# A Analysis of the Risk-Constrained Minimization of the KL-Divergence

**Proposition A.1.** *Let $\mathcal{Y}$ be a connected set (in practice $\mathbb{R}^n$), $J^\pi : \mathcal{Y} \to \mathbb{R}^+$ a continuous cost function, $p$ the density function of the random vector $Y$ on $\mathcal{Y}$, $\sigma \in \mathbb{R}$ a risk-level, and $\mathcal{R}_p(J^\pi(Y), \sigma) \in \mathbb{R}$ a risk measure such that $\inf(J^\pi) < \mathcal{R}_p(J^\pi(Y), \sigma) < \sup(J^\pi)$. Then, there exists at least one density function $q_\psi$ of a random vector $Y|_\sigma$ on $\mathcal{Y}$ such that $\mathbb{E}_{q_\psi}[J^\pi(Y|_\sigma)] = \mathcal{R}_p(J^\pi(Y), \sigma)$.*

*Proof.* For any value $j^* \in ]\inf(J^\pi), \sup(J^\pi)[$, the intermediate value theorem states that there exist $y^* \in \mathcal{Y}$ such that $J^\pi(y^*) = j^*$. In particular, for any value of $\sigma$, we can choose $j^*_\sigma = \mathcal{R}_p(J^\pi(Y), \sigma)$. Let us now define the density function $q^*_\sigma : y \to \delta(y - y^*)$, with $\delta$ the Dirac delta density function. We obtain the equality $\mathbb{E}_{q^*_\sigma}[J(Y|_\sigma)] = j^* = \mathcal{R}_p(J(Y), \sigma)$, which proves that there is always at least one solution to equation (5). □

**Proposition A.2.** *Let $\Delta_\sigma = \{q \text{ s.t. } \mathbb{E}_q[J^\pi(Y)] = \mathcal{R}_p(J, \sigma)\}$. Let $q_\psi$ be the density function on $\mathcal{Y}$ that is parameterized by $\psi$ and $p$ the density function on $\mathcal{Y}$ from which the dataset is sampled.*

*Then, there exists a unique density function $q_{\psi^*} \in \Delta_\sigma$ that minimizes KL$(q_\psi||p)$.*

*Proof.* Let $q_1, q_2 \in \Delta_\sigma$ that both minimize $\mathrm{KL}(q_\psi||p)$ for a given $p$:

$$\mathrm{KL}(q_1||p) = \mathrm{KL}(q_2||p) = \min_{q_\psi \in \mathcal{R}_\sigma} \big(\mathrm{KL}(q_\psi||p)\big)$$

As a first step for this proof, we show that for any $\alpha \in [0, 1]$, we have $\alpha q_1 + (1 - \alpha)q_2 \in \Delta_\sigma$. Then, given that the function $q \to \mathrm{KL}(q||p)$ is strictly convex, we use the equality case of Jensen's inequality to show that $q_1 = q_2$ almost everywhere, which proves the uniqueness.

$$\begin{aligned}
\mathbb{E}_{\alpha q_1 + (1-\alpha)q_2}[J] &= \int J(y)(\alpha q_1(y) + (1 - \alpha)q_2(y))dy \\
&= \alpha \int J(y)\alpha q_1(y)dx + (1 - \alpha) \int J(y)q_2(y)dy \\
&= \alpha \mathbb{E}_{q_1}[J] + (1 - \alpha)\mathbb{E}_{q_2}[J] \\
&= \alpha R_p(J, \sigma) + (1 - \alpha)R_p(J, \sigma) \\
&= R_p(J, \sigma)
\end{aligned}$$

Therefore, for any $\alpha \in [0, 1]$, $\alpha q_1 + (1 - \alpha)q_2 \in \Delta_\sigma$.

The Jensen's inequality with $\mathrm{KL}(\cdot||p)$ gives us:

$$\mathrm{KL}(\alpha q_1 + (1 - \alpha)q_2||p) \leq \alpha \mathrm{KL}(q_1||p) + (1 - \alpha)\mathrm{KL}(q_2||p)$$

From our definition of $q_1$ and $q_2$:

$$\begin{aligned}
\mathrm{KL}(q_1||p) = \mathrm{KL}(q_2||p) &= \min_{q \in \Delta_\sigma} \big(\mathrm{KL}(q||p)\big) \\
&= \alpha \mathrm{KL}(q_1||p) + (1 - \alpha)\mathrm{KL}(q_2||p)
\end{aligned}$$

Since $\alpha q_1 + (1 - \alpha)q_2 \in \Delta_\sigma$ and is lower or equal to $\min_{q \in \Delta_\sigma} \big(\mathrm{KL}(q||p)\big)$, it is equal. Therefore, there is equality in the Jensen's inequality with a strictly convex function. This means that $q_1 = q_2$ almost everywhere and concludes our proof.

□

In this proof, the uniqueness is established for $q$ minimizing $\mathrm{KL}(q||p)$ in the data (i.e., trajectory) space, which is not what we do in practice. In (6) we minimize the KL-divergence from the prior in the latent space.

One might be tempted to define $l_1$ and $l_2$ as two biased density functions in the latent space such that $l_1 = q_1 \circ g_\theta$ and $l_2 = q_2 \circ g_\theta$. However, such $l_1$ and $l_2$ might not be density functions at all because they would not always integrate to one. We need to assume that $g_\theta$ is volume preserving for $l_1$ and $l_2$ to be well defined.

**Definition A.1.** *A differentiable function $g : \mathbb{R}^n \to \mathbb{R}^n$ is **volume-preserving** if $|\det(J_g(z))| = 1$ for all $z \in \mathbb{R}^n$, where $J_g$ is the Jacobian of $g$.*

Let us make two remarks about volume preservation in our application: First, it requires that $z$ and $y$ share the same dimension ($z$ and $y \in \mathbb{R}^n$) for the determinant of the Jacobian to be defined. This means that the information bottleneck of the CVAE is not straightforward to enforce. Second, with the CVAE assumptions, the elements of the latent variable should be independent, thus $J_{g_\theta}$ should be diagonal and $|\det(J_{g_\theta})| = |\nabla \cdot g_\theta|$.

**Proposition A.3.** *Let $q_\psi$ be a latent density function, $\rho$ a prior latent density function and $g_\theta$ (the decoder) be a volume-preserving function, all defined on $\mathbb{R}^n$. We suppose that $g_\theta$ fits the dataset such that $\mathcal{R}_p(J, \sigma) = \mathcal{R}_\rho(J \circ g_\theta, \sigma)$, with $p$ the data distribution on $\mathcal{Y} = \mathbb{R}^n$. Let us define the density function $l_\psi = q_\psi \circ g_\theta$ on $\mathbb{R}^n$. Finally, we define the constraint domain $\Delta_\sigma = \{l \text{ s.t. } \mathbb{E}_l[J \circ g_\theta] = \mathcal{R}_p(J, \sigma) = \mathcal{R}_\rho(J \circ g_\theta, \sigma)\}$.*

*Then, there exists a unique density function $l_{\psi^*} \in \Delta_\sigma$ that minimizes KL$(l_\psi||\rho)$.*

*Proof.* Let $l_1, l_2 \in \Delta_\sigma$ be two latent density functions that minimize the KL-divergence with the prior $\rho$:

$$\text{KL}(l_1||\rho) = \text{KL}(l_2||\rho) = \min_{l \in \Delta_\sigma} \big(\text{KL}(l||\rho)\big)$$

Let us show that $\alpha l_1 + (1-\alpha) l_2 \in \Delta_\sigma$:

$$\begin{aligned}
R_p(J, \sigma) &= \mathbb{E}_{\alpha q_1 + (1-\alpha) q_2}[J] \\
&= \int J(y)(\alpha q_1(y) + (1-\alpha) q_2(y)) dy \\
&= \int J(g_\theta(z))(\alpha q_1(g_\theta(z)) + (1-\alpha) q_2(g_\theta(z)))|\det(J_{g_\theta}(z))| dz \\
&= \int J(g_\theta(z))(\alpha l_1(z) + (1-\alpha) l_2(z)) dz \\
&= \mathbb{E}_{\alpha l_1 + (1-\alpha) l_2}[J \circ g_\theta]
\end{aligned}$$

And thus $\alpha l_1 + (1-\alpha) l_2 \in \Delta_\sigma$. The rest of the proof of Proposition A.2 holds.

□

# B Details about the neural network architectures

## B.1 Learning to forecast

In section 6, we perform experiments that rely on specific implementations of trajectory forecasting models. We use two different architectures: the first is a small MLP-based model used in our proof-of-concept experiments (Sections 6.1, 6.2, and 6.3), and the second resembles state-of-the-art forecasting architectures used in challenging, real-world scenarios (Section 6.4). As described in Section 4, our method relies on a latent space to bias the forecasts. We use CVAEs in this work, however, our method can be applied more broadly to any forecasting model that conditions on a latent sample.

The small model is composed of two multi-layer perceptron (MLP) encoders and one MLP decoder. The inference encoder takes the past trajectory $x$ and outputs the parameters of a Normal distribution $\mu|_x$ and $\log(\text{diag}(\Sigma|_x))$. The posterior encoder takes the whole past and future trajectory $x, y$ and outputs the parameters of a Normal distribution $\mu|_{x,y}$ and $\log(\text{diag}(\Sigma|_{x,y}))$. Finally, the decoder takes the past trajectory $x$ and a latent sample $z$ and outputs a prediction $y$. The second architecture is designed to model the interactions with both the other agents and map elements. It takes a similar form as the small model, but with additional context inputs and larger hidden dimensions. The social and map interactions are accounted for with a modified multi-context gating block [38].

## B.2 Model architecture

The first architecture is a simple model used in our proof-of-concept experiments. It is composed of three multi-layer perceptron (MLP) encoders and one MLP decoder:

- The inference encoder takes the past trajectory $x$ and outputs the parameters of a Normal distribution $\mu|_x$ and $\log(\text{diag}(\Sigma|_x))$.
- The posterior encoder takes the whole past and future trajectory $x, y$ and outputs the parameters of a Normal distribution $\mu|_{x,y}$ and $\log(\text{diag}(\Sigma|_{x,y}))$.
- The biased encoder takes the past trajectory $x$, a risk-level $\sigma$, and the robot future trajectory $y_{\text{robot}}$. It outputs the parameters of a Normal distribution $\mu^{(b)}$ and $\log(\text{diag}(\Sigma^{(b)}))$.
- The decoder takes the past trajectory $x$ and a latent sample $z$ and outputs a prediction $y$.

The time-sequence trajectories are flattened before fed into the model. Conversely, the output of the model is reshaped back into time-sequence trajectories. Each MLP is composed of 3 fully connected layers with a hidden dimension of 64 and ReLU activations. We chose a latent space dimension of 2 that is enough to show a working model and allows us to represent the latent space in 2D plots. The overall model is defined with 54.4K parameters.

Our second architecture resembles state-of-the-art forecasting architectures used in challenging, real-world scenarios. It is designed to model the interactions of the agent to be predicted with the surrounding agents and map elements. It also takes the form of a CVAE architecture and uses two MLP encoders and an MLP decoder similar to those described above, but with additional context inputs and larger hidden dimensions. We chose a latent space dimension of 16 because that gave satisfactory results in terms of final displacement error. The social and map interactions are accounted for with a modified multi-context gating block [38] composed of 3 context gating blocks. The context gating blocks each count three MLP modules with a hidden dimension of 256 (twice the input dimension). The MLP modules each count three layers and ReLU activations. Our modified context-gating block is represented Fig. 5. We stack these modified CG blocks with a running average of their output exactly as in [38].

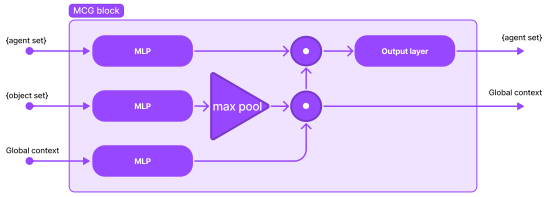



Figure 5: Diagram of the modified CG blocks (original CG blocks are defined in [38]). The circle dot is an element-wise multiplication (with each vector of the set).

The overall model represented in Fig. 6 counts 15.8M parameters.

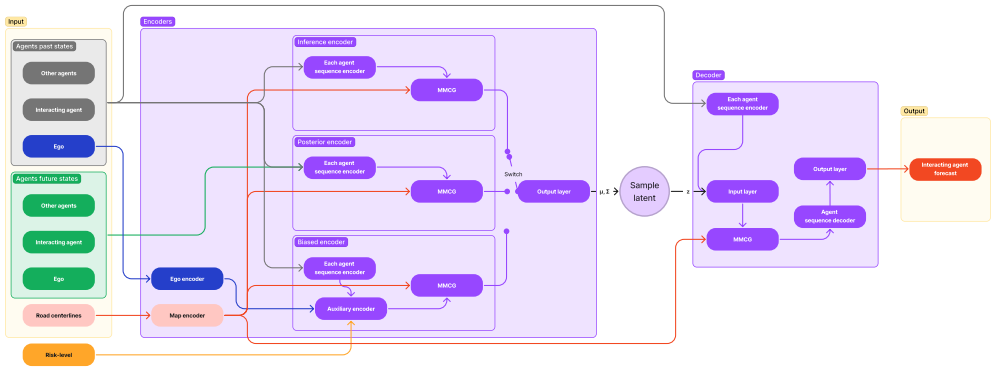



Figure 6: Diagram of the model used for the WOMD experiment. MMCG stands for modified multi-context gating blocks. MCG blocks are defined in [38].

### B.3 Model complexity

The CVAE model was trained for 6 hours 20 minutes on a single GPU Nvidia Titan Xp. Then its parameters are frozen and the biased encoder was trained for 4 days and 10 hours on the same GPU. This second training is time consuming because it involves the estimation of the risk with 64 then 256 samples which multiplies the tensor dimension and requires a smaller batch size to be used to fit in the GPU memory. This is exactly the computation overhead that our proposed method reduces at inference by reducing the number of samples needed for the risk estimation.

Because we only use fully connected layers, the overall complexity of the model is $O(b \times s \times a \times t \times f \times h) + O(b \times s \times a \times h^2) + O(b \times o \times m_s \times m_f \times h)$ with batch size $b$, number of samples $s$, number of agents $a$, hidden feature dimension $h$, time sequence length $t$, input feature dimension $f$, map element sequence length $m_s$, map input feature dimension $m_f$. With our choices of hyperparameters $s \times a \times t \times f > o \times m_s \times m_f$ and $t \times f > h$ so the complexity is $O(b \times s \times a \times (t \times f) \times h)$. At inference time, the number of samples $s$ can be kept small using our method and the batch dimension is 1. The most expensive operation is the first matrix multiplication but this operation can be easily parallelized and is often well optimized. The most limiting aspect might be the memory footprint. At test time with 20 samples, the allocated GPU memory reaches almost 2GiB.

## C Additional Experimental Details and Results

### C.1 Simulation Experiments

#### C.1.1 Biasing forecasts

Table 3 shows the distance metrics and risk error of the prediction model in our simulated environment (Fig. 1). In this didactic environment, the model is performing well with low distance-based errors and a low risk error.

Table 3: Motion forecasting error and risk estimation error on the simulation validation set. **minFDE (K)**: minimum final displacement error over K samples, **risk error (K)**: mean value of the signed difference between the average cost of the biased forecasts over K samples and the risk estimation using the unbiased forecasts, **risk |error| (K)**: mean value of the absolute values of the risk estimation error.

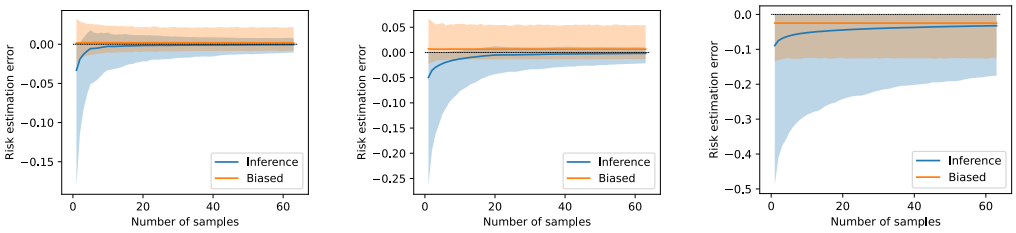
| Predictive Model | $\sigma$ | minFDE (16) | FDE (1) | Risk error (4) | Risk $\lvert$error$\rvert$ (4) |
|---|---|---|---|---|---|
| Unbiased CVAE | NA | $0.45 \pm 0.00$ | $0.81 \pm 0.00$ | NA | NA |
| Biased CVAE (RAP) | 0 | $0.48 \pm 0.00$ | $0.80 \pm 0.00$ | $0.00 \pm 0.00$ | $0.01 \pm 0.00$ |
| Biased CVAE (RAP) | 0.3 | $0.51 \pm 0.00$ | $0.80 \pm 0.00$ | $0.00 \pm 0.00$ | $0.01 \pm 0.00$ |
| Biased CVAE (RAP) | 0.5 | $0.54 \pm 0.00$ | $0.82 \pm 0.00$ | $0.00 \pm 0.00$ | $0.01 \pm 0.00$ |
| Biased CVAE (RAP) | 0.8 | $0.62 \pm 0.01$ | $0.91 \pm 0.00$ | $0.00 \pm 0.00$ | $0.01 \pm 0.00$ |
| Biased CVAE (RAP) | 0.95 | $0.74 \pm 0.01$ | $1.05 \pm 0.00$ | $0.01 \pm 0.00$ | $0.01 \pm 0.00$ |
| Biased CVAE (RAP) | 1 | $0.81 \pm 0.01$ | $1.11 \pm 0.00$ | $-0.02 \pm 0.00$ | $0.03 \pm 0.00$ |

Fig. 7 compares the Monte-Carlo risk estimation under the unbiased prediction (noted inference) and our proposed approach for risk estimation using biased predictions (noted biased). The plots show the average risk estimation error over the validation set as functions of the number of samples. The reference risk is computed with a Monte-Carlo risk estimation using 4096 samples. These plots show that our method may over-estimate risk more often (the 95% quantile is higher for our method across all number of samples). It also shows that our method makes an almost unbiased risk estimation even with only one sample while the Monte-Carlo approach underestimates the risk more often (the 5% quantile is lower than our method especially at low numbers of samples).

#### C.1.2 Planning with a biased prediction

We provide additional details and results on the CEM planning experiment presented in Section 6.2. Starting from $y^{\text{init}}_{\text{robot}}$, CEM iteratively updates the planned robot trajectory $y_{\text{robot}}$ by first sampling $n^{\text{robot}}_{\text{samples}} = 100$ robot trajectories in each iteration, from a Gaussian distribution with the mean being the $y_{\text{robot}}$ of the previous iteration. For each trajectory, we evaluate the following objective and select $n^{\text{robot}}_{\text{elites}} = 30$ elite trajectories that achieve the lowest objective values:

$$\mathcal{L}_{\text{plan}}(y_{\text{robot}}) = \mathcal{R}_p(J^{y_{\text{robot}}}(Y), \sigma) + \|y_{\text{robot}} - y_{\text{ref}}\|^2_Q, \tag{8}$$




(a) Risk-level $\sigma = 0.8$ (b) Risk-level $\sigma = 0.95$ (c) Risk-level $\sigma = 1$

Figure 7: Risk estimation error using the mean cost of the biased forecasts (biased) or the Monte-Carlo estimate of the risk under the unbiased forecasts (inference) as functions of the number of samples at different risk levels. Shaded regions indicate the 5% and 95% quantiles. The "ground-truth" used as a reference is computed with a Monte-Carlo estimate of the risk under the unbiased forecast using 4096 samples.

wherein the first term is the risk measure of the TTC cost (see Section E) with respect to the stochastic human future trajectory $Y$, and the second term is the quadratic tracking cost with respect to a given reference trajectory $y_{\text{ref}}$ under a symmetric, positive semi-definite cost matrix $Q$. The risk is estimated using $n_{\text{samples}}$ prediction samples of $Y$, through the Monte Carlo CVaR estimator [68] (for the unbiased CVAE) or Monte Carlo expectation (for the proposed RAP). The reference trajectory $y_{\text{ref}}$ is chosen to be a constant velocity trajectory at the desired speed of 14 m/s. Once the elites are chosen, CEM updates $y_{\text{robot}}$ to be the average of the elites. This iteration is repeated $n_{\text{iter}} = 10$ times, resulting in the overall time complexity of $O(n_{\text{iter}} \times n_{\text{samples}}^{\text{robots}} \times n_{\text{samples}})$. In particular it is linear in $n_{\text{samples}}$. This is confirmed in Fig. 8, when the planner was run on an Intel(R) Xeon(R) W-3345 CPU @ 3.00GHz. Leveraging a GPU would further reduce the overall run-time, but would also come at the cost of GPU memory consumption and may hinder other modules (such as perception) from performing real-time information processing. We refer the reader to prior work [69, 70] for further details of CEM.

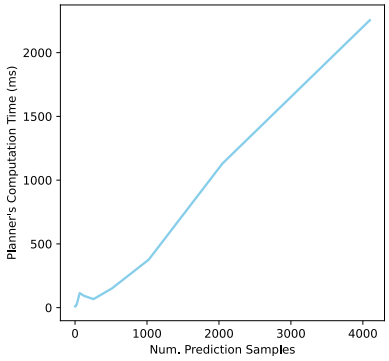



Figure 8: Overall computation time required by the planner at different values of $n_{\text{samples}}$.

In order to visualize the interplay of the two terms in the objective (8), we present in Fig. 9 the tracking cost plots from the same experiment as in Section 6.2. The only difference between Fig. 3 and Fig. 9 is that the y-axis of Fig. 3 shows the ground-truth TTC cost $J^{y^*_{\text{robot}}}(y)$ whereas that of Fig. 9 measures the tracking cost $\|y^*_{\text{robot}} - y_{\text{ref}}\|^2_Q$. The results depicted in Fig. 9 supports our claim in Section 6.2, which is that the planner over-optimistically optimizes trajectory tracking to the detriment of safety for the baseline CVAE predictor. Indeed, with fewer than 16 samples the baseline tends to achieve noticeably lower tracking costs at all risk-sensitivity levels. This is because the risk term in equation (8) is under-estimated due to the limited number of prediction samples. In turn, the optimization overestimates the relative importance of the tracking cost term. The proposed RAP predictor results in slightly higher tracking costs than the baseline at $\sigma = 0.8$ and $0.95$, but its performance is not affected by the number of prediction samples $n_{\text{samples}}$ that we draw from the risk-aware predictor for robust planning.

Lastly, Fig. 10 illustrates the difference in the optimized robot trajectory $y^*_{\text{robot}}$ between the baseline and the proposed RAP. The two trajectories of the robot provide a qualitative explanation to the statistical results presented in Fig. 3 and Fig. 9. That is, our proposed framework appropriately

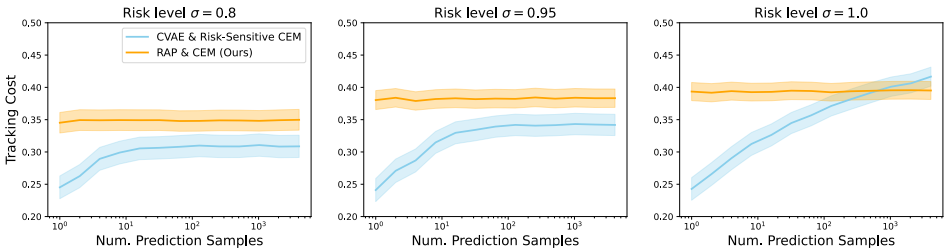
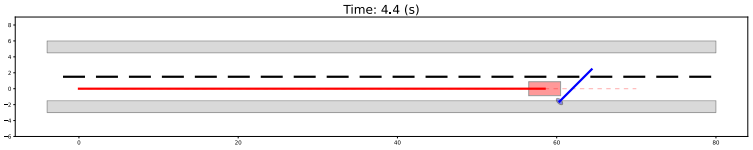
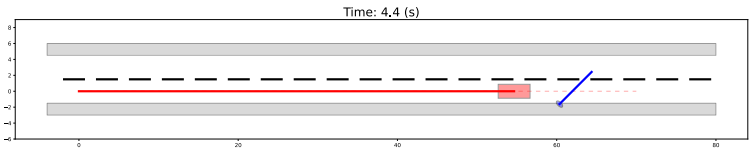



Figure 9: Tracking cost of the optimized $y^*_{\text{robot}}$, averaged over 500 episodes (lower the better). Ribbons show 95% confidence intervals of the mean.

slowed down the robot to keep more distance from the pedestrian, yielding a lower TTC cost and a higher tracking cost compared to the conventional risk-sensitive prediction-planning approach.




(a) CVAE & Risk-Sensitive CEM

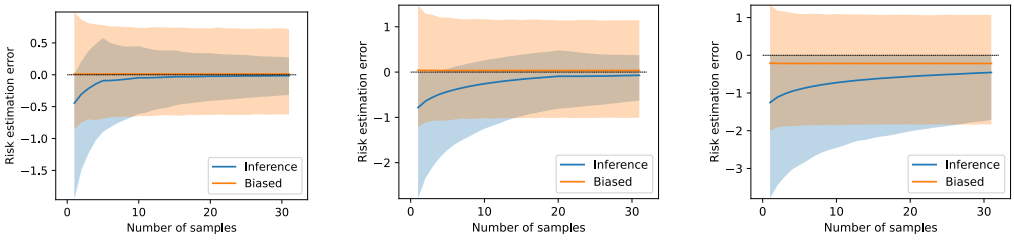



(b) RAP & CEM (Ours)

Figure 10: Optimized robot trajectory $y^*_{\text{robot}}$, executed and paused at $t = 4.4$ (s) in a test episode. Solid colored lines represent the executed trajectories of the robot and the pedestrian, and the dashed red line the robot's reference trajectory. (a) The robot almost collided with a pedestrian when the risk was taken into account in the planner. (b) When the risk was taken into account in the predictor, the robot slowed down slightly to keep more distance from the pedestrian. In both (a) and (b), the risk-sensitivity level was $\sigma = 0.95$ and $n_{\text{samples}} = 1$ prediction sample was given to the planner.

## C.2 Experiments on real-world data

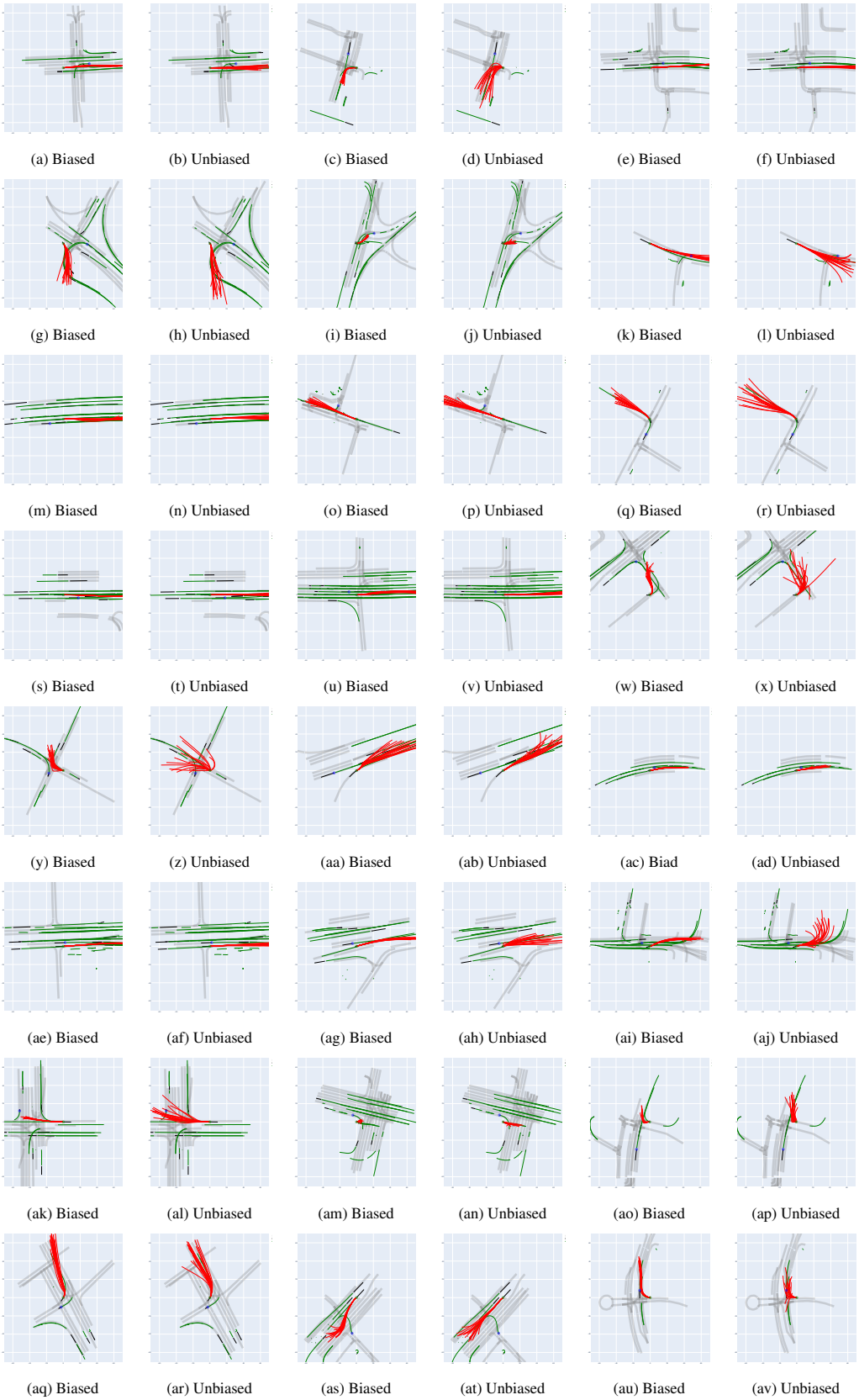
Fig. 11 compares the error from two methods for risk estimation as functions of the number of samples. Monte-Carlo risk estimation using the inference forecast distribution systematically underestimates the risk, especially with few samples. Our proposed method shows little estimation bias when the risk-level is below 0.95, and this relatively-small amount of bias is independent of the number of samples.




(a) Risk-level $\sigma = 0.8$ (b) Risk-level $\sigma = 0.95$ (c) Risk-level $\sigma = 1$

Figure 11: Risk estimation error using the mean cost of the biased forecasts (biased) or the Monte-Carlo estimate of the risk under the unbiased forecasts (inference) as functions of the number of samples at different risk levels. Shaded regions indicate the 5% and 95% quantiles.

Fig. 12 depicts many samples from the Waymo open motion dataset. It shows our results with biased and unbiased forecasts. Producing a good forecasting model is challenging, and as shown in the unbiased samples, our model counts a number of imperfections. One striking limitation is that the correlation between the map layout and the vehicle trajectories is not well captured (no lane-following behavior). This is a known limitation of trajectory forecasting models [72] that we do not solve



(a) Biased (b) Unbiased (c) Biased (d) Unbiased (e) Biased (f) Unbiased

(g) Biased (h) Unbiased (i) Biased (j) Unbiased (k) Biased (l) Unbiased

(m) Biased (n) Unbiased (o) Biased (p) Unbiased (q) Biased (r) Unbiased

(s) Biased (t) Unbiased (u) Biased (v) Unbiased (w) Biased (x) Unbiased

(y) Biased (z) Unbiased (aa) Biased (ab) Unbiased (ac) Biad (ad) Unbiased

(ae) Biased (af) Unbiased (ag) Biased (ah) Unbiased (ai) Biased (aj) Unbiased

(ak) Biased (al) Unbiased (am) Biased (an) Unbiased (ao) Biased (ap) Unbiased

(aq) Biased (ar) Unbiased (as) Biased (at) Unbiased (au) Biased (av) Unbiased

Figure 12: Sample results on WOMD. Each sample is depicted twice. Once with 16 samples of our biased model forecasts at risk-level $\sigma = 0.95$, and once with 16 samples of our unbiased model forecasts.

with our proposed forecasting model. In fact, merely defining what should be a "good" forecasting model is challenging. Good properties from a forecasting model could be: better interpretability (for example with a disentangled latent representation), higher dataset likelihood, more diverse forecasts, feasible predicted dynamics (acceleration and turn-rate in bounds), feasible predicted states (within drivable area bounds), accounting for many observations (agent types, different road elements, input uncertainties, etc...), accounting for unobserved hypothesis (ego plan, "what-if" [73], given maneuver, etc...). Improving the forecasting model before training the biasing encoder would probably improve our results in the same desired direction of improvement. In particular, restricting the state and dynamics to realistic ones might avoid some failure modes of the biased forecasts that achieve greater risk with unrealistic trajectories (see subfigures (g), (w), (y), (ao)). Learning to bias a model with an interpretable latent space could bring some exciting results by producing interpretable directions for cost in the latent space, thus yielding interpretable reasons to be cautious. A base forecasting model with a tighter likelihood fit would restrict the biased forecasts to ones that are closer to the trajectory distribution of the dataset, while a base model predicting more diverse trajectories would lead to a biased model that accounts for more possible costly outcomes. Improving all these aspects with the same model is challenging, but depending on the desired usage, the focus can be put on one or another good property.

# D Latent space exploration

In Section 6, we performed didactic experiments with a two dimensional latent space. This allows us to explore the latent representation with plots in the latent space.

Fig. 13 shows the cost associated with different latent samples, and a representation of the biased distributions at different risk-levels. It takes place in the situation described in Fig. 1. The biased distributions in latent space are represented with ellipses centered on the distribution mean $\mu$ with radii given by the standard deviation $\sqrt{\mathrm{diag}(\Sigma)}$ (the square root is applied on both terms). Notice that the unbiased latent distribution is neither centered on $(0, 0)$ nor has unit variance. This is because we did not train the model with respect to a prior $\mathcal{N}(0, I)$, but instead with respect to an inferred prior $\mathcal{N}(\mu|_x, \Sigma|_x)$. As the risk-level is increased, the latent mean is moved further from its starting point and the standard deviation becomes smaller. The biased distribution converges towards a riskier area in the nearby latent space. We can see on Fig. 13 that the direction of higher cost coincides with the direction of $\mu^{(b)}$ as the risk-level $\sigma$ is increased.

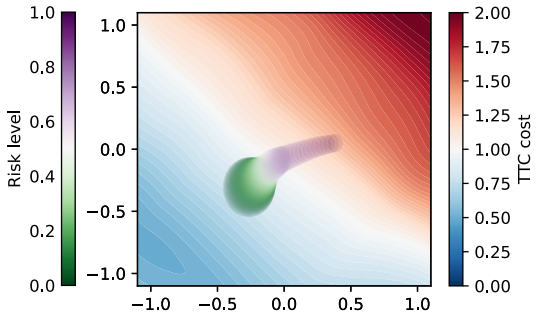



Figure 13: Representation of the latent space in the situation depicted in Fig. 1. On the blue to red scale, the cost of the decoded trajectories matching different latent values is mapped. On the green to purple scale, the encoded latent distributions from the biased encoder at different risk levels are represented as one-standard-deviation ellipses.

# E Time-to-collision cost

If a constant velocity model predicts an *imminent* collision, we can consider that it is not avoidable and the cost of the situation is high. This motivates the use of a time to collision cost (TTC cost) that corrects some short-comings of the distance based cost. It considers the danger due to greater relative speed and the anisotropy due to the velocity orientation.

Let us consider two agents $i$ and $j$ at positions $(x_i, y_i)$, $(x_j, y_j)$ and with velocities $(v_i^x, v_i^y)$, $(v_j^x, v_j^y)$. We want to compute a TTC cost for their relative states. The relative positions and velocities are

defined as:

$$(dx, dy) \triangleq (x^i - x^j, y^i - y^j), \tag{9}$$

$$(dv_x, dv_y) \triangleq (v_x^i - v_x^j, v_y^i - v_y^j). \tag{10}$$

Under the constant velocity assumption, the evolution of relative squared distance is given by:

$$d^2(t) = (dv_x t + dx)^2 + (dv_y t + dy)^2. \tag{11}$$

Finding the time to collision is finding $t_{\text{col}}$ at which $d^2(t_{\text{col}}) = 0$. Writing the relative speed $dv = \sqrt{dv_x^2 + dv_y^2}$ and the initial distance $d_0 = \sqrt{dx^2 + dy^2}$, we must solve the quadratic equation:

$$t_{\text{col}}^2 + 2t_{\text{col}} \frac{dv_x dx + dv_y dy}{dv^2} + \frac{d_0^2}{dv^2} = 0. \tag{12}$$

The solutions in $\mathbb{C}$ are:

$$t_{\text{col}}^+ = -\frac{dv_x dx + dv_y dy}{dv^2} + i\frac{dv_x dy - dv_y dx}{dv^2}, \tag{13}$$

$$t_{\text{col}}^- = -\frac{dv_x dx + dv_y dy}{dv^2} - i\frac{dv_x dy - dv_y dx}{dv^2}. \tag{14}$$

Of course, not every situation leads to a collision and when there is a collision, there is only one time of occurrence. Thus, we are not surprised that there is only one real solution when there is a solution. However, this assumes a collision only when the distance between agents is exactly $0$. We should relax this assumption to consider a cost when the relative distance is low.

The time of the lowest relative distance is given by the real part of the solution, and the distance at that time is given by the imaginary part multiplied by the velocity:

$$t_{\tilde{\text{col}}} = -\frac{dv_x dx + dv_y dy}{dv^2}, \tag{15}$$

$$d^2(t_{\tilde{\text{col}}}) = \frac{(dv_x dy - dv_y dx)^2}{dv^2}. \tag{16}$$

Using these two values, we want to define a cost function that penalize low relative distance in a near future. However, we must first consider two problematic cases: when $dv$ is close to $0$ and when $t_{\tilde{\text{col}}}$ is negative.

**When $dv$ is close to $0$,** the TTC would become large which is a low cost situation. We can simply consider a lowest possible value for $dv$ and use $\tilde{dv} \triangleq \max(dv, \varepsilon)$. This overestimates the cost in very low cost situations.

**When $t_{\tilde{\text{col}}}$ is negative,** the relative distance between the agents is increasing. Therefore, the actual TTC is infinite. Accounting for uncertainties, in this case, we do not set the TTC to infinity but fall back to the distance based cost by setting TTC to 0 and the distance at collision to the current distance.

The values used to define the cost are now:

$$\tilde{t}_{\tilde{\text{col}}} \triangleq \begin{cases} -\frac{dv_x dx + dv_y dy}{\tilde{dv}^2} & \text{if } t_{\tilde{\text{col}}} \geq 0, \\ 0 & \text{otherwise,} \end{cases} \tag{17}$$

$$\tilde{d}^2_{\tilde{\text{col}}} \triangleq \begin{cases} \frac{(dv_x dy - dv_y dx)^2}{\tilde{dv}^2} & \text{if } t_{\tilde{\text{col}}} \geq 0, \\ dx^2 + dy^2 & \text{otherwise.} \end{cases} \tag{18}$$

Using these two values, the instantaneous cost at $t = 0$ is given by :

$$J = \exp\left( -\frac{\tilde{t}^2_{\tilde{\text{col}}}}{2\lambda_t} - \frac{\tilde{d}^2_{\tilde{\text{col}}}}{2\lambda_d} \right). \tag{19}$$

For a finite-time prediction from $t = 0$ to the final time-step $T$, the cost of a trajectory is the average of the instantaneous costs along the trajectory:

$$J = \frac{1}{T} \sum_{t=0}^{T} \exp\left( -\frac{\tilde{t}^2_{\text{col}}(t)}{2\lambda_t} - \frac{\tilde{d}^2_{\text{col}}(t)}{2\lambda_d} \right), \tag{20}$$

with a time bandwidth parameter $\lambda_t$, and a distance bandwidth parameter $\lambda_d$. This cost is high if and only if the time to collision is low compared to the time bandwidth and the distance to collision is low compared to the distance bandwidth. This formula uses a constant velocity assumption that only holds at short time horizon, therefore a rather small time bandwidth should be chosen.

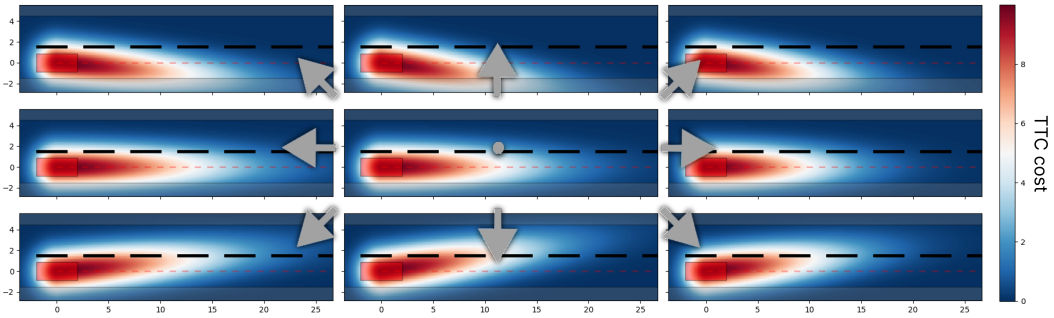



Figure 14: Cost maps of a single road scene with a car going at 14m/s from left to right. The instantaneous TTC cost associated with a second agent is represented by colors from red to blue at the different positions. In each image, the cost is computed at $t = 0$ for a second agent going at 2 m/s with an orientation represented by the arrows (except the center picture that considers a static second agent). The time bandwidth is 0.2 and the distance bandwidths is 2.