# Dita: Scaling Diffusion Transformer for Generalist Vision-Language-Action Policy

Supplementary Material

## A. Model and Training Scheme

The architecture of our model is illustrated in Figure 3. The language instruction is encoded using a pretrained CLIP model, with its encoder frozen during training. Input images are resized to $224 \times 224$ and processed by a pretrained DINOv2 model, with all parameters being finetuned. A Q-Former, trained from scratch with a depth of 4, is employed to reduce the dimensionality of the image features to a length of 32; within each block, text tokens are injected as FiLM conditions to augment the image features with linguistic information. The action is perturbed with noise via a DDPM scheduler with 100 timesteps, and a timestamp index is embedded using a sinusoidal positional embedding module. These multimodal inputs are then fed into a causal Transformer, which predicts the added noise. The Transformer adopts a LLaMA2-style architecture, trained from scratch, comprising 12 self-attention blocks with a hidden size of 768. All components are trained except for the CLIP text encoder. In total, the model comprises 334M parameters, with 221M being trainable. Achieving this level of performance with such a compact model represents a pioneering advancement in the field, underscoring the efficacy of the architectural design.

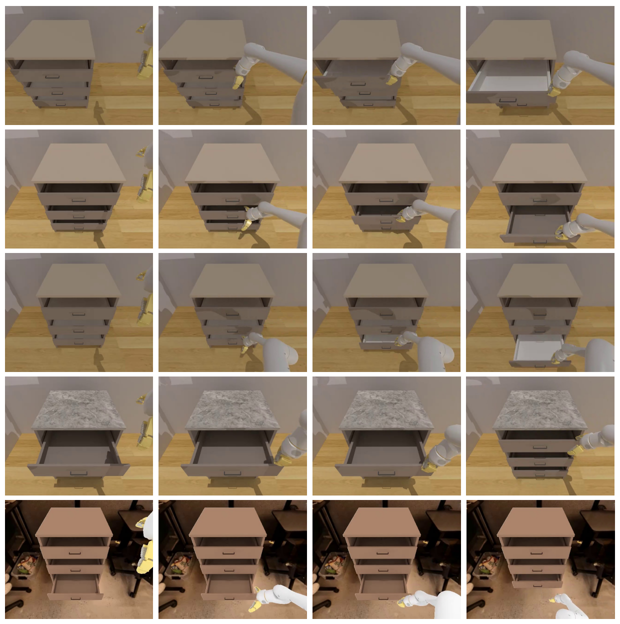
Figure 7. Qualitative results of Dita under variances in Google Robot.

## B. Simulation Benchmarks

### B.1. SimplerEnv

**Results.** As described in Figure 7, leveraging the in-context conditioning design, Dita exhibits enhanced robustness, relying solely on third-person view images to detect subtle nuances and generate more reliable actions.

### B.2. LIBERO

LIBERO comprises four subtasks: LIBERO-SPATIAL, LIBERO-OBJECT, LIBERO-GOAL, and LIBERO-100, each designed to evaluate different model capabilities. LIBERO-SPATIAL assesses spatial relationship understanding, containing data with identical object sets but varying layouts. LIBERO-OBJECT evaluates object transferability, featuring data with consistent layouts but different object sets. LIBERO-GOAL examines task comprehension and transferability, maintaining the same object sets and layouts while varying tasks. LIBERO-100 is further divided into LIBERO-90 and LIBERO-10 (also referred to as LIBERO-LONG), designated for policy pretraining and long-horizon task evaluation, respectively. LIBERO-100 encompasses a diverse range of objects, layouts, and backgrounds, providing a comprehensive benchmark for generalization in robot learning.

**Optimization.** We optimize the network using AdamW for 100,000 steps on LIBERO. The learning rate is set to $1e-4$ for LIBERO-SPATIAL, LIBERO-OBJECT, and LIBERO-GOAL, and $5e-4$ for LIBERO-LONG. Across all sub-datasets, a half-cycle cosine scheduler is applied to decay the learning rate. Denoising timestamps are set to 100 during finetuning, and training is conducted with a batch size of 512 across 8 NVIDIA A100 GPUs.

**Results.** Table 6 shows that Dita achieves a success rate of 77.93% on the most challenging task in LIBERO, *i.e.*, SPATIAL-LONG. We argue that the Droid dataset [31] serves as a more suitable pretraining dataset for LIBERO, as our model (334M) lacks the capacity to fully accommodate the entire OXE dataset. We anticipate that performance on the OXE-pretrained model can be significantly improved by scaling up the model size.

### B.3. CALVIN

**Setup.** We directly apply the proposed method to CALVIN using a single static RGB camera to predict the end-effector action, which includes three dimensions for translation,

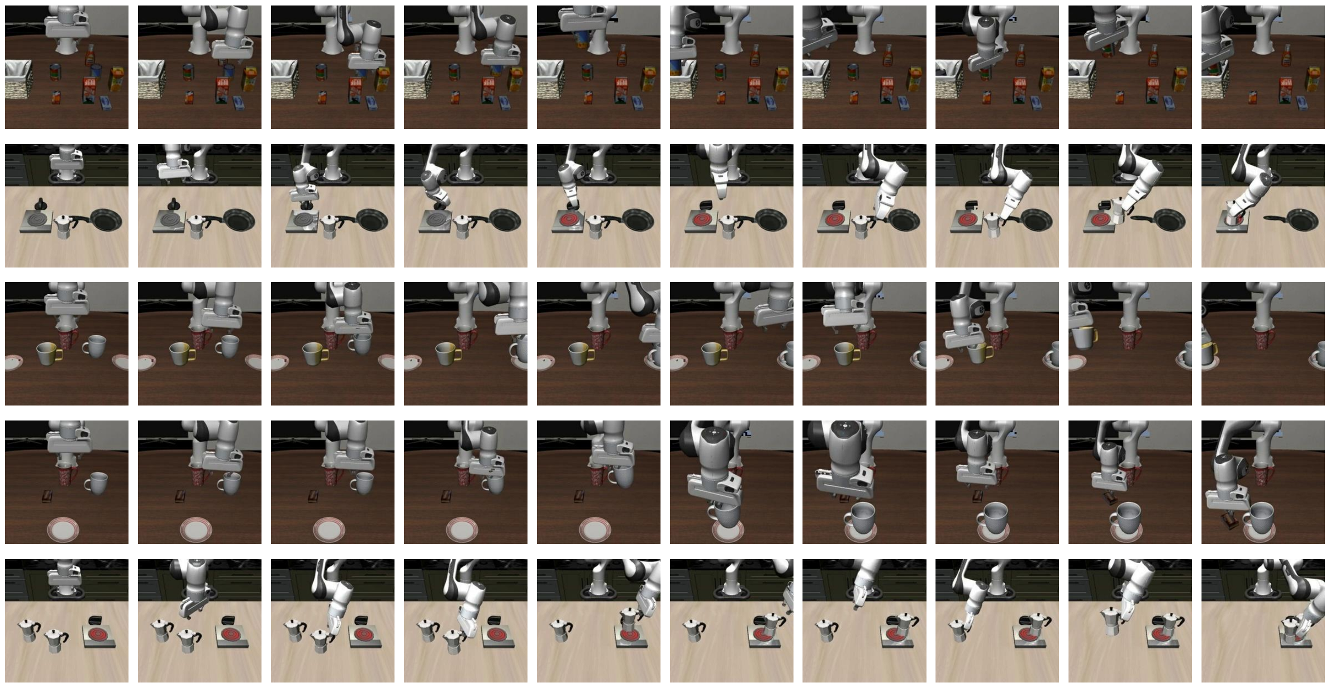

Figure 8. Qualitative results of Dita on LIBERO benchmark.

Table 6. Comparison with Diffusion Policy [17], Octo [72], and OpenVLA [32] on LIBERO [40]. Dita (OXE) denotes the use of a pretrained model on OXE, while Dita (Droid) refers to the use of a pretrained model on Droid.

| Method | LIBERO-LONG |
|---|---|
| Diffusion Policy* [17] | 50.5% |
| Octo [72] | 51.1% |
| OpenVLA [32] | 53.7% |
| Dita (pretrained on OXE) | 63.8% |
| Dita (pretrained on Droid) | **77.9%** |

three dimensions for Euler angle rotation, and one dimension for gripper position (open or close). We evaluate Dita and $\mathcal{E}_S^{Diff}$ on CALVIN, leveraging the pretrained model on the OXE dataset to initialize the model for CALVIN.

**Optimization.** For each training iteration, the model predicts 10 future action chunks supervised by MSE loss. An AdamW optimizer is used together with a decayed learning rate with half-cycle cosine scheduler after several steps of warming up. The learning rate is initialized as $1e-4$. Model is trained for 15 epochs with batch size of 128 across 4 NVIDIA A100 GPUs.

### B.4. Maniskill2

ManiSkill2 [22], the next generation of the SAPIEN ManiSkill benchmark [48], serves as a widely recognized platform for assessing the generalized manipulation capabilities of embodied models. It encompasses 20 distinct manipulation task families and over 4M demonstration frames across various configurations. Leveraging ManiSkill2, we establish a *novel camera view generalization* benchmark to evaluate the effectivenes of Dita.

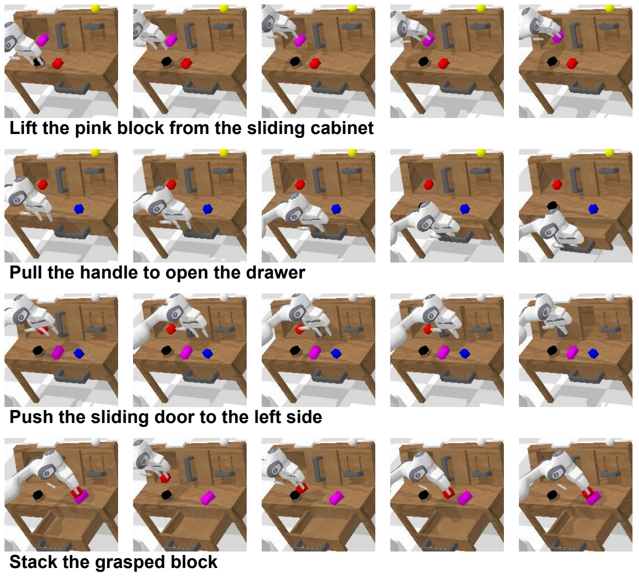
Lift the pink block from the sliding cabinet
Pull the handle to open the drawer
Push the sliding door to the left side
Stack the grasped block

Figure 9. Qualitative results of Dita on CALVIN ABC→D benchmark.

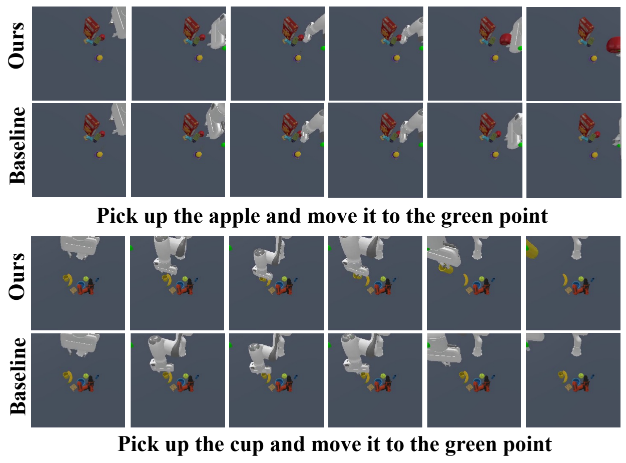
Ours
Baseline
Pick up the apple and move it to the green point
Ours
Baseline
Pick up the cup and move it to the green point

Figure 10. Qualitative comparison between Dita (top) and Diffusion Action Head baseline $\mathcal{E}^{Diff}_{\theta\sim s}$ (bottom) on ManiSkill2 (PickClutterYCB).

**Setup.** To construct the benchmark, we select 5 tasks (PickCube-v0, StackCube-v0, PickSingleYCB-v0,PickClutterYCB-v0, PickSingleEGAD-v0) from ManiSkill2 and create a camera pool comprising 300K random cameras. 20 cameras are sampled each time to render each trajectory, resulting in over 40K trajectories, which are utilized to train Dita from scratch. The generated dataset is divided into training and validation sets with a 19:1 ratio, ensuring that each category in task family, and trajectories rendered from different camera views are assigned to both, thereby preventing data leakage. During training, the number of data samples is balanced across task families by duplicating trajectories for task families with fewer samples. To construct a closed-loop evaluation dataset, we randomly sample 100 trajectories from the validation set for each task family. This evaluation dataset with 500 trajectories is used to assess the success rate for each task family and demonstrate the camera-view generalization capabilities of Dita.

**Optimization.** The network is optimized using AdamW for 50,000 steps on ManiSkill2, with a learning rate set to $1e-4$. The number of denoising timestamps is set to 100, and the batch size is 1024 distributed across 16 NVIDIA A100 GPUs.

## C. Real-Robot Experiments

### C.1. Real-Robot Setup

**Optimization.** We apply image augmentations using ColorJitter from the torchvision library, with brightness set to 0.3, contrast ranging from 0.7 to 1.3, saturation ranging from 0.7 to 1.3, and hue set to 0.07. Further details are provided in the code.

**Variance Robustness.** To evaluate the robustness of Dita, we further validate its performance under different variances, including:

- *Background changes.* The background includes both the tabletop color and the backdrop. We introduce variance in both aspects by using tablecloths in colors different from the tabletop and a black backdrop.
- *Non-target object arrangements.* We randomly place non-target objects in arbitrary poses within the robot's workspace to create a cluttered scene, whereas it remains clean during demonstration recording.
- *Lighting conditions.* We modify the lighting by turning off one of the two lights in the room to introduce variation in illumination.

### C.2. Details of Real-Robot Tasks

In addition to the fundamental tasks used for quantitative comparison with prior approaches, we incorporate complex long-horizon tasks that previous methods fail to complete for illustrative purposes. Below, we present all tasks along with their step-wise decomposition.

- *Pick the banana into the box.* We divided this task into two steps: first, successfully picking up the banana, and second, successfully placing it into the box.
- *Pick the kiwifruit in the box.* We divided this task into two steps: first, successfully picking up the kiwifruit, and second, successfully placing it into the box.
- *Pouring the coffee beans into the bowl.* This task is divided into two steps: first, successfully picking up the cup, and second, successfully pouring the coffee beans within the cup into the box.
- *Pouring the water from the teapot into the cup.* This task is divided into two steps: first, successfully picking up the teapot, and second, successfully pouring the water into the cup.
- *Stacking three bowls.* This task is divided into two steps: Stacking the first bowl successfully, and second stacking the left bowl into previous stacked bowls.
- *Stacking three nesting dolls.* This task is divided into two steps: Stacking the first two small dolls successfully, and second stacking the large doll into previous stacked dolls.
- *Pick the banana and insert into the small pen container.* This task is divided into two steps: first, successfully picking up the banana, and second, successfully inserting the banana into the pen container.
- *Open the Flip-top door box and the pick up the small cube inside.* This task is divided into two steps: first, successfully open the door box, and second, successfully picking up the small cube.
- *Open the drawer.* This task has only one step.
- *Close the drawer.* This task has only one step.
- *Pick up the bowl within the drawer and pouring the coffee beans into the outside bowl.* This is a long horizon task, and we demonstrate it with video in the supplementary appendix given that previous approaches fail to com-

plete the task.

- *Pick up the racket and hit the ball into the goal* This is a long horizon task, and we demonstrate it with video in the supplementary appendix given that previous approaches fail to complete the task.
- *Open the top drawer, then pick the cube into the drawer, and finally close the drawer.* This is a long horizon task, and we demonstrate it with video in the supplementary appendix given that previous approaches fail to complete the task.
- *Close the top drawer, then open the bottom drawer and put the bowl into the drawer, and finally close the drawer.* This is a long horizon task, and we demonstrate it with video in the supplementary appendix given that previous approaches fail to complete the task.
- *Open the box and move the green cube into the box then close the box.* This is a long horizon task, and we demonstrate it with video in the supplementary appendix given that previous approaches fail to complete the task.

### C.3. Finetuing Details

We adhere to [32] and employ LoRA for 10-shot finetuning. However, Dita comprises only 221M trainable parameters, with merely 5% (approximately 11M) remaining trainable under LoRA finetuning. We contend that this limited capacity is inadequate to effectively accommodate image augmentations, thereby compromising robustness against environmental variances. To this end, we evaluate robustness through full finetuning and observe a substantial improvement in the success rate for long-horizon tasks, alongside greater resilience to variances such as background changes, non-target object arrangements, and lighting conditions. Quantitatively, full finetuning achieves a success rate of 20%, whereas LoRA finetuning fails to complete tasks under extreme variances.

## D. Analysis, Ablations, and Discussions

### D.1. Practices on reproducing Octo and OpenVLA

We observe that OpenVLA [32] demonstrates superior pick-up performance compared to Octo [72]. However, for tasks requiring the learning of rotational operations, such as opening a box, Octo achieves better performance. We attribute this to Octo's ability to predict continuous actions, which are less sensitive to action normalization, whereas OpenVLA relies on action discretization based on action statistics. We compute the statistics from the 10-shot training samples across all tasks and find it challenging to obtain suitable statistics for discretization values, which are unnecessary for the diffusion policy.

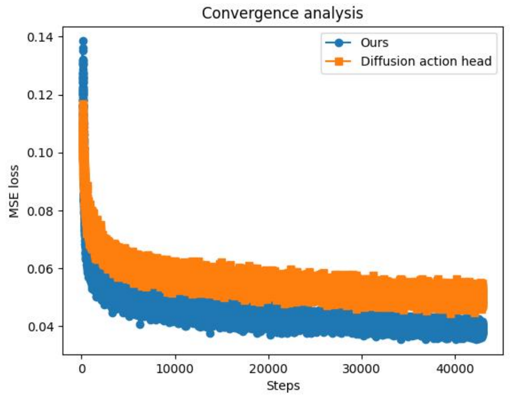

Figure 11. Convergence Analysis on OXE dataset [9]. The blue line is DiT Policy, and the orange line is Diffusion action head strategy with the same number of parameters.

Table 7. Additional experiments on Calvin (ABC→D). 'aug' indicates image augmentation during finetuning.

| Method | No. Instructions in a Row (1000 chains) | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | Avg.Len. |
| Dita w/o aug | 94.5% | 82.5% | 72.8% | 61.3% | 50.0% | 3.61 |
| Dita w/aug | 92.3% | 83.8% | 75.8% | 67.3% | 69.0% | **3.78** |
| w/o pretraining w/ aug | | | | | | |
| Dita (2 layers) | 75.8% | 47.3% | 29.3% | 19.0% | 12.5% | 1.84 |
| Dita (6 layers) | 80.5% | 60.0% | 39.0% | 26.3% | 15.0% | 2.21 |
| Dita (12 layers) | 84.5% | 65.5% | 48.8% | 34.5% | 22.5% | 2.56 |
| Dita (24 layers) | 87.8% | 67.5% | 48.5 % | 34.5 % | 23.0% | 2.61 |
| Dita | 84.5% | 65.5% | 48.8% | 34.5% | 22.5% | 2.56 |
| AdaLN DiT [17] | 68.3% | 40.0% | 21.0% | 11.3% | 5% | 1.45 |

### D.2. Convergence Analysis

Figure 11 illustrates the convergence comparison between the diffusion head baseline $\mathcal{E}_\theta^{Diff}$ and Dita. Dita achieves clearly faster convergence than $\mathcal{E}_\theta^{Diff}$. We believe this further highlights the scalability of Dita.

### D.3. Failure Analysis

Figure 12 illustrates two representative failure cases. The first case involves a failed grasp during execution. Although the model is capable of recovering and retrying, the failure suggests an inability to approach the correct position for a successful pickup. The second case concerns pouring while in motion, where the model tends to spill the contents due to unstable handling.

### D.4. More Ablations

**Learning Rate Scheduler.** As outlined in the main text, we utilize a standard learning rate scheduler to decay the learning rate during the experiments in Calvin, rather than using a fixed learning rate of $1e-4$ as in the pretraining

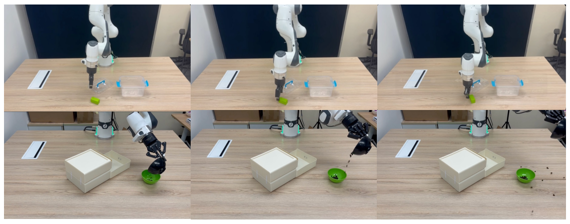

Figure 12. Failure Cases Analysis.

Table 8. The ablation study on the learning rate scheduler in the Calvin benchmark.

| **Strategy** | **No. Instructions in a Row (1000 chains)** | | | | | |
|---|---|---|---|---|---|---|
| w lr decay | **94.5%** | **82.5%** | **72.8%** | **61.3%** | **50.0%** | **3.61** |
| w/o lr decay | 91.8% | 80.0% | 68.0% | 56.9% | 45.9% | 3.43 |

Table 9. More action designs w/o pretraining on Calvin (ABC→D). MDT is from issue 9 of its GitHub repo and GR-MG.

| **Methods** | No. Instructions in a Row (1000 chains) | | | | | |
|---|---|---|---|---|---|---|
| MDT* [61] | 61.7% | 40.6% | 23.8% | 14.7% | 8.7% | 1.54 |
| Unet1D head [17] | 76.8% | 46.5% | 28.8% | 18.5% | 10.0% | 1.80 |
| Transformer head [17] | 75.8% | 44.8% | 26.5% | 16.5% | 8.0% | 1.72 |
| 8-layer MLP head | 69.8% | 42.5% | 26.3% | 16.8% | 11.0% | 1.66 |
| 3-layer MLP head | 75.5% | 44.8% | 25.0% | 15.0% | 7.5% | 1.68 |
| Single token act chunks | 56.5% | 18.3% | 6.0% | 2.8% | 0.8% | 0.84 |
| Ours | **89.5%** | **63.3%** | **39.8%** | **27.3%** | **18.5%** | **2.38** |

Table 10. The effect of the number of execution steps (# Steps) on ManiSkill2.

| **# Steps** | 1 | 2 | 4 | 8 | 16 |
|---|---|---|---|---|---|
| All | **61.6%** | 60.8 % | 60.6 % | 60.0 % | 58.0 % |

stage. This adjustment results in a slight performance improvement, as shown in Table 8.

Table 11. Ablation study of shuffle buffer size on SimplerEnv (both math and variant results of Google Robot [8]).

| **Shuffle Buffer Size** | **coke_can** | |
|---|---|---|
| | match | variant |
| 128000 | 71.2% | 73.6% |
| 256000 | **83.7%** | **85.5%** |

**Diffusion Head.** We implement several policies inspired by the core idea of the diffusion action head, which differ slightly from Octo [72] in predicting action chunks. Specifically, Octo [72] flattens action chunks into a single vector with a unified embedding. For instance, when predicting 8 actions, it generates a $8 \times 7 = 56$-dimensional vector. In contrast to the Octo-style diffusion action head, we adopt a diffusion action head, akin to Diffusion Loss [35] and Diffusion Force [14], which are more effective. We evaluate multiple diffusion heads, including Unet1D, Transformer, and MLP on Calvin without pretraining.

Table 9 shows that Dita achieves the better generalization on Calvin (ABC→D), compared to other diffusion head strategies [3, 17, 85].

**Execution steps.** Since Dita can anticipate multiple future actions, we can execute multiple steps within a single inference. Here, we analyze the impact of execution steps under a model with a trajectory length of 32, as presented in Table 10. The ablation study reveals that shorter execution steps yield slightly better results than longer ones; that is, the further the prediction extends from the current frame, the lower its accuracy. Nevertheless, the slight performance drop demonstrates that even with only 2-frame image observations, Dita can generate reliable action trajectories, underscoring its scalability.

**Shuffle Buffer Size.** The shuffle buffer size of TensorFlow datasets has a significant impact on performance. Following OpenVLA [32, 54], we utilize TensorFlow datasets for network optimization, where the shuffle buffer size similarly influences performance (Table 11), as observed in Octo [72].

**Denoising steps.** Typically, diffusion models require multiple denoising steps in image generation [62]. For diffusion-based policies in robot learning, the number of denoising steps during inference can impact control frequency. Surprisingly, we find that DDIM significantly reduces the denoising steps to 10 without compromising performance on the "Pick Coke" task as described in Table 12. With only 2 denoising steps, the model still achieves a 70.4% success rate. We attribute model performs best with 10 steps to the reduction of overfitting when fewer denoising steps are used. Unlike image generation, the action dimension in robot learning is much smaller, allowing effective denoising with fewer steps without requiring advanced techniques [70]. These results suggest that the in-context conditioning used by Dita does not hinder inference speed.

Table 12. The impact of the number of denoising steps (# Steps) of DDIM on Google Robot Simulation during inference, trained with 1000 DDPM denoising steps.

| **# Steps** | **100** | **50** | **20** | **10** | **5** | **2** |
|---|---|---|---|---|---|---|
| Pick Coke (variant) | 76.4 | 79.1 | **85.5** | 85.3 | 82.7 | 70.4 |
| Pick Coke (match) | 79.7 | 83.3 | **83.7** | 82.0 | 82. | 73.3 |
| Move Near (variant) | 52.1 | 66.0 | **73.0** | 69.5 | 63.5 | 51.6 |
| Move Near (match) | 49.1 | 72.0 | **76.0** | 74.0 | 72.0 | 65.0 |