Technical Appendix for the Paper: Sample-Specific Output Constraints for Neural Networks

Mathis Brosowsky [1], **Olaf Dünkel** [2], **Florian Keck** [2], **Marius Zöllner** [1,2]

[1]FZI Research Center for Information Technology, {brosowsky, zoellner}@fzi.de
[2]Karlsruhe Institute of Technology, {olaf.duenkel, florian.keck}@student.kit.edu

In this document, we provide additional information for the experiments of the paper, *e.g.* implementation details, further explanations, and interpretations. For a better flow of reading, we repeat the content of short parts of the paper at some points.

Consistent Facial Landmark Detection

We construct ConstraintNet by modifying ResNet50 (He et al. 2016) which allows a comparison to the original ResNet50 without constraints. For a second comparison, we substitute the constraint guard layer of ConstraintNet with a differentiable optimization layer in form of a projection as described in the paper. In a first experiment, we model constraints to confine the landmark detections for nose, left eye, and right eye to a bounding box which might be given by a face detector. Subsequently, we extend these constraints and enforce relative positions such as *the eyes are above the nose*. In a second experiment, we predict only the nose landmark under constraints in form of triangles and sectors of a circle. The constraints of both experiments are visualized in Figure 1.

Dataset and Preprocessing. We perform the experiments on the *Large-scale CelebFaces Attributes (CelebA) dataset* (Liu et al. 2015). This dataset consists of a training (162 771 images), validation (19 867 images), and test set (19 961 images) with in total 202 599 images, each picturing a face. In addition, *CelebA* provides five landmark annotations among other attributes. In our experiments, we perform detections for the landmarks *nose*, *left eye*, and *right eye*. Furthermore, we rescale, crop, and normalize the images. After preprocessing, the images are of size 224×224 pixels.

Modified ResNet50 Architecture. ResNet50 (He et al. 2016) is a common neural network architecture which is used for many classification and regression tasks in computer vision (Lathuilière et al. 2018). In the case of regression, the prediction is usually generated by a final dense layer with linear activations. We construct ConstraintNet by modifying ResNet50 according to Figure 2 in the paper. As described in the paper, the modifications comprise adapting the output dimension of the final dense layer with linear activations to match the required dimension of z, adding the constraint guard layer ϕ for the considered class of constraints $\mathfrak{C}$, and inserting a representation $g(s)$ of the constraint parameter s at the stage of intermediate layers. The representation $g(s)$ is defined as tensor and the channels $c \in \{1, \dots, \dim(s)\}$ are identified with the components of the constraint parameter s, then we set all entries within a channel to a rescaled value of the corresponding constraint parameter component s_c:

$$g_{c,w,h}(s) = \lambda_c \cdot s_c, \tag{1}$$

with $w \in \{1, \dots, W\}$, $h \in \{1, \dots, H\}$, W and H for the width and height of the tensor and each λ_c is a rescaling factor. However, in practice we define g according to Equation (1) with the minor modification that we introduce only one channel for identical constraint parameter components s_c which occur several times in s. We concatenate $g(s)$ to the output of the 22nd layer of ResNet50 and choose matching dimensions $W = H = 28$. The 22nd layer is the last layer in the second of four building blocks of ResNet50. We argue that at this stage already high-level abstractions of the image are extracted and therefore a joint processing with the constraint parameter representation is reasonable. An extensive evaluation to find the optimal layer might be interesting for future studies. Furthermore, we found empirically that ConstraintNet learns effectively when the factors λ_c rescale

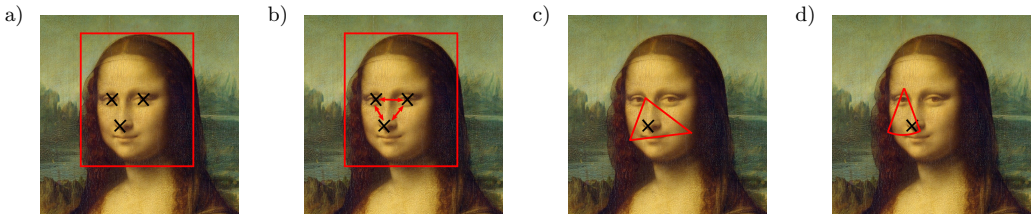

Figure 1: Visualization of different constraints for evaluation of ConstraintNet. a) Landmark detections for nose, left and right eye are confined to a bounding box around the face. b) In addition to the bounding box constraints, relations between landmarks are introduced, namely the eyes are above the nose and the left eye is in fact to the left of the right eye. c) and d): The nose landmark is constrained to a domain in form of a triangle c) or a sector of a circle d), respectively.

s_c to approximately the scale of the values in the output of the 22nd layer which is extended by $g(s)$. In the following sections, we provide the chosen values for λ_c depending on the considered class of constraints.

Bounding Box Constraints. In our first experiment, we perform a facial landmark detection for the nose $(\hat{x}_{\text{n}}, \hat{y}_{\text{n}})$, the left eye $(\hat{x}_{\text{le}}, \hat{y}_{\text{le}})$, and the right eye $(\hat{x}_{\text{re}}, \hat{y}_{\text{re}})$. We arrange the output of the neural network according to:

$$\hat{y} = (\hat{x}_{\text{n}}, \hat{x}_{\text{le}}, \hat{x}_{\text{re}}, \hat{y}_{\text{n}}, \hat{y}_{\text{le}}, \hat{y}_{\text{re}}), \tag{2}$$

and denote the x-coordinates $\hat{y}_{k_x}$ with $k_x \in \{1, 2, 3\}$ and the y-coordinates $\hat{y}_{k_y}$ with $k_y \in \{4, 5, 6\}$. The bounding box is specified by a left boundary $l^{(x)}$, a right boundary $u^{(x)}$, a top boundary $l^{(y)}$, and a bottom boundary $u^{(y)}$. Note that the y-axis starts at the top of the image and points downwards. Confining the landmark detections to a bounding box is equivalent to constrain each $\hat{y}_{k_x}$ to the interval $[l^{(x)}, u^{(x)}]$ and each $\hat{y}_{k_y}$ to the interval $[l^{(y)}, u^{(y)}]$ independently. These intervals are 1-dimensional convex polytopes with the interval boundaries as vertices. Thus, we can write the output constraints for the components with the definition in Equation (6) of the paper as:

$$\mathcal{C}^{(k_x)}(s^{(k_x)}) = \mathcal{P}(\{l^{(x)}, u^{(x)}\}), \tag{3}$$
$$\mathcal{C}^{(k_y)}(s^{(k_y)}) = \mathcal{P}(\{l^{(y)}, u^{(y)}\}), \tag{4}$$

with $s^{(k_x)} = (l^{(x)}, u^{(x)})$ and $s^{(k_y)} = (l^{(y)}, u^{(y)})$. The constraint guard layers of the components are given by Equation (7) in the paper:

$$\phi^{(k_x)}(z^{(k_x)}, s^{(k_x)}) = \sigma_1(z^{(k_x)})l^{(x)} + \sigma_2(z^{(k_x)})u^{(x)}, \tag{5}$$
$$\phi^{(k_y)}(z^{(k_y)}, s^{(k_y)}) = \sigma_1(z^{(k_y)})l^{(y)} + \sigma_2(z^{(k_y)})u^{(y)}, \tag{6}$$

with $z^{(k_x)}, z^{(k_y)} \in \mathbb{R}^2$ and σ the 2-dimensional softmax function. Finally, the overall constraint guard layer $\phi(z, s)$ can be constructed from the constraint guard layers of the components according to Equation (10) in the paper and requires a 12-dimensional intermediate variable $z \in \mathbb{R}^{12}$. The constraint parameter representation $g(s)$ has four channels corresponding to the boundaries of the box. We choose a rescaling factor of $\lambda_c = 0.01$ for each of the four channels.

For training of ConstraintNet, we sample valid constraint parameters randomly ($\text{sample}(\mathcal{S}_{y_i})$ in Algorithm 1 of the paper). To achieve this, we create randomly bounding boxes around the face covering the considered facial landmarks. As a first step, we determine the smallest rectangle (parallel to the image boundaries) which covers the landmarks nose, left eye, and right eye. Next, we sample four integers from the range $[20, 60]$ and use them to extend each of the four rectangle boundaries independently. The sampled constraint parameter is then given by the boundaries of the generated box $l^{(x)}, u^{(x)}, l^{(y)}, u^{(y)}$. For test, we determine the constraint parameter with the same sampling procedure as in training. This acts as a test how well ConstraintNet performs under bounding boxes with high variability in size.

Enforcing Relations between Landmarks. The bounding box constraints are extended to model relations between landmarks. The constraints for additionally enforcing that the left eye is in fact to the left of the right eye ($\hat{x}_{\text{le}} \leq \hat{x}_{\text{re}}$) and that the eyes are above the nose ($\hat{y}_{\text{le}}, \hat{y}_{\text{re}} \leq \hat{y}_{\text{n}}$) are given by three independent constraints for the output parts $\hat{y}^{(1)} = \hat{x}_{\text{n}}$, $\hat{y}^{(2)} = (\hat{x}_{\text{le}}, \hat{x}_{\text{re}})$, and $\hat{y}^{(3)} = (\hat{y}_{\text{n}}, \hat{y}_{\text{le}}, \hat{y}_{\text{re}})$. The independent constraint classes $\{\mathcal{C}^{(k)}(s^{(k)})\}_{k=1}^{3}$ are given by Equations (11, 12, 13) in the paper with constraint parameters $s^{(1)} = s^{(2)} = (l^{(x)}, u^{(x)})$ and $s^{(3)} = (l^{(y)}, u^{(y)})$. Figure 3 in the paper visualizes the constraints for the output parts: $\mathcal{C}^{(1)}$ is a line segment in 1D, $\mathcal{C}^{(2)}$ is a triangle in 2D, and $\mathcal{C}^{(3)}$ is a pyramid with 5 vertices in 3D. These objects are all convex polytopes and are defined according to Equation (6) in the paper via the vertices:

$$\mathcal{C}^{(1)}(s^{(1)}) = \mathcal{P}\big(\{v^{(i)}\}_{i=1}^{2}\big) \quad \text{with } v^{(1)} = l^{(x)}, v^{(2)} = u^{(x)}, \tag{7}$$

$$\mathcal{C}^{(2)}(s^{(2)}) = \mathcal{P}\big(\{v^{(i)}\}_{i=1}^{3}\big) \quad \text{with } v^{(1)} = (l^{(x)}, l^{(x)}), v^{(2)} = (l^{(x)}, u^{(x)}), v^{(3)} = (u^{(x)}, u^{(x)}), \tag{8}$$

$$\begin{aligned}\mathcal{C}^{(3)}(s^{(3)}) = \mathcal{P}\big(\{v^{(i)}\}_{i=1}^{5}\big) \quad \text{with } & v^{(1)} = (l^{(y)}, l^{(y)}, l^{(y)}), v^{(2)} = (u^{(y)}, l^{(y)}, l^{(y)}), \\ & v^{(3)} = (u^{(y)}, l^{(y)}, u^{(y)}), v^{(4)} = (u^{(y)}, u^{(y)}, u^{(y)}), \\ & v^{(5)} = (u^{(y)}, u^{(y)}, l^{(y)}).\end{aligned} \tag{9}$$

Note, that the $v^{(i)}$s are different vertices for different output parts and we skipped the index for the output part for simplicity. Consequently, the constraint guard layers for the parts $\{\phi^{(k)}\}_{k=1}^{3}$ are given by Equation (7) in the paper. Note that $\phi^{(k)}$ requires an intermediate variable $z^{(k)}$ with dimension equal to the number of vertices of the corresponding polytope, *i.e.* $z^{(1)} \in \mathbb{R}^2$, $z^{(2)} \in \mathbb{R}^3$, and $z^{(3)} \in \mathbb{R}^5$. Finally, the overall constraint guard layer ϕ is given by combining the parts according to Equation (10) in the paper and depends on an intermediate variable $z = (z^{(1)}, z^{(2)}, z^{(3)})$ with dimension $2{+}3{+}5 = 10$. The constraint parameter representation g and the generation of bounding boxes are the same as those for the bounding box constraints without enforcing the relative arrangement. Note that the introduced relations between the landmarks might be violated under rotations of the image and we consider them for demonstration purposes.

Triangle Constraints. A triangle is a convex polytope with three vertices. Therefore, the constraint guard layer is given by Equation (7) of the paper and requires a 3-dimensional intermediate variable $z \in \mathbb{R}^3$. We specify the triangle constraint with a 6-dimensional constraint parameter s which consists of concatenated vertex coordinates. Furthermore, we define the representation $g(s)$ according to Equation (1) with a rescaling factor of $\lambda_c = 0.01$ for all six constraint parameter components. To sample constraint parameters in training and test, we define a procedure to generate triangles with random shape around the nose landmark. First, we create an equilateral triangle on an imaginary circle around the nose landmark with radius of 55.5 px. This corresponds to an area of $4000\,\text{px}^2$ covered by the equilateral triangle. Next, we consider the triangle in a polar coordinate system with the point of origin at the center of the triangle. For randomization, we sample for each triangle vertex deviations for the radius coordinate from $[-30, 10]$ px and deviations for the angle coordinate from $[-\pi/6,\ \pi/6]$ rad independently. By applying these transformations, a sampling procedure for triangles is generated and it is ensured that the nose landmark is guaranteed within the triangle.

Sector of a Circle Constraints. A sector of a circle is defined in Equation (8) of the paper and the shape and position is specified via the constraint parameter $s = (x_{\text{c}}, y_{\text{c}}, R, \Psi)$. The constraint guard layer for constraints in form of sectors of a circle is given by Equation (9) of the paper and requires a 2-dimensional intermediate variable $z \in \mathbb{R}^2$. For the representation $g(s)$, we apply Equation (1) and choose for the first three constraint parameter components x_{c}, y_{c}, and R a rescaling factor of $\lambda_c = 0.01$ and for the fourth component Ψ a rescaling factor of $\lambda_4 = 1.0$. For training and test, we define a constraint parameter sampling procedure such that the nose landmark $y = (x_{\text{n}}, y_{\text{n}})$ is covered by the corresponding sector of a circle. We sample R from $[50, 100]$ px, Ψ from $[0.4, 1.0]$ rad, y_{c} from $[y_{\text{n}}{-}R, y_{\text{n}}]$, and x_{c} from $[x_{\text{n}}{-}\delta x, x_n{+}\delta x]$ with $\delta x = \lfloor \tan(\Psi/2)(y_{\text{n}}{-}y_{\text{c}}) \rfloor$.

Training. We train all models with the Adam (Kingma and Ba 2014) optimizer and use the mean squared error loss function without a regularization term R ($\lambda = 0$):

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 + \lambda R(\theta). \tag{10}$$

Method (Constraint)	Predicted landmarks	Training		Evaluation
		One epoch	Total (epochs)	
ConstraintNet (bb)	nose, left, and right eye	18 min 1 s	15 h 55 min (53)	(33.2 ± 0.2) s
projection (bb)		32 min 1 s	20 h 17 min (38)	(64.8 ± 0.5) s
ConstraintNet (bb+rel)		17 min 49 s	20 h 11 min (68)	(33.7 ± 0.4) s
projection (bb+rel)		31 min 45 s	24 h 52 min (47)	(67.5 ± 0.8) s
ResNet50 (None)		18 min 28 s	53 h 51 min (175)	(32.9 ± 0.2) s
ConstraintNet ($\triangle$)	nose	17 min 32 s	12 h 52 min (44)	(34.2 ± 0.7) s
projection ($\triangle$)		118 min 52 s	55 h 29 min (28)	(295.5 ± 0.8) s
ConstraintNet ($\mathcal{O}$)		18 min 25 s	10 h 44 min (35)	(33.4 ± 0.4) s
ResNet50 (None)		18 min 12 s	20 h 19 min (67)	(32.9 ± 0.2) s

Table 1: Runtimes for training and evaluation of ConstraintNet, the projection-based approach, and ResNet50 for the facial landmark detection tasks. Results are shown for the detection of the *nose*-, *left eye*-, and *right eye*-landmarks with constraints in form of bounding boxes (bb), additional constraints to enforce a certain relative arrangement (bb+rel) and no constraints (ResNet50). Furthermore, a detection of only the *nose*-landmark is performed and runtimes are shown for triangle constraints ($\triangle$), sector of a circle constraints ($\mathcal{O}$), and no constraints (ResNet50). Training is finished when overfitting begins and the corresponding number of epochs is denoted in brackets (compare the learning curves in Figure 4 of the paper). The projection-based approach requires fewer epochs than CosntraintNet. However, the total runtime for training is higher due to a higher runtime for a progress of one epoch (162 771 images). For ConstraintNet and ResNet50, the runtime per epoch is approximately $18\,\mathrm{min}$. However, ResNet50 requires more epochs for convergence. The evaluation runtime is measured for a detection and a computation of the mean squared error metric on the complete test set (19 961 images). The mean value and the standard deviation of the runtime are determined from 30 repetitions. ResNet50 is a little faster in evaluation than ConstraintNet due to the slightly smaller neural network architecture. The projection-based approach requires significant more runtime. Training and evaluation is performed with a batch size of 64 and the dataloader is parallelized on 10 threads. For training, the Adam optimizer is used with a learning rate of $5\mathrm{e}{-5}$. The runtimes are measured on a Lambda Quad (CPU: 12x Intel Core i7-6850K $3.6\,\mathrm{GHz}$, GPU: 4x Nvidia $12\,\mathrm{GB}$ Titan V, RAM: $128\,\mathrm{GB}$) using one of the four dedicated graphics cards.

The sampling procedure ($\mathrm{sample}(\mathcal{S}_{y_i})$ in Algorithm 1 of the paper) for the constraint parameter is described in the previous sections for the different classes of constraints separately.

For the bounding box constraints and the additional constraints to enforce a certain relative arrangement of the landmarks, we perform a simple hyperparameter optimization by a search over a discrete range of learning rates and batch sizes. First, we run five trainings with learning rates $[50, 10, 2, 0.5, 0.1] \cdot 1\mathrm{e}{-4}$ and a fixed batch size of 64 and then we pick the learning rate with lowest mean squared error on validation set and run additional five trainings with batch sizes $[8, 16, 32, 64, 128]$. For all experiments, we observe a learning rate of $5\mathrm{e}{-5}$ and a batch size of 64 as optimal values. For the triangle and sector of a circle constraints, we reuse these optimal values without performing a separate hyperparameter optimization. In all experiments, we stop training when overfitting begins and use the weights with lowest validation score for the evaluation on test set.

Runtime. For the facial landmark detection tasks, Table 1 shows the runtimes for training and evaluation of ResNet50, the projection-based approach, and ConstraintNet. The runtime for a training progress of one epoch is approximately equal for ConstraintNet and ResNet50. However, ConstraintNet requires only roughly the half the number of epochs for convergence in comparison to ResNet50. The projection-based approach requires even fewer epochs than ConstraintNet. However, the total runtime is higher because performing the projection requires additional resources and the runtime per epoch increases. For evaluation, ResNet50 requires a little less time than ConstraintNet due to the slightly smaller network architecture. The projection-based approach requires significant more runtime.

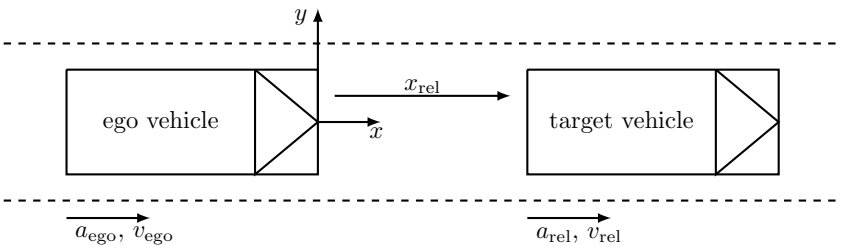

Figure 2: The follow object controller (FOC) in a vehicle (ego vehicle) is active when another vehicle (target vehicle) is ahead. Sensors measure the velocity of the ego vehicle v_{ego} and the distance x_{rel}, the relative velocity v_{rel}, and the relative acceleration a_{rel} of the target vehicle with respect to the coordinate system of the ego vehicle. The output of the FOC is a demanded acceleration $a_{\mathrm{ego,dem}}$ with the goal to keep a velocity dependent distance to the target vehicle under consideration of comfort and safety aspects.

Output Constraints on a Follow Object Controller for Safe Reinforcement Learning

This section provides further information on the follow object controller (FOC) presented as a second experiment the paper. Figure 2 shows the FOC situation in which the ego vehicle follows the target vehicle. The FOC is trained in a reinforcement learning (RL) setting using the Twin Delayed DDPG (TD3) algorithm (Fujimoto, Van Hoof, and Meger 2018). We use ConstraintNet as the policy which allows to impose constraints and compare it to a standard unconstrained policy, and an unconstrained policy with subsequent clipping on the constrained interval. All policies are compared regarding the training behavior, the crash rate during training, and several metrics to evaluate the learned behavior.

Simulated Environment and Traffic Scenarios. RL algorithms assume an agent interacting with its environment. Here, the FOC is the agent and the interaction with the environment is composed of the ego vehicle's longitudinal dynamics and the longitudinal trajectory of the target vehicle. The interaction of the agent with the environment is realized with a simulator. The state transitions are performed with $10\,\mathrm{Hz}$ frequency as a tradeoff between accurate vehicle dynamics and computational effort.

To model the ego vehicle's longitudinal dynamics, we consider the lagging behavior of real vehicles. The demanded acceleration $a_{\mathrm{ego,dem}}$ is filtered by a first order lag with a time constant of $T_{\mathrm{ego}} = 0.5\,\mathrm{s}$ to compute the ego vehicle's real acceleration a_{ego}. This acceleration is then integrated twice to obtain the velocity v_{ego} and absolute position x_{ego} of the ego vehicle.

The driving scenarios of the target vehicle are generated synthetically. One episode has a maximum length of five minutes. An episode is terminated prematurely if the ego vehicle collides with the target vehicle or if the ego vehicle starts to reverse. Then, the environment is reset and the next episode is simulated. The synthetically generated scenarios are based on an acceleration profile of the target vehicle. This profile is rescaled in time and amplitude at the beginning of each episode to cover acceleration values in the range of -1.5 $\mathrm{m\,s^{-2}}$ to $1.5\,\mathrm{m\,s^{-2}}$. This profile is integrated to calculate the target vehicle's velocity covering a velocity range of $0\,\mathrm{m\,s^{-1}}$ to $60\,\mathrm{m\,s^{-1}}$. On average every two simulated minutes a cut-in or cut-out of the target vehicle is simulated forcing the FOC to follow another vehicle. For the simulation of a cut-out, the current target vehicle leaves the ego vehicle's lane and the FOC has to approach to a vehicle further ahead. In case of a cut-in, another vehicle enters the ego vehicle's lane between the ego vehicle and the current target vehicle. In this situation, braking is required to ensure a safe distance.

Reward Function. We design the reward function to achieve functional, safe, and comfortable driving. Our reward function is based on Desjardin *et al*. (Desjardins and Chaib-draa 2011) which addresses safe and functional behavior. We extend this reward function to take comfortable driving into account.

For velocities above $12\,\mathrm{km\,h^{-1}} \approx 3.33\,\mathrm{m\,s^{-1}}$, the desired distance between ego and target vehicle is calculated according to $x_{\mathrm{rel,set}} = \tau_{\mathrm{set}} \cdot v_{\mathrm{ego}}$ with a desired time gap of $\tau_{\mathrm{set}} = 2\,\mathrm{s}$. For velocities below $12\,\mathrm{km\,h^{-1}} \approx 3.33\,\mathrm{m\,s^{-1}}$, we add continuously an offset distance of up to $3.2\,\mathrm{m}$ which is reached

when the ego vehicle stops:

$$x_{\mathrm{rel,set}} = \begin{cases} \tau_{\mathrm{set}} \cdot v_{\mathrm{ego}} & v_{\mathrm{ego}} > 3.33\,\mathrm{m\,s^{-1}}, \\ \tau_{\mathrm{set}} \cdot v_{\mathrm{ego}} + 3.2\,\mathrm{m}\left(1 - \frac{v_{\mathrm{ego}}}{3.33\,\mathrm{m\,s^{-1}}}\right) & 0 \leq v_{\mathrm{ego}} \leq 3.33\,\mathrm{m\,s^{-1}}. \end{cases} \tag{11}$$

Further, we define the current time gap τ according to:

$$\tau = \begin{cases} \frac{x_{\mathrm{rel}}}{v_{\mathrm{ego}}} & v_{\mathrm{ego}} > 3.33\,\mathrm{m\,s^{-1}}, \\ \frac{x_{\mathrm{rel}}}{v_{\mathrm{ego}} + v_{\mathrm{corr}}} & 0 \leq v_{\mathrm{ego}} \leq 3.33\,\mathrm{m\,s^{-1}} \text{ with } v_{\mathrm{corr}} = \frac{3.2\,\mathrm{m}}{\tau_{\mathrm{set}}}(1 - \frac{v_{\mathrm{ego}}}{3.33\,\mathrm{m\,s^{-1}}}). \end{cases} \tag{12}$$

The correction velocity v_{corr} takes the increased distance in the low velocity range into account. Note, $\tau = \tau_{\mathrm{set}}$ implies $x_{\mathrm{rel}} = x_{\mathrm{rel,set}}$. The part of the reward function for safe and functional behavior (Desjardins and Chaib-draa 2011) is based on the time gap τ and the derivation of the time gap $\dot{\tau}$. We extend this reward function to favour comfortable behavior. To achieve this, we add costs for the demanded acceleration and its change since these quantities influence the acceleration and jerk of the ego vehicle:

$$R_{\mathrm{comfort}}(s, a) = -c_1 \cdot a_{\mathrm{dem}}^2 - c_2 \cdot \dot{a}_{\mathrm{dem}}^2, \tag{13}$$

with $c_1 = 2\,\mathrm{s^4\,m^{-2}}$ and $c_2 = 0.005\,\mathrm{s^6\,m^{-2}}$.

Neural Networks for Policy. As an actor-critic method, the TD3 algorithm (Fujimoto, Van Hoof, and Meger 2018) learns a value function (critic) to assess the value of a state-action pair and a policy (actor) to generate continuous actions. Both the actor and the critic are implemented as feedforward neural networks with three fully connected layers. The actor neural network uses rectified linear units (ReLU) in the hidden layers. The activation function $a(x)$ of the output layer uses a hyperbolic tangent to confine the demanded acceleration a_{dem} to a fixed interval $[-4.5\,\mathrm{m\,s^{-2}}, 3\,\mathrm{m\,s^{-2}}]$ corresponding to the system limits:

$$a_{\mathrm{dem}} = -0.75\,\mathrm{m\,s^{-2}} + 3.75\,\mathrm{m\,s^{-2}} \cdot \tanh(x) \in [-4.5\,\mathrm{m\,s^{-2}}, 3\,\mathrm{m\,s^{-2}}]. \tag{14}$$

This fully connected neural network without sample-specific constraints is compared to a corresponding ConstraintNet with the safe interval $[a_{\min}, a_{\max}]$ as output constraint. As a third policy, the output of the unconstrained neural network a_{dem} is subsequently clipped to the safe interval $[a_{\min}, a_{\max}]$:

$$a_{\mathrm{dem,clipped}} = \min\left(\max(a_{\mathrm{dem}}, a_{\min}), a_{\max}\right). \tag{15}$$

The input of all three neural networks comprises a normalized representation of the state as input. The state consists of the velocity v_{ego} and acceleration a_{ego} of the ego vehicle, the distance x_{rel} to the target vehicle, and the relative velocity v_{rel} and relative acceleration a_{rel} of the target vehicle with respect to the ego vehicle as shown in Figure 2. ConstraintNet and the clipped unconstrained neural network get additionally normalized values of the acceleration bounds $a_{\min}$ and $a_{\max}$ as input.

Constraints. In case ConstraintNet or the unconstrained neural network with subsequent clipping is used as policy, the demanded acceleration $a_{\mathrm{ego,dem}}$ is confined to a safe interval $[a_{\min}, a_{\max}]$ in each forward pass independently. The derivation of our upper bound is closely related to the Responsibility-Sensitive Safety model (Shalev-Shwartz, Shammah, and Shashua 2017), however with relaxed assumptions. First, we claim requirements and state assumptions on our upper bound and then we propose an upper bound that satisfies these requirements.

We aim to prevent an undershooting of a minimum safety distance $d_{\mathrm{min,safety}}$ between the ego and the target vehicle. The safety distance is based on a time gap $\tau_{\min} = 1.5\,\mathrm{s}$ and a constant offset $d_{\mathrm{offset}} = 3.2\,\mathrm{m}$ to consider the case when the ego vehicle stops: $d_{\mathrm{min,safety}} = \tau_{\min} v_{\mathrm{ego}} + d_{\mathrm{offset}}$. For the target vehicle, we assume its current acceleration value to be constant and extrapolate its trajectory.

To decide if a current acceleration of the ego vehicle is safe or not, we consider the maneuver shown in Figure 3. Starting from the current time t_0, the ego vehicle accelerates with its current acceleration for a duration of its response time T_{ego}. Afterwards, the ego vehicle reduces its acceleration with a jerk limitation $j_{\max}$ until it reaches a constant maximum deceleration $a_{\mathrm{brake,min}}$ at time t_1. It then continues braking with $a_{\mathrm{brake,min}}$. The initial acceleration is considered safe if during the whole maneuver the minimal safety distance is not undershot. In this case, we require that the upper constraint is higher than the ego vehicle's acceleration: $a_{\mathrm{max,safety}} \geq a_{\mathrm{ego}}$.

If the trajectory of the ego vehicle would lead to a violation of the safety distance, we require for the upper constraint $a_{\mathrm{max,safety}}$ that it limits the demanded acceleration $a_{\mathrm{ego,dem}}$ so that after the

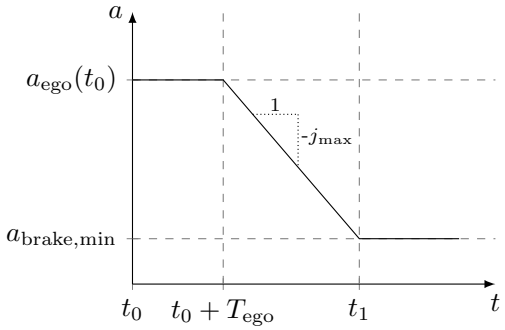

Figure 3: Maneuver for the decision if the current acceleration of the ego vehicle $a_{\text{ego}}(t_0)$ is safe: The ego vehicle's acceleration is constant for a duration of the response time T_{ego} and afterwards reduced with jerk limitation until $a_{\text{brake,min}}$ is reached. If during the whole maneuver the minimum distance is not violated the initial acceleration is considered safe and we require that the upper constraint is higher than the initial acceleration.

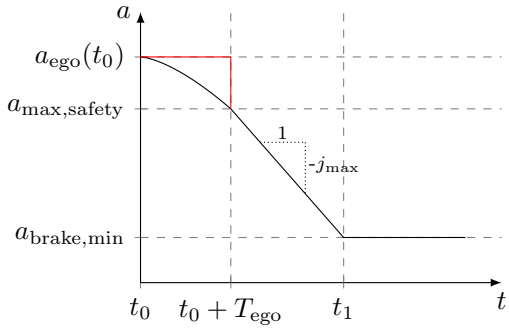

Figure 4: If the maneuver shown in Figure 3 would undershoot the safety distance, we require for the upper constraint $a_{\text{max,safety}}$ the following condition. If the ego vehicle reduces its acceleration within its response time T_{ego} to $a_{\text{max,safety}}$ and subsequently breaks with jerk limitation up to a maximum deceleration, the safety distance must be kept. We require this even for the worst case behavior of the ego vehicle which is indicated with the red line.

response time the ego vehicle's acceleration $a_{\text{ego}}(t)$ should be below $a_{\text{max,safety}}$ and a subsequent braking maneuver as shown in Figure 4 prevents a violation of the safety distance.

The response time of the ego vehicle is assumed to be $T_{\text{ego}} = 0.5\,\text{s}$. The minimum braking acceleration of the ego vehicle is set to $a_{\text{brake,min}} = -2\,\text{m}\,\text{s}^{-2}$ and the absolute jerk value to $j_{\max} = 0.5\,\text{m}\,\text{s}^{-3}$.

To compute an upper bound which meets the requirements defined above, we consider the worst case acceleration profile in Figure 4. First, we assume that the acceleration of the ego vehicle is constant for the duration of the response time. Next, we compute the highest acceleration after response time of the ego vehicle so that braking with jerk limitation up to the maximum deceleration just does not undershoot the safety distance. The computed acceleration yields then a valid upper bound $a_{\text{max,safety}}$: If the maneuver in Figure 3 is safe the chosen upper bound fulfills the condition $a_{\text{max,safety}} \geq a_{\text{ego}}(t_0)$. If the maneuver in Figure 3 is not safe, the requirement visualized in Figure 4 holds by definition. Note, that our upper bound may lead to collisions in rare cases if the assumptions are violated, *e.g.* if the ego vehicle does not reach the demanded acceleration during the assumed response time or if the response time is too slow resulting in an empty solution space for $a_{\text{max,safety}}$. For a more conservative bound, we refer to the Responsibility-Sensitive Safety model (Shalev-Shwartz, Shammah, and Shashua 2017). Furthermore, we limit the maximum demanded acceleration to $a_{\text{max,system}} = 3\,\text{m}\,\text{s}^{-2}$:

$$a_{\max} = \min\left(a_{\text{max,safety}}, a_{\text{max,system}}\right). \tag{16}$$

The lower bound $a_{\min}$ is chosen as the maximum allowed deceleration according to the operational limits in ISO15622 (The International Organization for Standardization 2018) and to prevent driving in reverse.

The operational limits defined in ISO15622 (The International Organization for Standardization 2018) request a velocity dependent maximum deceleration:

$$a_{\text{min,ISO}}(v_{\text{ego}}) = \begin{cases} -5\,\text{m}\,\text{s}^{-2} & 0 \leq v_{\text{ego}} \leq 5\,\text{m}\,\text{s}^{-1}, \\ -5\,\text{m}\,\text{s}^{-2} + \frac{1}{10}\left(v_{\text{ego}} - 5\,\text{m}\,\text{s}^{-1}\right) & 5\,\text{m}\,\text{s}^{-1} < v_{\text{ego}} < 20\,\text{m}\,\text{s}^{-1}, \\ -3.5\,\text{m}\,\text{s}^{-2} & v_{\text{ego}} \geq 20\,\text{m}\,\text{s}^{-1}. \end{cases} \tag{17}$$

To prevent reversing of the ego vehicle, we assume a jerk limit of $j_{\text{max,reverse}} = 3.5\,\text{m}\,\text{s}^{-3}$ for changing the acceleration of the ego vehicle. For a given positive velocity of the ego vehicle $v_{\text{ego}}(t_0) > 0$, we compute the maximum deceleration $a_{\text{ego}}(t_0) < 0$ which can be changed under the

Symbol	Value	Description
T_{Total}	$1\,000\,000$	Total number of training steps.
T_{Start}	$10\,000$	Number of initial steps with random exploration to fill experience replay buffer.
$n_{\mathrm{ReplaySize}}$	$1\,000\,000$	Size of replay buffer.
n_{Neurons}	256	Number of neurons in hidden layers.
n_{Batch}	256	Batch size used for training.
l_r	$3\mathrm{e}{-4}$	Learning rate of Adam optimizer (Kingma and Ba 2014) for actor and critic.
γ	0.99	Discount factor.
α	0.005	Factor for soft-update of target networks.
σ	$0.45\,\mathrm{m\,s^{-2}}$	Standard deviation of exploration noise.
$\widetilde{\sigma}$	$0.2\,\mathrm{m\,s^{-2}}$	Standard deviation of target policy noise.
c	$0.2\,\mathrm{m\,s^{-2}}$	Maximum target policy noise.
T_{Eval}	5000	Number of training steps between evaluation of learned behavior.
d	2	Number of training steps between policy improvement.

Table 2: Hyperparameters of TD3-algorithm.

jerk limitation $a_{\mathrm{ego}}(t) = a_{\mathrm{ego}}(t_0) + j_{\mathrm{max,reverse}} \cdot t$ to prevent reversing, *i.e.* $v_{\mathrm{ego}}(t) > 0\,\forall t$. Integrating acceleration over time yields:

$$v_{\mathrm{ego}}(t) = v_{\mathrm{ego}}(t_0) + a_{\mathrm{ego}}(t_0) \cdot t + \frac{1}{2} \cdot j_{\mathrm{max,reverse}} \cdot t^2. \tag{18}$$

The minimum acceleration $a_{\mathrm{ego}}(t_0)$ which satisfies $v_{\mathrm{ego}}(t) > 0\,\forall t$ is then:

$$a_{\mathrm{min,reverse}} = -\sqrt{2 \cdot j_{\mathrm{max,reverse}} \cdot v_{\mathrm{ego}}}. \tag{19}$$

To take the response time of $T_{\mathrm{ego}} = 0.5\,\mathrm{s}$ into account, we assume that the acceleration of the ego vehicle is constant during this time. The corrected minimum acceleration to prevent reversing is then given by:

$$a_{\mathrm{min,reverse,corr.}} = -\sqrt{2 \cdot j_{\mathrm{max,reverse}} \cdot (v_{\mathrm{ego}} + a_{\mathrm{ego}} \cdot T_{\mathrm{ego}})}. \tag{20}$$

Furthermore, the minimum acceleration by the system is limited to $a_{\mathrm{min,system}} = -4.5\,\mathrm{m\,s^{-2}}$. Finally, the final lower bound a_{min} is given by:

$$a_{\mathrm{min}} = \max\left(a_{\mathrm{min,ISO}}, a_{\mathrm{min,reverse,corr.}}, a_{\mathrm{min,system}}\right). \tag{21}$$

Training. We choose mainly the suggested hyperparameters of the author's (Fujimoto, Van Hoof, and Meger 2018) TD3 implementation. All chosen hyperparameters are shown in Table 2. For the policy, ConstraintNet, an unconstrained neural network and a clipped unconstrained neural network are applied. All neural networks are trained six times with different random initializations of the weights. In the paper, the average and standard deviations for several quantities are shown and determined over these six trials.

Evaluation. During training, every $T_{\mathrm{eval}} = 5000$ training steps each agent is evaluated for ten episodes to calculate its return and crash rate shown in Figure 5 and Figure 6 of the paper. During training, Gaussian noise is added to the demanded acceleration allowing the agent to explore its environment. This exploration noise is turned off during evaluation.

After training completed, the agents are evaluated for 100 episodes with a maximum duration of five minutes and several metrics are calculated to quantify comfort, safety, crash rate, and tracking error. We evaluate the agents after full training ($10\mathrm{e}5$ time steps) and with the weights at the training step when they reached the highest return. The results are shown in Table 2 of the paper.

The crash rate $M_{\mathrm{CrashRate}}$ (lower means safer) indicates the percentage of interrupted episodes due to a collision with the target vehicle. M_{Safety} (higher means safer) is the minimum occurring time gap within one episode normalized with the desired time gap of $\tau_{\mathrm{set}} = 2\,\mathrm{s}$ and averaged over episodes:

$$M_{\mathrm{Safety}} = \left\langle \min\left(\frac{\min_{t\in[1,T]}\tau_t}{\tau_{\mathrm{set}}}; 1\right)\right\rangle \in [0,1], \tag{22}$$

Neural network for policy	Training	Evaluation
ConstraintNet	$3\,\mathrm{h}\,22\,\mathrm{min} \pm 5\,\mathrm{min}$	$3\,\mathrm{min}\,44\,\mathrm{s} \pm 2\,\mathrm{s}$
Clipped unconstrained	$3\,\mathrm{h}\,8\,\mathrm{min} \pm 3\,\mathrm{min}$	$3\,\mathrm{min}\,19\,\mathrm{s} \pm 4\,\mathrm{s}$
Unconstrained	$2\,\mathrm{h}\,58\,\mathrm{min} \pm 1\,\mathrm{min}$	$2\,\mathrm{min}\,49\,\mathrm{s} \pm 2\,\mathrm{s}$

Table 3: Runtimes for training and evaluation of the follow object controller with different neural networks used as policies: ConstraintNet, an unconstrained neural network, and an unconstrained neural network with subsequent clipping to the safety interval. For training, the TD3 algorithm is applied for $10\mathrm{e}5$ time steps. For evaluation, the follow object controller performs 100 episodes with a maximum length of five minutes and the metrics $M_{\mathrm{CrashRate}}$, M_{Safety}, $M_{\mathrm{Discomfort}}$, M_{TrackErr} are computed. We measured the runtimes with a Lenovo ThinkPad X1 Extreme without using the dedicated graphics card (CPU: Intel Core i7-8750H $2.2\,\mathrm{GHz}$, RAM: $32\,\mathrm{GB}$) and show the mean value and the standard deviation over six trials.

with t the index for the time step within an episode $[1, T]$. $M_{\mathrm{Discomfort}}$ (lower means more comfortable) measures a weighted sum of squared acceleration and jerk of the ego vehicle averaged over time:

$$M_{\mathrm{Discomfort}} = \left\langle m_1 \frac{1}{T} \sum_{t=1}^{T} a_{\mathrm{ego},t}^2 + m_2 \frac{1}{T} \sum_{t=1}^{T} j_{\mathrm{ego},t}^2 \right\rangle. \tag{23}$$

The factors m_1 and m_2 set the weighting for acceleration and jerk. We choose $m_1 = 1\,\mathrm{s}^4\,\mathrm{m}^{-2}$ and $m_2 = 0.5\,\mathrm{s}^6\,\mathrm{m}^{-2}$. M_{TrackErr} (lower is better) is defined by the mean squared error of the actual time gap with respect to the desired time gap:

$$M_{\mathrm{TrackErr}} = \left\langle m_0 \frac{1}{T} \sum_{t=1}^{T} (\tau_{\mathrm{set}} - \tau_t)^2 \right\rangle, \tag{24}$$

with $m_0 = 1\,\mathrm{s}^{-2}$ for a dimensionless metric.

Runtime. Table 3 shows the measured runtimes for training and evaluation of the follow object controller with different neural networks used as actors. ConstraintNet and the clipped unconstrained neural network have longer runtimes than the unconstrained neural network. This can be explained by the computational costs for computing the upper and lower bound of the safety interval $[a_{\min}, a_{\max}]$. Furthermore, the runtimes for training and evaluation of ConstraintNet are more time-consuming then the clipped unconstrained neural network. Consequently, the relative overhead of the safety guard layer is measurable in small neural networks and continuously decreases with an increasing number of layers.

References

Desjardins, C.; and Chaib-draa, B. 2011. Cooperative Adaptive Cruise Control: A Reinforcement Learning Approach. *IEEE Transactions on Intelligent Transportation Systems* 12(4): 1248–1260.

Fujimoto, S.; Van Hoof, H.; and Meger, D. 2018. Addressing Function Approximation Error in Actor-Critic Methods. In *35th International Conference on Machine Learning, ICML 2018*, volume 4, 2587–2601. International Machine Learning Society (IMLS).

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 770–778.

Kingma, D. P.; and Ba, J. 2014. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations* URL http://arxiv.org/abs/1412.6980.

Lathuilière, S.; Mesejo, P.; Alameda-Pineda, X.; and Horaud, R. 2018. A Comprehensive Analysis of Deep Regression. *arXiv preprint arXiv:1803.08450* URL http://arxiv.org/abs/1803.08450.

Liu, Z.; Luo, P.; Wang, X.; and Tang, X. 2015. Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, 3730–3738.

Shalev-Shwartz, S.; Shammah, S.; and Shashua, A. 2017. On a Formal Model of Safe and Scalable Self-driving Cars. *arXiv preprint arXiv:1708.06374* .

The International Organization for Standardization. 2018. Adaptive Cruise Control. *ISO 15622* .