Efficient and Scalable Diffusion Transformer Policies with Mixture of Expert Denoisers for Multitask Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract: Diffusion Policies have become widely used in Goal-Conditioned Imitation Learning, offering several appealing properties, such as generating multimodal and discontinuous behavior. As models are becoming larger to capture more complex capabilities, their computational demands increase, as shown by recent scaling laws. Therefore, continuing with the current architectures will present a computational roadblock. To address this, we propose Mixture-of-Denoising Experts (MoDE) as a novel policy for guided behavior generation. MoDE is able to achieve competitive performance to current state-of-the-art dense transformer-based Diffusion Policies while requiring fewer active parameters, reducing the inference cost significantly. To achieve this, MoDE introduces a novel routing strategy that conditions the expert selection on the current noise level of the diffusion denoising process. MoDE achieves competitive or state of the art performance on four established imitation learning benchmarks, including CALVIN and LIBERO. In addition, we perform thorough ablations on the various components in MoDE.

1 Introduction

Diffusion models [1, 2], which learn to reverse a noise-adding process, have gained popularity as policies for Imitation Learning (IL) [3, 4, 5] due to their ability to generate diverse behaviors [6] and handle complex action spaces. However, their high computational cost, resulting from large architectures and multiple denoising steps, limits their use in real-time robotics applications, especially those with limited on-board computing resources. To address this challenge, we explore Mixture of Experts (MoE) models that can scale model capacity while reducing computational requirements by utilizing only a subset of parameters during each forward pass.

We introduce **M**ixture-of-**D**enoising **E**xperts Policy (MoDE), a scalable and efficient Mixture-of-Experts (MoE) Diffusion Policy. Our work is inspired by prior results showcasing the multitask nature of the denoising process [7], where there is little transfer between the different phases in the denoising process. We present a novel noise-conditioned routing mechanism, that distributes tokens to our experts based on the current noise level. MoDE leverages noise-conditioned self-attention combined with a noise input token for enhanced noise-injection. Our proposed Policy surpasses previous Diffusion Policies with higher efficiency and demonstrates improved performance across 134 diverse tasks in challenging goal-conditioned imitation learning benchmarks: CALVIN [8] and LIBERO [9]. Through comprehensive ablation studies, we investigate the impact of various design decisions, including token routing strategies, noise-injection techniques, and expert distribution.

Submitted to the 8th Conference on Robot Learning (CoRL 2024). Do not distribute.

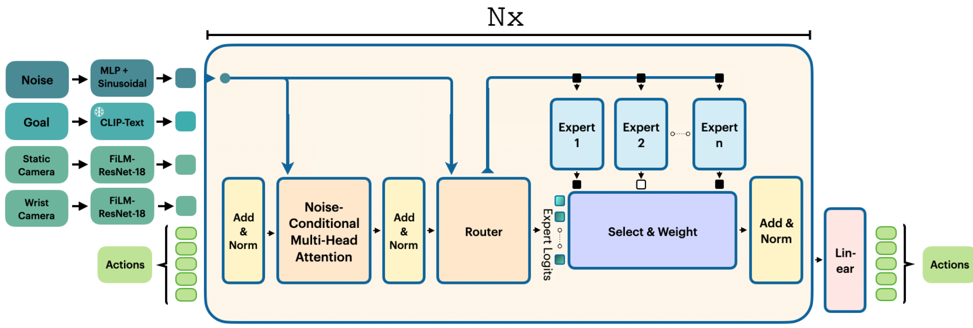


Figure 1: Overview of the proposed MoDE architecture. The model uses a transformer with causal masking from top to bottom. Each transformer block uses noise-conditional self-attention and is followed by a noise-conditioned router, that distributes tokens to specialized expert models conditioned on the current noise level. Each expert is a simple MLP with Swish-GLU activation.

2 Method

2.1 Problem Formulation

We consider the problem of learning a language-conditioned policy $\pi_\theta(\bar{\boldsymbol{a}}|\bar{\boldsymbol{s}}, \boldsymbol{g})$ given a dataset of robot demonstrations $\mathcal{T}$. The policy predicts a sequence of future actions $\bar{\boldsymbol{a}} = (\boldsymbol{a}, \ldots, \boldsymbol{a}_{i+j-1})$ of length j, conditioned on the history of state embeddings $\bar{\boldsymbol{s}} = (\boldsymbol{s}_{i-h+1}, \ldots, \boldsymbol{s}_i)$ of length h and a desired goal $\boldsymbol{g}$. The dataset contains $\tau \in \mathcal{T}$ trajectories, where trajectory consists of a sequence of triplets of state, actions, and goal $(\bar{\boldsymbol{s}}_i, \boldsymbol{a}_i, \boldsymbol{g})$. $\boldsymbol{g}$ is a language instruction. Our policy is trained to maximize the log-likelihood of the action sequence given the context of state history and goal:

$$\mathcal{L}_{\text{IL}} = \mathbb{E}\left[\sum_{(\bar{\boldsymbol{s}}, \bar{\boldsymbol{a}}, \boldsymbol{g}) \in \mathcal{T}} \log \pi_\theta\left(\bar{\boldsymbol{a}}|\bar{\boldsymbol{s}}, \boldsymbol{g}\right)\right]. \tag{1}$$

2.2 Diffusion Policy

MoDE uses the continuous-time diffusion model of EDM [10] as a policy representation. Diffusion models are a type of generative model for generating data by initially adding noise through Gaussian perturbations and then reversing this process. MoDE applies the score-based diffusion model to represent the policy $\pi_\theta(\bar{\boldsymbol{a}}|\bar{\boldsymbol{s}}, \boldsymbol{g})$. The perturbation and inverse process can be described using a stochastic differential equation:

$$\mathrm{d}\bar{\boldsymbol{a}} = \big(\beta_t \sigma_t - \dot{\sigma}_t\big)\sigma_t \nabla_a \log p_t(\bar{\boldsymbol{a}}|\bar{\boldsymbol{s}}, \boldsymbol{g})dt + \sqrt{2\beta_t}\sigma_t d\omega_t, \tag{2}$$

where β_t controls the noise injection, $d\omega_t$ refers to infinitesimal Gaussian noise, and $p_t(\bar{\boldsymbol{a}}|\bar{\boldsymbol{s}}, \boldsymbol{g})$ is the score function of the diffusion process, that moves samples away from regions of high-data density in the forward process. To generate new samples from noise a neural network is trained to approximate the score function $\nabla_{\bar{\boldsymbol{a}}} \log p_t(\bar{\boldsymbol{a}}|\bar{\boldsymbol{s}}, \boldsymbol{g})$ via Score matching (SM) [11]

$$\mathcal{L}_{\text{SM}} = \mathbb{E}_{\sigma, \bar{\boldsymbol{a}}, \boldsymbol{\epsilon}}\big[\alpha(\sigma_t)\|D_\theta(\bar{\boldsymbol{a}} + \boldsymbol{\epsilon}, \bar{\boldsymbol{s}}, \boldsymbol{g}, \sigma_t) - \bar{\boldsymbol{a}}\|_2^2\big], \tag{3}$$

where $D_\theta(\bar{\boldsymbol{a}} + \boldsymbol{\epsilon}, \bar{\boldsymbol{s}}, \boldsymbol{g}, \sigma_t)$ is the trainable neural network. During training, we sample noise from a training distribution and add it to an action sequence. The network predicts the denoised actions and computes the SM loss.

After training, we can generate a new action sequence starting from random noise by approximating the reverse SDE or related ODE in discrete steps using numerical ODE integrators. Therefore, we sample noise from the prior $\boldsymbol{a}_T \sim \mathcal{N}(\boldsymbol{0}, \sigma_T^2 \mathbf{I})$ and iteratively denoise it. MoDE uses the DDIM-solver, a numerical ODE-solver designed for diffusion models [12], that allows fast denoising of actions in a few steps. MoDE uses 10 denoising steps in all our experiments.

2.3 Mixture-of-Experts Denoising

We now present the details of MoDE's noise-conditioned expert routing. An overview of MoDE is shown in Figure 1. For language conditioning, MoDE uses a frozen CLIP language encoder model to generate a single latent goal vector. To encode images, MoDE uses FiLM-conditioned ResNets-18. Let $\mathbf{X} \in \mathbb{R}^{\text{tokens}\times\mathbf{D}}$ be a sequence of input tokens of dimension D, and σ_t be the noise level. Let $\phi(\sigma_t)$ encode the noise level into a token using a sinusoidal embedding followed by a small MLP, and let $\mathbf{X}$ also contain $\phi(\sigma_t)$ as a token. MoE $\text{MoDE}(\mathbf{X}, \phi(\sigma_\mathbf{t}))$ is a composition of L transformer blocks which we will now describe in detail. Each transformer block f^i takes $\phi(\sigma_t)$ as input as well. We now define each block f^i as a composition of a self-attention (SA) layer and an MoE layer,

$$f^i(\mathbf{X}, \phi(\sigma_\mathbf{t})) = \text{MoE}(\text{SA}(\hat{\mathbf{X}}) + \mathbf{X}, \phi(\sigma_\mathbf{t})) + \mathbf{X}. \tag{4}$$

We add the noise token, $\phi(\sigma_t)$ to all the tokens in $\mathbf{X}$ before computing the self-attention as done in [13].

$$\hat{\mathbf{X}} = \phi(\sigma_t) + \mathbf{X}, \tag{5}$$

and following [14],

$$\text{SA}(\hat{\mathbf{X}}) = \text{softmax}(\frac{1}{\sqrt{D}}[\hat{\mathbf{X}}W_Q][\hat{\mathbf{X}}W_K]^T)[\hat{\mathbf{X}}W_V]. \tag{6}$$

Given N experts $\{E_i\}_{i=1}^N$, we define the sparse MoE layer as

$$\text{MoE}(\mathbf{X}, \phi(\sigma_\mathbf{t})) = \sum_{\mathbf{i=1}}^{\mathbf{N}} \mathbf{R}(\phi(\sigma_\mathbf{t}))\mathbf{E_i}(\mathbf{X}), \tag{7}$$

where, the routing function $\mathbf{R}(\cdot) : \mathbb{R}^{\text{tokens}\times\mathbf{D}} \to \mathbb{R}^{\text{tokens}\times\mathbf{N}}$. Our noise-only conditioned routing function is defined as

$$\mathbf{R}(\phi(\sigma_\mathbf{t})) = \text{topk}(\text{softmax}(\phi(\sigma_\mathbf{t})\mathbf{W_R}), k) \tag{8}$$

While topk typically varies across different MoE implementations, we use multinomial sampling. Where we sample without replacement *k* elements according to their probabilities given by $\text{softmax}(\phi(\sigma_t)W_R)$. We set all non-chosen elements to 0. Since sampling is a non-differentiable process, scaling the expert outputs by the routing probability, $\mathbf{E_i}(\mathbf{X})$ is needed to pass gradients to the routing function. In addition, we optimally re-normalize the chosen *k* elements probabilities. After the final layer, we use a linear projection layer to get the denoised action sequence. The above formulation is general enough to explore routing variants; we tested what the router was conditioned on, such as noise only or token only. We report the results in section 3.

In addition, to mitigate expert collapse, we adopt an additional loss function (LB) that regularizes the router called load balancing [15]. Here for a given expert E_i, we compute the fraction of the tokens that were routed to it, under topk being an $\arg\max$ function, and we scale that by the average probability of routing to E_i across all tokens in a batch $\mathcal{B}$.

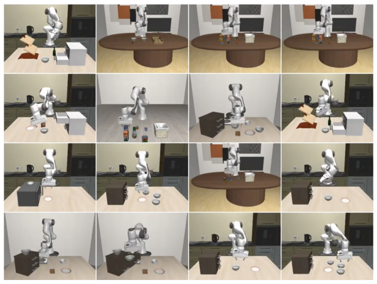
$$LB(\sigma_t) = N\sum_{n=1}^{N}\frac{1}{|\mathcal{B}|}(\sum_{i=1}^{|\mathcal{B}|}\mathbb{1}\{\mathbf{R}(\phi(\sigma_{\mathbf{t_i}}))_\mathbf{n} > \mathbf{0}\})\frac{\mathbf{1}}{|\mathcal{B}|}(\sum_{\mathbf{i=1}}^{|\mathcal{B}|}\text{softmax}(\phi(\sigma_{\mathbf{t_i}})\mathbf{W_R})_\mathbf{n}) \tag{9}$$

We use $\gamma = 0.01$ for the load-balancing loss.

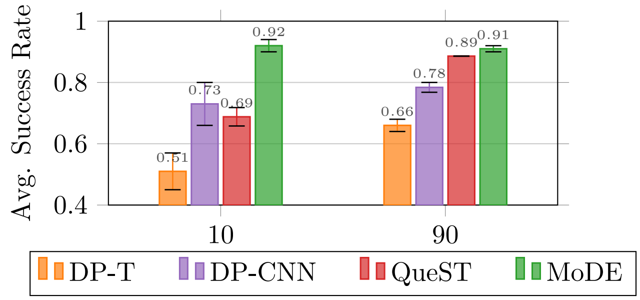
3 Evaluation

Our experiments aim to answer four key questions: (I) How does MoDE compare to other policies and prior diffusion transformer architectures in terms of performance?

We compare MoDE against prior diffusion transformer architecture [5], ensuring fair comparisons by using a similar number of active parameters. MoDE uses 8 layers with 4 experts and a latent dimension of 1024 in all experiments. Detailed hyperparameters are provided in the Appendix (Table 1).



(a) LIBERO-90 Tasks



(b) Results for LIBERO-10 and LIBERO-90

Figure 2: Visualization and Results for LIBERO environment. (a) Few example environments and tasks of the LIBERO-90 task suite. (b) Average results for both LIBERO challenges averaged over 3 seeds with 20 rollouts for each task.

3.1 Long-Horizon Multi-Task Experiments

We first evaluate MoDE on the LONG-challenge and LIBERO-90 challenge of the LIBERO benchmark [9]. The LONG challenge requires a model to learn 10 tasks in different settings. It demands long-horizon behavior generation with several hundreds of steps for completion. The 90 variant tests policies on 90 diverse short-horizon tasks in different environments. Figure 2a visualizes a few examples of these tasks. All environments feature two cameras: a static one and a wrist-mounted camera, used to encode the current observation using FiLM-ResNets-18. We test each policy 20 times on each task and report the average results over 3 seeds. We use an action chunking length of 10 and a history length of 1. MoDE and all other diffusion architectures use FiLM-conditioned ResNets-18 with a CLIP sentence embedding to encode the goal and the images.

Baselines. We compare MoDE against three state-of-the-art baselines: 1) The Diffusion Transformer (DP-T) architecture [5], which conditions on noise and observations using a cross-attention module. 2) The standard Diffusion Policy CNN-based architecture (DP-CNN). 3) QueST [16], a transformer-based policy that learns discrete action representations using vector-quantized embeddings of action sequences. We tested all baselines ourselves, except for QueST, whose results are taken directly from their paper.

Results. The performance of all models on the benchmark is summarized in Figure 2b. Overall, MoDE achieves the highest average performance in both benchmarks, while the QueST baseline is the second best in the LIBERO-90 setting and the CNN-architecture is second best in the long horizon setting. These results demonstrate MoDE's ability to learn long-horizon tasks with high accuracy. The performance gap is more pronounced in the challenging LIBERO-10 experiment, where MoDE is the first policy to achieve an over 90% success rate. Furthermore, MoDE surpasses prior best Diffusion baselines by an average of 16% in both settings, all while maintaining its computational advantage. This showcases MoDE's ability to achieve state-of-the-art performance with a more efficient use of computational resources.

4 Conclusion

In this work, we introduced Mixture-of-Denoising Experts (MoDE), a novel Diffusion Policy that leverages a mixture of experts Transformer to enhance the performance and efficiency of diffusion policies. We also proposed a noise-conditioned routing strategy for learning specialized experts within our model. In our extensive experiments and ablation studies across diverse benchmarks, we demonstrated the advantages of MoDE to outperform prior Diffusion Policies with a lower number of parameters and 40% less FLOPS during inference. In future work, we want to experiment with more routing strategies, such as expert-choice routing [17].

References

[1] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[2] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2020.

[3] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy. https://octo-models.github.io, 2023.

[4] M. Reuss, M. Li, X. Jia, and R. Lioutikov. Goal conditioned imitation learning using score-based diffusion policies. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[5] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[6] X. Jia, D. Blessing, X. Jiang, M. Reuss, A. Donat, R. Lioutikov, and G. Neumann. Towards diverse behaviors: A benchmark for imitation learning with human demonstrations. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=6pPYRXKPpw.

[7] T. Hang, S. Gu, C. Li, J. Bao, D. Chen, H. Hu, X. Geng, and B. Guo. Efficient diffusion training via min-snr weighting strategy, 2024.

[8] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 2022.

[9] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023.

[10] T. Karras, M. Aittala, T. Aila, and S. Laine. Elucidating the design space of diffusion-based generative models. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[11] P. Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011. doi:10.1162/NECO_a_00142.

[12] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. In *ICLR*, 2021.

[13] A. Hatamizadeh, J. Song, G. Liu, J. Kautz, and A. Vahdat. Diffit: Diffusion vision transformers for image generation. *arXiv preprint arXiv:2312.02139*, 2023.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[15] W. Fedus, B. Zoph, and N. Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

[16] A. Mete, H. Xue, A. Wilcox, Y. Chen, and A. Garg. Quest: Self-supervised skill abstractions for learning continuous control. *arXiv preprint arXiv:2407.15840*, 2024.

[17] Y. Zhou, T. Lei, H. Liu, N. Du, Y. Huang, V. Zhao, A. Dai, Z. Chen, Q. Le, and J. Laudon. Mixture-of-experts with expert choice routing, 2022.

[18] X. Li, M. Liu, H. Zhang, C. Yu, J. Xu, H. Wu, C. Cheang, Y. Jing, W. Zhang, H. Liu, et al. Vision-language foundation models as effective robot imitators. In *International Conference on Learning Representations*, 2024.

[19] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736, 2022.

[20] T. Brooks, A. Holynski, and A. A. Efros. Instructpix2pix: Learning to follow image editing instructions. In *CVPR*, 2023.

[21] K. Black, M. Nakamoto, P. Atreya, H. Walke, C. Finn, A. Kumar, and S. Levine. Zero-shot robotic manipulation with pretrained image-editing diffusion models. *arXiv preprint arXiv:2310.10639*, 2023.

[22] H. Wu, Y. Jing, C. Cheang, G. Chen, J. Xu, X. Li, M. Liu, H. Li, and T. Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations*, 2024.

[23] O. Mees, L. Hermann, and W. Burgard. What matters in language conditioned robotic imitation learning over unstructured data. *IEEE Robotics and Automation Letters (RA-L)*, 7(4):11205–11212, 2022.

[24] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[25] W. Peebles and S. Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023.

[26] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet. Learning latent plans from play, 2019.

[27] N. M. M. Shafiullah, Z. J. Cui, A. Altanzaya, and L. Pinto. Behavior transformers: Cloning k modes with one stone. In *Thirty-Sixth Conference on Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=agTr-vRQsa.

[28] Z. J. Cui, Y. Wang, N. M. M. Shafiullah, and L. Pinto. From play to policy: Conditional behavior generation from uncurated robot data. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=c7rM7F7jQjN.

[29] S. Lee, Y. Wang, H. Etukuru, H. J. Kim, N. M. M. Shafiullah, and L. Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024.

[30] Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

[31] Z. Xian, N. Gkanatsios, T. Gervet, T.-W. Ke, and K. Fragkiadaki. Chaineddiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=W0zgY2mBTA8.

[32] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *arXiv preprint arXiv:2402.10885*, 2024.

[33] X. Li, V. Belagali, J. Shang, and M. S. Ryoo. Crossway diffusion: Improving diffusion-based visuomotor policy via self-supervised learning. *arXiv preprint arXiv:2307.01849*, 2023.

[34] P. M. Scheikl, N. Schreiber, C. Haas, N. Freymuth, G. Neumann, R. Lioutikov, and F. Mathis-Ullrich. Movement primitive diffusion: Learning gentle robotic manipulation of deformable objects. *arXiv preprint arXiv:2312.10008*, 2023.

[35] M. Reuss, Ö. E. Yağmurlu, F. Wenzel, and R. Lioutikov. Multimodal diffusion transformer: Learning versatile behavior from multimodal goals. In *Robotics: Science and Systems*, 2024.

[36] A. Ajay, Y. Du, A. Gupta, J. B. Tenenbaum, T. S. Jaakkola, and P. Agrawal. Is conditional generative modeling all you need for decision making? In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=sP1fo2K9DFG.

[37] M. Janner, Y. Du, J. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pages 9902–9915. PMLR, 2022.

[38] J. Pari, N. Shafiullah, S. Arunachalam, and L. Pinto. The Surprising Effectiveness of Representation Learning for Visual Imitation. In *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022. doi:10.15607/RSS.2022.XVIII.010.

[39] T.-H. Wang, J. Zheng, P. Ma, Y. Du, B. Kim, A. E. Spielberg, J. B. Tenenbaum, C. Gan, and D. Rus. Diffusebot: Breeding soft robots with physics-augmented generative diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=1zo4iioUEs.

[40] Y. Du, M. Yang, B. Dai, H. Dai, O. Nachum, J. B. Tenenbaum, D. Schuurmans, and P. Abbeel. Learning universal policies via text-guided video generation. *arXiv preprint arXiv:2302.00111*, 2023.

[41] P.-C. Ko, J. Mao, Y. Du, S.-H. Sun, and J. B. Tenenbaum. Learning to Act from Actionless Video through Dense Correspondences. *arXiv:2310.08576*, 2023.

[42] A. Ajay, S. Han, Y. Du, S. Li, A. Gupta, T. Jaakkola, J. Tenenbaum, L. Kaelbling, A. Srivastava, and P. Agrawal. Compositional foundation models for hierarchical planning. *arXiv preprint arXiv:2309.08587*, 2023.

[43] J. Carvalho, A. T. Le, M. Baierl, D. Koert, and J. Peters. Motion planning diffusion: Learning and planning of robot motions with diffusion models. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1916–1923. IEEE, 2023.

[44] J. Urain, N. Funk, J. Peters, and G. Chalvatzaki. Se (3)-diffusionfields: Learning smooth cost functions for joint grasp and motion optimization through diffusion. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5923–5930. IEEE, 2023.

[45] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017.

[46] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[47] N. Du, Y. Huang, A. M. Dai, S. Tong, D. Lepikhin, Y. Xu, M. Krikun, Y. Zhou, A. W. Yu, O. Firat, B. Zoph, L. Fedus, M. Bosma, Z. Zhou, T. Wang, Y. E. Wang, K. Webster, M. Pellat, K. Robinson, K. Meier-Hellstern, T. Duke, L. Dixon, K. Zhang, Q. V. Le, Y. Wu, Z. Chen, and C. Cui. Glam: Efficient scaling of language models with mixture-of-experts, 2022.

[48] Z. Chi, L. Dong, S. Huang, D. Dai, S. Ma, B. Patra, S. Singhal, P. Bajaj, X. Song, X.-L. Mao, H. Huang, and F. Wei. On the representation collapse of sparse mixture of experts. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=mWaYC6CZf5.

[49] H. Hazimeh, Z. Zhao, A. Chowdhery, M. Sathiamoorthy, Y. Chen, R. Mazumder, L. Hong, and E. H. Chi. Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning. *CoRR*, abs/2106.03760, 2021. URL https://arxiv.org/abs/2106.03760.

[50] S. Roller, S. Sukhbaatar, A. Szlam, and J. Weston. Hash layers for large sparse models. *CoRR*, abs/2106.04426, 2021. URL https://arxiv.org/abs/2106.04426.

[51] M. Lewis, S. Bhosale, T. Dettmers, N. Goyal, and L. Zettlemoyer. Base layers: Simplifying training of large, sparse models, 2021.

[52] J. Obando-Ceron, G. Sokar, T. Willi, C. Lyle, J. Farebrother, J. Foerster, G. K. Dziugaite, D. Precup, and P. S. Castro. Mixtures of experts unlock parameter scaling for deep rl. *arXiv preprint arXiv:2402.08609*, 2024.

[53] O. Celik, D. Zhou, G. Li, P. Becker, and G. Neumann. Specializing versatile skill libraries using local mixture of experts. In A. Faust, D. Hsu, and G. Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1423–1433. PMLR, 08–11 Nov 2022. URL https://proceedings.mlr.press/v164/celik22a.html.

[54] O. Celik, A. Taranovic, and G. Neumann. Acquiring diverse skills using curriculum reinforcement learning with mixture of experts. *arXiv preprint arXiv:2403.06966*, 2024.

[55] K. Hansel, J. Urain, J. Peters, and G. Chalvatzaki. Hierarchical policy blending as inference for reactive robot control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10181–10188. IEEE, 2023.

[56] A. T. Le, K. Hansel, J. Peters, and G. Chalvatzaki. Hierarchical policy blending as optimal transport. In *Learning for Dynamics and Control Conference*, pages 797–812. PMLR, 2023.

[57] C. Riquelme, J. Puigcerver, B. Mustafa, M. Neumann, R. Jenatton, A. S. Pinto, D. Keysers, and N. Houlsby. Scaling vision with sparse mixture of experts, 2021.

[58] O. Mees, A. Eitel, and W. Burgard. Choosing smartly: Adaptive multimodal fusion for object detection in changing environments. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 151–156. IEEE, 2016.

[59] D. Blessing, O. Celik, X. Jia, M. Reuss, M. X. Li, R. Lioutikov, and G. Neumann. Information maximizing curriculum: A curriculum-based approach for learning versatile skills. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=7eW6NzSE4g.

[60] M. X. Li, O. Celik, P. Becker, D. Blessing, R. Lioutikov, and G. Neumann. Curriculum-based imitation of versatile skills. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2951–2957, 2023. doi:10.1109/ICRA48891.2023.10160543.

[61] B. Park, S. Woo, H. Go, J.-Y. Kim, and C. Kim. Denoising task routing for diffusion models. *arXiv preprint arXiv:2310.07138*, 2023.

[62] H. Go, Y. Lee, J.-Y. Kim, S. Lee, M. Jeong, H. S. Lee, and S. Choi. Towards practical plug-and-play diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1962–1971, 2023.

[63] B. Park, H. Go, J.-Y. Kim, S. Woo, S. Ham, and C. Kim. Switch diffusion transformer: Synergizing denoising tasks with sparse mixture-of-experts. *arXiv preprint arXiv:2403.09176*, 2024.

[64] Y. Lee, J. Kim, H. Go, M. Jeong, S. Oh, and S. Choi. Multi-architecture multi-expert diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 13427–13436, 2024.

[65] H. Bharadhwaj, J. Vakil, M. Sharma, A. Gupta, S. Tulsiani, and V. Kumar. Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking, 2023.

[66] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[67] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[68] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-1: Robotics transformer for real-world control at scale. In *arXiv preprint arXiv:2212.06817*, 2022.

[69] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, Q. Vuong, V. Vanhoucke, H. Tran, R. Soricut, A. Singh, J. Singh, P. Sermanet, P. R. Sanketi, G. Salazar, M. S. Ryoo, K. Reymann, K. Rao, K. Pertsch, I. Mordatch, H. Michalewski, Y. Lu, S. Levine, L. Lee, T.-W. E. Lee, I. Leal, Y. Kuang, D. Kalashnikov, R. Julian, N. J. Joshi, A. Irpan, brian ichter, J. Hsu, A. Herzog, K. Hausman, K. Gopalakrishnan, C. Fu, P. Florence, C. Finn, K. A. Dubey, D. Driess, T. Ding, K. M. Choromanski, X. Chen, Y. Chebotar, J. Carbajal, N. Brown, A. Brohan, M. G. Arenas, and K. Han. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=XMQgwiJ7KSX.

[70] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[71] J. Gu, S. Kirmani, P. Wohlhart, Y. Lu, M. G. Arenas, K. Rao, W. Yu, C. Fu, K. Gopalakrishnan, Z. Xu, P. Sundaresan, P. Xu, H. Su, K. Hausman, C. Finn, Q. Vuong, and T. Xiao. Rt-trajectory: Robotic task generalization via hindsight trajectory sketches. In *International Conference on Learning Representations*, 2024.

[72] O. X.-E. Collaboration, A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, A. Raffin, A. Wahid, B. Burgess-Limerick, B. Kim, B. Schölkopf, B. Ichter, C. Lu, C. Xu, C. Finn, C. Xu, C. Chi, C. Huang, C. Chan, C. Pan, C. Fu, C. Devin, D. Driess, D. Pathak, D. Shah, D. Büchler, D. Kalashnikov, D. Sadigh, E. Johns, F. Ceola, F. Xia, F. Stulp, G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Schiavi, H. Su, H.-S. Fang, H. Shi, H. B. Amor, H. I. Christensen, H. Furuta, H. Walke, H. Fang, I. Mordatch, I. Radosavovic, I. Leal, J. Liang, J. Kim, J. Schneider, J. Hsu, J. Bohg, J. Bingham, J. Wu, J. Wu, J. Luo, J. Gu, J. Tan, J. Oh, J. Malik, J. Tompson, J. Yang, J. J. Lim, J. Silvério, J. Han, K. Rao, K. Pertsch, K. Hausman, K. Go, K. Gopalakrishnan, K. Goldberg, K. Byrne, K. Oslund, K. Kawaharazuka, K. Zhang, K. Majd, K. Rana, K. Srinivasan, L. Y. Chen, L. Pinto, L. Tan, L. Ott, L. Lee, M. Tomizuka, M. Du, M. Ahn, M. Zhang, M. Ding, M. K. Srirama, M. Sharma, M. J. Kim, N. Kanazawa, N. Hansen, N. Heess, N. J. Joshi, N. Suenderhauf, N. D. Palo, N. M. M. Shafiullah, O. Mees, O. Kroemer, P. R. Sanketi, P. Wohlhart, P. Xu, P. Sermanet, P. Sundaresan, Q. Vuong, R. Rafailov, R. Tian, R. Doshi, R. Martín-Martín, R. Mendonca, R. Shah, R. Hoque, R. Julian, S. Bustamante, S. Kirmani, S. Levine, S. Moore, S. Bahl, S. Dass, S. Song, S. Xu, S. Haldar, S. Adebola, S. Guist, S. Nasiriany, S. Schaal, S. Welker, S. Tian, S. Dasari, S. Belkhale, T. Osa, T. Harada, T. Matsushima, T. Xiao, T. Yu, T. Ding, T. Davchev, T. Z. Zhao, T. Armstrong, T. Darrell, V. Jain, V. Vanhoucke, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Wang, X. Zhu, X. Li, Y. Lu, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Xu, Y. Wang, Y. Bisk, Y. Cho, Y. Lee, Y. Cui, Y. hua Wu, Y. Tang, Y. Zhu, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Xu, and Z. J. Cui. Open X-Embodiment: Robotic learning datasets and RT-X models. https://arxiv.org/abs/2310.08864, 2023.

A Appendix / supplemental material

Hyperparameter	CALVIN ABCD	CALVIN ABC	LIBERO-10	LIBERO-90
Number of Transformer Layers	8	8	8	8
Number Experts	4	4	4	4
Attention Heads	8	8	8	8
Action Chunk Size	10	10	10	10
History Length	1	1	1	1
Embedding Dimension	1024	1024	1024	1024
Image Encoder	FiLM-ResNet18	FiLM-ResNet18	FiLM-ResNet18	FiLM-ResNet18
Goal Lang Encoder	CLIP ViT-B/32	CLIP ViT-B/32	CLIP ViT-B/32	CLIP ViT-B/32
Attention Dropout	0.3	0.3	0.3	0.3
Residual Dropout	0.1	0.1	0.1	0.1
MLP Dropout	0.1	0.1	0.1	0.1
Optimizer	AdamW	AdamW	AdamW	AdamW
Betas	[0.9, 0.9]	[0.9, 0.9]	[0.9, 0.9]	[0.9, 0.9]
Learning Rate	1e-4	1e-4	1e-4	1e-4
Transformer Weight Decay	0.05	0.05	0.05	0.05
Other weight decay	0.05	0.05	0.05	0.05
Batch Size	512	512	512	512
Train Steps in Thousands	30	25	15	40
$\sigma_{\max}$	80	80	80	80
$\sigma_{\min}$	0.001	0.001	0.001	0.001
σ_t	0.5	0.5	0.5	0.5
EMA	True	True	True	True
Time steps	Exponential	Exponential	Exponential	Exponential
Sampler	DDIM	DDIM	DDIM	DDIM
Parameter Count (Millions)	460	460	460	460

Table 1: Summary of all the Hyperparameters for the MoDE policy used in our experiments.

A.1 Experiments Details

A.1.1 CALVIN Benchmark

The CALVIN benchmark [8] is an established IL benchmark for learning language-conditioned behavior from human play data. In contrast to other benchmarks the data does not contain structured demonstrations, where the robot completes one task, but instead, the dataset was collected by humans, that randomly interact with the environment. From these long-horizon trajectories across the 4 different settings, the authors randomly cut out short sequences with 64 frames and labeled them with the task label. While the dataset offers models to train on the unlabeled part too, we restricted MoDE to only train on the labeled parts. The Franke Emika Panda robot is controlled using a Delta-End-Effector Space with a discrete gripper. We use two cameras to encode the current scene: a static camera and a wrist one and predict the next 10 actions, before receiving the next observations and generating another set of 10 actions.

CALVIN ABC. We train MoDE and our dense transformer baseline for 25k training steps with a batch size of 512 on a 4 GPU Cluster Node with 4 A6000 NVIDIA GPUs for 6.5 hours with all 1000 rollouts at the end of training. We report the mean results averaged over 3 seeds as done in all relevant prior work. All baselines are reported from the original paper given the standardized evaluation protocol of CALVIN [8].

CALVIN ABCD. We train MoDE and our dense transformer baseline for 30k training steps with a batch size of 512 on a 4 GPU Cluster Node with 4 A6000 NVIDIA GPUs for 7.5 hours with all 1000 rollouts at the end of training. We report the mean results averaged over 3 seeds as done in all relevant prior work.

A.1.2 LIBERO Benchmark

LIBERO-10. The LIBERO-10 benchmark consists of 50 demonstrations for 10 different tasks that are all labeled with a text instruction. The Franka Emika Panda robot is controlled using an end-effector controller. Similar to CALVIN all models have access to two camera inputs: a static one and a wrist camera. We train MoDE and our dense transformer baseline for 50 epochs with a batch size of 512 on a 4 GPU Cluster Node with 4 A6000 NVIDIA GPUs for 2 hours with all 200 rollouts

Model	Block Push	Relay Kitchen	CAL ABC	CAL ABCD	L-10	Average
Dense T	0.96±0.02	3.73±0.12	**2.83±0.19**	4.13±0.11	0.91±0.02	0.839±0.144
Token-Router	0.97±0.01	**3.85±0.03**	2.67±0.04	4.29±0.08	0.90±0.01	0.845±0.161
σ_t-Router	*0.97±0.01*	3.79±0.04	*2.79±0.16*	**4.30±0.02**	**0.92±0.02**	**0.851±0.151**

Table 2: Overview of the performance of all different token routing strategies used for MoDE across 5 benchmarks. We mark the best result for each environment in **bold** and the second best in *cursive*. We use CAL to represent CALVIN. To average the results, we normalize all scores and compute the average over all environments.

at the end of training. The benchmark does require to test models on 10 different long-horizon tasks. We test each task 20 times for each model and report the final average performance overall 10 tasks.

LIBERO-90. The LIBERO-10 benchmark consists of 50 demonstrations for 90 different tasks that are all labeled with a text instruction. The Franka Emika Panda robot is controlled using an end-effector controller. We train MoDE and our dense transformer baseline for $50k$ steps with a batch size of 512 on a 4 GPU Cluster Node with 4 A6000 NVIDIA GPUs for 12 hours with all 1800 rollouts at the end of training. The benchmark does require to test models on 90 different tasks in many different environments. We test each task 20 times for each model and report the final average performance overall 90 tasks.

A.2 Baselines

Below we explain several baselines used in the experiments in detail:

Diffusion Policy-CNN/T Inspired by [5], we evaluate extension of the DDPM based Diffusion Policy framework for goal-conditioned Multi-task learning. We evaluate two versions: the CNN-based variant and the Diffusion-Transformer variant, that is conditioned on context and noise using cross-attention. For our experiments we also use EDM-based Diffusion framework for fair comparison against MoDE. We optimized the ideal number of layers and latent dimension for the Transformer baseline and our final version uses 8 layers with a latent dimension of 1024. Larger or smaller variants resulted in lower average performance.

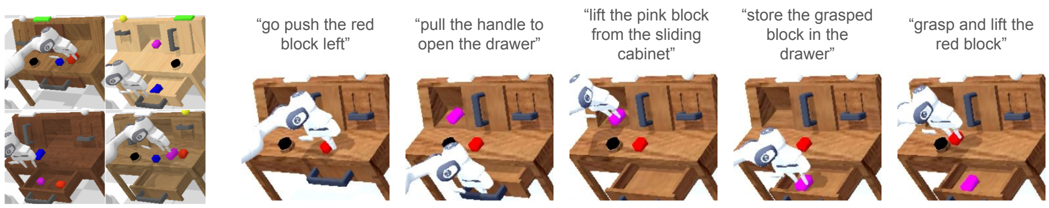
RoboFlamingo. RoboFlamingo [18] is a Vision-Language-Models (VLM) finetuned for behavior generation. The authors use a 3 billion parameter Flamingo model [19] and fine-tune it on CALVIN by freezing the forward blocks and only fine-tuning a new Perceiver Resampler module to extract features from a frozen vision-transformer image encoder and the cross-attention layers to process the image features. Finally, a new action head is learned to generate actions. Overall, the finetuning requires training approx. 1 billion of the parameters. We report the reported results from the paper since they use the standard CALVIN evaluation suite.

SuSIE. This model first finetunes Instruct2Pix, an image-generation diffusion model, that generates images conditioned on another image and a text description [20] on the local CALVIN robotics domain and uses it as a high-level goal generator. The low-level policy is a Convolutional neural network (CNN)-based Diffusion Policy, that predicts the next 4 actions given the current state embedding and desired sub-goal from the high-level policy [21].

GR-1 A causal GPT-Transformer model [22], that has been pretrained on large-scale generative video prediction of human videos. Later, the model is finetuned using co-training of action prediction and video prediction on CALVIN. We report the results directly from their paper for the CALVIN benchmark.

A.3 Scaling Multi-Task Experiments

Next, we evaluate MoDE efficacy on the demanding **CALVIN Language-Skills Benchmark** [8], an established image-based benchmark for IL. This benchmark contains a large dataset of human-recorded demonstrations. First, MoDE is tested on the **ABCD→D** challenge, which involves $22,966$



(a) Environments (b) Example CALVIN-Rollout

Figure 3: Overview of the CALVIN environment. (a) CALVIN contains four different settings (A,B,C,D) with different configurations of slides, drawers and textures. (b) Example rollout consisting of 5 tasks in sequence. The next goal is only given to the policy, if it manages to complete the prior.

Train→Test	Method	Active Params in Million	PrT	No. Instructions in a Row (1000 chains)					
				1	2	3	4	5	**Avg. Len.**
ABCD→D	Diff-P-CNN	321	×	86.3%	72.7%	60.1%	51.2%	41.7%	3.16±0.06
	Diff-P-T	194	×	78.3%	53.9%	33.8%	20.4%	11.3%	1.98±0.09
	RoboFlamingo	1000	✓	96.4%	89.6%	82.4%	74.0%	66.0%	4.09±0.00
	GR-1	130	✓	94.9%	89.6%	84.4%	78.9%	73.1%	4.21±0.00
	MoDE	277	×	**96.6%**	**90.6%**	**86.6%**	**80.9%**	**75.5%**	**4.30±0.02**
ABC→D	Diff-P-CNN	321	×	63.5%	35.3%	19.4%	10.7%	6.4%	1.35±0.05
	Diff-P-T	194	×	62.2%	30.9%	13.2%	5.0%	1.6%	1.13±0.02
	RoboFlamingo	1000	✓	82.4%	61.9%	46.6%	33.1%	23.5%	2.47±0.00
	SuSIE	860+	✓	87.0%	69.0%	49.0%	38.0%	26.0%	2.69±0.00
	GR-1	130	✓	85.4%	**71.2%**	**59.6%**	**49.7%**	**40.1%**	**3.06±0.00**
	MoDE	277	×	**87.7%**	69.8%	52.11%	40.2%	29.1%	2.79±0.18

Table 3: Performance comparison on the two CALVIN challenges. The table reports average success rates for individual tasks within instruction chains and the average rollout length (Avg. Len.) to complete 5 consecutive instructions, based on 1000 chains. Zero standard deviation indicates methods without reported average performance. ’Prt’ denotes models requiring policy pretraining. Parameter counts exclude language encoders.

interaction sequences across four environments (A, B, C, D), with each consisting of 64 timesteps and 34 diverse tasks. These tasks require the acquisition of complex, sequential behaviors and the ability to chain together different skills. Figure 3a depicts the diverse configurations of interactive elements within these environments. This particular challenge examines the scaling abilities of policies trained on a rich variety of data and skills across multiple settings. All policies are tested on 1000 instructions chains consisting of 5 tasks in sequence in environment D following the official protocol of CALVIN [8]. One example rollout with 5 different tasks is visualized in Figure 3b. In terms of scoring, the model receives 1 point for completing a task and only progresses to the next task upon completion of the prior one. We report the average sequence length over 3 seeds with 1000 instruction chains each.

Baselines. We test MODE against several methods specialized for learning language-conditioned behavior and against other baseline diffusion policy architectures. We also compare MoDE against RoboFlamingo and GR-1. RoboFlamingo is a fine-tuned Vision-Language-Action model, that contains around 3 billion parameters and has been pre-trained on diverse internet data. GR-1 is a generative decoder-only Transformer pretrained on large-scale video generation and then co-finetuned on CALVIN [22]. If available, we report the average performance of all prior work directly from their paper, given the standard evaluation protocol in CALVIN [23].

Results. Our findings, outlined in Table 3 reveal that MoDE outperforms all other policies in terms of average success rate. Moreover, MoDE outperforms well-established baselines like RoboFlamingo and GR-1, which depend on extensive internet-scale pretraining for their results. Notably, while GR-1 uses fewer active parameters (130M compared to MoDE’s 277M), it operates with a history length of 10 and 14 tokens for each timestep. Despite this, MoDE proves more computationally efficient, requiring fewer FLOPs during inference (7.03 vs 7.93 GFLOPs for GR-1). The combination of sota performance, lower computational demands, and no need for resource-intensive pretraining positions MoDE as a highly practical solution for multitask settings.

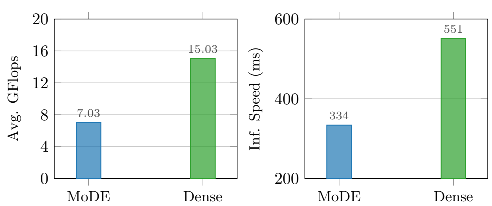
A.4 Zero-shot Generalization Experiments

Finally, we then extend our investigation to the **ABC→D** challenge in the second phase, testing MoDE's zero-shot generalization abilities. In this experiment, models are only trained on data from the first three CALVIN environments A,B,C and tested on the unseen setting of environment D, which has different positions of relevant objects and texture of the table. This requires policies, that are able to generalize their learned behavior to new environment configurations and different textures, which is especially challenging.

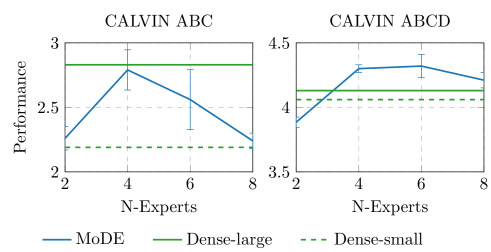
Baselines. For this experiment, we compare MODE against the previous CALVIN baselines, with the addition of SuSIE [21]. A hierarchical policy utilizing a finetuned image-generation model, Instruct2Pix [20], to generate goal images, which guide a low-level diffusion policy. The high-level goal generation model requires large-scale pretraining. SuSIE's results are based on 100 rollouts only, without standard deviation, due to the computational cost of generating subgoal images.

Results. The results of this experiment are summarized in the lower part of Table 3. MoDE outperforms all tested baselines except for GR-1 and surpasses all other Diffusion Policy architectures by a wide margin. While GR-1 is slightly better, it requires expensive large-scale pre-training with 32 GPUS and 7 days of total training time to achieve these results. In contrast, MoDE trains on 4 GPUs in 8 hours and achieves similar performance without additional pretraining requirements. Therefore, in response to Question (I), we affirmatively conclude that Mixture-of-Experts models not only enhance scaling performance but also demonstrate strong zero-shot generalization capabilities.

A.5 Computational Efficiency of MoDE



(a) Computational efficiency comparison between MoDE and Dense-Transformer model with the same number of parameters. Left: FLOP count for both model variants. Right: Average inference speed over 10 forward passes. MoDE demonstrates superior efficiency with lower FLOP count and faster inference.



(b) Scaling performance of MoDE and Dense-MoDE on CALVIN ABC and ABCD environments, showing average performance for 2 to 8 experts using best-performing variants for each environment.

We compute the average time for MoDE and a dense transformer baseline with a similar number of parameters for 10 forward passes to assess the computational efficiency of both. The results of this ablation are summarized in Figure 4a. MoDE is around 40% faster than the dense transformer model with lower FLOPS and fewer parameters than the dense model. Given the prior experimental results, MoDE does increase over dense models in terms of performance, efficiency and inference speed.

A.6 Ablation Studies

To thoroughly evaluate MoDE's performance and design choices, we conducted a series of ablation studies. These experiments address our research questions: the computational efficiency of MoDE (Question II), the impact of different routing strategies (Question III), and the distribution of tokens to experts (Question IV).

A.6.1 What design decisions affect MoDE's performance?

First, we assess the impact of various design decisions on MoDE's performance. We ablate the choice of noise-conditioning and various MoE strategies on the LIBERO-10 benchmark. The results are summarized in Table 4.

	Avg. Success.
MoDE	**0.92**
- Input Noise Token	0.90
- Noise-cond Attention	0.85
FiLM Noise Conditioning	0.81
TopK=1	0.91
Shared Expert	0.90
$\gamma = 0.1$	0.90
$\gamma = 0.001$	0.86
256 Embed Dim	0.86
512 Embed Dim	0.87

Table 4: Ablation Studies for MoDE on LIBERO-10. All results are averaged over 3 seeds with 20 rollouts each.

Noise-Injection Ablations. Our experiments reveal the importance of proper noise conditioning in MoDE. The full MoDE model, which uses both input noise tokens and noise-conditioned self-attention, achieves the best performance with an average success rate of 0.92. Removing the input noise token slightly decreases performance to 0.90, highlighting the complementary nature of both conditioning methods. Using only the noise token for conditioning, without noise-conditioned self-attention, further reduces performance to 0.85. Interestingly, using FiLM noise conditioning [24], a common approach in image-diffusion models [25], yields the lowest performance in this group at 0.81. These results underscore the effectiveness of our proposed noise conditioning strategy in MoDE, demonstrating a clear performance advantage of 0.08 over the FiLM approach.

MoE Ablations. Next, we ablate several design decisions regarding Mixture-of-Experts. First, we test the topk number of used experts. Setting topk to 1 only marginally lowers the average performance from 0.92 to 0.91. MoDE maintains robust performance even with a single expert. We also examine the effect of using a shared expert, where the model consistently employs the same expert in all cases. This approach achieves a comparable average success rate of 0.90. Different choices for the token-distribution loss are also tested. While MoDE uses $\gamma = 0.01$ as a default value, we experiment with γ values of 0.1 and 0.001, which result in average success rates of 0.90 and 0.86, respectively. These results indicate that a γ value of 0.01 strikes the best performance.

Latent Dimension. We investigate the impact of varying the latent dimension in MoDE, testing dimensions of 256, 512, and 1024 (our default). The results show that increasing the latent dimension from 256 to 512 yields a modest performance improvement from 0.86 to 0.87, while further increasing to 1024 provides a more substantial boost to 0.92. This suggests that a larger latent dimension allows MoDE to capture more complex representations, leading to improved performance.

A.7 Detailed Experimental Results

We summarize the ablations regarding the choice of routing in Table 2. Therefore, we test two 2 different routing strategies across 5 benchmarks.

A.8 Additional Ablation Studies

A.8.1 Optimal Routing Strategy for Diffusion Transformers

Next, we answer Question (II) by testing different routing strategies for our diffusion-transformer policy across several environments. We test two different token routing strategies: 1) Token-only conditioned Routing and 2) Noise-only Token Routing. (1) is commonly used in LLMs, where the routing is decided based on the tokens only. We test these strategies in five experiments and report the average performance over 3 seeds: Noise-only Routing achieves an average normalized performance of 0.851, slightly outperforming Token-only Routing, which achieves 0.845. Detailed results are summarized in Table 2 in the Appendix. The results demonstrate the effectiveness of our proposed routing strategy. While the performance difference is small, Noise-only Routing offers an additional advantage: the ability to predict all used experts based on noise levels once before roll-outs, enabling faster inference. This is particularly beneficial for robotics applications.

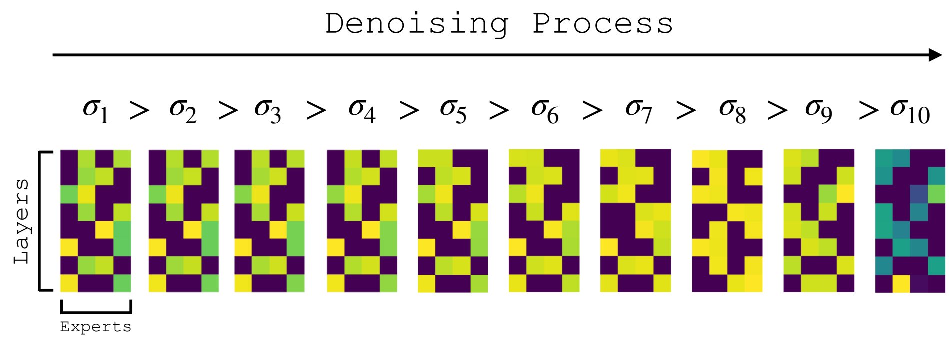


Figure 5: Visualized Expert Utilization. The average usage of all experts in MoDE across all layers is shown. Purple color corresponds to low usage and yellow color to high one, and each image is separately normalized. The average activation shows that MoDE learns to utilize different experts for different noise levels.

A.8.2 How does the model distribute the tokens to different experts?

To address Question IV, we analyzed how MoDE distributes tokens to different experts using a pre-trained model. Figure 5 visualizes the average usage of each expert in each model layer during inference across various noise levels, using 10 denoising steps for clarity. Our analysis reveals that MoDE learns to utilize different experts for various noise levels, suggesting that the router has specialized for different noise regimes. A transition in expert utilization occurs around σ_8. In the first layer, the model learns an expert specialized for low-noise levels, primarily used in the last denoising step at σ_{min}. These findings affirmatively answer Question IV, demonstrating that MoDE effectively distributes tokens across experts based on noise levels.

A.8.3 How does the model scale with more experts?

Finally, we analyze the effect of increasing the number of experts in MoDE. The results are presented in Figure 4b, where we evaluate MoDE on the CALVIN ABCD and CALVIN ABC benchmarks using 2, 4, 6, and 8 experts. For comparison, we include two dense MoDE baselines: Dense-small and Dense-large. Dense-small shares the same latent dimensionality as MoDE, while Dense-large is scaled up to 2024 dimensions, matching MoDE's overall parameter count. Our analysis focuses on how scaling affects both general performance (CALVIN ABCD) and zero-shot generalization (CALVIN ABC). In the ABCD environment, MoDE with 4 experts achieves the best performance. Interestingly, increasing beyond 4 experts degrades performance, possibly due to overfitting or increased routing complexity. In the zero-shot generalization (ABC), MoDE with 4 experts still performs well. Notably, the Dense-small variant consistently underperforms across both tasks, underscoring the efficiency of the MoE architecture in utilizing parameters more effectively. The Dense-small variant consistently underperforms. Overall, MoDE demonstrates that it can achieve comparable or superior performance to dense transformer models while requiring fewer computational resources.

A.9 State-based Experiments

We conduct additional experiments with MoDE on two established multi-task state-based environments:

Relay Kitchen. We utilize the Franka Kitchen environment from [26] to evaluate models. This virtual kitchen environment allows human participants to manipulate seven objects using a VR interface: a kettle, a microwave, a sliding door, a hinged door, a light switch, and two burners. The resulting dataset consists of 566 demonstrations collected by the original researchers, where each participant performed four predetermined manipulation tasks per episode. The Franka Emika Panda robot is controlled via a 9-dimensional action space representing the robot's joint and end-effector positions.

The 30-dimensional observation space contains information about the current state of the relevant objects in the environment. As a desired goal state, we randomly sample future states as a desired goal to reach.

For this experiment, we train all models for 40k training steps with a batch size of 1024 and evaluate them 100 times as done in prior work [27, 28, 4] to guarantee a fair evaluation. All reported results are averaged over 4 seeds. We train our models on a local PC RTX with an RTX 3070 GPU for approx. 2 hours for each run with the additional experimental rollouts.

Block Push. The PyBullet environment features an XArm robot tasked with pushing two blocks into two square targets within a plane. The desired order of pushing the blocks and the specific block-target combinations are sampled from the set of 1000 demonstrations as a desired goal state. The demonstrations used for training our models were collected using a hard-coded controller that selects a block to push first and independently chooses a target for that block. After pushing the first block to a target, the controller pushes the second block to the remaining target. This approach results in four possible modes of behavior, with additional stochasticity arising from the various ways of pushing a block into a target. The models only get a credit, if the blocks have been pushed in the correct target position and order. We consider a block successfully pushed if its center is within 0.05 units of a target square.

All models were trained on a dataset of 1000 controller-generated demonstrations under these randomized conditions. All models have been trained for 60k steps with a batch size of 1024. To evaluate them we follow prior work [27, 28, 4] and test them on 100 different instructions and report the average result over 4 seeds. We train our models on a local PC RTX 3070 GPU for approx. 3 hours for each run with a final evaluation. Demonstrations are sourced from a scripted oracle, which first pushes a randomly chosen block to a selected square, followed by the other block to a different square. The policies are conditioned to push the blocks in the desired configuration using a goal state-vector. We chose an action sequence length of 1 given a history length of 4 for these experiments, which are inspired by our dense diffusion transformer baseline BESO [4].

	Block Push	Relay Kitchen
C-BeT	0.87±(0.07)	3.09±(0.12)
VQ-BeT	0.87±(0.02)	3.78±(0.04)
BESO	0.96±(0.02)	3.73±(0.05)
MoDE	**0.97**±(**0.01**)	**3.79**±(**0.02**)

Table 5: Comparison of the performance of different policies on the state-based goal-conditioned relay-kitchen and block-push environment averaged over 4 seeds. MoDE outperforms the dense transformer variant BESO and other policy representations on all baselines.

Baselines. In this setting, we compare MoDE against several SOTA goal-conditioned policies. We test two transformer architectures, C-BeT [28] and VQ-BeT [29], that predict discretized actions with an offset. C-BeT uses k-means clustering together with an offset vector while VQ-BeT leverages residual Vector Quantization to embed actions into a hierarchical latent space. Further, we test against a dense diffusion policy transformer model BESO [4]. BESO uses the same continuous-time diffusion policy combined with a dense transformer to predict a single action given a sequence of prior states. To enable a fair comparison, we chose the same hyperparameters for BESO and MoDE in both settings. We test all models averaged over 4 seeds and report the mean values directly from prior work [29].

Results. The results of both experiments are summarized in Table 5. MoDE achieves a new SOTA performance on both benchmarks and outperforms the dense transformer variant of BESO in both settings. Further, MoDE achieves higher performance compared to other policy representation methods such as VQ-BeT and C-BeT.

A.10 Related Work

B Related Work

Diffusion in Robotics. In recent years, Diffusion Models [30, 1, 10] have gained widespread adoption in the context of robotics. They are used as a policy representation for Imitation Learning [5, 4, 31, 32, 33, 34, 35] and in Offline Reinforcement Learning [36, 37, 38]. Other applications of diffusion models in robotics include robot design generation [39], video-generation [40, 41, 42] and motion planning [43, 44]. The most common architecture for using diffusion models as a policy in robotics is a CNN with additional FiLM conditioning [24] to guide the generation based on context information. Recently, the transformer architecture has been adopted as a strong alternative backbone for diffusion policies, specifically in IL. Examples include Octo [3], BESO [4] and 3D-Diffusion-Actor [32]. However, no prior work considers using a Mixture of Experts architecture for improving the computational efficiency and inference time and solely relies on dense transformer architectures.

Mixture-of-Experts. MoE are a class of models where information is selectively routed through the model. The modern version of MoE was introduced in [45], where a routing or gating network conditionally chooses a subset of experts to send an input to. After Transformers [14] proved to be an effective model that scales well with data, they were modified to have expert feed-forward networks at each block of the model in [15] which presented Switch Transformers. Switch Transformers laid the groundwork that is still widely adopted in different Large-Language-Models (LLM) [46, 47]. This allowed for more total parameters while keeping the forward and backward FLOPs smaller than their dense counterpart, thus yielding significant performance gains. However, training both the router and experts in parallel is a non-trivial optimization problem, and it can yield suboptimal solutions such as expert collapse where experts learn similar functions instead of specializing [48]. In addition, router collapse occurs when the router selects a small subset of the experts and doesn't utilize all the experts. This is mitigated with load balancing losses [45, 15] which encourage the router to distribute inputs more evenly across experts. Multiple works have explored different methods to perform routing, such as expert choice routing [17], differential k-selection [49], frozen hashing functions [50], and linear assignment [51].

In the context of robotics, MoE models are used in many settings without being combined with a transformer architecture. Several works use a mixture of small MLP policies, that focus on different skills in Reinforcement Learning [52, 53, 54] or for robot motion generation[55, 56], another body of work utilizes combinations of small CNNs robot perception [57, 58]. Further applications include learning multimodal behavior using a mixture of Gaussian policies [59, 60]. Despite the extensive usage of MoE in many domains, no prior work has tried to utilize MoE together with Diffusion Policies for scalable and more efficient Diffusion Policies.

Multi Task Learning in Diffusion Models. It has been shown that the denoising process is multi-task [7]. Leveraging this idea, works have taken architectures that are suited for multi-task learning. Some works have explicitly scheduled which parameters are specialized to which stage in the denoising process [61, 62]. In extension to this [63] uses the scheduling as guidance during training but also learns how to modulate representations based on the denoising stage. Finally, some works have employed different architectures based on the denoising stage [64].

Transformers for Robot Learning. Transformer models have become the standard network architecture for many end-to-end robot learning policies in the last few years. They have been combined with different policy representations in the context of IL. One area of research focuses on generating sequences of actions with Variational Autoencoder (VAE) models [65, 66]. These action-chunking transformer models typically use an encoder-decoder transformer as a policy architecture. Several Diffusion Policies, such as Octo [3], BESO [4], ChainedDiffuser [31] and 3D-Diffusion-Actor leverage a transformer model as a policy backbone. Another direction of research treats behavior generation as discrete next-token predictions similar to auto-regressive language generation [67]. C-Bet, RT-1, and RT-2 use discretized action binning to divide seen actions into k-classes [28, 27, 68, 69], while VQ-BeT [29] learns latent actions with residual Vector Quantization. Several works have shown the advantages of using pre-trained LLM or VLM as a policy backbone, which are then finetuned for action generation [70, 71, 72, 18]. None of the recent work considers using any Mixture-of-Expert architecture for policy learning. MoDE is the first architecture to leverage MoE architecture combined with diffusion for behavior generation.

B.1 Limitations

MoDE still has certain limitations. In our experiments, we find that MoDE exhibits a slightly higher standard deviation compared to the baselines. We hypothesize that the router's initialization has significant impact on overall optimization, requiring future work on stabilizing routing models. In addition, when visualizing expert utilization, in some of our experiments we noticed that only a subset of the total experts were being utilized - a phenomenon known as expert collapse [48]. In addition to load balancing regularization, having more inductive biases that encourage the router to fully utilize all experts are needed.