# A Study of Visuomotor Behavior Cloning: Performance Considering Real-Time Requirements

Jared Weissberg
Stanford University
jared1@stanford.edu

Vignesh Anand
Stanford University
viganand@stanford.edu

## Abstract

*This paper studies current state-of-the-art algorithms for Visuomotor Behavior Cloning with an emphasis on inference speed and real-time deployment. A common behavior cloning baseline is used to compare a state-of-the-art Diffusion Policy with algorithms based on transformer and CNN architectures. We find that accuracy does not increase linearly with inference time, suggesting that certain architectures like a deep CNN are optimal for tasks that require a balance between inference and accuracy. We also see that fewer denoising iterations in the DDPM lead to no change in performance but significantly faster inference times. Furthermore, the BCT underperformed our deeper CNN architecture; meanwhile, inference time was only marginally faster, suggesting that deeper CNN architectures provide a better performance-inference tradeoff than BCT architectures. We find that changes to noise schedulers (e.g., DDPM versus DDIM) are ineffective at meaningfully reducing inference time. Our primary contribution is an accuracy/inference benchmark for the CNN, transformer, and diffusion architectures: 8-layer CNN, 57-layer CNN, BCT, and DDPM. See Figure 5.*

## 1. Introduction

Imitation learning (IL) is an important approach to the problem of learned autonomy in modern systems. IL is based on the premise that providing a learning algorithm with correct demonstrations is often easier than crafting a well-defined reward function. Behavior cloning (BC) is its simplest form, where the IL problem is set up as a simple supervised learning problem to learn a mapping from agent observations to actions. One of the earliest applications of this approach was in [11], where a simple 3-layer CNN was trained to perform the task of road following. Since then, advancements in the availability of data and the power of models have allowed behavior cloning to be applied to complex problems such as fine bimanual manipulation [5], complex autonomous driving tasks [1], and challenging field robotic navigation [7].

While progress has been made on problems such as data collection/augmentation, model architectures, and learning algorithms, learned policies remain relegated mostly to the simulation and research hardware. One of the key limitations remains the availability of high-performance models that are compact enough to deploy on low-power hardware for real-time inference. Inference time on hardware is often neglected in the field of behavior cloning and imitation learning. This work aims to compare important methods and architectures in behavior cloning. The study's primary aim is to classify BC approaches on the performance versus inference speed spectrum. Possible optimization and improvements to these models are also studied. All approaches are compared to a common block-pushing dataset that is commonly used in robotics.

To benchmark CNN, transformer, and diffusion architectures on the block-pushing dataset, we implemented a naive convolutional neural network (CNN-MLP), ResNet-50 CNN, which is called ResNet-50 and has additional layers, a Behavior Chunking Transformer (BCT), and a Denoising Diffusion Probabilistic Model (DDPM). The DDPM had the highest accuracy, followed by the ResNet-50 CNN, the BCT, and the CNN-MLP. We found that performance did not increase proportionally to inference time. Small improvements in accuracy require significant exponential increases in inference time. Furthermore, we saw that the diffusion model did not perform better with more inference, and 10 denoising iterations were sufficient. The BCT marginally outperformed the 8-layer CNN but underperformed the 57-layer ResNet-50 CNN. Its inference time was almost the same as the ResNet-50 CNN, suggesting that a deeper CNN architecture provides a better performance-inference tradeoff than the transformer architecture. The major findings of this study are presented in Figure 5.

| Dataset | # Traj. | # Verbs | # Scenes | Lang. Instruct. | Cam. Calibration | Public Robot | Collection |
|---|---|---|---|---|---|---|---|
| MIME [42] | 8.3k | 20 | 1 | ✗ | ✗ | ✓ | human teleop |
| RoboTurk [20] | 2.1k | 2 | 1 | ✗ | ✗ | ✓ | human teleop |
| RoboNet [7] | 162k | n/a | 10 | ✗ | ✗ | ✓ | scripted |
| MT-Opt [24, 25] | 800k | 2 | 1 | ✗ | ✗ | ✓ | scripted & learned |
| BridgeData [12] | 7.2k | 4 | 12 | ✓ | ✗ | ✓ | human teleop |
| BC-Z [22] | 26k | 3 | 1 | ✓ | ✗ | ✗ | human teleop |
| RT-1 [2] | 130k | 2 | 2 | ✓ | ✗ | ✗ | human teleop |
| RH20T [13] | 13k[2] | 33 | 7 | ✓ | ✓ | ✓ | human teleop |
| RoboSet [1] | 98.5k | 9 | 11 | ✓ | ✗ | ✓ | 30% human / 70% scripted |
| BridgeData V2 [49] | 60.1k | 82 | 24 | ✓ | ✗ | ✓ | 85% human / 15% scripted |
| DobbE [39]* | 5.6k | 6 | 216 | ✓ | n/a | (✓) | human tool-based |
| Open X-Embodiment [33]† | 1.4M | 217 | 311 | (✓) | ✗ | (✓) | dataset aggregation |

Figure 1: Robot Learning Datasets

## 2. Related Work

### 2.1. Datasets

Several large datasets have been built recently, both in simulation and the real world, for the robot manipulation problem. A summary of these datasets is taken from [8] and presented in Figure 1. Other important datasets for autonomous vehicles include [6], [15], and [9].

### 2.2. Learning Methods and Architectures

The earliest results [11] in behavior cloning involved rudimentary architectures and simple MSE losses at each time step to predict the correct action. Since then, further refinements and novel model architectures have been employed. [4] introduced a class of implicit methods with an Energy-inspired loss to improve performance. Recurrent models have also been widely used for BC. [10] introduced an LSTM-GMM model for the same task. [12] utilized a combination of K-means clustering to cluster possible actions and then utilized a transformer model to classify the correct action class. The current state-of-the-art in behavior cloning is Diffusion Policy, which uses a vision encoder followed by denoising diffusion to generate actions. Diffusion is uniquely well suited to the task of learning from demonstrations as the data are often strongly multimodal. However, the slow inference process involving multiple denoising steps limits real-time use.

### 2.3. Inference Times

While the Behavior Cloning task has been extensively studied with multiple algorithms and architectures being proposed, no prior work has studied the performance versus inference time tradeoffs presented by these models. Inference time on hardware is one of the key drivers of algorithm choice on real systems, and this area remains unstudied.

## 3. Data

This paper uses an adaption of the ubiquitous block-pushing task known as the Push-T benchmark, first used in the Robotics and Embodied AI Lab at Stanford University [2]. It consists of a T-block in a random initial position and a desired end position notated in the image by a green T-shape pictured. The robot is taught to push this block to fit the optimal T indentation in its field of vision. While

Figure 2: Push-T Task Environment

only a 2D task, it features complex multimodal trajectories and contact-rich dynamics. All models were trained on 200 demonstration trajectories, each approximately 300 steps long. At each time step, we receive 2 images each of shape [3, 96, 96] and agent poses. The action space is defined in a 2D positional context, with each dimension ranging from 0 to 512 for both the x and y coordinates. Prior to model training, preprocessing steps were applied to normalize agent positions and actions to the range [-1, 1] along each dimension. Additionally, image channels were normalized to the range [0, 1]. For most models, a 90/10 train/validation split was used.

## 4. Methods

This work studied three classes of BC approaches. (1) **CNN-MLP**: convolution layers followed by multi-layer perception, with MSE loss, (2)(a) **BET**: K-mean clustering of actions, followed by action classification using Behavior Transformer, (2)(b) **BCT**: direct action chunk prediction from observations using transformer encoder, and (3) **DDPM**: ResNet Encoder, followed by 1D denoising diffusion.

### 4.1. Convolutional Neural Network - Multi-Layer Perceptron

A small feedforward convolutional network was trained to establish a performance baseline using an MSE loss. The primary limitation of the CNN approach is its inability to model multimodal distributions (unlike the diffusion model) due to the averaging effect of MLE. The input to this model was two images of shape [Bx2x3x96x96], with the output being of shape [Bx16x2]. We utilize a receding horizon approach, where the agent implements and re-plans the trajectory. The CNN architecture consisted of 8 layers. A 3D

convolutional layer was the initial input, followed by five 2D convolutional layers interspersed with batch normalization and rectified linear unit (ReLU) activation functions. The network concluded with two fully connected layers. The input images, with dimensions [3, 96, 96], underwent an expansion to 256 channels before being flattened and input into the MLP. We trained with the Adam optimizer and a dynamic learning rate reduction strategy starting at a rate of 1e-2 to mitigate loss plateaus. Initial experiments did not reveal significant performance improvement with dropout or weight decay, so our final model used no regularization. To reduce image dimensions without pooling, a stride of 2 was utilized. This model had 1,365,494 parameters.

### 4.2. ResNet-50 Network

We then considered a much deeper CNN based on the ResNet-50 backbone. The structure consists of 57 layers: one 3D convolution layer, one 2D convolutional layer, forty-nine ResNet-50 layers, four additional convolutional layers, and two fully connected layers. We used ImageNet [3] pre-trainined. All layers except the fully connected layer were followed by batch normalization and ReLU. It was trained with a learning rate of 1e-4, the Adam optimizer, no dropout, and a learning scheduler that reduced the learning rate on plateau and was trained for 30 epochs. This model had 34,527,593 parameters.

### 4.3. Transformer Architectures

We performed experiments with two different models featuring a transformer architecture.

### 4.4. Behavior Transformer

Behavior Transformer was first proposed in [12]. It combines a clustering algorithm with a transformer to pick the correct action class. While the original BeT paper uses a low-dimensional state-based output, our implementation uses the 1024-dimensional ResNet-18 vision encoder outputs as the transformer's inputs.

### 4.5. Behaviour Cloning Transformer

We also implemented a self-designed policy based on the transformer encoder architecture from [14]. Early experiments with transformer architectures simply predicting the next action failed due to the covariant shift problem in Behavior Cloning. To mitigate this, we design the BCT, which predicts actions in chunks using the receding horizon, action chunking principle inspired by [5]. This implementation also acted on the outputs of the ResNet-18 encoder. The BCT model had 23,952,736 parameters.

### 4.6. Denoising Diffusion Probabilistic Model

We adopted the state-of-the-art action Denoising Diffusion Probabilistic Model from Diffusion Policy [2]. It consists of a vision encoder and 1D U-Net. The vision encoder is based on ResNet-18 (17 layers) with group normalization instead of batch normalization to ensure compatibility with exponential moving averages. It extracts features from input images and concatenates these features with low-dimensional observations. This serves as the conditioning signal for the noise prediction network. It is then passed through a 1D U-Net.

The 1D U-Net architecture consists of sinusoidal position embeddings for positional encoding of the diffusion iteration (1 layer), downsampling for reducing temporal resolution through strided convolutions (3 layers), and upsampling for increasing temporal resolution using transposed convolutions (3 layers). Each convolutional block consists of one convolution layer, GroupNorm, and Mish activation. There are six blocks in the downsampling path (6 layers), two in the middle blocks (2 layers), and six in the upsampling path (6 layers). A conditional residual block combines two conditional blocks with a residual connection and applies FiLM conditioning.

The entire network (38 layers) employs the Adam optimizer with a learning rate of 1e-4 with cosine scheduling for 100 epochs. A maximum of 100 noising and denoising steps were performed with a DDPM Scheduler with a capped squared cosine beta schedule and clipping to the range of [-1, 1] to improve stability. This model has 91,123,778 parameters: 11,176,512 in the vision encoder and 79,947,266 in the U-Net.

### 4.7. Denoising Diffision Implicit Model

A noted limitation of denoising diffusion is the long inference time necessitated by performing repeated denoising steps. A DDIM Scheduler [13] was studied to accelerate the sampling process in diffusion models via non-Markovian diffusion processes. We ran inference on the initial network with a DDPM scheduler. Again, we used a capped squared cosine beta schedule and clipped outputs to the range of [-1, 1] to improve stability. We also re-trained the full model with a linear beta schedule.

## 5. Experiments

Our primary goal was to understand the performance of the previously described architectures in terms of accuracy and inference time. To do this, we trained and evaluated CNN, ResNet-50 CNN, DDPM, DDIM, BeT, and BcT on the Push-T dataset. Our performance benchmark consisted of 40 versions of the environment, and the average scores and inference time were computed. Experiments were also performed to optimize and improve the baseline Diffusion Policy for the benchmark.

### 5.1. Convolutional Architectures

Our work implements two CNN-based architectures for the BC Task. The compact CNN-MLP model of section 4.1 and the deeper ResNet-50 model of section 4.2. The ResNet-50 model has approximately 25 times the parameters as the compact model. Both models were trained to completion on the demonstration set and tested by our earlier benchmark.

### 5.2. Early Stopping Diffusion

While the original DDPM Diffusion Policy paper [2] utilized 100 denoising steps to generate action trajectories, we studied the impact of generating actions with fewer denoising steps. The benchmark from section 5 was run on 10 versions of the policy using denoising iterations from 10 to 100. The results of performance and time are shown in Figure 3. Performance and inference times are studied.

### 5.3. Efficient Denoising and Sampling

Our work also studied the employment of more efficient denoising techniques at test time to accelerate inference time. Namely, the work studied sampling using the DDIM sampling technique from section 4.7. Models were trained with both the original cosine schedule and a linear schedule, and performance was evaluated at multiple denoising iteration steps.

### 5.4. Experiments with Transformer Architectures

Three different transformer architectures were implemented and trained on the Push-T Dataset. The first is the BeT. While the original BeT operated on low dimensional states, our implementation used the output from the ResNet-18 encoder as input. We then implemented the self-designed Behaviour Chunking Transformer (BCT) on the same dataset. The performance of all models was evaluated on the standard benchmark.

## 6. Evaluation

Our standard performance benchmarks involve 40 different environment initializations. We measure accuracy and inference time. Accuracy is defined by the percentage of the desired space covered by the Push-T block at the expected end state. 100% coverage would mean the block completely covered the desired T-space as expected. We refer to accuracy as the mean total score, the mean of the total coverage scores.

As we run the experiment on 40 seeds, we take the mean value across all 40 seeds for accuracy and inference time. Accuracy is considered to be the mean accuracy across all 40 seeds but also as two additional metrics: percent of scores above 0.5 (50% coverage) and percent of scores above 0.9 (90% coverage). These are important metrics because the mean percent of scores above 0.5 provides insight into the models' ability to generalize and perform decently. The percent of scores above 0.9 is an important metric in understanding how well the model would complete the task in real-world environments where high accuracy is desired. Inference time is discussed as the amount of time it takes to run inference for one timestep. The results are discussed below. All inference was performed on an Nvidia A100 GPU.

## 7. Results

### 7.1. CNN-MLP

Being the most compact model, The CNN-MLP had a relatively quick inference time of 0.0022 seconds per timestep. Results were overall poor, with a mean total score of 0.29, 25% of scores over 0.5, and 2.5% of scores over 0.9. This is our baseline performance for a simple CNN architecture.

### 7.2. ResNet-50 CNN

The ResNet-50 CNN had a mean inference time of 0.013 seconds (76Hz). The best-performing epoch (30) had a mean total score of 0.41, 40% of scores over 0.5, and 32.5% of scores over 0.9.

Overall, the ResNet-50 CNN saw an increase in mean total score from 0.29 to 0.41, scores over 0.5 from 25% to 40%, and scores over 0.9 from 2.5% to 32.5%. The most substantial increase was the increase in scores over 0.9. Deeper, more powerful CNN models are able to significantly improve performance while still running quickly on hardware

### 7.3. Transofrmer Architecture

The clustering plus classification approach of the BeT failed to generalize at test time. When employed on a vision-based clone task, the transformer is unable to learn correct action classes. This resulted in zero successful runs on the benchmark.

Experiments with Transformer architectures that predicted the next action sequence ran into the common problem of covariant shift and were unable to learn the underlying action distribution. The receding horizon approach of the BCT was able to overcome these problems.

The Behavior Chunking Transformer (BCT) mean inference time was 0.0120s (82Hz). This was a similar inference time to the ResNet-50 CNN. The BCT performed slightly worse than the ResNet-50 CNN with a total score of 0.32, 30% of scores above 0.5, and 10% of scores above 0.9. It slightly outperformed the CNN-MLP architecture but had significantly longer inference times.

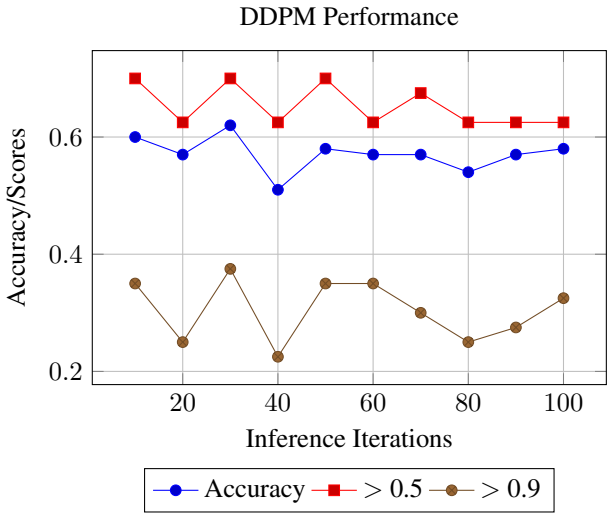


Figure 3: Performance Variation of Diffusion Policy with Denoising Iterations

### 7.4. Denoising Diffusion Policy

The Denoising Diffusion Policy DDPM had a significantly longer inference time compared to the CNN-MLP and ResNet-50 CNN. Even at 10 iterations, the DDPM had an inference time of 0.1165 seconds per timestep compared to 0.0022 for the CNN-MLP. At 100 iterations, the DDPM had a mean inference time of 1.1094 seconds. Overall, mean inference time increased linearly from 10 to 100 iterations. At 10 iterations, the DDPM had a mean inference time of 0.1165 seconds. Accuracy, percent of scores over 0.5, and percent of scores over 0.9 fluctuated across iterations with no clear improvement from 10 to 100. Ultimately, more iterations did not increase accuracy across all metrics. See Figure 3.

### 7.5. Denoising Diffusion with DDIM sampling

The DDIM scheduler with a capped squared cosine and linear beta schedule had a similar mean inference time to the DDPM scheduler. However, there was no noticeable change in the mean accuracy or number of scores above 0.5 or above 0.9. The DDIM scheduler with a linear beta schedule also showed no significant difference in mean accuracy, scores above 0.5, or scores above 0.9 compared to either the DDPM scheduler or the DDIM scheduler with a capped squared cosine beta schedule. See Figure 4. Overall, attempts to optimize the sampling technique failed to yield any improvement. Combined with early stopping, the final optimized policy represented a 10x improvement in inference time without losing any performance.

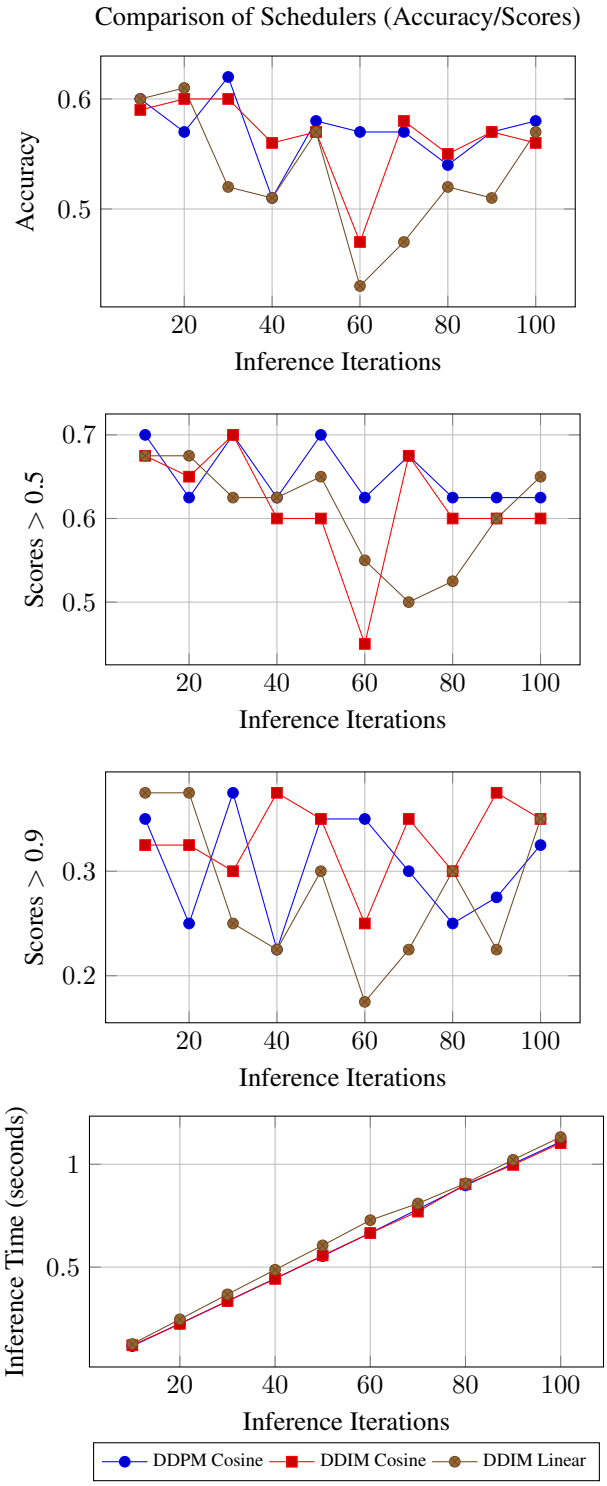


Figure 4: Performance Comparison of DDPM Scheduler with Cosine Beta Schedule, DDIM with Cosine Beta Schedule, and DDIM with Linear Beta Schedule

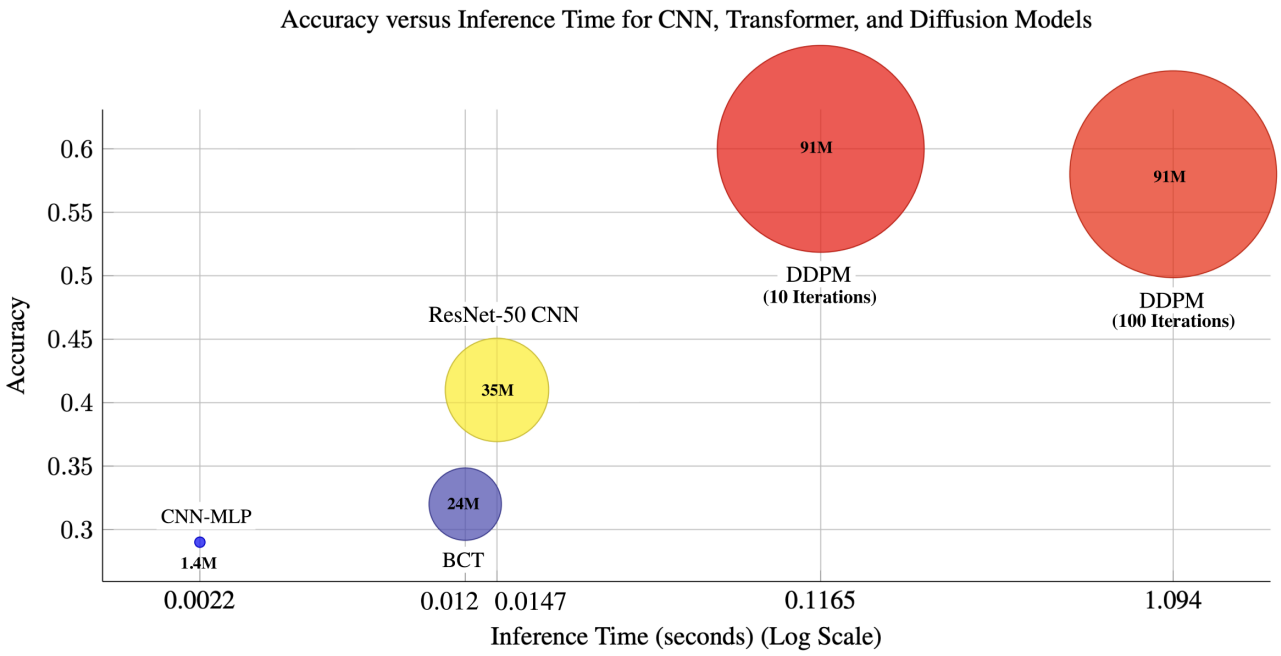


Figure 5: Performance comparison of CNN-MLP, ResNet-50 CNN, BCT, and DDPM. Inference time is computed across 40 seeds (environment initializations) and averaged. Accuracy is defined as the mean percentage of area covered during the Push-T task. The interior numbers refer to the parameter size in millions.

# 8. Conclusion

## 8.1. Key Findings

Our work categorized various popular behavior cloning approaches on performance versus inference speed. In general, as model complexity increased, so did inference time and accuracy. However, the relationship between complexity and performance was not as strong as initially imagined. The 57-layer pre-trained ResNet-50 had an inference time of almost 10x the 8-layer CNN-MLP, but the mean total accuracy only increased from 29% to 35%. There was also a substantial increase in accuracy above 0.9 between the CNN-MLP and ResNet-50 30-epoch CNN, suggesting that the larger model meaningfully improves its ability to make accurate predictions rather than generalize with scores above 50%.

Denoising diffusion policies are state-of-the-art for the behavior cloning task and are the most performant. This comes at a great cost to inference times. We found that the original work over-denoised the action predictions. As we increased the number of iterations (and, in turn, inference time) during the DDPM inference step, accuracy was relatively unchanged. Combined with the optimized DDIM scheduler, this represents a 10x improvement in sampling from the model.

Transformer-based approaches had a marginally better accuracy than the CNN-MLP model, although it was significantly larger with approximately 24M versus approximately 1.4M parameters. Inference time was almost 10x, suggesting that the ResNet-50 CNN offers a better inference-performance tradeoff compared to BCT.

Although previous literature suggests that adjusting the DDPM architecture can result in large changes in inference time with schedulers such as the DDIM Scheduler, we did not find this to be applicable to action diffusion for visuomotor policy. The DDIM Scheduler resulted in no change to the accuracy or inference time. Existing DDPM architectures have significantly longer inference times than CNNs (10x-100x). We found that a decrease in the number of iterations during the denoising process linearly decreased inference time without sacrificing training time.

For a comprehensive understanding of inference time versus accuracy, see Figure 5.

## 8.2. Limitations and Future Work

Because of computational constraints, we evaluated each model using 40 seeds (Push-T initializations) for accuracy. A more comprehensive study might evaluate each model across a larger number of initializations. Furthermore, we only evaluated these architectures on the Push-T task. Fu-

ture studies might evaluate architectures on a broad range of tasks, such as lifting, placing, object transport, tool hang, etc. One might also incorporate data with human interference and perturbation.

The number of models tested and the complexity (with attention to the amount of time spent tuning hyperparameters) of each model can be expanded beyond the 8-layer CNN, 57-layer CNN, BCT, and DDPM models. Finally, we recommend a comprehensive review of existing noise schedulers for DDPMs beyond the DDPM scheduler, DDIM scheduler, and cosine versus linear beta schedules we reviewed in this paper.

## References


[1] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars, 2016.

[2] Cheng Chi, Siyuan Feng, Yilun Du, Zhenhia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[4] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Proceedings of the 5th Annual Conference on Robot Learning*, 2021.

[5] Zipeng Fu, Tony Z. Zhao, and Chelsea Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation, 2024.

[6] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*.

[7] Alessandro Giusti, Jérôme Guzzi, Dan C. Cireşan, Fang-Lin He, Juan P. Rodríguez, Flavio Fontana, Matthias Faessler, Christian Forster, Jürgen Schmidhuber, Gianni Di Caro, Davide Scaramuzza, and Luca M. Gambardella. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1(2):661–667, 2016.

[8] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Donovon Jackson, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. Droid: A large-scale in-the-wild robot manipulation dataset. 2024.

[9] Will Maddern, Geoffrey Pascoe, Matthew Gadd, Dan Barnes, Brian Yeomans, and Paul Newman. Real-time kinematic ground truth for the oxford robotcar dataset. *arXiv preprint arXiv: 2002.10152*, 2020.

[10] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *5th Annual Conference on Robot Learning*, 2021.

[11] Dean A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988.

[12] Nur Muhammad Mahi Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning $k$ modes with one stone. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[13] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[15] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.