BEV2BEV Prediction with VQ-Transformer

Fatih Erdoğan [*1] Umut Zengin [*1]

Abstract

This project aims to predict future Bird's Eye View (BEV) representation of the traffic scene in autonomous driving using a VQVAE-Transformer model. Inspired from (Yan et al., 2021), we built an action conditioned model. By processing current BEV data and ego actions provided by the CARLA simulation, we target to explore the performance of our model for such a prediction task. The project is targeted to unfold in two phases: initially focusing on predicting the ego-centered road layout, and then expanding to include the prediction of other objects' movements in the environment. In this work, we present our study and the findings on the former phase. If successful, this model could help in action planning for autonomous driving. Our project code is publicly available on GitHub (Our baseline: https://github.com/F4718/COMP447-Baseline.git, VQVAE and VQ-Transformer Models: https://github.com/F4718/COMP447-VQ-Transformer.git)

1. Introduction

Autonomous driving evolved around imagery from camera and sensory data. Planning, which is to determine the most appropriate path or trajectory to a given destination is a crucial task in autonomous driving. There exist numerous works targeting to solve the planning problem. Some papers discuss both a pipelined approach and an end-to-end approach (Teng et al., 2023). Some calculates the cost of the possible scenarios and actualize the one with the least cost (Chen et al., 2023).

In our project, we will lean on the generation of such scenarios using an action and frame conditioned generative model. We will work on the Bird Eye View (BEV) representation of the scenes, as images and videos gathered from the cameras are very high dimensional and complex to be processed. BEV representation, provides sufficient summary about the road and traffic conditions for driving. To be precise, given the BEV representation of the scene until time t and the action of the ego between $t \rightarrow t + \Delta$, our target is to predict the $BEV_{t+\Delta}$.

The project is interesting because it addresses to an important aspect of the autonomous driving, planning. If the model succeeds in predicting the future environment conditioned on its movements, it would be possible to plan the actions with comparably low costs using BEV representations. As there exist successful models that generate BEV using the camera and sensory data, our model can be useful for exploration of the planning problem in another space.

This project consisting of 2 phases. In the first phase, we will try to predict the ego centered layout of the road given the previous frames together with the action condition. This part requires the rotation and translation of the road as well as the prediction of the unseen parts of the road based on the previous ones without the environmental objects. In the second part, we will try to predict the movements of the objects such as other cars as well, which is often an unpredictable behaviour. Therefore, we decided to split the objective into 2 phases where the second includes more uncertainty. Due to the time constraints, we couldn't finalize the second phase of the project. Therefore, in this work, we share our study and the findings regarding the first phase of the project.

2. Related Work

As of our knowledge, there exist no study focusing on action-frame conditioned BEV generation. There exist several studies (Liu et al., 2023; Chen et al., 2022; Franceschi et al., 2020; Zou et al., 2023; Yang et al., 2024) on the extraction of BEV representation of the scenes using imagery and sensory data. In our project, we will not deal with this aspect. We will operate under the assumption that these BEV representations are already provided, and our main task will be predicting the future ego centered BEV representations.

(Denton & Fergus, 2018) propose a stochastic video prediction model family named SVG without action conditioning. The target of this model is to generate video continuations based on the provided frames. In their research (Franceschi

[*]Equal contribution [1]Department of Computer Engineering. Correspondence to: Fatih Erdoğan <ferdogan20@ku.edu.tr>, Umut Zengin <uzengin19@ku.edu.tr>.

COMP547 Deep Unsupervised Learning, Spring 2022.

et al., 2020) propose another LSTM based model and compare their results with similar models including the SVG.

With slight modifications on the SVG model, (Villegas et al., 2019) focuses on action conditioned video prediction models. (Oh et al., 2015) propose two novel approaches with RNN's and feed forward networks revealing similar performances. We adapted the former model (Stochastic Video Generation with a Learned Prior) as our baseline model. We shared the model adaptation details and baseline results in Section **??**.

Another work focuses on predicting future instances on BEV perspective using monocular camera inputs with a supervised approach (Hu et al., 2021). Even though this work's target is very similar to our objective, the inputs the training methods are completely different.

There exist also research addressing the planning problem in another perspective by generating possible trajectories for the objects in the scene (Bertugli et al., 2021). Even though they will contribute to the same ultimate goal, our approach differs from that in the prediction target.

Some other work uses transformer based models for video prediction. (Petrovich et al., 2021) utilize Transformer based VAE for action conditioned variable length human motion generation. Similarly, (Yan et al., 2021) propose VideoGPT which is composed of a VQ-VAE and a decoder only transformer to generate both conditional and unconditional videos. Even though we created our model drawing inspiration from VideoGPT, it differs from it in many aspects. You can find the model details and architecture-wise comparison in Section 3.

3. The Approach

Inspired by VideoGPT (Yan et al., 2021), we built a VQVAE-Transformer model. In the original model, authors first train a VQ-VAE model to map the input frames to a discretized latent space. Then, they use a decoder only (GPT-like) transformer model to predict the next sequence latent indices. For frame conditioning, they train a separate 3D ResNet, whose output is to be fed to the transformer for cross-attention. For action conditioning, they use a GAN type approach. They parametrize the gain and bias in the Layer Normalization of the transformer as affine functions of the conditional vector. For generation, they use an auto-regressive approach for the next sequence token indices.

Our model differs from the VideoGPT model in several ways. Firstly, for frame conditioning, instead of training a 3D ResNet and using cross-attention, we encode the input frames via VQ-VAE and after discretization, we feed our encoder based transformer model with these corresponding indices. This approach is beneficial as it mitigates the training and generation cost stemming from the ResNet component, and it allows the model to generate the predictions all-at-once, without auto-regressive nature (for a time window Δ). Secondly, we experimented on an alternative way of action conditioning (other than Layer Normalization parametrization) revealing similar performances, which we found useful for decreasing the training and generation duration.

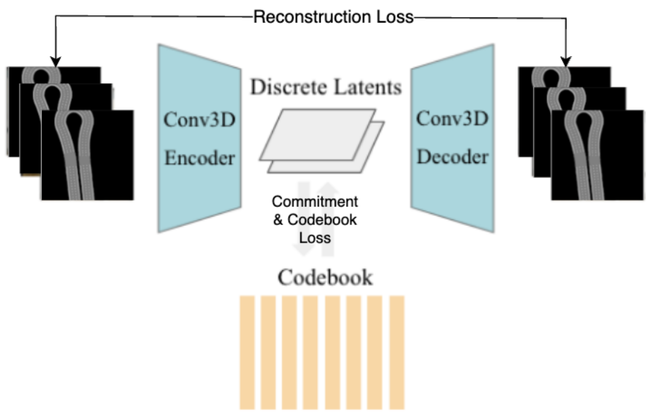


Figure 1. Training process of the 3D VQ-VAE model

In the rest of this section, we will dive into the details of our proposed model.

3.1. VQ-VAE

We used a 3D VQ-VAE model to map the input frames into a discretized latent space and then recover the from it. We stacked a number of convolution layers according to the downsampling ratio (we downsampled to 8×8). In encoder, each convolutional block halves the spatial dimensions of the input, keeping the temporal dimension unchanged. These blokcs are followed by a last convolutional layer, maintaining the dimensionality unchanged. As Residual Blocks, we used Axial Attention Residual Block proposed by (Yan et al., 2021), which gathers spatio-temporal information by applying 3 distinct self attention operations over the time, height and width dimensions. The final output preserves the time dimension of the input frames with targeted spatial dimensions. Then, the outputs of the encoder goes through the codebook, which discretizes the encoder outputs over channel dimension. The final output looks like $C' \times T \times H' \times W'$ where the C' is the codebook dimension, T is the original number of frames to encode, H' W' are the downsampled dimensions. The decoder applies the operations in the encoder in the reverse direction to reconstruct the input frames. You can see the training procedure in Figure 1

We trained the VQ-VAE with the traditional VQ-VAE objective:

$$L_{VQ}(E,G,Z) = \|x - \hat{x}\|^2 + \|sg[E(x)] - z_q\|_2^2 + \beta\|sg[z_q] - E(x)\|_2^2$$

where the first, second, and third terms are reconstruction loss (Mean Square Error), codebook loss, and commitment loss respectively.

We observed that the authors of VideoGPT employed a Moving Average codebook update; however, considering the size of our dataset and the limited diversity, we did not utilize this method.

We ablated on the codebook size and the latent dimensions. We first started with 128 as the codebook size and 128 as the latent dimension. Then, we inspected the codebook index usage and observed a very sparse distribution (see Appendix A). Additionally, considering the simplicity of the data as well, we decreased the codebook size and the dimensionality of the latents. You can see the evaluation of the VQ-VAE models in Table 1.

	S32-D64	**S32-D128**	**S128-D128**
MSE	0.009693	0.011188	0.007988
MIOU	0.949436	0.945192	0.960923
SSIM	0.849225	0.880208	0.889813

Table 1. Reconstruction performances of the VQ-VAE models on input size 128×128 downsampled to 8×8, with time window being 2. MSE is Mean Square Error, MIOU is Mean Intersection over Union, SSIM is Structural Similarity Index. S corresponds to the the codebook size and D corresponds to the latent dimensionality.

As seen in the table, the dimensionality and size reduction don't affect the quality of the reconstructions much. The primary motivation of us behind this ablation is to observe the affect of the VQVAE size on the learning of the transformer component. We discussed the mentioned results in Section 6.

3.2. Transformer

We used an encoder based transformer model as our predictor. After the input is passed through the VQVAE component, the transformer receives the corresponding codebook indices. For the initialization of the transformer embeddings, we tried 3 different approaches. First, we initialized the transformer embeddings with frozen codebook embeddings and used a linear layer to project the codebook size to the transformer hidden size. Secondly, we experimented on the previous approach again without freezing the transformer embeddings. Finally, we initialized the the embeddings independent from the codebook. We observed that the last approach provide the most stable and efficient training, with faster and better convergence. Thus, in the following sections, the shared results are obtained using this approach.

It should be noted that the VQ-VAE model is only able to work with a specific number of frames. Therefore, the T is set according to the number of conditioning frames of the model (in our case 2). As a consequence, the transformer is expected to generate T many frame predictions all-at-once. Therefore, given the conditioning frames between $x_{t+1:t+\Delta}$, and the action conditions for $x_{t+\Delta:t-1+2\Delta}$, the target is to predict the frames $x_{t+\Delta+1:t+2\Delta}$. During training, we used Cross Entropy Loss as our loss function and we froze the VQVAE component during the training of the transformer. Formally, we define the training pipeline as:

$$x = frames_{t+1:t+\Delta} \tag{1}$$
$$gt = frames_{t+\Delta+1:t+2\Delta} \tag{2}$$
$$a = actions_{t+\Delta:t-1+2\Delta} \tag{3}$$
$$h = Codebook(Enc(x)) \tag{4}$$
$$preds_{t+\Delta+1:t+2\Delta} = Transformer(h,a) \tag{5}$$
$$Loss = CrossEntropyLoss(preds, gt) \tag{6}$$

For the action conditioning, we used 2 distinct approaches. First approach is a GAN type approach that we adapted from VideoGPT. This conditioning method parametrizes the gain and bias in the Layer Normalization of the transformer as affine functions of the conditional vector. You can see the formal definition of the mentioned approach in the following equations:

$$c_i = actions_i \tag{7}$$
$$g_i = 1 + \mathbf{W}_g \mathbf{c}_i \tag{8}$$
$$b_i = \mathbf{W}_b \mathbf{c}_i \tag{9}$$
$$\mu_i = \frac{1}{D}\sum_{d=1}^{D} x_{i,d} \tag{10}$$
$$\sigma_i^2 = \frac{1}{D}\sum_{d=1}^{D}(x_{i,d} - \mu_i)^2 \tag{11}$$
$$\hat{x}_{i,d} = \frac{x_{i,d} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} \tag{12}$$
$$y_{i,d} = g_i \hat{x}_{i,d} + b_i \tag{13}$$

Even though the first conditioning method reveals good performance, still it adds an additional computational step, which increases the number of sequential operations. Observing this, we decided to concatenate the action vectors to the frame embeddings. In this way, despite the increase in the number of FLOP's, as the computational burden is embedded into an already existing parallel operation, it helps decreasing the training and generation duration per epoch

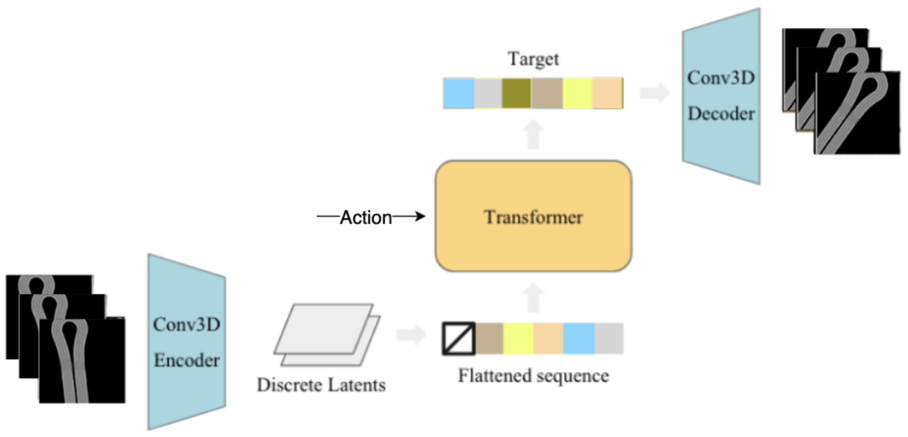


Figure 2. Illustration for the generation pipeline of our proposed model

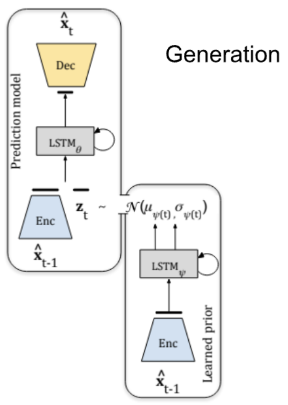


Figure 3. The generation procedure of the SVG-LP model (baseline)

and it doesn't affect the convergence speed in the number of epochs.

Another important problem that we needed address was the generation of the action vectors. Each frame was associated with 8 continuous waypoints, that we discretized during preprocessing (see Section 5.2). To obtain a single action vector out of Δ frames each having 8 different waypoints, we needed to adapt a gathering mechanism. To combine the waypoints, we concatenated the corresponding waypoint embeddings one to other. After this step, we end up having Δ action vectors to combine. To combine the information in a meaningful and consistent manner with variable length frames, we used a light weighted Gated Recurrent Unit (GRU).

You can see the sampling procedure in Figure 2. We shared the results for the mentioned conditioning mechanisms with different VQVAE backbones in Section 6.

4. Baseline

SVG model family is first introduced by (Denton & Fergus, 2018) to predict video continuations conditioning solely on the frames. (Villegas et al., 2019) proposes SVG-Prime by replacing the encoder-decoder component of the SVG with shallow VGG networks and modifies the predictor component to accept action conditioning. An unofficial implementation of this model, together with action conditioned version of the SVG models are available in (Tian, 2023). We used action conditioned version of the Stochastic Video Generation with a Learned Prior (SVG-LP) as our baseline by slightly changing the action conditioning method.

4.1. Model Architecture

SVG-LP is composed of 5 components. An encoder, decoder, prior network, posterior network, and a frame predictor. The encoder being a VGG network, encodes the input frame into a latent dimension (h_t). Then the prior and the posterior which are Convolutional LSTM models, predict the distribution of the frame $t+1$ using h_t and h_{t+1} respectively, and uses the reparametrization trick to generate a vector ($\hat{z}_{t+1}$, z_{t+1}). For a more stable training, z_{t+1} is concatenated to the h_t and fed to the frame predictor which is an LSTM. Therefore, the prior is optimized through $\hat{z}_{t+1}$ using z_{t+1} to approach to the posterior distribution. The latter is only used during training. In addition to the latent vector h_t and z_{t+1}, the frame predictor is also fed with 8 one-hot vectors representing the delta waypoints to be used for action conditioning. The frame predictor concatenates the corresponding embedding vectors to its input and generates $\hat{h}_{t+1}$ which is fed to the decoder to generate the $\hat{x}_{t+1}$. See the following Equations representing the operations for generation.

$$h_t = Encoder(x_t) \tag{14}$$

$$(\hat{\mu}_{t+1}, \hat{\sigma}_{t+1}) = Prior(h_t) \tag{15}$$

$$\hat{z}_t = \hat{\mu}_{t+1} + \hat{\sigma}_{t+1} \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I) \tag{16}$$

$$\hat{h}_{t+1} = FramePredictor(h_t, \hat{z}_t, a_t) \tag{17}$$

$$\hat{x}_{t+1} = Decoder(\hat{h}_{t+1}) \tag{18}$$

The baseline is optimized with Mean Square Error(MSE) as the Reconstruction Loss (RL) and Kullbeck-Leibler Divergence (KLD) between the prior and posterior distribution (see Equation 19).

$$MSE(\hat{x}_{t+1}, x_{t+1}) + \beta KLD(q(z_{t+1} \mid x_{t+1}) \| p(\hat{z}_{t+1} \mid x_t)) \quad (19)$$

5. Experimental Setup

5.1. Dataset

CARLA (Dosovitskiy et al., 2017) is an open-source simulator developed for autonomous driving research. It supports the development, training, and validation of autonomous driving systems. Even though CARLA simulator provides an official evaluation leaderboard, the usage of this platform is time restricted. Therefore, we used a similar dataset created by (Chitta et al., 2023) using "autopilot" on 8 CARLA towns. The dataset is structured in the following folder hierarchy: scenario (13), town (8), routes (*). Each route folder contains detailed road and traffic information including camera images, corresponding depth images, segmentation maps, etc. However, we will only use 2 of them. First one is the bird's-eye view (BEV) segmentation maps containing 3 channels per frame. R channel contains the future locations of the environmental objects, G contains the current environmental objects, and B contains the road layout with lane lines. The second feature we used is the waypoints describing the future action of the ego which is used as an additional conditioning information other than the frames themselves.

5.2. Preprocessing

We handled the preprocessing in several steps. After extracting the frames and the waypoints from the raw data, we first extracted the B channel to work on the road layout. Then, we resized the (500×500) BEV frames to (128×128) as BEV images doesn't require much resolution for interpretability (1.2GB data in total). In this way, we also aimed to decrease the model size. Then we normalized the pixel values between 0 to 1 for the baseline and between -1 to 1 for the VQ-Transformer model for a stable training. Another important preprocessing step is the discretization of the waypoints (actions). Raw waypoints were provided as x-y coordinates on the related frame. We first converted these positional waypoints to delta waypoints by taking the x and y difference separately between each consecutive waypoint (used the ego position to calculate delta values for first waypoint). Then, following (Seff et al., 2023), we uniformly constructed bins between 2 predetermined ($-a$ and a) values. Each waypoint is then mapped to 2 separate bins (a number between 0 and num_bins), 1 for x and 1 for y. Then we combined them by the following Equation.

$$final\_bin\_idx = y\_idx \times num\_x\_bins + x\_idx \quad (20)$$

In this manner, each waypoint is represented with a single index number which will be mapped to the corresponding waypoint embedding in the predictor models. The methods for combining the waypoint information for a frame and across frames are discussed in the Sections 3.2 and 4.1

Finally, we made the train-test split according to the towns. Out of 8 towns (see Section 5.1), we reserved 2 of them solely for testing and the models (including the VQ-VAE) were trained on the remaining 6 towns (size ratio train:test 4:1).

5.3. Evaluation Metrics

After training the baseline and our model in a self-supervised manner, to evaluate our model and compare with the baseline method we used Intersection over Union (IoU, *higher is better*, Equation 21) and Structured Similarity Index Measure (SSIM, *higher is better*, Equation 22).

$$J(\hat{y}, y) = \frac{|\hat{y} \bigcap y|}{|\hat{y} \bigcup y|} \quad (21)$$

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (22)$$

On one hand, IoU is essential for measuring the accuracy of spatial overlap between the predicted and ground truth segmentation, indicating how well the model captures the geometric placement and shape of road layout. On the other hand, SSIM provides a valuable evaluation focusing on aspects like brightness, contrast, and structural integrity. This is important when dealing with predictions that may appear blurry and ignored by the threshold that is applied during IoU calculation.

6. Experimental Results

In this section we describe the hyperparameters of the baseline model and 5 versions of our proposed model and share the performances on the evaluation metrics described in Section 5.3.

6.1. Hyperparameters

The variants of our proposed model differ from each other in 2 ways: whether the action conditioning type is different (concatenation based or Layer Normalization based), or the VQ-VAE backbone is different (with different latent dimension or the codebook size).

We set the number of conditioning frames as 2 and the number of frames to predict as 10. In each of the 5 models, we tried to use similar hyperparameters. As the concatenation based action conditioning increase the hidden dimension of the transformer, we set the frame embeddings size as 128

		Frame IDX									
Models	**Metric**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
SVG-LP (baseline)	**MIOU**	0.8546	0.8251	0.7989	0.7874	0.7765	0.7642	0.7480	0.7304	0.7142	0.6965
	SSIM	0.8312	**0.8241**	**0.8181**	**0.8077**	**0.8057**	**0.8016**	**0.7865**	**0.7799**	**0.7720**	**0.7585**
Concat-S32-D64	**MIOU**	0.9020	0.8735	0.8324	0.8088	0.7839	0.7660	0.7509	0.7367	0.7248	0.7119
	SSIM	0.8156	0.7715	0.7707	0.7341	0.7465	0.7147	0.7341	0.7046	0.7259	0.6969
Concat-S32-D128	**MIOU**	0.8932	0.8684	0.8202	0.8000	0.7678	0.7538	0.7348	0.7241	0.7087	0.6986
	SSIM	0.8459	0.8113	0.8012	0.7726	0.7754	0.7520	0.7629	0.7417	0.7544	0.7337
Concat-S128-D128	**MIOU**	**0.9106**	**0.8850**	**0.8392**	**0.8203**	**0.7893**	**0.7762**	**0.7581**	0.7487	0.7340	0.7244
	SSIM	**0.8512**	0.8184	0.8034	0.7782	0.7772	0.7571	0.7636	0.7461	0.7543	0.7370
Layer-S32-D128	**MIOU**	0.8930	0.8699	0.8227	0.8045	0.7733	0.7604	0.7429	0.7334	0.7191	0.7100
	SSIM	0.8455	0.8116	0.8011	0.7734	0.7769	0.7540	0.7652	0.7449	0.7578	0.7376
Layer-S128-D128	**MIOU**	0.9087	0.8843	0.8376	0.8193	0.7878	0.7756	0.7577	**0.7490**	**0.7351**	**0.7267**
	SSIM	0.8502	0.8183	0.8029	0.7783	0.7768	0.7571	0.7631	0.7465	0.7545	0.7381

Table 2. Table showing MIoU and SSIM metrics for action conditioned SVG-LP (baseline) model and the VQ-Transformer models (ours). ”Concat” and ”Layer” defines the action conditioning type. D defines the dimensionality of the codebook from the backbone VQ-VAE and the S defines the size of the codebook.

and reserved an additional 64 for the action conditioning. While in the Layer Normalization based action conditioning, we set the hidden dimension of the transformer as 192. We used 4 transformer layers in each model, with 3 as the number of attention heads. We used sinusoidal positional encodings. We used 2 layer GRU for combining the actions across frames. We trained each model for 50 epochs and evaluated the version revealing the minimum loss on test set (loss being MSE for baseline, Cross Entropy Loss for transformer models). The best epoch encountered at 13th epoch for the baseline and 7th for the transformer models. We set the batch size as 128 for the transformer models and 29 for the baseline model (limited by the GPU memory).

We trained the models on the GPU’s provided by Google Colab. For training duration per epoch, the baseline model takes 18 minutes on an A100, while our models with Layer Normalization action conditioning and concatenation-based action conditioning take 10 and 6 minutes on an L4, respectively. Therefore, the baseline model needed around 4 hours training on the A100 GPU, while the Layer Normalization and concatenation based action conditioned transformer models required 70 and 42 minutes on L4 GPU.

6.2. Results

You can see the scores of 6 models (including the baseline) on the evaluation metrics MIOU and SSIM in Table 5.3 with respect to the index of the predicted frame. As observed in the table, there exist a consistent decrease in the evaluation metrics as the predicted index increases. This is not much a surprise for the baseline model as it generates the frames one by one. But the VQ-Transformer models suffer from the same problem even inside the group of frames (2 per iteration) that it is predicting at once. Still, the amount of decrease seems to be sharper between different groups (from 2 to 3; 4 to 5 etc.) compared to inside of the group (from 1 to 2; 3 to 4 etc.).

Observing the table, especially for the first 4 frames, the VQ-Transformer models always reveal better performance in MIoU, compared to the baseline model. In the rest of the frames, due to the sharp decreases between groups, occasionally the baseline model surpasses our proposed models. On the other hand, the SSIM metric reveals that the baseline generations preserves the structural integrity better than the VQ-Transformer models. This can be caused by the amplification of a wrongly predicted index in the decoder of the VQ-VAE.

When the VQ-Transformer models are compared among themselves, it can be inferred that the action conditioning types do not affect the performance as the models with same VQ-VAE backbones on different action conditioning types reveals nearly the same results. When it comes to the affect of the VQ-VAE backbone, even though there isn’t a specific pattern, the increase of the codebook size seems to slightly help the performance increase in MIoU. Additionally, observing the concatenation based models SSIM scores, it can be said that the increase in the latent dimension definitely help increasing the structural integrity.

You can see the example samples from the baseline model and from our VQ-Transformer model in Figures 4 and 5.

6.3. Limitations

Although our proposed model marginally surpasses the baseline in MIoU metric, the visual quality of the generated samples are slightly worse than the baseline. Additionally the proposed model requires the future actions as well which

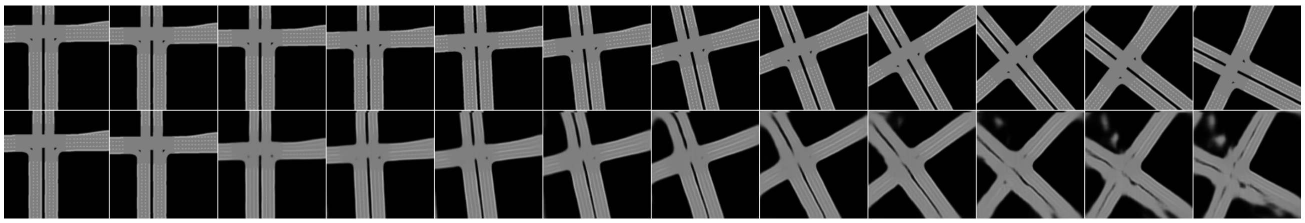


Figure 4. Samples from baseline model. First row shows the ground truth. The second row, first 2 frames are ground truth conditioning frames, the rest predictions.

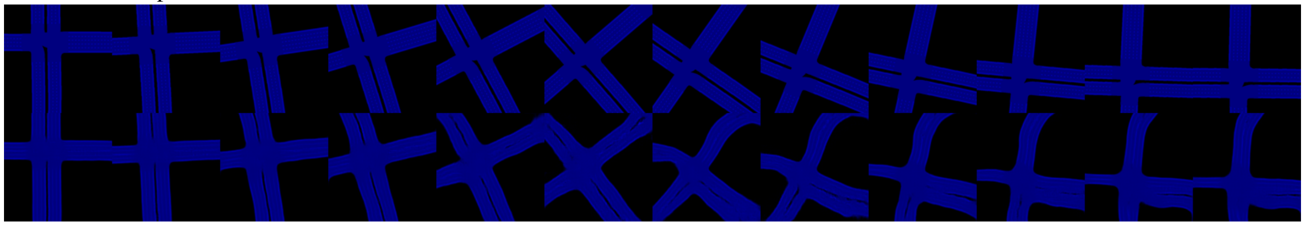


Figure 5. Samples from VQ-Transformer. First row shows the ground truth. The second row, first 2 frames are the reconstructed versions of the ground truth, the rest predictions.

seems slightly counter intuitive. Additionally, the VQ-VAE component might act as a limiting factor for the predictor as the final reconstructions are obtained through the VQ-VAE's decoder. Without a large dataset containing similar maps to the ones in the test set, the reconstructions might fail miserably. Finally, one might argue that the discretization of the input frames increase the complexity of the original task as a small mistake can be amplified by a wrong index prediction. Therefore, without having a very large dataset, our approach might not demonstrate it's full potential.

7. Conclusions

In this study we explored performances of different models on action conditioned BEV2BEV prediction of the ego-centered road layout. We observed that, even a simple LSTM based baseline model can generate satisfying results. Still, we came up with a new VQ-Transformer model and experimented with VQ-VAE's of different sizes. Additionally, we studied on different conditioning methods which revealed similar performances but very distinct training and generations speeds.

We have shown that, our proposed model can reach or even outperform the baseline model in capturing the spatial overlap. Still, in terms of visual quality it falls behind the baseline model. This is probably an indication of the task being harder in the latent space. Overall, this study provides valuable insights about the strengths and weaknesses of different kinds of models. These insights useful to obtain before working on a harder task, the second phase of the project, including the prediction of the environmental objects as well.

8. Future Work

There exist a variety of future directions to expand this project. Firstly, event though we experimented on 2 different action conditioning methods, the frame conditioning methods such as the combination of 3D ResNet encoder and cross-attention used in VideoGPT are still there to be explored. Additionally, due to the time constraints, we were only able to get results for 2 frame conditioned models on predicting the following 10 frames. The impact of increasing the number of frames and the generation quality on more than 10 frames can be investigated. Finally, the second phase of the project, involving the prediction of the environmental objects involving more stochasticity can be explored.

9. Acknowledgements and Contributions

The authors would like to thank Aykut Erdem and Fatma Güney for their expert consultation during the project.

Fatih Erdoğan handled the adaptation of the baseline model to the BEV2BEV prediction task, constructed the Transformer component of the proposed model, adapted concatenation based action conditioning, created the wrapper models, wrote the training functions, did code review for the peer, and conducted the half of the experiments on Google Colab.

Umut Zengin handled the data preparation and preprocessing, wrote the evaluation metrics for the models and adapted them, wrote the VQ-VAE model, wrote the sampler functions, wrote the evaluation functions, made code reviews for the peer, and conducted the half of the experiments on

Google Colab.

References

Bertugli, A., Calderara, S., Coscia, P., Ballan, L., and Cucchiara, R. Ac-vrnn: Attentive conditional-vrnn for multi-future trajectory prediction. *Computer Vision and Image Understanding*, 210:103245, September 2021. ISSN 1077-3142. doi: 10.1016/j.cviu.2021.103245. URL http://dx.doi.org/10.1016/j.cviu.2021.103245.

Chen, S., Cheng, T., Wang, X., Meng, W., Zhang, Q., and Liu, W. Efficient and robust 2d-to-bev representation learning via geometry-guided kernel transformer, 2022.

Chen, Y., Karkus, P., Ivanovic, B., Weng, X., and Pavone, M. Tree-structured policy planning with learned behavior models, 2023.

Chitta, K., Prakash, A., Jaeger, B., Yu, Z., Renz, K., and Geiger, A. Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. *Pattern Analysis and Machine Intelligence (PAMI)*, 2023.

Denton, R. and Fergus, R. Stochastic video generation with a learned prior, 2018.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. Carla: An open urban driving simulator, 2017.

Franceschi, J.-Y., Delasalles, E., Chen, M., Lamprier, S., and Gallinari, P. Stochastic latent residual video prediction, 2020.

Hu, A., Murez, Z., Mohan, N., Dudas, S., Hawke, J., Badrinarayanan, V., Cipolla, R., and Kendall, A. Fiery: Future instance prediction in bird's-eye view from surround monocular cameras, 2021.

Liu, C., Zhou, L., Huang, Y., and Knoll, A. Yolo-bev: Generating bird's-eye view in the same way as 2d object detection, 2023.

Oh, J., Guo, X., Lee, H., Lewis, R., and Singh, S. Action-conditional video prediction using deep networks in atari games, 2015.

Petrovich, M., Black, M. J., and Varol, G. Action-conditioned 3d human motion synthesis with transformer vae, 2021.

Seff, A., Cera, B., Chen, D., Ng, M., Zhou, A., Nayakanti, N., Refaat, K. S., Al-Rfou, R., and Sapp, B. Motionlm: Multi-agent motion forecasting as language modeling, 2023.

Teng, S., Hu, X., Deng, P., Li, B., Li, Y., Ai, Y., Yang, D., Li, L., Xuanyuan, Z., Zhu, F., and Chen, L. Motion planning for autonomous driving: The state of the art and future perspectives. *IEEE Transactions on Intelligent Vehicles*, 8(6):3692–3711, 2023. doi: 10.1109/TIV.2023.3274536.

Tian, S. SVG-Prime: A GitHub Repository for SVG Prime, 2023. URL https://github.com/s-tian/svg-prime. Accessed: 2023-05-04.

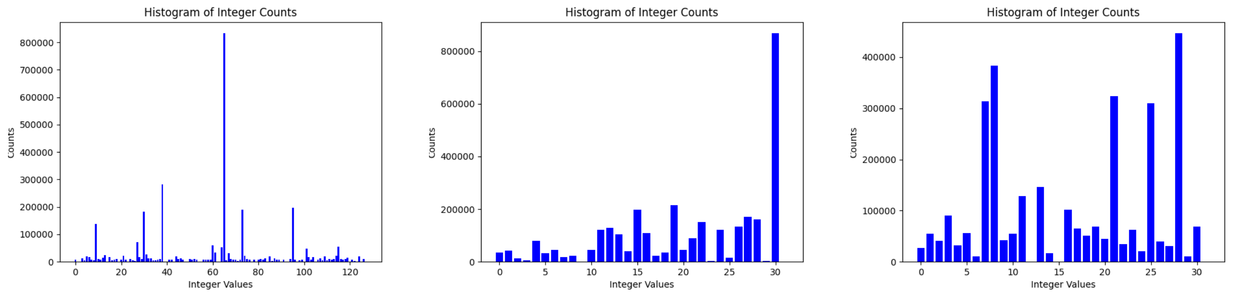
Villegas, R., Pathak, A., Kannan, H., Erhan, D., Le, Q. V., and Lee, H. High fidelity video prediction with large stochastic recurrent neural networks, 2019.

Yan, W., Zhang, Y., Abbeel, P., and Srinivas, A. Videogpt: Video generation using vq-vae and transformers, 2021.

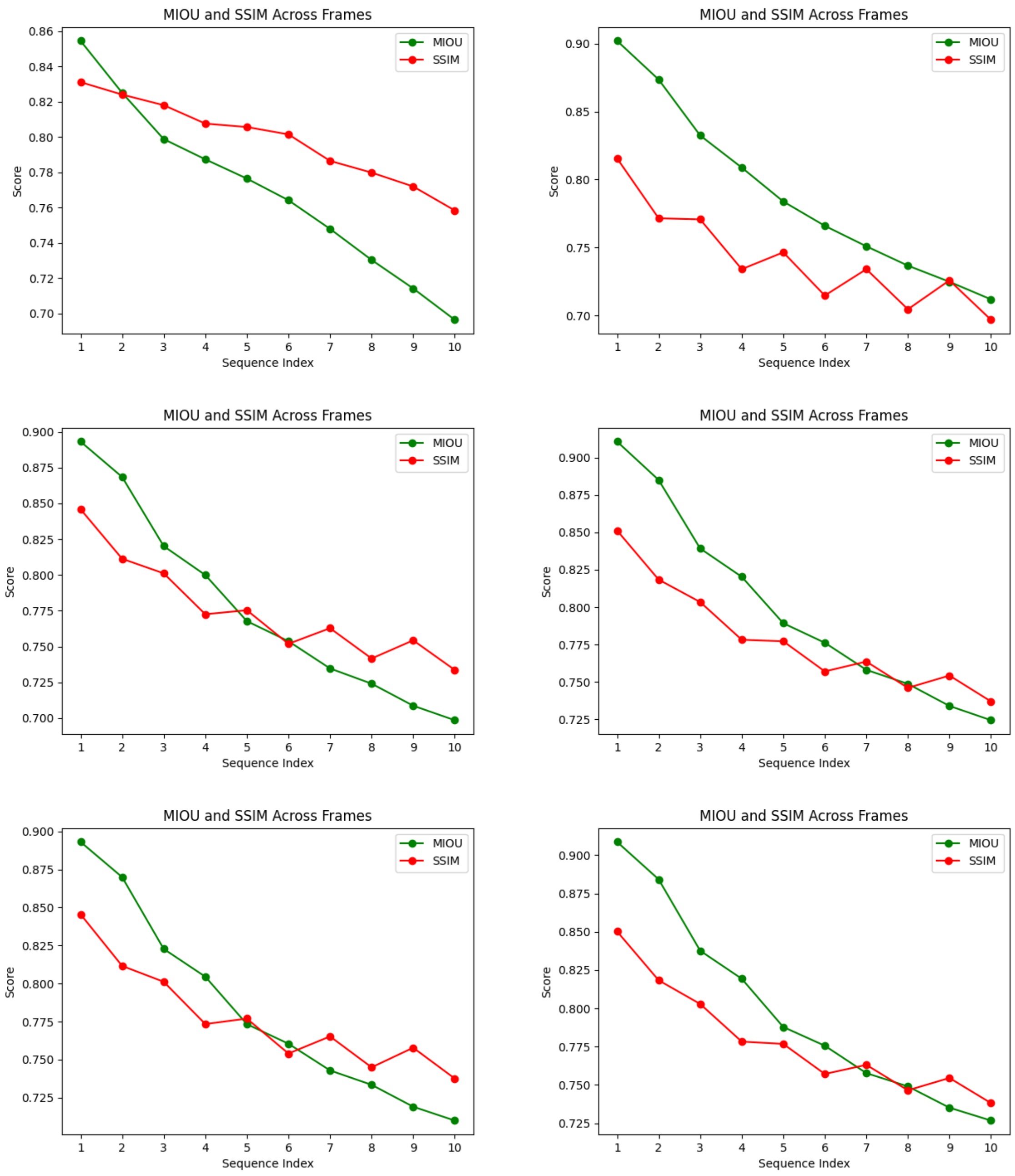
Yang, C., Lin, T., Huang, L., and Crowley, E. J. Widthformer: Toward efficient transformer-based bev view transformation, 2024.

Zou, J., Zhu, Z., Ye, Y., and Wang, X. Diffbev: Conditional diffusion model for bird's eye view perception, 2023.

Appendix



Appendix A. Comparison of index usage across different configurations.



Appendix B. Comparison of MIOU and SSIM Loss across different models, respectively left to right starting from top left. (Baseline , Layer Small, Layer Big, Concat Small, Concat Mid, Concat Big)