Action-Conditioned Robotic Frame Prediction via Vision-Language Diffusion Models

Chentao Xie Yulun Wu Hao Huang Da Ge
The Chinese University of Hong Kong, Shenzhen
{122090895, 122090589, 121090196, 121090138}@link.cuhk.edu.cn

Abstract

In this paper, we propose a new approach for action-conditional robotic frame prediction based on a vision-language diffusion model. We fine-tune the InstructPix2Pix model and train a diffusion model to predict future visual outcomes of robotic manipulation tasks given RGB images and natural language instructions. The framework adopts a conditional generative model that combines visual and textual encodings and employs a denoising diffusion process to generate plausible future frames. We evaluate our approach on tasks such as block tapping, block stacking, and block switching using a dataset collected from the RoboTwin simulation environment. We demonstrate the effectiveness of our model using evaluation metrics such as SSIM and PSNR, which show significant improvements over baseline models such as InstructPix2Pix and DALL·E. The proposed model provides a powerful framework to improve the accuracy and efficiency of robotic action prediction and is expected to be applied in the fields of industrial automation and robotic manipulation.

1. Introduction

Recent advancements in vision-language models have transformed the landscape of image generation, enabling systems to produce highly realistic visuals from textual descriptions. Models such as DALL·E [5] and Imagen [7] have demonstrated remarkable capabilities in zero-shot text-to-image synthesis, while InstructPix2Pix [1] has extended these abilities to image editing guided by natural language instructions. These developments have opened new avenues for robotic vision, where the ability to anticipate visual outcomes based on actions and instructions is critical for tasks requiring precision and adaptability.

In this work, we address the challenge of action-conditioned robotic frame prediction, a task that involves generating the expected visual state of a robotic manipulation scenario given an initial RGB image and a textual instruction describing the intended action. For instance, given an image of a robotic arm and the instruction "hit the block with the hammer," our model predicts the visual outcome 50 frames later. This capability is essential for applications such as industrial automation, where predictive models can enhance operational efficiency, and robotic manipulation, where anticipating outcomes improves task accuracy.

We propose a diffusion-based framework built upon the InstructPix2Pix model, fine-tuned on a custom dataset generated using the RoboTwin simulation environment [4]. Our dataset includes 93,103 samples across three tasks: block stacking, block hammering, and block handover. By conditioning the model on both visual and textual inputs, we enable it to synthesize future frames that align closely with ground-truth observations. Our approach achieves significant performance gains, with evaluation metrics showing up to 0.7767 in SSIM and 16.2329 in PSNR, surpassing baseline models like DALL-E and InstructPix2Pix in task-specific prediction.

This paper is organized as follows: Section 2 reviews related work in vision-language models and their applications in robotics. Section 3 details our methodology, including task definition, data collection, and model architecture. Section 4 presents experimental results and comparative studies, while Section 5 concludes with insights and future directions.

2. Related Work

2.1. Vision-Language Models

Recent advances in vision-language models have greatly enhanced the ability to generate realistic images from text instructions. DALL[5] achieved zero-shot text-to-image generation using a large-scale Transformer trained on paired image-text data. Imagen [7]further improved image fidelity and semantic alignment through a cascaded diffusion model guided by a powerful text encoder. Pix2Seq[2] extended vision-language modeling to structured tasks like object detection, highlighting the potential of autoregressive frameworks. InstructPix2Pix[1] introduced image edit-

ing based on natural language and visual inputs, leveraging instruction-tuned diffusion models. Its capacity for fine-grained semantic modification makes it a strong foundation for action-conditioned robotic frame prediction.

These models shift image generation from an artistic tool to a functional capability in robotic vision, enabling machines to anticipate task outcomes. By extending text- and image-conditioned generation to action-conditioned prediction, robots can synthesize plausible future frames from current observations and high-level instructions—offering practical benefits in applications such as robotic manipulation and industrial automation.

2.2. Vision-Language Models in Robotic Arm Action Prediction

With the rapid development of multimodal models, generating images based on input visual observations and textual instructions has become a mature and widely used task. Combining image generation with future frame prediction provides a promising direction for further exploration. For example, in industrial settings, applying machine learning to predict robot actions can assist in monitoring manufacturing processes, thereby reducing labor costs while improving operational efficiency and safety.[6][3] In addition, visual language models enable robots to interpret high-level human instructions and predict outcomes visually, which is critical for tasks that require accuracy, adaptability, and semantic understanding.

3. Method

3.1. Task Definition

We aim to train an conditional image generation model to predict a future visual observation of a robotic manipulation task. Given an RGB image $x \in \mathbb{R}^{3\times H\times W}$ that captures the robot's current first-person view, and a natural language instruction c that describes an intended action (e.g., "hit the block with the hammer"), the model is required to generate the expected visual outcome y—the image the robot would observe 50 frames later.

We define this task as learning a conditional generative function:

$$G_\theta : (x, c) \mapsto \hat{y}, \tag{1}$$

where $\hat{y}$ approximates the ground-truth future frame y. Model performance is evaluated using Structural Similarity Index (SSIM) and Peak Signal-to-Noise Ratio (PSNR).

3.2. Data Collection and Preprocessing

We collected data using the RoboTwin simulation environment, generating paired visual data and textual instructions for three distinct manipulation tasks: `blocks_stack_easy`, `block_hammer_beat`, and `block_handover`. For each task, we executed simulation scripts to obtain 100 independent episodes, resulting in a total of 300 episodes (100 per task). Raw simulation data were initially stored in `pkl` files organized by task.

We subsequently converted these serialized files into RGB images using a custom extraction script. Specifically, each frame from an episode was converted into a JPG image named following the pattern `episode{i}_frame{j}.jpg`, stored within separate folders corresponding to each task. After conversion, original `pkl` files were discarded to conserve storage space.

Each processed sample in our dataset comprises a triplet: an original image frame at time step i, the edited frame at time step $i + 50$, and the corresponding textual edit prompt describing the robotic action. All images are RGB with a resolution of 320×240. We compiled these image-prompt triplets into structured datasets readable by the Hugging Face datasets library.

Two dataset scales were created: a large-scale set containing 93,103 samples (83,793 for training) and a smaller-scale set containing 22,695 samples (20,426 for training). Both datasets were split into training and testing subsets at a ratio of approximately 9:1.

3.3. Model Architecture

Our architecture fine-tunes the InstructPix2Pix conditional diffusion model, integrating visual and textual conditioning:

- **Visual Encoder (VAE):** Input images are transformed into latent representations using the pretrained Stable Diffusion VAE encoder.
- **Text Encoder (CLIP):** Textual instructions are encoded into semantic embeddings via the pretrained CLIP text encoder.
- **Conditional UNet:** The UNet serves as the core denoising network in our diffusion model. It receives noisy latent and produces edited image latents. Conditioning is achieved through cross-attention layers to generate edits aligned with the given prompts. The UNet is initialized from the pretrained Stable Diffusion checkpoint and fine-tuned on our instruction-labeled dataset.
- **Noise Scheduler:** We adopt the default `DDPM Scheduler` to manage the forward diffusion and reverse denoising process. It governs how Gaussian noise is introduced at each step and how the model iteratively refines latent representations during training and inference.
- **Latent Decoder:** The predicted latent output from the UNet is decoded into RGB pixel space by the pretrained Stable Diffusion VAE decoder, reconstructing the predicted future frame.

3.4. Training Procedure

The optimizer configuration and training hyperparameters for both fine-tuning and training are summarized in Appendix A. Sepecific fine-tuning and training procedures are as following:

3.4.1. Fine-tuning:

We fine-tune the model based on the pretrained InstructPix2Pix model, with a focus on adapting the UNet parameters to our robotic instruction dataset. The text encoder (CLIP) and VAE components are frozen during fine-tuning, and only the UNet weights are updated.

Input images are resized to 256×256 resolution, randomly horizontally flipped for augmentation, and normalized to the range $[-1, 1]$. Textual prompts are tokenized using the pretrained CLIP tokenizer and passed to the frozen text encoder.

The fine-tuning objective follows the standard denoising diffusion formulation. Gaussian noise is added to the latent representation of the target image, and the UNet is trained to predict this noise. We use a mean squared error (MSE) loss between the predicted and true noise:

$$\mathcal{L}_{\text{MSE}} = \mathbb{E}_{z,\epsilon,t}\left[\|\epsilon_\theta(z_t, t, c) - \epsilon\|^2\right],$$

where z_t is the noised latent at timestep t, ϵ is the added noise, and c denotes the text and image conditioning inputs.

3.4.2. Training:

To train the model for instruction-image-editing from scratch, we initialize from the Stable Diffusion v1.5 model. Specifically, we load the Stable Diffusion v1.5 model, and modify the input layer of UNet to accommodate 8-channel inputs.

This modification allows the model to jointly process both instruction and original image condition, aligning it with the InstructPix2Pix task setup. All other components, including the VAE and text encoder, are loaded from Stable Diffusion v1.5 and used with parameters frozen.

The data preprocessing, and MSE denoising loss function remain the same as the fine-tuning configuration.

4. Experiments

4.1. Experimental Setup

We used the RoboTwin [4] platform to generate video trajectories for three target tasks: "beat the block with the hammer", "handover the blocks", and "stack blocks". For each task, we generated 100 trajectories using 100 valid random seeds. By adjusting the `save_freq` parameter, we ensured that each trajectory for the "beat the block with the hammer" task contained approximately 210 frames, while trajectories for the other two tasks consisted of roughly 420 frames each. We then named the extracted frames according to their corresponding task and frame indices. To prepare the dataset for our prediction objective—generating the image 50 frames into the future—we paired frames accordingly (e.g., frame 4 with frame 54). Each image pair was then associated with its corresponding task prompt to form the final dataset for training. In total, we generated 91,303 data pairs, with 82,173 used for training after selecting 90% as the training set. During training, we set `max_train_samples=50000`, selecting 50,000 image pairs to train the model.

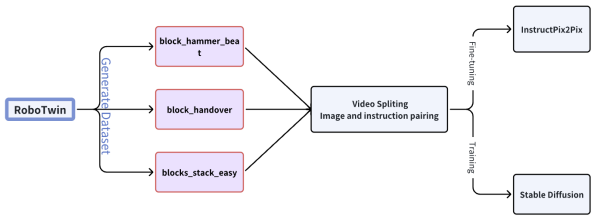


Figure 1. Overall Pipeline of Our Robotic Frame Prediction Framework

4.2. Fine-tuning and Training

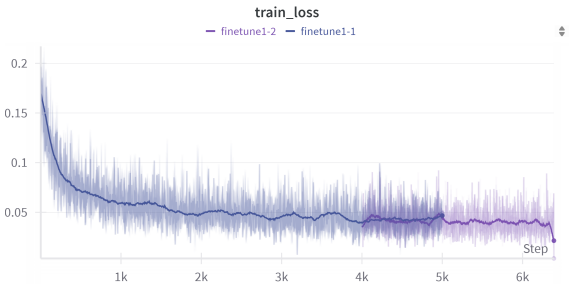
In this project, we conducted two rounds of fine-tuning and two rounds of full model training. We first fine-tuned the InstructPix2Pix model and experimented with different combinations of training parameters. However, we found that the benefits of fine-tuning for our specific task were limited, with the generated images exhibiting poor visual quality. To rule out the impact of hardware or time constraints, we increased the number of training epochs, but observed no significant improvement in either loss convergence or test performance. To further stabilize the training process, we applied smoothing using a running average with a window size of 50 to the model's output during, which is showed by the purple line in Figure2. training. Despite this, the performance still did not improve significantly. Consequently, we proceeded to train a model from scratch using Stable Diffusion. The resulting model demonstrated considerably better performance on the test set compared to the fine-tuned variant.

4.3. Evaluation metrics

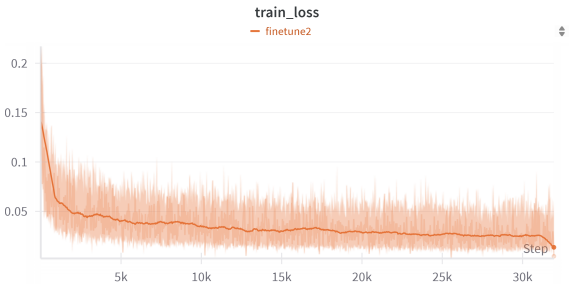
To quantitatively assess the quality of the predicted images, we use two evaluation metrics: Structural Similarity Index Measure (SSIM) and Peak Signal-to-Noise Ratio (PSNR).

SSIM (Structural Similarity Index Measure)

SSIM is a perceptual metric that quantifies image similarity by taking into account changes in structural information, luminance, and contrast. Unlike pixel-wise measures such as



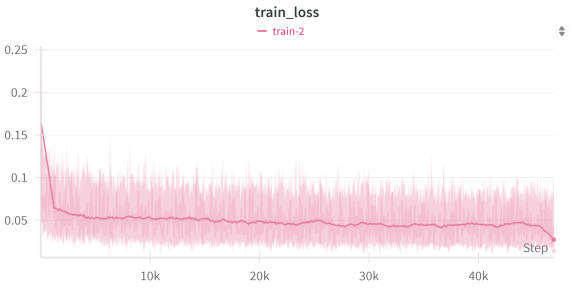
(a) Training loss of Fine-tuning 1



(b) Training loss of Fine-tuning 2

Figure 2. Fine-tuning task loss comparison

(a) Training loss of Training 1

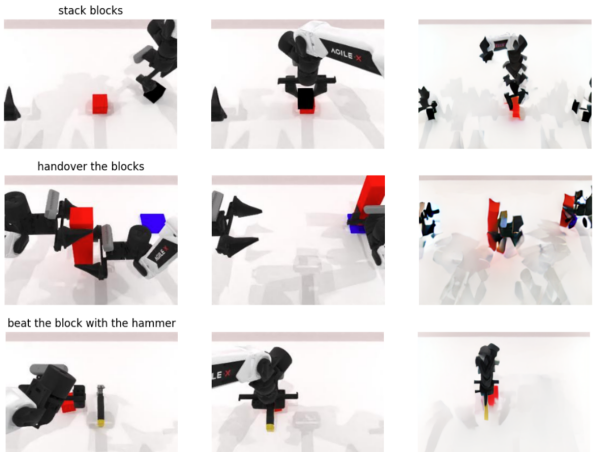


(b) Training loss of Training 2

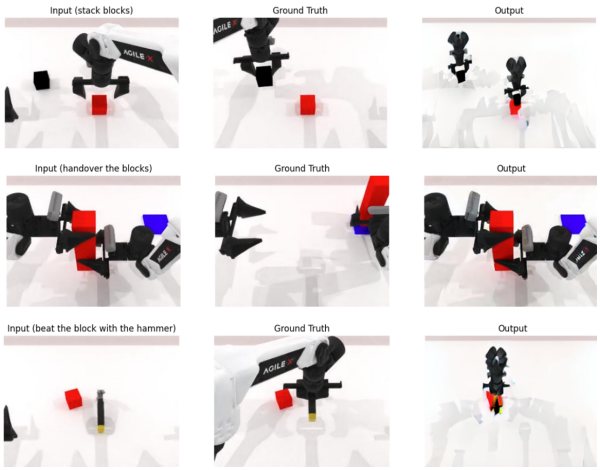
Figure 3. Training task loss comparison

MSE or PSNR, SSIM better reflects human visual perception by comparing local patterns of pixel intensities.

The SSIM between two images x and y is calculated as:



(a) Generated image after Fine-tuning 1



(b) Generated image after Fine-tuning 2

Figure 4. Fine-tuning tasks' results comparison

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

where μ_x and μ_y are the means of x and y, σ_x^2 and σ_y^2 are the variances, σ_{xy} is the covariance, and C_1, C_2 are constants to stabilize the division.

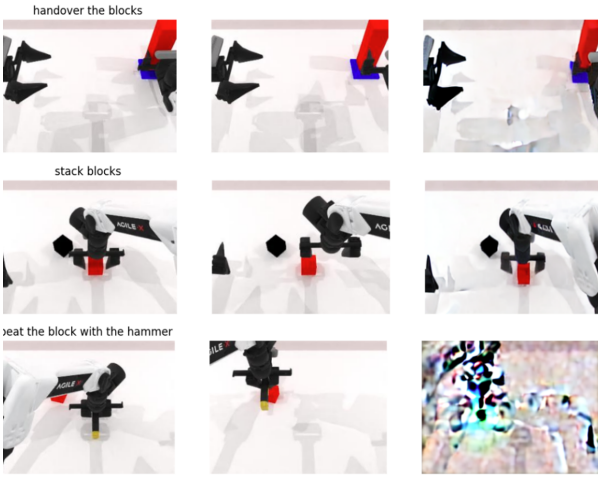
SSIM values range from 0 to 1, with higher values indicating better perceptual similarity.

PSNR (Peak Signal-to-Noise Ratio)

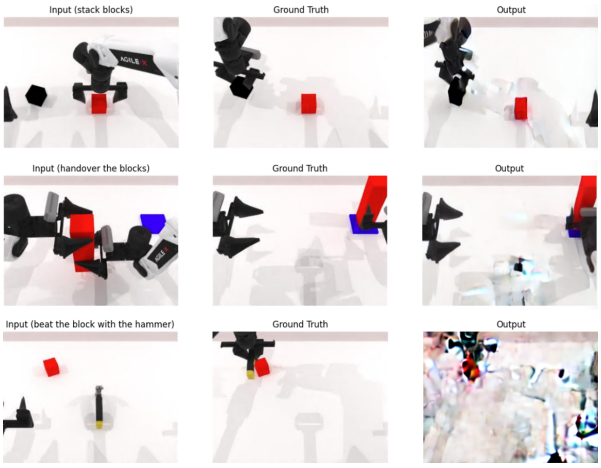
PSNR measures the fidelity of the predicted image compared to the ground truth based on pixel-wise differences. It is defined in terms of the Mean Squared Error (MSE) and is typically expressed in decibels (dB):

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{MAX^2}{\text{MSE}}\right)$$

where MAX is the maximum possible pixel value (e.g.,



(a) Generated image after Training 1



(b) Generated image after Training 2

Figure 5. Training tasks' results comparison

255 for 8-bit images), and MSE is the mean squared error between the predicted and reference images.

Higher PSNR values generally indicate higher reconstruction quality, although PSNR may not always align with human visual perception.

Based on the results, the evaluation metrics demonstrate significant improvements in SSIM and PSNR across various tasks and training approaches. Fine-tuning consistently enhances performance, with notable gains in both image SSIM and PSNR compared to the original models. For instance, the fine-tuned 2-block hammer-beat task achieves an SSIM of 0.6568 and PSNR of 11.8213, while the training 2-handover-blocks task reaches an SSIM of 0.7767 and PSNR of 16.2329, highlighting the effectiveness of iterative training and fine-tuning strategies.

Task	SSIM	PSNR
Original_FT_block_hammer_beat	0.6498	10.4372
Fine-tuning_1_block_hammer_beat	0.6670	11.7406
Fine-tuning_2_block_hammer_beat	0.6568	11.8213
Original_FT_stack_easy	0.5948	9.5843
Fine-tuning_1_stack_easy	0.5856	10.0733
Fine-tuning_2_stack_easy	0.5816	10.0408
Original_FT_handover_blocks	0.5393	8.2371
Fine-tuning_1_handover_blocks	0.5050	9.1933
Fine-tuning_2_handover_blocks	0.4778	8.149
Original_Tr_block_hammer_beat	0.6526	10.3879
Training_1_block_hammer_beat	0.3804	9.9207
Training_2_block_hammer_beat	0.6169	12.9143
Original_Tr_stack_easy	0.5935	9.5801
Training_1_stack_easy	0.7244	13.1432
Training_2_stack_easy	0.7419	14.5552
Original_Tr_handover_blocks	0.5384	8.2207
Training_1_handover_blocks	0.7557	15.2098
Training_2_handover_blocks	0.7767	16.2329

Table 1. Comparison of Fine-tuning and Full Training Results about SSIM and PSNR

4.4. Comparative Study

We conducted a comparative evaluation of our model against InstructPix2Pix and DALL·E using the same prediction tasks. Since these two models were not specifically trained for temporal sequence modeling of robotic arm trajectories, we did not use instructions like "predict the image 50 frames later." Instead, we used alternative prompts such as "generate an image of a robotic arm hitting the red block with a hammer," "generate an image of a robotic arm handing over the block," and "generate an image of a robotic arm stacking blocks." The input images provided to these models were the same as those used in our training process.

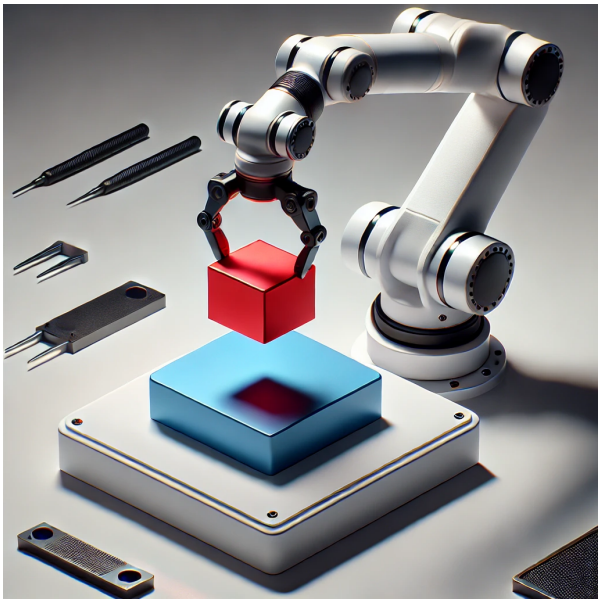
From the images generated by the two models, it can be seen that they only have the ability to use the input image as a reference and generate entirely new images based on instructions. They do not possess the capability to predict the future content of an image based on the input image. This further highlights the significant success of our training, as well as the optimizations and breakthroughs we have made in image prediction.

4.5. Workload Distribution

Chentao Xie handled environment setup and core code implementation, including model fine-tuning and training.

Hao Huang was responsible for RoboTwin data generation and training, as well as experimental design.

Yulun Wu conducted the literature review, contributed to experimental design, and was responsible for writing the



(a) DALL result

(b) InstructPix2Pix result

Figure 6. Results of two model under blocks_stack_easy task

report.

Da Ge assisted in model fine-tuning and was responsible for writing the report.

5. Conclusion and Future Work

5.1. Conclusion

In this study, we proposed an action-conditional robotic frame prediction framework based on vision-language diffusion model. Our approach, which involves fine-tuning the InstructPix2Pix model and training a diffusion model, shows significant improvements in predicting future frames for robotic manipulation tasks. We evaluate our model on three tasks: block tapping, block stacking, and block switching, and show that it outperforms the baseline model in terms of SSIM and PSNR.

However, we observed that the performance of the "hammering" task is slightly lower than the other tasks. This may be due to the small dataset size of this task, which may lead to limited generalization ability of the model. In addition, we also noticed that the setting of the learning rate scheduler may not be optimal, as evidenced by the failure of the training loss to decrease effectively in the late training stage. This issue may affect the overall convergence and performance of the model.

5.2. Future Work

Future improvements will focus on balancing the dataset size of different tasks to ensure better generalization ability, especially for tasks with fewer samples like "hammering". In addition, optimizing the learning rate scheduler is crucial to improve the training dynamics, especially in the late training stage, to ensure that the model converges consistently and effectively. These adjustments will help refine the framework and improve its performance on a wider range of robotic manipulation tasks.

References

[1] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Instructpix2pix: Learning to follow image editing instructions, 2023. 1

[2] Ting Chen, Saurabh Saxena, Lala Li, David J. Fleet, and Geoffrey Hinton. Pix2seq: A language modeling framework for object detection, 2022. 1

[3] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion, 2017. 2

[4] Yao Mu, Tianxing Chen, Shijia Peng, Zanxin Chen, Zeyu Gao, Yude Zou, Lunkai Lin, Zhiqiang Xie, and Ping Luo. Robotwin: Dual-arm robot benchmark with generative digital twins (early version), 2025. 1, 3

[5] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021. 1

[6] Iago Richard Rodrigues, Marrone Dantas, Assis Oliveira Filho, Gibson Barbosa, Daniel Bezerra, Ricardo Souza, Maria Valéria Marquezini, Patricia Takako Endo, Judith Kelner, and Djamel H. Sadok. A framework for robotic arm pose estimation and movement prediction based on deep and extreme learning models, 2022. 2

[7] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding, 2022. 1

Appendix

A. Key Training and Fine-Tuning Parameters

Parameter	**Fine-tuning 1**	**Fine-tuning 2**	**Training 1**	**Training 2**
Pretrained Model Path	instruct-pix2pix	instruct-pix2pix	stable-diffusion-v1-5	stable-diffusion-v1-5
Mixed Precision	fp16	fp16	fp16	fp16
Resolution	128	128	256	256
Train Batch Size	8	8	4	8
Gradient Accumulation Steps	4	4	4	4
Number of Training Epochs	10	50	32	30
Checkpointing Steps	2000	5000	5000	5000
Learning Rate	5e-06	5e-06	5e-05	1e-04
Learning Rate Warmup Steps	100	100	0	0
Seed	42	42	42	42
Validation Image URL	val_image/0.jpg	val_image/0.jpg	val_image/0.jpg	val_image/0.jpg
Max Train Samples	None	None	None	50000

Table 2. Key Training and Fine-Tuning Parameters