Multi-Abstractive Neural Controller: An Efficient Hierarchical Control Architecture for Interactive Driving

Xiao Li[1], Igor Gilitschenski[4], Guy Rosman[3], Sertac Karaman[2] and Daniela Rus[1]

***Abstract*—As learning-based methods make their way from perception systems to planning/control stacks, robot control systems have started to enjoy the benefits that data-driven methods provide. Because control systems directly affect the motion of the robot, data-driven methods, especially black box approaches, need to be used with caution considering aspects such as stability and interpretability. In this paper, we describe a differentiable and hierarchical control architecture. The proposed representation, called *multi-abstractive neural controller*, uses the input image to control the transitions within a novel discrete behavior planner (referred to as the visual automaton generative network, or *vAGN*). The output of a vAGN controls the parameters of a set of dynamic movement primitives which provides the system controls. We train this neural controller with real-world driving data via behavior cloning and show improved explainability, sample efficiency, and similarity to human driving.**

I. INTRODUCTION

With robotic and autonomous driving applications expanding from structured environments (factory floors, warehouses, etc) to open environments (road, homes, etc), traditional optimal planning/control methods will be insufficient in handling the large variety of edge-cases as manual specifications for them is intractable. Data-driven methods such as imitation learning have shown promising results in learning generalizable and capable robot control policies from human demonstrations. However, enabling blackbox policies such as neural networks to consistently produce stable behaviors outside of the training data distribution remains a challenge, hindering their adoption in safety-critical applications such as autonomous driving. In this work, our aim is to address the following question: "*can we design a robot control policy representation that (1) is explainable in its decision making process, (2) is resilient to unstable behaviors , and (3) is trainable end-to-end using expert demonstrations?*"

Combining model-based planning and control with learning components allows the robot system to have the stability and safety properties of model-based controllers while complexity and uncertainty of the environment (elements that are challenging to manually and exhaustively integrate into model/rule-based components) can be delegated to the data-driven components. Most efforts in this space integrate learning into a certain component of a motion planner including the cost function [1], dynamics [2], the solver [3], [4] and constraints [5]. Less work has been done to integrate learning one level up the planning stack - into the behavior planner. The concept of a multi-abstractive neural controller provides an initial effort to fill this gap and expand the learnability of a robotic planning/control stack into the high-level discrete planning domains.

In this paper, we start by referring to the classical planning/control stack where a discrete behavior planner (commonly in the form of a finite state machine) feeds high-level decisions (along with additional environment feedback) into a motion planner, which in turn outputs controls for the dynamic system. **Our approach is to design a differentiable architecture of this stack, and define their interfaces such that its structure can be learned from data**. We refer to this representation as the *multi-abstractive neural controller*. Figure 1 illustrates the desired architecture.

For the behavior planner, we introduce the visual automaton generative network (vAGN) - a differentiable automaton that takes in visual features and learns its transition structure from demonstration data. We adopt dynamic movement primitives (DMP) [6] as the motion controller (for its stability properties and simplicity) and introduce a novel method that interfaces vAGN with DMP that significantly improves the stability of the controlled system. To summarize our contributions, we

- introduce vAGN - an automaton planner with learnable latent structure;
- introduce the multi-abstractive neural controller - a hierarchical robot control representation that interfaces vAGN with DMP through DMP parameter control;
- demonstrate in a real world driving dataset that the proposed neural robot controller achieves high sample efficiency while balancing safety, optimality and comfort.

In this work, we focus on applying the neural controller on autonomous driving applications.

II. RELATED WORK

Joint learning and planning. Constructing learnable planners have been looked at in the past. The authors of [7] use a ResNet50 to extract features from camera images which are used to predict the acceleration and metric components of Riemannian motion policy (RMP). Similarly, [8] use learning components to generate parameters the model predictive controller. In [1], the authors use a neural network to generate cost maps from LiDAR and map inputs which are used to rank trajectory samples. In [2], the authors use neural networks to approximate the dynamics used in an model predictive controller (MPC). On a different line of idea, the authors of [3] propose to learn the update rules for the model predictive path integral (MPPI) planner. In [9], the authors use reinforcement learning to learn short horizon policies and probabilistic

[1] Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology {xiaoli, rus}@mit.edu
[2] Laboratory for Information and Decision Systems, Massachusetts Institute of Technology sertac@mit.edu
[3] Toyota Research Institute {guy.rosman}@tri.global
[4] University of Toronto {gilitschenski}@cs.toronto.edu

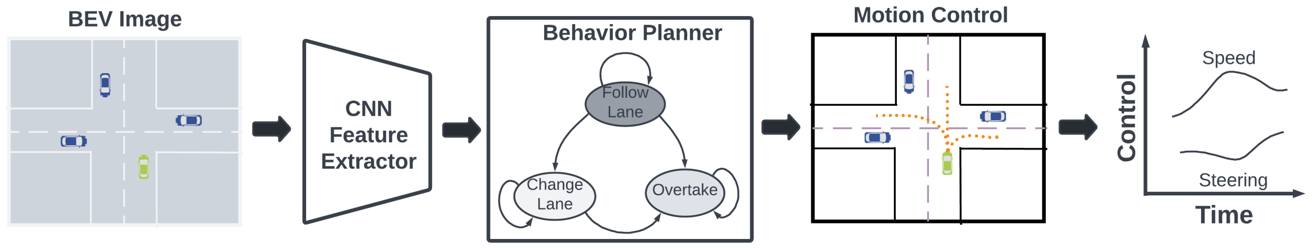

Fig. 1 : A differentiable planning and control representation. By designing the discrete behavior planner and motion controller as differentiable components and connecting them such that their structure can be learned from demonstrations, both generalizability from data-driven methods and stability from model-based methods can be achieved with a more efficient control representation.

road map for global navigation. The authors of [10] learn hierarchical task planners using imitation learning, but their planner structures are either untrainable or hard to interpret. Most of the work in the this space aim to learn certain component of a motion planner (dynamics, cost, optimizer, etc), but little work is done to develop learnable (discrete) behavior planners. Our work fills this gap by introducing a differentiable automaton with learnable transition structure (often a challenge to manually design). Then using CNN as the input processing unit, it is common practice to visualize its feature maps for explainability purposes. The class activation map [11], [12] is one such method which we have adopted in this work. In addition, the visual backpropagation method [12], [13] is also commonly used in self-driving applications.

Neural state machines. While little work has been done in learnable behavior planners, differentiable state machines have been developed in the field of visual question answering (VQA) and natural language processing (NLP). In [14], [15], the authors propose the neural state machine - a scene graph constructed from images that is able to make inferences based on natural language instructions. In [16], the authors introduce the differentiable weighted finite-state transducers to express and design sequence-level loss functions (used in handwriting and speech recognition). The authors of [17] proposes an approach to learn causal Bayesian networks from data. And finally, the authors of [18] learn a neural state machine used for character-scene interactions. Because these works are not tailored to robotic planning applications, they lack one or more of the following features that prevents them from being readily available as behavior planners, (a) the inability to learn state transition structures (b) the requirement of having ground truth graphs (or data-to-graph distribution) as supervision, (c) not using an underlying graph structure ([18] uses a 3 layer fully connected gating network). Our work addresses these problem and is demonstrated to be effective in the self-driving domain. The neural hybrid automaton (NHA) proposed by [19] learns a hybrid control system and is perhaps the closest to our work. However, NHA requires state estimates (e.g. position, speed, etc) as input whereas our method takes images as inputs. Explainability for general machine learning methods is discussed in [20], [21]. In this work, we focus on explainable learning systems for planning and control.

III. BACKGROUND

A. Linear Dynamic Movement Primitives (DMP)

A DMP [6] consists of a second order point attractor system added with a forcing function as below

$$\tau\ddot{\boldsymbol{y}} = \alpha_y\left(\beta_y(\boldsymbol{g}-\boldsymbol{y})-\dot{\boldsymbol{y}}\right)+\boldsymbol{f}(s,x|\theta_y), \quad \tau\dot{x}=\alpha_x x \tag{1}$$

where g is the goal state; α_y and β_y define the behavior of the second order system;τ is a time constant; x is a phase variable controlling the influence of the forcing function on the point attractor system. Appropriately setting τ, α_y, β_y, the convergence of the underlying dynamic system to $y = g$ is ensured [22] and the system is stable with respective to the goal. The first part of Equation (1) is often referred to as the transformation system. The transformation system serves to stably guide the robot to the goal with a trajectory jointly controlled by the point attractor and the forcing function. The second part (first order system of x) is the canonical system which controls the decay of x and hence reduces the effect of the forcing function as the robot gets close to the goal. $f(s,x|\theta_y)$ (s can be additional state information) is a learnable forcing function often in the form of a linear combination of N nonlinear Radial Basis Functions (RBFs). This allows the robot to reach the goal state by following a desired path influenced by $f(\cdot)$.

B. Quaternion Dynamic Movement Primitives

The DMP introduced in the previous section generates only linear movement. As orientation is just as important in defining robot motion, the authors of [23] introduced the equivalent of Equations (1) and (2) for unit quaternions $\mathrm{q} = [v, \mathbf{u}] \in \mathcal{S}^3$ ($\mathcal{S}^3$ is a unit sphere in $\mathbb{R}^4, v \in \mathbb{R}$, and $\mathbf{u} \in \mathbb{R}^3$). as follows

$$\begin{aligned}\tau\dot{\boldsymbol{\eta}} &= \alpha_q\left(\beta_q 2\log^q\left(\boldsymbol{g}_q * \bar{\mathrm{q}}\right)-\boldsymbol{\eta}\right)+\mathbf{f}_q(s,x|\theta_q)\\ \tau\dot{\mathrm{q}} &= \frac{1}{2}\boldsymbol{\eta} * \mathrm{q}\end{aligned} \tag{2}$$

where $\mathbf{g}_q \in \mathcal{S}^3$ denotes the goal orientation, the quaternion conjugation is defined as $\bar{\mathrm{q}} = \overline{v+\mathbf{u}} = v - \mathbf{u}$, and $*$ denotes the the quaternion product

$$\begin{aligned}\mathrm{q}_1 * \mathrm{q}_2 &= (v_1+\mathbf{u}_1)*(v_2+\mathbf{u}_2)\\ &= \left(v_1v_2 - \mathbf{u}^\top{}_1\mathbf{u}_2\right)+\left(v_1\mathbf{u}_2+v_2\mathbf{u}_1+\mathbf{u}_1\times\mathbf{u}_2\right)\end{aligned} \tag{3}$$

$\boldsymbol{\eta} \in \mathbb{R}^3$ is the scaled angular velocity $\boldsymbol{\omega}$ and treated as unit quaternion with zero scalar ($v = 0$). The function $\log^q(\cdot) : \mathcal{S}^3 \mapsto \mathbb{R}^3$ is given as

$$\log^q(\mathsf{q}) = \begin{cases} \arccos(v) \frac{\mathbf{u}}{\|\mathbf{u}\|}, & \mathbf{u} \neq \mathbf{0} \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top, & \text{otherwise} \end{cases} \tag{4}$$

where $\|\cdot\|$ denotes ℓ_2 norm. Equation (4) can be integrated as

$$\mathsf{q}(t + \delta t) = \operatorname{Exp}^q \left(\frac{\delta t}{2} \frac{\boldsymbol{\eta}(t)}{\tau} \right) * \mathsf{q}(t), \tag{5}$$

where $\delta_t > 0$ denotes a small constant. The function $\operatorname{Exp}^q(\cdot) : \mathbb{R}^3 \mapsto \mathcal{S}^3$ is given

$$\operatorname{Exp}^q(\boldsymbol{\omega}) = \begin{cases} \cos(\|\boldsymbol{\omega}\|) + \sin(\|\boldsymbol{\omega}\|) \frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|}, & \boldsymbol{\omega} \neq \mathbf{0} \\ 1 + \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top, & \text{otherwise.} \end{cases} \tag{6}$$

Both mappings become one-to-one, continuously differentiable and inverse to each other if the input domain of the mapping $\log^q(\cdot)$ is restricted to $\mathcal{S}^3$ except for $-1 + \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top$, while the input domain of the mapping $\operatorname{Exp}^q(\boldsymbol{\omega})$ should fulfill the constraint $\|\boldsymbol{\omega}\| < \pi$.

IV. Multi-Abstractive Neural Controller

In this section, we introduce the general architecture of the multi-abstractive neural controller with its three main component, (1) a CNN-based visual predicate extractor that learns and outputs visual features meaningful for high-level behavior planning; (2) a differentiable graphical planner with learnable structure and (3) a DMP with parameters controlled by the output of the behavior planner. The overall architecture is illustrated in Figure 2 . we refer to our controlled vehicle as the *ego* vehicle, and vehicles in traffic as the *ado* vehicles.

A. Visual Predicate Extractor

The purpose of the visual predicate extractor is to learn a feature vector where each of its element corresponds to the existence of a semantic feature in the input image (e.g. lane, stop sign, etc). Let $\mathcal{X}$ be the input image, here we use a bird's eye view image (BEV) with traffic components semantically colored. We perform the following operations to obtain the visual predicate feature vector

$$\begin{aligned} \boldsymbol{f}^v &= \texttt{ConvEncoder}(\mathcal{X}) \\ \boldsymbol{p}^v &= \texttt{Linear}\big(\texttt{GlobalAveragePooling}(\boldsymbol{f}^v)\big). \end{aligned} \tag{7}$$

In the equation above, $\texttt{ConvEncoder}(\cdot)$ is a set of convolution layers applied to the image. Its output is a set of feature maps to which we apply global average pooling [24] to. This serves to identify whether certain components exists in the image (i.e. pedestrians, intersections, etc). Lastly we apply a linear transformation to the output of the pooling layer. This allows each element in $\boldsymbol{p}^v$ to contain a weighted sum of all the identified features, which provides richer information to the downstream planner. The architecture of this component is inspired by the class activation map [25]. As shown in Figures 2 and 3, the visual predicates constitute the explainability components of vAGN that serves to illustrate the neural controller's internal decision making process.

B. Visual Automaton Generative Network (vAGN)

Given the visual predicates $\boldsymbol{p}^v \in \mathrm{I\!R}^M$ and the number of automaton nodes N (as a hyperparameter), the current automaton state $\boldsymbol{q}_t \in \mathrm{I\!R}^N$ is represented as an N-vector with each entry corresponding to the probability of being in q_i ($\boldsymbol{q}$ can be seen as the hidden state equivalent of an LSTM). As an example, for an automaton with 3 states $[q_1, q_2, q_3]$, $\boldsymbol{q} = [0.4, 0.3, 0.3]$ is the state distribution of this probabilistic graph. The learnable parameters of vAGN is its set of weighted transition matrices $\mathcal{W} \in \mathrm{I\!R}^{M \times N \times N}$. The vAGN update law is as follows

$$\mathcal{W}^{p^v} = \sum_{i \in \{0, M-1\}} (\boldsymbol{p}_i^v \times W_i) \tag{8}$$

$$\boldsymbol{q}_t = \underset{\text{along column}}{\operatorname{softmax}} \left(\operatorname{ReLU}(\mathcal{W}^{p^v})\right) \cdot \boldsymbol{q}_{t-1} \tag{9}$$

The dimension of $\mathcal{W}$ is such that each element of $p_i^v \in \boldsymbol{p}^v$ has a corresponding $W_i \in \mathcal{W} \in \mathrm{I\!R}^{N \times N}$. In Equation (8), $\mathcal{W}^{p^v} \in \mathrm{I\!R}^{N \times N}$ indicates the combined influence of the visual predicates on the transition of $\boldsymbol{q}$ probabilities. In Equation (9), we first apply a $\operatorname{ReLU}(\cdot)$ to $\mathcal{W}^{p^v}$ to preserve only the positive transitions. Softmax is applied to ensure that the columns of $\mathcal{W}^{p^v}$ sum to one (the sum of probabilities of transitioning out of any q state to another state is 1). Finally, the resultant transition matrix is applied to $\boldsymbol{q}_{t-1}$ via dot product.

C. Interfacing vAGN with DMP

The most common way of learning with DMPs is to learn the parameters of the forcing functions $\boldsymbol{f}(\cdot)$ and $\boldsymbol{f}_q(\cdot)$ in Equation (1). This works well with finite horizon tasks where the effects of the forcing functions diminish over time (with the canonical system) and the system eventually reaches the goal state. However, for potentially long/infinite horizon tasks (such as driving), it is less obvious how the learned forcing functions should scale. If they are always kept in effect, then the system may never reach the goal state. As an alternative, instead of adding the forcing function with point attractor system, we propose to use the forcing function to learn parameters of the point attractor system. As a result, Equations (1) becomes

$$\begin{aligned} \ddot{\boldsymbol{y}} &= \alpha_y(s|\theta_y) \left(\beta_y(s|\theta_y)(\boldsymbol{g} - \boldsymbol{y}) - \dot{\boldsymbol{y}}\right) \\ \dot{\boldsymbol{\eta}} &= \alpha_q(s|\theta_q) \left(\beta_q(s|\theta_q) 2 \log^q \left(\boldsymbol{g}_q * \bar{\mathsf{q}}\right) - \boldsymbol{\eta}\right), \end{aligned} \tag{10}$$

where $\{\alpha_y(s|\theta_y), \beta_y(s|\theta_y), \alpha_y(s|\theta_q), \beta_y(s|\theta_q)\}$ are the learned functions (the time constant τ can be incorporated in these 4 parameters). This can then be easily connected with the output of vAGN by

$$\begin{bmatrix} \ddot{\boldsymbol{y}} \\ \dot{\boldsymbol{\eta}} \end{bmatrix} = \texttt{DMP}(\texttt{FC}(\boldsymbol{q}), \boldsymbol{g}, \boldsymbol{g}_q) \tag{11}$$

where $\texttt{FC}(\boldsymbol{q}) = \{\alpha_y, \beta_y, \alpha_y, \beta_y\}$ is a set of fully connected layers transforming the vAGN states to the DMP parameters. The parameters θ_y, θ_q becomes the upstream network parameters (those of vAGN and CNN). Note that Equation (11) outputs accelerations. We use linear and angular velocities as controls, therefore we numerically integrate Equation (11) once to obtain the final output. Having the output of vAGN control the parameters of the DMP point attractor system run

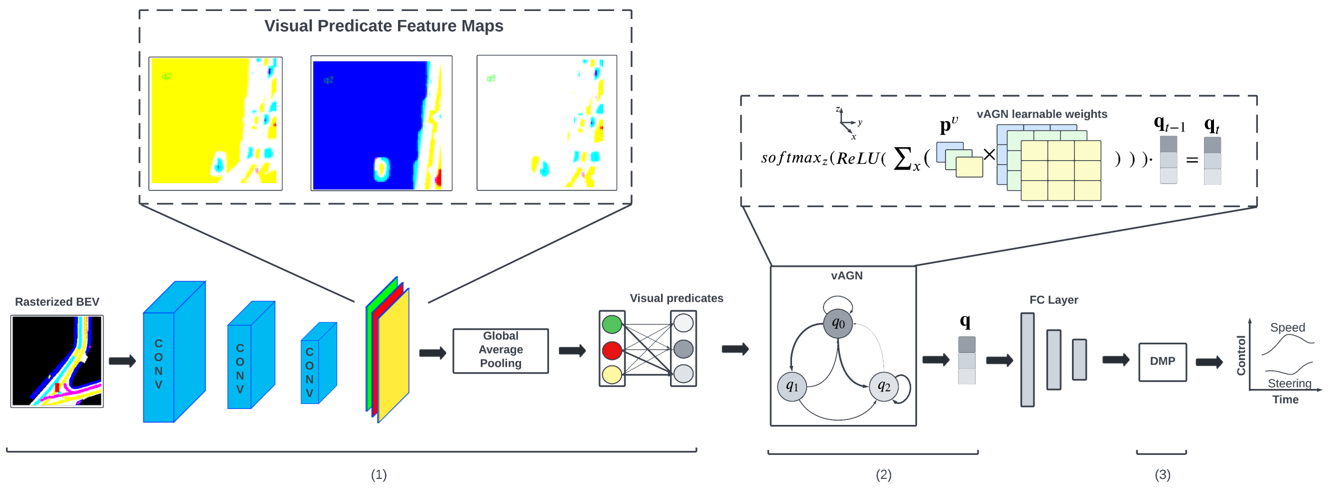

Fig. 2 : Multi-abstractive neural controller. The architecture contains three components, (1) a CNN-based visual predicate extractor that learns and outputs visual features meaningful for high-level behavior planning; (2) a differentiable graph-based planner (vAGN) with learnable structure and (3) a DMP with parameters controlled by the output of vAGN.

the risk of rendering the tracking behavior unstable. The point attractor system of the DMP can be rewritten as

$$\ddot{y} + \alpha\dot{y} + \alpha\beta y = 0 \tag{12}$$

where the constant term is neglected (does not effect stability analysis). Equation (12) is a typical second order system where stability is determined by

$$\zeta = \frac{\alpha}{2\sqrt{\alpha\beta}}. \tag{13}$$

The system can become unstable when $\zeta < 0$ and periodic when $\zeta \approx 0$. In our experiments, we have not run into this particular problem because we used the sigmoid activation when outputing α and β, ζ is always greater than zero. Fortunately, during training and deployment we have not run into the case where $\zeta \approx 0$. For the driving task that we are targeting, the ego vehicle is tracking a moving target along the center lane, and the parameters α, β are time varying functions of the upstream network, therefore, the system under our neural controller is resilient to unstable α, β values as long as they don't stay unstable for an extended period of time. To avoid $\zeta \approx 0$, one can set $\beta = \alpha/4$ such that the system is always critically damped (at the cost of somewhat limiting vehicle behaviors). One could also use an auxiliary loss at training time to prevent the upstream vAGN+FC from outputing α and β that are too close together.

vAGN will thus serve to control the tracking "aggressiveness". We show later in the experiments that various driving maneuvers can emerge from this combination and will also discuss its limitations. Note also that [26] introduces an alternative formulation of the forcing function that provides better stability. This does not affect our neural controller as we directly alter the point attractor's parameters (α, β) and neglect the forcing function. This is to avoid instabilities as a result of the forcing terms overturning the point attractor terms. Such overturning can lead to crashes in driving tasks (may cause less of a safety concern in manipulation tasks).

Algorithm 1 describes the procedures of learning a multi-abstractive neural controller from demonstrations.

Algorithm 1 Learning with Multi-Abstractive Neural Controller

```
1:  Inputs: number of vAGN nodes $N$; dataset $\boldsymbol{X}$; number
    of iterations $I$; learning rate $\gamma$
2:  $\theta^{vAGN}, \theta^{CNN} \leftarrow$ Initialize($N$)
3:  for i=1 ...I do
4:      Sample a minibatch of $m$ data samples $(\mathcal{X}, \boldsymbol{g}, \boldsymbol{g}_q, \boldsymbol{a})$
5:      $\boldsymbol{p}^v$ = VisualPredicateExtractor($\mathcal{X}$)    ▷ Equation (7)
6:      $\boldsymbol{q}'$ = UniformInit() or RandomInit()
7:      $\boldsymbol{q}$ = vAGN($\boldsymbol{p}^v, \boldsymbol{q}'$)    ▷ Equation (9)
8:      $\hat{\boldsymbol{a}}$ = DMP(FC($\boldsymbol{q}$), $\boldsymbol{g}, \boldsymbol{g}_q$)    ▷ Equation (11)
9:      $L$ = MSE($\hat{\boldsymbol{a}}, \boldsymbol{a}$)
10:     $(\theta^{vAGN}, \theta^{CNN}) \leftarrow (\theta^{vAGN}, \theta^{CNN}) - \gamma\frac{1}{m}\nabla L$
11: end for
```

In Algorithm 1, the loop for $i = 1...I$ refers to iteration over minibatches of data (not over time in a rollout). During training, we do not know the groundtruth $\boldsymbol{q}$, therefore, we assume a uniform or random $\boldsymbol{q}$ to pass into the vAGN() module. This is a preliminary resolution and yields reasonable results, but can definitely be improved (for example by consuming groundtruth $\boldsymbol{q}$ from a high-level behavior filter)

V. Experiments and Results

A. Setup

Nuscenes dataset. [27] is a dataset for autonomous driving based in Boston and Singapore. It contains 850 scenes each 20s long (sampled at 2hz, therefore a max of 40 steps), containing 23 object classes and HD semantic maps with 11 annotated layers. We chose this particular dataset for the rich semantics it provides which is well suited for rule definitions. We will use 650 scenes for training and 200 scenes for

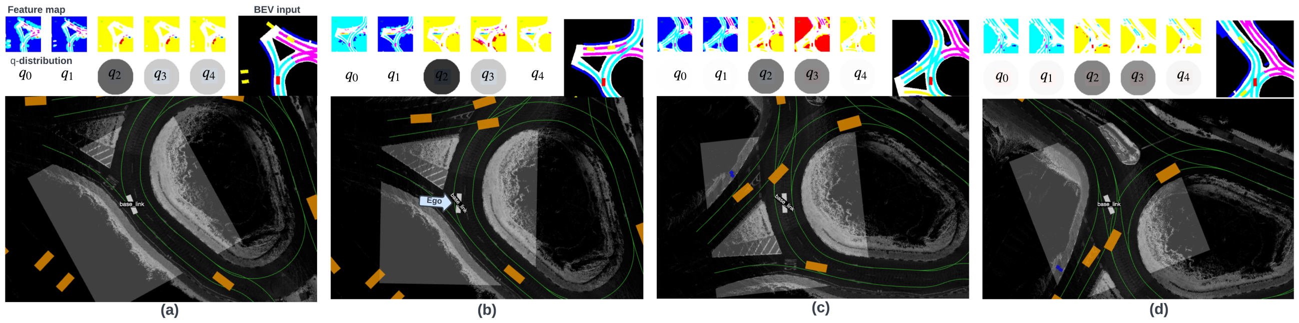

Fig. 3 : An example execution trace. Within each sub-figure we also show the current vAGN state distribution (darker the color means higher probability) and the saliency map indicating where each q-state is attending to. Color indicates the level of attention and ranks from high to low as: yellow → red → cyan → magenta. To the right of each q-distribution plot we also show the semantically colored BEV image sent as input to the network.**(a)** Ego vehicle drives in an open area with no nearby ado vehicles, vAGN attends mainly to the road boundaries. **(b)** vAGN starts to notice more of ado vehicles. **(c)** The ado vehicles cut in front of the ego vehicle and vAGN shifts part of its focus to these vehicles. **(d)** The ego vehicle proceeds out of the roundabout and vAGN attends back to mainly road boundaries.

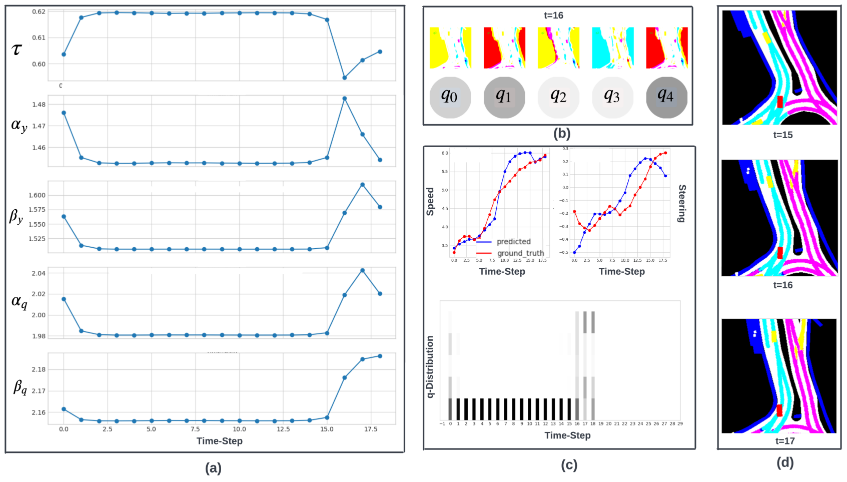

Fig. 4 : Explainability traces. (a) The evolution of DMP parameters over time. **(b)** The vAGN state distribution and corresponding saliency maps. **(c)** Control output from our neural controller (blue) and human controls (red) with synchronized q-distribution. **(d)** BEV image input at 3 timesteps. After step 15, the ego vehicle drives too close to the road boundary. As a result, vAGN shifts attention and the q-distribution to reduce the DMP parameters, which in turn reduces the speed and steering to prevent collision.

validation. During close-loop evaluation, all agents are rolled out synchronously and the ego agent's motion is controlled by our neural controller.

Methods of evaluation. We evaluate our method and comparison cases with the following metrics. *Percentage of close encounters* measures the average of times the ego vehicles comes to the vicinity of ado vehicles in a scene (safety measure). *Acceleration* is the maximum magnitude acceleration during a scene (comfort measure). *Similarity to human driving* is the L2-norm between the planner and human trajectories (driving style measure). *Goal distance* is the ego's final distance to the goal. All metrics are the lower the better. During evaluation, we control the ego vehicle with our learned planner, the ado vehicles move according to the trajectories recorded in the dataset with synchronized time. All results are averaged over the validation set.

It is worth mentioning that, the four evaluation metrics that we use are more trade-offs than objectives that can be optimized concurrently. For example, a controller that achieves a low goal distance over the evaluation scenes (within the fixed time horizon) is bound to drive fast and hence obtain a higher percentage close encounter and acceleration. Similarly, a controller that achieves high human driving similarity may obtain moderate scores on the other metrics. In our case, our objective function is the MSE loss with respective to the human ego vehicle's controls, therefore we put high emphasis on human driving similarity. In general planning and control scenarios, it ultimately depends on the users' preferences (which to a certain level can be controlled by tuning α, β).

Comparison cases. Five planner variants are used for comparison. *Ours* refers to the proposed method; *Human* refers the human driver in the dataset; *CNN* refers to a planner that maps BEV directly to controls (Implemented similarly to [28]); and *CNN-LSTM* refers the previous planner with an added LSTM component to keep track of history. *RvS-G* refers to the goal conditioned offline RL via supervised learning proposed in [29]. For all planners, the same CNN backbone is used to extract features from the rasterized BEV image (similar

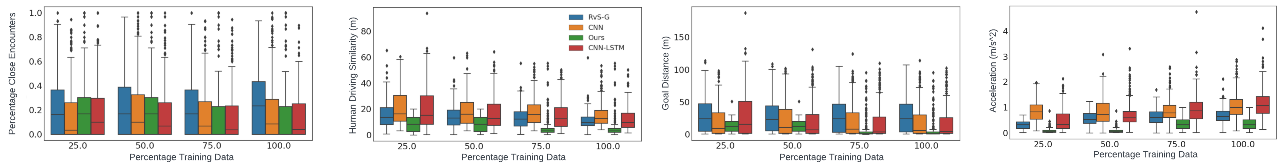

Fig. 5 : Sample and model efficiency study. Model performance trained at 25%, 50%, 75% and 100% training data. Our approach is able to achieve relatively high human driving similarity (low ADE) with a small percentage of training data.

to [30]). For all cases other than *Human*, we use the same CNN backbone to process the input BEV image. In the table, FC denotes fully connected layer, F denotes number of filters, K denotes kernel size, S denotes stride, U denotes number of units in the fully connected layer. For *CNN*, we concatenate the CNN features with the goal pose, which are passed through 2 FC layers that output speed and steering. For *CNN-LSTM*, the flattened features of the CNN backbone with concatenated goal pose are passed through 1 FC layer. The output of the FC layer is passed to an LSTM (with 64 dimensional hidden state) as input, which in turn provides speed and steering. This is a modified architecture of [31] that is used for behavior cloning of self-driving policies. RvS-G refers to the goal conditioned offline RL via supervised learning proposed in [29] which has been shown to outperform a number of behavior cloning and RL methods. In our implementation, we pass the goal and the current BEV image into the same CNN feature extractor, the concatenated feature vector is passed to the LSTM for control generation. In addition to providing the current BEV image, we also provide the goal BEV image as input.

B. Results and discussion

vAGN learns an explainable behavior planner that attends to semantically meaningful regions on the BEV. Figure 3 illustrates 3 time-steps during navigation of a roundabout. Within each sub-figure we also show the current vAGN state distribution (darker the color means higher probability) and the saliency map indicating where each q-state is attending to. Color indicates the level of attention and is ranks from high to low as: yellow $\rightarrow$ red $\rightarrow$ cyan $\rightarrow$ magenta. To the right of each q-distribution plot we also show the semantically colored BEV image sent as input to the network. In Figure 3 (a)(b), the ego vehicle is driving in an area of sparse traffic, the 2 vAGN nodes with highest probabilities are q_2 and q_3 both of which focuses on the undrivable areas which helps to prevent collision with those areas. As the ego comes to an intersection with ado vehicles in front (Figure 3 (c)), some probabilities shift from q_2 to q_3 where q_3 puts more attention of the ado vehicles. Finally, as the ego vehicle navigates out of the aroundabout (Figure 3 (d)), the q-states goes back to attending the undrivable areas. Please see the video attachment for full execution runs. Note that The meaning of each q-state depends on the training and the q-states may not always possess clear semantic meanings in human terms (i.e. turning, accelerating, etc). The execution trace in Figure 3 shows that q2 corresponds to avoiding collision with the road boundary (more weight on q3 when the ego vehicle comes close to the road boundary). Whereas q3 corresponds to avoiding vehicle-to-vehicle collision (from Figure 3-b to 3-c, q3's weighting increased as the ego vehicle comes close to the ados vehicles in the roundabout.)

vAGN learns to modulate DMP parameters to exhibit safe behaviors. Recall in Section IV-C we described that the output of vAGN is used to control the DMP parameters which in turn controls the "aggressiveness" of goal reaching. In Figure 4 we show an example of how this is achieved. Figure 4 (a) shows the evolution of the DMP parameters within a scene execution. It is noticeable that there is an abrupt change in the parameters starting from step 15. Figure 4 (d) shows the BEV input to the network, during steps 15, 16 and 17 the ego vehicle is trying to make a left turn but comes very close to the boundry of the road, therefore vAGN decides to focus on the road boundries (shown by highest weighted nodes q_0, q_1, q_4 and their corresponding saliency maps), which controls the change in the DMP parameters. Figure 4 (c) shows how the shift in q-distribution effects the resultant speed and steering (controls). From step 15 onwards, this shift has resulted in a mild speed decrease and more aggressive steering relieve, both of which steers the ego vehicle away from the road boundary. Refer to the video supplementary for a complete execution of this scene (along with others).

vAGN strikes a balance between safety and comfort, and exhibits near human driving behavior. Table I shows the close-loop rollout performance of all comparison cases over the validation set. We evaluated each model with 3 random seeds and Table I reports the mean performance with standard deviation. Our method is able to achieve the best goal reaching performance and similarity to human driving (ADE). Because each scene rollout is fixed maximum time-step (20 seconds sampled at 2hz or 40 steps defined by the dataset), driving conservatively (high safety and comfort scores) is a trade-off to goal achievement. We can see that *CNN* and *RvS-G* achieves the best safety and comfort scores respectively but in turn performs less than ideal in other metrics. In comparison, our method exhibits safety and comfort level similar to that of the human driver in the dataset.

The multi-abstractive neural controller achieves high sample efficiency. Because of the DMP, our neural controller already has a level of lane following capabilities built-in (even before any training). Given this structure, we can expect our controller to exhibit improved sample efficiency. This is shown in Figure 5 (a) where we train on 4 different levels sub-training set (evaluation is done on the full validation set). The results show that our model is able to achieve relatively high human

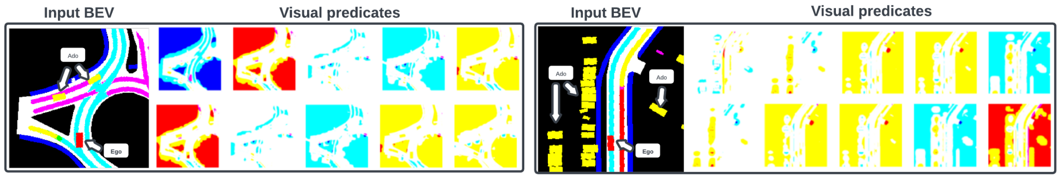

Fig. 6 : Visual predicate feature maps. visualizations of the feature maps (overlayed on the input BEV image) from the visual predicate extractor. Color indicates the level of attention and is ranks from high to low as: yellow → red → cyan → magenta

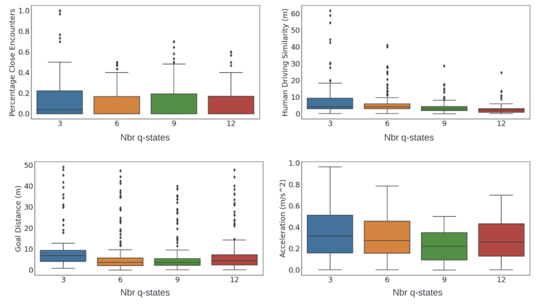

Fig. 7 : Study on varying the number of q-states. With increasing number of q-states, human driving similarity considerably improves. Beyond a minimal number of q-states, performance of vAGN can improve but tuning is required to obtain the desired balance between performance and explainability.

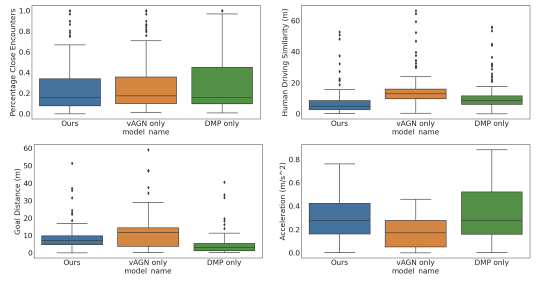

Fig. 8 : Self-ablation study. vAGN helps in learning coarse and interactive maneuvers whereas DMP is responsible for accurate and sample efficient navigation (e.g. reference path following). The combination of the two addresses different components of driving

TABLE I: Performance Comparison

Model	%Close Encounters	Acceleration
	mean ± std	**mean ± std**
Human	19.0%	0.30
CNN	**11.0% ± 2.0%**	0.92 ± 0.32
CNN-LSTM	12.2% ± 3.2%	1.20 ± 0.25
RvS-G	26.8% ± 5.0%	**0.22 ± 0.05**
Ours	16.2% ± 1.3%	0.43 ± 0.10
Model	**Human Driving Sim.**	**Goal Distance**
	mean ± std	**mean ± std**
Human	n/a	n/a
CNN	25.30 ± 8.20	14.20 ± 2.80
CNN-LSTM	19.23 ± 5.30	19.80 ± 3.10
RvS-G	14.25 ± 2.80	32.91 ± 6.30
Ours	**5.87 ± 1.40**	**8.20 ± 1.50**

driving similarity (low average displacement error) even when trained with a small training set. Because of its structure, the size of our model is also significantly smaller (more than 30% less parameters) than comparison models as shown in Figure 5 (b). This is important for planning and control modules as smaller models promote higher inference frequencies at runtime.

vAGN facilitates the CNN feature extractor to learn semantically meaningful visual predicates. Figure 6 shows visualizations of the feature maps (overlayed on the input BEV image) from the visual predicate extractor. Color indicates the level of attention and is ranks from high to low as: yellow → red → cyan → magenta. In both examples, we can see that the visual predicates attend to semantically meaningful components of the road (lanes, road boundaries, ado vehicles, etc). For the current design of the visual predicate extractor, we have found empirically that the objects each visual predicate is attending to evolves over time (during execution) and there are at times duplications (as shown in both figures). Looking into this behavior will be future work. Overall, these predicates and their corresponding feature vector provides the basis for vAGN to perform appropriate transitions which in turn controls the DMP to output interactive behaviors.

Altering the number of q-states trade off between explainability and performance. We treat the number of q-states as a hyperparameter that requires tuning. The trade-off is between explainability and performance (vAGN with less q-states are more explainable, may learn more semantically meaningful driving modes at each node, but with reduced overall performance). To study the effect of vAGN size on the neural controller, we conducted a study where the number of q-states are varied from 3 to 12 at increments of 3 and the results are presented in Figure 7 . The trend that stood out is that with increasing number of q-states, human driving similarity improves significantly. This is because as number of weights (hence the representative power) of vAGN improves as the number q-states increase. The performance of other metrics also improves with the number of q-states (especially when $\#q > 3$). However, the ego vehicle starts to drive more aggressively at $\#q = 12$ as shown by the jump in acceleration. In summary, beyond a minimal number of q-states, performance of vAGN can improve but tuning is required to obtain the desired balance between performance and explainability.

Within the multi-abstractive neural controller, vAGN takes care of safety and interactive behaviors and DMP is responsible for driving along lanes. In Figure 8 , we study the influence of the two main components - vAGN and DMP on our hierarchical neural controller. As a reminder, our controller follows a $CNN \rightarrow vAGN \rightarrow DMP$ structure. In the figure, *Ours* refers to the architecture with both components. *vAGN only* refers to $CNN \rightarrow vAGN \rightarrow FC$. *DMP only* refers to $CNN \rightarrow DMP$. The results for *DMP only* shows that DMP contributes most to goal reaching and human driving similarity, which is reasonable because DMP takes the main responsibility for lane following. However, with only DMP, the ego vehicle drives aggressively (shown in acceleration and percentage close encounter to nearby cars). Meanwhile, using vAGN improves the safety aspect of driving (lower acceleration and close encounter) because it learns to attend to road boundaries and ado vehicles and issues commands to avoid them. But vAGN alone performs less ideally in reaching the goal and human driving similarity is also worse. In a nutshell, vAGN helps in learning coarse and interactive maneuvers whereas DMP is useful for accurate navigation (e.g. reference path following). The combination of the two addresses different components of driving.

VI. Conclusion

In this work, we introduced the multi-abstractive neural controller which is a differentiable representation of a simplified plan/control stack. Within which we introduced the visual automaton generative network that acts as a behavior with learnable structure. We show that just by using supervised learning this policy representation is able to achieve high sample efficiency and a well balanced performance in terms of safety, optimality and comfort. Because of its structure, the decision making process of vAGN is highly interpretable. We show that it learns to attend to semantically meaningful regions of the input image while making transitions among (learned) modes of operations.

References

[1] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun, “End-to-end interpretable neural motion planner,” CVPR, pp. 8652–8661, 2019.

[2] T. Salzmann, E. Kaufmann, M. Pavone, D. Scaramuzza, and M. Ryll, “Neural-mpc: Deep learning model predictive control for quadrotors and agile robotic platforms,” ArXiv, vol. abs/2203.07747, 2022.

[3] J. Sacks and B. Boots, “Learning to optimize in model predictive control,” ICRA, 2022.

[4] J. Ichnowski, P. Jain, B. Stellato, G. Banjac, M. Luo, F. Borrelli, J. E. Gonzalez, I. Stoica, and K. Goldberg, “Accelerating quadratic optimization with reinforcement learning,” in NeurIPS, 2021.

[5] C. Dawson, S. Gao, and C. Fan, “Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods,” ArXiv, vol. abs/2202.11762, 2022.

[6] A. J. Ijspeert, J. Nakanishi, and S. Schaal, “Movement imitation with nonlinear dynamical systems in humanoid robots,” ICRA, vol. 2, pp. 1398–1403 vol.2, 2002.

[7] X. Meng, N. D. Ratliff, Y. Xiang, and D. Fox, “Neural autonomous navigation with riemannian motion policy,” ICRA, pp. 8860–8866, 2019.

[8] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, “Learning-based model predictive control for autonomous racing,” IEEE Robotics and Automation Letters, vol. 4, pp. 3363–3370, 2019.

[9] A. Faust, O. Ramirez, M. Fiser, K. Oslund, A. G. Francis, J. Davidson, and L. Tapia, “Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning,” 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 5113–5120, 2018.

[10] R. Fox, R. Berenstein, I. Stoica, and K. Goldberg, “Multi-task hierarchical imitation learning for home automation,” in CASE, 2019.

[11] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva, and A. Torralba, “Learning deep features for discriminative localization,” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2921–2929, 2015.

[12] M. Bojarski, A. Choromańska, K. Choromanski, B. Firner, L. D. Jackel, U. Muller, and K. Zieba, “Visualbackprop: visualizing cnns for autonomous driving,” ArXiv, vol. abs/1611.05418, 2016.

[13] M. Lechner, R. M. Hasani, A. Amini, T. A. Henzinger, D. Rus, and R. Grosu, “Neural circuit policies enabling auditable autonomy,” Nature Machine Intelligence, vol. 2, pp. 642–652, 2020.

[14] D. A. Hudson and C. D. Manning, “Learning by abstraction: The neural state machine,” in NeurIPS, 2019.

[15] L. Kochiev, “Neural state machine for 2D and 3D visual question answering,” 2021.

[16] A. Y. Hannun, V. Pratap, J. Kahn, and W.-N. Hsu, “Differentiable weighted finite-state transducers,” ArXiv, vol. abs/2010.01003, 2020.

[17] N. R. Ke, S. Chiappa, J. X. Wang, J. Bornschein, T. Weber, A. Goyal, M. Botvinic, M. C. Mozer, and D. J. Rezende, “Learning to induce causal structure,” ArXiv, vol. abs/2204.04875, 2022.

[18] S. Starke, H. Zhang, T. Komura, and J. Saito, “Neural state machine for character-scene interactions,” ACM Transactions on Graphics (TOG), vol. 38, pp. 1 – 14, 2019.

[19] M. Poli, S. Massaroli, L. Scimeca, S. J. Oh, S. Chun, A. Yamashita, H. Asama, J. Park, and A. Garg, “Neural hybrid automata: Learning dynamics with multiple modes and stochastic transitions,” in NeurIPS, 2021.

[20] Z. C. Lipton, “The mythos of model interpretability,” Queue, vol. 16, pp. 31 – 57, 2016.

[21] M. Narayanan, E. Chen, J. He, B. Kim, S. J. Gershman, and F. Doshi-Velez, “How do humans understand explanations from machine learning systems? an evaluation of the human-interpretability of explanation,” ArXiv, vol. abs/1902.00006, 2018.

[22] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, “Dynamical movement primitives: Learning attractor models for motor behaviors,” Neural Computation, vol. 25, pp. 328–373, 2013.

[23] F. J. Abu-Dakka, B. Nemec, J. A. Jørgensen, T. R. Savarimuthu, N. Krüger, and A. Ude, “Adaptation of manipulation skills in physical contact with the environment to reference force profiles,” Autonomous Robots, vol. 39, pp. 199–217, 2015.

[24] M. Lin, Q. Chen, and S. Yan, “Network in network,” CoRR, vol. abs/1312.4400, 2014.

[25] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva, and A. Torralba, “Learning deep features for discriminative localization,” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2921–2929, 2016.

[26] L. Koutras and Z. Doulgeri, “A correct formulation for the orientation dynamic movement primitives for robot control in the cartesian space,” in CoRL, 2019.

[27] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, “nuscenes: A multimodal dataset for autonomous driving,” arXiv preprint arXiv:1903.11027, 2019.

[28] W. Farag and Z. Saleh, “Behavior cloning for autonomous driving using convolutional neural networks,” 2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT), pp. 1–7, 2018.

[29] S. Emmons, B. Eysenbach, I. Kostrikov, and S. Levine, “Rvs: What is essential for offline rl via supervised learning?” arXiv preprint arXiv:2112.10751, 2021.

[30] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric, “Multimodal trajectory predictions for autonomous driving using deep convolutional networks,” ICRA, pp. 2090–2096, 2019.

[31] W. Farag and Z. Saleh, “Behavior cloning for autonomous driving using convolutional neural networks,” in 2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT). IEEE, 2018, pp. 1–7.