efficient feature representation designed to address this challenge. Inspired by implicit shape representations [39, 50], NEAT represents large dynamic scenes with a fixed memory footprint using a multi-layer perceptron (MLP) query function. The core idea is to learn a function from any query location \((x, y, t)\) to an attention map for features obtained by encoding the input images. NEAT compresses the high-dimensional image features into a compact low-dimensional representation relevant to the query location \((x, y, t)\), and provides interpretable attention maps as part of this process, without attention supervision [79]. As shown in Fig. 1, the output of this learned MLP can be used for dense prediction in space and time. Our end-to-end approach predicts waypoint offsets to solve the main trajectory planning task (described in detail in Section 3), and uses BEV semantic prediction as an auxiliary task.

Using NEAT intermediate representations, we train several autonomous driving models for the CARLA driving simulator [20]. We consider a more challenging evaluation setting than existing work based on the new CARLA Leaderboard [1] with CARLA version 0.9.10, involving the presence of multiple evaluation towns, new environmental conditions, and challenging pre-crash traffic scenarios. We outperform several strong baselines and match the privileged expert’s performance on our internal evaluation routes. On the secret routes of the CARLA Leaderboard, NEAT obtains competitive driving scores while incurring significantly fewer infractions than existing methods.

Contributions: (1) We propose an architecture combining our novel NEAT feature representation with an implicit decoder [39] for joint trajectory planning and BEV semantic prediction in autonomous vehicles. (2) We design a challenging new evaluation setting in CARLA consisting of 6 towns and 42 environmental conditions and conduct a detailed empirical analysis to demonstrate the driving performance of NEAT. (3) We visualize attention maps and semantic scene interpolations from our interpretable model, yielding insights into the learned driving behavior. Our code is available at https://github.com/autonomousvision/neat.

2. Related Work

Implicit Scene Representations: The geometric deep learning community has pioneered the idea of using neural implicit representations of scene geometry. These methods represent surfaces as the boundary of a neural classifier [12, 13, 35, 39, 59] or zero-level set of a signed distance field regression function [36, 40, 50, 62, 63, 77]. They have been applied for representing object texture [44, 45, 64], dynamics [43] and lighting properties [41, 46, 61]. Recently, there has been progress in applying these representations to compose objects from primitives [11, 18, 19, 24], and to represent larger scenes, both static [8, 31, 51] and dynamic [21, 33, 34, 74]. These methods obtain high-resolution scene representations while remaining compact, due to the constant memory footprint of the neural function approximator. While NEAT is motivated by the same property, we use the compactness of neural approximators to learn better intermediate features for the downstream driving task.

End-to-End Autonomous Driving: Learning-based autonomous driving is an active research area [30, 65]. IL for driving has advanced significantly [5, 15, 42, 53, 72, 78] and is currently employed in several state-of-the-art approaches, some of which predict waypoints [7, 10, 23], whereas others directly predict vehicular control [4, 6, 16, 29, 47, 54, 75, 80]. While other learning-based driving methods such as affordances [60, 76] and Reinforcement Learning [9, 66, 70] could also benefit from a NEAT-based encoder, in this work, we apply NEAT to improve IL-based autonomous driving.

BEV Semantics for Driving: A top-down view of a street scene is powerful for learning the driving task since it contains information regarding the 3D scene layout, objects do not occlude each other, and it represents an orthographic projection of the physical 3D space which is better correlated with vehicle kinematics than the projective 2D image domain. LBC [10] exploits this representation in a teacher-student approach. A teacher that learns to drive given BEV semantic inputs is used to supervise a student aiming to perform the same task from images only. By doing so, LBC achieves state-of-the-art performance on the previous CARLA version 0.9.6, showcasing the benefits of the BEV representation. NEAT differs from LBC by directly learning in BEV space, unlike the LBC student model which learns a classical image-to-trajectory mapping.

Other works deal with BEV scenes, e.g., obtaining BEV projections [2, 81] or BEV semantic predictions [26, 28, 38, 49, 56] from images, but do not use these predictions for driving. More recently, LSS [52] and OGMs [37] demonstrated joint BEV semantic reconstruction and driving from camera inputs. Both methods involve explicit projection based on camera intrinsics, unlike our learned attention-based feature association. They only predict semantics for static scenes, while our model includes a time component, performing prediction up to a fixed horizon. Moreover, unlike us, they only evaluate using offline metrics which are known to not necessarily correlate well with actual downstream driving performance [14]. Another related work is P3 [58] which jointly performs BEV semantic prediction and driving. In comparison to P3 which uses expensive LiDAR and HD map inputs, we focus on image modalities.

3. Method

A common approach to learning the driving task from expert demonstrations is end-to-end trajectory planning, which uses waypoints \(w_t\) as outputs. A waypoint is defined
as the position of the vehicle in the expert demonstration at time-step \( t \), in a BEV projection of the vehicle’s local coordinate system. The coordinate axes are fixed such that the vehicle is located at \((x, y) = (0, 0)\) at the current time-step \( t = T \), and the front of the vehicle is aligned along the positive y-axis. Waypoints from a sequence of future time-steps \( t = T + 1, ..., T + Z \) form a trajectory that can be used to control the vehicle, where \( Z \) is a fixed prediction horizon.

As our agent drives through the scene, we collect sensor data into a fixed-length buffer of \( T \) time-steps, \( \mathcal{X} = \{x_{s,t}\}_{s=1; t=1:T}\) where each \( x_{s,t} \) comes from one of \( S \) sensors. The final frame in the buffer is always the current time-step \( (t = T) \). In practice, the \( S \) sensors are RGB cameras, the standard input modality in existing work on CARLA [10]. By default, we use \( S = 3 \) cameras, one oriented forward and the others 60 degrees to the left and right. After cropping these camera images to remove radial distortion, these \( S = 3 \) images together provide a full \( 180^\circ \) view of the scene in front of the vehicle. While NEAT can be applied with different buffer sizes, we focus in our experiments on the setting where the input is a single frame \((T = 1)\), as several studies indicate that using historical observations can be detrimental to the driving task [69,73].

In addition to waypoints, we use BEV semantic prediction as an auxiliary task to improve driving performance. Unlike waypoints which are small in number (e.g. \( Z = 4 \)) and can be predicted discretely, BEV semantic prediction is a dense prediction task, aiming to predict semantic labels at any spatiotemporal query location \((x, y, t)\) bounded to some spatial range and the time interval \( 1 \leq t \leq T + Z \). Predicting both observed \((1 \leq t < T)\) and future \((T < t \leq T + Z)\) semantics provides a holistic understanding of the scene dynamics. Dynamics prediction from a single input frame is possible since the orientation and position of vehicles encodes information regarding their motion [68].

The coordinate system used for BEV semantic prediction is the same as the one used for waypoints. Thus, if we frame waypoint prediction as a dense prediction task, it can be solved simultaneously with BEV semantic prediction using the proposed NEAT as a shared representation. Therefore, we propose a dense offset prediction task to locate waypoints as visualized in Fig. 2 using a standard optical flow color wheel [3]. The goal is to learn the field of 2-dimensional offset vectors \( o \) from query locations \((x, y, t)\) to the waypoint \( w_t \) (e.g. \( o = (0, 0) \) when \((x, y) = w_T\) and \( t = T \)). In certain situations, future waypoints along different trajectories are plausible (e.g. taking a left or right turn at an intersection), thus it is important to adapt \( o \) based on the driver intention. We do this by using provided target locations \((x', y')\) as inputs. Target locations are GPS coordinates provided by a navigational system along the route to be followed. They are transformed to the same coordinate system as the waypoints before being used as inputs. These target locations are sparse and can be hundreds of meters apart.

Figure 2: Dense offset prediction. We visualize the target location \((x', y')\) (blue dot), waypoint \( w_t \) (red dot) and waypoint offsets \( o \) (arrows) for a scene at three time instants. The offsets \( o \) represent the 2D vector from any query location \((x, y)\) to the waypoint \( w_t \) at time \( t \) and thus implicitly represent the waypoint. The arrows illustrate \( o \) for four different query locations \((x, y)\). We also show a color coding based visualization of the dense flow field learned by our model, representing \( o \) from any \((x, y)\) location in the scene.

### 3.1. Architecture

As illustrated in Fig. 3, our architecture consists of three neural networks that are jointly trained for the BEV semantic prediction and dense offset prediction tasks: an encoder \( e_\theta \), neural attention field \( a_\phi \), and decoder \( d_\psi \). In the following, we go over each of the three components in detail.

**Encoder:** Our encoder \( e_\theta \) takes as inputs the sensor data buffer \( \mathcal{X} \) and a scalar \( v \), which is the vehicle velocity at the current time-step \( T \). Formally, it is denoted as

\[
e_\theta : \mathbb{R}^{S \times T \times W \times H \times 3} \times \mathbb{R} \rightarrow \mathbb{R}^{(S \times T) \times P \times C}
\]  

(1)

where \( \theta \) denotes the encoder parameters. Each image \( x_{s,t} \in \mathbb{R}^{W \times H \times 3} \) is processed by a ResNet [25] to provide a grid of features from the penultimate layer of size \( \mathbb{R}^{P \times C} \), where \( P \) is the number of spatial features per image and \( C \) is the feature dimensionality. For the \( 256 \times 256 \) pixel input resolution we consider, we obtain \( P = 64 \) patches from the default ResNet architecture. These features are further processed by a transformer [67]. The goal of the transformer is to integrate features globally, adding contextual cues to each patch with its self-attention mechanism. This enables interactions between features across different images and over a large spatial range. Note that the transformer can be removed from our encoder without changing the output dimensionality, but we include it since it provides an improvement as per our ablation study. Before being input to the transformer, each patch feature is combined (through addition)
with (1) a velocity feature obtained by linearly projecting $v$ to $\mathbb{R}^C$, and broadcasting to all patches of all sensors at all time-steps, as well as (2) a learned positional embedding, which is a trainable parameter of size $(S \times T \times P) \times C$. The transformer outputs patch features $c \in \mathbb{R}^{(S \times T \times P) \times C}$.

**Neural Attention Field:** While the transformer aggregates features globally, it is not informed by the query and target location. Therefore, we introduce NEAT (Fig. 1), which identifies the patch features from the encoder relevant for making predictions regarding any query point in the scene $p = (x, y, t, x', y')$. It introduces a bottleneck in the network and improves interpretability (Fig. 6). Its operation can be formally described as

$$a_\phi : \mathbb{R}^5 \times \mathbb{R}^C \rightarrow \mathbb{R}^{S \times T \times P}$$

(2)

Note that the target location $(x', y')$ input to NEAT is omitted in Fig. 1 for clarity. While NEAT could in principle directly take as inputs $p$ and $c$, this would be inefficient due to the high dimensionality of $c \in \mathbb{R}^{(S \times T \times P) \times C}$. We instead use a simple iterative attention process with $N$ iterations. At iteration $i$, the output $a_i \in \mathbb{R}^{S \times T \times P}$ of NEAT is used to obtain a feature $c_i \in \mathbb{R}^C$ specific to the query point $p$ through a softmax-scaled dot product between $a_i$ and $c$:

$$c_i = \text{softmax}(a_i)^T \cdot c$$

(3)

The feature $c_i$ is used as the input of $a_\phi$ along with $p$ at the next attention iteration, implementing a recurrent attention loop (see Fig. 3). Note that the dimensionality of $c_i$ is significantly smaller than that of the transformer output $c$, as $c_i$ aggregates information (via Eq. (3)) across sensors $S$, time-steps $T$ and patches $P$. For the initial iteration, $c_0$ is set to the mean of $c$ (equivalent to assuming a uniform initial attention). We implement $a_\phi$ as a fully-connected MLP with 5 ResNet blocks of 128 hidden units each, conditioned on $c$, using conditional batch normalization [17, 22] (details in supplementary). We share the weights of $a_\phi$ across all iterations which works well in practice.

**Decoder:** The final network in our model is the decoder:

$$d_\phi : \mathbb{R}^5 \times \mathbb{R}^C \rightarrow \mathbb{R}^{K \times 2}$$

(4)

It is an MLP with a similar structure to $a_\phi$, but differing in terms of its output layers. Given $p$ and $c_i$, the decoder predicts the semantic class $s_i \in \mathbb{R}^K$ (where $K$ is the number of classes) and waypoint offset $o_i \in \mathbb{R}^2$ at each of the $N$ attention iterations. While the outputs decoded at intermediate iterations ($i < N$) are not used at test time, we supervise these predictions during training to ease optimization.

### 3.2. Training

**Sampling:** An important consideration is the choice of query samples $p$ during training, and how to acquire ground truth labels for these points. Among the 5 dimensions of $p$, $x'$ and $y'$ are fixed for any $X'$, but $x$, $y$, and $t$ can all be varied to access different positions in the scene. Note that in the CARLA simulator, the ground truth waypoint is only available at discrete time-steps, and the ground truth semantic class only at discrete $(x, y, t)$ locations. However, this is not an issue for NEAT as we can supervise our model using arbitrarily sparse observations in the space-time volume. We consider $K = 5$ semantic classes by default: none, road, obstacle (person or vehicle), red light, and green light. The location and state of the traffic light affecting the ego-vehicle are provided by CARLA. We use this to set the semantic label for points within a fixed radius of the traffic light pole to the red light or green light class, similar to [10]. In our work, we focus on the simulated setting where this information is readily available, though BEV semantic annotations...
of objects (obstacles, red lights, and green lights) can also be obtained for real scenes using projection. The only remaining labels required by NEAT (for the road class) can be obtained by fitting the ground plane to LiDAR sweeps in a real dataset or more directly from localized HD maps.

We acquire these BEV semantic annotations from CARLA up to a fixed prediction horizon $Z$ after the current time-step $T$ and register them to the coordinate frame of the ego-vehicle at $t = T$. $Z$ is a hyper-parameter that can be used to modulate the difficulty of the prediction task. From the aligned BEV semantic images, we only consider points approximately in the field-of-view of our camera sensors. We use a range of 50 meters in front of the vehicle and 25 meters to either side (detailed sensor configurations are provided in the supplementary).

Since semantic class labels are typically heavily imbalanced, simply using all observations for training (or sampling a random subset) would lead to several semantic classes being under-represented in the training distribution. We use a class-balancing sampling heuristic during training to counter this. To sample $M$ points for $K$ semantic classes, we first group all the points from all the $T + Z$ time-steps in the sequence into bins based on their semantic label. We then attempt to randomly draw $M/K$ points from each bin, starting with the class having the least number of available points. If we are unable to sample enough points from any class, this difference is instead sampled from the next bin, always prioritizing classes with fewer total points.

We obtain the offset ground truth for each of these $M$ sampled points by collecting the ground truth waypoints for the $T + Z$ time-steps around each frame. The offset label for each of the $M$ sampled points is calculated as its difference from the ground truth waypoint at the corresponding time-step $w_t$. Being a regression task, we find that offset prediction does not benefit as much from a specialized sampling strategy. Therefore, we use the same $M$ points for supervising the offsets even though they are sampled based on semantic class imbalance, improving training efficiency.

**Loss:** For each of the $M$ sampled points, the decoder makes predictions $s_i$ and $o_i$ at each of the $N$ attention iterations. The encoder, NEAT and decoder are trained jointly with a loss function applied to these $MN$ predictions:

$$
\mathcal{L} = \frac{1}{MN} \sum_{i}^{N} \sum_{j}^{M} \gamma_i \left| \left| o_{i,j}^* - o_{i,j} \right|_1 \right| + \lambda \mathcal{L}_{CE}(s_{i,j}^*, s_{i,j}) \tag{5}
$$

where $||.||_1$ is the $L_1$ distance between the true offset $o^*$ and predicted offset $o_i$, $\mathcal{L}_{CE}$ is the cross-entropy between the true semantic class $s^*$ and predicted class $s_{i,j}$, $\lambda$ is a weight between the semantic and offset terms, and $\gamma_i$ is used to down-weight predictions made at earlier iterations ($i < N$). These intermediate losses improve performance, as we show in our experiments.

### 3.3. Controller

To drive the vehicle at test time, we generate a red light indicator and waypoints from our trained model; and convert them into steer, throttle, and brake values. For the red light indicator, we uniformly sample a sparse grid of $U \times V$ points in $(x, y)$ at the current time-step $t = T$, in the area 50 meters to the front and 25 meters to the right side of the vehicle. We append the target location $(x', y')$ to these grid samples to obtain 5-dimensional queries that can be used as NEAT inputs. From the semantic prediction obtained for these points at the final attention iteration $s_N$, we set the red light indicator $r$ as 0 if none of the points belongs to the red light class, and 1 otherwise. In our ablation study, we find this indicator to be important for performance.

To generate waypoints, we sample a uniformly spaced grid of $G \times G$ points in a square region of side $A$ meters centered at the ego-vehicle at each of the future time-steps $t = T + 1, \ldots, T + Z$. Note that predicting waypoints with a single query point ($G = 1$) is possible, but we use a grid for robustness. After encoding the sensor data and performing $N$ attention iterations, we obtain $o_N$ for each of the $G^2$ query points at each of the $Z$ future time-steps. We offset the $(x, y)$ location coordinates of each query point towards the waypoint by adding $o_N$, effectively obtaining the waypoint prediction for that sample, i.e. $(p[0], p[1]) = o_N$. After this offset operation, we average all $G^2$ waypoint predictions at each future time instant, yielding the final waypoint predictions $w_t$. To obtain the throttle and brake values, we compute the vectors between waypoints of consecutive time-steps and input the magnitude of these vectors to a longitudinal PID controller along with the red light indicator. The relative orientation of the waypoints is input to a lateral PID controller for turns. Please refer to the supplementary material for further details on both controllers.

### 4. Experiments

In this section, we describe our experimental setting, showcase the driving performance of NEAT in comparison to several baselines, present an ablation study to highlight the importance of different components of our architecture, and show the interpretability of our approach through visualizations obtained from our trained model.

**Task:** We consider the task of navigation along pre-defined routes in CARLA version 0.9.10 [20]. A route is defined by a sequence of sparse GPS locations (target locations). The agent needs to complete the route while coping with background dynamic agents (pedestrians, cyclists, vehicles) and following traffic rules. We tackle a new challenge in CARLA 0.9.10: each of our routes may contain several predefined dangerous scenarios (e.g. unprotected turns, other vehicles running red lights, pedestrians emerging from occluded regions to cross the road).
Routes: For training data generation, we store data using an expert agent along routes from the 8 publicly available towns in CARLA, randomly spawning scenarios at several locations along each route. We evaluate NEAT on the official CARLA Leaderboard [1], which consists of 100 secret routes with unknown environmental conditions. We additionally conduct an internal evaluation consisting of 42 routes from 6 different CARLA towns (Town01-Town06). Each route has a unique environmental condition combining one of 7 weather conditions (Clear, Cloudy, Wet, MidRain, WetCloudy, HardRain, SoftRain) with one of 6 daylight conditions (Night, Twilight, Dawn, Morning, Noon, Sunset). Additional details regarding our training and evaluation routes are provided in the supplementary. Note that in this new evaluation setting, the multi-lane road layouts, distant traffic lights, high density of background agents, diverse daylight conditions, and new metrics which strongly penalize infractions (described below) make navigation more challenging, leading to reduced scores compared to previous CARLA benchmarks [16, 20].

Metrics: We report the official metrics of the CARLA Leaderboard, Route Completion (RC), Infraction Score (IS)\footnote{The Leaderboard refers to this as infraction penalty. We use the terminology ‘score’ since it is a multiplier for which higher values are better.} and Driving Score (DS). For a given route, RC is the percentage of the route distance completed by the agent before it deviates from the route or gets blocked. IS is a cumulative multiplicative penalty for every collision, lane infraction, red light violation, and stop sign violation. Please refer to the supplementary material for additional details regarding the penalty applied for each kind of infraction. Finally, DS is computed as the RC weighted by the IS for that route. After calculating all metrics per route, we report the mean performance over all 42 routes. We perform our internal evaluation three times for each model and report the mean performance over all 42 routes. We observe that in both evaluation settings, CILRS and LBC perform poorly. However, a major improvement can be obtained with a different conditioning signal. CILRS uses discrete navigational commands for conditioning, and LBC uses target locations represented in image space. By using target locations in BEV space and predicting waypoints, AIM and NEAT can more easily adapt their predictions to a change in driver intention, thereby achieving better scores. We show this adaptation of predictions for NEAT in Fig. 4, by predicting semantics and waypoint offsets for different target locations \( x' \) and time steps \( t \). The waypoint offset formulation of NEAT introduces a bias that leads to smooth trajectories between consecutive waypoints (red lines in \( o_N \)) towards the provided target location in blue.

AIM-MT and Expert: We observe that AIM-MT is a strong baseline that becomes progressively better with denser forms of auxiliary supervision. The final variant which incorporates dense supervision of both 2D depth and
Table 1: Quantitative Evaluation on CARLA. We report the RC, IS and DS over our 42 internal evaluation routes (Table 1a) and 100 secret routes on the evaluation server [1] (Table 1b). We abbreviate semantics with “Sem” and affordances with “Aff”.

(a) CARLA 42 Routes. We show the mean and standard deviation over 3 evaluations for each model. NEAT obtains the best driving score, on par with (and sometimes even outperforming) the expert used for data collection.

(b) CARLA Leaderboard. Among all publicly visible entries (accessed in July 2021), NEAT obtains the third-best DS. Of the top 3 methods, NEAT has the highest IS, indicating safer driving on unseen routes.

Figure 4: NEAT Visualization. We show $s_N$ (left) and $o_N$ (right) as we interpolate $x'$ and $t$ for the scene in Fig. 1. The green circles highlight the different predicted ego-vehicle positions. On the right, we show the predicted trajectory and waypoint $w_t$ in red. Note how the model adapts its prediction to the target location $(x', y')$ (shown in blue).

Figure 5: Ablation Study. We show the mean DS over our 42 evaluation routes for different NEAT model configurations. Expert performance is shown as a dotted line.

4.2. Ablation Study

In Fig. 5, we compare multiple variants of NEAT, varying the following parameters: training seed, semantic class count ($K$), attention iterations ($N$), prediction horizon ($Z$), input sensor count ($S$), transformer layers ($L$), and loss weights ($\gamma_1, \lambda$). While a detailed analysis regarding each factor is provided in the supplementary, we focus here on four variants in particular: Firstly, we observe that different random training seeds of NEAT achieve similar performance, which is a desirable property not seen in all end-to-end driving models [4]. Second, as observed by [4], 2D semantic models (such as AIM-VA and AIM-MT) rely heavily on lane marking annotations for strong performance. We observe that these are not needed by NEAT for which the default configuration with 5 classes outperforms the variant that includes lane markings with 6 classes. Third, in the shorter horizon variant ($Z = 2$) with only 2 predicted waypoints, we observe that the output waypoints do not deviate sharply enough from the vertical axis for the PID controller to perform certain maneuvers. It is also likely that the additional supervision provided by having a horizon of $Z = 4$ in our default configuration has a positive effect on performance. Fourth, the gain of the default NEAT model
Figure 6: Attention Maps. We visualize the semantics $s_N$ for 4 frames of a driving sequence (legend: none, road, obstacle, red light, green light). We highlight one particular $(x, y)$ location as a white circle on each $s_N$, for which we visualize the input and corresponding attention map $a_N$. NEAT consistently attends to the region corresponding to the object of interest (from top left to bottom right: bicyclist, green light, vehicle and red light). Best viewed on screen, zoom in for details.

compared to its version without the semantic loss ($\lambda = 0$) is 30%, showing the benefit of performing BEV semantic prediction and trajectory planning jointly.

**Runtime:** To analyze the runtime overhead of NEAT’s offset prediction task, we now create a hybrid version of AIM and NEAT. This model directly regresses waypoints from NEAT’s encoder features $c_0$ using a GRU decoder (like AIM) instead of predicting offsets. We still use a semantic decoder at train time supervised with only the cross-entropy term of Eq. (5). At test time, the average runtime per frame of the hybrid model (with the semantics head discarded) is 15.92 ms on a 3080 GPU. In comparison, the default NEAT model takes 30.37 ms, i.e., both approaches are real-time even with un-optimized code. Without the compute-intensive red light indicator, NEAT’s runtime is only 18.60 ms. Importantly, NEAT (DS = 65.10) significantly outperforms the AIM-NEAT hybrid model (DS = 33.63). This shows that NEAT’s attention maps and location-specific features lead to improved waypoint predictions.

**4.3. Visualizations**

Our supplementary video\(^2\) contains qualitative examples of NEAT’s driving capabilities. For the first route in the video, we visualize attention maps $a_N$ for different locations on the route in Fig. 6. For each frame in the video, we randomly sample BEV $(x, y)$ locations and pass them through the trained NEAT model until one of the locations corresponds to the class obstacle, red light, or green light. Four such frames are shown in Fig. 6. We observe a common trend in the attention maps: NEAT focuses on the image corresponding to the object of interest, albeit sometimes at a slightly different location in the image. This can be attributed to the fact that NEAT’s attention maps are overlearned image features that capture information aggregated over larger receptive fields. To quantitatively evaluate this property, we extract the $32 \times 32$ image patch which NEAT assigns the highest attention weight for one random $(x, y)$ location in each scene of our validation set and analyze its ground truth 2D semantic segmentation labels. The semantic class predicted by NEAT for $(x, y)$ is present in the 2D patch in 79.67% of the scenes.

**5. Conclusion**

In this work, we take a step towards interpretable, high-performance, end-to-end autonomous driving with our novel NEAT feature representation. Our approach tackles the challenging problem of joint BEV semantic prediction and vehicle trajectory planning from camera inputs and drives with the highest safety among state-of-the-art methods on the CARLA simulator. NEAT is generic and flexible in terms of both input modalities and output task/supervision and we plan to combine it with orthogonal ideas (e.g., DAgger, RL) in the future.

**Acknowledgements:** This work is supported by the BMBF through the Tübingen AI Center (FKZ: 01IS18039B) and the BMWi in the project KI Delta Learning (project number: 19A19013O). We thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Kashyap Chitta. The authors also thank Micha Schilling for his help in re-implementing AIM-VA.

---

\(^2\)https://www.youtube.com/watch?v=gtO-gjhKkRs
References

[5] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. arXiv.org, 1604.07316, 2016. 1, 2
[22] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. In Proc. of the International Conf. on Learning Representations (ICLR), 2017. 4
[28] Anthony Hu, Zak Murez, Nikhil Mohan, Sofía Dudas, Jeff Hawke, Vijay Badrinarayanan, Roberto Cipolla, and

[29] Zhiyu Huang, Chen Lv, Yang Xing, and Jingda Wu. Multimodal sensor fusion-based deep neural network for end-to-end autonomous driving with scene understanding. IEEE Sensors Journal, 2020. 1, 2


[55] Aditya Prakash, Kashyap Chitta, and Andreas Geiger. Multimodal fusion transformer for end-to-end autonomous driv-
ing. In Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2021. 6, 7


[57] Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Conference on Artificial Intelligence and Statistics (AISTATS), 2011. 7


[70] Jingke Wang, Yue Wang, Dongkun Zhang, Yezhou Yang, and Rong Xiong. Learning hierarchical behavior and motion planning for autonomous driving. arXiv.org, 2020. 2


