Prediction by Imagination: A prediction method for handling low-probable actions

Ershad Banijamali
Noah's Ark, Huawei Technologies, Canada
ershad.banijamali1@huawei.com

Mohsen Rohani
Noah's Ark, Huawei Technologies, Canada
mohsen.rohani@huawei.com

Abstract

We propose "Prediction by Imagination", a method for action-based observation prediction for self-driving cars. Our prediction model has three different modules. Two rule-based image processing modules and one prediction module. Only the prediction module has trainable parameters and the two rule-based modules help the prediction module learn a simpler task, i.e a prediction task in which only the position of the moving objects change from one frame to the next one and the rest remains the same. The prediction module does not take actions directly and learn the interactions between the cars during training. Therefore, it does not just learn how to react to our own actions but learns the dynamics of the whole traffic. We train our model in the framework of conditional varitional autoencoders (CVAEs) to maximize the evidence lower bound (ELBO) of the log-likelihood of a conditional observation distribution.

1 Introduction

Action-based prediction models have applications in different areas of machine learning [15, 10, 1]. Action-based prediction has two major benefits. First, an action-based prediction model can be used for training a planning model or a reinforcement learning (RL) agent, either end-to-end [17] or model by model [7]. Secondly, by changing the actions we can observe how the environment changes and generate new scenarios, without a real and potentially costly interaction with the environment. Both of these important downstream tasks heavily rely on the quality of the observations predicted by the model. In the case of autonomous vehicles this issue becomes even more important because of two main reasons: 1) taking safe actions is crucial in the case of self-driving cars. 2) Collecting data in which extreme scenarios, e.g. hard breaks or sharp change of steering wheel angel, is a hard task as these situations do not happen often in a real world. Therefore using a model that can handle such extreme actions and produce valid prediction will be helpful. An important reason that most of the current action-based prediction models cannot handle these type of actions is that the observation to be predicted changes almost entirely compared to the previous observations based on actions. Here we want to build a model that does not posses this drawback, i.e. at each step we want our prediction to have minimal changes compared to the input of the model.

Authors in [12] proposed a model for multi-step prediction of occupancy grid maps (OGM) where all the frames from past and future are mapped to a reference frame in which the ego vehicle (the vehicle for which we learn the prediction or do planning) is freezed and just moving objects in the scene change their location. There are two major differences here. First of all, unlike [12] we consider

Machine Learning for Autonomous Driving Workshop at the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

actions in our prediction. Therefore we can change the prediction based on changing the actions. Secondly, we do not freeze the ego car. Therefore, invalid scenarios, e.g. a car comes from behind and runs over our car, do not happen in our prediction.

Model-predictive policy with uncertainty regularization (MPUR) [7], is a state-of-the-art prediction and planning model in this area. Although the model is successful in predicting the effect of action within the range of training samples, in the case of extreme actions it fails to predict a valid observation.

A large body of literature on prediction tasks in self-driving cars is dedicated to object tracking [9, 6, 11, 4], which is a classical approach to tackle the problem of finding a model for decision making for autonomous vehicles. More recently, multimodal object tracking has become a popular topic in this area [5, 2, 16]. These methods are mostly based on generative models and variational inference that find the most probable paths for the objects in the environment. However, all of these methods need online object detection, which is computationally expensive and require labeled data. Moreover, any error in object detection can affect the whole system and result in catastrophic failure.

Here we propose a model that works almost similar to driving behavior of humans. Given current observation of the road and a history of the past observations, we as drivers decide to take an action. But, before immediately applying that action we *imagine* how such an action will change our position in the road and *predict* the reaction of other cars around us to such a change[3]. Here we also introduce the same idea, i.e. we first imagine how the actions change the position of the ego car in an image and then predict how the moving objects in the environment will react to it. In the next section we explain the components of our model in more details.

2 Prediction by Imagination

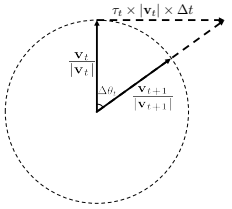
The prediction task that we consider in this work is described as follows. We are given a set of observations from the environment. Let's denote the observation at time t by $\mathbf{o}_t$. The problem is to predict the future k observations, $\mathbf{o}_{t+1:t+k}$, given the past observation $\mathbf{o}_{1:t}$ and a series of actions $\mathbf{a}_{t:t+k-1}$. The observations include (a) A bird's-eye view image in which the ego car has a fixed position, in the form of an OGM. We denote the image at time t by $\mathbf{i}_t$.(b) Position and velocity of the car in each direction, which are denoted by $\mathbf{p}_t$ and $\mathbf{v}_t$, respectively, and we refer them as the measurements. These two parts of the observation are the same in nature, i.e. both are sensory data from the vehicle. However, we are focused on learning a model that can predict the images, as the position and velocity can be deterministically computed based on the actions, as described in the next sections. This is why we call them with different names. The measurements are crucial for learning the dynamic of the system as well as predicting the future frames.

2.1 Base model

Our model consists of five blocks. The main block is a prediction module, which is a conditional observation prediction model in the framework of variational autoencoders (VAEs) [8]. There are three rule-based image processing blocks, two of them before the prediction module preparing input data, and the other rule-based module processes the output of the prediction module for the next time steps. The rule-based modules are not trainable and only parameters of the prediction module are learned. In this section we explain the structure and application of each of these modules in details.

2.1.1 Measurements estimator module

The actions that we consider are two-dimensional, which include acceleration, α, and rotation of the steering wheel, τ, $\mathbf{a}_t = [\alpha_t, \tau_t]$. Given these actions and measurements at each time step, the measurements for the next time step can be determined. This module also provides input for the image processing modules. The elements of translation and rotation matrices that are used in the image processing modules are computed in the measurement



$$\Delta\mathbf{p}_t = \mathbf{v}_t \Delta t$$
$$\mathbf{p}_{t+1} = \mathbf{p}_t + \Delta\mathbf{p}_t$$
$$|\mathbf{v}_{t+1}| = \alpha_t \Delta t + |\mathbf{v}_t|$$
$$\Delta\theta_t = \arctan(\tau_t |\mathbf{v}_t| \Delta t)$$

Figure 1: Computing the effect of actions on the future position, velocity and change in the direction of the car ($\Delta\theta_t$). Δt depends on the sample frequency.

estimator module. Fig. 1 shows how the actions change the velocity and direction of the car. In summary, the measurement estimator module acts as the following function:

$$\mathbf{p}_{t+1}, \mathbf{v}_{t+1}, \Delta\mathbf{p}_t, \Delta\theta_t = f_m(\mathbf{p}_t, \mathbf{v}_t, \mathbf{a}_t), \tag{1}$$

where the components are shown in Fig. 1.

2.1.2 Input image processing modules (IIPs)

The goal of designing our model is to make the task of prediction as easy as possible for the prediction module. Therefore we want the frame to be predicted next, $\mathbf{i}_{t+1}$, to have minimal changes compared to the last input frame, $\mathbf{i}_t$, i.e. we just want the moving objects in the next frame change their positions in the image and all other objects remain the same. To do so we use two image processing modules at the input of the prediction module. These two modules transform $\mathbf{i}_t$ and $\mathbf{i}_{t+1}$ based on $\Delta\theta_t$ and $\Delta\mathbf{p}_t$ from the measurement estimator modules.

The first component, IIP1, takes $\Delta\theta_t$ and $\Delta\mathbf{p}_t$ and image at time t, $\mathbf{i}_t$, and change the position of the ego car in the image accordingly, as if we *imagined* that action $\mathbf{a}_t$ has been applied to the car. We denote the output by $\mathbf{i}_{a_t}$. The other component, IIP2, takes $\Delta\theta_t$ and $\Delta\mathbf{p}_t$ and the next frame and remove the ego motion from the image, as if we are seeing this frame from the point of view of the car in $\mathbf{i}_t$. This module changes the position of the ego-car and all other objects in the image according to the action. We denote the output of IIP2 by $\mathbf{i}_{\mathbf{a}_{t+1}}$. Note that by these two transformations the positions of the ego car as well as other fixed objects in the image remain the same in both $\mathbf{i}_{\mathbf{a}_t}$ and $\mathbf{i}_{\mathbf{a}_{t+1}}$. Only moving objects change their position in these two images.

2.1.3 Prediction module

Prediction module is the only component of our model that is trainable. The task that is learned by this module is predicting $\mathbf{i}_{\mathbf{a}_{t+1}}$. This task is much easier for the prediction module, compared to predicting $\mathbf{i}_{t+1}$, since the geometry of the image remains the same as its input. Therefore the prediction module only needs to learn the dynamics of the moving objects in the image. No matter how harsh or mild the the driver's actions are, the location of the fixed items remain the same as the input image, $\mathbf{i}_{\mathbf{a}_t}$. This is especially important for the case of self-driving cars where the prediction model should reliably predict the future images online. The prediction module only needs to predict the location of the dynamic objects and rotations and translations do not need to be learned during the training of the model.

We solve the prediction problem in the framework of VAEs. Since this is a conditional prediction task we use the conditional VAE model (CVAE) [14]. A common problem with vanilla CVAE model is that it does not learn the distribution of its input and therefore is prone to overfitting. Therefore, we use the special case of CVAEs, called bottleneck conditional density estimation (BCDE) [13], in which the prior is also conditioned on the input. In our case this input is the set of previous observations.

We are interested in the conditional log-likelihood $\log p(\mathbf{o}_{t+1}|\mathbf{o}_{1:t}, \mathbf{a}_t)$. Since the image and measurements in $\mathbf{o}_{t+1}$ are conditionally independent we can write the log-likelihood in the following form:

$$\log p(\mathbf{o}_{t+1}|\mathbf{o}_{1:t}, \mathbf{a}_t) = \log p(\mathbf{i}_{t+1}|\mathbf{o}_{1:t}, \mathbf{a}_t) + \log p(\mathbf{p}_{t+1}, \mathbf{v}_{t+1}|\mathbf{o}_{1:t}, \mathbf{a}_t), \tag{2}$$

The second likelihood in Eq. 2 can be removed from our calculations since according to Eq. 1:

$$p(\mathbf{p}_{t+1}, \mathbf{v}_{t+1}|\mathbf{o}_{1:t}, \mathbf{a}_t) = p(\mathbf{p}_{t+1}, \mathbf{v}_{t+1}|\mathbf{p}_t, \mathbf{v}_t, \mathbf{a}_t) = \delta(f_m(\mathbf{p}_t, \mathbf{v}_t, \mathbf{a}_t)) \tag{3}$$

where $\delta(.)$ is the Dirac delta function.

Since both $\mathbf{i}_{\mathbf{a}_t}$ and $\mathbf{i}_{\mathbf{a}_{t+1}}$ are one-to-one functions of the action $\mathbf{a}_t$ (given $\mathbf{i}_t$ and $\mathbf{i}_{t+1}$), we can re-write the first term of Eq. 2 as $\log p(\mathbf{i}_{\mathbf{a}_{t+1}}|\mathbf{o}_{1:t}, \mathbf{i}_{\mathbf{a}_t})$. We consider the graphical model in Fig. 2 at each time step for this prediction task. We would like to maximize the evidence low-bound (ELBO) of the conditional likelihood $\log p(\mathbf{i}_{\mathbf{a}_{t+1}}|\mathbf{o}_{1:t}, \mathbf{i}_{\mathbf{a}_t})$.

According to our definition of the approximating variational distribution in the graphical model, and also considering $\mathbf{z}_t$ as an information bottleneck between the conditions, $\mathbf{o}_{1:t}$ and $\mathbf{i}_{\mathbf{a}_t}$, and the target, $\mathbf{i}_{\mathbf{a}_{t+1}}$, the ELBO will have the following form:

$$\begin{aligned}\log p(\mathbf{i}_{\mathbf{a}_{t+1}}|\mathbf{o}_{1:t}, \mathbf{i}_{\mathbf{a}_t}) \geq & \mathbb{E}_{q(\mathbf{z}_t|\mathbf{o}_{1:t}, \mathbf{i}_{\mathbf{a}_t}, \mathbf{i}_{\mathbf{a}_{t+1}})}[\log p(\mathbf{i}_{\mathbf{a}_{t+1}}|\mathbf{z}_t)] \\ & - \mathrm{KL}\big(q(\mathbf{z}_t|\mathbf{o}_{1:t}, \mathbf{i}_{\mathbf{a}_t}, \mathbf{i}_{\mathbf{a}_{t+1}})||p(\mathbf{z}_t|\mathbf{o}_{1:t}, \mathbf{i}_{\mathbf{a}_t})\big)\end{aligned} \tag{4}$$

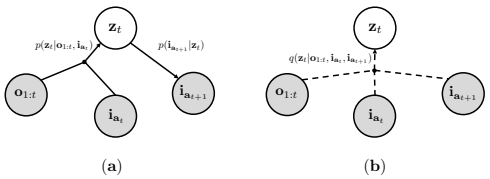


Figure 2: Graphical model of the prediction module. Gray circles are observed variables. (**a**) the generative links, $p(.)$. (**b**) Variational approximation links, $q(.)$.

We aim to maximize this ELBO. We implement each of the conditional probability distributions in Eq. 4 using a neural network and denote the parameters of $p(.)$ and $q(.)$ by ψ and ϕ, respectively. Authors in [7] also suggested a CVAE-based model for prediction. However, in their model the prior is not conditioned on the previous observations. Therefore, samples from the prior at the test time can be independent of the previous seen images. This can potentially hurt the performance of the prediction especially when the prediction horizon is large.

2.1.4 Output image processing module (OIP)

The OIP module takes the output of the prediction module and the action at time t and changes the image such that the ego car goes to the center of the image again. In fact, OIP module's function is the inverse of IIP2's. The output should be ideally the same as $\mathbf{i}_{t+1}$. This module is necessary for multi-step prediction where the current prediction is fed to the model as the input for the next step prediction.

Fig. 3 shows the components of the model and how they are connected to each other. We use convolutional layers for encoding and decoding the images and fully-connected layers for encoding the position and velocity.

2.2 Practical considerations

The first term in the ELBO in Eq. 4, can be interpreted as a reconstruction loss in the pixel space. For the reconstruction loss we compute a weighted sum of mean-squared error (MSE) between prediction and target. For the second term, we consider a Gaussian distribution for the conditional prior. However, since the observations are highly dynamic we set the variance of the distribution to zero, we split the latent code into two parts and for one part set the variance of the distribution to zero. The encoder of $p(.)$ and $q(.)$ distributions share parameters for the part with zero variance. For the part with non-zero variance, we try to match the output distributions by minimizing the KL-divergence according to Eq. 4 .At the test time, sampling from the prior will generate new scenarios.

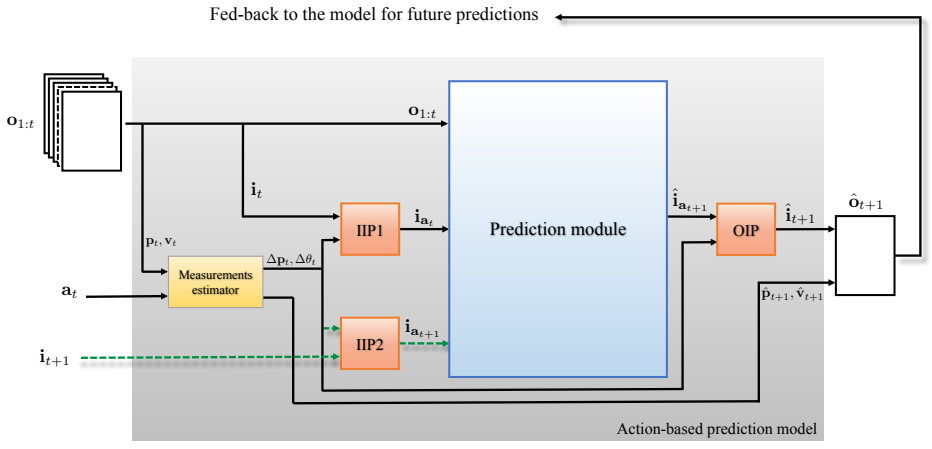


Figure 3: The prediction model with all of its components. For multi-step prediction the output of the model at each time step is fed-back to the model for prediction of the next steps. The dashed green links and IIP2 block only exist at the training time and they are disconnected at the test (inference) time.

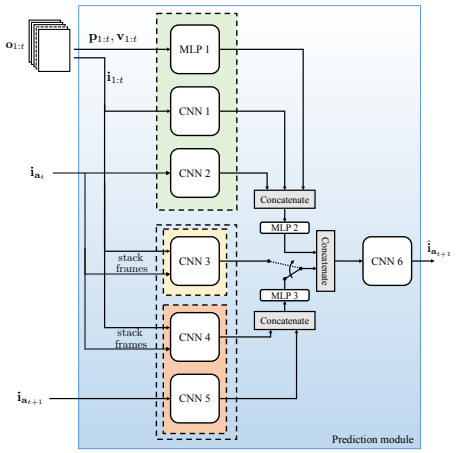


Figure 4: The components of prediction module. The top part of the encoder (green) is common between $p(.)$ and $q(.)$. The yellow block of the bottom part belongs to $p(.)$ and the red block belongs to $q(.)$. Some links are shown by double lines for better visualization.

The training objective for the model is to minimize the following expression:

$$\mathcal{L}_t = \mathcal{L}_t^{\text{rec.}} + \mathcal{L}_t^{\text{KL}} = \| \mathbf{i}_{\mathbf{a}_{t+1}} - \hat{\mathbf{i}}_{\mathbf{a}_{t+1}} \|^2 + \text{KL}\big(q_\phi(\mathbf{z}_t|\mathbf{o}_{1:t}, \mathbf{i}_{\mathbf{a}_t}, \mathbf{i}_{\mathbf{a}_{t+1}}) || p_\psi(\mathbf{z}_t|\mathbf{o}_{1:t}, \mathbf{i}_{\mathbf{a}_t})\big)$$

For a multi-step prediction task with horizon k a summation over $\mathcal{L}_t$ is minimized:

$$\min_{\psi,\phi} \sum_{j=0}^{k-1} \mathcal{L}_{t+j}^{\text{rec.}} + \mathcal{L}_{t+j}^{\text{KL}} \tag{5}$$

3 Experiments

In this section we compare the prediction capacity of our model with the forward-model in MPUR [7]. We use the Next Generation Simulation program's Interstate 80(NGSIM I-80) dataset. The dataset consists of 45 minutes of recordings (3 batches of 15 minutes recordings) from traffic cameras mounted over a stretch of highway. Behaviour of drivers are complex and difficult to predict. We follow the same preprocessing proposed in [7] to make the datasets. The images in this dataset are 117×24 with three channels (RGB). The ego-car is in the center of the blue channel. Other cars are in the green channel, which can be interpreted as the OGM. The red channel has the road map information, i.e. lines.

Prediction with regular actions: For actions that are high probable according to the training distribution, we train each model using one batch of 15 minutes recording and test it on the actions of the other two batches. Therefore the test actions are going to be within the same range of training actions. In this case, we have the ground truth for prediction. Therefore we can compute the loss based on mean square error (mse) for different prediction horizons k, averaged over k. Results are reported in table 1. Fig. 6 shows a sequence of predictions for different models. As we can see PI and FM-MPUR perform closely in the visual sense, as well.

Method	$k=1$	$k=5$	$k=10$	$k=20$
FM-MPUR	3.41 ± 0.87	4.72 ± 0.82	$\mathbf{5.24 \pm 1.1}$	7.82 ± 1.4
PI	$\mathbf{3.22 \pm 0.59}$	$\mathbf{4.58 \pm 0.88}$	5.66 ± 1.04	$\mathbf{6.04 \pm 1.34}$

Table 1: Mean squared error of predictions

Prediction for low-probable actions: For the low-probable actions, we use the trained models with each of the batches of 15 minutes recordings and apply actions that are rarely seen in the training set but are still in the dynamic range of vehicles. We use the distribution shown in Fig. 5 to sample these actions.

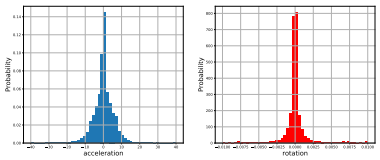


Figure 5: Distribution of actions in the training dataset.

Defining a quantitative evaluation metric for the performance of the models on these actions is not straightforward. Therefore, we apply such actions to 25 randomly selected sequences of the test set. Given

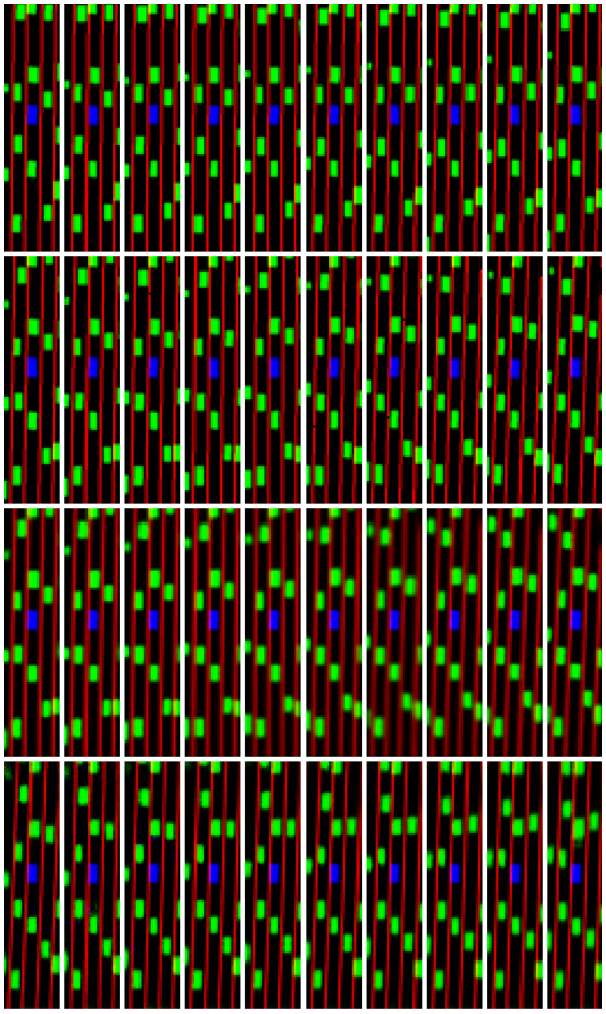


Figure 6: A sequence of predictions by different models. First row is the input sequence. Second row is the target sequence. Row three and four are the predictions of PI and FM-MPUR. models, respectively. For the FM-MPUR we used 20 input frames but only the last 10 frames are shown in the first row.

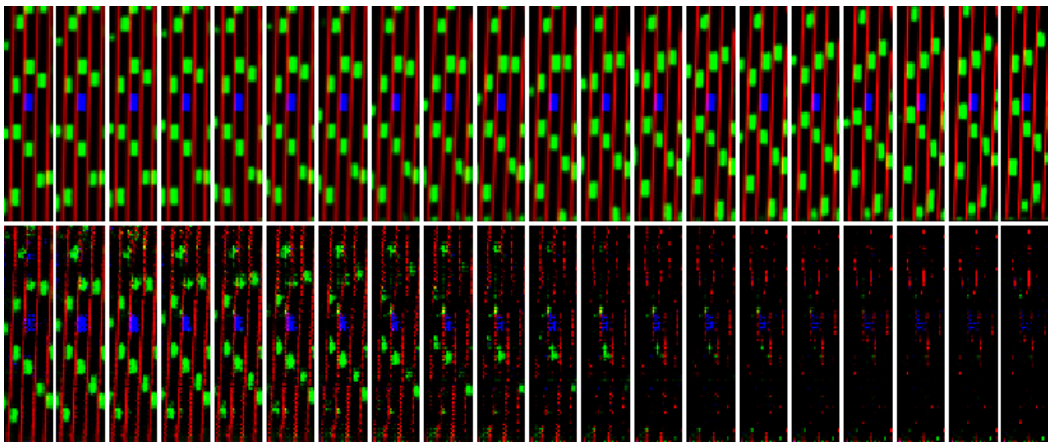


Figure 7: Effect of constantly applying low-probable actions on the prediction of PI (top row) and FM-MPUR models (bottom row).

an input sequence, we predict $k = 20$ images and count the number of visually corrupted images and call them invalid predictions. In table 2 we show the percentage of invalid predictions for the first $k = \{1, 2, 5, 10, 20\}$ predictions (over $k \times 25$ possible images). It shows that our model outperforms FM-MPUR significantly for this task. This is due to the fact that the prediction task for the prediction module in our model is much easier than the one that other module should handle.

Fig. 7 shows the result of applying one low-probable action to the ego-car. The input sequence is the same as Fig. 6. We apply $\mathbf{a}_t = [-35, 0]$ for 20 consecutive steps, which is a very low-probable action according to Fig. 5. This is equivalent to a hard break in the middle of the road. As we can see our model can predict almost perfectly, while the prediction of FM-MPUR model breaks after a few prediction. Compared to the predictions that correspond the original actions in Fig. 6, we can see that the ego-car gets closer to the car behind it and the car in front of it gets farther away.

Method	$k = 1$	$k = 5$	$k = 10$	$k = 20$
FM-MPUR	4	12	20.4	28.8
PI	**0**	**2.8**	**6.2**	**11.4**

Table 2: Percentage of invalid predicted observations

4 Conclusion

We proposed a model for action-based prediction. Our model, compared to its rivals, has a higher capacity for predicting out-of-distribution actions, i.e. actions that are not seen at the training time. Therefore is can handle extreme scenarios better. An immediate direction for extending this work is training a policy network on top of the learned prediction model that can generate actions based on the observations. Since the prediction can perform well under applying extreme actions, we expect the policy network to produce safer and more reliable actions.

References

[1] E. Banijamali, R. Shu, M. Ghavamzadeh, H. Bui, and A. Ghodsi. Robust locally-linear controllable embedding. *arXiv preprint arXiv:1710.05373*, 2017.

[2] M. Bansal, A. Krizhevsky, and A. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018.

[3] A. Bucchi, C. Sangiorgi, and V. Vignali. Traffic psychology and driver behavior. *Procedia-social and behavioral sciences*, 53:972–979, 2012.

[4] S. Casas, W. Luo, and R. Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *Conference on Robot Learning*, pages 947–956, 2018.

[5] 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. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2090–2096. IEEE, 2019.

[6] N. Deo and M. M. Trivedi. Convolutional social pooling for vehicle trajectory prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1468–1476, 2018.

[7] M. Henaff, A. Canziani, and Y. LeCun. Model-predictive policy learning with uncertainty regularization for driving in dense traffic. *arXiv preprint arXiv:1901.02705*, 2019.

[8] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[9] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 336–345, 2017.

[10] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[11] Y. Ma, X. Zhu, S. Zhang, R. Yang, W. Wang, and D. Manocha. Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6120–6127, 2019.

[12] N. Mohajerin and M. Rohani. Multi-step prediction of occupancy grid maps with recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10600–10608, 2019.

[13] R. Shu, H. H. Bui, and M. Ghavamzadeh. Bottleneck conditional density estimation. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3164–3172. JMLR. org, 2017.

[14] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pages 3483–3491, 2015.

[15] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in neural information processing systems*, pages 2746–2754, 2015.

[16] R. S. Yichuan Charlie Tang. Multiple futures prediction. *arXiv preprint arXiv:1911.00997*, 2015.

[17] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun. End-to-end interpretable neural motion planner. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8660–8669, 2019.