Neural Rendering for Robotic Locomotion

Aritra Ray
Electrical and Computer Engineering
Duke University
Durham, USA
aritra.ray@duke.edu

***Abstract*—Neural rendering techniques [1]–[4] synthesizes novel views of complex scenes through optimization of an underlying continuous volumetric scene function using a sparse set of inputs. However, owing to the resource intensive constraint for synthesizing novel scenes, translating neural rendering to robotic locomotion is greatly hindered. Our work introduces faster neural rendering through optimization of hyper-parameters, and offline neural radiance field training on predicted future in-sequence frames based on the current Field Of View (FOV) of the deployed robot. The hyper-parameter optimization on neural radiance fields (NeRF) [1] is carried out through an MLOps platform [5] wherein we monitor the system metrics to estimate the hardware accelerator resource requirements in the Cloud. The in-sequence frame predictor borrows NVIDIA's Pix2PixHD [6] model and is further simplified to suit our application. The quality of the predicted scenes are evaluated using videohash [7]. Offline novel scenes are synthesized on the predicted scenes, and rendered back from the Cloud in real-time to the robot if the similarity score of the predicted scene is above the threshold to the robot's relocated FOV. We estimate the quality of predicted scenes on moving-MNIST [8] benchmark dataset and real-world videos. The implementations can be found in our GitHub repository [9].**

***Index Terms*—Neural rendering, In-sequence frame prediction, Robotic locomotion**

I. Introduction

Neural rendering involves synthesizing novel views of complex scenes through optimization of an underlying continuous volumetric scene function using a sparse set of input. However, generating photorealistic outputs from novel viewpoints requires efficient handling of complex geometry and material reflectance properties which makes it very resource intensive. NeRF [1] requires about two days to optimize for a single scene. This major limitation hinders the applicability of neural rendering engines in robotic locomotive ecosystem.

Based on the deployed robot's present FOV, we harness NVIDIA's Pix2PixHD [6] model to help predict future in-sequence video frames. We hypothesize that the predicted in-sequence video frames would be a close match of the robot's FOV when it physically relocates in the environment. This is based on the assumption that there remains a scene-continuity in all physical environments which can be leveraged to predict next in-sequence video frames. The strength of the hypothesis depends partly on the complexity of the environment the robot is deployed in, the robot's actions, and the quality of in-sequence video frame prediction. Subsequently, we synthesize novel scenes on the predicted frames. Given our hypothesis holds true, when the robot relocates to a new FOV, the neural rendering can be performed in real-time. The similarity score between the predicted in-sequence video and the robot's scenes when it physically relocates in the environment is obtained by videohash [7], a tool designed to detect near-similar videos. Our **key contribution** revolves around advocating offline synthesis of novel views on predicted next in-sequence FOV of the deployed robot, and rendering them in real-time if the similarity score of the predicted scene is above the threshold for the robot's relocated FOV. Our work is organized as follows. Section II introduces the related work, followed by our proposed method in Section III. Section IV presents our evaluation results while Section V concludes the paper.

II. Related Works

Our related work is broadly categorized into neural radiance fields for synthesizing novel views, and predicting next in-sequence video frames from given video data. We discuss both of them in some detail in the following sub-sections.

A. Neural Radiance Fields for Synthesizing Novel Views

NeRF [1] was the first to introduce the synthesis novel complex scenes via representing the static scene as a continuous 5D function of spatial location and viewing direction, to output a view-independent volume density and view-dependent RGB color of every point. It uses a differentiable volume rendering function that can help optimize the scene representation via minimization of the residual between the synthesized and ground truth images. Building on NeRF, HyperNeRF [2] introduces a higher dimensional representation for topologically varying neural radiance fields. Essentially, it represents the 5D radiance field corresponding to each individual input image as a slice through a proposed hyper-space. A family of shapes are modelled in the higher-dimensional space which thereby helps to generate realistic renderings and more accurate geometric reconstructions compared to the state of the art approaches. Nerfies [11] built on top of NeRF incorporates an additional continuous volumetric deformation field to help warp each observed point into a canonical 5D NeRF. Through an elastic regularization of the deformation field, the proposed technique is able to reconstruct novel scenes from videos captured on handheld devices.

The **major limitation** of neural radiance fields for synthesizing novel views remains to be the resource intensiveness of the methods (for e.g., NeRF requires training on a single NVIDIA v100 GPU for 2 days to converge), making a smooth

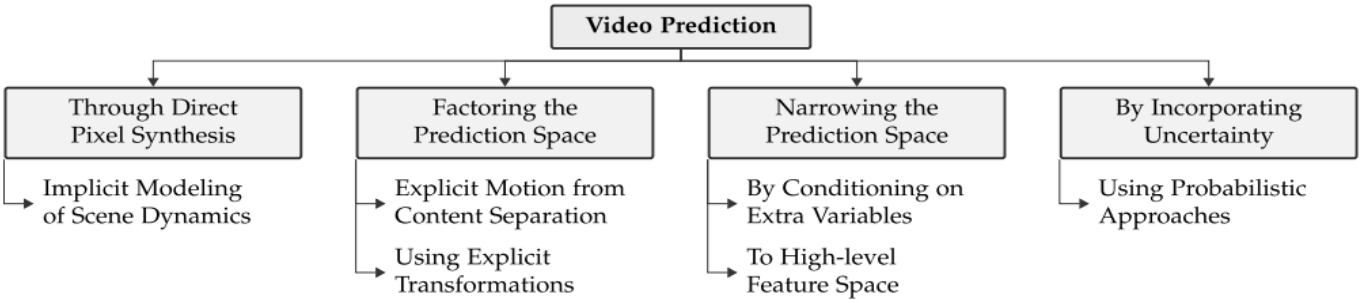

Fig. 1. Overview of deep learning based video prediction methodologies [10]. In our work, we use direct pixel synthesis technique for next in-sequence video prediction.

transition to robotic locomotive ecosystem challenging. To address the limitation, we propose to simplify the model by doing away with hierarchical sampling, performing a hyper-parameter optimization search to optimize them, and largely advocate proposing offline NeRF training on in-sequence predicted video frames.

B. Next In-sequence Video Prediction

Video prediction aims to predict the next in-sequence video frames to assist intelligent-decision making into the future [10]. The proposed techniques delves into direct pixel synthesis [12], [13] by implicit modeling of scene dynamics, factoring the prediction space through explicit motion from content separation [14] or by using explicit transformations [15], narrowing the prediction space by conditioning additional variables [16] or mapping to high-level feature space [17], and by incorporating uncertainty using probabilistic approaches [18]. Fig. 1 illustrates the broad categories of video prediction based on deep learning techniques. In our work, we opt for direct pixel to pixel synthesis through simplifying NVIDIA's Pix2PixHD [6] generator architecture.

III. Proposed Method

This section introduces our proposed method of translating NeRF as a neural rendering engine to robotic locomotion. The deployed robot captures the video, v_i, of the immediate environment and sends the video to the cloud for NeRF training. For simplicity of explanation, we consider $i = 0$, indicating the inception stage. This implies absence of any prior views of the environment in which the robot is deployed in the Cloud. Based on the captured video by the robot at i^{th} time instance, v_i, *online* optimized NeRF training of the scene is performed in the Cloud. Thereafter the rendered video is sent back to the robot, vr_i. Consequently, given the environmental scene, v_i, at i^{th} time instance, the *next scene predictor* module attempts to predict the next in-sequence video frame, $p \longrightarrow v_{i+1}$, where the robot would locate next based on the available data, v_i. Neural rendering is performed based on the prediction results, $p(v_i)$, of the next in-sequence video frame, denoted by $(vr)_{i+1}$. We call this *offline* NeRF tarining. In the $(i+1)^{th}$ time instance, when the robot physically relocates to a new coordinate position in its surroundings, it captures its immediate environment, v_{i+1}, and sends the video to the Cloud. Video similarity module compares captured video, v_{i+1} with the results from the next in-sequence scene predictor, $p(v_i)$, in the Cloud. If $p(v_i)$ matches above a certain threshold, τ, then $(vr)_{i+1}$ is rendered back in real-time to the robot as it was already pre-trained. If $p(v)_i$ is below the certain threshold, τ, then $(vr)_{i+1}$ is rendered in online mode. The cycle keeps repeating for every video uploaded by the robot. Fig. 2 further illustrates our proposed method. The following subsections delves more into our proposed method.

A. Optimizing NeRF Training

To optimize NeRF [1], we do away with the hierarchical sampling. This sampling strategy is designed to allocate the Multi-Layer Perceptron's (MLP) capacity towards space with visible scene content and to reduce the number of queries required to adequately sample the high-frequency scene representation. We reproduce the neural rendering in without hierarchical sampling and analyze the results.

Next, we aim to optimize the neural rendering over learning rates, the embedding size, number of layers, size of dense layer, and activation functions. Additionally, as neural rendering is resource intensive, being aware of the hardware accelerator requirements is essential for cloud deployment. Therefore, we also monitor the system metrics while training the neural rendering model.

B. Next Scene In-Sequence Video Prediction

Given the scene the robot is located at in the environment, v_i, at i^{th} time instance, the *next scene predictor* module attempts to predict the next in-sequence video frame, $p \longrightarrow v_{i+1}$, where the robot would locate next based on the available data, v_i. To train the model, we depend on NVIDIA's pix2pixHD architecture [6]. Pix2pixHD is designed to ingest a lot of information about the input image like a label map that identifies where are the pedestrians, the cars, and several others. We simplify our implementation to use a RGB image an input and predict an RGB frame as the next frame. Doing so however deteriorates contextual information received by the generator. This method however does work in simple cases as demonstrated by our examples [9]. We first create the dataset structure from v_i by extracting the frames into train

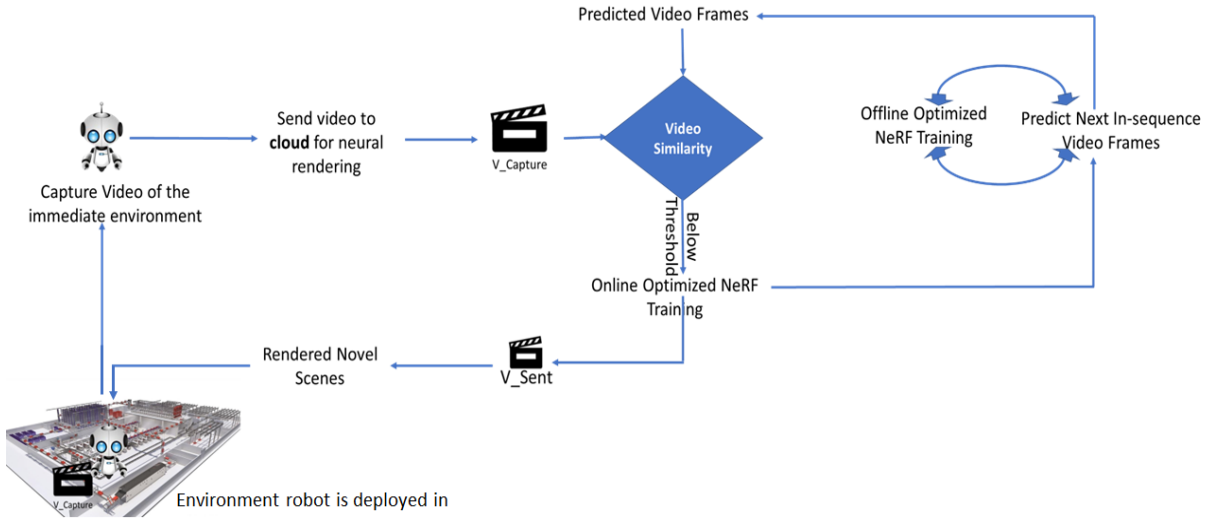

Fig. 2. Our proposed method: Deployed robot captures the video, v_i, and sends to Cloud for neural rendering. Given the environmental scene, v_i, at i^{th} time instance, the *next scene predictor* module attempts to predict the next in-sequence video frame, $p \longrightarrow v_{i+1}$, where the robot would locate next based on the available data, v_i. Neural rendering is performed based on the prediction results, $p(v_i)$, of the next in-sequence video frame, denoted by $(vr)_{i+1}$. We call this *offline* NeRF training. In the $(i+1)^{th}$ time instance, when the robot physically relocates to a new coordinate position in its surroundings, it captures its immediate environment, v_{i+1}, and sends the video to the Cloud. Video similarity module compares captured video, v_{i+1} with the results from the next in-sequence scene predictor, $p(v_i)$, in the Cloud. If $p(v_i)$ matches above a certain threshold, τ, then $(vr)_{i+1}$ is rendered back in real-time to the robot.

and test frames. Then we set up several hyperparameters for experimentation as detailed out in our implementation [9]

C. Video similarity

The *next scene predictor* module attempts to predict the next in-sequence video frame, $p \longrightarrow v_{i+1}$, where the robot would locate next based on the available data, v_i. In the $(i+1)^{th}$ time instance, when the robot physically relocates to a new coordinate position in its surroundings, it captures its immediate environment, v_{i+1}, and sends the video to the Cloud. The predicted in-sequence video frame is to be compared with the captured video. If the similarity score is above a threshold, we can perform real-time neural rendering of novel complex scenes as NeRF training is performed in offline mode for the predicted video frames.

To detect near-duplicate videos, we opt for videohash [7], a library that takes any input video and generates a 64-bit equivalent hash value. Video hash values for identical or near-duplicate videos are the same or similar. The hash value is immune to the videos being upscaled or downscaled, transcoded, color variations, frame rate variation, changes in aspect ratio, and cropped portions.

IV. Evaluations

A. Optimizing NeRF Training

At first, we implemented a simplied version of NeRF wherein we refrain from hierarchical sampling and ignore 5D input including view directions. The implementation results are detailed out in our GitHub repository [9]. We train the model for 15000 epochs on A100-SXM4 GPU resources in Google Colab and the results, monitoring the peak signal to noise ratio (PSNR) are presented in Fig 7.

Next, we perform an extensive hyperparameter search to study the effect of learning rates, the embedding size, number of layers, number of neurons, and activation functions. We do the same using Weights and Biases [5], a MLOps platform that helps us visualize the effect of different hyperparameters so as to obtain the best optimized design. As presented in Fig. 4, we observe the best training for size of dense layer to be 256, embedding size of 2, number of hidden layers in MLP as 8, learning rate of 0.0007 with adam optimizer. We train our model for only 15000 epochs, whereas the NeRF was trained over two days for several hundred thousand epochs. So there could be minor variations in the results.

Owing to the high GPU requirements, we also monitor several system metrics so as to have insights into the hardware accelerator requirements. The hyperparameter search is implemented on Google Colab Pro platform (paid version to access more resources). We use A100-SXM4 GPU resources, on a high-RAM runtime environment. We monitor metrics of GPU power usage (in W), GPU power usage percent, GPU Memory allocated percent, GPU time spent accessing memoy percent, GPU temperature (in C), GPU utilization percent, network traffic (in bytes), disk utilization percent, process CPU threads used, process memory availability (non-swap), process memory in use (non-swap) percent, and process memory in use (non-swap) (in MB). Fig. 5 shows a snapshot of 10 random runs. Detailed results are available through the MLOps platform [5] (interactive visualizations are made public, link available in our Github repository [9]).

B. Next Scene Sequence Prediction

At first, we implement a convolutional LSTM to evaluate next in-sequence prediction on moving MNIST dataset [8].

TABLE I
MODEL ARCHITECTURE OF CONVOLUTION LSTM

Layer (type)	Output Shape	Number of Parameters
Input Layer	[(None, None, 64, 64, 1)]	0
Convolutional LSTM 2D	(None, None, 64, 64, 64)	416256
Batch Normalization	(None, None, 64, 64, 64)	256
Convolutional LSTM 2D_1	(None, None, 64, 64, 64)	295168
Batch Normalization_1	(None, None, 64, 64, 64)	256
Convolutional LSTM 2D_2	(None, None, 64, 64, 64)	33024
Convolutional 3D	(None, None, 64, 64, 1)	1729

The model architecture is detailed out in Table I. Fig. 6 shows the results. Next, simplifying NVIDIA's Pix2PixHD [6] model as described in our proposed section, we predict future in-sequence on real-world videos. A snapshot of the results are illustrated in Fig. 8 with demo (refer presentation slides) available in our GitHub repository [9].

C. Video similarity

We compute the video similarity using videohash [7]. The absolute difference between the ground truth and predicted videos are visualized in Fig. 3. We assume having access to more computing resources would help realize the effectiveness of the proposed metric as difference in predicted frame from 50 epochs of training does not significantly vary from 15 epochs.

V. CONCLUSION

We are successful to demonstrate the translation of NeRF as a neural rendering engine to robotic locomotion through offline training on in-sequence video frame prediction. The overall goal is to minimize the percent of online NeRF training compared to offline ones. In a multi-robot setting, overlapping FOV from multiple robots can play an essential part for improving the next in-sequence frame prediction. Depending on the application scenario, like for instance picking up a item in a complex environment, flow-edge guided video completion techniques can also be used [19] to delete the item picked up from the environment to predict the next frames.

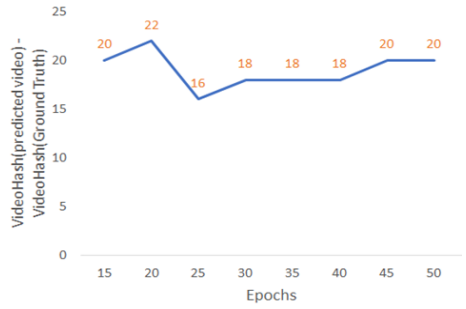

Fig. 3. Quality of predicted video with respect to ground truth. Expected trend line: exponential decay with access to more computing resources.

REFERENCES

[1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *CoRR*, vol. abs/2003.08934, 2020. [Online]. Available: https://arxiv.org/abs/2003.08934

[2] K. Park, U. Sinha, P. Hedman, J. T. Barron, S. Bouaziz, D. B. Goldman, R. Martin-Brualla, and S. M. Seitz, "Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields," *arXiv preprint arXiv:2106.13228*, 2021.

[3] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5855–5864.

[4] A. R. Kosiorek, H. Strathmann, D. Zoran, P. Moreno, R. Schneider, S. Mokrá, and D. J. Rezende, "Nerf-vae: A geometry aware 3d scene generative model," in *International Conference on Machine Learning*. PMLR, 2021, pp. 5742–5752.

[5] L. Biewald, "Experiment tracking with weights and biases," 2020, software available from wandb.com. [Online]. Available: https://www.wandb.com/

[6] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[7] "v 3.0.1," May, 2022, mIT License. [Online]. Available: https://github.com/akamhy/videohash.git

[8] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using lstms," in *International conference on machine learning*. PMLR, 2015, pp. 843–852.

[9] 2022. [Online]. Available: https://github.com/Aritra-14/ME555-Robot-Learning

[10] S. Oprea, P. Martinez-Gonzalez, A. Garcia-Garcia, J. A. Castro-Vargas, S. Orts-Escolano, J. Garcia-Rodriguez, and A. Argyros, "A review on deep learning techniques for video prediction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 6, pp. 2806–2826, 2022.

[11] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla, "Nerfies: Deformable neural radiance fields," *ICCV*, 2021.

[12] W. Lotter, G. Kreiman, and D. D. Cox, "Unsupervised learning of visual structure using predictive generative networks," *CoRR*, vol. abs/1511.06380, 2015. [Online]. Available: http://arxiv.org/abs/1511.06380

[13] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using lstms," *CoRR*, vol. abs/1502.04681, 2015. [Online]. Available: http://arxiv.org/abs/1502.04681

[14] R. Villegas, J. Yang, S. Hong, X. Lin, and H. Lee, "Decomposing motion and content for natural video sequence prediction," *CoRR*, vol. abs/1706.08033, 2017. [Online]. Available: http://arxiv.org/abs/1706.08033

[15] V. Michalski, R. Memisevic, and K. Konda, "Modeling deep temporal dependencies with recurrent grammar cells"","," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/paper/2014/file/cd89fef7ffdd490db800357f47722b20-Paper.pdf

[16] C. Finn, I. Goodfellow, and S. Levine, "Unsupervised learning for physical interaction through video prediction," *Advances in neural information processing systems*, vol. 29, 2016.

[17] X. Jin, X. Li, H. Xiao, X. Shen, Z. Lin, J. Yang, Y. Chen, J. Dong, L. Liu, Z. Jie *et al.*, "Video scene parsing with predictive feature learning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5580–5588.

[18] R. Villegas, A. Pathak, H. Kannan, D. Erhan, Q. V. Le, and H. Lee, "High fidelity video prediction with large stochastic recurrent neural networks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[19] C. Gao, A. Saraf, J.-B. Huang, and J. Kopf, "Flow-edge guided video completion," in *European Conference on Computer Vision*. Springer, 2020, pp. 713–729.

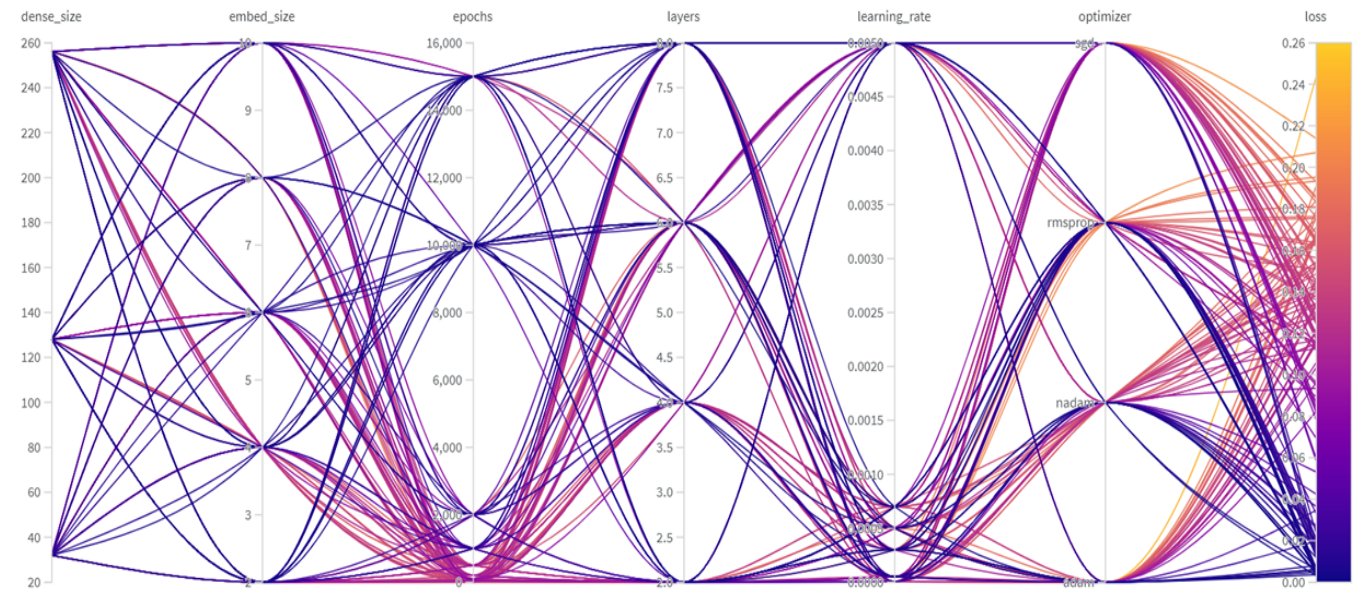

Fig. 4. Hyperparameter search over size of dense layers, embedding size, number of epochs, number of hidden layers of MLP model, learning rate, and optimizer for NeRF [1], using MLOps platform [5]. *Note: Interactive visualization available in our GitHub [9].*

Fig. 5. A snapshot of system metrics from 10 random runs while performing the hyperparameter search to estimate the resource requirements in Cloud for offline NeRF training. *Note: Visualization of all the runs can be accessed through the MLOps platform (link in our GitHub) [9].*

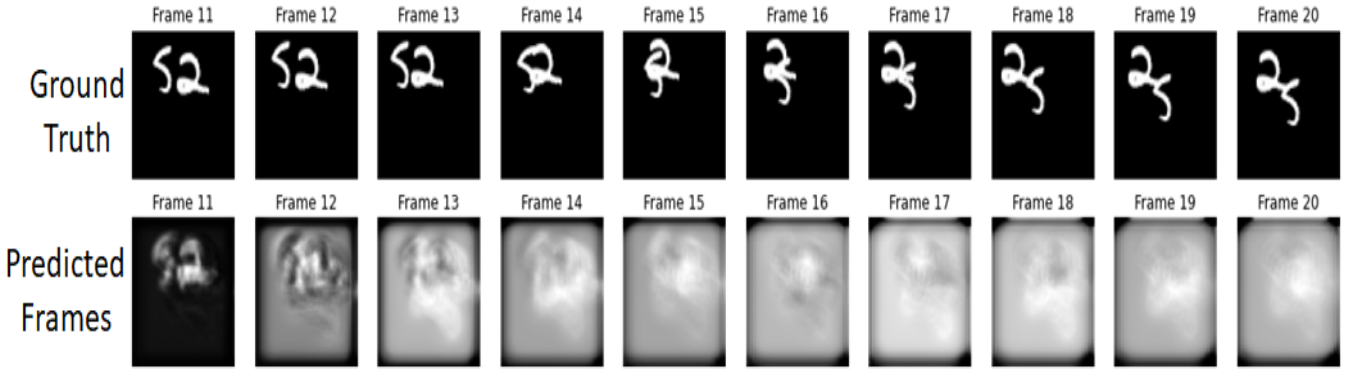

Fig. 6. Visualization of predicted frames with respect to ground truth on moving MNIST [8] dataset using convolutional LSTM. Quality of predicted frames degrades in temporal progression.

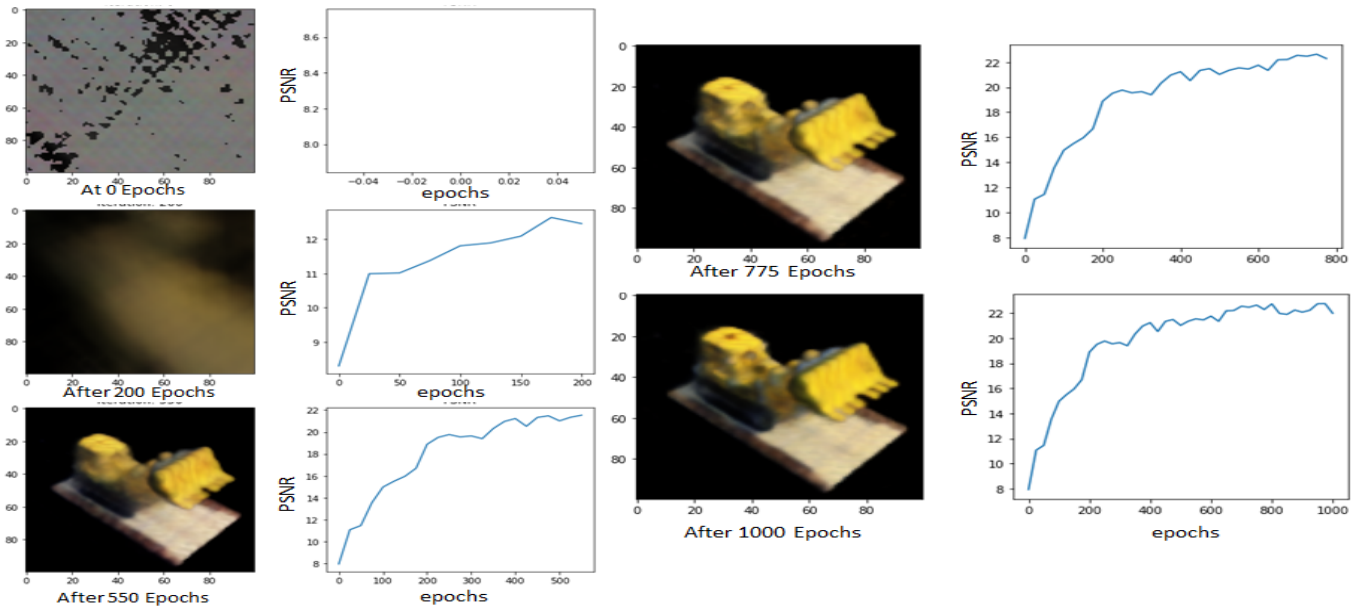

Fig. 7. Visualization of NeRF training progression over peak signal to noise ratio analysis, without hierarchical sampling.

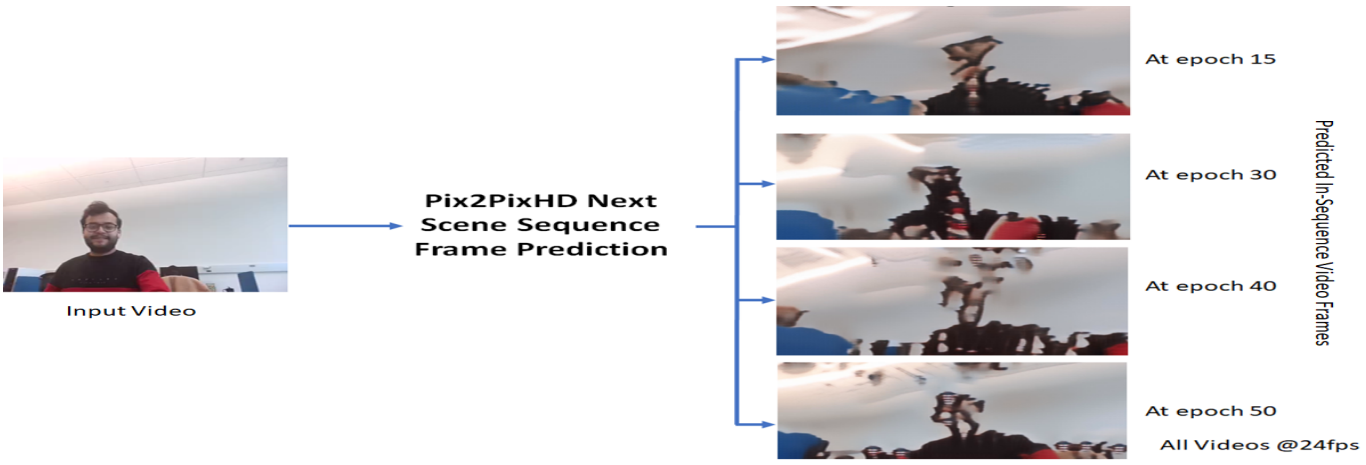

Fig. 8. Visualization of next in-sequence frame generation using a simplified version of Pix2PixHD model [6].Quality of predicted frames degrades in temporal progression. More training epochs corresponds to better similarity score compared to ground truth. *Note: Demo videos are available in GitHub [9] (presentation link).*