



Research Article

Beyond performance: Explaining generalisation failures of Robotic Foundation Models in industrial simulation

David Kube^{a,b,*}, Simon Hadwiger^{a,b}, Tobias Meisen^b^a Siemens AG, Nuremberg 90475, Germany^b Bergische Universität Wuppertal, Wuppertal 42119, Germany

ARTICLE INFO

Article history:

Received 3 April 2025

Revised 8 June 2025

Accepted 15 June 2025

Available online 9 July 2025

Keywords:

Explainability

Industrial robotics

Robotic Foundation Models

Reasoning visualisation

Zero-shot generalisation

ABSTRACT

This study investigates the generalisation and explainability challenges of Robotic Foundation Models (RFMs) in industrial applications, using Octo as a representative case study. Motivated by the scarcity of domain-specific data and the need for safe evaluation environments, we adopt a simulation-first approach: instead of transitioning from simulation to real-world scenarios, we aim to adapt real-world-trained RFMs to synthetic, simulated environments – a critical step towards their safe and effective industrial deployment. While Octo promises zero-shot generalisation, our experiments reveal significant performance degradation when applied in simulation, despite minimal task and observation domain shifts. To explain this behaviour, we introduce a modified Grad-CAM technique that enables insight into Octo's internal reasoning and focus areas. Our results highlight key limitations in Octo's visual generalisation and language grounding capabilities under distribution shifts. We further identify architectural and benchmarking challenges across the broader RFM landscape. Based on our findings, we propose concrete guidelines for future RFM development, with an emphasis on explainability, modularity, and robust benchmarking – critical enablers for applying RFMs in safety-critical and data-scarce industrial environments.

© 2025 The Author(s). Published by Elsevier B.V. on behalf of Shandong University. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Most efforts in developing generalist or foundation models in robotics have concentrated on table-top or kitchen tasks. This focus likely stems from data scarcity in robotics, wherein the largest publicly available robotics dataset to date is the Open-X dataset [1], featuring real-world trajectories within exactly these domains. As noted by Hu et al. [2], generalisation and data scarcity pose significant challenges for general-purpose robots. However, when transitioning towards industrial relevant applications, the issue of data scarcity becomes even more pronounced. Moreover, in industrial applications, it is almost invariably essential to have safe environments to evaluate models before they are deployed in real-world scenarios. Untested control models could potentially cause damage, and testing in real-world conditions is often tedious and less effective compared to simulation. Therefore, a simulation environment frequently becomes mandatory, even if the intention is to train such models with real-world, manually created datasets. To explore the current capabilities of RFMs within industrial contexts and acknowledging the scarcity of data, our objective is to employ an existing RFM that performs well within its domain and, with minimal effort, adapt it for industrial

applications. We selected Octo [3] as the pivotal model for our study, with this choice elaborated in Section 2.1. Due to the aforementioned data scarcity and to minimise additional training efforts, we utilise synthetic data from a simulation environment and focus on training only Octo's action-predicting component, the action-head. We assume the Octo-transformer has acquired sufficient generalisation capabilities from its training on a large dataset to effectively handle synthetic images within table-top domains, given the premise, that the dataset is diverse enough to perceive synthetic images as just another variant within the encountered distribution. This assumption is supported by Octo's claim to outperform state-of-the-art models like RT-2X [1] in zero-shot scenarios. To enable training with simulated images while keeping manual efforts to a minimum, we chose not to manually create datasets in the simulation. Instead, we adapted the action-head and model flow to be trainable with SAC Reinforcement Learning (RL) within the simulation, further enhancing future applicability to industrial contexts. Before transitioning to industrial use-cases, we implement a straightforward real-to-simulation transfer, involving simple table-top tasks, akin to those found in the Open-X dataset but conducted within simulation instead of real-world settings. This ensures, that initially, the model only encounters slight observational distribution shifts while maintaining task similarity. Despite expectations of success, Octo exhibited significant challenges in adapting to those

* Corresponding author.

E-mail address: david.kube@siemens.com (D. Kube).

simulated settings. Given that both, the simple real-to-simulation transfer and our extensive training approaches, failed to produce satisfactory performance from Octo in simulation, it became imperative to understand and explain the underlying reasons. However, interpreting Octo's reasoning proved challenging, given its nature as a non-interpretible black-box. Consequently, we investigated methods to illuminate the internal workings and reasoning processes of Octo, and RFMs in general. We propose a slight modification of Grad-CAM and apply this adapted method to ascertain, in detail, why Octo struggles in simulated environments. Our approach even allows us to distinguish the specific factors affecting Octo-small versus Octo-base, explaining why they struggle in simulation. Even though they exhibit very similar behaviour when exposed to such observational distribution shifts, when no explainability-promoting methods are applied. Throughout our experiments, additional difficulties and discrepancies led us to delve deeper into various aspects of Octo, including its architecture as well as modularity, design choices, benchmarking, evaluation methods, examination of generalisation capabilities and explainability of RFMs in general.

Thus, our primary contributions encompass the examination of potential pitfalls and challenges in RFM research, alongside offering guidelines for future RFM research to circumvent these issues, with a focus on:

- Empirical evaluation of the Octo RFM's generalisation performance under real-to-simulation transfer, towards industrial-style tasks.
- Demonstrating patterns of failure within simulated environments, despite minimal task shifts – revealing critical limitations in current RFM generalisation.
- Introducing a modification to the Grad-CAM [4] method for internal reasoning visualisation in RFMs, applicable to arbitrary model output.
- Demonstrating the critical importance of explainability in RFMs, underscored by a systematic reasoning analysis across input modalities, highlighting shortcomings in Octo's interpretability under distribution shifts.
- Practical design insights and guidelines for future RFM architectures, emphasising modularity, interpretability, and robustness to domain shifts.
- Critical review of benchmarking practices, offering recommendations for more meaningful RFM evaluation, relevant to industrial contexts.

2. Background and related works

To underpin our arguments and establish a foundation for further discussion, we will offer an overview over RFMs in Section 2.1, present benchmarks commonly employed to evaluate such models in Section 2.2, detail fundamentals to our work in Sections 2.3 & 2.4, and finally review further publications related to our research in Section 2.5.

2.1. Robotic foundation models

Foundation models, in general, are trained on extensive datasets with the objective of not only generalising over these distributions but also cultivating new, emergent behaviours that are not explicitly defined. This is intended to enable seamless adaptation to numerous downstream tasks with minimal effort. [5] RFMs are foundation models specifically tailored for robotics, primarily focusing on low-level control, though they can also be utilised for high-level planning [6]. Due to their generalist nature, task specification is crucial for effective task execution, which can be accomplished through various means such as language prompts or goal images [2]. Such generalist models for

robotic use-cases are either fine-tuned or applied zero-shot for specific, low-level control tasks. To get an overview over existing robotic models that may fall into this category, and decide on the pivot to base our study on, we have performed an extensive literature research in 2024. At the date, we sighted a total of roughly 2000 papers of various categories, including about 50 promising models and architectures further considered as baseline. While most RFMs exhibit common traits, including vision input, low-level action prediction and an application focus on table-top or kitchen tasks, we observe substantial variations in their claimed usability, flexibility, modularity, action and observation spaces, and generalisation capability. For instance, RVT [7], PerAct [8], PolarNet [9], GNFactor [10] and Hiveformer [11] all rely on RGB-D input and are specifically targeted to consume depth or 3D-data. While this may result in a performance advantage over RGB input, due to increased precision and dimensionality of environment representation, those models are less applicable to be used within industrial use-cases, where hardware costs and inference times should be reduced to a minimum. PerAct, PolarNet and Hiveformer even rely on *several* RGB-D sensors, which further reduces potential inference throughput and increases required hardware, thus multiplies cost for each robot the given model should be applied to. Besides input, e.g. PaLM-E [12] only outputs mid- to high-level plans rather than executable actions and is thus not applicable to control the robot directly. RFMs should have the capability to be directed on what task to solve, ideally through natural language to aid flexibility, as demonstrated by Octo [3] or PLEX [13]. Other models are bound by given tasks or prompting schemes. For instance, SayCan [14] maps language instructions to a fixed set of tasks, CLIPort [15] confines the model to pick-and-place operations, and MOO [16] requires language instructions to conform to a set of specific templates. Meanwhile, some models, like DiffusionPolicy [17], do not support prompting at all.

Besides all mentioned differences, many claim very similar appealing traits. Those claims are especially prevalent in a plethora of further, recently published and promising RFMs, like Octo [3], Gato [18], PLEX [13], RoboAgent [19], RT-1 [20], RT-2 [21], BAKU [22], CAGE [23], OpenVLA [24], QueST [25], RDT-1B [26], CrossFormer [27], RoboDual [28], $\pi 0$ [29], Figure AI's Helix [30] or Google DeepMind's Gemini Robotics [31], among others:

Modularity: Many of those claim to be highly flexible or modular in adaption to observation-spaces [3,18,26,27], action-spaces [3,18,26,27,29], task definition [3,18,27], or require none to minimal fine-tuning [3,13,18,19,24,26–29] after adapting the model to specific, novel use-cases.

Generalisation Capabilities: Additionally, a fundamental claim is generalisation, especially over kinematics [3,18,26,27,29], goal specifications [3,18,19,21,22,24,26–29], observation distribution shifts [3,18–24,26–29] and novel tasks [3,13,18–29].

Performance: Most importantly, almost every model claims to outperform its contemporaries across some or all tasks, despite the fact that many of these models have been published in a closely packed time sequence, illustrating the rapidly evolving nature of foundation models in robotics. In summary, Table 1 summarises which publications' recent RFM claims to set the state of the art by surpassing the subsequent baselines. These claims collectively illustrate the competitive advancements within the domain of RFMs. However, do all these assertions withstand scrutiny? While benchmark results presented in these publications appear promising, verifying emergent behaviour remains a challenge, complicating the task of determining whether such benchmarks are sufficient for making informed decisions, such as selecting the optimal architecture for specific applications. To tackle these questions, we initially selected a model available at the time of our research to serve as a case-study target, based

Table 1
Comparing RFMs' claimed superiority.

Recent RFM publications	Baseline
Octo [3] ²⁰²⁴	> RT-1X [1] ²⁰²³ , RT-2X [1] ²⁰²³ , VC-1 [32] ²⁰²³
RoboAgent [19] ²⁰²³	> CACTI [33] ²⁰²³ , RT-1 [20] ²⁰²³ , BeT [34] ²⁰²²
RT-1 [20] ²⁰²³	> Gato [18] ²⁰²² , BC-Z [35] ²⁰²² , SayCAN [14] ²⁰²²
RT-2 [21] ²⁰²³	> MOO [16] ²⁰²³ , RT-1 [20] ²⁰²³ , VC-1 [32] ²⁰²³ , R3M [36] ²⁰²² , BC-Z [35] ²⁰²² , LAVA [37] ²⁰²²
BAKU [22] ²⁰²⁴	> RT-1 [20] ²⁰²³ , RoboAgent [19] ²⁰²³
CAGE [23] ²⁰²⁴	> DiffusionPolicy [17] ²⁰²³ , RISE [38] ²⁰²⁴
OpenVLA [24] ²⁰²⁴	> RT-1X [1] ²⁰²³ , Octo [3] ²⁰²⁴ , RT-2X [1] ²⁰²³ , DiffusionPolicy [17] ²⁰²³
QueST [25] ²⁰²⁴	> PRISE [39] ²⁰²⁴ , ACT [40] ²⁰²³ , DiffusionPolicy [17] ²⁰²³ , ResNet-T [41] ²⁰²³ , VQ-BeT [42] ²⁰²⁴
RDT-1B [26] ²⁰²⁴	> ACT [40] ²⁰²³ , Octo [3] ²⁰²⁴ , OpenVLA [24] ²⁰²⁴
CrossFormer [27] ²⁰²⁴	> Yang et al. [43] ²⁰²⁴ , Octo [3] ²⁰²⁴ , ViNT [44] ²⁰²³
RoboDual [28] ²⁰²⁵	> Octo [3] ²⁰²⁴ , RT-1 [20] ²⁰²³ , DiffusionPolicy [17] ²⁰²³ , many others
$\pi 0$ [29] ²⁰²⁴	> OpenVLA [24] ²⁰²⁴ , DiffusionPolicy [17] ²⁰²³ , ACT [40] ²⁰²² , Octo [3] ²⁰²⁴

on its *claims*. During the decision-making process, the availability of code and model weights was a significant factor, ultimately narrowing our options.

This led us to choose Octo [3] as the pivot point for our study, noting that many of the previously mentioned models have not been published at the date making this decision. Octo claims to deliver a highly modular end-to-end architecture, enabling customisation of observation and action spaces, as well as independent modification and replacement of action-heads, without the need for comprehensive model tuning. Their modular input definition allows for up to two RGB-images, proprioception, or combinations thereof as inputs. It can be prompted with either goal images or natural language, providing flexibility to choose among the digested task distribution. Input modalities are encoded and tokenised separately, e.g. by using shallow CNNs for image encoding. Those tokens are subsequently fed into their main contribution, the Octo transformer, which integrates all of those modalities to provide readout tokens for the action-head. As this model is trained on a diverse robotic dataset – encompassing various tasks, camera angles and scene-setups – they claim promising generalisation capabilities. Finally, the action-head utilises readout tokens produced by the Octo transformer to predict low-level actions for direct robot control. As an addition, their complete code-base and two pre-trained Octo models (small and base) are publicly available, which builds a solid foundation for our experimentation.[3]

2.2. RFM benchmarks

To compare themselves with each other, the RFMs outlined in Section 2.1 utilise benchmarks such as Meta-World [45], Robosuite [46], LIBERO [41], Deepmind Control Suite [47], CALVIN [48], FMB [49], RLBench [50] or custom evaluation setups. Each of these benchmarks provides unique environments that challenge RFMs in different aspects of robotic control and behaviour. For instance, Meta-World [45] provides 50 fixed robotic manipulation tasks, such as “sweep”, “pull”, “pick” or “place” in a very simplistic simulated environment. In contrast, Robosuite [46] emphasises modularity and customisability supporting ten different robot models within realistic simulated table-top environments. LIBERO [41] aims to benchmark lifelong learning algorithms, and provides four task suites, namely LIBERO-Spatial, LIBERO-Object, LIBERO-Goal and LIBERO-100, containing 130 tasks in total. Meanwhile, the Deepmind Control Suite [47] provides a set of tasks including reward signals, intended to bench RL agents performing various continuous control tasks, mostly detached from

robotic use-cases. CALVIN [48] tests ability to execute language-conditioned behaviours for long-horizon tasks. Much like our configuration, they employ the PyBullet physics engine, utilising similar objects and visual representations to those used in our experiments. Additionally, RLBench [50] offers a suite of 100 hand-crafted table-top tasks designed to reflect varying difficulty setups. Custom evaluation tasks, as used by RT-1 [20], RT-2 [21], RDT-1B [26] or $\pi 0$ [29], mostly centre around kitchen or table-top applications and further offer tailored scenarios that challenge specific capabilities of RFMs, but may lack comparability to other works.

2.3. Visualising internal model reasoning

Although no significant attempts to visualise internal reasoning and to promote explainability for RFMs specifically have been undertaken, there is a multitude of such approaches for models of other domains, like image classification [4,51–57] or for CNNs in general [58–61]. Those methods produce saliency maps that can deliver straightforward and easily comprehensible representations, facilitating interpretability, ensuring transparency, aiding analytics and assisting in debugging, which is especially beneficial when diagnosing unexpected model behaviour. We argue, that such representations are equally important for RFMs, especially during model crafting and comparison of model accuracy or applicability when exposed to observational distribution shifts and could be a potent tool besides benchmarks alone, as presented for similar domains like Atari agents in [62,63]. Many of such methods, for classification, rely on Class Activation Maps (CAMs), which reveal the specific regions in the input image that are leveraged by the CNN to identify a given class [51]. When interested in features of a given category, this often requires to manually set the class of interest for gradient calculation, as in [4]. Additionally, in [51], the impact of distinct layers cannot be exposed directly, since individual model’s parameters impact on the saliency map is not explicitly visualised, whereas propagation-based methods like [4], use gradients, e.g. of a class, to further weigh model decisions, based on CNN-subsequent parameter influence. To illustrate, Grad-CAM [4] produces heatmaps that are not only influenced by the activation A of the CNN-produced feature map k and the class of interest c , but are grounded on neuron importance weights retrieved by the gradient of c , with respect to A^k :

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k} \quad (1)$$

2.4. Reinforcement Learning and Soft Actor-Critic

Since RL and specifically the SAC algorithm are integral to our work, we hereby offer a brief summary of their key elements.

RL: In essence, RL enables an agent to learn from interaction with a given environment. The goal of this interaction, as depicted in Fig. 1, is to find an optimal mapping from state to action with the highest value for the given task. To find an optimal policy that reflects this mapping, the environment returns a rating of the action, the reward r_{t+1} , alongside the updated state representation s_{t+1} : $(s_t, a_t) \mapsto (s_{t+1}, r_{t+1})$, at each given time step t , after the current action a_t is applied in state s_t . The reward is returned from a defined function, the reward signal, which maps s_t and a_t to a reward r_{t+1} : $(s_t, a_t) \mapsto r_{t+1}$, $r_t \in \mathbb{R} \forall t$. At any arbitrary environment state s_t , reaching the goal, implicitly defined by the given reward signal, can thus be optimised by applying the action sequence that yields the highest future reward, the *return* G_t , $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$. The discount factor γ , $\gamma \in [0; 1]$ is applied to prioritise immediate over future reward. [64,65]

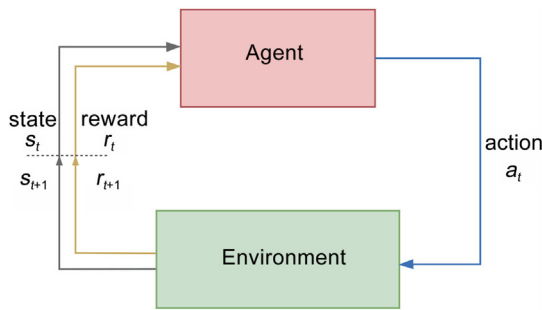


Fig. 1. Agent-Environment Interface (based on [64]).

SAC [66]: Is an off-policy, Policy Gradient (PG), Actor-Critic (AC), RL algorithm. It uses parametrised behaviour and target actor policies in conjunction with a soft value-function approximation, the critic, to utilise bootstrapping. [64–66] Additionally, a scaled expected entropy term is added to the sum of expected rewards, to increase stability and exploration of the training procedure [66]. Since actors and critics are parametrised, they can be represented by any arbitrary Machine Learning (ML) architecture, ranging from simple neural networks to more complex and sophisticated variants.

2.5. Comparable studies

Few approaches have been undertaken that explicitly focus on evaluating the ability of recent RFMs to cope with domain shifts and validate their purported generalisation capabilities. However, two notable efforts deserve mention: Firstly, ReVLA [67] compared Out-Of-Distribution (OOD) generalisation capabilities among three models: RT-1 [20], OpenVLA [24], and most notably, Octo [3]. Their findings highlighted that these vanilla models, particularly Octo, considerably underperform, casting doubt on their claimed effectiveness. ReVLA suggests that measures aimed at countering catastrophic forgetting in vision encoders might enhance OOD performance. Moreover, they also conclude that default benchmarks may exhibit limitations for RFM evaluation, particularly concerning inter-domain gaps in visual representations and differences in control outcomes, which led them to extend the SIMPLER benchmark. [67] SIMPLER [68] on the other hand, claims to serve as a scalable and reliable proxy for real-world evaluation in simulation, aiming to minimise these identified gaps. Within their simulated benchmark that focuses on reducing distribution-shifts to real-world data, they assessed Octo and also displayed a slight gap between Octo-base’s performance within this simulated benchmark and real-world application. [68] Nevertheless, they rather try to bench models as close to their known distributions as possible, thus do not adequately address the concrete problems models may face in OOD scenarios, nor do their evaluation techniques pinpoint why distribution shifts could pose difficulties to concrete models.

Besides directly evaluating the focus of domain shifts, Li et al. [69] endeavour to identify and explain the key factors that notably affect the performance of Vision-Language-Action Models (VLAMs), essentially Vision-Language Models (VLMs) specifically tailored to predict actions, thus a category most RFMs fall into. Similar to our assertion that incorporating synthetic data from diverse sources into the training set of RFMs is crucial for promoting generalisation, Li et al. [69] discover that pre-training of VLAMs with cross-embodiment data, followed by in-domain fine-tuning strengthens their ability to generalise effectively. Beyond their studies on cross-embodiment data, they concentrate on two main points: Firstly, extracting patterns from current VLAMs, with little

focus on enhancing or improving these models to solve emerging problems. Secondly, identifying existing backbones that can be integrated into these architectures to maximise success rates. [69] Their overall strategy seeks to enhance performance using current solutions but falls short to address future challenges. Through our research, we reveal that overcoming OOD shifts extends beyond catastrophic forgetting – a point noted by ReVLA [67]. In contrast to SIMPLER [67], which primarily seeks to minimise distribution shift during evaluation, we advocate for explicitly presenting these elements to clearly illustrate OOD performance.

3. Methods

The subsequent sections will provide detailed insights into our methodologies, serving as foundation for experiment execution as outlined in Section 4. This encompasses modifications to architecture and model flow we implemented in Octo, as well as training parameters and the environment setup used to train and apply models in simulation. To verify correctness of this implementation, we also define a privileged training setup. Finally, we detail our approach to explain model reasoning by highlighting the regions of visual focus in input images, leveraging the Grad-CAM [4] methodology.

3.1. Simulation and environment setup

To tailor the Octo architecture for RL training, as described in Section 3.3, and thereby facilitate the use of simulation environments for training with synthetic data and secure testing, it is essential to provide a simulation environment. This environment must enable the model to interact, collect simulated data for observation and implement diverse action-space representations drawn from the model for robot control. To fulfil those experimental requirements, train the adapted Octo architecture in various setups, test the performance of our trained models and verify Octo’s zero-shot capability when transitioning from real-world to simulated conditions, we utilise the Bullet real-time physics simulation [70] in Python. Its lightweight nature, multi-platform support and straightforward Python API enable rapid prototyping and experimentation as required by our research. Although we acknowledge that the basic nature of PyBullet could result in a more pronounced sim-to-real gap than more realistic counterparts like Isaac-Sim, all simulation environments exhibit such gaps inherently. As we strive to showcase these specific influences, we consider PyBullet to be the optimal choice. By focusing on simple table-top tasks aligning with Octo’s training distribution, utilising relatively low-resolution images, various camera angles, and diverse object types, we further seek to minimise influences on generalisation intrinsic to our chosen simulation. This methodology aims to highlight the impacts of transferring into the realm of simulation in general. We integrate a UR5e collaborative robot arm, paired with a Robotiq Hand-E gripper, and two distinct scene-setups to provide a testbed for our methodology. To replicate Octo’s camera setup and mimic camera view angles similar to the training distribution, we use two synthetic RGB-cameras. The primary camera (Fig. 2a) provides an overview of the entire scene with a resolution of 256×256 pixels, while a wrist-mounted secondary camera captures images with a resolution of 128×128 pixels (Fig. 2b). We apply two different scene-setups to facilitate straightforward table-top tasks, akin to those found in the Open-X dataset [1], as Octo’s training data form a subset of this dataset, allowing us to minimise distribution shifts. Each setup is designed to allow the model to tackle basic tasks that demand specific visual skills, thereby mitigating the risk of selecting object traits that may be underrepresented in the training set or are challenging for the

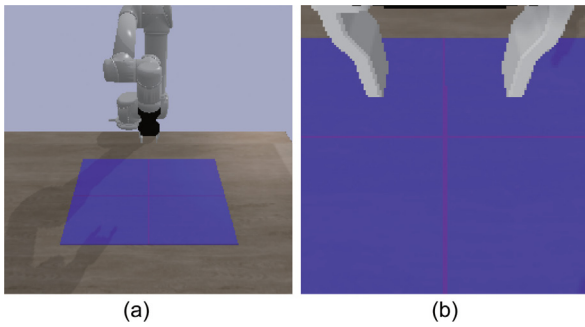


Fig. 2. Camera setup and view angles used as model input. The spawning area, as shown in Fig. 4, is included for reference only. (a) Primary camera view. (b) Wrist camera view.

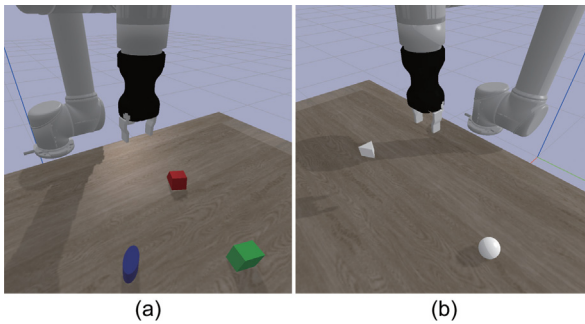


Fig. 3. Debug view of applied table-top scene setups. (a) Colour-based. (b) Shape-based.

model to differentiate. In the first configuration, we load two to three objects with distinct colours, as demonstrated in Fig. 3a, to emphasise visual colour understanding. Conversely, the second setup, illustrated in Fig. 3b, concentrates on shape distinction capabilities by eliminating colour and incorporating a sphere and triangular object. To increase initial state variation, all objects are positioned randomly within a specified spawning area, illustrated in Fig. 4, regardless of the scene setup. To ensure differentiation, a minimum 2D Euclidean distance (XY) of 15 cm is kept between objects. Additionally, to prevent potential immediate success for specific task scenarios, we ensure a 2D distance of at least 10 cm to the Tool Center Point (TCP)'s position. The robot arm is positioned in its initial pose as shown in Fig. 2a, at the beginning of each episode. We then subsequently rotate each joint of the robot by a pseudo-random value within the range $[-10; 10]$ degrees. To ensure replicability and equity in agent comparisons, we seed each randomisation and simulation environment using predetermined values. Additionally, during evaluation, the robot's initial pose is not randomised and all objects are placed in predefined positions, promoting consistent performance evaluation.

3.2. Training configuration

Since each type of scene, as depicted in Fig. 3, contains different objects to evaluate distinct visual reasoning skills, we need to create individual natural language prompts describing the task so be solved for both scenes. Our task specifications are simple, aiming to highlight the distinct feature of each scene. For the first scene configuration (Fig. 3a), instructions such as “Touch the <colour> object” focus solely on colour. Conversely, for the second scene setup (Fig. 3b), we use task descriptions like “Touch the <shape> object”, referring exclusively to the given objects' shapes. To assess Octo's comprehension of language, we provide different instructions in each training episode. For a given

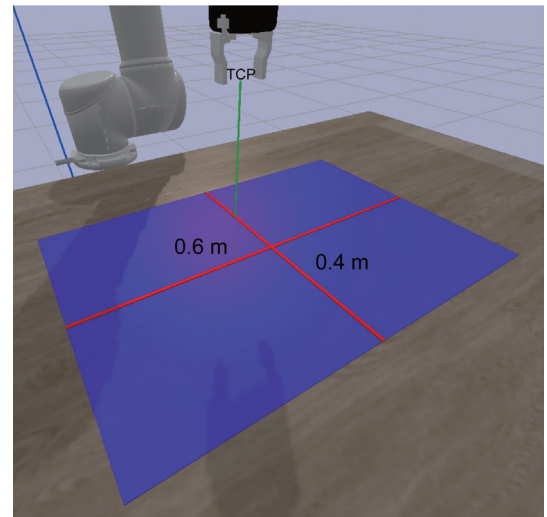


Fig. 4. Spawning area visualisation in debug view, applied to both scene-setups.

Table 2
Termination reasons and conditions.

Terminate reason	Condition
Success	Eq. (2)
Timeout	$t \geq T$

scene, we craft one instruction for each object and randomly substitute them with another from the same set of instructions. Alongside one of such language prompts, the complete observation space comprises two images, denoted as primary and wrist, as outlined in Section 3.1. To address potential challenges in the model's interpretation of actions, we adopted two different action-space representations for our experiments: The first representation involves absolute angular velocities, forming a six-dimensional vector tailored for the UR5e robot arm. The second representation utilises a TCP pose delta, encompassing changes in X, Y, Z, roll, pitch, and yaw, which is applied through a homogeneous matrix transformation to compute the target TCP pose to be approached. Termination of each episode is handled by the environment as shown in Table 2.

Based on the objects' sizes, we consider an episode to be successful, as shown in Eq. (2), if the TCP is within a 6 cm radius from the origin of the object mentioned in the language instruction.

$$\text{Success}_t = \begin{cases} \text{True} & , \delta(p_{\text{tcp}}, p_{\text{object}}) \leq 0.06 \\ \text{False} & , \text{else} \end{cases} \quad (2)$$

For both scene definitions, we implement two task setups of varying difficulty, where similar reward formulations apply: The first setup, utilising the language instructions “Touch/Grab object <X>”, is treated as a 3D problem. Consequently, the success formulation detailed in Eq. (2) uses a 3D Euclidean distance as the measure of difference δ . For the second, less challenging setup with language instructions “Move towards/ Approach object <X>”, we consider the task as solved, when the TCP either touches or hovers above the target object. As a result, the difference measure δ takes into account only the distances along the X- and Y-axes. For both task setups (2D-hovering and 3D-reaching), we apply both scenes as depicted in Fig. 3, and formulate the reward R_t at each time step t according to the respective task setup's difference measure δ , penalising the distance to the object as follows:

$$R_t = -\delta(p_{\text{tcp}}, p_{\text{object}}), t \in \mathbb{N} \wedge t \leq T \quad (3)$$

At each episode's terminal time step T , we apply an additional final reward R_T to ensure, that a success outweighs any accumulated punishment received over the course of the episode, even when the trajectory towards reaching the desired object is not immediately comprehensible.:

$$R_T = \begin{cases} \mathbb{E}[\delta] \times T_{\max} \times 1.1 & , \text{if success} \\ 0 & , \text{else} \end{cases}, T \in \mathbb{N}^+ \quad (4)$$

Where $\mathbb{E}[\delta]$ denotes the expected mean euclidean distance from the robot's TCP to the object of interest's position during an episode. In most cases, this is sufficiently defined by the mean of the minimum and maximum possible distance between the robot's base link and the furthest starting position from the robot's base link while taking into account the kinematics and distance-based termination criteria. Since solving those simple tasks does not require time-awareness of our agent, and we want to keep complexity for the agent to a minimum, we do not include time to the agent's observation. Hence, as proposed by Pardo et al. [71], we do not punish timeout and also exclude this component from the reward formulation.

To incorporate all of those definitions and to leverage the existing Octo code-base, built on the JAX Python library, we created a custom implementation of SAC in JAX, building on stable baselines 3 and the SBX proof-of-concept implementation available for Python.

3.3. Architectural modifications

To efficiently train an action head within our simulation environment, designed to work alongside the Octo model and capable of processing Octo's embeddings for generating valid actions, we integrate several architectural changes to Octo and its components, with particular emphasis on the action head, and made modifications to the model's overall flow during training. Fig. 5 shows the continuous action-head that serves as baseline, while an overview of the mentioned flow adaption is shown in Fig. 8. The observation-action tuples for the critic are sampled from a replay buffer, which is excluded from the figure for simplicity and readability. The general concept is as follows: We remove the default action-head used by Octo and create two versions of heads, based on the continuous variant from [3]. The actor-head (Fig. 6) is used to predict actions to be executed by our robot in simulation. To aid exploration, we apply a stochastic actor during training by sampling from a Gaussian distribution. We let our actor-network decide on the degree of stochasticity as it directly predicts the log std used for sampling, while restricting this value to the range $[-5; -1.5]$. We empirically identified these values and set the lower bound to exclude insignificant stochasticity. This decision helps in avoiding excessively large log probabilities from the underlying density function, thereby preventing adverse impacts on the SAC algorithm's updates. Conversely, to prevent saturation effects in the hyperbolic tangent, the upper bound constrains excessive deviations in action-values that could result from sampling the Gaussian distribution with large values. After training and during evaluation, we apply the action-head deterministically by directly using the predicted action without applying noise. To obtain a valid action representation, the predicted action is consistently squashed to a predefined range suitable for the target robot before being returned from the head, regardless of its application. In order to accommodate those changes, we revised the network structure, including its input and output dimensions, and pursued shallower architectures to facilitate better integration with RL. To estimate the return given an observation-action tuple, we apply a second type of head, the critic, as shown in Fig. 7. For a given time step t , it concatenates the observed state representation obs_t and corresponding

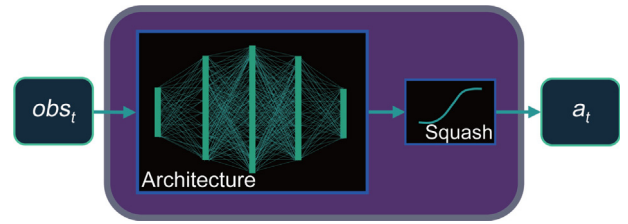


Fig. 5. Default continuous action head from Octo [3]: The hyperbolic tangent is used to squash the prediction to the given action-range.

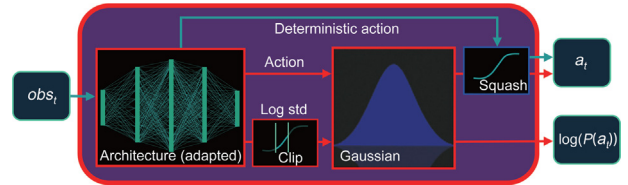


Fig. 6. Actor-head: Adaption of default version (Fig. 5): During training, a stochastic and exploratory actor is applied, while a deterministic and greedy actor exploits its knowledge during deployment.

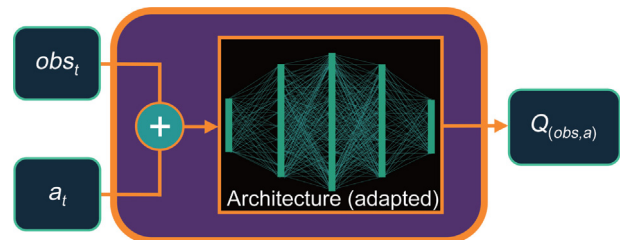


Fig. 7. Critic-head: Adaption of default version (Fig. 5): The transformer readout and the actor's action at a given time step are concatenated to estimate a Q-value.

actor-head's action a_t . The result is then fed into the critic's network to estimate a scalar Q-value for the state-action tuple (obs_t, a_t) . Aiming to limit overestimation as proposed by Hasselt [72], Fujimoto et al. [73], we apply two of those critic-heads at once during training, by using only the minimum estimation of both critics. The remaining architecture of Octo, including encoders and the transformer-stack, are frozen and essentially serve as a generalist feature extractor as shown in Fig. 8. Since Octo has been trained on a relatively large robotic dataset, we hypothesise it has developed sufficient generalisation capabilities to provide valid observations for our action-heads without the need to fine-tune its parameters. Therefore, both images and the task description in natural language, are propagated through Octo's respective encoders and transformer-stack to produce a readout vector. This reduced observation representation normally serves as input to Octo's diffusion action-head, however, in our case as input for both, the actor-head and the critic-heads. Following this procedure, we can train both heads variants in simulation using SAC and utilise the combination of Octo and the deterministic actor-head's option after training for inference to predict actions for task execution.

3.4. Privileged training setup

To validate correctness of our implementation, which includes the content of all previous sections of the methods Chapter 3 (i.e. any issues potentially impeding our training setup's functionality), we chose to perform a sanity check by removing the Octo-encoding and -transformer stack, as shown in Fig. 8, and

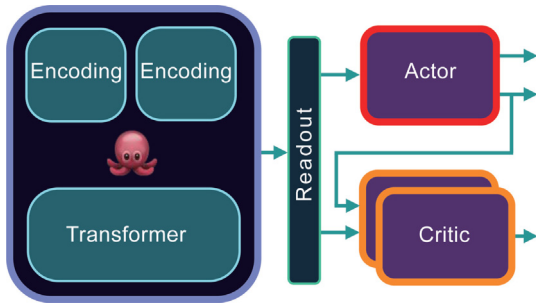


Fig. 8. Modification overview during training: The Octo architecture from [3] is split, removing the default action head. Instead, two modified types of heads, one serving as actor (Fig. 6) and the other type as critic (Fig. 7) are utilised to enable SAC training.

substitute its readout representation by ground-truth information from the simulation (Fig. 9). This observation then consists of the position-differences along the X, Y and Z axes only. Although this representation is in fact a huge simplification, this setup grants privileged actor- and critic-heads that therefore must be capable to solve the task, provided our own implementation does not contain any major errors. The remaining logic and network architectures remain the same and we test with a fixed subset of the shape-based scenario with the goal to move towards the sphere object.

3.5. Grad-CAM based reasoning visualisation

While robotic benchmarks provide valuable metrics, they alone do not paint a complete picture of emergent behaviours and reasoning within foundation models used in robotics. Thus, it is imperative to develop and apply additional methods that can offer deeper insights into model reasoning and enhance explainability. Since most models in this domain rely on RGB vision input, a potent tool during crafting, evaluation and subsequently as an addition to benchmarks, is to shed light on the model's internal visual reasoning behaviour, akin to the Grad-CAM [4] approach or derivatives for classification models. As stated in Section 2.3, such methods deliver straightforward an easily interpretable representations to visualise internal model reasoning. However, as detailed for Grad-CAM in Section 2, such methods often require adaption of the raw output with respect to a given class of interest before backpropagation. One of the challenges in utilising methods such as [4] for models that predict continuous or complex outputs, such as actions to be applied to a robot or embeddings extracted from images, is the difficulty in interpreting these outputs.

In contrast, studies involving discrete actions, such as [63], assume a well-trained actor model, allowing and requiring the use of discrete actions (e.g., left, right). These actions are easy to interpret, thereby facilitate straightforward adjustments to the discrete action output. In contrast to [4], where the output is easily guided based on a given class of interest, or [63], where discrete actions are treated as distinct classes, our approach subtly differs, by removing the necessity to interpret model outputs before backpropagation. Instead, building on [4], we directly use inference-generated outputs, whether from the actor or feature-extracting components, such as the frozen Octo stack in our case, to streamline the backpropagation process.

3.5.1. Algorithmic and mathematical foundations

Specifically, we execute an initial forward pass to capture the activation A^k of a feature map k from the last CNN layer for each input image. We then conduct a secondary forward pass

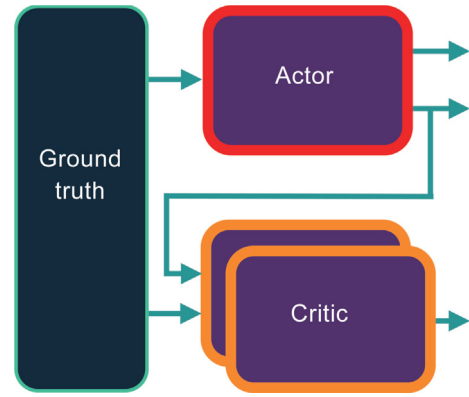


Fig. 9. Setup for privileged training: We kept all of our changes to the training setup and action heads, while substituting the frozen Octo (Fig. 8) with ground-truth input.

without the original images, using the captured feature maps as input to bypass the CNN layers. This second forward pass can propagate towards any chosen subsequent layer within the network, producing an output y of arbitrary dimension, depending on the model and the layer targeted. To provide a differentiable function for gradient calculation, we then condense y into a single scalar value s by considering each component equally¹ using an appropriate function $f(x)$, such as the mean or sum, resulting in $s = f(y)$. Finally, we compute the gradient of the prediction score, represented by the accumulated output s , with respect to the feature maps captured during the initial forward pass. To recapitulate, unlike Grad-CAM we do not derive the importance weights α_k^c using a specific class score y^c as shown in Eq. (1). Instead, our method relies on calculating the gradient of the scalar value s obtained by condensing the prediction y to calculate the neuron importance weights α_k , which are thus defined as follows:

$$\alpha_k = \frac{1}{Z} \sum_i \sum_j \frac{\partial f(y)}{\partial A_{ij}^k} \quad (5)$$

We thereafter proceed with the vanilla Grad-CAM algorithm [4]

$$L_{\text{Grad-CAM}} = \text{ReLU}\left(\sum_k \alpha_k A^k\right) \quad (6)$$

but finally normalise the output for better differentiation of values, thus increased visualisation quality

$$L_{\text{Grad-CAM norm}}(i, j) = \frac{L_{\text{Grad-CAM}}(i, j) - \min_{i,j}(L_{\text{Grad-CAM}})}{\max_{i,j}(L_{\text{Grad-CAM}}) - \min_{i,j}(L_{\text{Grad-CAM}})}, \quad \forall (i, j) \in A_k \quad (7)$$

We expect this procedure to visualise the *general* model focus, indicating which objects in the scene are pertinent for producing the output y , without relying on manually adjusting y based on specific predefined classes or action interpretations. Such flexibility means that, theoretically, any layer's output can be used to calculate importance weights, making it possible to observe the influence of each individual layer. To implement our approach, we typically assume an architecture containing initial CNN layers, as required by [4], or any methods that yield representations which we can interpret to generate saliency maps. These methods or layers generate the feature maps that we capture during the initial forward pass and are subsequently bypassed in the

¹ Our experiments revealed, that giving equal consideration produces the most effective results in visualising the general model focus.

second forward pass. Therefore, the only required adaption to the model architecture, is to provide an optional input for the feature maps to bypass said layers. Our methodology supports two complementary strategies for visualising focus: Initially, to employ those feature maps directly, we assign uniform importance weights of one, instead of utilising their gradient-derived variants α_k , allowing us to visualise the CNN's general focus in absence of the subsequent architecture. Secondly, through the proposed adaptations, we seek to determine weights that reflect the model's general focus and positively influence its concrete decision. Those weights enable adjustments to the feature map to accurately identify significant areas within the image necessary for generating the specific output of arbitrary type, including embeddings and continuous actions also in domains like RL.

3.5.2. Data

To explore Octo's focus on both, real and synthetic images, and to examine its generalisation capability to synthetic imagery, we employed data from various sources, acting as a foundational anchor in our study.

Synthetic Data: Firstly, we obtain images from our simulation environment, as described in Section 3.1, capturing both primary and wrist camera view angles, as shown in Fig. 2. These simulated images are crucial to assess internal model focus and reasoning, when shifting to synthetic scenarios.

Real-World Image Data: Secondly, we explore images from two trajectories sourced from the Bridge dataset [74]. The choice of this dataset is driven by its significant contribution to Octo's training data and we carefully selected the following trajectories to address specific priorities and scenarios:

- Presented within a zero-shot example Python script from Octo [3], the first trajectory comprises solely primary images and is characterised by a sparse scene configuration as depicted in Fig. 10.
- To complement the first trajectory, we include a second one captured in a cluttered environment, featuring both primary and wrist view angles as shown in Fig. 11. This choice introduces diverse complexities, facilitating a comprehensive model assessment and enriching our analysis diversity.

Through these collected data sources, we aim to provide extensive insights into the model's ability to interpret and prioritise various elements across different observational environments and complexities.

3.5.3. Specific application details

To apply this procedure to Octo, we capture the feature map activation A_k immediately after the final CNN layer of the SmallStem network, but prior to the tokenisation stage. In accordance with our approach, we provide Octo with an additional, optional feature map input, alongside the standard image inputs. This optional input facilitates the bypassing of the SmallStem network, as any input images are disregarded when this input is utilised for inference. Given the absence of a well-trained action-head for application in simulation, we decided against propagating through the action head. Instead, we focus on accumulating the readout tokens to produce a scalar value, which serves as the final output of the Octo transformer model. This decision is driven by our goal to visualise reasoning processes specifically within the Octo transformer, without the potential distortion introduced by an inadequately trained actor component. The readout tokens are subsequently condensed by summing their values, and the resulting scalar value is then backpropagated towards the last CNN layer, from which A_k was initially captured. By not relying on the actor, we thus ensure that our visualisation accurately reflects the inherent reasoning behaviour of the Octo transformer itself,



Fig. 10. Image with index 0 of our first trajectory taken from the Bridge Dataset [74].

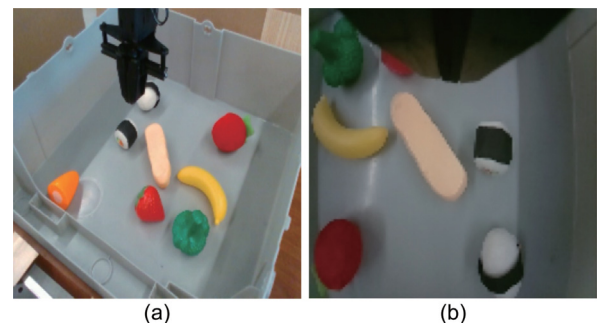


Fig. 11. Images with index 0 of our second trajectory taken from the Bridge Dataset [74]. (a) Primary camera view. (b) Wrist camera view.

independent of subsequent action-related distortions. In theory, we could utilise the outputs of the action heads to visualise their individual impact on model focus. However, for the purpose of equally comparing real and synthetic data, we opted to exclude the action heads from the visualisation process.

4. Experiments and results

Within the following sections, we will elaborate on all of our experiments conducted, grounded in our methodology outlined in Section 3. This includes the verification of our reasoning visualisation approach, domain transfer and in-simulation training from Octo and its components. Lastly, we provide visual based reasoning explanations of Octo, aiming to elucidate and corroborate our results from the remaining experiments.

4.1. Reasoning visualisation

Utilising the visualisation technique detailed in Section 3.5, we are able to create valid heatmaps that effectively reveal the model's general focus. These saliency maps, as shown in Fig. 12, facilitate an accessible understanding of the model's reasoning and clearly exhibit which objects in the scene are most influential in producing the output y . Additionally, we demonstrate the following capabilities of our approach:

Contrasting CNN-Focus with Full-Model Focus: By employing uniform weights of one to isolate the CNN focus, as outlined

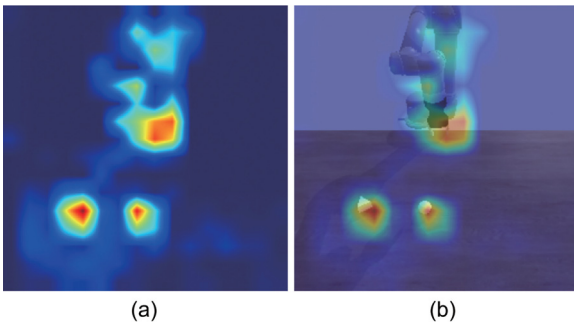


Fig. 12. Visualisation of the model focus: Utilising the method described in Section 3.5, (a) shows the raw heatmap produced, while (b) shows the superimposed image and heatmap (using Octo-small).

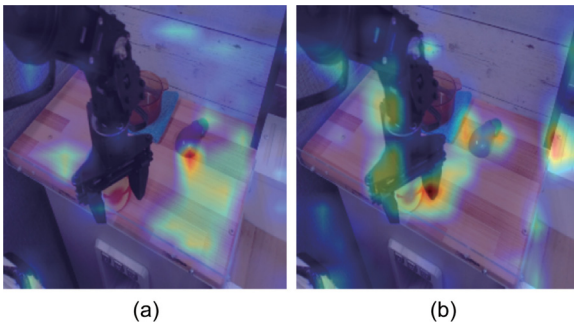


Fig. 13. Comparison of CNN-only and full model focus: Applying uniform weights of one, (a) shows the focus of the SmallStem CNN, while (b) visualises the subsequent model's influence on the CNN prediction (using Octo-base).

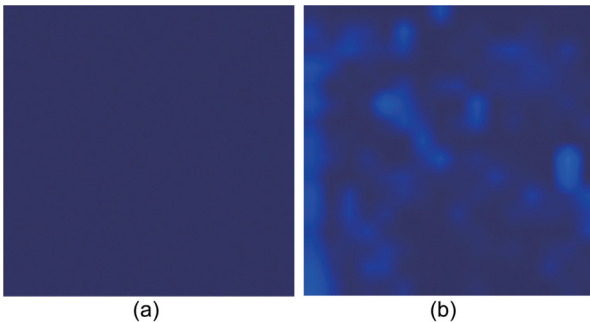


Fig. 14. Sanity Check [75] Results: Most heatmaps produced after randomising model parameters equal the heatmap in (a), while the remaining are similar to (b).

in Section 3.5.1, we can successfully distinguish between the CNN prediction and the underlying reasoning of the subsequent model architecture. This differentiation reveals the influence of succeeding layers on targeted objects, as visualised in Fig. 13.

Sanity Check: Furthermore, we conduct a model-based sanity check for saliency maps, as proposed by [75], which involves randomising the weights of the Octo transformer model without modifying its architecture. We observe that this leads to significant variations in the resulting heatmaps – some exhibiting random noise (Fig. 14b), while most showed no noticeable highlights at all (Fig. 14a) – affirming that model parameters substantially impact resulting weights and thus saliency maps. According to [75], these observations confirm the method's utility for model debugging, thus enable us to gain insights into model reasoning.

Visualising the Impact of Diverse Inputs: The experiments further reveal that the visualised focus encompasses the entirety

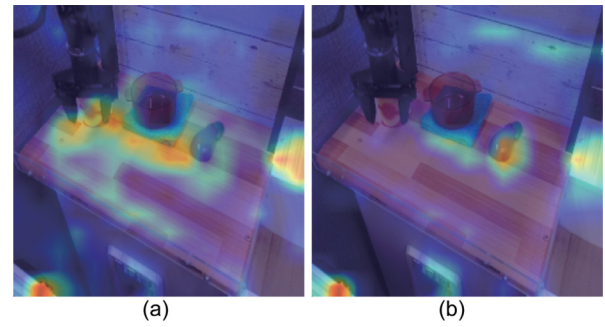


Fig. 15. Comparison of language instructions: (a) shows the resulting heatmap from Octo-base, when the area next to the pod is targeted in the language instruction, whereas for (b), the eggplant was part of the instruction.

of the model input, effectively integrating not only visual data from images but also the language instruction. This allows us to comprehensively visualise the impact of diverse inputs on the model's reasoning. Upon exploring various experiments in altering the language instruction, we observe discernible shifts in model focus, validating the method's ability to capture the effect of changes in input, as demonstrated through a specific example depicted in Fig. 15. In scene setups where the model performs effectively, the impact of language instruction is clearly visualised. For instance, Fig. 15a utilises the language instruction "Place the can to the left of the pot", whereas Fig. 15b, with identical visual input, illustrates the influence of language instruction "Grab the purple eggplant". Comparing both superimposed images, the prompts' influences become readily apparent, indicating a distinct shift in focus towards the area around the pod or the eggplant, contingent upon the specified input. In contrast, the general CNN focus in this scene is merely directed at the overall table surface, as depicted in Fig. 13a. Besides visual reasoning, we can thus clearly visualise the model's focus in response to language instructions or any arbitrary inputs provided to the model.

However, there is a limitation: When the model struggles with the scene or exhibits generally poor performance – such as being unable to differentiate between numerous objects within a cluttered environment – the influence of language instructions becomes less pronounced. To deduce the impact of non-visual input on the model's reasoning, a well-trained model is required, especially with regard to visual input. As demonstrated in Fig. 16, this restriction is evident when applying the cluttered scene shown in Fig. 11. The model exhibits clear difficulties in differentiating between objects or in orienting itself within this complex, cluttered environment, thus reducing language instruction impact evidence.

Interpreting Individual Layer Impact: Finally, our results demonstrate, that the method enables visualisation of reasoning processes within the Octo model, even in absence of the actor component. This validates our assumption from Section 3.5.1, demonstrating that the impact of each individual layer can be observed and visualised when our method is additionally applied to the outputs of preceding layers.

To conclude, we show that this procedure, followed by accumulation of model outputs, generates valid visual interpretations. These interpretations are grounded in the complete input, including non-visual types, and offer insights into the model's parameter-based internal reasoning. We have proven that these visualisations are applicable for model debugging and can be extended to subsequent layers to deliver nuanced reasoning explanations, even when dealing with complex, non-interpretible, or intermediate representations like embeddings or continuous

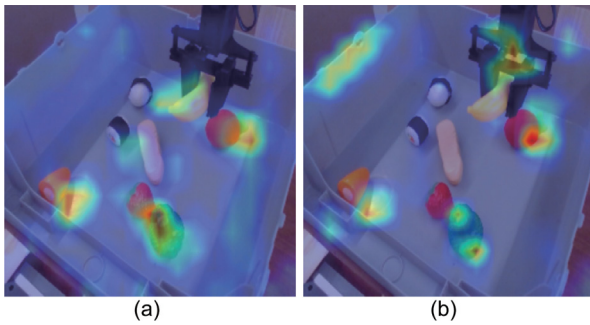


Fig. 16. Limitation in non-visual Input Reasoning Representation: We see less pronounced impacts of language instructions when the model (here Octo-base) has difficulties in visual scene interpretation. (a) Banana. (b) Red & Round Apple.

actions. Limitations of the method include potential susceptibility to outliers. Occasionally, produced heatmaps might not align with expected results, which could suggest imperfections within the model itself. Nonetheless, by generating multiple heatmaps, we can reliably differentiate outliers from valid performance, thereby affirming the robustness of the visualisation approach.

4.2. Real to sim transfer & training in simulation

Zero-shot: In applying Octo zero-shot within the simulation environment described in Section 3.1 across various tasks that include the colour- and shape-based scene setups, with task descriptions outlined in Section 3.2, we encountered several significant findings: Neither the Octo base, nor the small variant can solve any of the tasks presented within the simulation environment. Experimenting with different language instructions showed a negligible impact on the models' behaviour, failing to produce significant enhancements in performance. Efforts to enhance performance by implementing various camera view angles and positions, adjusting fields of view, testing different backgrounds and experimenting with various action interpretations, including absolute or relative angular velocities and positions, as well as TCP pose deltas, were similarly ineffective. This considerable lack of performance, which renders the model unusable in simulation, prompted us to initiate training within the simulation environment. To evaluate Octo's capacity to generalise to simulated environments, while avoiding the creation of exhaustive datasets and minimising training effort, we employed our method from Section 3.3. This method specifically allows training the action-head using SAC RL, while keeping the remaining Octo architecture components frozen. During this process, only the action head parameters are subject to updates, thus enabling fast-paced training, and allowing us to validate Octo's generalisation capability – shifting from real to simulated visual inputs. We anticipate that Octo, trained on a diverse dataset, would retain the ability to generate valid embeddings for the heads, enabling the actor to resolve simple tasks effectively, especially when the actor is trained to interpret those very embeddings using SAC.

Privileged Training: Before advancing with this training method using Octo, we test our implementation using the privileged training setup as detailed in Section 3.4. The results within this setup are promising: The best actors achieve a success ratio of 1.0 after merely 80k training steps in the simulation, as shown in Fig. 17a. After the success ratios converge to one, we can further observe decreasing actor losses in Fig. 17b, that thereafter level off at roughly -100. These findings demonstrate, that our training setup, including the baseline reward formulation, simulation setup, SAC implementation, as well as actor-/ and critic- head

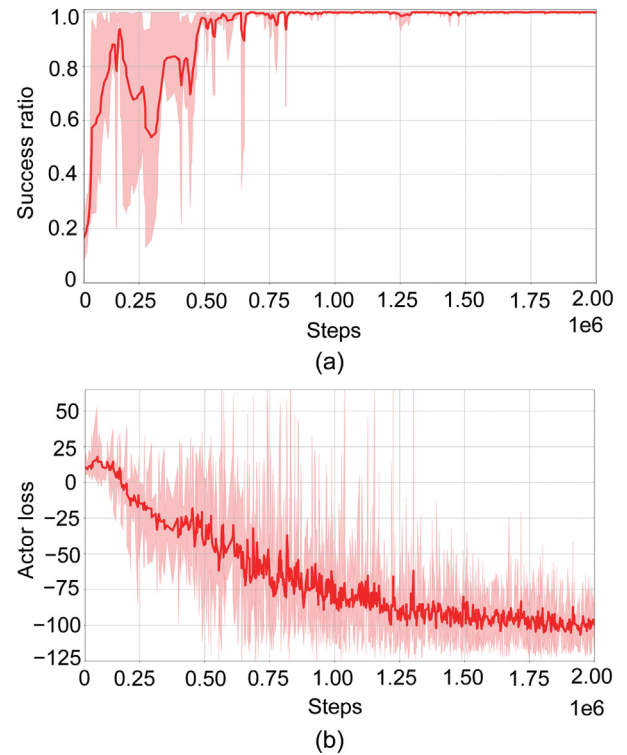


Fig. 17. Privileged Training (Fig. 9) Metrics: Showing the mean over all seeds in bold, including min and max values for the given seeds. Mean values are smoothed through an EMA with a weight of 0.6.

modifications contain no major errors and are well defined to train the heads with SAC RL utilising the PyBullet simulation.

SAC Training: We proceed to utilise our approach in combination with the frozen Octo stack (Fig. 8), which now provides embeddings as input to the heads, to train various actor-/ critic-head combinations from scratch using our well-known action-space representations for action-head initialisation as specified in Section 3.2. While the training configuration described in Sections 3.1 and 3.2 have shown effectiveness with privileged input for our models, we conduct a thorough series of experiments incorporating numerous adaptations. For each adaptation, we commence new training runs for the combinations of both scene- and task-definitions as well as both action spaces, as detailed in Sections 3.1 and 3.2, with each training targeted at lasting two million steps: Using the reward formulations from Section 3.2, we examine various adaptations, such as implementing squared distances as punishment measure, assigning penalties to very small or insignificant actions, and altering the reward range through different scales. Nevertheless, training metric analyses indicated, that the presented reward formulations (Eqs. (3) and (4)) remain the most fitting for the desired training behaviour. In addition, we examined diverse hyperparameter settings, varying learning rates and batch sizes, and assessed different configurations for both actor-/ and critic-network architectures. We experimented with several action-/ and critic-head structures, using both Octo-base and Octo-small models as static components. The most effective setup in terms of performance and hardware efficiency, featured training of 11 parallel environments with a batch size of 64, a static learning rate of 3×10^{-4} , a smoothing coefficient τ of 5×10^{-3} , the discount rate γ at 0.99, shallow head architectures and did not include random sampling to initially fill the replay buffer. Ultimately, we also executed experiments without applying scene randomisation. However, despite conducting approximately 50 experimental setups across different scene and task definitions

during training, no actors produced were capable of solving even the simplified task setups. As such, we do not enumerate all the hyperparameter combinations here. Fig. 18 shows the best among those experiments utilising the Octo transformer, where object placement randomisation was enabled. All three metrics indicate a stagnating learning process, illustrated by slowly decreasing reward values and increasing actor losses. Even though the success ratio during training across the last 200 episodes, shown in Fig. 18a, peaks at about 25% for one seed, the majority of these successes arise from advantageous object placement and exploitative behaviour ignoring visual input, and therefore, fail to produce favourable behaviour in the agent. As an illustration, the highest-performing actors merely learned to touch one particular object at a given position in the scene, when randomisation was not utilised. Conversely, when randomisation was employed, a few agents developed the tendency to perform swift movements within the 2D-task configurations, enhancing their chances of accidentally hovering over the target object. This outcome indicates that, despite the reward formulation being valid for solving the task, the actor-heads entirely neglect embeddings obtained from Octo that include visual and language input.

Ultimately, these findings conclude, that Octo demonstrates poor visual generalisation when transitioning to unfamiliar environments, such as simulation settings, even when these tasks closely resemble the training distribution. Thus, the embeddings produced by either Octo network insufficiently capture the required state representation to solve the task at hand. Nevertheless, these experiments alone do not precisely pinpoint the root of the problem. It remains unclear whether it arises from Octo's inherent limitations in dealing with visual distribution shifts, its ability to accurately interpret language instructions, or other underlying factors.

4.3. Explaining Octo's visual reasoning

As such, to delve deeper into the model and its reasoning processes, we apply our proposed visualisation method detailed in Section 3.5, previously validated for precision and efficacy in Section 4.1. Observations from our visualisation methods with Octo present a nuanced picture:

Octo-base: In training distribution scenarios, like the trajectories depicted in Fig. 10, the model generally concentrates effectively on the relevant objects, exhibiting only a few outliers. The model demonstrates effectiveness in differentiating between objects based on language instructions, as observed in Fig. 15. The CNN typically targets generally applicable aspects for task resolution, while the combined model refines this selection further (e.g. Fig. 19a vs. Fig. 19b). In cluttered settings with multiple objects in the scene, as seen in Fig. 11, the CNN generally maintains accurate focus. However, the full model's efficacy diminishes (Fig. 19d), facing difficulties in efficiently distinguishing between objects, likely due to the large number of elements highlighted by the CNN (Fig. 19c). As a result, the impact of language instructions is reduced considerably, which can be observed in Fig. 16. Additionally, there is an increase in outliers across the trajectories when compared to tidier scene setups. In our simulation environment, Octo-base's CNN (Fig. 19e) appears nearly non-functional, failing entirely to concentrate on relevant scene components. Occasionally, the full model achieves to partially focus on relevant objects, though burdened by numerous outliers and huge performance decrease compared to both real-world datasets. Distinctly, wrist images perform reasonably well in simulation compared to primary images.

In summary, Octo-base seems to be overfitted to its training distribution, showcasing relatively competent performance in environments that are familiar from training. This is particularly

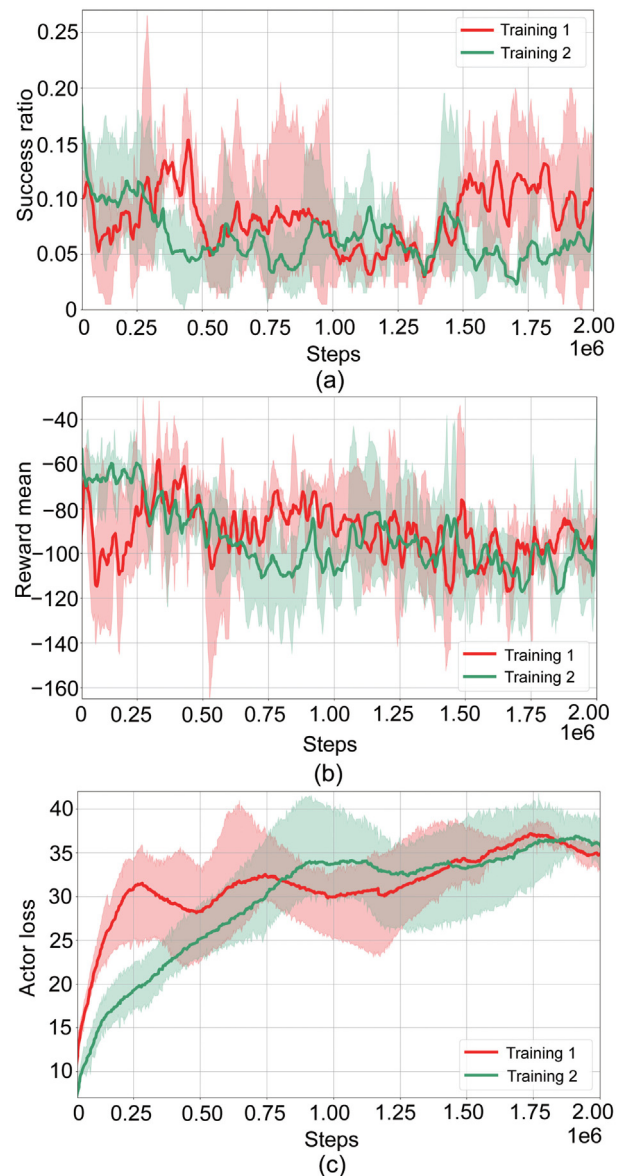


Fig. 18. Best Training Runs' Metrics utilising Octo as specified in Fig. 8: Showing the mean over all seeds in bold, including max and min values for the given seed for both trainings. Mean values are smoothed through an EMA with a weight of 0.6.

evident in scenarios like those illustrated in Fig. 15, where language meaningfully influences the model's focus. However, the base model's performance noticeably declines when interpreting cluttered scenes, and it also displays high sensitivity to distribution shifts as strongly evident in simulated scenarios. Even when subjected to clean scene configurations in simulation, the model remains largely ineffective within simulated environments, predominantly focusing on irrelevant or random elements across the scene, as reflected in Fig. 19f. This sensitivity hinders its ability to generalise effectively beyond its established training data, posing challenges in adapting to novel contexts without joint fine-tuning of the complete parameter-set.

Octo-small: Within the training distribution and tidy scene setups (Fig. 10), the CNN mainly prioritises relevant objects (Fig. 20a), however, applying the full model (Fig. 20b) sometimes even reduces focus accuracy. In general, we observe a perceptible performance drop compared to Octo-base. Language interpretation capabilities, though noticeably impacted compared

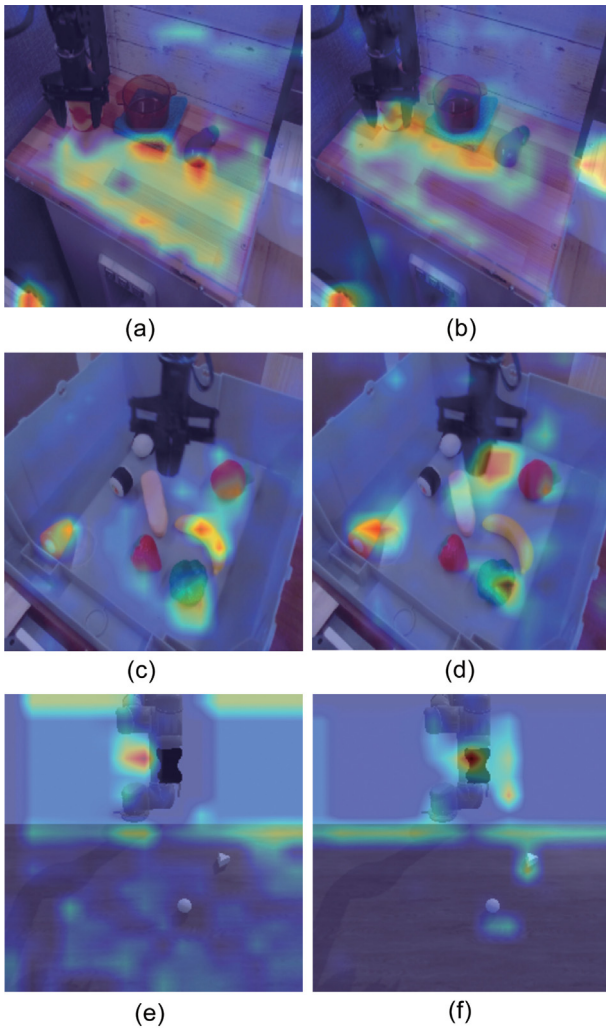


Fig. 19. Octo-base Reasoning Visualisation: Comparing CNN- vs complete model-focus across all environments. (a) CNN-only Tidy. (b) Full-Model Tidy. (c) CNN-only Cluttered. (d) Full-Model Cluttered. (e) CNN-only Simulation. (f) Full-Model Simulation.

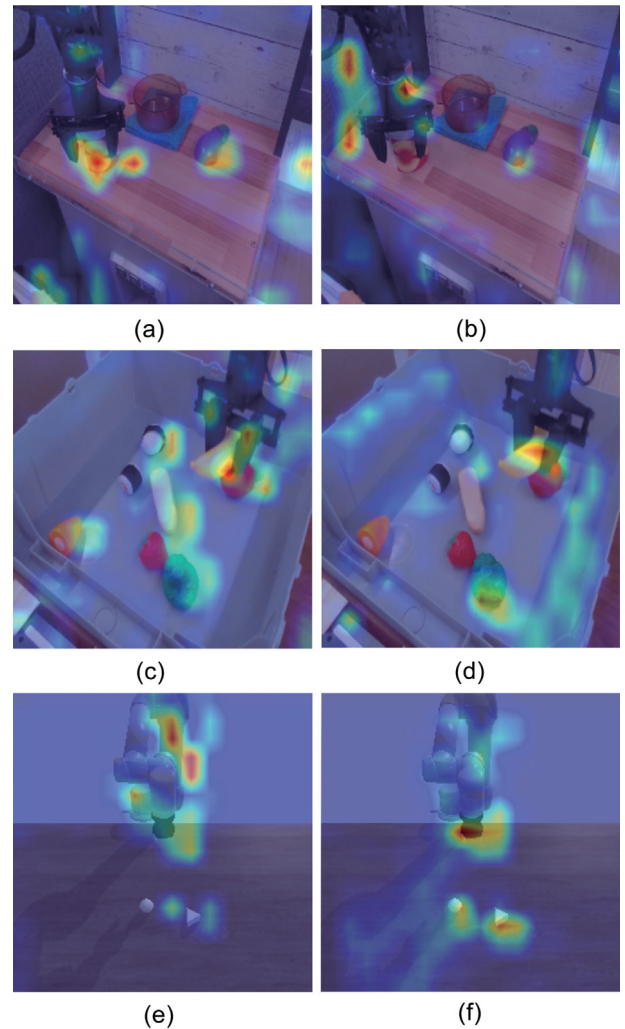


Fig. 20. Octo-small Reasoning Visualisation: Comparing CNN- vs complete model-focus across all environments. (a) CNN-only Tidy. (b) Full-Model Tidy. (c) CNN-only Cluttered. (d) Full-Model Cluttered. (e) CNN-only Simulation. (f) Full-Model Simulation.

to the base model, still show considerable effects within the training distribution (Fig. 21c vs. Fig. 21d). However, in cluttered real-world scenes, the CNN- and full-model sometimes yield conflicting results: We can see a general good focus of the CNN-only (Fig. 20c), while the full model generally refines the focus based on the language instruction, but also adds various outliers, making the prediction more unstable in total (Fig. 20d). Still, Octo-small occasionally surpasses the base model in handling those cluttered contexts. In simulated settings, Octo-small's CNN shows a remarkably good general focus (Fig. 20e), particularly on the objects and the gripper, significantly outperforming its Octo-base counterpart. Additionally, the full model refines this focus even further as depicted in Fig. 20f, effectively reducing outliers introduced by the CNN. Nevertheless, it struggles to distinguish between objects in simulation based on language input. We observe, that language instructions do not have any impact on Octo-small's focus when applied to the simulated environment (Fig. 21a vs. Fig. 21b), highlighting a gap in language interpretation when exposed to distribution shifts. Overall, the focus within wrist images across all domains is similar, although slightly worse when compared to the base model's.

Conclusively, while Octo-small does not seem to be overfitted to the training distribution, demonstrating potential for

generalisation, issues in language understanding persist, limiting usability beyond the original training context without comprehensive model adjustment. These observations contribute to explaining why our attempts as described in Section 4.2 failed to yield successful models. This analysis deduces, that inadequate language processing potentially contributes to hurdles, faced in Octo-small's performance within unfamiliar settings.

5. Discussion

In the following sections, we will discuss and interpret our results from Section 4 to establish general guidelines for RFM research. These guidelines will focus on generalisation capabilities, evaluation and benchmarking, explainability, and finally architecture and modularity of RFMs.

5.1. Generalisation

By evaluating Octo [3] in simulation and utilising our visualisation approach as outlined in Section 4, we observe significant difficulties in transferring Octo and its components from real to simulated environments:

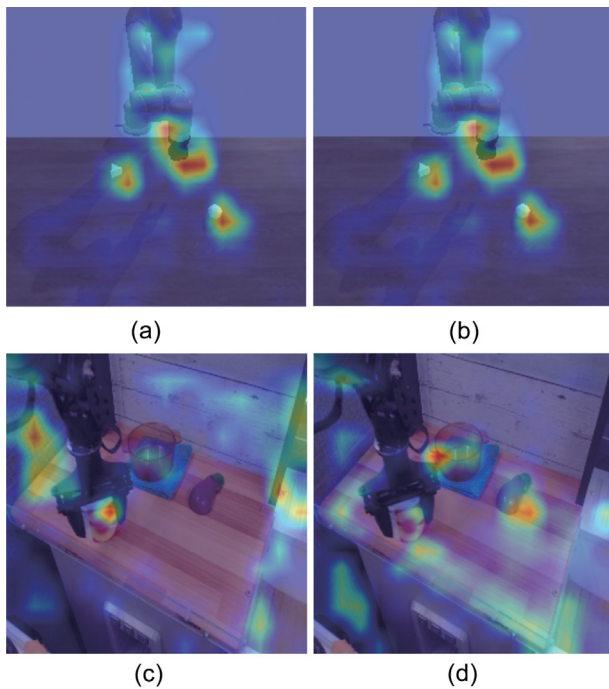


Fig. 21. Octo-small Language Reasoning Visualisation. (a) Round Object (Simulation). (b) Angular Object (Simulation). (c) Area of Pod. (d) Eggplant.

The **Octo-base** model demonstrated good visual focus with in-distribution data, yet failed to produce meaningful focus when transitioning to simulation, unable to handle synthetic data, despite its similarity to real-world conditions. The model displayed noticeable and effective language interpretation within in-distribution observations. For instance, we could verify low performance of the CNN in simulated settings, probably caused by overfitting. This low performance negatively influences subsequent layers, hindering application in simulated settings. Most of those insights derive from visualising and explaining the model's reasoning, enabling us to effectively pinpoint overfitting within specific layers or segments of the model. This indicates, that reasoning visualisation can be a simple and effective approach to diagnose lack of generalisation in the interpretation of visual information by propagating data from various domains or distributions through the model.

Octo-small, on the other hand, faced generalisation challenges in another modality despite not suffering extensive overfitting. While it exhibited poorer visual focus overall, it demonstrated consistent visual performance across all domains, including an exceptional good focus within OOD scenarios as in simulation, indicating minimal to no overfitting. Noticeably, the language input showed noticeable impacts to the model's reasoning in-distribution, yet showed no impact in OOD settings. This underscores the need to evaluate generalisation and performance across multiple axes, not only in terms of visual capabilities, but also concerning other modalities, such as language and their interconnections. For example, even with remarkable visual performance, Octo-small struggled to connect language and visuals in OOD scenarios, rendering the model non-functional despite its visual effectiveness. As stated before, our observations regarding Octo's lack of performance is also verified by authors of other publications, such as $\pi 0$ [29], RDT-1B [26], ReVLA [67], and the SIMPLER [68] benchmark. Although they could not identify the differentiated problems regarding generalisation capabilities and language interpretations, as shown by our experiments

Furthermore, Hu et al. [2] underscore the challenges of generalisation and data scarcity as key obstacles for general-purpose robots. These challenges are particularly relevant in industrial applications, where the limited availability of real-world data necessitates the use of simulation or synthetic data for training and evaluation. Yet, sim-to-real transfers present additional hurdles for generalisation, necessitating an exploration of current models' generalisation capabilities. Ultimately, diversifying datasets – not only with novel objects, noise, and other dimensions – but also by incorporating simulated data could counter both generalisation and data scarcity challenges. The authors of AutoRT [76] for instance, state the difficulty of real-world data collection and propose an autonomous system to collect real-world data using VLMs. While this could contribute to reduce data scarcity, it underscores the necessity of data for robotics, yet lacks considerations of combining real-world with synthetic data. In contrast, Hu et al. [2] discuss two separate paths: utilising either simulation and real-world data, each presenting its own challenges. We propose that a combination of both methods could improve generalisation capabilities while broadening the data pool.

In summary, even as this field progresses rapidly, the asserted performance of RFMs like Octo currently appear to be very unstable and heavily dependent on specific tasks or observation distributions. It is essential to cultivate generalisation across diverse aspects – vision, language, along with other modalities, and notably, their interconnections. If a model is exposed to distribution changes along one axis, we must be aware that this shift might influence its performance across various modalities. We observed this in Octo-small, where transitioning from real to simulated environments negatively impacted its language skills within the new domain, despite stable visual understanding. Additionally, augmenting real-world datasets with synthetic data presents a straightforward method to enhance generalisation. This not only diversifies the dataset but also tackles data scarcity, particularly in industrial contexts.

5.2. Robotic benchmark critique

Although benchmarks as outlined in Section 2.2 are instrumental in assessing specific aspects of RFMs and might give valuable information about the performance of a given model, within a given data- and task- distribution, they are intrinsically limited, especially when used for model-to-model comparisons: While all these benchmarks assess a model's ability across a range of tasks – implying an evaluation of generalisation behaviour – there are substantial differences in the types of tasks involved, the benchmark's objectives, and features. These differences can include whether additional modalities like language are incorporated, the focus on robotic table-top tasks versus standard RL tasks such as cart-pole, and the environment representation. Within the same task domain, such as table-top scenarios, the range of visual representation extends from the minimalism and simplicity of Meta-World [45] to the realism of Robosuite [46], and the scene complexity of LIBERO [41]. Thus, even within a single domain, benchmarks can exert more pronounced distribution shifts on some models than others, which is a critical aspect to consider when analysing benchmark performance. Additionally, the simplicity of some tasks may not reflect real-world complexities, potentially leading to overfitting when models are trained specifically for these tasks. As a result, some models may excel in benchmark-specific environments but struggle to showcase generalisation. Conversely, within a single benchmark, visuals typically remain consistent even across various object types, as the same simulation environment is used and objects are often constrained to task-categories such as table-top

settings, even if a relatively diverse task-set is employed, such as in LIBERO [41]. Consequently, to accurately assess a model's performance and generalisation capabilities, ensure a comprehensive evaluation across different distributions and mitigate many limitations associated with using a singular benchmark, it is imperative to employ multiple benchmarks that are distinct from one another, featuring varied visual representations and scene configurations. The applied set of benchmarks and evaluation methods, in its completeness, must include all – tailored fitting to unseen scenarios, zero-shot applications without specific fine-tuning for each benchmark and OOD representations from training data. Furthermore, RFMs trained on diverse datasets aiming for generalisation sometimes underperform when assessed against targeted benchmarks, especially compared to models optimised for specific tasks, hindering fair cross-comparison. This is particularly relevant when some models are trained for diversity, while counterparts have been exclusively fitted to a given benchmark, as done for PLEX [13]. Moreover, reliance on simulated environments can lead to observation distribution shifts for models trained on varied real-world datasets, unlike those solely trained on synthetic data. Specifically, when comparing “seen” and “unseen” objects or environments, the distribution shifts may be significantly larger for one model than for another with a different data composition. For instance, even the authors of QueST [25] themselves acknowledge that the pretraining dataset bears a strong similarity to the benchmark tasks used. We thus need to evaluate on how each benchmark could benefit one model vs. another (e.g. based on task- or data- distribution) when choosing between benchmarks. Ultimately, similar to foundation models in other domains, assessing the capabilities of models that display emergent behaviour outside their training distribution or initial task definition has become increasingly complex, as noted by Bommasani et al. [5]. Thus, while valuable for certain assessments, benchmarks do not provide insights into the reasoning behind model behaviours and fail to explain why a model acts as it does. Additionally, merely assessing success ratios across various benchmarks fails to evaluate generalisation across multiple axes, such as visual and language modalities, as discussed in Section 5.1. This absence of explainability underscores the need for additional methods to gain insights into model reasoning. To evaluate not just plain performance but also more nuanced aspects, also during model creation, it is vital to assess the impact of different modalities. This includes evaluating generalisation or overfitting across various axes, such as visual input (objects, backgrounds, etc.), language understanding, any other modalities the model may utilise as input along with their integration (e.g., switching modalities, including different objects, targeting them differently within the language instruction) and their relation to another. Beyond visual reasoning interpretation (such as presented in Section 3.5), exploring additional pathways for achieving explainability within RFMs is essential, warranting consideration in future research. This holistic approach ensures a more comprehensive understanding of a model's functionality, moving beyond surface-level performance metrics to deeper insights into their operational mechanisms.

All of those observations regarding current robotic benchmarks boil down to the following aspects: Firstly, as noted by Ponbavathi et al. [77], high accuracies on standard benchmarks can artificially present an overly positive picture of a model's performance. We evidently noticed this in our experiments when comparing Octo's OOD performance with its claims, to surpass state-of-the-art models, such as RT-2X, in zero-shot performance. Although their results might hold within the specified context, we contend that those results may not fully capture the entire situation, as we, including others, could not verify the claimed performance for different, yet slight, distribution shifts, as outline

in Sections 4 and 5.1. Hence, when selecting a model for one's own application, it is crucial to make a careful choice, not relying solely on benchmark statistics. Instead, consideration should also be given to evaluations and possible limitations that benchmarks might not reflect, including architectural aspects and other inherent model features. Thus, while benchmarking provides a fundamental tool for comparing RFMs' performance, careful consideration of benchmark selection, combination of several varying benchmarks, task diversity, tailored fitting, zero-shot application and especially a critical evaluation of applicability to *all* benched models is vital to ensure proper differentiation of generalisation capabilities, especially when comparing models by using those benchmarks. Additionally, incorporating explainability into the evaluation process is indispensable, as highlighted by Sun et al. [78], who categorise explainability into three components: data explainability, model explainability, and post-hoc explainability. In the field of robotics, the focus has predominantly been on post-hoc explainability, often centring on interpreting outcomes. However, even here, the tendency has been to simply present benchmark results without thorough explanation or interpretation. Utilising explainability methods such as reasoning visualisation, can provide clearer insights into post-hoc explainability, thereby offering a more profound understanding of model behaviour, limitations and function beyond mere performance metrics.

5.3. Explainability in RFMs

All of the aforementioned points underscore the need to delve into the explainability of RFMs. In this context, explainability extends beyond merely making models more comprehensible when they function correctly. It also involves identifying deficits, enhancing model comparability, and uncovering optimisation potential across several dimensions, such as vision, language, and other modalities, alongside their interrelations. While the traditional aspect of explainability should not be neglected, it becomes even more pertinent when applying these models to industrial use cases. In such domains, precision is paramount, and classical control approaches are often preferred due to their strict tolerances. Employing methods to elucidate model behaviour could also help evaluate a model's reliability. Insights into specific parts of the model may not only render them more acceptable alternatives for industrial applications, but also facilitate judgements regarding their reliability in such scenarios. Bommasani et al. [5] highlight the importance of safety and robustness in foundation models. We add, that gaining insights into a model can yield conclusions about system robustness, both during modelling and evaluation stages, that becomes evident when we reflect on the real-to-sim and in-simulation experiments (Section 4): Initially, these experiments merely indicated that neither Octo-base nor Octo-small could effectively handle synthetic data. However, the reasons for these failures were unclear, leading to assumptions that both variants faced similar underlying problems. Moreover, distinct visual reasoning performances of these models within simulation were unknown, as was the CNN's impact. The visualisation experiments and their outcomes (Section 4.1) have now pinpointed these issues, suggesting specific improvements – such as tuning Octo-base's CNN, while focusing on Octo-small's underperformance in language comprehension within OOD settings. Through our reasoning visualisation experiments, it became apparent that despite their similar architecture, the models encountered distinct challenges when transitioning from real to simulated data. One model was overfitted to the training dataset, while the other struggled with language interpretation when combined with OOD visuals, as detailed in Section 5.1. These insights are crucial for judging a model's performance and are

even more significant during the creation and training of such models.

In conclusion, evaluating these models requires focusing not solely on performance but also taking a closer look at their internals – augmenting explainability through architectural design and evaluation methods, aiding in assessment of model safety and predictability. This explainability is indispensable for industrial applications by revealing hard-to-detect weaknesses, resulting in increased robustness. When executed correctly, explainability can elucidate reasoning along multiple axes, encompassing not only visual reasoning but also language understanding and the interpretation of other modalities. Critically, it reveals the dependencies between axes and the distortions caused in others by distribution shifts in one axis, as outlined in Sections 5.1 and 5.2. In addition to visualising reasoning for explainability, we should consider other methods such as reinforcement or imitation learning-based explanations, as stated by Atakishiyev [79]. Sun et al. [78] emphasised the necessity of explainability across multiple dimensions. Thus, explaining not only the results but also the internal mechanisms of a model is crucial. In order to facilitate the development of models capable of displaying robust and reliable performance across diverse applications and scenarios, we should also consider explainability in the design of models and their architecture.

5.4. Architectural challenges and modularity

Our assessment of Octo revealed several design aspects that restricted its explainability and modularity, contradicting its claims in the latter. A major issue is the absence of defined interfaces or embedding representations within the model – not just from the CNN to the Octo-transformer, but also from the Octo-transformer to the action-head. This became particularly evident when transitioning from Octo-small to Octo-base, where the Octo-transformer readout-embedding dimensionality changes. Consequently, switching from the small to the base model requires adaptation of the succeeding architecture, the action-head itself. As already mentioned, we observed Octo-base's CNN to be overfitted, thus incapable of handling synthetic data, rendering it ineffective in simulation. However, without interfaces, fine-tuning the CNN independently is questionable, as the subsequent model might not manage to interpret resulting new representations, necessitating joint fine-tuning of the CNN in combination with the complete subsequent architecture. Ensuring consistent representation dimensionality could address structural compatibility issues, facilitating easy exchange of components, such as action-heads, without needing adjustment to other components. Specifically, since transitioning from Octo-small to Octo-base doubles the embedding size input to the action-head instead of only improving quality, Octo could overcome this inconsistency by maintaining consistent embedding sizes at the connection points of architecture parts. Promoting such intermediate representation definitions, as by utilising uniform embedding sizes, we can easily foster modularity, allowing individual components to be selectively fine-tuned and incorporated seamlessly into the architecture, even if the *preceding* model changes – although retraining of the *succeeding* new model part would still be required. Thus, the effectiveness, computational cost, and reliability of such fine-tuning depend on the component's position within the model – as modifying early layers can affect the entire system, while task-specific modules are often more amenable to isolated updates.

However, to advance modularity even further, we should not only restrict ourselves to fixed intermediate representations' dimensionality, but leverage those definitions to allow for integration of concrete semantic interfaces between given components

of the model architecture. Such interfaces would allow *any* part of the model to be exchanged without requiring *any* retraining or fine-tuning, as long as exchanged components conform to the interfaces. For this, we have two general types in our mind:

Firstly, leveraging **semantic interfaces by representation**, such as bounding boxes as an allegorical task-specific example, models can benefit from being guided through expert knowledge, by assigning specific interpretations or representations to given interface definitions, essentially guiding the model's reasoning. This could be suitable for intermediate representations, that are not implemented as singular, centralised embedding within the model, but rather as addition, before merging of various input types happens within the model. Since such architectural characteristics are mostly located at early stages of the model, implementing semantic interfaces by representation would greatly increase modularity, allowing for flexible exchange of succeeding model parts without required adaption or training to *any* other model part. For instance, Müller et al. [80] demonstrated a modular approach, separating perception, policy, and control. Implemented interfaces, such as segmentation maps, enables them to exchange components without requiring adaptation in other parts of the architecture. Though these interface representations would facilitate maximum modularity and expert knowledge injection, we acknowledge, that their definition and integration may not always be feasible in large RFMs' architectures. This is due to the constraints on positioning within the architecture previously noted, and additionally, since the necessity for manual engineering might overlook unforeseen representations or introduce vulnerabilities. However, Hu et al. [2] explicitly state, that the combination of flexible modular approaches combined with end-to-end learning is *not* mutually exclusive. We think, that this combination is crucial for well-performing and generalising RFMs, allowing flexible modular components implementing specialised functionalities within unified learning. To be direct, we recommend, that a different form of interpretation can successfully achieve this combination.

Hence, we introduce the notion of **semantic interfaces by interpretation**: Here, we also build on our initial idea of embeddings that keep a fixed dimensionality representation, even if the preceding architecture changes. However, to transform this approach into a standardised interface, rather than explicitly defining the representation as previously done, our focus shifts to the implicit content within the embedding. One option is to initiate a broad interface definition and then iteratively refine this representation empirically by interpreting the interface's meaning, using methods such as masking or removing individual dimensions to assess their significance. By examining outcomes when a dimension is partially masked, we can infer the meaning of certain embedding dimensions. Such an approach echoes permutation importance, an importance measure initially applied by Breiman [81] to random forests and proposed by Baker [82] for post-hoc explainability. Finally, to achieve model-explainability, aspects found insignificant can be removed, leaving for concise and fixed interfaces. On the other hand, we could reduce manual effort, by leveraging latent-space interpretation as found in variational autoencoders and deploy components specifically tailored to interpret given embeddings. This interpretation might either be achieved implicitly, by the trained, operating model architecture that consumes the corresponding interfaced embedding, or explicitly via particular semantic interpretation components. These components are intended to evaluate if the representation sufficiently aligns with the original, broad interface definition to be used for task-execution.

Any of the mentioned approaches results in model-based modularity that enables targeted retraining or fine-tuning of individual components, without the need to re-train the entire

model jointly. This can be especially effective when combined with end-to-end learning strategies that preserve global performance. Baker et al. [82] note, that some models are “explainable by design” due to their simplicity or transparency. We posit, that by increasing modularity and assigning such standardised interfaces, this modularity will result in explainability. While our visualisation approach achieves post-hoc explainability, as detailed in Section 5.2, modularity and interface definitions can offer model-level explainability, by design. Conclusively, as previously mentioned, Sun et al. [78] separate explainability into data, model, and post-hoc explainability, highlighting the critical role that model design and architecture play in fostering explainability. We assert, that modularity is a core component in achieving model-explainability, and when combined with post-hoc explainability, as detailed in Section 5.2, it will not only greatly enhance our understanding of model behaviour, but also enable us to build more powerful models, especially in industrial contexts. This integration not only offers insights into why models behave the way they do but also allows for the identification of potential improvements and optimisations, ultimately leading to more robust and reliable models across diverse applications and scenarios.

6. Conclusion: Guidelines for RFM research

As highlighted throughout the discussions in Section 5, several key insights emerge that can guide future research in the domain of RFMs. Firstly, the need for enhanced generalisation capabilities is eminent. Our evaluations of Octo demonstrate, that significant discrepancies exist in models’ performance when transitioning between real-world and simulated environments. To address these challenges, future research must explore the integration of diverse datasets, encompassing both simulated and real-world data, to support generalisation and mitigate against data scarcity issues, especially in industrial settings. Furthermore, the evaluation of all modalities, beyond just visual inputs, is essential in achieving broader generalisation across different task domains. Secondly, while benchmarks provide valuable means to evaluate model performance, they often fall short when capturing the nuances of model reasoning and generalisation across multiple axes. Thus, future research should be cautious in relying solely on benchmark metrics. Instead, a critical evaluation of model performance should incorporate assessments of all modalities, like visual, language, the integration between these modalities, and observations across varied distribution shifts. This will lead to a more comprehensive understanding of model capabilities and limitations, thus enable more nuanced and robust evaluations. Moreover, the importance of explainability in RFMs must not be underestimated. While performance metrics highlight success, explainability elucidates the underlying model behaviour, helping to identify deficiencies to foster reliability and acceptance, particularly in industrial applications. Future research should prioritise explainability by considering architectural design and evaluation methods that provide insights into both, visual and language reasoning processes. Incorporating methods such as reasoning visualisation, alongside traditional performance metrics, will ensure a deeper understanding of model operations. In addition, enhancing model modularity can be crucial for advancing RFMs. Introducing standardised interfaces and consistent representations across components in end-to-end models will facilitate greater flexibility and enable easier component exchange. This will not only enhance the adaptability of models but will also offer model-level explainability by design, allowing for targeted model improvements aiding understanding of model functions. Ultimately, by adopting a holistic approach that facilitates generalisation by careful benchmarking, implementation of explainability methods throughout model training and development,

the evaluation of all input modalities including their correlation, modular architectures and synthetically enriched datasets, future research can drive the development of RFMs that are capable of robust performance across diverse scenarios. This comprehensive strategy will ensure the creation of models that are not only high-performing in specific scenarios but also reliable, transparent, and adaptable to the multifaceted challenges encountered in real-world applications. Specifically, the proposed interpretability method detailed in Section 3.5 provides a critical tool for understanding model reasoning, especially when navigating transitions between various modalities. In industrial contexts, this means leveraging gained interpretability to elucidate how models adapt to different types of input, such as transitioning to synthetic data. This enhanced understanding is pivotal for deploying RFMs in environments where such data variances are commonplace or required for safe operation. Furthermore, employing such explainability techniques can reveal critical insights into model performance and highlight areas requiring refinement, addressing potential discrepancies before models are deployed in high-stakes industrial applications. By integrating these insights into the design phase, organisations can pre-emptively tackle engineering challenges, ensuring the system’s reliability and robustness. Thus, our holistic approach not only fosters better model development practices, but also instills confidence in the deployability of RFMs across diverse industrial environments by fulfilling operational transparency and adaptability demands of modern industry.

Although this reasoning visualisation represents a move towards increased explainability in RFMs, illustrating how different input modalities affect model reasoning, there remains scope for future work. For instance, to create interpretability methods tailored for models without visual inputs, those with alternative architectures, or approaches that emphasise other facets of model reasoning. In our ongoing efforts, we intend to utilise and adapt RFMs for applicability within simulation, striving to bridge the gaps and address the deficiencies identified during our research. We plan to extend the application of RFMs to tackle more intricate industrial tasks, possibly including mobile robots within our scope. Additionally, we aim to advance methods for automated synthetic data generation, thereby enriching our domain-specific datasets, specifically tackling data scarcity for industrial use cases. In summary, our objective is to develop models that are more robust and capable of generalising across diverse domains, facilitating their deployment in industrial settings where classical approaches may not present the most optimal solutions.

CRedit authorship contribution statement

David Kube: Writing – original draft, Validation, Software, Project administration, Investigation, Data curation, Writing – review & editing, Visualization, Supervision, Resources, Methodology, Formal analysis, Conceptualization. **Simon Hadwiger:** Supervision, Conceptualization, Writing – review & editing, Methodology. **Tobias Meisen:** Writing – review & editing, Conceptualization, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at <https://doi.org/10.1016/j.birob.2025.100249>.

References

- [1] O.X.-E. Collaboration, A. Padalkar, A. Pooley, A. Mandlekar, A. Jain, A. Tung, A. Bewley, et al., Open X-embodiment: Robotic learning datasets and RT-X models, 2023, [arXiv:2310.08864](https://arxiv.org/abs/2310.08864).
- [2] Y. Hu, Q. Xie, V. Jain, J. Francis, J. Patrikar, N. Keetha, et al., Toward general-purpose robots via foundation models: A survey and meta-analysis, 2024, [http://dx.doi.org/10.48550/arXiv.2312.08782](https://dx.doi.org/10.48550/arXiv.2312.08782), [arXiv:2312.08782](https://arxiv.org/abs/2312.08782).
- [3] D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, et al., Octo: An open-source generalist robot policy, 2024.
- [4] R.R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-CAM: Visual explanations from deep networks via gradient-based localization, *Int. J. Comput. Vis.* 128 (2) (2020) 336–359, [http://dx.doi.org/10.1007/s11263-019-01228-7](https://dx.doi.org/10.1007/s11263-019-01228-7), [arXiv:1610.02391](https://arxiv.org/abs/1610.02391).
- [5] R. Bommasani, D.A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, et al., On the opportunities and risks of foundation models, 2022, [http://dx.doi.org/10.48550/arXiv.2108.07258](https://dx.doi.org/10.48550/arXiv.2108.07258), [arXiv:2108.07258](https://arxiv.org/abs/2108.07258).
- [6] Z. Xu, K. Wu, J. Wen, J. Li, N. Liu, Z. Che, et al., A Survey on robotics with foundation models: Toward embodied AI, 2024, [http://dx.doi.org/10.48550/arXiv.2402.02385](https://dx.doi.org/10.48550/arXiv.2402.02385), [arXiv:2402.02385](https://arxiv.org/abs/2402.02385).
- [7] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, D. Fox, RVT: Robotic view transformer for 3D object manipulation, 2023, [arXiv:2306.14896](https://arxiv.org/abs/2306.14896).
- [8] M. Shridhar, L. Manuelli, D. Fox, Perceiver-actor: A multi-task transformer for robotic manipulation, 2022.
- [9] S. Chen, R. Garcia, C. Schmid, I. Laptev, PolarNet: 3D point clouds for language-guided robotic manipulation, 2023, [arXiv:2309.15596](https://arxiv.org/abs/2309.15596).
- [10] Y. Ze, G. Yan, Y.-H. Wu, A. Macaluso, Y. Ge, J. Ye, et al., GNFactor: Multi-task real robot learning with generalizable neural feature fields, 2023, [arXiv:2308.16891](https://arxiv.org/abs/2308.16891).
- [11] P.-L. Guhur, S. Chen, R. Garcia, M. Tapaswi, I. Laptev, C. Schmid, Instruction-driven history-aware policies for robotic manipulations, 2022.
- [12] D. Driess, F. Xia, M.S.M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, et al., PaLM-E: An embodied multimodal language model, 2023, [arXiv:2303.03378](https://arxiv.org/abs/2303.03378).
- [13] G. Thomas, C.-A. Cheng, R. Loynd, F.V. Frujeri, V. Vineet, M. Jalobeanu, et al., PLEX: Making the most of the available data for robotic manipulation pretraining, 2023, [arXiv:2303.08789](https://arxiv.org/abs/2303.08789).
- [14] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, et al., Do as I can, not as I say: Grounding language in robotic affordances, 2022, [arXiv:2204.01691](https://arxiv.org/abs/2204.01691).
- [15] M. Shridhar, L. Manuelli, D. Fox, CLIPort: What and where pathways for robotic manipulation, 2021, [arXiv:2109.12098](https://arxiv.org/abs/2109.12098).
- [16] A. Stone, T. Xiao, Y. Lu, K. Gopalakrishnan, K.-H. Lee, Q. Vuong, et al., Open-world object manipulation using pre-trained vision-language models, 2023, [arXiv:2303.00905](https://arxiv.org/abs/2303.00905).
- [17] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, et al., Diffusion policy: Visuomotor policy learning via action diffusion, 2023, [arXiv:2303.04137](https://arxiv.org/abs/2303.04137).
- [18] S. Reed, K. Zolna, E. Parisotto, S.G. Colmenarejo, A. Novikov, G. Barth-Maron, et al., A generalist agent, 2022, [arXiv:2205.06175](https://arxiv.org/abs/2205.06175).
- [19] H. Bharadhwaj, J. Vakil, M. Sharma, A. Gupta, S. Tulsiani, V. Kumar, RoboAgent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking, 2023, [arXiv:2309.01918](https://arxiv.org/abs/2309.01918).
- [20] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, et al., RT-1: Robotics transformer for real-world control at scale, 2023, [arXiv:2212.06817](https://arxiv.org/abs/2212.06817).
- [21] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, et al., RT-2: Vision-language-action models transfer web knowledge to robotic control, 2023.
- [22] S. Haldar, Z. Peng, L. Pinto, BAKU: An efficient transformer for multi-task policy learning, 2024, [http://dx.doi.org/10.48550/arXiv.2406.07539](https://dx.doi.org/10.48550/arXiv.2406.07539), [arXiv:2406.07539](https://arxiv.org/abs/2406.07539).
- [23] S. Xia, H. Fang, C. Lu, H.-S. Fang, CAGE: Causal attention enables data-efficient generalizable robotic manipulation, 2024, [http://dx.doi.org/10.48550/arXiv.2410.14974](https://dx.doi.org/10.48550/arXiv.2410.14974), [arXiv:2410.14974](https://arxiv.org/abs/2410.14974).
- [24] M.J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, et al., OpenVLA: An open-source vision-language-action model, 2024, [http://dx.doi.org/10.48550/arXiv.2406.09246](https://dx.doi.org/10.48550/arXiv.2406.09246), [arXiv:2406.09246](https://arxiv.org/abs/2406.09246).
- [25] A. Mete, H. Xue, A. Wilcox, Y. Chen, A. Garg, QueST: Self-supervised skill abstractions for learning continuous control, 2024, [http://dx.doi.org/10.48550/arXiv.2407.15840](https://dx.doi.org/10.48550/arXiv.2407.15840), [arXiv:2407.15840](https://arxiv.org/abs/2407.15840).
- [26] S. Liu, L. Wu, B. Li, H. Tan, H. Chen, Z. Wang, et al., RDT-1B: A diffusion foundation model for bimanual manipulation, 2024, [http://dx.doi.org/10.48550/arXiv.2410.07864](https://dx.doi.org/10.48550/arXiv.2410.07864), [arXiv:2410.07864](https://arxiv.org/abs/2410.07864).
- [27] R. Doshi, H. Walke, O. Mees, S. Dasari, S. Levine, Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation, 2024, [http://dx.doi.org/10.48550/arXiv.2408.11812](https://dx.doi.org/10.48550/arXiv.2408.11812), [arXiv:2408.11812](https://arxiv.org/abs/2408.11812).
- [28] Q. Bu, H. Li, L. Chen, J. Cai, J. Zeng, H. Cui, et al., Towards synergistic, generalized, and efficient dual-system for robotic manipulation, 2025, [http://dx.doi.org/10.48550/arXiv.2410.08001](https://dx.doi.org/10.48550/arXiv.2410.08001), [arXiv:2410.08001](https://arxiv.org/abs/2410.08001).
- [29] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, et al., *IT0: A vision-language-action flow model for general robot control*, 2024.
- [30] Helix: A vision-language-action model for generalist humanoid control, 2025, <https://www.figure.ai/news/helix>. (Accessed 18 March 2025).
- [31] Gemini robotics, 2025, <https://deepmind.google/technologies/gemini-robotics/>. (Accessed 13 March 2025).
- [32] A. Majumdar, K. Yadav, S. Arnaud, Y.J. Ma, C. Chen, S. Silwal, et al., Where are we in the search for an artificial visual cortex for embodied intelligence? 2023.
- [33] Z. Mandi, H. Bharadhwaj, V. Moens, S. Song, A. Rajeswaran, V. Kumar, CACTI: A framework for scalable multi-task multi-scene visual imitation learning, 2023, [http://dx.doi.org/10.48550/arXiv.2212.05711](https://dx.doi.org/10.48550/arXiv.2212.05711), [arXiv:2212.05711](https://arxiv.org/abs/2212.05711).
- [34] N.M.M. Shafiullah, Z.J. Cui, A. Altanzaya, L. Pinto, Behavior transformers: Cloning k modes with one stone, 2022, [http://dx.doi.org/10.48550/arXiv.2206.11251](https://dx.doi.org/10.48550/arXiv.2206.11251), [arXiv:2206.11251](https://arxiv.org/abs/2206.11251).
- [35] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, et al., BC-Z: Zero-Shot task generalization with robotic imitation learning, 2022, [http://dx.doi.org/10.48550/arXiv.2202.02005](https://dx.doi.org/10.48550/arXiv.2202.02005), [arXiv:2202.02005](https://arxiv.org/abs/2202.02005).
- [36] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, A. Gupta, R3M: A universal visual representation for robot manipulation, 2022, [arXiv:2203.12601](https://arxiv.org/abs/2203.12601).
- [37] C. Lynch, A. Wahid, J. Tompson, T. Ding, J. Betker, R. Baruch, et al., Interactive language: Talking to robots in real time, 2022, [arXiv:2210.06407](https://arxiv.org/abs/2210.06407).
- [38] C. Wang, H. Fang, H.-S. Fang, C. Lu, RISE: 3D perception makes real-world robot imitation simple and effective, 2024, [http://dx.doi.org/10.48550/arXiv.2404.12281](https://dx.doi.org/10.48550/arXiv.2404.12281), [arXiv:2404.12281](https://arxiv.org/abs/2404.12281).
- [39] R. Zheng, C.-A. Cheng, H.D. III, F. Huang, A. Kolobov, PRISE: LLM-style sequence compression for learning temporal action abstractions in control, 2024, [http://dx.doi.org/10.48550/arXiv.2402.10450](https://dx.doi.org/10.48550/arXiv.2402.10450), [arXiv:2402.10450](https://arxiv.org/abs/2402.10450).
- [40] T.Z. Zhao, V. Kumar, S. Levine, C. Finn, Learning fine-grained bimanual manipulation with low-cost hardware, 2023, [http://dx.doi.org/10.48550/arXiv.2304.13705](https://dx.doi.org/10.48550/arXiv.2304.13705), [arXiv:2304.13705](https://arxiv.org/abs/2304.13705).
- [41] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, et al., LIBERO: Benchmarking knowledge transfer for lifelong robot learning, 2023, [http://dx.doi.org/10.48550/arXiv.2306.03310](https://dx.doi.org/10.48550/arXiv.2306.03310), [arXiv:2306.03310](https://arxiv.org/abs/2306.03310).
- [42] S. Lee, Y. Wang, H. Etukuru, H.J. Kim, N.M.M. Shafiullah, L. Pinto, Behavior generation with latent actions, 2024, [http://dx.doi.org/10.48550/arXiv.2403.03181](https://dx.doi.org/10.48550/arXiv.2403.03181), [arXiv:2403.03181](https://arxiv.org/abs/2403.03181).
- [43] J. Yang, C. Glossop, A. Bhorkar, D. Shah, Q. Vuong, C. Finn, et al., Pushing the limits of cross-embodiment learning for manipulation and navigation, 2024, [http://dx.doi.org/10.48550/arXiv.2402.19432](https://dx.doi.org/10.48550/arXiv.2402.19432), [arXiv:2402.19432](https://arxiv.org/abs/2402.19432).
- [44] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, et al., ViNT: A foundation model for visual navigation, 2023, [http://dx.doi.org/10.48550/arXiv.2306.14846](https://dx.doi.org/10.48550/arXiv.2306.14846), [arXiv:2306.14846](https://arxiv.org/abs/2306.14846).
- [45] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, et al., Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning, 2019.
- [46] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, K. Lin, et al., Robosuite: A modular simulation framework and benchmark for robot learning, 2025, [http://dx.doi.org/10.48550/arXiv.2009.12293](https://dx.doi.org/10.48550/arXiv.2009.12293), [arXiv:2009.12293](https://arxiv.org/abs/2009.12293).
- [47] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D.d. Casas, et al., DeepMind control suite, 2018, [http://dx.doi.org/10.48550/arXiv.1801.00690](https://dx.doi.org/10.48550/arXiv.1801.00690), [arXiv:1801.00690](https://arxiv.org/abs/1801.00690).
- [48] O. Mees, L. Hermann, E. Rosete-Beas, W. Burgard, CALVIN: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks, 2022, [http://dx.doi.org/10.48550/arXiv.2112.03227](https://dx.doi.org/10.48550/arXiv.2112.03227), [arXiv:2112.03227](https://arxiv.org/abs/2112.03227).
- [49] J. Luo, C. Xu, F. Liu, L. Tan, Z. Lin, J. Wu, et al., FMB: A functional manipulation benchmark for generalizable robotic learning, *Int. J. Robot. Res.* (2024) 02783649241276017, [http://dx.doi.org/10.1177/02783649241276017](https://dx.doi.org/10.1177/02783649241276017).
- [50] S. James, Z. Ma, D.R. Arrojo, A.J. Davison, RL-Bench: The robot learning benchmark & learning environment, 2019, [http://dx.doi.org/10.48550/ARXIV.1909.12271](https://dx.doi.org/10.48550/ARXIV.1909.12271).
- [51] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, A. Torralba, Learning deep features for discriminative localization, 2015, [http://dx.doi.org/10.48550/arXiv.1512.04150](https://dx.doi.org/10.48550/arXiv.1512.04150), [arXiv:1512.04150](https://arxiv.org/abs/1512.04150).
- [52] J. Zhang, Z. Lin, J. Brandt, X. Shen, S. Sclaroff, Top-down neural attention by excitation backprop, 2016, [http://dx.doi.org/10.48550/arXiv.1608.00507](https://dx.doi.org/10.48550/arXiv.1608.00507), [arXiv:1608.00507](https://arxiv.org/abs/1608.00507).
- [53] J. Gu, Y. Yang, V. Tresp, Understanding individual decisions of CNNs via contrastive backpropagation, 2019, [http://dx.doi.org/10.48550/arXiv.1812.02100](https://dx.doi.org/10.48550/arXiv.1812.02100), [arXiv:1812.02100](https://arxiv.org/abs/1812.02100).
- [54] R. Fong, M. Patrick, A. Vedaldi, Understanding deep networks via extremal perturbations and smooth masks, 2019, [http://dx.doi.org/10.48550/arXiv.1910.08485](https://dx.doi.org/10.48550/arXiv.1910.08485), [arXiv:1910.08485](https://arxiv.org/abs/1910.08485).
- [55] R. Fu, Q. Hu, X. Dong, Y. Guo, Y. Gao, B. Li, Axiom-based grad-CAM: Towards accurate visualization and explanation of CNNs, 2020, [http://dx.doi.org/10.48550/arXiv.2008.02312](https://dx.doi.org/10.48550/arXiv.2008.02312), [arXiv:2008.02312](https://arxiv.org/abs/2008.02312).

- [56] A. Chattopadhyay, A. Sarkar, P. Howlader, V.N. Balasubramanian, Grad-CAM++: Improved visual explanations for deep convolutional networks, in: 2018 IEEE Winter Conference on Applications of Computer Vision, WACV, 2018, pp. 839–847, <http://dx.doi.org/10.1109/WACV.2018.00097>, arXiv:1710.11063.
- [57] H. Wang, Z. Wang, M. Du, F. Yang, Z. Zhang, S. Ding, et al., Score-CAM: Score-weighted visual explanations for convolutional neural networks, 2020, <http://dx.doi.org/10.48550/arXiv.1910.01279>, arXiv:1910.01279.
- [58] M.B. Muhammad, M. Yeasin, Eigen-CAM: Class activation map using principal components, in: 2020 International Joint Conference on Neural Networks (IJCNN), IEEE, Glasgow, United Kingdom, 2020, pp. 1–7, <http://dx.doi.org/10.1109/IJCNN48605.2020.9206626>.
- [59] B. Zhou, D. Bau, A. Oliva, A. Torralba, Interpreting deep visual representations via network dissection, 2018, <http://dx.doi.org/10.48550/arXiv.1711.05611>, arXiv:1711.05611.
- [60] M.A.A.K. Jalwana, N. Akhtar, M. Bennamoun, A. Mian, CAMERAS: Enhanced resolution and sanity preserving class activation mapping for image saliency, 2021, <http://dx.doi.org/10.48550/arXiv.2106.10649>, arXiv:2106.10649.
- [61] S. Sarkar, A.R. Babu, S. Mousavi, S. Ghorbanpour, V. Gundecha, A. Guillen, et al., RL-CAM: Visual explanations for convolutional networks using reinforcement learning, in: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, CVPRW, IEEE, Vancouver, BC, Canada, 2023, pp. 3861–3869, <http://dx.doi.org/10.1109/CVPRW59228.2023.00400>.
- [62] S. Greydanus, A. Koul, J. Dodge, A. Fern, Visualizing and understanding atari agents, 2018, <http://dx.doi.org/10.48550/arXiv.1711.00138>, arXiv:1711.00138.
- [63] H.-T. Joo, K.-J. Kim, Visualization of deep reinforcement learning using grad-CAM: How AI plays atari games? in: 2019 IEEE Conference on Games, CoG, IEEE, London, United Kingdom, 2019, pp. 1–2, <http://dx.doi.org/10.1109/CIG.2019.8847950>.
- [64] R.S. Sutton, A.G. Barto, *Reinforcement Learning: An Introduction, Second Edition*, Adaptive Computation and Machine Learning, MIT Press, Cambridge, Mass, 2018.
- [65] H. Dong, Z. Ding, S. Zhang (Eds.), *Deep Reinforcement Learning: Fundamentals, Research and Applications*, Springer Singapore Pte. Limited, 2020, Singapore, 2020.
- [66] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018, arXiv:1801.01290.
- [67] S. Dey, J.-N. Zaech, N. Nikolov, L.V. Gool, D.P. Paudel, ReVLA: Reverting visual domain limitation of robotic foundation models, 2024, <http://dx.doi.org/10.48550/arXiv.2409.15250>, arXiv:2409.15250.
- [68] X. Li, K. Hsu, J. Gu, K. Pertsch, O. Mees, H.R. Walke, et al., Evaluating real-world robot manipulation policies in simulation, 2024, <http://dx.doi.org/10.48550/arXiv.2405.05941>, arXiv:2405.05941.
- [69] X. Li, P. Li, M. Liu, D. Wang, J. Liu, B. Kang, et al., Towards generalist robot policies: What matters in building vision-language-action models, 2024, <http://dx.doi.org/10.48550/arXiv.2412.14058>, arXiv:2412.14058.
- [70] Bullet real-time physics simulation, 2025, <https://pybullet.org/wordpress/>. (Accessed 2 March 2025).
- [71] F. Pardo, A. Tavakoli, V. Levdiuk, P. Kormushev, Time limits in reinforcement learning, 2022, <http://dx.doi.org/10.48550/arXiv.1712.00378>, arXiv:1712.00378.
- [72] H.V. Hasselt, Double Q-learning, in: Proceedings of the 24th International Conference on Neural Information Processing Systems, 2010, <http://dx.doi.org/10.5555/2997046.2997187>.
- [73] S. Fujimoto, H. van Hoof, D. Meger, Addressing function approximation error in actor-critic methods, 2018, <http://dx.doi.org/10.48550/arXiv.1802.09477>, arXiv:1802.09477.
- [74] H. Walke, K. Black, A. Lee, M.J. Kim, M. Du, C. Zheng, et al., BridgeData V2: A dataset for robot learning at scale, 2024, arXiv:2308.12952.
- [75] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, B. Kim, *Sanity checks for saliency maps*, 2018.
- [76] M. Ahn, D. Dwibedi, C. Finn, M.G. Arenas, K. Gopalakrishnan, K. Hausman, et al., AutoRT: Embodied foundation models for large scale orchestration of robotic agents, 2024, <http://dx.doi.org/10.48550/arXiv.2401.12963>, arXiv:2401.12963.
- [77] T.T. Ponbagavathi, K. Peng, A. Roitberg, Probing fine-grained action understanding and cross-view generalization of foundation models, 2024, <http://dx.doi.org/10.48550/arXiv.2407.15605>, arXiv:2407.15605.
- [78] S. Sun, W. An, F. Tian, F. Nan, Q. Liu, J. Liu, et al., A review of multimodal explainable artificial intelligence: Past, present and future, 2024, <http://dx.doi.org/10.48550/arXiv.2412.14056>, arXiv:2412.14056.
- [79] S. Atakishiyeve, M. Salameh, H. Yao, R. Goebel, Explainable artificial intelligence for autonomous driving: A comprehensive overview and field guide for future research directions, IEEE Access 12 (2024) 101603–101625, <http://dx.doi.org/10.1109/ACCESS.2024.3431437>.
- [80] M. Müller, A. Dosovitskiy, B. Ghanem, V. Koltun, Driving policy transfer via modularity and abstraction, 2018, <http://dx.doi.org/10.48550/arXiv.1804.09364>, arXiv:1804.09364.
- [81] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32, <http://dx.doi.org/10.1023/A:1010933404324>.
- [82] S. Baker, W. Xiang, Explainable AI is responsible AI: How explainability creates trustworthy and socially responsible artificial intelligence, 2023, <http://dx.doi.org/10.48550/ARXIV.2312.01555>.